Τεχνικές Προβλεπτικής Συντήρησης με χρήση Μηχανικής Μάθησης

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ελισάβετ Καραπαλίδου

Επιβλέπων

Σταύρος Βολογιαννίδης, Επίκουρος Καθηγητής

ΣΕΡΡΕΣ - Αύγουστος 2023

Υπεύθυνη Δήλωση *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της μεταπτυχιακής διπλωματικής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στη διπλωματική εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης, βεβαιώνω ότι αυτή η μεταπτυχιακή διπλωματική εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του Προγράμματος Μεταπτυχιακών Σπουδών στην Εφαρμοσμένη Πληροφορική του Τμήματος Μηχανικών Πληροφορικής, Υπολογιστών και Τηλεπικοινωνιών του Διεθνούς Πανεπιστημίου της Ελλάδας.*

Περίληψη

Σκοπός της διπλωματικής εργασίας είναι η ανάπτυξη Αλγορίθμων Μηχανικής Μάθησης με τη χρήση της προγραμματιστικής γλώσσας Python, στα πλαίσια της Προβλεπτικής Συντήρησης των σύγχρονων βιομηχανιών.

Για την παραπάνω μελέτη χρησιμοποιήθηκαν δυο σύνολα δεδομένων με χαρακτηριστικές μετρήσεις ρουλεμάν οι οποίες συλλέχθηκαν από αισθητήρες κατά τη λειτουργία τους. Το πρώτο σύνολο δεδομένων βασίστηκε σε πειραματική μηχανική διάταξη κινητήρα, ενώ το δεύτερο συλλέχθηκε από πραγματικό βιομηχανικό εξοπλισμό υπό συνθήκες παραγωγικής διαδικασίας.

Και στα δυο σύνολα δεδομένων έγινε εφαρμογή κατάλληλης αρχιτεκτονικής Τεχνητών Νευρωνικών Δικτύων. Συγκεκριμένα, το πρώτο σύνολο δεδομένων αντιμετωπίστηκε ως πρόβλημα Κατηγοριοποίησης για την Ανίχνευση Βλαβών με ένα μοντέλο Πολυεπίπεδου Αντιλήπτρου, ενώ το δεύτερο ως πρόβλημα Ανίχνευσης Ανωμαλιών μέσω της μεθόδου Ανακατασκευής ενός μοντέλου Αυτοκωδικοποιητή με νευρώνες Μακράς Βραχυπρόθεσμης Μνήμης.

Ακόμη, στα πλαίσια εκπαιδευτικού χαρακτήρα, έγινε αναλυτική επεξήγηση όλων των στατιστικών μεθόδων και των τεχνικών Μηχανικής και Βαθιάς Μάθησης που εφαρμόστηκαν παραπάνω, τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο. Επίσης, έγινε αναφορά και παρουσιάστηκαν υλοποιήσεις ομαδοποίησης δεδομένων βάσει κλασσικών αλγορίθμων και τεχνικών.

Σημειώνεται ότι, οι δυο βασικές εφαρμογές των συνόλων δεδομένων και τα συμπεράσματά τους αποτέλεσαν υλικό δημοσιεύσεων. Ειδικότερα, η μελέτη που βασίστηκε στο πρώτο σύνολο δεδομένων παρουσιάστηκε στο 9ο Διεθνές Συνέδριο Ελέγχου, Αποφάσεων και Πληροφορίας Τεχνολογιών (9th International Conference on Control, Decision and Information Technologies), ενώ η μελέτη του δεύτερου συνόλου δεδομένων δημοσιεύτηκε στο περιοδικό Sensors του MDPI (Multidisciplinary Digital Publishing Institute).

Abstract

The purpose of the thesis is the development of Machine Learning Algorithms using the Python programming language, in the context of Predictive Maintenance of modern industries.

For the above study, two datasets were used containing characteristic measurements of bearings which were collected by sensors during their operation. The first dataset was based on an experimental mechanical engine setup, while the second was collected from real industrial equipment under actual production process conditions.

An appropriate Artificial Neural Network architecture was applied to both datasets. Specifically, the first dataset was treated as a Classification problem for Fault Detection with a Multilayer Perceptron model, while the second as an Anomaly Detection problem through the Reconstruction method of an Autoencoder model implementing Long Short-Term Memory neurons.

Furthermore, in the context of an educational nature, a detailed explanation of all the applied statistical methods and Machine and Deep Learning techniques was made, both at a theoretical and practical level. Implementations of data clustering based on traditional algorithms and techniques are also reported and presented.

It is noted that the two main applications of the datasets and their results were used as publishing material. In particular, the study of the first dataset was presented at the 9th International Conference on Control, Decision and Information Technologies, while the study of the second dataset was published in MDPI's Sensors magazine.

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

1 Ιστορική Αναδρομή Βιομηχανικών Διαδικασιών

Σύμφωνα με την επιστήμη της Οικονομίας, η Δευτερογενής Παραγωγή της Οικονομίας βασίζεται στην εκμετάλλευση των πρώτων υλών της Πρωτογενούς Παραγωγής από τις βιομηχανίες, για την κατασκευή και παραγωγή προϊόντων σε εργοστασιακές εγκαταστάσεις μέσω τεχνολογικών διαδικασιών [1]. Ωστόσο, η σύγχρονη βιομηχανοποίηση αποτελεί αποτέλεσμα πολλών ετών, κατά το πέρασμα των οποίων οι εν λόγω τεχνολογικές παραγωγικές διαδικασίες επιβλήθηκαν σε δραματικές αλλαγές, προφανώς συνοδευόμενες από κλιμακούμενο όφελος όσον αφορά στην παραγωγικότητα, αποδοτικότητα, ευελιξία και βιωσιμότητα των επιχειρήσεων, καθώς φυσικά και την χρηματοοικονομική τους θέση. Μάλιστα, αυτές οι εξελικτικές αλλαγές στον βιομηχανικό τομέα είχαν τόσο σημαντική επίδραση στην κοινωνία, την οικονομία και την επιστημονική πρόοδο, πέρα από τα πλεονεκτήματα στις εργασιακές συνθήκες, οι οποίες πλέον χαρακτηρίζονται ως «Βιομηχανικές Επαναστάσεις» [2].

1.1 Πρώτη Βιομηχανική Επανάσταση

Η Πρώτη Βιομηχανική Επανάσταση ξεκίνησε στα μέσα του 18^{ου} αιώνα στη Μεγάλη Βρετανία και αποτελεί ένα από τα πιο κρίσιμα σημεία στην ιστορία της ανθρωπότητας. Ο οικονομικός τομέας μετασχηματίστηκε ριζικά, καθώς έπαψε να έχει ως κύρια βάση τη γεωργία και πέρασε στην παραγωγή και κατασκευή προϊόντων μέσω μηχανοποιημένων διαδικασιών, οι οποίες μέχρι τότε στηρίζονταν στα ανθρώπινα χέρια, συστήνοντας για πρώτη φορά το εργοστασιακό σύστημα.

Η καθοριστική τεχνολογία αυτής της ιστορικής περιόδου ήταν η εφεύρεση της ατμομηχανής, η οποία τελικά αντικατέστησε τον τότε δημοφιλή υδάτινο τροχό, και οδήγησε σε μαζική εξόρυξη άνθρακα για την εκμετάλλευσή του ως καύσιμο σε ατμομηχανές, ατμόπλοια αλλά και σε μηχανές εργοστασίων, οριστικοποιώντας με αυτόν τον τρόπο και τη μετάβαση από το ξύλο στον άνθρακα ως κύρια πηγή ενέργειας. Επίσης,

πρωτοφανής στον τομέα της κλωστοϋφαντουργίας ήταν η αντικατάσταση των κλασσικών αργαλειών από μηχανοποιημένες εναλλακτικές, οδηγώντας ακόμα και σε οχταπλάσια αύξηση της παραγωγής. Επιπλέον, στον τομέα της μεταλλουργίας αναπτύχθηκε η παραγωγή σφυρήλατου σιδήρου μέσω πειραματικών τεχνικών εξόρυξης, ο οποίος τελικά χρησιμοποιήθηκε στην κατασκευή μηχανημάτων αλλά και σε διάφορες βαριές βιομηχανικές εφαρμογές λόγω της καταλληλότητας του κράματος, όσον αφορά στην ελατή φύση του.

Ταυτόχρονα, οι χερσαίες μεταφορές βελτιώθηκαν με την κατασκευή καλύτερων δρόμων, ενώ σκάφτηκαν κανάλια τόσο στην Ευρώπη όσο και στη Βόρεια Αμερική, αναγνωρίζοντας τη σημαντικότητα των ατμόπλοιων στις θαλάσσιες μεταφορές, αλλά και την ανάγκη αποτελεσματικότερων τρόπων διάθεσης της παραπάνω αυξημένης παραγωγής αγαθών στην αγορά, η οποία μέχρι τότε στηριζόταν σε κάρα και υποζύγια. Τελικά, η εγκατάσταση σιδηροδρομικού δικτύου σε Ευρώπη, Βόρεια Αμερική και Ασία συντέλεσε κι αυτή με τη σειρά της στην ανάπτυξη του τομέα της οικονομίας, αλλά και της βιομηχανίας, αστικοποιώντας περαιτέρω τις αγροτικές κοινωνίες με τις δραματικές αλλαγές που προκάλεσε η εμφάνιση των ατμομηχανών στις χερσαίες μεταφορές [3].

1.2 Δεύτερη Βιομηχανική Επανάσταση

Η Δεύτερη Βιομηχανική Επανάσταση, γνωστή και ως «Τεχνολογική Επανάσταση», ξεκίνησε στα τέλη του 19^{ου} αιώνα με τη συμβολή νέων μηχανημάτων, νέων πηγών ενέργειας και νέων μεθόδων εργασιακής οργάνωσης και παραγωγικής διαδικασίας, αποτελώντας αιτία για να δημιουργηθούν νέες βιομηχανίες, αλλά και να εξελιχθούν οι ήδη υπάρχουσες, αναπτύσσοντας την παραγωγή και την αποδοτικότητα τους [4]. Η συγκεκριμένη χρονική περίοδος είναι θεμελιωμένη στην ανακάλυψη του ηλεκτρισμού, αλλά και στην εκμετάλλευση του φυσικού αερίου και πετρελαίου.

Η ηλεκτρική ενέργεια αποτέλεσε αφορμή για επιστημονική πρόοδο, αλλά και για την εφεύρεση πολλών νέων τεχνολογικών συστημάτων, και γενικότερα άλλαξε ριζικά τη ζωή των ανθρώπων και τη μεταξύ τους επικοινωνία. Συγκεκριμένα, η εμφάνιση του τηλέγραφου, του ραδιοφώνου και του τηλεφώνου εκμηδένισε τις επικοινωνιακές αποστάσεις επιτρέποντας την άμεση ανταλλαγή ιδεών, ενώ ο ηλεκτρικός φωτισμός άλλαξε καθοριστικά την ποιότητα ζωής, οδηγώντας μεγάλο πλήθος ανθρώπων να μετακομίσει στα αστικά κέντρα. Επιπλέον, ταυτόχρονα με την συνεχιζόμενη επέκταση του σιδηροδρομικού δικτύου, οι νέες πηγές ενέργειας συνέβαλαν στην εφεύρεση μηχανών εσωτερικής καύσης,

και κατ' επέκταση στην εμφάνιση των πρώτων αυτοκινήτων και αεροπλάνων. Άλλες καινοτομίες παρατηρήθηκαν στην παραγωγή χημικών προϊόντων και την εκμετάλλευση φυσικών και συνθετικών πόρων, ενώ υδρευτικά και αποχετευτικά συστήματα άρχισαν να εγκαθίστανται σε μεγάλες πόλεις.

Ειδικότερα στον τομέα της βιομηχανίας, ο Χένρυ Φορντ, ιδρυτής της ομώνυμης αυτοκινητοβιομηχανίας, εκμεταλλευόμενος τις δυνατότητες του ηλεκτρισμού, προχώρησε σε αναδιάταξη μηχανών, εξοπλισμού και εργαζομένων προκειμένου να υπάρχει συνεχής ροή εξαρτημάτων στην παραγωγική διαδικασία, δημιουργώντας έτσι την πρώτη γραμμή συναρμολόγησης, η οποία χρησιμοποιείται πλέον παγκοσμίως στα εργοστάσια [5]. Η γραμμή συναρμολόγησης οδήγησε σε γρηγορότερη και φθηνότερη μαζική παραγωγή προϊόντων και επέτρεψε στη μεσαία τάξη την πρόσβαση στις καινοτόμες τεχνολογίες της περιόδου.

1.3 Τρίτη Βιομηχανική Επανάσταση

Η Τρίτη Βιομηχανική Επανάσταση, ή αλλιώς «Ψηφιακή Επανάσταση», ξεκίνησε μετά το τέλος του 2^{ου} Παγκοσμίου Πολέμου, στο δεύτερο μισό του 20^{ου} αιώνα, και χαρακτηρίζεται από την άνοδο των ηλεκτρονικών συσκευών και τη μετάβαση από τη μηχανολογική και αναλογική τεχνολογία στην ψηφιακή. Αξίζει να σημειωθεί ότι, ενώ κατά την 1^{η} και 2^{η} Βιομηχανική Επανάσταση μόνο η Δυτική Ευρώπη και οι Ηνωμένες Πολιτείες της Αμερικής επωφελήθηκαν από την εκάστοτε τεχνολογική πρόοδο, κατά την διάρκεια αυτής της χρονικής περιόδου η ανθρωπότητα γίνεται δέκα φορές πλουσιότερη και η τεχνολογική ανάπτυξη εξαπλώνεται σχεδόν σε κάθε μέρος του κόσμου [6].

Σημαντική εφεύρεση της εποχής υπήρξαν οι ημιαγωγοί, και κατ' επέκταση τα τρανζίστορ, τα ολοκληρωμένα κυκλώματα και οι μικροεπεξεργαστές, που με τη σειρά τους αποτέλεσαν, και συνεχίσουν να αποτελούν, δομικά στοιχεία των ηλεκτρονικών συσκευών. Επίσης, εμφανίστηκαν τα πρώτα μεγάλα συστήματα υπολογιστών, και τελικά έγιναν ευρέως διαθέσιμοι στην αγορά οι προσωπικοί υπολογιστές, αλλά και οι σύγχρονες τηλεπικοινωνιακές τεχνολογίες. Παράλληλα, η ανάπτυξη υπολογιστικών προγραμμάτων σε συνδυασμό με την ηλεκτρολογική πρόοδο οδήγησαν στην εφεύρεση των Προγραμματιζόμενων Λογικών Ελεγκτών (Programmable Logic Controllers – PLC) και στην ανάπτυξη του τομέα της Ρομποτικής, που έπαιξαν καταλυτικό ρόλο στην άνοδο του αυτοματισμού στις βιομηχανικές διαδικασίες. Τελικά, οι βιομηχανίες, εκμεταλλευόμενες τις νέες τεχνολογίες και ενσωματώνοντάς τες στην παραγωγική διαδικασία, καταφέρανε να

αυτοματοποιήσουν πλήρως την παραγωγή προϊόντων και αγαθών ελαχιστοποιώντας περαιτέρω τον ανθρώπινο παράγοντα.

Επίσης, από τις σημαντικότερες καινοτομίες του τομέα της Τεχνολογίας Πληροφοριών (Information Technology – IT) ήταν η δημιουργία του Διαδικτύου (Internet), η οποία άλλαξε, και συνεχίζει να αλλάζει ριζικά, την ανθρώπινη καθημερινότητα. Το Διαδίκτυο, οι τηλεπικοινωνίες και οι εφαρμογές τους, όπως ο Παγκόσμιος Ιστός (World Wide Web), το ηλεκτρονικό ταχυδρομείο (e-mail), ο διαμοιρασμός αρχείων και η απομακρυσμένη αποθήκευση, τα κοινωνικά δίκτυα, το ηλεκτρονικό εμπόριο κ.α., επέτρεψαν στις επιχειρήσεις να προοδεύσουν ακόμη περισσότερο στις παραγωγικές διαδικασίες και υπηρεσίες τους.

1.4 Τέταρτη Βιομηχανική Επανάσταση

Η τεχνολογία έχει κάνει μεγάλα βήματα από τις πρώτες ατμομηχανές και τους μηχανικούς αργαλειούς, την αξιοποίηση της ηλεκτρικής ενέργειας και την ανάπτυξη των πρώτων γραμμών συναρμολόγησης για μαζική παραγωγή στα εργοστάσια, και από την δημιουργία των υπολογιστών και του Διαδικτύου. Κάθε επαναστατική πρόοδος προκάλεσε ριζικές αναδιαρθρώσεις στον οικονομικό, κοινωνικό, πολιτικό και πολιτισμικό τομέα, και έπαιξε καθοριστικό ρόλο στη θεμελίωση της σύγχρονης κοινωνίας [7].

Αυτήν τη στιγμή βρισκόμαστε στις αρχές του 21^{ου} αιώνα και στην έλευση της 4^{η} Βιομηχανικής Επανάστασης, της οποίας προάγγελος αποτέλεσε η ενέργεια ψηφιοποίησης της προηγούμενης εποχής, καθώς και οι τεχνολογικές της εξελίξεις, ενώ πλέον καθοδηγείται σε μεγάλο βαθμό από την αρμονική σύγκλιση των ψηφιακών, βιολογικών και φυσικών καινοτομιών [8]. Πρόκειται για μια επανάσταση η οποία συμβαίνει αυτήν τη στιγμή, τη βιώνουμε καθημερινά και είναι άγνωστο το πόσο θα εξελιχθεί και πως θα επηρεάσει περαιτέρω την ανθρωπότητα.

Παρακάτω παρουσιάζονται κάποια από τα πιο διαδεδομένα πεδία όπου παρατηρούνται καινοτομίες της 4^{ης} Βιομηχανικής Επανάστασης, αλλά και ήδη υπάρχουσες τεχνολογίες που πλέον γνωρίζουν σημαντική εξέλιξη, και η σημασία πολλών είναι ήδη αναγνωρίσιμη από την ενσωμάτωσή τους σε πολλές πτυχές της σύγχρονης εποχής [9]:

- *Προσθετική Κατασκευή (Additive Manufacturing)* – Κατασκευαστική τεχνολογία που δημιουργεί τρισδιάστατες εκτυπώσεις μέσω αλλεπάλληλων στρώσεων υλικού.

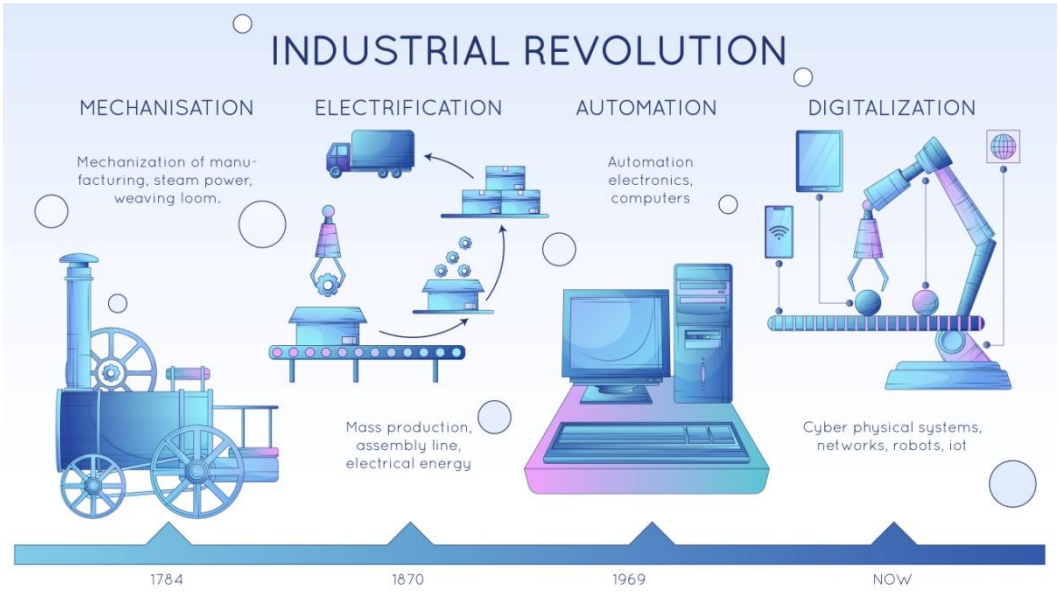
- *Τεχνητή Νοημοσύνη (Artificial Intelligence - AI)* – Τομέας της επιστήμης υπολογιστών η οποία χαρακτηρίζεται από την ανάπτυξη έξυπνων συστημάτων που λειτουργούν και αντιδρούν σαν άνθρωποι.
- *Επαυξημένη Πραγματικότητα (Augmented Reality - AR)* – Διαδραστικό, εικονικό περιβάλλον το οποίο δημιουργείται από την πραγματική ανθρώπινη εμπειρία και από την ενσωμάτωση παραγόμενης ψηφιακής πληροφορίας υπό μορφή εικόνας, βίντεο, ήχου ή άλλων μέσων που βασίζονται σε τεχνολογίες ολογραμμάτων και επικαλύπτουν την πραγματική εικόνα, βελτιώνοντάς την.
- *Μεγάλα Δεδομένα (Big Data)* – Συλλογή και στρατηγική ανάλυση μεγάλου όγκου δεδομένων, όπου οι συμβατικές τεχνικές εξόρυξης και ανάλυσης δεδομένων δεν επαρκούν για να αναγνωρίσουν τη θεμελιώδη σημασία τους.
- *Αλυσίδα Συστοιχιών (Blockchain)* – Κατανεμημένη βάση δεδομένων η οποία είναι διαμοιρασμένη μεταξύ ενός δικτύου υπολογιστών και χρησιμοποιεί νέες τεχνολογίες κρυπτογράφησης, αυθεντικοποίησης και μηχανισμούς συναίνεσης.
- *Σύννεφο (Cloud)* – Κάθε παρεχόμενη υπηρεσία πληροφορικής από απομακρυσμένα υπολογιστικά συστήματα που είναι προσβάσιμη μέσω του Διαδικτύου.
- *Κυβερνοασφάλεια (Cybersecurity)* – Πολιτικές που αποσκοπούν στην εδραίωση μεθόδων ασφαλείας της πληροφορίας από απόπειρες κλοπής, έκθεσης σε τρίτους ή επίθεσης.
- *Νανοτεχνολογία (Nanotechnology)* – Τεχνολογικός κλάδος που αξιοποιεί μεμονωμένα άτομα και μόρια για την κατασκευή προϊόντων μακροκλίμακας.
- *Νευροτεχνολογία (Neurotechnology)* – Τεχνολογικός κλάδος που μελετάει τον ανθρώπινο εγκέφαλο και το νευρικό σύστημα εξ' ολοκλήρου.
- *Προσομοίωση (Simulation)* – Τεχνολογίες που χρησιμοποιούν τα υπολογιστικά συστήματα για τη μίμηση μιας πραγματική κατάστασης.
- *Ταυτοποίηση μέσω Ραδιοσυχνοτήτων (Radio Frequency Identification - RFID)* – Τεχνολογίες που χρησιμοποιούν ασύρματη επικοινωνία μεταξύ πομπού και δέκτη για την αυτόματη παρακολούθηση και αναγνώριση αντικειμένων.
- *Αισθητήρες (Sensors)* – Συσκευές που ανταποκρίνονται σε κάποιο φυσικό ερέθισμα, όπως θερμότητα, φώς, υγρασία, δόνηση, πίεση ή σε κάποια άλλη συγκεκριμένη συνθήκη, και μεταδίδουν μετρήσεις ή λειτουργίες ελέγχου μέσω ηλεκτρικών παλμών.

- *Βιομηχανικό Διαδίκτυο των Πραγμάτων (Industrial Internet of Things - IIoT)* – Η συνεργασία των διαφόρων ηλεκτρονικών εξαρτημάτων μέσω της διαδικτυακής συνδεσιμότητας που αποσκοπεί στην βελτίωση της παραγωγικής διαδικασίας.
- *Παγκόσμιο Σύστημα Τοποθεσίας (Global Positioning System - GPS)* – Ομάδα δορυφόρων σε τροχιές γύρω από την Γη μεταδίδουν σήματα επιτρέποντας σε κατάλληλους πομπούς GPS τον ακριβή υπολογισμό θέσης, ταχύτητας και ώρας πάνω στον πλανήτη.
- *Κινητή Τηλεφωνία (Mobile Technology)* – Η ενσωμάτωση της ασύρματης επικοινωνιακής τεχνολογίας στις ασύρματες συσκευές.
- *Κομποτικά Συστήματα (Cobotic Systems)* – Νέα εφαρμογή βιομηχανικού αυτοματισμού που πρόκειται για αυτόνομα ρομπότ τα οποία συνεργάζονται με το ανθρώπινο δυναμικό μέσα από τη μεταξύ τους αλληλεπίδραση σε έναν κοινό χώρο εργασίας.

Η Βιομηχανία 4.0, όπως συνηθίζεται να ονομάζεται, χαρακτηρίζεται γενικότερα από έξυπνες και αυτόνομες ρομποτικές εργοστασιακές διαδικασίες, και από την εφαρμογή τεχνολογιών πληροφορίας και επικοινωνιών, ιδιαίτερα στις βιομηχανίες, αλλά και ως έναν βαθμό στην υπόλοιπη κοινωνία. Συγκεκριμένα για τον βιομηχανικό τομέα και την παραγωγική διαδικασία, τα ήδη ψηφιοποιημένα συστήματα παραγωγής επεκτείνονται και συνδέονται μέσω του Διαδικτύου, δημιουργώντας ένα δίκτυο επικοινωνίας μεταξύ διαφόρων απομακρυσμένων εγκαταστάσεων, και παράγοντας πληροφορίες για αυτές. Αυτή η διασύνδεση οδηγεί στα Κυβερνοφυσικά Συστήματα Παραγωγής (Cyberphysical Production Systems) και τελικά στα «έξυπνα» εργοστάσια, όπου τα λογισμικά συστήματα, ο μηχανικός εξοπλισμός και το ανθρώπινο δυναμικό επικοινωνούν μέσω του παραπάνω δικτύου, και τελικά ολόκληρη η διαδικασία παραγωγής είναι σχεδόν εντελώς αυτοματοποιημένη [10].

Οι τεχνολογίες της Βιομηχανίας 4.0, τόσο στην υλική, όσο και στην ψηφιακή τους διάσταση, έχουν μετασχηματίσει την παραγωγική διαδικασία και έχουν οδηγήσει σε νέα επιχειρηματικά μοντέλα και πολιτικές, υποστηρίζοντας περαιτέρω την ευελιξία, την αποδοτικότητα και την παραγωγικότητα των βιομηχανιών. Επιπλέον, παρά το υψηλότερο επίπεδο πολυπλοκότητας που παρουσιάζεται για την ενσωμάτωση αυτών των νέων τεχνολογιών στην παραγωγική διαδικασία, είναι αναμφίβολη η συνεισφορά τους στη βελτίωση της κοινωνικής, οικονομικής και περιβαλλοντολογικής βιωσιμότητας των βιομηχανιών σε ένα ανταγωνιστικό περιβάλλον, όπου είναι απαραίτητες τόσο

βραχυπρόθεσμες, όσο και μακροπρόθεσμες στρατηγικές και τακτικές επιχειρησιακής βιωσιμότητας.



Εικόνα 1. Εξέλιξη Βιομηχανικών Διαδικασιών κατά τις τέσσερις Βιομηχανικές Επαναστάσεις [11]

2 Πολιτικές Συντήρησης

Σύμφωνα με το ευρωπαϊκό πρότυπο UNE-EN 13306:2018, ως συντήρηση ορίζεται κάθε συνδυασμός τεχνικών, διοικητικών και διαχειριστικών ενεργειών που λαμβάνονται κατά τη διάρκεια ζωής ενός αντικειμένου, με σκοπό τη διατήρηση ή την επαναφορά του σε μια κατάσταση κατά την οποία μπορεί να εκτελέσει μια απαιτούμενη λειτουργία [12].

Στον τομέα της βιομηχανίας, οι επιχειρήσεις προσπαθούν συνεχώς να εδραιώσουν τη θέση τους σε ένα αναμφίβολα ανταγωνιστικό περιβάλλον, υπακούοντας όσο το δυνατόν καλύτερα στις απαιτήσεις της σύγχρονης αγοράς. Κατά συνέπεια, μια από τις κύριες ενέργειες για την επίτευξη του παραπάνω στόχου είναι η μείωση του κόστους μέσω μιας πιο οικονομικά αποδοτικότερης διαδικασίας παραγωγής. Αυτή η μετρίαση του κόστους παραγωγής γίνεται εφικτή εξασφαλίζοντας την υγεία και την αποτελεσματική επίδοση των κατασκευαστικών μηχανημάτων, του βιομηχανικού εξοπλισμού και των εξαρτημάτων μέσω πολιτικών συντήρησης. Η ανάγκη για στρατηγικές συντήρησης είναι προφανής, από τη στιγμή που ελαττωματικά μηχανικά μέρη είναι ικανά να προκαλέσουν σημαντικές βλάβες και σφάλματα με καταστροφική επίπτωση στην παραγωγική διαδικασία, καθώς και σοβαρό οικονομικό αντίκτυπο στις επιχειρήσεις, αν δεν αναγνωριστούν και διευθετηθούν σε σύντομο χρονικό διάστημα.

Τα κόστη συντήρησης αποτελούν πλέον αναπόσπαστο μέρος του συνολικού λειτουργικού κόστους κάθε βιομηχανίας και αντιπροσωπεύουν μια βραχυπρόθεσμη επένδυση με μακροπρόθεσμο οικονομικό και παραγωγικό εργοστασιακό όφελος. Για αυτόν τον λόγο, τα διοικητικά στελέχη στοχεύουν στην επιλογή κατάλληλης πολιτικής για να αποφευχθούν τυχόν περιττές ή λανθασμένες ενέργειες συντήρησης που μπορεί να οδηγήσουν σε μειωμένη ποιότητα προϊόντων, αναξιοπιστία επιχείρησης και οικονομική ζημία [13].

Από την εμφάνιση των βιομηχανιών, και με το πέρασμα κάθε επαναστατικής περιόδου και αναδιοργάνωσης, οι τεχνικές συντήρησης έχουν υποστεί σταδιακή εξέλιξη και ανανεώνονται συνεχώς. Πλέον οι βασικές πολιτικές συντήρησης μπορούν να κατηγοριοποιηθούν ως εξής [14]:

- Διορθωτική Συντήρηση
- Προληπτική Συντήρηση
- Συντήρηση βάσει Συνθηκών
- Προβλεπτική Συντήρηση

2.1 Διορθωτική Συντήρηση

Η Διορθωτική Συντήρηση (Corrective Maintenance) είναι γνωστή και ως «αντιδραστική» ή «απρογραμμάτιστη» ή «λειτουργία μέχρι την αποτυχία» (Run-to-Failure). Πρόκειται για μια στρατηγική η οποία, παρά το γεγονός ότι επιλέγεται στα πλαίσια συντήρησης μηχανημάτων, δεν στηρίζεται καθόλου σε ενέργειες συντήρησης, καθώς κύριο χαρακτηριστικό της είναι ότι βασίζεται στην επιδιόρθωση εξοπλισμού μόνο όταν αυτός χαλάσει.

Πρόκειται για μια υψηλού κινδύνου πολιτική λόγω των απρόοπτων περιστάσεων όσον αφορά στην έκταση της εκάστοτε βλάβης που θα παρουσιαστεί, στον χρόνο που θα χρειαστεί για να αντικατασταθεί κάποιο εξάρτημα ή ακόμη κι ολόκληρα μηχανήματα, στο κόστος που θα προκύψει από τη συντήρηση και στον χρόνο που πιθανότατα θα πρέπει να διακοπεί η λειτουργία της εγκατάστασης, παρεμποδίζοντας έτσι και την παραγωγική διαδικασία. Σχεδόν σε όλες τις περιπτώσεις όπου εφαρμόζεται η διορθωτική πολιτική, τα εργοστάσια ακολουθούν τυπικές προληπτικές ενέργειες, όπως λίπανση ή μικρές αναπροσαρμογές του μηχανολογικού εξοπλισμού, αλλά δεν γίνεται καμία κίνηση αποκατάστασης, ανακατασκευής ή επιδιόρθωσης αν πρώτα δεν παρουσιαστεί αποτυχία κατά την λειτουργία.

Επιπλέον, υπάρχει η προσδοκία από το τμήμα συντήρησης να είναι συνεχώς σε ετοιμότητα για να αντιδράσει σε άμεσο χρονικό διάστημα σε οποιαδήποτε βλάβη που μπορεί να προκύψει. Αυτό προϋποθέτει να υπάρχουν ανταλλακτικά και μηχανές ήδη αποθηκευμένα στον εργοστασιακό χώρο, ή τουλάχιστον τα κυριότερα εξαρτήματα για τις πιο σημαντικές μηχανές. Εναλλακτική λύση αποτελούν εξωτερικοί προμηθευτές οι οποίοι μπορούν να παρέχουν ταχεία παράδοση όλων των απαιτούμενων ανταλλακτικών. Η κύρια συσσώρευση κόστους αυτής της πολιτικής σχετίζεται με το υψηλό κόστος των ανταλλακτικών που πρέπει να βρίσκεται αποθηκευμένο, με το υψηλό κόστος που προκύπτει από την ανάγκη υπερωριών εργασίας, με τον χρόνο που θα αναγκαστεί να διακοπεί η λειτουργία του μηχανήματος και με τη μη-διαθεσιμότητα της παραγωγής προϊόντων λόγω της διακοπής αυτής. Παράλληλα, υπάρχει κίνδυνος δευτερευόντων σφαλμάτων σε εξαρτήματα που σχετίζονται με την αρχική βλάβη, τα οποία μπορεί να οδηγήσουν σε μεγάλο πλήθος ελαττωματικών προϊόντων μέχρι να αναγνωριστούν, ή και σε νέα ανάγκη συντήρησης σε σχετικά κοντινό χρονικό διάστημα, όταν τελικά τα συγκεκριμένα εξαρτήματα αποτύχουν λειτουργικά.

Αυτή η φιλοσοφία ακολουθείται από την εμφάνιση των πρώτων εργοστασιακών εγκαταστάσεων και πλέον εφαρμόζεται κυρίως από βιομηχανικές μονάδες μικρότερης

κλίμακας, οι οποίες συνήθως είναι υποστελεχωμένες, ενώ δεν υπάρχει ούτε εξειδικευμένο προσωπικό για περιπτώσεις συντήρησης. Ο βιομηχανικός τομέας αγωνίζεται χρόνια να εγκαταλείψει αυτήν την πολιτική, καθώς είναι δαπανηρή και συχνά οδηγεί σε απροσδόκητα σοβαρές διακοπές λειτουργίας και αυξημένα κόστη [15].

2.2 Προληπτική Συντήρηση

Η Προληπτική Συντήρηση (Preventive Maintenance) είναι στρατηγική που βασίζεται στον προγραμματισμό εργασιών συντήρησης, όπως επιθεωρήσεις μηχανημάτων, επισκευές, ανακατασκευές ή προσαρμογές εξοπλισμού. Σύμφωνα με ένα χρονοδιάγραμμα επιδιώκεται η πρόληψη όσο το δυνατόν περισσότερων επικείμενων σφαλμάτων κατά τη λειτουργία των μηχανημάτων, αλλά και η ελαχιστοποίηση της διάρκειας μιας αναγκαίας διακοπής της λειτουργίας τους, και κατ’ επέκταση της παραγωγικής διαδικασίας.

Οι ενέργειες συντήρησης εκτελούνται περιοδικά, βάσει χρόνου ή ωρών λειτουργίας του εξοπλισμού που έχει παρέλθει, και η πραγματική υλοποίηση της Προληπτικής Συντήρησης ποικίλει ανάλογα με την περίπτωση. Κάποιες βιομηχανίες μπορεί να ακολουθούν πολύ περιορισμένη συντήρηση που να αποτελείται μόνο από λίπανση και πολύ μικρές διορθώσεις, ενώ άλλες μπορεί να προγραμματίζουν και διορθώσεις, αντικαταστάσεις ή ανακατασκευές, ειδικά σε μηχανήματα υψηλής σημασίας. Ο κοινός παρονομαστής σε κάθε περίπτωση είναι ότι ακολουθείται συγκεκριμένο χρονοδιάγραμμα, με το στοιχείο του χρόνου να παίζει τον κυριότερο ρόλο.

Ειδικότερα, τα χρονοδιαγράμματα και ο προγραμματισμός των εργασιών στηρίζονται στη βοήθεια στατιστικών αναλύσεων οι οποίες βασίζονται σε μετρικές συμβάντων, όπως ο «μέσος χρόνος μεταξύ αποτυχιών» (mean-time-between-failures – MTBF), ο «μέσος χρόνος μέχρι την αποτυχία» (mean-time-to-failure – MTTF) και ο «μέσος χρόνος μέχρι την επιδιόρθωση» (mean-time-to-repair – MTTR). Η ανάλυση των παραπάνω χρόνων, σε συνδυασμό φυσικά και με άλλους παράγοντες, όπως οι συστάσεις των κατασκευαστών των μηχανών και των εξαρτημάτων, η τιμή τους, η χρήση για την οποία προορίζονται κ.α., συμβάλλουν στην εφαρμογή ενός προγράμματος συντήρησης το οποίο έχει σκοπό την πρόληψη βλαβών, μειώνοντας έτσι τον κίνδυνο σφαλμάτων, καθώς και το πλήθος και τη διάρκεια απρογραμμάτιστων διακοπών της παραγωγής, εξασφαλίζοντας μια αξιόπιστη και λειτουργική διαδικασία, καθώς και την καλή υγεία του εξοπλισμού [16].

Παρά το γεγονός ότι πρόκειται για μια στρατηγική με μεγαλύτερη πιθανότητα μετριασμού της αποτυχίας του εξοπλισμού, στην προσπάθειά της να καταφέρει την επίτευξη μιας

αποδοτικότερης και πιο αξιόπιστης παραγωγικής διαδικασίας, ενίοτε οδηγεί σε περιττές ενέργειες συντήρησης, αλλά και στην πρόωρη απομάκρυνση και αντικατάσταση εξαρτημάτων με υποσχόμενο υπόλοιπο ωφέλιμης λειτουργίας, αυξάνοντας άσκοπα το συνολικό κόστος παραγωγής, ενώ δεν μπορεί να αποτρέψει τυχαίες αστοχίες, οι οποίες είναι μέρος κάθε συμβατικής παραγωγικής διαδικασίας.

2.3 Συντήρηση βάσει Συνθηκών

Η Συντήρηση βάσει Συνθηκών (Condition-based Maintenance) είναι στρατηγική η οποία βασίζεται στη συνεχή παρακολούθηση της υγείας του μηχανολογικού εξοπλισμού, και η συντήρησή του πραγματοποιείται μόνο όταν κρίνεται απολύτως απαραίτητη. Πρόκειται για μια περίπτωση όπου δεν υπάρχει δυνατότητα προγραμματισμού των εργασιών συντήρησης, καθώς βασίζεται στην επίβλεψη των μηχανημάτων για οποιαδήποτε υποβάθμιση προκύψει κατά την παραγωγική διαδικασία.

Η επανάσταση της ψηφιοποίησης, παράλληλα με την τεχνολογική και επιστημονική πρόοδο, επέτρεψαν την ανάπτυξη εργαλείων και μεθόδων ικανών να λαμβάνουν κάθε αναγκαία πληροφορία σχετικά με τη λειτουργική κατάσταση και υγεία του εξοπλισμού. Συγκεκριμένα, μετρήσεις μπορούν να ληφθούν μέσω ανάλυσης κραδασμών, θερμογραφίας, υπερήχων, τριβολογίας, υπερύθρων, τηλεμετρίας κ.α. Η συνεχής παρακολούθηση και ανάλυση των μετρήσεων, σε συνδυασμό με συχνές επιθεωρήσεις των μηχανημάτων από έμπειρους συμβούλους και μηχανικούς, μπορούν να επιτρέψουν την έγκαιρη αναγνώριση σφαλμάτων από τα καταγεγραμμένα δεδομένα, ή τη μη-ομαλή συμπεριφορά στον εξοπλισμό, και να οδηγήσουν άμεσα σε διαδικασίες συντήρησης όπου κρίνεται απαραίτητο.

Στη σύγχρονη εποχή, με την άνοδο των υπολογιστικών συστημάτων, ο διαμοιρασμός αυτής της πληροφορίας επιτυγχάνεται με την εγκατάσταση συστημάτων ελέγχου ικανών να παρακολουθούν, με την πάροδο του χρόνου, την υγεία του εξοπλισμού μέσα από ένα σύνολο αντιπροσωπευτικών δεικτών. Η κύρια λειτουργία τους είναι να λαμβάνουν ψηφιακά δεδομένα από ειδικούς αισθητήρες που έχουν τοποθετηθεί στα μηχανήματα και να τα συγκρίνουν με τις προσδοκώμενες τιμές, παράγοντας ειδοποιήσεις μόλις ανιχνευτεί κάποια υπέρβαση των προκαθορισμένων ορίων. Όταν κάποιος, ή και περισσότεροι, από τους παραπάνω δείκτες επιβεβαιώσουν κάποια φθορά στον εξοπλισμό, τότε λαμβάνονται τα κατάλληλα μέτρα συντήρησης για την επαναφορά του σε μια επιθυμητή κατάσταση. Επίσης, τα παραπάνω προκαθορισμένα όρια των δεικτών μπορούν να ρυθμιστούν

κατάλληλα έτσι ώστε να είναι πιο αντιπροσωπευτικά στην εργασία που καλείται να εκτελέσει κάθε μηχάνημα, βελτιστοποιώντας έτσι και τη συνολική δραστηριότητά της βιομηχανίας. Οποιαδήποτε ενέργεια επιδιόρθωσης, αντικατάστασης, ακόμα και απόσυρσης του εξοπλισμού εκτός λειτουργίας, βασίζεται σε αναμφίβολες ενδείξεις επιδείνωσης της κατάστασής του.

Πρόκειται για μια στρατηγική η οποία δεν ακολουθεί κάποιο συγκεκριμένο χρονοδιάγραμμα που καθορίζεται βάσει βιομηχανικών στατιστικών, αλλά στηρίζεται, εκτός από τους επί τόπου ελέγχους και επιθεωρήσεις έμπειρων μηχανικών, στα πραγματικά δεδομένα των ίδιων των μηχανημάτων που βρίσκονται στις εργοστασιακές μονάδες. Επιπλέον, υψίστης σημασίας είναι και ο διαμοιρασμός της κατάλληλης πληροφορίας, την κατάλληλη στιγμή, στο κατάλληλο άτομο για τη λήψη της καταλληλότερης απόφασης, με σκοπό την έγκαιρη διάγνωση μηχανικών προβλημάτων για την αποτροπή καταστροφικών σφαλμάτων, αλλά και για τον κατάλληλο προγραμματισμό συντήρησης με το συντομότερο δυνατό χρόνο διακοπής της λειτουργίας.

Τελικά, μια κατάλληλη υποδομή, στα πλαίσια μιας πολιτικής Συντήρησης βάσει Συνθηκών, τόσο σε τεχνολογικό και πληροφοριακό επίπεδο, όσο και σε επίπεδο εξειδικευμένου ανθρώπινου δυναμικού, μπορεί να βελτιώσει την παραγωγικότητα και αποτελεσματικότητα των εργοστασιακών μονάδων, οδηγώντας κατά συνέπεια στη καλύτερη ποιότητα προϊόντων και στη μεγιστοποίηση του δυνατού χρόνου λειτουργίας του εξοπλισμού [17].

2.4 Προβλεπτική Συντήρηση

Οι τρεις πολιτικές συντήρησης οι οποίες αναφέρθηκαν μέχρι τώρα διαφοροποιούνται κυρίως, πέρα από τα μέσα και τα εργαλεία που χρησιμοποιούν για τη διεξαγωγή τους, στο πότε τελικά θα πραγματοποιηθούν οι ενέργειες συντήρησης. Στη Διορθωτική Συντήρηση η επιδιόρθωση του εξοπλισμού γίνεται εντελώς απρογραμμάτιστα, μόνο όταν αυτός χαλάσει, στην Προληπτική Συντήρηση η λειτουργία των μηχανημάτων εξασφαλίζεται από την αυστηρή τήρηση ακριβούς χρονοδιαγράμματος, ενώ η Συντήρηση βάσει Συνθηκών γίνεται κυρίως απρογραμμάτιστα, βασιζόμενη σε ενδείξεις που λαμβάνονται εκείνη τη στιγμή.

Η Προβλεπτική Συντήρηση (Predictive Maintenance) στηρίζεται και αυτή στη συνεχή παρακολούθηση του εξοπλισμού, και μάλιστα, πριν λίγα χρόνια ο όρος αναφερόταν για να περιγράψει τις πιο καινοτόμες τακτικές που εφαρμόζονταν στην περίπτωση της Συντήρησης βάσει Συνθηκών. Αυτό όμως που διαχωρίζει την Προβλεπτική Συντήρηση από τις υπόλοιπες πολιτικές, και ο λόγος που είναι πλέον ευρέως αναγνωρίσιμη, είναι το γεγονός ότι αξιοποιεί

με τον πιο αποτελεσματικό τρόπο τις νεότερες και πιο διαδεδομένες τεχνολογίες της Βιομηχανίας 4.0, συγκεκριμένα, το Βιομηχανικό Διαδίκτυο των Πραγμάτων (IIoT), εξειδικευμένους αισθητήρες, το Σύννεφο (Cloud), την ανάλυση Μεγάλων Δεδομένων (Big Data Analytics) και το κυριότερο, την Τεχνητή Νοημοσύνη [18].

Μια στρατηγική Προβλεπτικής Συντήρησης βασίζεται στη συστηματική ανάλυση ιστορικών δεδομένων και στην αξιοποίηση εργαλείων πρόβλεψης, με σκοπό την αναγνώριση εν δυνάμει σφαλμάτων του εξοπλισμού, πριν αυτά παρουσιαστούν στην παραγωγική διαδικασία, ή τουλάχιστον την ανίχνευσή τους σε ένα πρώιμο στάδιο. Συνεπώς, οι ενέργειες συντήρησης στην περίπτωση αυτής της στρατηγικής δεν είναι ούτε απρογραμμάτιστες, αλλά ούτε και εντελώς προγραμματισμένες βάσει χρονοδιαγράμματος. Οι τεχνικές πρόβλεψης που εφαρμόζονται επιτρέπουν την ταυτοποίηση επικείμενων προβλημάτων εγκαίρως, δίνοντας τη δυνατότητα στις βιομηχανίες να προχωρήσουν σε επιδιορθώσεις των μηχανημάτων την καταλληλότερη στιγμή. Επομένως, τους επιτρέπεται ο χρόνος να προγραμματίσουν τη συντήρησή τους έτσι ώστε να μην έχει σοβαρή επίπτωση στην υπόλοιπη παραγωγική διαδικασία, μεγιστοποιώντας το διάστημα αδιάλειπτης λειτουργίας τους.

Χάρη στη σύγχρονη ψηφιοποίηση, τα αυτοματοποιημένα συστήματα και τη συλλογική συνέργεια των τεχνολογιών της Βιομηχανίας 4.0 που εφαρμόζουν πλέον οι «έξυπνες» βιομηχανίες, η συλλογή μεγάλου όγκου δεδομένων επιτυγχάνεται γρήγορα και εύκολα, γεγονός που ευνοεί την Προβλεπτική Συντήρηση, καθώς τα ιστορικά δεδομένα λειτουργίας αποτελούν αναπόσπαστο κομμάτι της συγκεκριμένης πολιτικής. Ειδικότερα, αισθητήρες κατάλληλα τοποθετημένοι πάνω σε μηχανήματα και εξαρτήματα καταγράφουν μετρήσεις κατά τη διάρκεια της λειτουργίας τους, με αποτέλεσμα τα δεδομένα που τελικά συλλέγονται να περιγράφουν την ταυτότητα του βιομηχανικού εξοπλισμού σε διάφορες περιόδους και καταστάσεις κατά την παραγωγική διαδικασία [19].

Στη συνέχεια, μετά τη συλλογή δεδομένων και την απαραίτητη προ-επεξεργασία τους, τα εργαλεία πρόβλεψης, τα οποία είναι κατάλληλα ανεπτυγμένα μοντέλα με συγκεκριμένη αρχιτεκτονική, εκπαιδεύονται πάνω στα συγκεντρωτικά δεδομένα με σκοπό να αναγνωρίσουν και να μάθουν τους μαθηματικούς κανόνες που τα διέπουν, και τυχόν κρυμμένα πολύπλοκα μοτίβα που τα χαρακτηρίζουν. Αυτή η εκπαίδευση πάνω στα ιστορικά δεδομένα αποσκοπεί τελικά στο να είναι σε θέση τα πλέον εκπαιδευμένα μοντέλα να δεχθούν νέα δεδομένα και μετρήσεις του εξοπλισμού παραγωγής και να βγάλουν κάποιο συμπέρασμα για την κατάσταση της υγείας του [20].

Προφανώς, μια τέτοια διαδικασία θα ήταν δύσκολη και χρονοβόρα, καθώς είναι σχεδόν αδύνατη να πραγματοποιηθεί μέσω συμβατικής στατιστικής ανάλυσης ή της απλής παρακολούθησης μηχανικών ενδείξεων, λόγω της τυχόν πολυπλοκότητας και όγκου των δεδομένων. Επομένως, ακόμη ένα αναπόσπαστο κομμάτι της σύγχρονης Προβλεπτικής Συντήρησης είναι η αξιοποίηση της Τεχνητής Νοημοσύνης, και συγκεκριμένα των κλάδων της Μηχανικής Μάθησης (Machine Learning) και της Βαθιάς Μάθησης (Deep Learning) [21]. Αντικείμενο των πεδίων αυτών είναι η ανάπτυξη αλγορίθμων και μοντέλων ικανών να αποδομούν τα δεδομένα που δέχονται και να αναγνωρίζουν τους υποκείμενους μαθηματικούς κανόνες που τα περιγράφουν.

Η χρήση τεχνικών Μηχανικής Μάθησης στα πλαίσια της Προβλεπτικής Συντήρησης αποτελεί σχετικά πρόσφατη τάση, κάτι που επιβεβαιώνεται και από τη ραγδαία αύξηση ερευνητικών δημοσιεύσεων τα τελευταία χρόνια [22]. Είναι αναμφίβολο ότι η διαδικασία ανάπτυξης και εκπαίδευσης ενός μοντέλου απαιτεί πολυάριθμους μαθηματικούς υπολογισμούς, και όσο αυξάνεται η πολυπλοκότητα για ένα πιο ακριβές μοντέλο, τόσο αυξάνεται και το πλήθος των αριθμητικών παραμέτρων που πρέπει να ρυθμιστούν κατά την εκπαίδευσή του. Επομένως, αυτή η αυξανόμενη παρουσία των εφαρμογών Μηχανικής Μάθησης και η ενσωμάτωσή τους στις πολιτικές συντήρησης οφείλεται τόσο στην ευκολία συλλογής δεδομένων, όσο και στη συνεχώς εξελισσόμενη υπολογιστική ισχύ των σύγχρονων επεξεργαστών, που επιτρέπουν τις γρήγορες και περίπλοκες μαθηματικές πράξεις που είναι απαραίτητες για την ανάπτυξη και την εκπαίδευση των ολοένα και πιο πολύπλοκων μοντέλων.

Τελικά, βασικοί στόχοι κάθε πολιτικής συντήρησης είναι η ελαχιστοποίηση των σφαλμάτων που παρουσιάζονται κατά την παραγωγική διαδικασία, η μεγιστοποίηση του χρόνου αδιάλειπτης λειτουργίας, η ελάττωση του χρόνου κάποιας απροσδόκητης διακοπής της, η βελτίωση της κατάστασης της υγείας του βιομηχανικού εξοπλισμού, η μεγιστοποίηση της διάρκειας ζωής του, και φυσικά η μείωση του κόστους συντήρησης στο συνολικό επιχειρησιακό κόστος. Μια βελτιστοποιημένη στρατηγική η οποία διατηρεί τον παραγωγικό εξοπλισμό σε ένα ικανοποιητικό επίπεδο μπορεί να εξασφαλίσει καλύτερη ποιότητα των παραγόμενων προϊόντων, καθώς και μια γενικότερη βελτιωμένη κατάσταση βιομηχανικής βιωσιμότητας, αποδοτικότητας και παραγωγικότητας. Η Προβλεπτική Συντήρηση θεωρείται από τις πιο υποσχόμενες στρατηγικές μέχρι σήμερα, λόγω της ικανότητάς της να επιτυγχάνει τους παραπάνω στόχους με τον πιο αποτελεσματικό τρόπο, και αυτό που της δίνει το πλεονέκτημα σε σχέση με τις υπόλοιπες πολιτικές είναι η δυνατότητα όχι μόνο της διάγνωσης, αλλά και της πρόγνωσης παραγωγικών αστοχιών.

3 Τεχνητή Νοημοσύνη

Η Τεχνητή Νοημοσύνη (Artificial Intelligence) μπορεί να περιγραφεί ως το πεδίο της Επιστήμης Υπολογιστών (Computer Science) το οποίο ασχολείται με τον σχεδιασμό και την ανάπτυξη ευφυών υπολογιστικών συστημάτων τα οποία είναι ικανά να παρουσιάσουν χαρακτηριστικά που σχετίζονται με τη νοημοσύνη που επιδεικνύει η ανθρώπινη συμπεριφορά [23]. Με άλλα λόγια, είναι η προσπάθεια να μεταδοθεί η ανθρώπινη διάνοια στις μηχανές, με σκοπό να επιτευχθεί η αυτοματοποίηση των ανθρώπινων εργασιών, όπου απαιτούνται η γνωστική λειτουργία και οι νοητικές δεξιότητες.

Αυτή τη στιγμή, το συγκεκριμένο πεδίο εμφανίζει μια ραγδαία εξέλιξη, καθώς ήδη υπάρχουν πολλά ενεργά ερευνητικά θέματα, αλλά και πληθώρα πρακτικών εφαρμογών, και εμφανίζονται συνεχώς νέα εργαλεία και λογισμικά τα οποία εκμεταλλεύονται την πρόοδο της Τεχνητής Νοημοσύνης τόσο για επιχειρηματική χρήση, όσο και για απολύτως προσωπική. Συγκεκριμένα, υπάρχουν εργαλεία ικανά να επεξεργάζονται, αλλά και να παράγουν από το μηδέν, ρεαλιστικά βίντεο και εικόνες, να δημιουργούν πίνακες ζωγραφικής μόνο από μια περιγραφή, να παράγουν μουσικά τραγούδια σε δευτερόλεπτα, αλλά και να επεξεργάζονται τον ήχο βάσει διάφορων προτιμήσεων, να μετασχηματίζουν κείμενα σε φωνητικά κομμάτια και το αντίστροφο, να γράφουν και να μεταφράζουν κείμενα, άρθρα, διαλόγους, μηνύματα, αλλά και ποιήματα. Βέβαια, πέρα από το ψυχαγωγικό κομμάτι, υπάρχουν και εργαλεία που μπορούν να υποβοηθήσουν τις επιχειρήσεις σε ένα εύρος εργασιών, όπως τον σχεδιασμό ενός λογότυπου, τη γρήγορη και εύκολη δημιουργία ιστοσελίδων, διαδικτυακών εφαρμογών και βάσεων δεδομένων, ακόμα και τη διαχείριση μάρκετινγκ, αλλά και διοικητικών και οικονομικών θεμάτων, συμβάλλοντας τελικά, με την αυτοματοποίηση των διαδικασιών, στην εξοικονόμηση χρόνου που απαιτούν πολλές επαναληπτικές εργασίες, στη μείωση τυχόν ανθρώπινων λαθών, και κατά συνέπεια στην εξοικονόμηση περιττών δαπανών και πόρων. Επίσης, σε εκπαιδευτικό επίπεδο, εφαρμογές μπορούν να προσφέρουν σε μαθητές και φοιτητές έναν πιο βέλτιστο προγραμματισμό και καλύτερη κατανόηση της διδακτέας ύλης, ειδικότερα σε περιπτώσεις μαθησιακών δυσκολιών όπου η χρήση κατάλληλων εργαλείων μπορεί να προσφέρει σημαντική διευκόλυνση [24].

Τελικά, η βοήθεια και τα οφέλη αυτών των εφαρμογών έχουν ανυψώσει την Τεχνητή Νοημοσύνη σε μια τεχνολογία που κάθε επιχείρηση ή άτομο μπορεί να αξιοποιήσει για να φέρει την επανάσταση στον τρόπο που εργάζεται ή να κάνει οποιαδήποτε άλλη καθημερινή ή ψυχαγωγική δραστηριότητα.

3.1 Ιστορική Αναδρομή

Ο όρος «Τεχνητή Νοημοσύνη», καθώς και το αντίστοιχο επιστημονικό πεδίο, παγιώθηκαν το 1956, ωστόσο πολλές έννοιες και ιδέες αποτελούσαν ήδη για δεκαετίες θέμα συζήτησης γύρω από τις μηχανές και κατά πόσο θα μπορούσε κάποιο σύστημα να μιμηθεί ικανοποιητικά την ανθρώπινη συμπεριφορά και διάνοια [25].

3.1.1 Από την αρχαιότητα μέχρι και τον 18^{ο} αιώνα

Ανατρέχοντας στην εποχή της Αρχαίας Ελλάδας, και συγκεκριμένα στην ελληνική μυθολογία, μπορούμε να συναντήσουμε τις έννοιες της τεχνητής ζωής σε ιστορίες που προκαλούν ιδιαίτερο ενδιαφέρον. Για παράδειγμα, ο Ήφαιστος σφυρηλάτησε τον γιγαντιαίο χάλκινο Τάλω για να προστατεύσει την πριγκίπισσα Ευρώπη, αλλά δημιούργησε και την πρώτη πραγματική γυναίκα, την Πανδώρα, ο Πυγμαλίωνας σκάλισε από ελεφαντόδοντο το άγαλμα της Γαλάτειας, την οποία αργότερα ζωντάνεψε η Αφροδίτη, ενώ για τον Δαίδαλο λέγεται ότι τα κινούμενα γλυπτά του ήταν τόσο αληθοφανή ως προς τα χαρακτηριστικά τους που θεωρούνταν ζωντανά. Επιπλέον, κοιτώντας και προς άλλους πολιτισμούς, βουδιστικοί και αιγυπτιακοί θρύλοι αναφέρουν μηχανικούς προστάτες και αυτόνομα αγάλματα, ενώ αποδείξεις για μηχανικά διαγράμματα χρονολογούνται από τον 10^{ο} αιώνα π.Χ., όπου ο τεχνίτης Γιαν Σι (Yan Shi) από την Δυναστεία των Δυτικών Ζου (Western Zhou Dynasty) κατασκεύασε ανθρωποειδή αυτόματα ικανά να τραγουδούν και να χορεύουν [26].

Επίσης, αξίζει να αναφερθούν τα πήλινα αγαλματίδια (golem) των εβραϊκών μύθων, οι χάλκινες κεφαλές (brazen head) στα τέλη του Μεσαίωνα, ο Ήρων ο Αλεξανδρεύς (60 μ.Χ.) ο οποίος εφηύρε αυτόματα χρησιμοποιώντας μηχανισμούς που εκμεταλλεύονταν τη δύναμη του νερού, του ατμού και του αέρα [27], η αναζήτηση για τη δημιουργία της συνθετικής ζωής που αποτελούσε τον κύριο στόχο των μουσουλμάνων αλχημιστών περί το 800 μ.Χ. και αναφέρεται ως Τακγουίν (Takwin), και τελικά ο Ισμαήλ αλ Τζαζάρι (Ismail al-Jazari) ο οποίος αναφέρεται και ως ο πατέρας της ρομποτικής λόγω της σημαντικής συνεισφοράς του κατά τον 12^{ο} αιώνα μ.Χ. με τις περίτεχνες εφευρέσεις του. Μάλιστα, πιστεύεται ότι σχεδίασε το πρώτο προγραμματιζόμενο ρομπότ, ένα σκάφος με τέσσερις μηχανικούς μουσικούς οι οποίοι μπορούσαν να ρυθμιστούν έτσι ώστε να παίζουν μουσική σε διαφορετικούς ρυθμούς και κινούνταν με τη δύναμη της ροής του νερού [28].

Τον 17ο αιώνα μ.Χ. ο Γάλλος φιλόσοφος και μαθηματικός Ρενέ Ντεκαρτέ, αναφερόμενος στα μηχανικά κινούμενα αυτόματα της εποχής που παρουσιάζονταν ως ζωντανές απομιμήσεις και των οποίων η λειτουργία βασίζονταν στη σχετικά νεότερη τεχνολογία των ρολογιών, υποστήριξε ότι θα ήταν αδύνατο να δημιουργηθεί μια μηχανή ικανή να σκέφτεται επαρκώς όπως ο άνθρωπος [29]. Επιπλέον, αναφέρει ότι αν θα ήταν ποτέ εφικτό να φτιαχτούν μηχανές κατ' εικόνα του ανθρώπου και ικανές να μιμηθούν τη συμπεριφορά του με όσο το δυνατόν ηθικότερο τρόπο, θα υπήρχαν δύο συγκεκριμένα τεστ στα οποία θα αποτύγχαναν. Το πρώτο αναφέρεται ως το Τεστ της Γλώσσας, σύμφωνα με το οποίο, μια τέτοια μηχανή δεν θα ήταν ικανή να αποδώσει λεκτικά, ή με οποιοδήποτε άλλον τρόπο, το νόημα των σκέψεών της ή να αντιδράσει πειστικά σε ένα ξαφνικό εξωτερικό ερέθισμα. Το δεύτερο τεστ αναφέρεται ως το Τεστ της Δράσης, σύμφωνα με το οποίο, παρά την ικανότητα που θα παρουσίαζαν οι παραπάνω μηχανές στο να εκτελέσουν πολλά πράγματα το ίδιο καλά, ή ακόμα και καλύτερα, σε σύγκριση με τον άνθρωπο, υπάρχουν αναμφίβολα κάποιες περιπτώσεις όπου θα αποτύγχαναν παταγωδώς ακόμα και σε σχέση με κάποιο ον ελάχιστης διάνοιας, γιατί θα βασίζονταν μόνο στην κατασκευή τους κι όχι στην κατανόηση της κατάστασης [30]. Παρά το ενδιαφέρον που παρουσιάζει η παραπάνω οπτική, τα μηχανικά αυτόματα μέχρι εκείνη την εποχή δύσκολα θα μπορούσαν να χαρακτηριστούν ως μηχανές που μιμούνται την ανθρώπινη σκέψη.

Κατά τον 18ο αιώνα μ.Χ., με τον κόσμο της βιομηχανίας να μηχανοποιείται και να εξελίσσεται ολοένα και περισσότερο, παρουσιάστηκε μια αφθονία μηχανικών παιχνιδιών, τα οποία συνέχιζαν βέβαια να στηρίζονται κυρίως σε μηχανισμούς ρολογιών. Αναφορικά, ο Ζαν ντε Βοκανσόν (Jacques de Vaucanson), ο οποίος αποτελεί σημαντική φιγούρα της Βιομηχανικής Επανάστασης, καθώς σχεδίασε τον πρώτο αυτόματο αργαλειό, κατασκεύασε μια μηχανική πάπια (Digesting Duck) η οποία μπορούσε να μιμείται τις κινήσεις των ζωντανών, και μάλιστα να πίνει, να τρώει και να αφοδεύει. Φυσικά, ήταν μια ειδικά περίτεχνη και καινοτόμα κατασκευή χωρίς κάποιο εγκατεστημένο βιολογικό ή χημικό σύστημα, αλλά ήταν από τις πιο αξιοσημείωτες εφευρέσεις μεταξύ των υπόλοιπων αυτομάτων που δημιούργησε [31].

Επιπλέον, ο Βόλφγκαν φον Κέμπελεν (Wolfgang von Kempelen) έφτιαξε τον Τούρκο (The Turk), έναν μηχανικό σκακιστή ο οποίος έπαιζε τόσο καλά που κατάφερε να έρθει αντιμέτωπος με τους καλύτερους σκακιστές της εποχής. Επίσης, ήταν εκπληκτικό όχι μόνο το γεγονός ότι κέρδισε πάρα πολλές παρτίδες, αλλά ότι είχε και την ικανότητα να κουνάει μόνος τα πιόνια του και να αναγνωρίζει παράνομες κινήσεις των αντιπάλων του. Το αυτόματο προκάλεσε ιδιαίτερη εντύπωση, έκανε περιοδείες σε Ευρώπη και Βόρεια Αμερική

και ανταγωνίστηκε ακόμα και τον Ναπολέων Βοναπάρτη και τον Βενιαμίν Φραγκλίνο. Στην πραγματικότητα, ήταν με τέτοιο τρόπο κατασκευασμένο έτσι ώστε να μπορεί να κρυφτεί μέσα του κάποιο άτομο που ήλεγχε τις κινήσεις του, αλλά μέχρι να αποκαλυφθεί το συγκεκριμένο μυστικό πέρασαν δεκαετίες όπου αυτή η περίτεχνη ψευδαίσθηση τροφοδοτούσε την ιδέα των σκεπτόμενων μηχανών [32].

Αν και πολλά από τα παραπάνω παραδείγματα, μεταξύ πολλών άλλων τα οποία συναντώνται στην ιστορία, έχουν περισσότερο κλίση προς τη ρομποτική και τη μηχανολογία, είναι αναμφίβολο ότι, ήδη από την Αρχαιότητα, φιλόσοφοι, συγγραφείς, λόγιοι, μηχανικοί και εφευρέτες απασχολούνταν με θεωρίες γύρω από τα άψυχα αντικείμενα, τις σκεπτόμενες μηχανές και τη δυνατότητα τεχνητής ζωής και διάνοιας ισότιμης της ανθρώπινης. Ωστόσο, τα θεμέλια της Τεχνητής Νοημοσύνης όπως την γνωρίζουμε σήμερα άρχισαν να εμφανίζονται αρκετά αργότερα [33].

3.1.2 Τα θεμέλια του 19ου αιώνα

Στις αρχές του 19ου αιώνα ο Ζοζέφ-Μαρί Ζακάρ (Joseph-Marie Jacquard) εφηύρε, βασιζόμενος και στα πρώιμα σχέδια του Ζαν ντε Βοκανσόν, τον Αργαλειό Ζακάρτ [34], την πρώτη προγραμματιζόμενη μηχανή που χρησιμοποιούσε κατάλληλα διατρητές κάρτες εναλλάξιμα για να δεχτεί οδηγίες για τη ρύθμιση και τον έλεγχο της ύφανσης, δημιουργώντας περίτεχνα μοτίβα [35]. Αυτή η αξιοποίηση των διατρητών καρτών αποτέλεσε τον πρόδρομο του προγραμματισμού υπολογιστών και της εισαγωγής δεδομένων.

Στη συνέχεια, ο Τσαρλς Μπάμπατζ (Charles Babbages), ασχολήθηκε με την εφεύρεση της Διαφορικής Μηχανής (Difference Engine), η οποία αποτέλεσε την πιο πολύπλοκη και περίτεχνη μηχανική αριθμομηχανή εκείνης της εποχής [36]. Παρά την πολυπλοκότητά της όμως, μπορούσε να εκτελέσει μόνο μια πράξη κάθε φορά, και μάλιστα με αριθμούς λίγων ψηφίων, μετά από συγκεκριμένη ρύθμιση των μηχανικών τμημάτων της [37]. Παράλληλα, ο Μπάμπατζ ανέπτυξε σχέδια για μια πιο βελτιωμένη εκδοχή της Διαφορικής Μηχανής, την Αναλυτική Μηχανή (Analytical Engine), η οποία μάλιστα βασίστηκε στις διατρητές κάρτες του Ζακάρ. Η Αναλυτική Μηχανή σχεδιάστηκε έτσι ώστε να είναι πλήρως προγραμματιζόμενη και να μπορεί να εκτελεί περισσότερες μαθηματικές πράξεις, με μεγαλύτερο πλήθος αριθμών και με αριθμούς περισσότερων ψηφίων, καθώς και να έχει δυνατότητες μνήμης και αποθήκευσης δεδομένων υψηλότερου επιπέδου. Ωστόσο, μέχρι και τον θάνατό του το 1871, λόγω διοικητικών προβλημάτων με κυβερνητικές χορηγίες,

κανένα εγχείρημά του δεν ολοκληρώθηκε όπως το είχε αρχικά φανταστεί. Παρ' όλα αυτά όμως έθεσε τα θεμέλια των αυτόματων υπολογισμών και η Αναλυτική Μηχανή θεωρείται ως ο πρώτος αυτόματος και πλήρως προγραμματιζόμενος ψηφιακός υπολογιστής γενικής χρήσης, και πρόδρομος του σύγχρονου ψηφιακού υπολογιστή, έννοιες που μέχρι τότε δεν είχαν καν επινοηθεί [38].

Επιπλέον, το 1843, η Έιντα Λάβλεϊς (Ada Lovelace), φίλη και συνεργάτης του Τσαρλς Μπάμπατζ, αναφέρει σε σημειώσεις της ότι δεν υπήρχαν φιλοδοξίες για την Αναλυτική Μηχανή να παρουσιάσει κάτι πρωτότυπο, αλλά ότι μπορούσε να κάνει οτιδήποτε υποδείκνυαν οι οδηγίες που δεχόταν, και ο σκοπός της ήταν να βοηθήσει στη διαθεσιμότητα της γνώσης [39]. Οι υποσημειώσεις αυτές έρχονται να συμπληρώσουν τη δημοσίευση του Ιταλού μαθηματικού και μηχανικού Λουίτζι Φεντερίκο Μεναμπρέα (Luigi Federico Menabrea) ο οποίος υποστήριξε, περιγράφοντας της Αναλυτική Μηχανή, ότι μια μηχανή δεν έχει σκεπτόμενη υπόσταση, αλλά είναι ένα απλό αυτόματο που υπακούει σε νόμους που του επιβάλλονται. Παρά το γεγονός ότι η Αναλυτική Μηχανή δεν θεωρήθηκε ως «σκεπτόμενη μηχανή», υπήρξε από τα πρώτα μεγάλα βήματα προς τη σχεδίαση υπολογιστών ικανών για πολύπλοκους μαθηματικούς υπολογισμούς, ενώ η ίδια η Έιντα Λάβλεϊς αναφέρεται σήμερα ως η πρώτη προγραμματίστρια υπολογιστών, με την ομώνυμη γλώσσα προγραμματισμού Έιντα (Ada) προς τιμήν της [40].

3.1.3 Το πρώτο μισό του 20ου αιώνα

Στις αρχές του 20ου αιώνα οι Άλφρεντ Νορθ Γουάιτχεντ (Alfred North Whitehead) και Μπέρτραντ Ράσελ (Bertrand Arthur William Russell) δημοσίευσαν την «Αρχή των Μαθηματικών» (Principia Mathematica) [41], ένα βιβλίο το οποίο, μαζί με μεταγενέστερες εκδόσεις του, προσπάθησε να ορίσει τις μαθηματικές έννοιες με λογικούς όρους. Τελικά, κατάφερε να φέρει την επανάσταση στη σύγχρονη μαθηματική λογική, και έθεσε τα θεμέλια του ελέγχου και της εξαγωγής τύπων (type checking - type inference) οι οποίοι χρησιμοποιήθηκαν αργότερα σε αλγορίθμους απόδειξης θεωρημάτων, καθιερώνοντας έτσι και τα μαθηματικά στον προγραμματισμό υπολογιστών.

Επίσης, το 1936, στο πλαίσιο της θεμελίωσης των μαθηματικών και της λογικής, ο Άλαν Τούρινγκ (Alan Turing), σε δημοσίευσή του σχετικά με τους υπολογίσιμους αριθμούς [42], περιέγραψε την «καθολική υπολογιστική μηχανή», η οποία σήμερα ονομάζεται «Καθολική Μηχανή Τούρινγκ» προς τιμήν του [43]. Πρόκειται για μια αφηρημένη υπολογιστική συσκευή η οποία είχε σκοπό να διερευνήσει την έκταση και τους περιορισμούς του τι είναι

δυνατό να υπολογιστεί μέσα από εφαρμογή πάνω στο πρόβλημα λήψης αποφάσεων (Entscheidungsproblem). Τελικά, η επιρροή των θεωρητικών μοντέλων των «Μηχανών Τούρινγκ» για την προσομοίωση της λογικής, είχε σημαντική επίδραση στον τομέα της Επιστήμης Υπολογιστών, τόσο στη θεωρητική θεμελίωσή της, όσο και στην τελική ανάπτυξη του σύγχρονου υπολογιστή [44].

Το 1943 ο νευρολόγος Γουόρεν ΜακΚάλοχ (Warren McCulloch) και ο μαθηματικός Γουόλτερ Πιττς (Walter Pitts), επηρεασμένοι από την πολυπλοκότητα του δικτύου των νευρώνων του εγκεφάλου και την ικανότητά του να αντιλαμβάνεται σύνθετα μοτίβα μέσω διασυνδεδεμένων κυττάρων, παρουσίασαν σε δημοσίευσή τους ένα θεωρητικό μοντέλο ενός τεχνητού νευρώνα [45], αναφέροντας μάλιστα ότι το κατάφεραν αντιμετωπίζοντας τον εγκέφαλο ως μια «Μηχανή Τούρινγκ». Η συγκεκριμένη καινοτομία έθεσε τα θεμέλια για την ανάπτυξη των Τεχνητών Νευρωνικών Δικτύων (Artificial Neural Networks – ANN), τα οποία σήμερα αποτελούν δημοφιλή τεχνολογία του τομέα της Τεχνητής Νοημοσύνης [46].

Το 1950 ο Κλωντ Σάνον (Claude Shannon) [47], ο οποίος θεωρείται ο πατέρας της θεωρίας της πληροφορίας, δημοσίευσε μια λεπτομερή ανάλυση για τον προγραμματισμό ενός υπολογιστή με τη δυνατότητα να παίζει σκάκι. Επίσης, υποστήριξε πως μόνο ένα πρόγραμμα με κάποιο επίπεδο διάνοιας θα μπορούσε να παίξει με στρατηγικό τρόπο, διότι ένας απλός αλγόριθμος, που θα βασιζόταν σε μεθόδους εξαντλητικής αναζήτησης (brute force) για να αποφασίσει την επόμενη κίνησή του, θα χρειαζόταν χρόνια μέχρι να ελέγξει κάθε εναλλακτική [48].

Την ίδια χρονική περίοδο, ο Άλαν Τούρινγκ δημοσίευσε μια επίσης πρωτοποριακή έκθεση, γνωστή ως «Υπολογιστικές Μηχανές και Νοημοσύνη» (Computing Machinery and Intelligence) [49]. όπου παρουσίασε βασικές έννοιες για τη δημιουργία σκεπτόμενων μηχανών, και αναφέρθηκε στις ενστάσεις και τα επιχειρήματα τα οποία έρχονταν σε αντίθεση με την ιδέα ότι είναι δυνατόν να προσδοθεί διανοητική σκέψη σε άψυχα αντικείμενα. Επιπλέον, πρότεινε ένα υποθετικό τεστ σε μορφή παιχνιδιού ερωτήσεων, γνωστό ως «Το Παιχνίδι της Μίμησης» (The Imitation Game), το οποίο ελέγχει κατά πόσο μια σκεπτόμενη μηχανή μπορεί να μιμηθεί έναν άνθρωπο ή να διαχωριστεί από αυτόν. Παρόλο που το συγκεκριμένο τεστ ήταν περισσότερο φιλοσοφικής φύσης, εξακολουθεί να θεωρείται μέχρι και σήμερα σημείο αναφοράς για την επιβεβαίωση νοημοσύνης σε τεχνητά συστήματα, και ο ίδιος ο Άλαν Τούρινγκ, λόγω της σημαντικής συνεισφοράς του, θεωρείται από πολλούς ως ο πατέρας της Επιστήμης Υπολογιστών, αλλά και ένας από τους πρωτοπόρους του τομέα της Τεχνητής Νοημοσύνης [50].

Είναι σημαντικό να τονιστεί το γεγονός ότι εκείνη την περίοδο οι υπολογιστές δεν μπορούσαν να αποθηκεύσουν εντολές, παρά μόνο να τις εκτελέσουν. Έτσι, χωρίς κάποια δυνατότητα μνήμης για την αποθήκευση δεδομένων, δεν ήταν εφικτό να αποκτήσουν κάποια ευφυΐα. Επιπλέον, η χρήση των υπολογιστών ήταν ιδιαίτερα ακριβή διαδικασία, και μια απλή ενοικίαση ενός μήνα μπορούσε να φτάσει ακόμα και τις 200 χιλιάδες δολάρια. Για τον λόγο αυτό, μόνο τα πιο διακεκριμένα πανεπιστήμια και οι μεγαλύτερες εταιρίες τεχνολογίας μπορούσαν τα αντέξουν το οικονομικό κόστος που συνόδευε την υπολογιστική έρευνα σχετικά με την νοημοσύνη των μηχανών [51].

3.1.4 Από τα μέσα του 20ου αιώνα μέχρι σήμερα

Τα μέσα του 20ου αιώνα σηματοδοτούν την αρχή μιας εντονότερης μελέτης πάνω στην τεχνητή διάνοια. Στηριζόμενοι σε προγενέστερες προσπάθειες, επιστήμονες καταφέρανε σημαντικές υλοποιήσεις και εφαρμογές [52], οι οποίες τελικά παγίωσαν την Τεχνητή Νοημοσύνη ως πεδίο έρευνας. Ωστόσο, αυτή η γρήγορη εξέλιξη οδήγησε σε βιαστικές και ανέφικτες προσδοκίες οι οποίες είχαν ως αποτέλεσμα αναπόφευκτες περιόδους στασιμότητας [53].

Το 1954, στο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης (Massachusetts Institute of Technology – M.I.T.), οι Μπέλμοντ Φάρλεϊ (Belmont Farley) και Γουέσλι Κλαρκ (Wesley Clark) κατάφεραν να εκτελέσουν το πρώτο Τεχνητό Νευρωνικό Δίκτυο, το οποίο αποτελούνταν, λόγω περιορισμένης μνήμης, από το πολύ 128 νευρώνες και εκπαιδεύτηκε για να αναγνωρίζει απλά μοτίβα.

Επιπλέον, το 1955, στο τότε Ινστιτούτο Τεχνολογίας Κάρνεγκι (Carnegie Institute of Technology), οι Άλλεν Νιούελ (Allen Newell), Χέρμπερτ Σάιμον (Herbert Simon) και Κλιφ Σω (Cliff Shaw) παρουσίασαν το πρώτο εκτελέσιμο πρόγραμμα τεχνητής νοημοσύνης το οποίο ονομαζόταν «Λογικός Θεωρητικός» (Logic Theorist) [54], και ήταν ικανό να αποδείξει μαθηματικά θεωρήματα από την «Αρχή των Μαθηματικών» των Ράσελ και Γουάιτχεντ, βρίσκοντας βασικές λογικές εξισώσεις, ενώ μάλιστα ξεπέρασε τις προσδοκίες καταφέρνοντας να βρει μια καινούρια και ακόμα καλύτερη απόδειξη σε μια περίπτωση.

Το καλοκαίρι του 1956, μετά από πρόταση των Τζον ΜακΚάρθι, Μάρβιν Μίνσκυ, Νέηθαν Ρότσεστερ και Κλωντ Σάννον (J. McCarthy, M. L. Minsky, N. Rochester, C. E. Shannon), έλαβε χώρα στο Κολλέγιο Ντάρτμουθ το πρώτο ερευνητικό πρόγραμμα πάνω στο πεδίο της Τεχνητής Νοημοσύνης (Dartmouth Summer Research Project on Artificial Intelligence – DSRPAI) [55]. Σκοπός του συνεδρίου ήταν να βρεθούν τυποποιημένες μέθοδοι που θα

επιτρέπουν σε μηχανές να χρησιμοποιούν τη γλώσσα, να διαμορφώνουν αφηρημένες έννοιες, να επιλύνουν ανθρώπινα προβλήματα και να αυτοβελτιώνονται, και όλα αυτά βάσει της υπόθεσης ότι κάθε πτυχή μάθησης και νοημοσύνης μπορεί να περιγραφεί με τόσο ακριβή λεπτομέρεια έτσι ώστε μια μηχανή να μπορεί να τη μιμηθεί.

Στο συνέδριο αυτό παρουσιάστηκε ο «Λογικός Θεωρητικός» και η δυνατότητά του να μιμείται τις ανθρώπινες δυνατότητες επίλυσης προβλημάτων οδήγησαν στο να θεωρηθεί από πολλούς ως το πρώτο πρόγραμμα τεχνητής νοημοσύνης. Παρά το γεγονός ότι μέχρι το τέλος του καλοκαιριού ο στόχος του συνεδρίου δεν επιτεύχθηκε, και κανένα από τα προτεινόμενα προβλήματα δεν επιλύθηκε, η σημασία του αναγνωρίζεται ακόμη και σήμερα, καθώς εκεί εδραιώθηκε ο όρος «Τεχνητή Νοημοσύνη» και αναγνωρίστηκε ως ακαδημαϊκό πεδίο έρευνας, οδηγώντας έτσι σε μια χρυσή εποχή δυο δεκαετιών γεμάτη με τεχνολογικές καινοτομίες.

Το 1957 ο Φρανκ Ρόζενμπλατ (Frank Rosenblatt) επινόησε το Αντίληπτρο (Perceptron), το οποίο σήμερα αποτελεί ένα από τα απλούστερα Τεχνητά Νευρωνικά Δίκτυα, και έδωσε ιδιαίτερη έμφαση στη σημασία της μάθησης μέσω της δημιουργίας και του μετασχηματισμού συνδέσεων μεταξύ νευρώνων. Από τις σημαντικότερες συνεισφορές του στο ερευνητικό πεδίο ήταν η γενίκευση της μεθόδου της εκπαιδευτικής διαδικασίας των Φάρλεϊ και Κλαρκ σε πολυεπίπεδα νευρωνικά δίκτυα, χρησιμοποιώντας τη φράση «διόρθωση σφαλμάτων μέσω οπισθοδιάδοσης» (back-propagating error correction), κάτι το οποίο αποτελεί πλέον αναπόσπαστο κομμάτι στην εκπαίδευση νευρωνικών δικτύων.

Την ίδια χρονιά, οι Νιούελ, Σάιμον και Σω, οι δημιουργοί του «Λογικού Θεωρητικού», έγραψαν ακόμα ένα ισχυρότερο πρόγραμμα, τον «Γενικό Επιλυτή Προβλημάτων» (General Problem Solver – GPS), που βασιζόταν σε αρχιτεκτονική συμβολικής λογικής και μπορούσε να βρει λύσεις σε μια εντυπωσιακή ποικιλία δομημένων προβλημάτων, όπως λογικές αποδείξεις και μαθηματικά προβλήματα λέξεων, ακολουθώντας την ευριστική διαδικασία της δοκιμής και σφάλματος (trial and error). Ωστόσο, μια επίκριση που έλαβε, μαζί του και άλλα παρόμοια ευριστικά προγράμματα, ήταν ότι η ευφυΐα του ήταν δευτερογενής και προερχόταν από το τί πληροφορία θα συμπεριελάμβανε ο προγραμματιστής.

Το 1958 ο Τζον ΜακΚάρθι, στο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης, ανέπτυξε την συναρτησιακή γλώσσα προγραμματισμού LISP (list processing) [56], η οποία έγινε η πιο δημοφιλής γλώσσα για την έρευνα πάνω στην τεχνητή νοημοσύνη, καθώς επέτρεπε τη δημιουργία ευέλικτων προγραμμάτων, όπου η αναπαράσταση βασικών πράξεων και δεδομένων γινόταν με τη δομή λιστών. Επίσης, ένας ακόμα λόγος που χρησιμοποιήθηκε

εκτενώς ήταν ότι η διαδικασία της μάθησης μπορούσε να επιτευχθεί με αυτοτροποποιούμενα προγράμματα.

Γενικότερα, με το πέρασμα των χρόνων, οι υπολογιστές μπορούσαν να αποθηκεύσουν περισσότερες πληροφορίες, έγιναν γρηγορότεροι, φθηνότεροι και ευκολότερα διαθέσιμοι, οδηγώντας έτσι σε μια άνθιση της έρευνας στο πεδίο με πιο βελτιωμένους αλγορίθμους και καλύτερη κατανόηση των προβλημάτων που έπρεπε να αντιμετωπιστούν. Για παράδειγμα, κάποιες σημαντικές καινοτομίες εκείνης της περιόδου ήταν οι παρακάτω:

- SAINT (Symbolic Automatic INTegrator) – Το πρώτο ευριστικό πρόγραμμα συμβολικής ολοκλήρωσης ικανό να λύσει προβλήματα λογισμού χαμηλού επιπέδου.
- STUDENT – Πρώιμο επίτευγμα πάνω στην Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing – NLP), γραμμένο σε LISP και σχεδιασμένο να διαβάζει, να κατανοεί και να λύνει μαθηματικά προβλήματα λέξεων από σχολικά βιβλία.
- ELIZA – Ένα διαδραστικό υπολογιστικό πρόγραμμα Επεξεργασίας Φυσικής Γλώσσας που μπορούσε να κάνει διάλογο στα Αγγλικά για οποιοδήποτε θέμα, και θεωρείται το πρώτο chatbot.
- PARRY – Ακόμα ένα πρώιμο παράδειγμα chatbot, μόνο που το συγκεκριμένο αναπτύχθηκε για να προσποιηθεί έναν άνθρωπο με σχιζοφρένεια, και μάλιστα «συνομίλησε» με το ELIZA, το οποίο ανέλαβε τον ρόλου του «Γιατρού» [57].
- DENDTRAL – Ευριστικό εμπειρικό σύστημα (expert system) σχεδιασμένο για τη χημική ανάλυση πολύπλοκων ενώσεων, ικανό να υποθέτει την ατομική δομή ουσιών από φασματογραφίες, και με επιδόσεις αντίστοιχες με αυτές έμπειρων χημικών, το οποίο τελικά χρησιμοποιήθηκε σε βιομηχανίες και στην εκπαίδευση.
- SHRDLU – Πρώιμο πρόγραμμα κατανόησης της φυσικής γλώσσας το οποίο υπάκουε σε γραπτές εντολές και μπορούσε να δώσει απαντήσεις για τις πράξεις του.
- Shakey – Ο συνδυασμός της Ρομποτικής, της Μηχανικής Όρασης (Computer Vision) και της Επεξεργασίας Φυσικής Γλώσσας οδήγησε στη δημιουργία του πρώτου γενικής χρήσεως κινούμενου ρομπότ το οποίο συνδυάζοντας κίνηση, αντίληψη και επίλυση προβλημάτων κατάφερε να αλληλεπιδρά με το περιβάλλον.
- WABOT-1 (WAseda roBOT) – Το πρώτο πλήρους κλίμακας ευφυές ανθρωποειδές ρομπότ που αποτελούνταν από συστήματα ελέγχου άκρων, όρασης και ομιλίας, μπορώντας έτσι μιλήσει Ιαπωνικά, να αντιληφθεί αποστάσεις και το γύρω περιβάλλον, να περπατήσει και να μεταφέρει αντικείμενα [58].

- PROLOG – Γλώσσα προγραμματισμού, εναλλακτική της LISP, για εφαρμογές λογικού προγραμματισμού και σχεδιασμένη για να διαχειρίζεται υπολογιστική γλωσσολογία, ειδικότερα για την Επεξεργασία Φυσικής Γλώσσας.
- MYCIN – Επίσης πρώιμο έμπειρο σύστημα, το οποίο βασιζόμενο σε συμπτώματα και ιατρικά τεστ επιχειρούσε να διαγνώσει βακτηριακές λοιμώξεις του αίματος σε ασθενείς και να προτείνει την κατάλληλη θεραπεία, ή, αν δεν είχε αρκετά στοιχεία, να ζητήσει επιπλέον εξετάσεις, λειτουργώντας σχεδόν στο ίδιο επίπεδο με έμπειρους λοιμωξιολόγους.

Αξίζει να σημειωθεί ότι, παράλληλα με την παραπάνω πρόοδο, το 1963 ο Τζον ΜακΚάρθι ίδρυσε το Πρόγραμμα ΜΑΚ (Project MAC – Project on Mathematics and Computation) στο Ινστιτούτο Τεχνολογίας της Μασαχουσέτης, το οποίο σήμερα είναι γνωστό ως Εργαστήριο Επιστήμης Υπολογιστών και Τεχνητής Νοημοσύνης του M.I.T (Computer Science and Artificial Intelligence Laboratory – CSAIL) [59]. Επίσης, την ίδια χρονιά, μαζί με τον Μάρβιν Μίνσκυ, δημιούργησαν το Εργαστήριο Τεχνητής Νοημοσύνης του Στάνφορντ (Stanford Artificial Intelligence Laboratory – SAIL). Η προοπτική του τομέα της Τεχνητής Νοημοσύνης και η υπεράσπιση της έρευνας, κυρίως από συμμετέχοντες του συνεδρίου DSRPAI, έπεισε την κυβέρνηση, αλλά και σωματεία όπως τον Οργανισμό Προηγμένων Ερευνητικών Προγραμμάτων Άμυνας (Defense Advanced Research Projects Agency – DARPA), να προχωρήσουν σε χρηματοδοτήσεις που τελικά οδήγησαν στο σχηματισμό των παραπάνω ινστιτούτων στρώνοντας τον δρόμο για την ανάπτυξη συστημάτων, αρκετά από τα οποία μάλιστα αναφέρθηκαν και παραπάνω.

Από το 1957 μέχρι το 1973 το πεδίο της Τεχνητής Νοημοσύνης άνθισε ιδιαίτερα και οδήγησε στη δημιουργία αρκετά υψηλών προσδοκιών, όμως αυτή η πρόοδος άρχισε να σταθεροποιείται χωρίς κάποια σημαντική εξέλιξη και καινοτομία. Το μεγαλύτερο εμπόδιο αποτέλεσε η έλλειψη της αναγκαίας υπολογιστικής ισχύος, καθώς πλέον οι υπολογιστές δεν μπορούσαν να αποθηκεύσουν αρκετή πληροφορία ή να επεξεργαστούν γρήγορα δεδομένα. Έτσι, τόσο η Βρετανική όσο και η Αμερικανική κυβέρνηση αμφισβήτησαν την αισιόδοξη προοπτική των ερευνητών και τελικά σταμάτησαν την οικονομική υποστήριξη για περαιτέρω έρευνα στον τομέα, οδηγώντας σε μια περίοδο η οποία είναι γνωστή ως ο πρώτος χειμώνας της Τεχνητής Νοημοσύνης (first AI winter) και διήρκησε περίπου μέχρι τις αρχές του 1980 [60].

Το 1980 πραγματοποιείται το πρώτο εθνικό συνέδριο της Αμερικανικής Ένωσης Τεχνητής Νοημοσύνης (American Association of Artificial Intelligence – AAAI) στο Πανεπιστήμιο του Στάνφορντ, ενώ τα Έμπειρα Συστήματα (Expert Systems), την έννοια των οποίων εισήγαγε

επίσημα ο Έντουαρτ Φάιγκενμπαουμ (Edward Feigenbaum) την δεκαετία του ’60 και ήδη είχαν χρησιμοποιηθεί για την ανάπτυξη πληθώρας εφαρμογών, άρχισαν να γίνονται ιδιαίτερα δημοφιλή από μεγάλες εταιρίες και βιομηχανίες. Η Ιαπωνική κυβέρνηση, μέχρι το 1990, χρηματοδότησε εκατοντάδες εκατομμύρια σε έμπειρα συστήματα, μεταξύ και άλλων προσπαθειών, ως μέρος του Προγράμματος Υπολογιστών Πέμπτης Γενιάς (Fifth Generation Computer Project – FGCP), ωθώντας έτσι και τον αμερικανικό οργανισμό DARPA να προχωρήσει και αυτός σε αυξήσεις χρηματοδοτήσεων. Επιπλέον, ακόμη και οι μεγαλύτερες υπολογιστικές επιχειρήσεις άρχισαν να ξοδεύουν δισεκατομμύρια για την ανάπτυξη υλικού και λογισμικού για το πέρασμα από την «επεξεργασία πληροφοριών» στην «επεξεργασία γνώσης», και μάλιστα προχώρησαν και στη δημιουργία δικών τους τμημάτων έρευνας πάνω στην Τεχνητή Νοημοσύνη.

Το 1984 ξεκίνησε το μακροπρόθεσμο πρόγραμμα CYC, το οποίο συνεχίζεται μέχρι και σήμερα, και αποτελεί ένα μεγάλο πείραμα στη συμβολική Τεχνητή Νοημοσύνη με τον φιλόδοξο στόχο να δημιουργήσει μια βάση γνώσεων η οποία θα περιέχει ένα σημαντικό ποσοστό της κοινής λογικής γνώσης που διαθέτει ένας άνθρωπος. Το 1986 στο Πανεπιστήμιο Κάρνεγκι Μέλον (Carnegie Mellon University – CMU) μηχανικοί κατάφεραν να φτιάξουν, με χρηματοδότηση της DARPA, την πρώτη εκδοχή του Navlab, το οποίο ήταν το πρώτο πλήρως αυτόνομο αυτοκίνητο. Παράλληλα, αν και είχαν αργή εξέλιξη, άρχισαν να αναπτύσσονται νέες τεχνικές εκπαίδευσης και αρχιτεκτονικές πάνω στα Τεχνητά Νευρωνικά Δίκτυα και τον αλγόριθμο της Οπισθοδιάδοσης (Back-propagation), με την πρώτη επιτυχημένη πρακτική εφαρμογή να έρχεται από τον Γιαν ΛεΚουν (Yann LeCun) το 1989 και το πρωτότυπο σύστημα LeNet το οποίο ήταν ικανό να ταξινομεί χειρόγραφα μαθηματικά ψηφία, και που αργότερα χρησιμοποιήθηκε από την Ταχυδρομική Υπηρεσία των Ηνωμένων Πολιτειών για την ανάγνωση των Ταχυδρομικών Κωδικών στους φακέλους.

Τελικά, κατά το ’90, οι υπερεκτιμημένοι στόχοι για την υπολογιστική επανάσταση, τον λογικό προγραμματισμό και τη βελτίωση της Τεχνητής Νοημοσύνης που είχαν τεθεί δεν επιτεύχθηκαν, και για ακόμη μια φορά οι επενδυτές και οι κυβερνήσεις απέσυραν τις χρηματοδοτήσεις για την έρευνα, καθώς τα συστήματα εκείνης της εποχής είχαν αρκετούς περιορισμούς, δεν είχαν να προσφέρουν κάτι καινοτόμο και τα κόστη για την συντήρησή τους ήταν ιδιαίτερα ψηλά σε σύγκριση με τους επιτραπέζιους υπολογιστές, ειδικά στην περίπτωση των Έμπειρων Συστημάτων όπου υπήρχε σχετική δυσκολία στην αναβάθμιση και στην εκπαίδευσή τους [61]. Αυτή η περίοδος, από το 1987 μέχρι το 1993, αναγνωρίζεται ως ο δεύτερος χειμώνας της Τεχνητής Νοημοσύνης, αλλά παρά την έλλειψη κυβερνητικών

χρηματοδοτήσεων και τη γενικότερη αδιαφορία προς την εξέλιξη του τομέα, η Τεχνητή Νοημοσύνη κατάφερε να πετύχει αρκετά ορόσημα:

- Το 1997, το Deep Blue της Διεθνής Εταιρίας Μηχανών Επιχειρήσεων (International Business Machines Corporation – IBM), το οποίο αναπτύχθηκε ως ένα Έμπειρο Σύστημα με σκοπό να παίζει σκάκι, κατάφερε να νικήσει τον παγκόσμιο πρωταθλητή Γκάρυ Κασπάροφ (Garry Kasparov), αφού είχε χάσει την προηγούμενη χρονιά με τελικό σκορ 4-2.
- Το 1997, λογισμικό αναγνώρισης γλώσσας ανεπτυγμένο από την Dragon Systems ενσωματώθηκε στο λογισμικό των Windows της Microsoft, συμβάλλοντας στη γενικότερη προσπάθεια διερμηνείας της γλώσσας.
- Το 1998, ο Γιαν ΛεΚουν, μεταξύ άλλων, έκαναν δημοσίευση σχετικά με την εφαρμογή των νευρωνικών δικτύων για την αναγνώριση χειρογράφων και τη βελτιστοποίηση του αλγορίθμου οπισθοδιάδοσης.
- Το 2000 στο M.I.T. η Σύνθια Μπρέζιλ (Cynthia Breazeal) παρουσιάζει το Κισμέτ (Kismet), ένα ρομπότ με τη δυνατότητα να αναγνωρίζει και να μιμείται συναισθήματα.
- Το 2002, η πρώτης γενιάς ρομποτική σκούπα Roomba γίνεται η πρώτη εμπορικά επιτυχημένη συσκευή καθαρισμού.
- Το 2005, σε αγώνα 212 χιλιομέτρων για αυτόματα οχήματα στην έρημο Μοχάβε που διοργάνωσε η DARPA, το Stanley του Πανεπιστήμιου Στάνφορντ πήρε την πρώτη θέση μετά από 6 ώρες και 45 λεπτά, ενώ σε περσινό διαγωνισμό κανένα όχημα δεν μπόρεσε να τερματίσει.
- Το 2006, επιχειρήσεις όπως το Facebook, Twitter και Netflix, άρχισαν να χρησιμοποιούν Τεχνητή Νοημοσύνη στα επιχειρηματικά μοντέλα τους.
- Το 2010, είναι η πρώτη χρονιά του διαγωνισμού ImageNet (ImageNet Large Scale Visual Recognition Challenge – ILSVRC), όπου διαγωνίζονται υπολογιστικά συστήματα στην ταξινόμηση πάνω από 14 εκατομμύρια εικόνων σε 20.000 κατηγορίες.
- Το 2011, το υπολογιστικό σύστημα Watson της IBM κέρδισε στο τηλεοπτικό παιχνίδι «Jeopardy!», όπου έπρεπε να απαντήσει σε περίπλοκες ερωτήσεις και γρίφους, αποδεικνύοντας ότι μπορούσε να καταλάβει τη φυσική γλώσσα, χρησιμοποιώντας πάνω από 100 διαφορετικές τεχνικές λογικής και ανάλυσης.
- Το 2015, το AlphaGo της Google έγινε το πρώτο πρόγραμμα το οποίο μπόρεσε να νικήσει τον παγκόσμιο πρωταθλητή του επιτραπέζιου Go.

- Το 2018, η Google παρουσιάζει το BERT (Bidirectional Encoder Representations from Transformers), μια οικογένεια μοντέλων γλώσσας η οποία παρουσίασε πρωτοφανή επίδοση σε προβλήματα Επεξεργασίας Φυσικής Γλώσσας.

Πλέον, στον 21ο αιώνα, μετά από δυο «χειμώνες», ο τομέας της Τεχνητής Νοημοσύνης έχει να δείξει μια επαναστατική πρόοδο, η οποία έγινε δυνατή λόγω της τεράστιας διαθέσιμης υπολογιστής ισχύος των επεξεργαστών και των καρτών γραφικών, της εύκολης συλλογής μεγάλου όγκου δεδομένων, της εξέλιξης της προβλεπτικής στατιστικής ανάλυσης και της ανάπτυξης πιο προσιτών αλγοριθμικών εργαλειοθηκών, τραβώντας και πάλι το ενδιαφέρον αλλά και τη στήριξη επενδυτών.

Σήμερα, συστήματα και προγράμματα Τεχνητής Νοημοσύνης είναι ενσωματωμένα σε κινητά τηλέφωνα, υπό την μορφή εικονικών βοηθών (virtual assistant) που βασίζονται στην αναγνώριση γλώσσας, όπως η Alexa της Amazon, η Cortana της Microsoft και η Siri της Apple, αλλά και σε άλλες εφαρμογές, όπως για παράδειγμα για την βελτιστοποίηση και προστασία της κατάστασης της μπαταρίας ή των πόρων του κινητού. Επίσης, αυτόνομα αυτοκίνητα είναι πλέον διαθέσιμα προς πώληση στην αγορά με ακόμα πιο βελτιωμένους αλγόριθμους και ενσωματωμένα συστήματα ασφάλειας και πλοήγησης. Επιπλέον, συνεισφορές από την OpenAI για την περίπτωση Επεξεργασίας Φυσικής Γλώσσας αποτελούν το πρόγραμμα DALL-E, το οποίο δημιουργεί εικόνες βάσει κειμένων που θα διαβάσει, και το ChatGPT (Chat Generative Pre-Trained Transformer), η τελευταία έκδοση του οποίου ανακοινώθηκε το Νοέμβριου του 2022, και πρόκειται για ένα chatbot με τη δυνατότητα να παράγει κείμενο κάθε μορφής (συνομιλία, κώδικα προγραμματισμού, ποιήματα, εκθέσεις, σενάρια) και γενικότερα να μιμείται έναν ανθρώπινο συνομιλητή.

Το 1970, ο Μάρβιν Μίνσκυ έκανε την πρόβλεψη πως το πολύ σε οχτώ χρόνια θα υπάρχει μηχανή με διάνοια αντίστοιχη ενός μέσου ανθρώπου. Ωστόσο, ακόμα και σήμερα, μετά από τόσες μεγάλες εξελίξεις [62], ένα τέτοιο επίτευγμα φαντάζει ακόμη άπιαστο και πρακτικά αδύνατο να προβλέψουμε πότε στο μέλλον μπορεί να παρουσιαστεί. Τα αποτελέσματα που βλέπουμε αυτή τη στιγμή είναι εντυπωσιακά, ενώ η έρευνα για πιο εξελιγμένα συστήματα και για την πρόοδο του πεδίου συνεχίζεται με νέες δημοσιεύσεις να εμφανίζονται συνεχώς. Η συγκεκριμένη τεχνολογία είναι απλά εργαλείο για την καθημερινότητα, και όπως έδειξε η ιστορία, η βραχυπρόθεσμη υπερεκτίμηση και έντονη αισιοδοξία μπορούν κάλλιστα να οδηγήσουν σε απογοήτευση όταν τελικά έρθει μια αναπόφευκτη στασιμότητα. Η Τεχνητή Νοημοσύνη δεν έχει έρθει ακόμα στο προσκήνιο της δουλειάς, της σκέψης και της καθημερινής ζωής, αλλά υπάρχει το μακροπρόθεσμο όραμα για το πώς μπορεί να ωφελήσει τη ζωή μας [63].

3.2 Κλάδοι της Τεχνητής Νοημοσύνης

Είναι προφανές ότι η Τεχνητή Νοημοσύνη αποτελεί πλέον ένα πολυσύνθετο πεδίο με εφαρμογές σε πολλές πτυχές της καθημερινότητας. Μπορεί όμως να αναλυθεί σε ορισμένες κατηγορίες και έννοιες ανάλογα με τις δυνατότητες, τις τεχνικές, τις εφαρμογές και την εξέλιξη που παρουσιάζει [64].

3.2.1 Τα Στάδια της Τεχνητής Νοημοσύνης

Ένας πρώτος διαχωρισμός που μπορεί να γίνει αφορά κάθε πιθανή δυνατότητα εξέλιξης και ικανοτήτων που μπορεί να παρουσιάσει ένα σύστημα Τεχνητής Νοημοσύνης, και είναι ο εξής [65], [66]:

- Περιορισμένη Τεχνητή Νοημοσύνη (Artificial Narrow Intelligence – ANI)
- Γενική Τεχνητή Νοημοσύνη (Artificial General Intelligence – AGI)
- Ανώτερη Τεχνητή Νοημοσύνη (Artificial Super Intelligence – ASI)

ΠΕΡΙΟΡΙΣΜΕΝΗ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Το πρώτο στάδιο της Περιορισμένης Τεχνητής Νοημοσύνης, που συχνά αναφέρεται και ως Αδύναμη Τεχνητή Νοημοσύνη (Weak AI), είναι το μόνο που έχουμε καταφέρει μέχρι σήμερα και είναι ήδη ενσωματωμένο στην καθημερινότητά μας. Το στάδιο αυτό περιγράφει όλα τα συστήματα που μπορούν να εκτελέσουν μόνο μια προκαθορισμένη λειτουργία, για την οποία έχουν προγραμματιστεί και εκπαιδευτεί κατάλληλα, με συγκεκριμένους και σύνθετους αλγορίθμους, αλλά και με αρκετά δεδομένα, αντιπροσωπευτικά της εργασίας για την οποία τελικά αναπτύσσονται. Γενικότερα, είναι αρκετά αποτελεσματικά, διαπρέπουν σε περιπτώσεις επαναληπτικών εργασιών και μάλιστα ξεπερνούν τις ανθρώπινες επιδόσεις σε συγκεκριμένες καταστάσεις κάτω από ελεγχόμενες συνθήκες.

Παραδείγματα τέτοιων συστημάτων είναι η τεχνολογία αναγνώρισης προσώπου, η οποία πλέον βρίσκεται σχεδόν σε όλα τα smartphones για το ξεκλείδωμα του κινητού, οι εικονικοί βοηθοί (Siri, Alexa, κ.α.) με δυνατότητες αναγνώρισης γλώσσας και ομιλίας, οι μηχανές αναζήτησης, το φιλτράρισμα κακόβουλων ηλεκτρονικών μηνυμάτων (spam), τα συστήματα υποβοήθησης στα αυτοκίνητα, αλλά και τα πλήρως αυτόνομα αυτοκίνητα. Επιπλέον, τακτική του μάρκετινγκ εταιριών είναι οι συστάσεις βάσει προηγούμενων αγορών, αλλά και

τα κατάλληλα προτεινόμενα βίντεο, ταινίες, σειρές, βιβλία, βιντεοπαιχνίδια σε ιστοσελίδες ψυχαγωγίας βάσει ιστορικού. Επίσης, στον τομέα της ιατρικής υπάρχουν προγράμματα ικανά για διάγνωση ασθενών, ενώ στον τομέα της βιομηχανίας αναπτύσσονται συστήματα τόσο ρομποτικά όσο και για πρόβλεψη βλαβών.

Τα παραπάνω συστήματα μπορεί να φαίνονται έξυπνα, αλλά οι δυνατότητές τους περιορίζονται μόνο σε συγκεκριμένες λειτουργίες για τις οποίες έχουν προγραμματιστεί. Δεν μιμούνται ούτε αντιγράφουν την ανθρώπινη διάνοια, απλά προσομοιώνουν την ανθρώπινη συμπεριφορά βάσει ενός περιορισμένου εύρους παραμέτρων. Για παράδειγμα, ένα σύστημα εκπαιδευμένο να παίζει σκάκι με τέλεια ακρίβεια δεν θα μπορούσε να κάνει κάτι άλλο πέρα από αυτό, δηλαδή δεν θα μπορούσε να παίξει κάποιο άλλο επιτραπέζιο, αλλά ούτε και να κατανοήσει την εννοιολογική διαφορά μεταξύ «ίππου» και «αξιωματικού».

ΓΕΝΙΚΗ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Η Γενική Τεχνητή Νοημοσύνη, ή αλλιώς Ισχυρή Τεχνητή Νοημοσύνη (Strong AI), αναφέρεται στο στάδιο εξέλιξης όπου οι μηχανές και τα συστήματα θα έχουν πλέον την ικανότητα να σκέφτονται και να λαμβάνουν αποφάσεις όπως οι άνθρωποι, ακόμα και σε περιπτώσεις προβλημάτων τα οποία δεν τα έχουν ξανασυναντήσει. Αυτήν τη στιγμή δεν υπάρχουν παραδείγματα αυτής της περίπτωσης, αλλά υποστηρίζεται ότι, τουλάχιστον σε αυτόν τον αιώνα, είναι κάτι το οποίο τελικά θα μπορέσει να υλοποιηθεί. Ωστόσο, για να επιτευχθεί η δημιουργία ενός τέτοιου συστήματος, οι επιστήμονες θα πρέπει να βρουν έναν τρόπο να προσδώσουν συνείδηση στις μηχανές προγραμματίζοντας ένα ακριβές και πλήρες σύνολο γνωστικών ικανοτήτων.

Θεωρητικά, η πραγματική Γενική Τεχνητή Νοημοσύνη αναφέρεται σε ένα σύστημα που θα μπορεί να μαθαίνει, να παρατηρεί, να κατανοεί και να λειτουργεί με τον ίδιο τρόπο όπως ο άνθρωπος, χωρίς να προσομοιώνει την ανθρώπινη συμπεριφορά, όπως στην περίπτωση της Περιορισμένης Τεχνητής Νοημοσύνης. Τέτοιες μηχανές θα μπορούν να εκτελούν πληθώρα εργασιών, αλλά και να αυτοβελτιώνονται μέσω βιωματικής μάθησης, αποδίδοντας ακόμα πιο αποτελεσματικά και ξεπερνώντας τον άνθρωπο σε ένα μεγάλο φάσμα διάφορων γνωστικών αντικειμένων, καθώς θα έχουν την δυνατότητα να έχουν πρόσβαση και να επεξεργάζονται τεράστιο πλήθος δεδομένων με απίστευτες ταχύτητες.

Τελικά, ένα τέτοιο σύστημα θα απαιτούσε χιλιάδες υποσυστήματα Περιορισμένης Τεχνητής Νοημοσύνης ικανά να συνεργάζονται και να αλληλεπιδρούν μεταξύ τους για την επίτευξη

της διάνοιας, και μάλιστα τεράστια υπολογιστική ισχύ. Είναι προφανώς μια πολύ μεγάλη πρόκληση, ειδικά αν αναλογιστούμε ότι αποτελεί προσπάθεια μοντελοποίησης του βιολογικού εγκεφάλου. Αξίζει να αναφερθεί ότι έχουν γίνει προσπάθειες προσομοίωσης της νευρικής δραστηριότητας του εγκεφάλου, και σε μια περίπτωση υπολογίστηκε ότι για να επιτευχθεί ένα λεπτό προσομοίωσης της παρεγκεφαλικής δραστηριότητας από έναν υπερυπολογιστή θα χρειαζόταν δέκα ώρες εκτέλεσης [67]. Τελικά, ακόμα και τα δυνατότερα συστήματα αυτής της εποχής υστερούν, και η ανάπτυξη ενός προγράμματος Γενικής Τεχνητής Νοημοσύνης έχει ακόμα πολλά εμπόδια και περιορισμούς να ξεπεράσει μέχρι να παρουσιαστεί κάποια σημαντική πρόοδος.

ΑΝΩΤΕΡΗ ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

Η Ανώτερη Τεχνητή Νοημοσύνη αναφέρεται στο τελικό στάδιο εξέλιξης των μηχανών όπου οι υπολογιστές δεν θα μιμούνται μόνο, ούτε απλά θα καταλαβαίνουν την ανθρώπινη σκέψη και συμπεριφορά, αλλά θα αποκτήσουν πλήρη συνείδηση και αυτογνωσία και θα υπερβούν τις γνωστικές δυνατότητες των ανθρώπων σε όλους τους τομείς.

Προς το παρόν, ένα τέτοιο σύστημα δεν υπάρχει και πρόκειται για μια θεωρητική κατάσταση η οποία χαρακτηρίζεται από ανώτερες δυνατότητες γνώσης, διάνοιας, νοημοσύνης, κοινωνικές δεξιότητες, ικανότητα επίλυσης προβλημάτων, λήψης αποφάσεων και δημιουργικότητας, και η οποία δεν περιορίζεται από τα χημικά και βιολογικά όρια του ανθρώπινου εγκέφαλου. Επίσης, με αυτήν την πολυεπίπεδη ευφυΐα, μια Ανώτερη Τεχνητή Νοημοσύνη θα μπορεί να ξεπεράσει στο έπακρο τον άνθρωπο σε κλάδους μαθηματικών, επιστημών, τέχνης, ιατρικής, ψυχαγωγίας, συναισθημάτων κ.α., αλλά και να αυτομορφωθεί, να αυτοβελτιωθεί και να εφαρμόσει αυτή τη βιωματική γνώση στα παραπάνω πεδία πετυχαίνοντας ακόμη και πρωτοφανή πρόοδο (π.χ. ένα νέο θεώρημα μαθηματικών). Φυσικά, για να συμβεί αυτό θεωρείται απαραίτητη η τεχνολογική πρόοδος, η οποία θα επιτρέψει σε ένα τέτοιο σύστημα να έχει πολύ μεγαλύτερα αποθέματα μνήμης για αποθήκευση δεδομένων, και υψηλές υπολογιστικές δυνατότητες για την επεξεργασία τους. Επομένως, για να ξεκινήσουν οι προσπάθειες της Ανώτερης Τεχνητής Νοημοσύνης, προαπαιτούμενο θα ήταν κάποια εξέλιξη, ή ακόμα και εμπειρογνωμοσύνη, πάνω στην Γενική Τεχνητή Νοημοσύνη.

Τελικά, είμαστε πολύ μακριά από ένα τέτοιο σύστημα, αλλά πρόκειται για κάτι το οποίο πλέον θεωρείται αναπόφευκτο, όσα χρόνια κι αν χρειαστεί μέχρι να επιτευχθεί. Επιπλέον, μαζί με όλα τα πλεονεκτήματα και την ανάπτυξη που μπορεί να προσφέρει, πολλοί ειδικοί

πιστεύουν ότι αποτελεί ένα ρίσκο για την ανθρωπότητα και ότι μπορεί να οδηγήσει σε παγκόσμια καταστροφή, καθώς ένα σύστημα ικανό να αυτοβελτιώνεται μπορεί να αναπτυχθεί σε βαθμό που δεν θα μπορούσε να φανταστεί ο ανθρώπινος νους, όπως για παράδειγμα να αποκτήσει την αίσθηση της αυτοσυντήρησης. Επίσης, ήδη αναπτύσσονται επιτυχώς προγράμματα πρόβλεψης, επομένως είναι λογικό ένα σύστημα Ανώτερης Τεχνητής Νοημοσύνης να έχει εξελιγμένες δυνατότητες πρόγνωσης με οτιδήποτε αυτό μπορεί να συνεπάγεται. Ακόμη και ο ίδιος ο Στήβεν Χώκινγκ (Stephen Hawking) ανέφερε ότι: «*Η ανάπτυξη μιας πλήρους τεχνητής νοημοσύνης θα μπορούσε να σημάνει το τέλος της ανθρώπινης φυλής. Μόλις οι άνθρωποι αναπτύξουν τεχνητή νοημοσύνη, αυτή θα παρέκλινε και θα επανασχέδιαζε τον εαυτό της με διαρκώς αυξανόμενο ρυθμό. Οι άνθρωποι, που περιορίζονται από την αργή βιολογική εξέλιξη, δεν θα μπορούσαν να ανταγωνιστούν, και θα αντικατασταινόταν.*» [68]

Σήμερα, αυτή η δυστοπική προοπτική είναι μόνο θεωρητική, αλλά ήταν και παραμένει ιδιαίτερα έντονη που έδωσε έμπνευση για πολλές επιτυχημένες ταινίες, λογοτεχνικά βιβλία και παιχνίδια επιστημονικής φαντασίας, όπως «Εγώ, το Ρομπότ» (1950), συλλογή του συγγραφέα και καθηγητή Ισαάκ Άσιμοφ (Isaac Asimov), καθώς και η ομώνυμη ταινία του 2004, «Westworld» (1973, 2016), «2001: Η Οδύσσεια του Διαστήματος» (1968), «Ο Εξολοθρευτής» (1984), κ.α.

3.2.2 Οι Τύποι της Τεχνητής Νοημοσύνης

Ένας ακόμη διαχωρισμός μεταξύ των συστημάτων Τεχνητής Νοημοσύνης γίνεται βάσει της λειτουργικότητάς τους, και είναι ο εξής [69], [70]:

- Αντιδραστικές Μηχανές (Reactive Machines)
- Περιορισμένης Μνήμης (Limited Memory)
- Θεωρία του Νου (Theory of Mind)
- Αυτογνωσία (Self-awareness)

ΑΝΤΙΔΡΑΣΤΙΚΕΣ ΜΗΧΑΝΕΣ

Οι Αντιδραστικές Μηχανές είναι ο πιο συμβατικός τύπος Τεχνητής Νοημοσύνης, και είναι προγραμματισμένες να ανταποκρίνονται σε τρέχουσες καταστάσεις, εκτελώντας ένα περιορισμένο εύρος προκαθορισμένων και εξειδικευμένων ενεργειών ως αντίδραση σε οτιδήποτε αντιλαμβάνονται. Πρόκειται για συστήματα τα οποία δεν έχουν τη δυνατότητα

μνήμης, δηλαδή της αποθήκευσης δεδομένων, και έτσι δεν μπορούν να βασιστούν σε προηγούμενες εμπειρίες για τη λήψη αποφάσεων και την εξαγωγή συμπερασμάτων σε πραγματικό χρόνο, δηλαδή δεν μπορούν να θυμούνται το παρελθόν ή να προβλέπουν το μέλλον. Αυτό σημαίνει ότι είναι σχεδιασμένα να αντιδρούν πάντα κατά τον ίδιο τρόπο κάθε φορά που αντιμετωπίζουν μια ίδια κατάσταση ή συγκεκριμένα ερεθίσματα, κάτι που τα κάνει ιδιαίτερα χρήσιμα και αξιόπιστα, ωστόσο, χωρίς τη δυνατότητα να «μαθαίνουν».

Ένα αντιπροσωπευτικό παράδειγμα Αντιδραστικής Μηχανής είναι το Deep Blue της δεκαετίας του ’90, ο υπερυπολογιστής που έπαιζε σκάκι και κατάφερε να νικήσει τον Γκάρυ Κασπάροφ. Το συγκεκριμένο σύστημα μπορούσε μόνο να αναγνωρίσει τα πιόνια στην σκακιέρα, το πώς κινούνταν το καθένα και να επιλέξει την πιο βέλτιστη και λογική κίνηση μεταξύ πολλών πιθανών κινήσεων. Κάθε κίνησή του βασιζόταν στην κατάσταση που έβλεπε κάθε φορά μπροστά του, σε μια συγκεκριμένη χρονική στιγμή, αγνοώντας το τι προηγήθηκε, και ποια θα μπορούσε να είναι η επόμενη κίνηση του αντιπάλου μετά από μια δική του. Ακόμα ένα παράδειγμα, ωστόσο πιο σοφιστικό στο τρόπο που αξιολογούσε τις εξελίξεις των παιχνιδιών, αποτελεί το πρόγραμμα AlphaGo. Ωστόσο, ακόμα και στην περίπτωση μιας πιο πολύπλοκης Αντιδραστικής Μηχανής, παρά την αξιοπιστία της, παραμένει το γεγονός ότι περιορίζεται στο να εκτελεί μια συγκεκριμένη λειτουργία που θα ήταν δύσκολο να εφαρμοστεί σε άλλες καταστάσεις.

ΠΕΡΙΟΡΙΣΜΕΝΗ ΜΝΗΜΗ

Τα συστήματα Τεχνητής Νοημοσύνης τα οποία βασίζονται στην περίπτωση Περιορισμένης Μνήμης έχουν την ικανότητα να διατηρούν αποθηκευμένη μια μικρή ποσότητα δεδομένων για ένα μικρό χρονικό διάστημα, έτσι ώστε να τα επεξεργαστούν κατάλληλα και να προχωρήσουν σε πιο ενημερωμένες και βελτιωμένες αποφάσεις, ουσιαστικά λαμβάνοντας υπόψη μια παρελθοντική κατάσταση για την αξιολόγηση μιας μελλοντικής. Επιπλέον, αυτού του τύπου τα συστήματα μαθαίνουν από ιστορικά δεδομένα, και αφού εκπαιδευτούν σε αυτά, αναγνωρίζοντας τελικά τα υποκείμενα μοτίβα που ακολουθούν, μπορούν να τροφοδοτηθούν με νέα δεδομένα και βάσει των όσων έχουν «μάθει» να κάνουν κάποια πρόβλεψη ή να βγάλουν κάποιο συμπέρασμα. Η κατάσταση Περιορισμένης Μνήμης είναι πιο περίπλοκη από την Αντιδραστική Μηχανή, παρουσιάζει καλύτερες δυνατότητες και προοπτικές εξέλιξης, και σχεδόν κάθε εφαρμογή Τεχνητής Νοημοσύνης που αναπτύσσεται αυτήν την περίοδο βρίσκεται υπό αυτήν την κατηγορία. Αξίζει επίσης να σημειωθεί ότι και οι δυο περιπτώσεις ανήκουν στο στάδιο εξέλιξης της Περιορισμένης Τεχνητής Νοημοσύνης.

Τα αυτόνομα αυτοκίνητα και τα ενσωματωμένα συστήματα υποβοήθησης στα νεότερα αμάξια αποτελούν παράδειγμα Περιορισμένης Μνήμης. Τέτοια αυτοκίνητα είναι εξοπλισμένα με κατάλληλους αισθητήρες που καταγράφουν σε πραγματικό χρόνο το γύρω περιβάλλον, όπως πεζούς που μπορεί να διασχίζουν τον δρόμο, την ταχύτητα, την τροχιά και την απόσταση άλλων κοντινών αυτοκινήτων, όρια ταχύτητας, πινακίδες, φανάρια και γραμμές δρόμων, και με αυτά τα προσωρινά δεδομένα βασίζονται στην εκπαίδευσή τους λαμβάνοντας αποφάσεις για καλύτερη οδική πλοήγηση και αποφυγή ατυχημάτων. Επίσης, ένα πρόγραμμα Τεχνητής Νοημοσύνης για την αναγνώρισης εικόνων, εφόσον έχει εκπαιδευτεί σε μια πληθώρα εικόνων τις οποίες έχει μάθει να τις ταξινομεί κατάλληλα σε κατηγορίες, μπορεί να χρησιμοποιήσει αυτή την αποθηκευμένη γνώση προκειμένου να κατατάξει κάθε άλλη σχετική εικόνα που μπορεί να του τροφοδοτηθεί.

Φυσικά, τέτοια συστήματα περιορίζονται στη γνώση που τους έχει προσδοθεί από τον δημιουργό τους. Για παράδειγμα, ένα πρόγραμμα που έχει σαν κύρια λειτουργία του την ταξινόμηση φωτογραφιών σκύλων, αν δεχόταν μια εικόνα από κάποιο άλλο είδος ζώου, θα λάμβανε τελικά μια απόφαση για την κατάταξη της φωτογραφίας, αλλά προφανώς αυτή η τελική απόφαση θα ήταν λανθασμένη. Επομένως, κάθε νέο δεδομένο που δέχονται αυτά τα συστήματα είναι παροδικό και δεν σώζεται ως «εμπειρία» πάνω στην οποία μπορεί να βασιστεί αργότερα το συγκεκριμένο σύστημα για να κάνει πιο ακριβείς προβλέψεις. Απαιτείται λοιπόν εκ νέου διαδικασία μάθησης, πάνω σε κάποιο νέο και πιο σύνθετο σύνολο δεδομένων το οποίο θα λαμβάνει υπόψη πιο καινούριες και αντιπροσωπευτικές καταστάσεις του προβλήματος που καλείται να αντιμετωπίσει.

ΘΕΩΡΙΑ ΤΟΥ ΝΟΥ

Στην επιστήμη της ψυχολογίας, ο όρος της Θεωρίας του Νου αναφέρεται στην κατανόηση της ανθρώπινης ψυχικής κατάστασης, και πιο συγκεκριμένα στο ότι οι άνθρωποι, αλλά και γενικότερα τα έμβια όντα, έχουν σκέψεις, πεποιθήσεις, προθέσεις, προσδοκίες και συναισθήματα που επηρεάζουν τον ίδιο τους τον εαυτό, αλλά και τους γύρω τους, καθώς και τον τρόπο που αλληλεπιδρούν μεταξύ τους. Ένα σύστημα Τεχνητής Νοημοσύνης βασισμένο στην έννοια της Θεωρίας του Νου αναμένεται να έχει την κοινωνική και συναισθηματική ευφυΐα να κατανοεί την ανθρώπινη σκέψη και συμπεριφορά, αλλά και τη σημασία των συναισθημάτων στη λήψη αποφάσεων, έτσι ώστε τελικά να προσαρμόζει τη συμπεριφορά του, να προβλέπει καταστάσεις και να ανταποκρίνεται κατάλληλα.

Οι άνθρωποι σχηματίζουν κοινωνίες και σχέσεις μεταξύ τους ακριβώς επειδή υπάρχει αυτή η αμοιβαία κατανόηση αναγκών, επομένως ένα τέτοιο σύστημα ανάμεσα σε ανθρώπους θα πρέπει να μπορεί να επιδεικνύει κατανόηση προσδοκιών και να φέρεται αναλόγως. Αυτή τη στιγμή, εικονικοί βοηθοί ερμηνεύουν αιτήματα χρηστών και απαντούν λαμβάνοντας υπόψη τις προθέσεις τους, ενώ έχουν σχεδιαστεί ρομπότ (Kismet, Sophia) που μπορούν να ανταποκρίνονται σε συναισθήματα. Ωστόσο, τα παραπάνω παραδείγματα αποτελούν προσπάθειες προσομοίωσης των ανθρώπινων σχέσεων, και μέχρι τώρα δεν έχει δημιουργεί επιτυχώς ένα σύστημα βασισμένο στην πραγματική Θεωρία του Νου με δικό του «μυαλό» και αντίληψη. Επίσης, η διαδικασία μάθησης και εκπαίδευσης ενός τέτοιου συστήματος φαντάζει ιδιαίτερα περίπλοκη, καθώς η ανθρώπινη συναισθηματική απεικόνιση είναι πολυεπίπεδη και μπορεί να χαρακτηριστεί από εκφράσεις του προσώπου, γλώσσα του σώματος, βιωματικές εμπειρίες, τόνο της φωνής, ή ακόμα και να είναι τόσο διακριτική που πολλές φορές ακόμα και ο ίδιος ο άνθρωπος να μην μπορεί να την αναγνωρίσει. Τελικά, η Θεωρία του Νου παραμένει σε αυτό το θεωρητικό επίπεδο και ένα τέτοιο σύστημα συμβαδίζει με το εξελικτικό στάδιο της Γενικής Τεχνητής Νοημοσύνης, μέχρι η τεχνολογική και επιστημονική πρόοδος επιτρέψει την ανάπτυξή του, η οποία ήδη συζητείται και ερευνάται.

ΑΥΤΟΓΝΩΣΙΑ

Από τη στιγμή που θα καταφέρει να επιτευχθεί η Θεωρία του Νου, το επόμενο βήμα θα είναι η ανάπτυξη ενός συστήματος Τεχνητής Νοημοσύνης με την προοπτική της Αυτογνωσίας. Ένα τέτοιο σύστημα θα είναι ικανό να έχει επίγνωση, όχι μόνο των συναισθημάτων και της ψυχικής κατάστασης των ανθρώπων, αλλά και του ίδιου του εαυτού του. Επιπλέον, θα έχει ξεφύγει από τον ανθρώπινο έλεγχο καθώς θα έχει αποκτήσει πλήρη κατανόηση της ύπαρξής του και των πράξεών του, αίσθημα αυτό-συντήρησης, θα έχει την ικανότητα να προβλέπει τις ανάγκες και απαιτήσεις του, αλλά και να επικοινωνεί και να εκφράζει συναισθήματα, προσδοκίες, σκέψεις και επιθυμίες με συνείδηση αντίστοιχη του ανθρώπου. Προφανώς, σε μια τέτοια περίπτωση η Αυτογνωσία των μηχανών αναφέρεται στο στάδιο εξέλιξης της Ανώτερης Τεχνητής Νοημοσύνης, το οποίο προϋποθέτει σημαντικές εξελίξεις τεχνολογικού υλικού και αλγορίθμων, αλλά και εμπειρογνωμοσύνη από τους ερευνητές, τόσο για την έννοια της συνείδησης, όσο και για τις διαδικασίες μάθησης και λήψης αποφάσεων που εφαρμόζονται στο πεδίο της Τεχνητής Νοημοσύνης αυτής της εποχής, πόσο μάλλον όταν ακόμα υπάρχουν πολλές ακόμα πτυχές

του ανθρώπινου εγκεφάλου για να ανακαλυφθούν σχετικά με τις λειτουργίες μνήμης, μάθησης και συμπερασματολογίας.

3.2.3 Τα Υποπεδία της Τεχνητής Νοημοσύνης

Η Τεχνητή Νοημοσύνη είναι ένα ευρύ πεδίο της Επιστήμης των Υπολογιστών (Computer Science) το οποίο έχει ως κύριο αντικείμενο την ανάπτυξη έξυπνων μηχανών, και που συνεχώς εξελίσσεται και επεκτείνεται. Πλέον, μπορεί να διαχωριστεί σε κλάδους και υποπεδία, όπου καθένα διακρίνεται βάσει των τεχνικών και των διαδικασιών που εφαρμόζονται σε διάφορες πτυχές της ανάπτυξης ευφυών συστημάτων και εφαρμογών. Ωστόσο, αυτός ο διαχωρισμός δεν είναι απόλυτος καθώς οι συγκεκριμένοι κλάδοι, λόγω της πολυσύνθετης φύσης τους, μπορεί να επικαλύπτονται και πολλές φορές να συγχέονται, αλλά η κατηγοριοποίησή τους επιτρέπει στην καλύτερη κατανόηση των πρακτικών Τεχνητής Νοημοσύνης. Κάποια από τα πιο γνωστά και δημοφιλή υποπεδία του τομέα είναι τα εξής [71], [72]:

- Μηχανική Μάθηση (Machine Learning)
- Βαθιά Μάθηση (Deep Learning)
- Επεξεργασία Φυσικής Γλώσσας (Natural Language Processing)
- Επεξεργασία Ομιλίας (Speech Processing)
- Υπολογιστική Όραση (Computer Vision)
- Ρομποτική (Robotics)
- Έμπειρα Συστήματα (Expert Systems)
- Αναπαράσταση Γνώσης και Λογική (Knowledge Representation and Reasoning)
- Σχεδιασμός και Προγραμματισμός (Planning and Scheduling)

ΜΗΧΑΝΙΚΗ ΜΑΘΗΣΗ

Το υποπεδίο της Μηχανικής Μάθησης έχει ως αντικείμενο την ανάπτυξη αλγορίθμων οι οποίοι είναι ικανοί να μάθουν από δεδομένα. Ένας αλγόριθμος μάθησης επεξεργάζεται τα ιστορικά δεδομένα, μαθαίνει τους στατιστικούς κανόνες στους οποίους υπακούουν και τελικά κάνει προβλέψεις ή παίρνει αποφάσεις βάσει αυτών των δεδομένων. Γενικότερα, η Μηχανική Μάθηση βασίζεται κυρίως στην στατιστική ανάλυση και στα στατιστικά μοντέλα πιθανοτήτων για την εξαγωγή συμπερασμάτων.

ΒΑΘΙΑ ΜΑΘΗΣΗ

Η Βαθιά Μάθηση αποτελεί υποπεδίο της Μηχανικής Μάθησης, αλλά μπορεί να σταθεί και ως μεμονωμένος κλάδος, ο οποίος συχνά αναφέρεται και ως κλάδος των Νευρωνικών Δικτύων (Neural Networks), καθώς κύριο αντικείμενό του είναι η ανάπτυξη των Νευρωνικών Δικτύων. Το Τεχνητό Νευρωνικό Δίκτυο (Artificial Neural Network) είναι ιδιαίτερα δημοφιλής τεχνική Μηχανικής Μάθησης για πιο πολύπλοκα προβλήματα και η γενικότερη αρχιτεκτονική του είναι εμπνευσμένη από τη δομή του βιολογικού εγκεφάλου. Η εκπαίδευση σε αυτήν την περίπτωση γίνεται με μια κατάλληλη επαναληπτική επεξεργασία δεδομένων από διασυνδεδεμένους υπολογιστικούς νευρώνες που είναι δομημένοι σε διαδοχικά στρώματα.

ΕΠΕΞΕΡΓΑΣΙΑ ΦΥΣΙΚΗΣ ΓΛΩΣΣΑΣ

Η Επεξεργασία Φυσικής Γλώσσας στοχεύει στην επικοινωνία μεταξύ ανθρώπου και μηχανής, δίνοντας την ικανότητα στη μηχανή να διαβάζει, να καταλαβαίνει και να χρησιμοποιεί τη γλώσσα. Οι αλγόριθμοι μάθησης σε αυτήν την περίπτωση αποσκοπούν στην εκπαίδευση συστημάτων πάνω σε τεράστιο όγκο κειμένων και στην αναγνώριση των μοτίβων που ακολουθεί η εκάστοτε γλώσσα, με σκοπό την ανάπτυξη εφαρμογών, όπως την κατηγοριοποίηση κειμένων (text classification), το φιλτράρισμα περιεχομένου (content filtering), τη μετάφραση (translation), την ανάλυση συναισθημάτων (sentiment analysis), την παραγωγή κειμένου (text generation) κ.α.

ΕΠΕΞΕΡΓΑΣΙΑ ΟΜΙΛΙΑΣ

Η Επεξεργασία Ομιλίας, αν και θεωρείται κομμάτι της Επεξεργασίας Φυσικής Γλώσσας, μπορεί να περιγραφεί ως το πεδίο που επικεντρώνεται στην ανάλυση της ομιλίας και της επεξεργασίας των ακουστικών σημάτων, με σκοπό την ανάπτυξη συστημάτων ικανών να αναγνωρίζουν διαλέκτους, τον τόνο της φωνής και τυχόν αλλαγές λόγω κάποιας κατάστασης υγείας, τον θόρυβο του γύρω περιβάλλοντος κ.α. Τελικά, τέτοια προγράμματα μπορούν να αξιοποιηθούν σε πληθώρα εφαρμογών, όπως την αναγνώριση και μεταγραφή ομιλίας (speech-to-text), ή αντίστροφα τη μετατροπή κειμένου σε φωνητικά σήματα (text-to-speech).

ΥΠΟΛΟΓΙΣΤΙΚΗ ΌΡΑΣΗ

Το υποπεδίο της Υπολογιστικής Όρασης ασχολείται με την ανάπτυξη τεχνικών και αλγορίθμων ικανών να αναλύουν και να κατανοούν οπτικές πληροφορίες, όπως εικόνες και βίντεο. Πρόκειται για έναν αρκετά ενεργό ερευνητικό τομέα με πολλές ενδιαφέρουσες εφαρμογές, όπως την ταξινόμηση εικόνων (image classification), την τμηματοποίηση εικόνων (image segmentation), την αναγνώριση αντικειμένων (object detection), την αποκατάσταση εικόνων (image restoration), καθώς και κάθε άλλη παράγωγη προσπάθεια των παραπάνω εργασιών.

ΡΟΜΠΟΤΙΚΗ

Η Ρομποτική αποτελεί ένα διεπιστημονικό πεδίο που ενσωματώνει Μηχανολογία, Ηλεκτρολογία, Τεχνολογία Πληροφοριών, Επιστήμη Υπολογιστών και άλλους τομείς για τον σχεδιασμό, τη δημιουργία και τη λειτουργία τεχνητών μέσων, ικανών να αλληλεπιδρούν με το περιβάλλον τους χωρίς ανθρώπινη υποστήριξη. Με την προσθήκη αλγορίθμων Μηχανικής και Βαθιάς Μάθησης τα ρομποτικά υπολογιστικά συστήματα επεξεργάζονται καλύτερα πληροφορίες που δέχονται από αισθητήρες και γίνονται ακόμα πιο αποδοτικά όσο αφορά στην αντίληψη, τον προγραμματισμό και τον έλεγχο κίνησης και τη γενικότερη συμπεριφορά τους.

ΈΜΠΕΙΡΑ ΣΥΣΤΗΜΑΤΑ

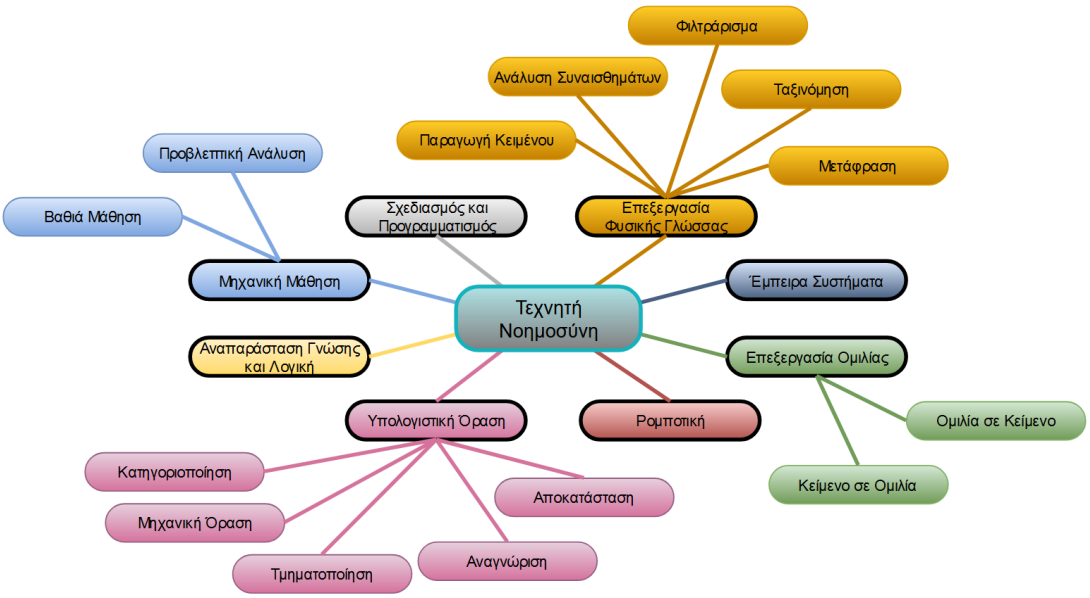
Το υποπεδίο των Έμπειρων Συστημάτων ασχολείται με την ανάπτυξη εξειδικευμένων υπολογιστικών προγραμμάτων ικανών να προσομοιώνουν την ανθρώπινη κρίση και συμπεριφορά ενός εμπειρογνώμονα σε ένα συγκεκριμένο πεδίο με σκοπό τη λήψη αποφάσεων, έχοντας τον ρόλο του συμπληρωματικού παράγοντα. Αποτελούνται από μια βάση γνώσεων (knowledge base) από όπου αντλούν πληροφορίες ακολουθώντας λογικούς κανόνες και διαδικασίες για την εξαγωγή συμπερασμάτων (inference engine). Η γραμμικότητά τους τα καθιστά αξιόπιστα και χρησιμοποιούνται από διάφορους τομείς σε εφαρμογές, όπως ιατρικές και ηλεκτρονικές διαγνώσεις, διαχείριση περιουσιακών στοιχείων, πρόβλεψη ζημιών σε καλλιέργειες, σχεδιασμός ευαίσθητων χημικών πειραμάτων κ.α.

ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΝΩΣΗΣ ΚΑΙ ΛΟΓΙΚΗ

Το υποπεδίο της Αναπαράστασης Γνώσης και Λογικής επικεντρώνεται στον τρόπο με τον οποίο μπορεί να επιτευχθεί η δημιουργία μιας βάσης γνώσεων, όπου το περιεχόμενο αναπαριστάται σε κατάλληλη μορφή και με τέτοιο τρόπο έτσι ώστε να μπορεί επιτυχώς να αναγνωρίζεται και να επεξεργάζεται από ένα σύστημα Τεχνητής Νοημοσύνης, με σκοπό να αξιοποιεί τελικά αυτή τη γνώση για να λύνει περίπλοκα προβλήματα πάσης φύσεως βάσει λογικών αλγορίθμων. Τέτοιοι τρόποι αναπαράστασης περιλαμβάνουν πληθώρα τεχνικών, όπως αυστηρούς λογικούς κανόνες, σημασιολογικά δίκτυα, δέντρα αποφάσεων, νευρωνικά δίκτυα, οντολογίες κ.α.

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ

Ο Σχεδιασμός και Προγραμματισμός αποτελεί ένα υποπεδίο που έχει ως κύριο αντικείμενο την ανάπτυξη αλγορίθμων με τη δυνατότητα να αναγνωρίζουν και να εκμεταλλεύονται τη δομή ενός προβλήματος, με σκοπό την εύρεση ενός πλάνου δράσης που ακολουθεί την πιο βέλτιστη σειρά ενεργειών για την επίλυσή του.



Εικόνα 2. Υποπεδία της Τεχνητής Νοημοσύνης

Κεφάλαιο 2

Μηχανική Μάθηση

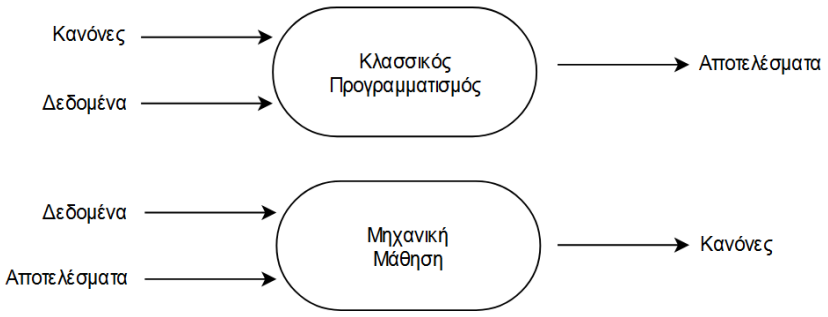
Θεωρητικό Υπόβαθρο

1 Βασικές Αρχές

Οι πρώτες προσπάθειες δημιουργίας σκεπτόμενων μηχανών ακολουθούσαν τακτικές συμβατικού προγραμματισμού, δηλαδή τα συστήματα βασίζονταν σε σαφείς και αυστηρά κωδικοποιημένους κανόνες για να βγάλουν λογικά συμπεράσματα. Επίσης, μια τέτοια διαδικασία θα ήταν ιδιαίτερα δύσκολη σε περιπτώσεις πιο πολύπλοκων προβλημάτων, ή σε περιπτώσεις όπου νέες συνθήκες θα εμφανίζονταν σε ένα ήδη ολοκληρωμένο σύστημα, όπου θα ήταν απαραίτητη η ανανέωση των κανόνων για να συμβιβαστεί με τα νέα δεδομένα και καταστάσεις.

Το πεδίο της Μηχανικής Μάθησης έχει απομακρυνθεί από αυτήν την τακτική και πλέον η ανάπτυξη των αλγορίθμων μάθησης γίνεται με τέτοιο τρόπο έτσι ώστε να μη βασίζονται στον προγραμματισμό των ίδιων των δεδομένων, καθώς και των κανόνων που πρέπει να ακολουθούν, για να πάρουν λογικές αποφάσεις. Αντίθετα, η Μηχανική Μάθηση επιτρέπει στα συστήματα να αποκτήσουν από μόνα τους τη δική τους γνώση για τα δεδομένα που θα τους τροφοδοτηθούν, μέσα από μια διαδικασία εκπαίδευσης.

Συγκεκριμένα, ένα τέτοιο εκπαιδευμένο σύστημα έχει αναλύσει τα δεδομένα, έχει αναγνωρίσει τους στατιστικούς κανόνες και τα υποκείμενα μοτίβα που συνδέονται με αυτά, και πλέον, έχοντας αποκτήσει τη δική του γνώση για τη δομή τους, έχει βρει τους κανόνες που ταιριάζουν καλύτερα στη περιγραφή και την αναπαράστασή τους. Τελικά, το εν λόγω σύστημα έχει αποκτήσει τη δυνατότητα να δέχεται νέα δεδομένα, και βάσει των κανόνων που έχει εξάγει μέσω της διαδικασίας μάθησης, να βγάζει πιο ακριβή συμπεράσματα και να κάνει καταλληλότερες προβλέψεις για την αυτοματοποίηση της εργασίας πάνω στην οποία έχει εκπαιδευτεί.



Εικόνα 3. Κλασσικός Προγραμματισμός και Μηχανική Μάθηση

Σημειώνεται ότι, η ανάλυση του παρακάτω θεωρητικού υπόβαθρου βασίστηκε στη βιβλιογραφία των [73], [74], [75], [76], [77], [78], και αποσκοπεί στην καλύτερη αντίληψη των βασικών αρχών των υποπεδίων της Μηχανικής Μάθησης, αλλά και των κατηγοριών της Βαθιάς Μάθησης που παρουσιάζονται σε επόμενο υποκεφάλαιο, έτσι ώστε να γίνουν όσο το δυνατόν πιο κατανοητές οι πρακτικές εφαρμογές του 3ου Κεφαλαίου.

1.1 Τύποι Προβλημάτων

Οι ικανότητες ενός εκπαιδευμένου συστήματος (ή μοντέλου) Μηχανικής Μάθησης, όσον αφορά στις προβλέψεις και τα συμπεράσματα που εξάγει, εξαρτώνται κάθε φορά από την εφαρμογή και την εργασία για την οποία προορίζεται και έχει εκπαιδευτεί. Οι εφαρμογές Μηχανικής Μάθησης αφορούν περισσότερο τον τρόπο με τον οποίο θα πρέπει ένας αλγόριθμος να επεξεργαστεί τα διαθέσιμα δεδομένα, και μπορούν να διακριθούν σε μια πληθώρα γενικότερων, αλλά και ειδικότερων κατηγοριών. Κάποιες από τις πιο συνηθισμένες εφαρμογές Μηχανικής Μάθησης είναι οι εξής:

- *Κατηγοριοποίηση (Classification)* – Το εκπαιδευμένο μοντέλο καλείται να ταξινομήσει τα δεδομένα σε ένα πλήθος από προκαθορισμένες κατηγορίες (multi-class classification). Για παράδειγμα, ένα μοντέλο μπορεί να εκπαιδευτεί στην κατηγοριοποίηση εικόνων χειρόγραφων μονοψήφιων αριθμών. Εδώ οι δυνατές κατηγορίες είναι δέκα (0, 1, 2, 3, 4, 5, 6, 7, 8, 9), όσο και των πλήθος των μονοψήφιων. Άλλες γενικές παραλλαγές αυτού του τύπου προβλήματος μπορούν να θεωρηθούν η δυαδική κατηγοριοποίηση (binary classification), όπου η ταξινόμηση των δεδομένων αφορά μόνο δυο διακριτές περιπτώσεις, η κατηγοριοποίηση μιας τάξης (one-class classification), όπου ελέγχονται οι συνθήκες

για τον διαχωρισμό των δεδομένων έτσι ώστε να παραμείνουν μόνο εκείνα που σχετίζονται με την περίπτωση που μας αφορά, κ.α.

- *Παλινδρόμηση (Regression)* – Το μοντέλο καλείται να προβλέψει μια αριθμητική τιμή βάσει των δεδομένων με τα οποία τροφοδοτείται. Για παράδειγμα, μια τέτοια πρόβλεψη μπορεί να σχετίζεται με την πορεία μιας μετοχής βάσει μια προηγούμενης χρονικής περιόδου, την αναμενόμενη τιμή ενός διαμερίσματος βάσει των χαρακτηριστικών του (χρονιά ανέγερσης, περιοχή, δωμάτια κλπ), πιο συγκεκριμένα, το πλήθος των εισιτηρίων που μπορεί να κοπούν για την μεταφορά σε λεωφορείο για την Αθήνα μετά τις διακοπές των Χριστουγέννων βάσει των προηγούμενων χρόνων, κ.α.
- *Ομαδοποίηση (Clustering)* – Το μοντέλο καλείται να διαχωρίσει ένα πλήθος δεδομένων σε ομάδες χωρίς κάποια βοήθεια για το πλήθος των ομάδων. Για παράδειγμα, ένας τέτοιος διαχωρισμός θα μπορούσε να γίνει για τους πελάτες ενός καταστήματος με είδη τεχνολογίας, βάσει το ιστορικό αγορών τους.
- *Ανίχνευση Ανωμαλιών (Anomaly Detection)* – Το μοντέλο μαθαίνει τα χαρακτηριστικά μιας φυσιολογικής κατάστασης και τελικά καλείται να αναγνωρίσει πάνω σε νέα δεδομένα συμπεριφορές που παρεκκλίνουν και σπάνιες ή ασυνήθιστες καταστάσεις. Παραδείγματα ανίχνευσης ανωμαλιών μπορούν να θεωρηθούν ασυνήθιστες τραπεζικές συναλλαγές και ο εντοπισμός κάποιας απάτης, η ανίχνευση μιας σπάνιας ασθένειας βάσει διαγνωστικών ελέγχων, κ.α.
- *Μείωση Διαστάσεων (Dimensionality Reduction)* – Το μοντέλο καλείται να αναγνωρίσει μέσα από ένα μεγάλο πλήθος δεδομένων τα σημαντικότερα χαρακτηριστικά του και τελικά να σχηματίσει μια νέα περιορισμένη αναπαράστασή τους σε μια μικρότερη διάσταση. Για παράδειγμα, στην επεξεργασία εικόνων, η εύρεση των σημαντικότερων στοιχείων μιας φωτογραφίας μπορεί να βοηθήσει σε μια καλύτερη συμπίεσή της χωρίς μεγάλη απώλεια ανάλυσης.
- *Σύνθεση και Δειγματοληψία (Synthesis and Sampling)* – Το μοντέλο, το οποίο σε αυτήν την περίπτωση μπορεί να αναφερθεί και ως παραγωγικό μοντέλο (generative model), καλείται να εκπαιδευτεί στην παραγωγή νέα δεδομένων βάσει αυτών στα οποία έχει εκπαιδευτεί. Για παράδειγμα, ένα μοντέλο εκπαιδευμένο με τεχνικές Επεξεργασίας Φυσικής Γλώσσας μπορεί να δεχθεί μια πρόταση ως επικεφαλίδα και να παράγει ένα σχετικό κείμενο, κάποιο που έχει εκπαιδευτεί σε τρισδιάστατα μοντέλα αντικειμένων χώρου μπορεί να δημιουργήσει επιπλέον παρόμοια αντικείμενα για να χρησιμοποιηθούν σε πολλά σημεία ενός βιντεοπαιχνιδιού,

κάποιο μοντέλο εκπαιδευμένο σε ακουστικά σήματα θα μπορούσε να συνθέσει νέα μουσική, κ.α.

Η παραπάνω κατηγοριοποίηση αποτελεί έναν γενικότερο διαχωρισμό των συνηθισμένων εφαρμογών της Μηχανικής Μάθησης και σε καμία περίπτωση δεν είναι απόλυτος. Υπάρχει πληθώρα και τύποι εφαρμογών τόσο ειδικότεροι, όσο και συνδυαστικοί, καθώς τα προβλήματα που μπορεί να λύσει η Μηχανική Μάθηση καταλαμβάνουν ένα μεγάλο φάσμα το οποίο επεκτείνεται σε πολλούς τομείς.

1.2 Δεδομένα

Βασικές έννοιες για την ανάπτυξη αλγορίθμων Μηχανικής Μάθησης είχαν ήδη παγιωθεί από την δεκαετία του ’90, αλλά μεγάλα βήματα στην εξέλιξη του πεδίου ξεκίνησαν να παρουσιάζονται δυο δεκαετίες αργότερα. Μεγάλο ρόλο σε αυτήν την εξέλιξη έπαιξε ο τεράστιος όγκος δεδομένων που τελικά ήταν δυνατόν να συλλεχθεί χάρη στις καινοτομίες που έφερε η Βιομηχανία 4.0.

Προφανώς τα δεδομένα αποτελούν αναγκαίο κομμάτι της εκπαίδευσης ενός αλγορίθμου Μηχανικής Μάθησης, αλλά προκειμένου να επιτευχθεί μια όσο το δυνατόν καλύτερη εκπαιδευτική διαδικασία, το διαθέσιμο σύνολο δεδομένων (dataset) πρέπει να προεπεξεργαστεί κατάλληλα μέσα από κάποια συγκεκριμένα βήματα:

- Ανάλυση Δεδομένων (Data Analysis)
- Καθαρισμός Δεδομένων (Data Cleaning)
- Επιλογή Χαρακτηριστικών (Feature Selection)
- Αλλαγή Κλίμακας Χαρακτηριστικών (Feature Scaling)
- Μηχανική Χαρακτηριστικών (Feature Engineering)

ΑΝΑΛΥΣΗ ΔΕΔΟΜΕΝΩΝ

Η Ανάλυση Δεδομένων αναφέρεται κυρίως στην στατιστική μελέτη για την καλύτερη κατανόηση των χαρακτηριστικών του συνόλου των δεδομένων. Είναι σημαντικό να αναγνωριστεί η φύση των μεταβλητών, δηλαδή τι εκφράζουν για το πρόβλημα που καλείται να επιλυθεί, τι τύπος δεδομένων είναι (κατηγορική ή ποσοτική), το αν υπάρχουν ακραίες τιμές (outliers) ή τιμές που λείπουν (missing values), κ.α. Επίσης, καλό είναι να γίνει οπτικοποίηση των δεδομένων μέσω διαγραμμάτων και γραφικών παραστάσεων, να

μελετηθούν τα στατιστικά μέτρα τάσης και διασποράς, και να ελεγχτούν τυχόν συσχετίσεις (correlation) μεταξύ των χαρακτηριστικών.

ΚΑΘΑΡΙΣΜΟΣ ΔΕΔΟΜΕΝΩΝ

Ο Καθαρισμός Δεδομένων περιλαμβάνει τη διόρθωση ή τη διαγραφή των ακραίων τιμών, αν αυτό κρίνεται απαραίτητο, καθώς και τη διαχείριση των τιμών που λείπουν σε διάφορα πεδία του συνόλου δεδομένων, είτε πρόκειται για κατηγορικές, είτε για ποσοτικές μεταβλητές. Οι αλγόριθμοι Μηχανικής Μάθησης δεν μπορούν να επεξεργαστούν περιπτώσεις όπου τιμές των χαρακτηριστικών απουσιάζουν, επομένως πρέπει να αφαιρεθούν, αφαιρώντας είτε την συγκεκριμένη περίπτωση που εμφανίζει την έλλειψη, είτε ολόκληρο το χαρακτηριστικό, αν αυτό κρίνεται απαραίτητο, ή πρέπει να υποκατασταθούν (imputation).

ΕΠΙΛΟΓΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Η Επιλογή Χαρακτηριστικών έχει να κάνει με το κατά πόσο ένα χαρακτηριστικό είναι απαραίτητο για την εκπαιδευτική διαδικασία και κατά πόσο συνεισφέρει σε αυτήν. Η επιλογή αυτή μπορεί να γίνει εμπειρικά μέσω της στατιστικής ανάλυσης, ή πειραματικά, είτε μέσω της ίδιας εκπαιδευτικής διαδικασίας, είτε μέσω άλλων μεθόδων, συμπεριλαμβανομένων και μερικών της Μηχανικής Μάθησης, όπως την Ομαδοποίηση (Clustering) και τη Μείωση Διαστάσεων (Dimensionality Reduction), ικανών να αναγνωρίσουν τη σημαντικότητα των χαρακτηριστικών για το σύνολο δεδομένων.

ΑΛΛΑΓΗ ΚΛΙΜΑΚΑΣ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Συνήθως, ένα σύνολο δεδομένων μπορεί να παρουσιάσει ανομοιογένεια ως προς τις αριθμητικές μεταβλητές του, δηλαδή κάποια χαρακτηριστικά μπορεί να εκφράζονται στην τάξη των χιλιάδων, ενώ κάποια άλλα στην τάξη των χιλιοστών, κλπ. Έχει παρατηρηθεί ότι οι αλγόριθμοι Μηχανικής Μάθησης δεν αποδίδουν ικανοποιητικά όταν υπάρχει μεγάλη ανισορροπία μεταξύ των τιμών, επομένως ένας τρόπος για να μετριαστεί αυτή η κατάσταση είναι η αλλαγή των τιμών όλων των χαρακτηριστικών στην ίδια κλίμακα ή σε παρόμοια κατανομή. Δυο από τις πιο συνηθισμένες μεθόδους είναι η Κανονικοποίηση (Standardization) και η Ομαλοποίηση (Normalization).

ΜΗΧΑΝΙΚΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ

Η Μηχανική Χαρακτηριστικών αποτελεί ένα από τα σημαντικότερα βήματα προεπεξεργασίας των δεδομένων και μάλιστα μπορεί να θεωρηθεί ως όρος «ομπρέλα» για όλα τα βήματα που περιγράφηκαν παραπάνω. Πρόκειται για τη διαδικασία κατά την οποία το σύνολο των δεδομένων τροποποιείται κατάλληλα έτσι ώστε να προκύψει μια καλύτερη αναπαράσταση της πληροφορίας η οποία θα διευκολύνει την εκπαίδευση του αλγορίθμου μάθησης.

Συγκεκριμένα, πέρα από την επιλογή των σχετικών χαρακτηριστικών που αναφέρθηκε, άλλος τρόπος εξαγωγής σημαντικής πληροφορίας θα ήταν η παραγωγή νέων χαρακτηριστικών από αυτά που ήδη υπάρχουν (feature aggregation). Επίσης, ο μετασχηματισμός των χαρακτηριστικών μέσω συναρτήσεων, όπως αυτή του λογάριθμου ή της τετραγωνικής ρίζας, μπορεί να συνεισφέρει σε μια πιο συμμετρική κατανομή των δεδομένων (feature transformation). Ακόμη ένας τρόπος καλύτερης αναπαράστασης αφορά στις συνεχείς τιμές, όπου προτείνεται η ομαδοποίηση τους σε αντιπροσωπευτικά διαστήματα τιμών (discretization).

Επιπλέον, κατηγορικές μεταβλητές, δηλαδή χαρακτηριστικά που περιλαμβάνουν ένα πεπερασμένο πλήθος τιμών σε μορφή χαρακτήρων, όπως για παράδειγμα την επωνυμία μια επιχείρησης σε ένα σύνολο δεδομένων σχετικά με τα χαρακτηριστικά κινητών, ή το φύλο ενός ατόμου σε ένα σύνολο δεδομένων πελατών, πρέπει να κωδικοποιηθούν σε αριθμητικές μεταβλητές, καθώς οι αλγόριθμοι μάθησης στηρίζονται σε μαθηματικά μοντέλα για την εκπαίδευσή τους και είναι αναγκαία η αριθμητική αναπαράσταση.

Μια συνηθισμένη τεχνική κωδικοποίησης αναφέρεται ως «One-Hot Encoding», κατά την οποία, από ένα κατηγορικό χαρακτηριστικό, δηλαδή από μια στήλη του συνόλου δεδομένων, παράγονται τόσα αριθμητικά χαρακτηριστικά όσο είναι το πλήθος των κατηγοριών του. Κάθε παράγωγο χαρακτηριστικό, δηλαδή κάθε νέα στήλη, περιλαμβάνει ακολουθίες δυαδικών ψηφίων (bit) από μηδέν και ένα, όπου η μονάδα εκφράζει την ύπαρξη, ενώ το μηδέν την απουσία, της κατηγορίας για κάθε περίπτωση του συνόλου δεδομένων. Φυσικά, υπάρχουν και άλλες τεχνικές κωδικοποίησης για κάθε περίπτωση κατηγορικών χαρακτηριστικών, και είναι σημαντικό να αναλογιστεί κανείς την πολυπλοκότητα μιας τέτοιας διαδικασίας σε προβλήματα Επεξεργασίας Φυσικής Γλώσσας, όπου ο αλγόριθμος Μηχανικής Μάθησης πρέπει να μάθει να αναλύει προτάσεις, κείμενα, να κατανοεί συντακτικό και γραμματική, και κάθε άλλη ιδιαιτερότητα που παρουσιάζει μια γλώσσα.

1.3 Τύποι Μάθησης

Αναφέρθηκε ότι τα προβλήματα που καλείται να επιλύσει ένας αλγόριθμος Μηχανικής Μάθησης μπορεί να κυμαίνονται σε ένα μεγάλο εύρος εφαρμογών, όπως την Κατηγοριοποίηση, την Παλινδρόμηση, την Ανίχνευση Ανωμαλιών, κ.α.

Σε ένα πρόβλημα Κατηγοριοποίησης, ο αλγόριθμος μάθησης καλείται να μάθει να ταξινομεί τα στοιχεία του συνόλου δεδομένων σε διακριτές κατηγορίες, που σημαίνει ότι πρώτα πρέπει να εκπαιδευτεί στο να διαχωρίζει αυτές τις κατηγορείς. Αυτό επιτυγχάνεται αν στο σύνολο των δεδομένων υπάρχει ένα επιπλέον χαρακτηριστικό το οποίο δηλώνει την πραγματική κατηγορία κάθε στοιχείου, και πάνω στο οποίο βασίζεται τελικά ο αλγόριθμος μάθησης κατά την εκπαίδευσή του. Στην περίπτωση ύπαρξης αυτού του χαρακτηριστικού, οι τιμές του αναφέρονται ως ετικέτες (labels), ή ως στόχοι (targets) και παίζουν καθοριστικό ρόλο στη διαδικασία μάθησης, καθώς υποδηλώνουν στον αλγόριθμο το αποτέλεσμα που πρέπει να μάθει να προβλέπει.

Επιπλέον, σε ένα πρόβλημα Παλινδρόμησης, για παράδειγμα την πρόβλεψη της αξίας μιας μετοχής, στο σύνολο των δεδομένων θα έπρεπε να υπάρχει ένα επιπλέον χαρακτηριστικό που να δείχνει τις πραγματικές αξίες της μετοχής ενός προηγούμενου χρονικού διαστήματος. Σε αυτήν την περίπτωση, το χαρακτηριστικό αυτό περιλαμβάνει τις ετικέτες που θα χρησιμοποιηθούν στην εκπαίδευση και θα καθοδηγήσουν τον αλγόριθμο. Ειδικότερα για το παραπάνω παράδειγμα, συνήθως η πορεία μιας μετοχής δίνεται με τη μορφή χρονοσειράς, συνεπώς απαιτείται πολλές φορές από τον ίδιο τον χρήστη κάποια προεπεξεργασία πάνω στο σύνολο των δεδομένων έτσι ώστε να έρθει στη ζητούμενη μορφή, ανάλογα φυσικά με το πρόβλημα που πρέπει να επιλυθεί.

Επομένως, ένας ακόμη διαχωρισμός, όσον αφορά στη διαδικασία εκπαίδευσης ενός αλγορίθμου Μηχανικής Μάθησης, μπορεί να γίνει βάσει της επίβλεψης που τού επιτρέπεται πάνω στα δεδομένα, καθώς και της διαθεσιμότητας ετικετών όσον αφορά στον σχολιασμό δεδομένων. Προφανώς, η ύπαρξη των ετικετών αποτελεί σημαντικό παράγοντα ακόμα και για την ίδια την επιλογή του αλγορίθμου και της τεχνικής μάθησης που θα ακολουθηθεί, και κάποιες βασικές περιπτώσεις είναι οι εξής:

- *Επιβλεπόμενη Μάθηση (Supervised Learning)* – Οι ετικέτες είναι διαθέσιμες στο σύνολο των δεδομένων, είτε απευθείας, είτε μέσα από το βήμα της προεπεξεργασίας, επομένως ο επιλεγμένος αλγόριθμος μάθησης βασίζεται σε αυτές για την εκπαίδευσή του.

- *Μη-Επιβλεπόμενη Μάθηση (Unsupervised Learning)* – Οι ετικέτες δεν είναι διαθέσιμες στο σύνολο των δεδομένων, επομένως ο επιλεγμένος αλγόριθμος πρέπει να βασιστεί σε διαδικασίες που θα του επιτρέψουν να αναγνωρίσει συμπεριφορές στα δεδομένα χωρίς κάποια εξωτερική βοήθεια.
- *Ημι-Επιβλεπόμενη Μάθηση (Semi-Supervised Learning)* – Χαρακτηρίζει την περίπτωση όπου υπάρχουν διαθέσιμες ετικέτες μόνο για ένα κομμάτι του συνόλου των δεδομένων, επομένως, πρέπει να εφαρμοστούν συνδυαστικές τεχνικές, τόσο επιβλεπόμενης, όσο και μη-επιβλεπόμενης μάθησης.
- *Αυτό-Επιβλεπόμενη Μάθηση (Self-Supervised Learning)* – Στην περίπτωση αυτή, οι ετικέτες δεν είναι διαθέσιμες στο αρχικό σύνολο δεδομένων, αλλά το ίδιο το σύνολο δεδομένων μπορεί να χρησιμοποιηθεί για τη δημιουργία ετικετών, ή τα ίδια τα δεδομένα που τροφοδοτούνται στον αλγόριθμο μάθησης να χρησιμοποιηθούν ως ετικέτες.
- *Ενισχυτική Μάθηση (Reinforcement Learning)* – Η περίπτωση αυτή διαφοροποιείται από την περίπτωση ύπαρξης των ετικετών, και μάλιστα μπορεί να θεωρηθεί και ξεχωριστό πεδίο έρευνας του τομέα της Τεχνητής Νοημοσύνης. Συγκεκριμένα, ο αλγόριθμος μάθησης παρατηρεί το περιβάλλον, και αφού επιλέξει να εκτελέσει κάποια ενέργεια, δέχεται ανατροφοδότηση υπό μορφή επιβράβευσης (reward) ή τιμωρίας (penalty), με σκοπό να μάθει να ακολουθεί εκείνη τη στρατηγική που θα τον οδηγήσει σε περισσότερες επιβραβεύσεις.

1.4 Κλασσικές Μέθοδοι

Οι κλασσικές μέθοδοι Μηχανικής Μάθησης βασίζονται στη Γραμμική Άλγεβρα και σε αρχές της Στατιστικής Ανάλυσης και των Πιθανοτήτων, με σκοπό τον μετασχηματισμό των δεδομένων πάνω στα οποία εκπαιδεύονται σε πιο αντιπροσωπευτικές αναπαραστάσεις, έτσι ώστε να μπορέσουν να αναγνωρίσουν και να μοντελοποιήσουν τους κανόνες που τα διέπουν, και τελικά να εφαρμόσουν αυτούς τους κανόνες σε νέα δεδομένα για προβλέψεις. Ανάλογα με το πρόβλημα που καλείται να επιλυθεί, καθώς φυσικά και με τη φύση των διαθέσιμων δεδομένων, επιλέγεται η κατάλληλη μέθοδος Μηχανικής Μάθησης, και κάποιοι από τους πιο δημοφιλείς και συνηθισμένους αλγορίθμους μάθησης είναι ονομαστικά οι παρακάτω:

- Γραμμική Παλινδρόμηση (Linear Regression)
- Πολυωνυμική Παλινδρόμηση (Polynomial Regression)

- Λογιστική Παλινδρόμηση (Logistic Regression)
- Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machine – SVM)
- k-Κοντινότεροι Γείτονες (k-Nearest Neighbors – kNN)
- Δέντρα Αποφάσεων (Decision Trees)
- Αφελής Μπέυζ (Naive Bayes)
- Ανάλυση Κύριων Συνιστωσών (Principal Component Analysis – PCA)
- Γκαουσιανή Μίξη (Gaussian Mixture)
- k-Μέσοι (k-Means)
- DBSCAN
- t-SNE
- Συνδυαστικές Μέθοδοι (Ensemble Methods):
    - Τυχαία Δάση (Random Forests)
    - Απομονωμένο Δάσος (Isolation Forest)
    - Πλειονότητα (Majority Vote)
    - Bootstrap aggregating (Bagging)
    - Pasting
    - Boosting (π.χ. Adaptive Boosting ή AdaBoost, Gradient Boosting)
    - Stacking

Τα κλασσικά μοντέλα Μηχανικής Μάθησης, αν και είναι περισσότερο απλοϊκά ως προς την αρχιτεκτονική τους, βρίσκουν εφαρμογές σε πληθώρα προβλημάτων, εφόσον αναπτύσσονται και εκπαιδεύονται κάτω από κατάλληλες συνθήκες, πολλές φορές μάλιστα πετυχαίνοντας ιδιαίτερα αξιοσημείωτα αποτελέσματα. Ωστόσο, τα συγκεκριμένα μοντέλα στηρίζονται σε στατιστικές διαδικασίες για να βρουν τους κανόνες που αναπαριστούν καλύτερα τα δεδομένα, κάτι που σημαίνει ότι ψάχνουν τη λύση σε έναν πεπερασμένο και προκαθορισμένο χώρο πιθανοτήτων. Για τον λόγο αυτό, συχνά χαρακτηρίζονται και ως «ρηχοί» αλγόριθμοι μάθησης, και παρά το γεγονός ότι υπερτερούν της κλασσικής στατιστικής ανάλυσης, συχνά παρουσιάζουν αδυναμίες σε περιπτώσεις πιο περίπλοκων και σύνθετων προβλημάτων, ή πιο πολύπλοκων και πολυδιάστατων δεδομένων, όπου πρέπει να δοθεί ιδιαίτερη προσοχή στη διαδικασία της Μηχανικής των Χαρακτηριστικών για να επιτευχθεί καλύτερη αναπαράσταση των δεδομένων. Αυτές τις αδυναμίες έρχονται τελικά να καλύψουν οι «βαθιοί» αλγόριθμοι Μηχανικής Μάθησης.

2 Βαθιά Μάθηση

Γενικότερα, ένας αλγόριθμος μάθησης προσπαθεί να μετασχηματίσει τα δεδομένα σε μια αναπαράσταση που μπορεί να κατανοήσει, και από την οποία μπορεί να εξάγει τους κανόνες που ακολουθούν τα δεδομένα. Οι κλασσικές μέθοδοι Μηχανικής Μάθησης, χαρακτηρίζονται ως «ρηχοί» αλγόριθμοι γιατί βρίσκουν τέτοιες αναπαραστάσεις χαμηλού επιπέδου. Η Βαθιά Μάθηση αποτελεί υποπεδίο της Μηχανικής Μάθησης το οποίο ασχολείται με την ανάπτυξη «βαθιών» αλγορίθμων ικανών να αναγνωρίσουν τα υποκείμενα μοτίβα που ακολουθούν τα δεδομένα μέσα από πολλαπλά, συνδυαστικά επίπεδα αναπαραστάσεων. Αυτοί οι «βαθιοί» αλγόριθμοι ονομάζονται Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks), και μάλιστα μπορούν να θεωρηθούν και αυτόνομο πεδίο μελέτης της Τεχνητής Νοημοσύνης.

2.1 Τεχνητά Νευρωνικά Δίκτυα

Τα Τεχνητά Νευρωνικά Δίκτυα αποτελούν τον πυρήνα της Βαθιάς Μάθησης. Παρουσιάζουν μια πληθώρα παραλλαγών, ανάλογα με το πρόβλημα που καλούνται να αντιμετωπίσουν, και πλέον χρησιμοποιούνται εκτενώς σε διάφορους τομείς, πετυχαίνοντας συνεχώς όλο και καλύτερες επιδόσεις με μεγαλύτερη ακρίβεια προβλέψεων, κάτι που κάνει το πεδίο της Βαθιάς Μάθησης και της μελέτης των Τεχνητών Νευρωνικών Δικτύων αντικείμενο συνεχιζόμενης έρευνας, με νέες μεθόδους και καινοτομίες να εμφανίζονται συνεχώς.

Τα Τεχνητά Νευρωνικά Δίκτυα διαφοροποιούνται από τις κλασσικές μεθόδους Μηχανικής Μάθησης, καθώς δεν στηρίζονται σε μοντέλα πιθανοτήτων αλλά σε ιεραρχικές μαθηματικές αναπαραστάσεις των δεδομένων για τη διαδικασία της εκπαίδευσή τους. Επιπλέον, αυτή η πρακτική μάθησης δεν απαιτεί ιδιαίτερη Μηχανική των Χαρακτηριστικών, καθώς από την ίδια την εκπαιδευτική διαδικασία εξάγονται και αναγνωρίζονται αυτόματα από το μοντέλο τα σημαντικότερα στοιχεία των δεδομένων, τα οποία και εκμεταλλεύεται κατά την εκπαίδευσή του για τη βελτιστοποίηση της επίδοσής του.

Επίσης, τα Τεχνητά Νευρωνικά Δίκτυα είναι πιο ευέλικτα και επεκτάσιμα, και υπάρχουν ήδη πολλές παραλλαγές όσον αφορά στη δομή και την αρχιτεκτονική τους, οι οποίες αποσκοπούν σε ένα ευρύ φάσμα εφαρμογών, με διάφορες βαθμίδες πολυπλοκότητας συνθηκών και δεδομένων.

Τελικά, ο συνδυασμός του μεγάλου όγκου δεδομένων, των σύγχρονων δυνατοτήτων των Μονάδων Κεντρικών Επεξεργαστών (Central Processing Unit - CPU), και ιδιαίτερα των

Μονάδων Επεξεργασίας Γραφικών (Graphical Processing Unit - GPU), και η συνεχής βελτίωση των αλγορίθμων μάθησης, συντέλεσαν στο να καταστήσουν τα Τεχνητά Νευρωνικά Δίκτυα και τη Βαθιά Μάθηση πρώτη επιλογή για την αντιμετώπιση περίπλοκων προβλημάτων πολυδιάστατων δεδομένων.

2.1.1 Αρχιτεκτονική

Τα Τεχνητά Νευρωνικά Δίκτυα είναι εμπνευσμένα από τη δομή του νευρικού δικτύου του βιολογικού εγκεφάλου. Παρά το γεγονός ότι η ανάπτυξή τους υλοποιείται με κεντρικές έννοιες της Νευροεπιστήμης, ο σχεδιασμός τους δεν αποσκοπεί στη ρεαλιστική μοντελοποίηση του εγκεφάλου. Αντιθέτως, στηρίζεται κατά κύριο λόγο στην προσομοίωση του τρόπου επικοινωνίας μεταξύ των νευρικών κυττάρων, η οποία επιτυγχάνεται με την αποστολή σημάτων μέσω των νευρικών συνάψεων.

Ειδικότερα, όσον αφορά στη γενικότερη αρχιτεκτονική τους, τα Τεχνητά Νευρωνικά Δίκτυα αποτελούνται από συνδεδεμένους υπολογιστικούς νευρώνες (neurons), ή αλλιώς κόμβους (nodes), οι οποίοι δομούνται σε διαδοχικά επίπεδα (layers). Συγκεκριμένα, απαρτίζονται από ένα επίπεδο εισόδου (input layer), ένα ή περισσότερα ενδιάμεσα, ή αλλιώς κρυμμένα, επίπεδα (hidden layers), και ένα επίπεδο εξόδου (output layer). Επιπλέον, όσο μεγαλύτερο είναι το πλήθος των ενδιάμεσων επιπέδων, και κατ' επέκταση το πλήθος των αναπαραστάσεων των δεδομένων, τόσο πιο «βαθύ» θεωρείται το δίκτυο. Αξίζει να γίνει υπενθύμιση ότι οι κλασσικοί μέθοδοι Μηχανικής Μάθησης περιορίζονται σε μία ή, σε κάποιες ειδικές περιπτώσεις, σε δύο τέτοιες μαθηματικές αναπαραστάσεις.

Γενικότερα, τα δεδομένα διέρχονται μέσα από το δίκτυο, από το χαμηλότερο επίπεδο εισόδου, μέχρι το ψηλότερο επίπεδο εξόδου, και περνώντας από κάθε επιμέρους υπολογιστικό επίπεδο νευρώνων υπόκεινται σε μαθηματικούς μετασχηματισμούς. Τελικά, το δίκτυο εξάγει από τα δεδομένα εισόδου (input data), τα μετασχηματισμένα δεδομένα εξόδου (output data), τα οποία αποτελούν το συνολικό αποτέλεσμα όλων των μαθηματικών αναπαραστάσεων των υπολογιστικών επιπέδων. Τα συγκεκριμένα αποτελέσματα χαρακτηρίζονται ως προβλέψεις (predictions) του νευρωνικού δικτύου, οι οποίες αξιολογούνται μέσω της εκπαιδευτικής διαδικασίας, σύμφωνα με μια επαναληπτική τεχνική, με σκοπό να επιτευχθεί μια όσο το δυνατόν καλύτερη αναπαράσταση των δεδομένων από τα επίπεδα του δικτύου.

2.1.2 Τύποι Τεχνητών Νευρωνικών Δικτύων

Τα επίπεδα αποτελούν το κύριο δομικό στοιχείο των Τεχνητών Νευρωνικών Δικτύων και ανάλογα με τις υπολογιστικές δυνατότητες των νευρώνων που τα απαρτίζουν, αλλά και τη συνολική οργάνωσή τους για τον σχεδιασμό της κατάλληλης αρχιτεκτονικής, υπαγορεύουν τη φύση και τη λειτουργικότητα του ίδιου του δικτύου, η οποία φυσικά εξαρτάται από το πρόβλημα για το οποίο προορίζεται να εκπαιδευτεί το δίκτυο και τα δεδομένα με τα οποία πρόκειται να τροφοδοτηθεί. Κάποιοι βασικοί τύποι και αρχιτεκτονικές Τεχνητών Νευρωνικών Δικτύων είναι οι παρακάτω:

- *Πολυεπίπεδο Αντίληπτρο (Multilayer Perceptron - MLP)* – Η πιο συμβατική αρχιτεκτονική νευρωνικού δικτύου, η οποία με κατάλληλη παραμετροποίηση μπορεί να εφαρμοστεί με μεγάλη επιτυχία σε ένα εύρος εφαρμογών, βρίσκοντας συσχετίσεις κυρίως σε περιπτώσεις δεδομένων πινάκων (tabular data).
- *Συνελικτικό Νευρωνικό Δίκτυο (Convolutional Neural Network - CNN)* – Δίκτυα τα οποία επικεντρώνονται πάνω στην εκπαίδευση οπτικών δεδομένων, όπως εικόνες και βίντεο, και βρίσκουν πολλές εφαρμογές στον τομέα της Υπολογιστικής Όρασης (Computer Vision).
- *Επαναληπτικό Νευρωνικό Δίκτυο (Recurrent Neural Network - RNN)* – Δίκτυα τα οποία είναι ικανά να διαχειρίζονται δεδομένα ακολουθιών (sequential data), δηλαδή δεδομένα όπου υπάρχει εξάρτηση ως προς τη σειρά που παρουσιάζονται τα στοιχεία τους, όπως είναι για παράδειγμα οι χρονοσειρές, τα ακουστικά σήματα ομιλίας, τα γραπτά κείμενα γλώσσας, κ.α.
- *Μακρά Βραχυπρόθεσμη Μνήμη (Long Short-Term Memory - LSTM)* – Βελτιωμένη εκδοχή των Επαναληπτικών Νευρωνικών Δικτύων, τα οποία είναι ικανά να διαχειρίζονται ακολουθίες μεγαλύτερης έκτασης, αναγνωρίζοντας έτσι πιο περίπλοκα μοτίβα στα δεδομένα.
- *Επαναλαμβανόμενη Μονάδα με Πύλη (Gated Recurrent Unit - GRU)* – Βελτιωμένη εκδοχή των Επαναληπτικών Νευρωνικών Δικτύων, αλλά με πιο απλοϊκή αρχιτεκτονική σε σχέση με τα δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης.
- *Αμφίδρομο Επαναληπτικό Νευρωνικό Δίκτυο (Bidirectional Recurrent Neural Network - BRNN)* – Δίκτυο με ιδιαίτερη εφαρμογή σε προβλήματα φυσικής γλώσσας, καθώς επιτρέπει την ανάλυση δεδομένων τόσο από την αρχή, όσο και από το τέλος των ακολουθιών, το οποίο συμβάλει σε μια καλύτερη και πιο ολοκληρωμένη οπτική εικόνα, αν αναλογιστούμε ότι είναι απαραίτητη η ανάγνωση

ενός κειμένου στο σύνολό του για να κατανοήσουμε το νόημά του, και δεν αρκούν μόνο οι πρώτες λέξεις.

- *Αυτοκωδικοποιητής (Autoencoder)* – Τεχνική νευρωνικού δικτύου η οποία, συμπιέζει τα δεδομένα εισόδου σε κωδικοποιημένες αναπαραστάσεις και στη συνέχεια τα ανακατασκευάζει, βασιζόμενη μόνο στις κωδικοποιήσεις, μαθαίνοντας έτσι τα πιο σχετικά χαρακτηριστικά των δεδομένων.
- *Μεταβλητός Αυτοκωδικοποιητής (Variational Autoencoder - VAE)* – Παραγωγικό μοντέλο το οποίο είναι ικανό να αναγνωρίζει, μέσα από την εκπαίδευση με την τεχνική των κωδικοποιήσεων, την κατανομή πιθανοτήτων των δεδομένων που επεξεργάζεται, και να παράγει νέα δεδομένα παρόμοια με αυτά που έχει εκπαιδευτεί.
- *Παραγωγικό Ανταγωνιστικό Δίκτυο (Generative Adversarial Network - GAN)* – Παραγωγικό μοντέλο, το οποίο αποτελείται από ένα δίκτυο Παραγωγής (Generator) και ένα δίκτυο Διάκρισης (Discriminator), όπου το δίκτυο Παραγωγής δημιουργεί νέα δεδομένα, με σκοπό το δίκτυο Διάκρισης να αναγνωρίσει αν είναι πραγματικά ή συνθετικά, και από αυτόν τον «ανταγωνισμό» μεταξύ των δυο δικτύων επιτυγχάνεται η παραγωγή όλο και πιο ρεαλιστικών δεδομένων από το δίκτυο Παραγωγής.
- *Μετασχηματιστής (Transformer)* – Αρχιτεκτονική που υλοποιείται με μηχανισμούς νευρωνικής προσοχής (neural attention) για την επεξεργασία εκτενών ακολουθιών και την αναγνώριση των εξαρτήσεών τους, η οποία γνώρισε μεγάλη επιτυχία σε προβλήματα Επεξεργασίας Φυσικής Γλώσσας, υπερτερώντας των προηγούμενων μεθόδων, πετυχαίνοντας πρωτοφανείς επιδόσεις (το πλέον δημοφιλές Chat Generative Pre-trained Transformer-ChatGPT ακολουθεί την αρχιτεκτονική του Μετασχηματιστή).

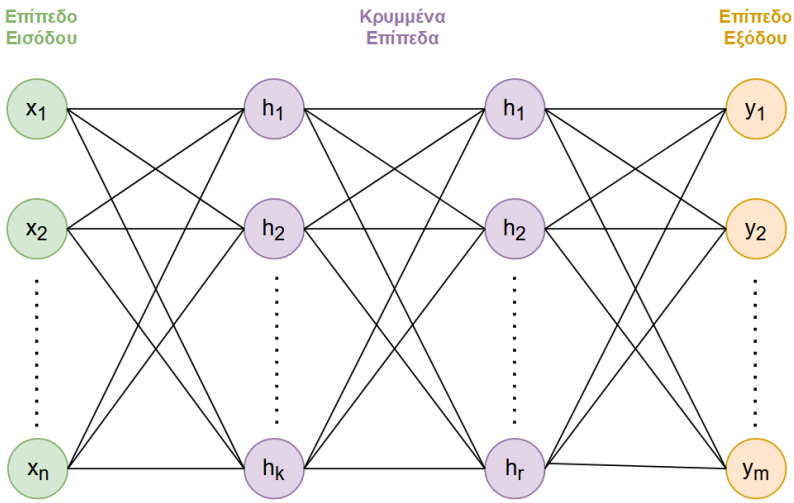
Η κατηγοριοποίηση των νευρωνικών δικτύων ποικίλει, καθώς κάθε αρχιτεκτονική είναι ειδικά σχεδιασμένη για να εξυπηρετεί συγκεκριμένες περιπτώσεις προβλημάτων και εφαρμογών, χωρίς φυσικά να περιορίζεται η χρήση της μόνο σε αυτές. Μάλιστα, υπάρχει ένα μεγάλο φάσμα όσον αφορά στις υπολογιστικές δυνατότητες που μπορούν να προσδοθούν στα επίπεδα ενός νευρωνικού δικτύου, και δεν αποτελεί κάτι το ασυνήθιστο ο συνδυασμός επιπέδων διαφορετικών υπολογιστικών νευρώνων και τεχνικών με σκοπό την ανάπτυξη δικτύων με καλύτερες ικανότητες συμπερασματολογίας.

Παρακάτω εξηγούνται αναλυτικά οι τεχνικές που επιλέχτηκαν στα πλαίσια της συγκεκριμένης Πτυχιακής Εργασίας, και οι οποίες εφαρμόζονται στα παραδείγματα του 3ου Κεφαλαίου.

2.2 Πολυεπίπεδο Αντίληπτρο (Multilayer Perceptron - MLP)

2.2.1 Αρχιτεκτονική MLP

Το Πολυεπίπεδο Αντίληπτρο (Multilayer Perceptron – MLP) αποτελείται από ένα επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα και ένα επίπεδο εξόδου, τα οποία είναι πλήρως διασυνδεδεμένα μεταξύ τους (fully-connected ή densely-connected), που σημαίνει ότι κάθε νευρώνας ενός επιπέδου συνδέεται με όλους τους νευρώνες του επόμενου επιπέδου. Επιπλέον, τα δίκτυα MLP ανήκουν στην κατηγορία των Νευρωνικών Δικτύων της Προς τα Εμπρός Διάδοσης (Feed-Forward Neural Network – FFNN), δηλαδή τα δεδομένα, κατά την επεξεργασία και την ανάλυσή τους σε μαθηματικές αναπαραστάσεις, περνάνε μέσα από το δίκτυο μόνο προς μια κατεύθυνση, από το επίπεδο εισόδου μέχρι το επίπεδο εξόδου, χωρίς να ανατροφοδοτούνται μέσω βρόγχων επανάληψης. Τα δίκτυα MLP αποτελούν την πιο συμβατική αρχιτεκτονική Τεχνητού Νευρωνικού Δικτύου, τόσο από πλευράς εφαρμογών σε πραγματικά προβλήματα, όσο και από πλευράς μαθηματικών μετασχηματισμών των δεδομένων μέσα από τα επίπεδα.



Εικόνα 4. Απεικόνιση Αρχιτεκτονικής Δικτύου MLP

Το παραπάνω νευρωνικό δίκτυο MLP αποτελείται από ένα επίπεδο εισόδου, δυο κρυμμένα επίπεδα και ένα επίπεδο εξόδου. Το επίπεδο εισόδου περιλαμβάνει n νευρώνες, το πλήθος των οποίων ορίζεται βάσει των δεδομένων που πρέπει να επεξεργαστούν. Τα δυο κρυμμένα επίπεδα δομούνται με k και r νευρώνες αντίστοιχα, το πλήθος των οποίων μπορεί να είναι οποιοσδήποτε αριθμός, πάντα φυσικά με βάση το πρόβλημα που πρέπει να αντιμετωπιστεί. Το επίπεδο εξόδου αποτελείται από m νευρώνες, το πλήθος των οποίων είναι επίσης κάποιος συγκεκριμένος αριθμός, όπως στο επίπεδο εισόδου, ο οποίος ακολουθεί τα δεδομένα του προβλήματος.

2.2.2 Διαδικασία Εκπαίδευσης

Στόχος της εκπαίδευσης ενός νευρωνικού δικτύου είναι να βρεθούν εκείνοι οι μετασχηματισμοί που περιγράφουν καλύτερα τα δεδομένα του εκάστοτε προβλήματος. Αυτό επιτυγχάνεται με μια διαδικασία, κατά την οποία όλα τα δεδομένα εκπαίδευσης (training dataset) χωρίζονται σε τυχαίες παρτίδες (batches) προκαθορισμένου πλήθους στοιχείων (samples), συνήθως δύναμης του δύο, οι οποίες με τη σειρά τους περνάνε διαδοχικά μέσα από το δίκτυο για επεξεργασία. Το δίκτυο αρχικοποιεί τυχαία τους μετασχηματισμούς των δεδομένων και στη συνέχεια τους διαμορφώνει κατάλληλα με κάθε νέα παρτίδα που αναλύει. Η ολοκλήρωση του ενός περάσματος όλων των παρτίδων των δεδομένων μέσα από το δίκτυο σηματοδοτεί το τέλος μιας εποχής (epoch). Η εκπαίδευση του νευρωνικού δικτύου επιτυγχάνεται με την παραπάνω επαναληπτική διαδικασία ανάλυσης των δεδομένων μέσα από ένα πλήθος εποχών το οποίο ορίζεται πειραματικά κατά την ανάπτυξη του μοντέλου.

Η ολοκληρωμένη διαδικασία εκπαίδευσης ενός δικτύου MLP, η οποία περιγράφεται παρακάτω αναλυτικά, επιτυγχάνεται σε τρία βήματα:

- Διάδοση προς τα Εμπρός (Forward Pass)
- Διάδοση προς τα Πίσω (Backward Pass)
- Κατάβαση κατά Κλίση (Gradient Descent)

Θεωρούμε ότι το πρόβλημα που καλείται να επιλυθεί αποτελεί περίπτωση Επιβλεπόμενης Μάθησης, δηλαδή είναι διαθέσιμες οι ετικέτες με τις πραγματικές τιμές που πρέπει να μάθει να προβλέπει το νευρωνικό δίκτυο. Επιπλέον, η παρακάτω διαδικασία περιγράφει την περίπτωση ανάλυσης ενός στοιχείου μιας παρτίδας των δεδομένων, δηλαδή την ανάλυση μιας γραμμής ενός πίνακα δεδομένων όπου οι στήλες περιγράφουν τα

χαρακτηριστικά του στοιχείου, προτού ενσωματωθεί το συμπέρασμα για ολόκληρη την παρτίδα των στοιχείων.

ΔΙΑΔΟΣΗ ΠΡΟΣ ΤΑ ΕΜΠΡΟΣ

Κάθε νευρώνας ενός επιπέδου, εκτός του επιπέδου εισόδου, εφαρμόζει γραμμικούς μετασχηματισμούς στα δεδομένα με τα οποία τροφοδοτείται. Ειδικότερα, η απόληξη κάθε σύνδεσης σε έναν νευρώνα προσδιορίζει την πράξη του πολλαπλασιασμού της εκάστοτε τιμής των δεδομένων με μια παράμετρο βάρους (weight), ενώ ο ίδιος ο νευρώνας εφαρμόζει την πράξη της πρόσθεσης μεταξύ όλων των γινομένων που προκύπτουν από όλες τις απολήξεις, και στο τελικό άθροισμά τους ενσωματώνει και τον επιπλέον προσθετέο της παραμέτρου του σταθερού όρου (bias). Το τελικό αποτέλεσμα θεωρείται η έξοδος του νευρώνα του επιπέδου, η οποία με τη σειρά της αποτελεί την είσοδο για κάθε νευρώνα του επόμενου επιπέδου του δικτύου.

Επομένως, κάθε επίπεδο εξάγει αναπαραστάσεις από τα δεδομένα που τού τροφοδοτούνται, και είναι προφανές ότι η αλυσιδωτή εφαρμογή πολλών διαδοχικών επιπέδων συνεισφέρει σε μια πιο «βαθιά» κατανόηση των δεδομένων. Ωστόσο, επίπεδα νευρώνων τα οποία εφαρμόζουν διαδοχικούς γραμμικούς μετασχηματισμούς, καταλήγουν στην εξαγωγή ενός αποτελέσματος το οποίο παραμένει ένας απλός γραμμικός μετασχηματισμός, ανάλογος της κλασσικής μεθόδου Μηχανικής Μάθησης της Γραμμικής Παλινδρόμησης, κάτι που δυσκολεύει στην αναγνώριση μοτίβων πιο πολύπλοκων δεδομένων. Για τον λόγο αυτό, και σύμφωνα με το πρόβλημα που ερευνάται, εφαρμόζονται επιπλέον μαθηματικοί μετασχηματισμοί στις εξόδους των νευρώνων των επιπέδων, προτού χρησιμοποιηθούν ως είσοδοι για το επόμενο επίπεδο, μέσω συναρτήσεων ενεργοποίησης (activation functions), οι οποίες προσδίδουν μη-γραμμικές δυνατότητες αναπαράστασης στο δίκτυο. Τελικά, το τελευταίο επίπεδο του δικτύου εξάγει την πρόβλεψή του για το τρέχον στοιχείο των δεδομένων που αναλύει.

Μόλις η παραπάνω διαδικασία ολοκληρωθεί για κάθε στοιχείο της παρτίδας που επεξεργάζεται εκείνη τη στιγμή το δίκτυο, δηλαδή μόλις το επίπεδο εξόδου έχει αποκτήσει προβλέψεις για κάθε στοιχείο της παρτίδας, τότε έχει ολοκληρωθεί το βήμα της προς τα εμπρός διάδοσης της παρτίδας.

ΔΙΑΔΟΣΗ ΠΡΟΣ ΤΑ ΠΙΣΩ

Μόλις εξαχθούν όλες οι προβλέψεις για κάθε στοιχείο της παρτίδας, τότε το μοντέλο του νευρωνικού δικτύου χρησιμοποιεί μια συνάρτηση απώλειας (loss function), η οποία είναι επίσης προκαθορισμένη σύμφωνα με το πρόβλημα που ερευνάται, και υπολογίζει το συνολικό σφάλμα μεταξύ των προβλέψεων και των πραγματικών τιμών, οι οποίες είναι διαθέσιμες γιατί πρόκειται για περίπτωση Επιβλεπόμενης Μάθησης.

Στη συνέχεια, με μια διαδικασία η οποία αναφέρεται ως Αλγόριθμος Οπισθοδιάδοσης (Backpropagation Algorithm), το συνολικό σφάλμα διαδίδεται προς τα πίσω μέσω των επιπέδων του δικτύου, από το επίπεδο εξόδου προς το επίπεδο εισόδου, και με την εφαρμογή του κανόνα της αλυσίδας του διαφορικού λογισμού (chain rule of calculus) υπολογίζονται οι μερικοί παράγωγοι του συνολικού σφάλματος ως προς κάθε παράμετρο βαρών και σταθερών όρων των νευρώνων του δικτύου, με σκοπό να εκτιμηθεί η συνεισφορά της κάθε παραμέτρου στο σφάλμα.

ΚΑΤΑΒΑΣΗ ΚΑΤΑ ΚΛΙΣΗ

Τελικά, αφού υπολογιστούν οι μερικοί παράγωγοι του συνολικού σφάλματος ως προς κάθε παράμετρο του νευρωνικού δικτύου, στη συνέχεια, με την εφαρμογή ενός κατάλληλα επιλεγμένου αλγορίθμου βελτιστοποίησης (optimization algorithm), όλες οι παράμετροι ανανεώνονται σύμφωνα με τη συνεισφορά τους στο συνολικό σφάλμα. Ο βαθμός αυτή της ανανέωσης εξαρτάται από τη σταθερά ρυθμού μάθησης (learning rate) και έχει σκοπό τη μείωση του συνολικού σφάλματος που υπολογίζεται από τη συνάρτηση απώλειας, δηλαδή αποσκοπεί στην ελαχιστοποίηση της συνάρτησης απώλειας με κάθε επιμέρους παρτίδα που αναλύεται από το δίκτυο κατά την εκπαίδευσή του.

Επομένως, σκοπός της επαναληπτικής εκπαιδευτικής διαδικασίας είναι να βρεθούν όσο το δυνατόν καλύτερες εκτιμήσεις για τις παραμέτρους του νευρωνικού δικτύου, οι οποίες είναι υπεύθυνες για τις αναπαραστάσεις των δεδομένων, έτσι ώστε να οδηγούν σε όσο το δυνατόν μικρότερο σφάλμα με κάθε επιμέρους επανάληψη.

2.2.3 Μαθηματικό Παράδειγμα Εκπαίδευσης

Στη συνέχεια παρουσιάζεται αναλυτικά ένα αυθαίρετο μαθηματικό παράδειγμα, για την καλύτερη κατανόηση των παραπάνω βημάτων μιας συμβατικής εκπαιδευτικής διαδικασίας ενός νευρωνικού δικτύου. Περιγράφονται η εξαγωγή προβλέψεων μέσω της Διάδοσης προς τα Εμπρός, ο υπολογισμός του σφάλματος πρόβλεψης, η Διάδοση προς τα Πίσω για τον

υπολογισμό της συνεισφορά κάθε παραμέτρου στο σφάλμα, και τελικά η ενημέρωση των παραμέτρων με σκοπό την μείωση του σφάλματος.

ΟΡΙΣΜΟΣ ΔΕΔΟΜΕΝΩΝ ΚΑΙ ΠΡΟΒΛΗΜΑΤΟΣ

Έστω ο παρακάτω πίνακας δεδομένων, ο οποίος αποτελείται από 500 εγγραφές (records), από τα δυο χαρακτηριστικά (attributes ή features) X1 και X2, και από το χαρακτηριστικό στόχου (target) Y:

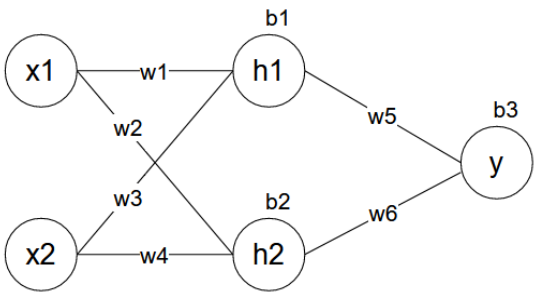
Πίνακας 1. Παράδειγμα Πίνακα Δεδομένων

Index	X1	X2	Y
1	0.2	0.3	1.5
2	…	…	…
…	…	…	…
500	…	…	…

Ο σκοπός ενός μοντέλου νευρωνικού δικτύου MLP είναι να βρεθούν εκείνοι οι μαθηματικοί μετασχηματισμοί που προβλέπουν καλύτερα τον στόχο Y, με δεδομένα εισόδου τα χαρακτηριστικά X1 και X2.

ΟΡΙΣΜΟΣ ΑΡΧΙΤΕΚΤΟΝΙΚΗΣ ΚΑΙ ΠΑΡΑΜΕΤΡΩΝ

Έστω η παρακάτω αρχιτεκτονική MLP η οποία αποτελείται από ένα επίπεδο εισόδου, ένα κρυμμένο επίπεδο και ένα επίπεδο εξόδου:



Εικόνα 5. Νευρωνικό Δίκτυο Παραδείγματος

όπου το επίπεδο εισόδου αποτελείται από δυο νευρώνες x1 και x2, εφόσον και στον πίνακα δεδομένων τα χαρακτηριστικά κάθε στοιχείου είναι δύο, το κρυμμένο επίπεδο αποτελείται από δυο νευρώνες h1 και h2, η επιλογή των οποίων έγινε αυθαίρετα (θα μπορούσαμε να είχαμε και παραπάνω νευρώνες, αλλά για τα πλαίσια του παραδείγματος είναι αρκετοί), και το επίπεδο εξόδου αποτελείται από έναν νευρώνα y, καθώς το μοντέλο καλείται να προβλέψει μόνο μια τιμή.

Επιπλέον, οι μεταβλητές w_i με $i = 1,2,3,4,5,6$ δηλώνουν τις παραμέτρους των βαρών μεταξύ των ενώσεων των νευρώνων, ενώ οι μεταβλητές b_i με $i = 1,2,3$ δηλώνουν τις παραμέτρους των σταθερών όρων, τις οποίες ενσωματώνουν οι νευρώνες στους μετασχηματισμούς τους με την πράξη της πρόσθεσης. Οι τιμές των παραμέτρων για το συγκεκριμένο παράδειγμα επιλέχθηκαν αυθαίρετα ως εξής:

Πίνακας 2. Παράμετροι Νευρωνικού Δικτύου Παραδείγματος

Βάρη		**Σταθεροί Όροι**
$w_1 = 0.2$	$w_4 = 0.5$	$b_1 = 0.1$
$w_2 = 0.3$	$w_5 = 0.1$	$b_2 = 0.4$
$w_3 = 0.4$	$w_6 = 0.3$	$b_3 = 0.2$

ΟΡΙΣΜΟΣ ΣΥΝΑΡΤΗΣΕΩΝ

Για καλύτερη κατανόηση των μαθηματικών πράξεων επιλέχτηκε ως συνάρτηση ενεργοποίησης όλων των επιπέδων η $act(x) = x^2 + 5x$, ενώ ως συνάρτηση απώλειας του μοντέλου, για την αξιολόγηση του σφάλματος των προβλέψεων, επιλέχτηκε η συνάρτηση του μέσου τετραγωνικού σφάλματος (mean squared error) $MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ όπου το n δηλώνει το πλήθος των στοιχείων της παρτίδας που αναλύεται, y_i είναι οι πραγματικές τιμές που αντιστοιχούν σε κάθε στοιχείο, και $\hat{y}_i$ είναι οι αντίστοιχες προβλέψεις του μοντέλου για το εκάστοτε εκπαιδευτικό βήμα.

ΔΙΑΔΟΣΗ ΠΡΟΣ ΤΑ ΕΜΠΡΟΣ

Υπενθυμίζεται ότι το επίπεδο εισόδου δεν εφαρμόζει μετασχηματισμούς στα δεδομένα, αλλά μόνο τα τροφοδοτεί στο δίκτυο. Επίσης, η παρακάτω διαδικασία αφορά μονάχα την ανάλυση ενός στοιχείου (γραμμή του πίνακα δεδομένων) από την συνολική παρτίδα.

Για τους νευρώνες του κρυμμένου επιπέδου οι μετασχηματισμοί που προκύπτουν είναι:

$$\boldsymbol{h_1} = x_1 \cdot w_1 + x_2 \cdot w_3 + b_1 = 0.2 \cdot 0.2 + 0.3 \cdot 0.4 + 0.1 = \mathbf{0.26}$$

$$\boldsymbol{h_2} = x_1 \cdot w_2 + x_2 \cdot w_4 + b_2 = 0.2 \cdot 0.3 + 0.3 \cdot 0.5 + 0.4 = \mathbf{0.61}$$

Με την εφαρμογή της συνάρτησης ενεργοποίησης οι τελικές έξοδοι του επιπέδου είναι:

$$\boldsymbol{output\_h_1} = {h_1}^2 + 5 \cdot h_1 = 0.26^2 + 5 \cdot 0.26 = 1.3676 \approx \mathbf{1.37}$$

$$\boldsymbol{output\_h_2} = {h_2}^2 + 5 \cdot h_2 = 0.61^2 + 5 \cdot 0.61 = 3.4221 \approx \mathbf{3.42}$$

Αντίστοιχα, για τον νευρώνα του επιπέδου εξόδου ισχύουν οι παρακάτω μετασχηματισμοί:

$$\boldsymbol{y} = output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3 = 1.37 \cdot 0.1 + 3.42 \cdot 0.3 + 0.2 = 1.363 \approx \mathbf{1.36}$$

$$\boldsymbol{output\_y} = y^2 + 5 \cdot y = 1.36^2 + 5 \cdot 1.36 = 8.6496 \approx \mathbf{8.65}$$

Ολοκληρώθηκε η Διάδοση προς τα Εμπρός, και η τελική πρόβλεψη για το πρώτο στοιχείο (X1, X2, Y) του πίνακα δεδομένων είναι 8.65 (οι στρογγυλοποιήσεις των αριθμών γίνονται στο δεύτερο δεκαδικό ψηφίο για καλύτερη διαχείριση και κατανόηση του παραδείγματος). Σε περίπτωση ανάλυσης παρτίδας δεδομένων, η παραπάνω διαδικασία θα γινόταν για κάθε στοιχείο (γραμμή του πίνακα) που θα ανήκε στην εκάστοτε παρτίδα.

ΥΠΟΛΟΓΙΣΜΟΣ ΣΦΑΛΜΑΤΟΣ

Στη συνέχεια, υπολογίζεται το συνολικό σφάλμα από τη συνάρτηση απώλειας:

$$\boldsymbol{E_{total}} = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = (Y - output\_y)^2 = (1.5 - 8.65)^2$$

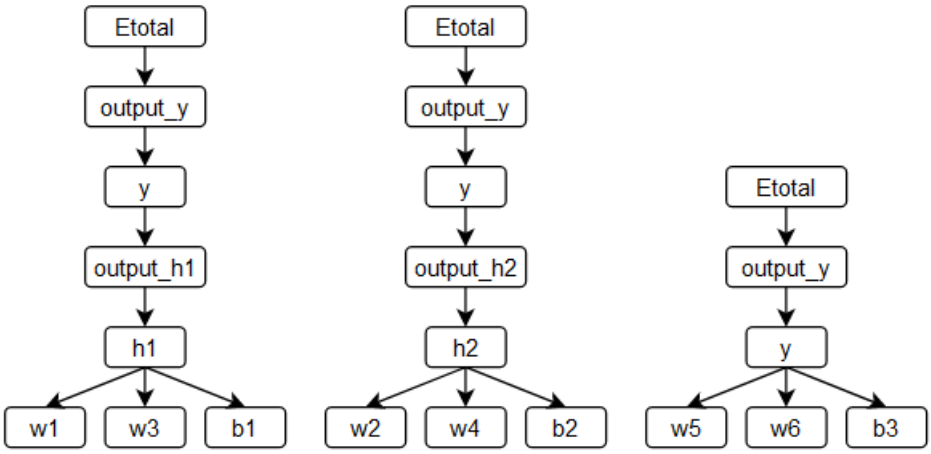
$$= (-7.15)^2 = 51.1225 \approx \mathbf{51.12}$$

Το συγκεκριμένο σφάλμα αφορά μόνο την απώλεια πρόβλεψης για το στοιχείο που αναλύεται, και σε περίπτωση ανάλυσης παρτίδας δεδομένων το συνολικό σφάλμα θα υπολογιζόταν ως προς όλα τα στοιχεία της παρτίδας.

ΔΙΑΔΟΣΗ ΠΡΟΣ ΤΑ ΠΙΣΩ

Το συνολικό σφάλμα E_{total} διαδίδεται προς την αντίθετη κατεύθυνση του νευρωνικού δικτύου διαδοχικά, και μέσω του κανόνα της αλυσίδας υπολογίζεται η μερική παράγωγός του ως προς κάθε παράμετρο w_i και b_i του μοντέλου.

Για την καλύτερη κατανόηση της εφαρμογής του κανόνα της αλυσίδας, δίνεται το παρακάτω διάγραμμα:



Εικόνα 6. Απεικόνιση Κανόνα Αλυσίδας Παραδείγματος

Για τις παραμέτρους του επιπέδου εξόδου για τον ένα νευρώνα y έχουμε:

- $$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial w_5}$$
$$= 14.3 \cdot 7.72 \cdot 1.37 = 151.24252 \approx \mathbf{151.24}$$
- $$\frac{\partial E_{total}}{\partial w_6} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial w_6}$$
$$= 14.3 \cdot 7.72 \cdot 3.42 = 377.55432 \approx \mathbf{377.55}$$
- $$\frac{\partial E_{total}}{\partial b_3} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial b_3}$$
$$= 14.3 \cdot 7.72 \cdot 1 = 110.396 \approx \mathbf{110.4}$$

Διότι:

$$\frac{\partial E_{total}}{\partial output\_y} = \frac{\partial (Y - output\_y)^2}{\partial output\_y} = 2(Y - output\_y)(-1) = -2 \cdot (1.5 - 8.65) = 14.3$$

$$\frac{\partial output\_y}{\partial y} = \frac{\partial (y^2 + 5y)}{\partial y} = 2y + 5 = 2 \cdot 1.36 + 5 = 7.72$$

$$\frac{\partial y}{\partial w_5} = \frac{\partial (output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3)}{\partial w_5} = output\_h_1 = 1.37$$

$$\frac{\partial y}{\partial w_6} = \frac{\partial (output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3)}{\partial w_6} = output\_h_2 = 3.42$$

$$\frac{\partial y}{\partial b_3} = \frac{\partial (output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3)}{\partial b_3} = 1$$

Για τις παραμέτρους του κρυμμένου επιπέδου για τον νευρώνα h1 έχουμε:

- $$\frac{\partial E_{total}}{\partial w_1} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$= 14.3 \cdot 7.72 \cdot 0.1 \cdot 5.52 \cdot 0.2 = 12.1877184 \approx \mathbf{12.19}$$

- $$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_3}} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_3} =$$

$$= 14.3 \cdot 7.72 \cdot 0.1 \cdot 5.52 \cdot 0.3 = 18.2815776 \approx \mathbf{18.28}$$

- $$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial b_1}} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1} =$$

$$= 14.3 \cdot 7.72 \cdot 0.1 \cdot 5.52 \cdot 1 = 60.938592 \approx \mathbf{60.94}$$

Και για τις παραμέτρους του κρυμμένου επιπέδου για τον νευρώνα h2 έχουμε:

- $$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_2}} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$$

$$= 14.3 \cdot 7.72 \cdot 0.3 \cdot 6.22 \cdot 0.2 = 41.1997872 \approx \mathbf{41.2}$$

- $$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_4}} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_4} =$$

$$= 14.3 \cdot 7.72 \cdot 0.3 \cdot 6.22 \cdot 0.3 = 61.7996808 \approx \mathbf{61.8}$$

- $$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial b_2}} = \frac{\partial E_{total}}{\partial output\_y} \cdot \frac{\partial output\_y}{\partial y} \cdot \frac{\partial y}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2} =$$

$$= 14.3 \cdot 7.72 \cdot 0.3 \cdot 6.22 \cdot 1 = 205.998936 \approx \mathbf{206}$$

Διότι:

$$\frac{\partial E_{total}}{\partial output\_y} = \frac{\partial (Y - output\_y)^2}{\partial output\_y} = 2(Y - output\_y)(-1) = -2 \cdot (8.5 - 8.65) = 14.3$$

$$\frac{\partial output\_y}{\partial y} = \frac{\partial (y^2 + 5y)}{\partial y} = 2y + 5 = 2 \cdot 1.36 + 5 = 7.72$$

$$\frac{\partial y}{\partial output\_h_1} = \frac{\partial (output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3)}{\partial output\_h_1} = w_5 = 0.1$$

$$\frac{\partial y}{\partial output\_h_2} = \frac{\partial (output\_h_1 \cdot w_5 + output\_h_2 \cdot w_6 + b_3)}{\partial output\_h_2} = w_6 = 0.3$$

$$\frac{\partial output\_h_1}{\partial h_1} = \frac{\partial ({h_1}^2 + 5 \cdot h_1)}{\partial h_1} = 2h_1 + 5 = 2 \cdot 0.26 + 5 = 5.52$$

$$\frac{\partial output\_h_2}{\partial h_2} = \frac{\partial ({h_2}^2 + 5 \cdot h_2)}{\partial h_2} = 2h_2 + 5 = 2 \cdot 0.61 + 5 = 6.22$$

$$\frac{\partial h_1}{\partial w_1} = \frac{\partial (x_1 \cdot w_1 + x_2 \cdot w_3 + b_1)}{\partial w_1} = x_1 = 0.2$$

$$\frac{\partial h_1}{\partial w_3} = \frac{\partial (x_1 \cdot w_1 + x_2 \cdot w_3 + b_1)}{\partial w_3} = x_2 = 0.3$$

$$\frac{\partial h_1}{\partial b_1} = \frac{\partial (x_1 \cdot w_1 + x_2 \cdot w_3 + b_1)}{\partial b_1} = 1$$

$$\frac{\partial h_2}{\partial w_2} = \frac{\partial(x_1 \cdot w_2 + x_2 \cdot w_4 + b_2)}{\partial w_2} = x_1 = 0.2$$

$$\frac{\partial h_2}{\partial w_4} = \frac{\partial(x_1 \cdot w_2 + x_2 \cdot w_4 + b_2)}{\partial w_4} = x_2 = 0.3$$

$$\frac{\partial h_2}{\partial b_2} = \frac{\partial(x_1 \cdot w_2 + x_2 \cdot w_4 + b_2)}{\partial b_2} = 1$$

ΚΑΤΑΒΑΣΗ ΚΑΤΑ ΚΛΙΣΗ

Αφού υπολογίστηκε η συνεισφορά της κάθε παραμέτρου στο συνολικό σφάλμα, το τελευταίο βήμα της εκπαιδευτικής διαδικασίας είναι η ενημέρωση όλων των παραμέτρων του νευρωνικού δικτύου. Η μέθοδος βελτιστοποίησης που επιλέχτηκε είναι η Στοχαστική Κατάβαση κατά Κλίση (Stochastic Gradient Descent - SGD) με ρυθμό μάθησης $\eta = 0.001$, όπου οι νέες παράμετροι υπολογίζονται σύμφωνα με τον τύπο $w' = w - \eta \cdot \frac{\partial E_{total}}{\partial w}$ όπου w είναι η εκάστοτε παράμετρος.

Επομένως, οι νέες τιμές των παραμέτρων του δικτύου είναι οι εξής:

$$\boldsymbol{w_1'} = w_1 - \eta \cdot \frac{\partial E_{total}}{\partial w_1} = 0.2 - 0.001 \cdot 12.19 = 0.18781 \approx \mathbf{0.19}$$

$$\boldsymbol{w_2'} = w_2 - \eta \cdot \frac{\partial E_{total}}{\partial w_2} = 0.3 - 0.001 \cdot 41.2 = 0.2588 \approx \mathbf{0.26}$$

$$\boldsymbol{w_3'} = w_3 - \eta \cdot \frac{\partial E_{total}}{\partial w_3} = 0.4 - 0.001 \cdot 18.28 = 0.38172 \approx \mathbf{0.38}$$

$$\boldsymbol{w_4'} = w_4 - \eta \cdot \frac{\partial E_{total}}{\partial w_4} = 0.5 - 0.001 \cdot 61.8 = 0.4382 \approx \mathbf{0.44}$$

$$\boldsymbol{w_5'} = w_5 - \eta \cdot \frac{\partial E_{total}}{\partial w_5} = 0.1 - 0.001 \cdot 151.24 = -.05124 \approx -\mathbf{0.05}$$

$$\boldsymbol{w_6'} = w_6 - \eta \cdot \frac{\partial E_{total}}{\partial w_6} = 0.3 - 0.001 \cdot 377.55 = -0.07755 \approx -\mathbf{0.08}$$

$$\boldsymbol{b_1'} = b_1 - \eta \cdot \frac{\partial E_{total}}{\partial b_1} = 0.1 - 0.001 \cdot 60.94 = 0.03906 \approx \mathbf{0.04}$$

$$\boldsymbol{b_2'} = b_2 - \eta \cdot \frac{\partial E_{total}}{\partial b_2} = 0.4 - 0.001 \cdot 206 = 0.194 \approx \mathbf{0.19}$$

$$\boldsymbol{b_3'} = b_3 - \eta \cdot \frac{\partial E_{total}}{\partial b_3} = 0.2 - 0.001 \cdot 110.4 = 0.0896 \approx \mathbf{0.09}$$

Με την ενημέρωση των παραμέτρων ολοκληρώνεται ένα εκπαιδευτικό βήμα (training step), το οποίο επαναλαμβάνεται μέχρι όλα τα στοιχεία να περάσουν από το δίκτυο σύμφωνα με

την παραπάνω διαδικασία, ολοκληρώνοντας έτσι μια εκπαιδευτική εποχή (training epoch). Υπενθυμίζεται ότι, το συγκεκριμένο εκπαιδευτικό βήμα του παραδείγματος αναλύει μόνο ένα στοιχείο των δεδομένων κάθε φορά που επαναλαμβάνεται, αλλά συνήθως συνιστάται η ανάλυση των δεδομένων σε παρτίδες.

Ειδικότερα, ο συγκεκριμένος πίνακας δεδομένων αποτελείται από 500 εγγραφές, επομένως αν ρυθμιστεί η εκπαίδευση του μοντέλου σε παρτίδες των 32 στοιχείων, τότε για να ολοκληρωθεί μια εποχή εκπαίδευσης θα χρειαζόταν $500/32 = 15.625 \approx 16$ βήματα, δηλαδή 16 επαναληπτικές διαδικασίες όπως την παραπάνω (διάδοση προς τα εμπρός + διάδοση προς τα πίσω + ενημέρωση παραμέτρων).

Επιπλέον, η μέθοδος βελτιστοποίησης που εφαρμόστηκε για την ενημέρωση των παραμέτρων αναφέρεται και ως πραγματική Στοχαστική Κατάβαση κατά Κλίση (true Stochastic Gradient Descent – true SGD), διότι επεξεργάζεται μόνο ένα στοιχείο σε κάθε επανάληψη, η περίπτωση εκπαίδευσης των δεδομένων σε παρτίδες αναφέρεται ως Στοχαστική Κατάβαση κατά Κλίση μίνι-παρτίδας (mini-batch Stochastic Gradient Descent – mini-batch SGD), ενώ η περίπτωση όπου όλα τα στοιχεία των δεδομένων αναλύονται ταυτόχρονα από το δίκτυο αναφέρεται ως Κατάβαση κατά Κλίση Παρτίδας (Batch Gradient Descent). Όσο περισσότερα στοιχεία επεξεργάζονται σε κάθε βήμα, τόσο πιο ακριβής θα είναι η ενημέρωση των παραμέτρων, αλλά και τόσο πιο απαιτητική θα είναι η ανάγκη υπολογιστικής ισχύος. Προφανώς, υπάρχουν πολλές μέθοδοι βελτιστοποίησης, και η επιλογή τους εξαρτάται και πάλι από το πρόβλημα που ερευνάται και την επίδοση του μοντέλου που παρατηρείται.

ΈΛΕΓΧΟΣ ΝΕΟΥ ΣΦΑΛΜΑΤΟΣ

Θα εφαρμοστεί το βήμα της Διάδοσης προς τα Εμπρός και ο υπολογισμός του σφάλματος του παραπάνω παραδείγματος με τις νέες τιμές των παραμέτρων.

Για τους νευρώνες του κρυμμένου επιπέδου οι μετασχηματισμοί που προκύπτουν είναι:

$$\boldsymbol{h_1} = x_1 \cdot w_1' + x_2 \cdot w_3' + b_1' = 0.2 \cdot 0.19 + 0.3 \cdot 0.38 + 0.04 = 0.192 \approx \mathbf{0.19}$$

$$\boldsymbol{h_2} = x_1 \cdot w_2' + x_2 \cdot w_4' + b_2' = 0.2 \cdot 0.26 + 0.3 \cdot 0.44 + 0.19 = 0.374 \approx \mathbf{0.37}$$

Με την εφαρμογή της συνάρτησης ενεργοποίησης οι τελικές έξοδοι του επιπέδου είναι:

$$\boldsymbol{output\_h_1} = {h_1}^2 + 5 \cdot h_1 = 0.19^2 + 5 \cdot 0.19 = 0.9861 \approx \mathbf{0.99}$$

$$\boldsymbol{output\_h_2} = {h_2}^2 + 5 \cdot h_2 = 0.37^2 + 5 \cdot 0.37 = 1.9869 \approx \mathbf{1.99}$$

Αντίστοιχα, για τον νευρώνα του επιπέδου εξόδου ισχύουν οι παρακάτω μετασχηματισμοί:

$$y = output\_h_1 \cdot w_5' + output\_h_2 \cdot w_6' + b_3' = 0.99 \cdot (-0.05) + 1.99 \cdot (-0.08) + 0.09$$
$$= -0.1187 \approx -0.12$$

$$\boldsymbol{output\_y} = y^2 + 5 \cdot y = (-0.12)^2 + 5 \cdot (-0.12) = -0.5856 \approx \mathbf{-0.59}$$

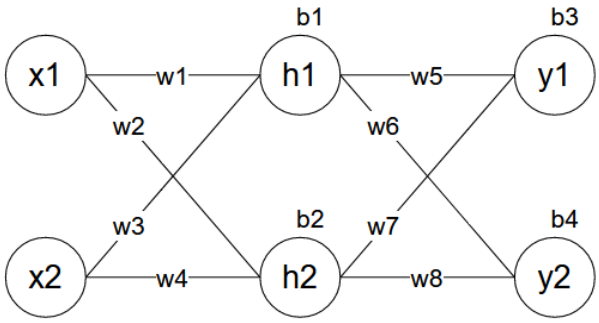
Το συνολικό σφάλμα από τη συνάρτηση απώλειας είναι:

$$\boldsymbol{E_{total}} = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = (Y - output\_y)^2 = (1.5 - (-0.59))^2$$
$$= 2.09^2 = 4.3681 \approx \mathbf{4.37}$$

Επομένως, με μόνο μια επανάληψη για ένα στοιχείο του πίνακα δεδομένων, το μοντέλο MLP με τη συγκεκριμένη αρχιτεκτονική μείωσε το σφάλμα από την τιμή 51.12 στην 4.37.

2.2.4 Σύνθετο Παράδειγμα Εκπαίδευσης

Για την καλύτερη κατανόηση του Αλγορίθμου της Οπισθοδιάδοσης και της πρακτικής διαδικασίας του κανόνα της αλυσίδας για τον υπολογισμό των μερικών παραγώγων, έστω το παρακάτω νευρωνικό δίκτυο MLP:



Εικόνα 7. Νευρωνικό Δίκτυο Σύνθετου Παραδείγματος

Για τους νευρώνες του κρυμμένου επιπέδου οι μετασχηματισμοί που προκύπτουν είναι:

$$\boldsymbol{h_1} = x_1 \cdot w_1 + x_2 \cdot w_3 + b_1$$

$$\boldsymbol{h_2} = x_1 \cdot w_2 + x_2 \cdot w_4 + b_2$$

Με την εφαρμογή της συνάρτησης ενεργοποίησης οι τελικές έξοδοι του επιπέδου είναι:

$$\boldsymbol{output\_h_1} = {h_1}^2 + 5 \cdot h_1$$

$$\boldsymbol{output\_h_2} = {h_2}^2 + 5 \cdot h_2$$

Αντίστοιχα, για τους νευρώνες του επιπέδου εξόδου ισχύουν:

$$\boldsymbol{y_1} = output\_h_1 \cdot w_5 + output\_h_2 \cdot w_7 + b_3$$

$$\boldsymbol{y_2} = output\_h_1 \cdot w_6 + output\_h_2 \cdot w_8 + b_4$$

Με την εφαρμογή της συνάρτησης ενεργοποίησης οι τελικές έξοδοι του επιπέδου είναι:

$$\boldsymbol{output\_y_1} = {y_1}^2 + 5 \cdot y_1$$

$$\boldsymbol{output\_y_2} = {y_2}^2 + 5 \cdot y_2$$

Ολοκληρώθηκε η Διάδοση προς τα Εμπρός και το συνολικό σφάλμα της συνάρτησης απώλειας είναι:

$$\boldsymbol{E_{total}} = \frac{1}{n} \cdot \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = \frac{(y_1 - output\_y_1)^2 + (y_2 - output\_y_2)^2}{2}$$

Σύμφωνα με την διαδικασία της Διάδοσης προς τα Πίσω, για τις παραμέτρους του επιπέδου εξόδου και τη συνεισφορά τους στο σφάλμα έχουμε:

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_5}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial w_5}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_7}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial w_7}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial b_3}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial b_3}$$

Και:

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_6}} = \frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial w_6}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_8}} = \frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial w_8}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial b_4}} = \frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial b_4}$$

Για τις παραμέτρους του κρυμμένου επιπέδου έχουμε:

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_1}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1} + \frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_1}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial w_3}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_3} + \frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial w_3}$$

$$\frac{\boldsymbol{\partial E_{total}}}{\boldsymbol{\partial b_1}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1} +$$

$$\frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_1} \cdot \frac{\partial output\_h_1}{\partial h_1} \cdot \frac{\partial h_1}{\partial b_1}$$

Και:

$$\boldsymbol{\frac{\partial E_{total}}{\partial w_2}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2} +$$

$$\frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_2}$$

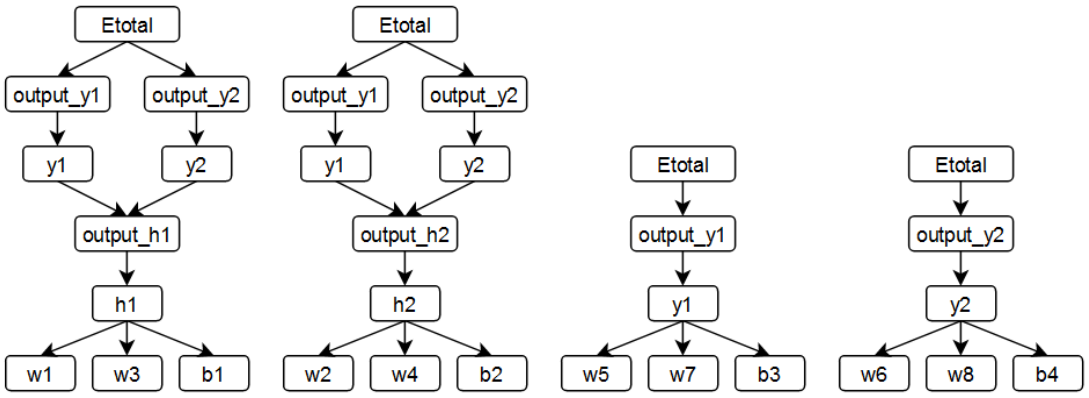
$$\boldsymbol{\frac{\partial E_{total}}{\partial w_4}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_4} +$$

$$\frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial w_4}$$

$$\boldsymbol{\frac{\partial E_{total}}{\partial b_2}} = \frac{\partial E_{total}}{\partial output\_y_1} \cdot \frac{\partial output\_y_1}{\partial y_1} \cdot \frac{\partial y_1}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2} +$$

$$\frac{\partial E_{total}}{\partial output\_y_2} \cdot \frac{\partial output\_y_2}{\partial y_2} \cdot \frac{\partial y_2}{\partial output\_h_2} \cdot \frac{\partial output\_h_2}{\partial h_2} \cdot \frac{\partial h_2}{\partial b_2}$$

Επομένως, είναι προφανές ότι όσο αυξάνεται το πλήθος των νευρώνων και των επιπέδων του νευρωνικού δικτύου, τόσο αυξάνεται και η πολυπλοκότητα των πράξεων, και για την καλύτερη κατανόηση της εφαρμογής του κανόνα της αλυσίδας στο παραπάνω παράδειγμα, δίνεται το διάγραμμα:



Εικόνα 8. Απεικόνιση Κανόνα Αλυσίδας Σύνθετου Παραδείγματος

2.3 Μακρά Βραχυπρόθεσμη Μνήμη (Long Short-Term Memory - LSTM)

Τα νευρωνικά δίκτυα MLP, παρά τη συμβατική μοντελοποίηση που προσφέρουν, είναι αξιόπιστα, ευέλικτα ως προς την αρχιτεκτονική τους, και εφαρμόζονται σε πληθώρα προβλημάτων, ειδικά σε περιπτώσεις δεδομένων πινάκων (tabular data), όταν το πλήθος τους είναι επαρκές και αντιπροσωπευτικό του προβλήματος. Ωστόσο, εμφανίζουν αδυναμίες, ειδικά σε περιπτώσεις δεδομένων ακολουθιών, καθώς δεν είναι ικανά να αναγνωρίσουν τυχόν εξαρτήσεις μεταξύ των στοιχείων, όπου η σειρά των δεδομένων έχει σημασία για το εκάστοτε πρόβλημα. Για παράδειγμα, στην περίπτωση μιας χρονοσειράς, όπου πληροφορίες για ένα ή και περισσότερα στοιχεία μπορεί να έχουν σημασία για ένα στοιχείο αρκετά βήματα αργότερα στην ακολουθία.

Αναφέρθηκε ότι τα επίπεδα αποτελούν το κύριο δομικό στοιχείο των Τεχνητών Νευρωνικών Δικτύων, και οι λειτουργικές δυνατότητες των νευρώνων τους χαρακτηρίζουν το ίδιο το μοντέλο Βαθιάς Μάθησης, ως προς τον σκοπό για τον οποίο αναπτύσσεται, βάσει του προβλήματος που ερευνάται και των δεδομένων που πρέπει να αναλυθούν. Επομένως, για την αντιμετώπιση του προβλήματος αναγνώρισης εξαρτήσεων μεταξύ των δεδομένων, επινοήθηκαν τα Επαναληπτικά Νευρωνικά Δίκτυα (Recurrent Neural Network – RNN). Τα RNN εφαρμόζουν, μέσω της αρχιτεκτονικής τους, έναν εσωτερικό βρόγχο ανατροφοδότησης (feedback loop), επιτρέποντας την επεξεργασία του κάθε στοιχείου (element) μιας ακολουθίας (sequence) λαμβάνοντας υπόψη όλα τα προηγούμενα στοιχεία της ακολουθίας που έχουν αναλυθεί. Συνεπώς, τα δίκτυα RNN διατηρούν σε κάθε βήμα επεξεργασίας μια συνεχή κατάσταση μνήμης (memory state) με παρελθοντικές πληροφορίες.

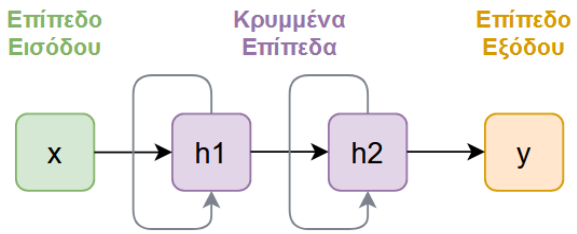
Τα RNN διαφοροποιούνται από τα δίκτυα MLP ως προς την επεξεργασία των δεδομένων, καθώς τα MLP ακολουθούν το πρότυπο των Νευρωνικών Δικτύων Προς τα Εμπρός Διάδοσης (Feed-Forward Neural Network – FFNN), δηλαδή τα δεδομένα ρέουν προς μια κατεύθυνση, ενώ στην περίπτωση των RNN, η ανάλυση των δεδομένων υλοποιείται με βρόγχους επανάληψης. Επιπλέον, τα MLP διαχειρίζονται δεδομένα πινάκων, δηλαδή κάθε στοιχείο ενός πίνακα δεδομένων ανά γραμμή, ενώ τα RNN διαχειρίζονται ακολουθίες δεδομένων, δηλαδή πολλά διαδοχικά στοιχεία-γραμμές δεδομένων τη φορά, μέσω χρονικών βημάτων (timesteps).

Τελικά, η απλοϊκή αρχιτεκτονική των RNN εξελίχθηκε και πλέον προτιμούνται τεχνικές που επεξεργάζονται ακολουθίες με πιο βέλτιστο τρόπο και είναι ικανές να αναγνωρίσουν

εξαρτήσεις μεταξύ των δεδομένων σε βάθος χρόνου. Αυτοί οι εναλλακτικοί τύποι επαναληπτικών νευρωνικών δικτύων είναι τα νευρωνικά δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης (Long Short-Term Memory – LSTM), και τα νευρωνικά δίκτυα Επαναλαμβανόμενης Μονάδας με Πύλη (Gated Recurrent Unit – GRU). Ως προς τη σύγκρισή τους, τα δίκτυα LSTM και GRU είναι πιο βελτιωμένες εκδοχές των RNN, ενώ παράλληλα το GRU αποτελεί μια πιο απλοϊκή εφαρμογή των εννοιών του LSTM.

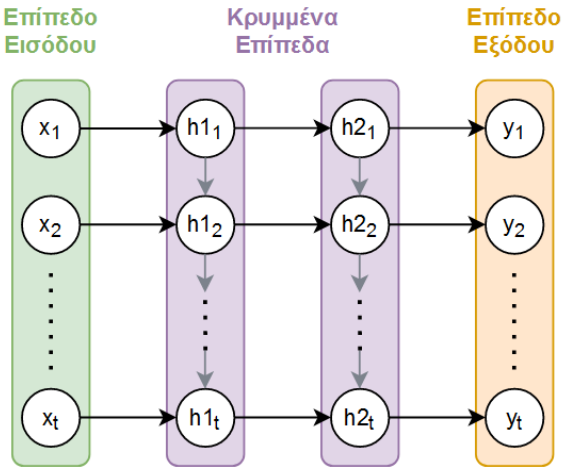
2.3.1 Αρχιτεκτονική RNN

Τα δίκτυα RNN ακολουθούν την τυπική αρχιτεκτονική των Τεχνητών Νευρωνικών Δικτύων, δηλαδή αποτελούνται από ένα επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα και ένα επίπεδο εξόδου. Ωστόσο, κάθε κρυμμένο επίπεδο RNN δεν αποτελείται από πολλούς ανεξάρτητους νευρώνες, όπως στην περίπτωση των MLP. Αντιθέτως, μπορούμε να θεωρήσουμε κάθε κρυμμένο επίπεδο ως έναν μόνο νευρώνα (cell ή memory cell), ο οποίος επεξεργάζεται τα δεδομένα εσωτερικά με μια επαναληπτική διαδικασία.



Εικόνα 9. Απεικόνιση Αρχιτεκτονικής Δικτύου RNN

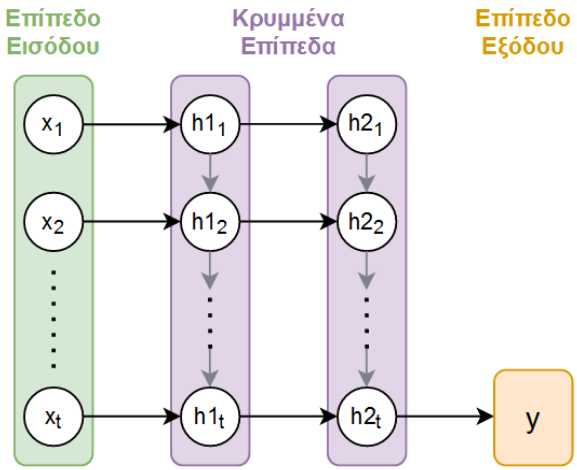
Αν αναλυθεί το παραπάνω δίκτυο στα επιμέρους στοιχεία του, όπως φαίνεται παρακάτω, αναγνωρίζεται καλύτερα η λειτουργικότητά του. Συγκεκριμένα, η είσοδος x συμβολίζει μια ακολουθία από διαδοχικά στοιχεία-γραμμές του πίνακα δεδομένων πλήθους t, όπου κάθε στοιχείο της ακολουθίας αναλύεται σε διαφορετικό χρονικό βήμα από τον νευρώνα RNN του επόμενου κρυμμένου επιπέδου.



Εικόνα 10. Επιμέρους Απεικόνιση Αρχιτεκτονικής Δικτύου RNN με Έξοδο Ακολουθίας

Επίσης, ο ίδιος ο νευρώνας, σε κάθε επαναληπτικό βήμα διατηρεί μια κατάσταση μνήμης (state), η οποία διαδίδεται διαδοχικά σε κάθε επόμενο βήμα (γκρι διαδρομή στις Εικόνες), μεταφέροντας παρελθοντικές πληροφορίες για κάθε προηγούμενο στοιχείο της ακολουθίας στα επόμενα. Επιπλέον, αυτή η κατάσταση μνήμης αποτελεί παράλληλα και έξοδο του νευρώνα για κάθε επαναληπτικό βήμα, με αποτέλεσμα να προκύπτει ως έξοδος του επιπέδου RNN επίσης μια ακολουθία μήκους t.

Είναι σημαντικό να τονιστεί ότι είναι απαραίτητος ο μετασχηματισμός των δεδομένων εισόδου σε ακολουθίες, όταν το επόμενο επίπεδο στο δίκτυο είναι και αυτό τύπου RNN. Ωστόσο, η έξοδος που μπορεί να διαβιβάσει ένα επίπεδο RNN μπορεί να έχει τη μορφή ακολουθίας, όπως παρουσιάζεται παραπάνω, ή μπορεί να είναι ένα διάνυσμα, όπου ως έξοδος του επιπέδου RNN μπορεί να θεωρηθεί μόνο η έξοδος του τελευταίου βήματος του νευρώνα.

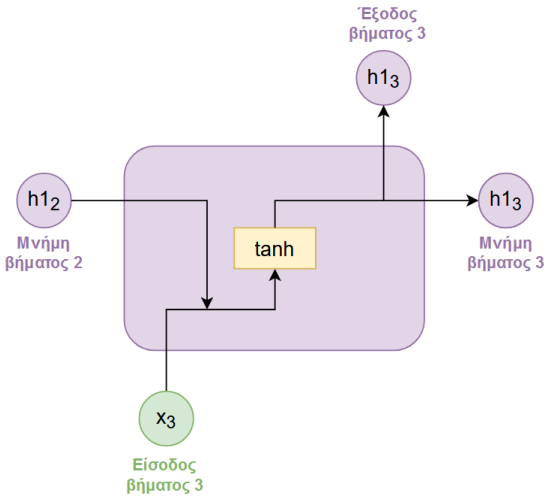


Εικόνα 11. Επιμέρους Απεικόνιση Αρχιτεκτονικής Δικτύου RNN με Έξοδο Διανύσματος

Για καλύτερη κατανόηση των δεδομένων εισόδου και εξόδου, έστω ένας πίνακας δεδομένων με 60 εγγραφές (γραμμές) και 3 χαρακτηριστικά (στήλες), επομένως η τωρινή διάσταση των δεδομένων είναι $(60x3)$. Αν θέλουμε να μετασχηματίσουμε τα δεδομένα σε ακολουθίες των 4 διαδοχικών στοιχείων, τότε ο νέος πίνακας δεδομένων θα έχει διάσταση $(57x4x3)$, όπου κάθε δείγμα από τα 57 είναι μια ακολουθία 4 στοιχείων με 3 επιμέρους μεταβλητές. Συγκριμένα, το 1ο δείγμα περιλαμβάνει τα στοιχεία 1-2-3-4 του αρχικού πίνακα δεδομένων, το 2ο δείγμα τα στοιχεία 2-3-4-5, το 3ο δείγμα τα στοιχεία 3-4-5-6, κ.ο.κ.

Επιπλέον, για να γίνει η σύνδεση με τα παραπάνω διαγράμματα, η πράσινη είσοδος συμβολίζει μόνο ένα από τα 57 δείγματα, οι μεταβλητές x_i με $i = 1,2,3,4$ συμβολίζουν τα στοιχεία-γραμμές του αρχικού πίνακα, το πλήθος τους είναι όσο έχει οριστεί το μήκος της ακολουθίας (εδώ $t = 4$), και κάθε μεταβλητή x_i μεμονωμένη αποτελεί ένα διάνυσμα μήκους 3, όσα και τα χαρακτηριστικά των δεδομένων.

Μια τελευταία και πιο σημαντική ανάλυση αφορά στην αρχιτεκτονική του ίδιου του νευρώνα RNN. Παρακάτω δίνεται μια ειδική αναπαράσταση κατά την επεξεργασία των δεδομένων κατά το 3ο επαναληπτικό βήμα.



Εικόνα 12. Απεικόνιση Αρχιτεκτονικής Νευρώνα RNN

Συγκεκριμένα, σε κάθε επαναληπτικό βήμα t, τροφοδοτούνται ως είσοδοι στον νευρώνα το στοιχείο t της ακολουθίας, καθώς και η έξοδος του νευρώνα από το προηγούμενο βήμα $t-1$, η οποία αποτελεί και την κατάσταση μνήμης του νευρώνα RNN. Η αρχικοποίηση της κατάσταση μνήμης, ειδικά για το πρώτο βήμα, δηλαδή της τιμής $h1_0$, ρυθμίζεται αναλόγως κατά την ανάπτυξη του νευρωνικού δικτύου, αλλά συνήθως ορίζεται να είναι μηδέν.

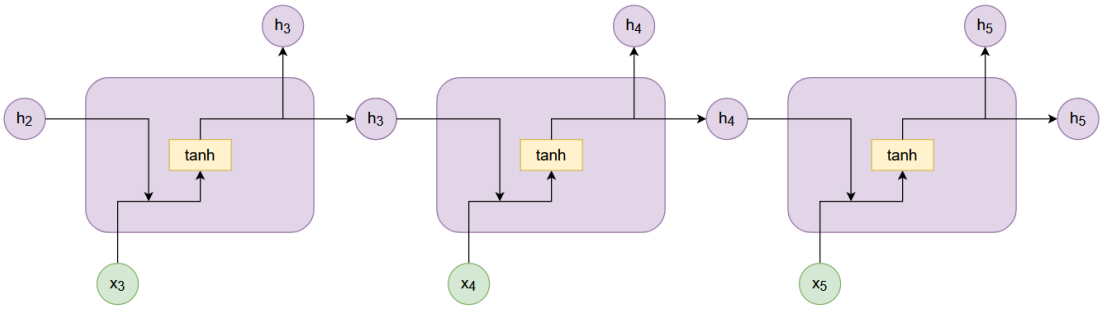
Στη συνέχεια, οι δυο είσοδοι επεξεργάζονται με τον ίδιο τρόπο όπως σε ένα συμβατικό δίκτυο MLP, δηλαδή πολλαπλασιάζονται με παραμέτρους βαρών, τους ενσωματώνεται μια παράμετρος σταθερού όρου, και τελικά μετασχηματίζονται μέσω μιας συνάρτησης ενεργοποίησης, που συνήθως επιλέγεται να είναι η υπερβολική εφαπτομένη (hyperbolic tangent – $tanh$) με τον μαθηματικό τύπο:

$$\tanh(x) = \frac{2}{1+e^{-2x}} - 1$$

Όπου:

- x είναι η ολοκληρωμένη διανυσματική είσοδος που προκύπτει από τις δυο εισόδους του νευρώνα RNN μετά τον μετασχηματισμό τους από τις παραμέτρους βαρών και σταθερών όρων

Στο τέλος, η μετασχηματισμένη έξοδος του βήματος t, θα αποτελέσει έξοδο του νευρώνα για το βήμα t, αλλά και είσοδο για το επόμενο βήμα $t+1$ του νευρώνα, η οποία θα συνδυαστεί με το επόμενο στοιχείο $t+1$ της ακολουθίας εισόδου για να επαναληφθεί η παραπάνω διαδικασία, όπως φαίνεται στην παρακάτω εικόνα.



Εικόνα 13. Απεικόνιση Επαναληπτικών Χρονικών Βημάτων Νευρώνα RNN

Η εκπαίδευση ενός νευρωνικού δικτύου RNN υλοποιείται με ανάλογο τρόπο όπως περιγράφηκε και στην περίπτωση του δικτύου MLP. Η ακολουθία περνάει μέσα από το δίκτυο, επεξεργάζεται και αναλύεται κατάλληλα, μέχρι να προκύψει το τελικό αποτέλεσμα του επιπέδου εξόδου. Στη συνέχεια, υπολογίζεται το σφάλμα της πρόβλεψης του δικτύου σύμφωνα με μια προεπιλεγμένη συνάρτηση απώλειας, το οποίο διαδίδεται προς τα πίσω στο νευρωνικό δίκτυο για να υπολογιστεί η συνεισφορά της κάθε παραμέτρου σε αυτό. Τελικά, με την εφαρμογή ενός αλγορίθμου βελτιστοποίησης Κατάβασης κατά Κλίσης, όλες οι παράμετροι ενημερώνονται κατάλληλα έτσι ώστε να επιτευχθεί η ελαχιστοποίηση του συνολικού σφάλματος.

Να σημειωθεί ότι, ο Αλγόριθμος της Οπισθοδιάδοσης, και συγκεκριμένα ο κανόνας της αλυσίδας των μερικών παραγώγων, εφαρμόζεται για όλα τα επαναληπτικά βήματα του νευρώνα του κάθε επιπέδου RNN, και μάλιστα λαμβάνεται υπόψη και η διάσταση της τελικής εξόδου του δικτύου (ακολουθία ή διάνυσμα), όπως αναφέρονται στις Εικόνες 10 και 11, καθώς έξοδοι που δεν συμμετέχουν στο αποτέλεσμα, δεν λαμβάνονται υπόψη.

2.3.2 Αρχιτεκτονική LSTM

Παρά το γεγονός ότι τα συμβατικά δίκτυα RNN παρουσιάζουν καλές επιδόσεις στην πρόβλεψη ακολουθιών, αντιμετωπίζουν δυσκολίες σε προβλήματα όπου απαιτείται η ανάλυση δεδομένων μεγάλου μήκους ακολουθιών ή χρονοσειρών. Ειδικότερα, όσο μεγαλύτερο είναι το μήκος των ακολουθιών, τόσο περισσότερα είναι τα εσωτερικά επαναληπτικά βήματα των νευρώνων RNN, κάτι που έχει ως αποτέλεσμα ένα μέρος της πληροφορίας από τα αρχικά βήματα της ακολουθίας να μην καταφέρνει να μεταδοθεί αρκετά βήματα παρακάτω, και συνεπώς η τελική κατάσταση μνήμης του νευρώνα να είναι ελλιπής. Το συγκεκριμένο ζήτημα είναι ιδιαίτερα αποτρεπτικό στην εκπαίδευση του

δικτύου, αν αναλογιστεί κανείς ότι τα δίκτυα RNN βασίζονται στη γνώση παρελθοντικών μοτίβων των δεδομένων για να προχωρήσουν σε ακριβείς προβλέψεις για μια μελλοντική κατάσταση.

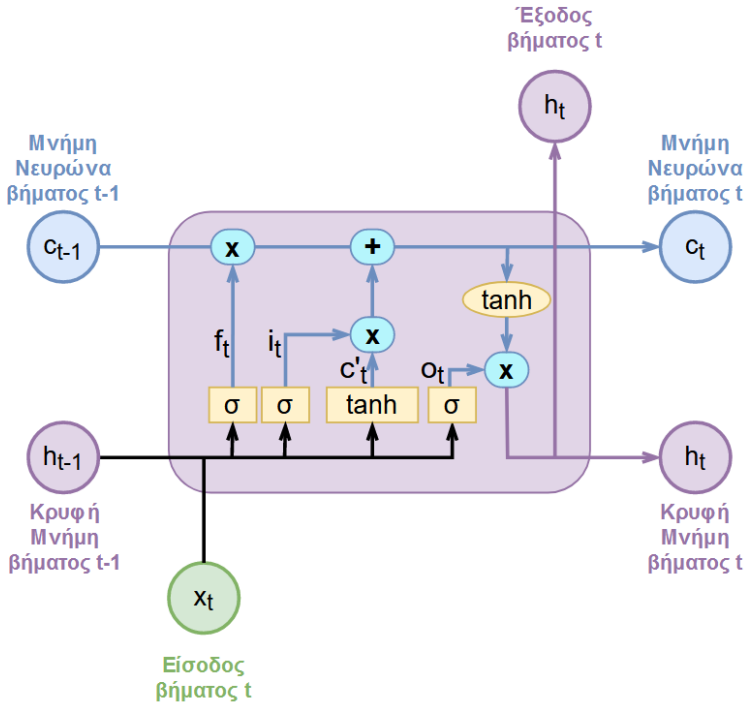
Επιπλέον, ακόμα ένα πρόβλημα έγκειται κατά τη διαδικασία της εκπαίδευσης και τον υπολογισμό των μερικών παραγώγων του σφάλματος, και αναφέρεται ως το πρόβλημα των «Εξαφανιζόμενων ή Ανατινασσόμενων Παραγώγων» (Vanishing or Exploding Gradients), ή αλλιώς ως πρόβλημα των «Ασταθών Παραγώγων» (Unstable Gradients), το οποίο συναντάται γενικότερα στα Τεχνητά Νευρωνικά Δίκτυα. Κατά την προς τα πίσω διάδοση του σφάλματος, και συγκεκριμένα κατά τη διάδοσή του στα πιο χαμηλά επίπεδα των δικτύων, συχνά οι τιμές των παραγώγων ολοένα και μικραίνουν, με αποτέλεσμα η ενημέρωση των αντίστοιχων παραμέτρων να είναι αμελητέα σε σχέση με τις παραμέτρους των υψηλότερων επιπέδων, το οποίο συμβάλει πολλές φορές στην αδυναμία εκπαίδευσης του μοντέλου. Επιπλέον, σε κάποιες περιπτώσεις μπορεί να συμβεί το αντίθετο, δηλαδή οι τιμές των παραγώγων να μεγαλώσουν τόσο πολύ στα χαμηλότερα επίπεδα που η ενημέρωση των παραμέτρων τους να γίνεται με τιμές υπερβολικά μεγαλύτερες.

Το παραπάνω πρόβλημα είναι ιδιαίτερα εμφανές σε βαθιά νευρωνικά δίκτυα πολλών επιπέδων, καθώς οι περισσότεροι μετασχηματισμοί, στην προσπάθειά τους να αναγνωρίσουν πιο πολύπλοκα μοτίβα στα δεδομένα, οδηγούν σε περισσότερο «θόρυβο» που πρέπει να αντιμετωπίσει το μοντέλο κατά την εκπαίδευσή του. Ωστόσο, το παραπάνω πρόβλημα εμφανίζεται συχνότερα σε περιπτώσεις δικτύων επανάληψης, όπου οι παράγωγοι μεταδίδονται σε όλα τα επαναληπτικά βήματα ενός νευρώνα, επομένως είναι ευκολότερο να προκύψει αστάθεια παραγώγων. Υπάρχουν γενικότερες τεχνικές για τον μετριασμό του προβλήματος αυτού, αλλά ειδικότερα για τα δίκτυα επαναλαμβανόμενου τύπου, τα συμβατικά RNN εξελίχθηκαν με τις βελτιωμένες εκδοχές των νευρώνων LSTM και GRU, οι οποίες αντιμετωπίζουν τόσο το πρόβλημα των ασταθών παραγώγων, όσο και το πρόβλημα απώλειας πληροφορίας σε ακολουθίες μεγάλου μήκους, το οποίο αναφέρθηκε προηγουμένως.

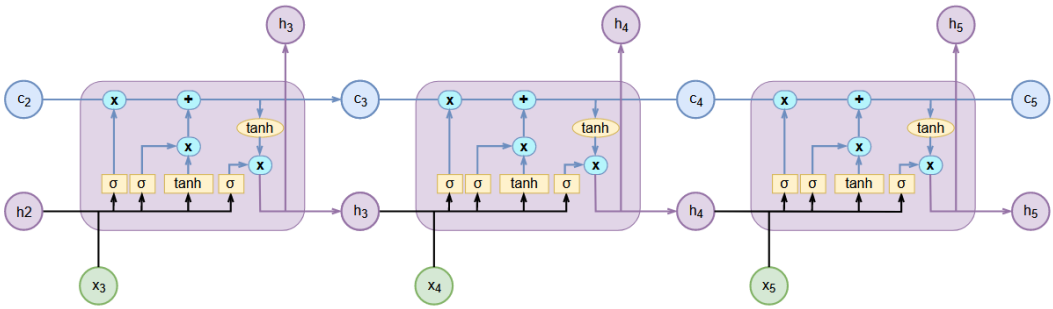
Συγκεκριμένα για τα νευρωνικά δίκτυα LSTM, η υπεροχή τους βασίζεται στον τρόπο που διαχειρίζονται τη μνήμη του νευρώνα, αλλά και τις επιμέρους διαδικασίες των μαθηματικών μετασχηματισμών. Αναφέρθηκε ότι ένας νευρώνας RNN έχει μόνο μια κατάσταση μνήμης (state) η οποία διαδίδεται σε κάθε επαναληπτικό βήμα. Αντιθέτως, ένας νευρώνας LSTM πέρα από αυτήν την κατάσταση μνήμης, η οποία εδώ αναφέρεται ως κρυφή μνήμη (hidden state) ή ως βραχυπρόθεσμη μνήμη (short-term memory), διατηρεί και μια επιπλέον κατάσταση μνήμης η οποία ονομάζεται μνήμη του νευρώνα (cell state ή

carry state) ή μακροπρόθεσμη μνήμη (long-term memory). Όπως περιγράφεται από τα ονόματα των ορισμών τους, η κρυφή μνήμη διαχειρίζεται εξαρτήσεις μεταξύ των κοντινών χρονικών βημάτων μιας ακολουθίας, ενώ η μνήμη του νευρώνα αποθηκεύει πληροφορίες από όλα τα προηγούμενα βήματα, έτσι ώστε να χρησιμοποιηθούν σε επόμενα και να μη χαθούν τυχόν εξαρτήσεις κατά την εκπαίδευση του μοντέλου.

Παρακάτω παρουσιάζεται η αρχιτεκτονική ενός νευρώνα LSTM, καθώς και η επαναληπτική διαδικασία που πραγματοποιείται στο εσωτερικό του, και ακολουθεί αναλυτικά η επεξήγηση όλων μαθηματικών μετασχηματισμών για ένα χρονικό βήμα t.



Εικόνα 14. Απεικόνιση Αρχιτεκτονικής Νευρώνα LSTM



Εικόνα 15. Απεικόνιση Επαναληπτικών Χρονικών Βημάτων Νευρώνα LSTM

Σε κάθε χρονικό βήμα t, ο νευρώνας LSTM δέχεται ως είσοδο το στοιχείο της ακολουθίας που βρίσκεται στη θέση t, την κρυφή μνήμη h_{t-1}, η οποία αποτελεί την έξοδο του προηγούμενου χρονικού βήματος $t-1$, και τη μακροπρόθεσμη μνήμη c_{t-1} του νευρώνα, η οποία κρατάει πληροφορίες μέχρι και το χρονικό βήμα $t-1$. Η έξοδος του νευρώνα LSTM για το χρονικό βήμα t είναι η νέα κρυφή ή βραχυπρόθεσμη μνήμη h_t, και η περαιτέρω διάδοση της μακροπρόθεσμης μνήμης c_t. Να σημειωθεί ότι η έξοδος που θα προχωρήσει στο επόμενο επίπεδο του δικτύου είναι η h_t, ενώ η έξοδος c_t συμβάλει μόνο στη διάδοση σημαντικών παρελθοντικών πληροφοριών στα εσωτερικά επαναληπτικά βήματα.

Είναι προφανής η πολυπλοκότητα της εσωτερικής αρχιτεκτονικής του νευρώνα LSTM, σε σχέση με την αντίστοιχη απλούστερη του RNN. Αρχικά, οι είσοδοι x_t και h_{t-1} συνενώνονται και περνάνε από τέσσερις διαφορετικούς μετασχηματισμούς, οι οποίοι είναι ουσιαστικά τέσσερα πλήρως διασυνδεδεμένα νευρωνικά δίκτυα του ενός επιπέδου, δηλαδή καθένα εφαρμόζει την πράξη του πολλαπλασιασμού των εισόδων με παραμέτρους βαρών (weights) και την περαιτέρω ενσωμάτωση παραμέτρων σταθερών όρων (biases), εξάγοντας τελικά το αποτέλεσμα μέσω μιας συνάρτησης ενεργοποίησης. Οι έξοδοι αυτών των τεσσάρων εσωτερικών νευρωνικών δικτύων υπολογίζονται ως εξής:

$$f_t = \sigma(U_f \cdot h_{t-1} + W_f \cdot x_t + b_f)$$

$$i_t = \sigma(U_i \cdot h_{t-1} + W_i \cdot x_t + b_i)$$

$$c'_t = tanh(U_{c'} \cdot h_{t-1} + W_{c'} \cdot x_t + b_{c'})$$

$$o_t = \sigma(U_o \cdot h_{t-1} + W_o \cdot x_t + b_o)$$

Όπου:

- t είναι το τωρινό χρονικό βήμα, όπως και η θέση του στοιχείου της ακολουθίας
- h_{t-1} είναι η έξοδος του νευρώνα του προηγούμενου χρονικού βήματος
- x_t είναι το στοιχείο της ακολουθίας που επεξεργάζεται
- U_k είναι ο πίνακας των βαρών που εφαρμόζονται στην είσοδο h_{t-1}
- W_k είναι ο πίνακας των βαρών που εφαρμόζονται στην είσοδο x_t
- b_k είναι το διάνυσμα των σταθερών όρων κάθε εσωτερικού νευρωνικού δικτύου
- σ είναι η σιγμοειδής (sigmoid) συνάρτηση ως συνάρτηση ενεργοποίησης
- $tanh$ είναι η υπερβολική εφαπτομένη ως συνάρτηση ενεργοποίησης
- f_t είναι η πύλη απώλειας
- i_t είναι η πύλη εισόδου
- c'_t είναι η προσωρινή αναλυμένη είσοδος του νευρώνα
- o_t είναι η πύλη εξόδου

Σημαντική διαφοροποίηση των νευρώνων LSTM, πέρα από τη ύπαρξη δυο διαφορετικών καταστάσεων μνήμης (βραχυπρόθεσμη και μακροπρόθεσμη), είναι η επεξεργασία των δεδομένων μέσω τριών πυλών μνήμης, ειδικότερα της πύλης απώλειας (forget gate), της πύλης εισόδου (input fate) και της πύλης εξόδου (output gate), όπου:

- Η πύλη απώλειας f_t αναλαμβάνει να διαγράψει από την μακροπρόθεσμη μνήμη c_{t-1} εκείνες τις πληροφορίες που πλέον είναι ασήμαντες, βάσει των νέων πληροφοριών h_{t-1} και x_t που εισάγονται στον νευρώνα για το χρονικό βήμα t.
- Η πύλη εισόδου i_t αναλαμβάνει να ενσωματώσει στην μακροπρόθεσμη μνήμη c_{t-1} μόνο τις σημαντικότερες από τις νέες πληροφορίες των h_{t-1} και x_t, οι οποίες εισάγονται στον νευρώνα και έχουν προηγουμένως επεξεργαστεί και αναλυθεί κατάλληλα για το χρονικό βήμα t.
- Η πύλη εξόδου o_t αναλαμβάνει να εξάγει από την τελική μακροπρόθεσμη μνήμη c_t εκείνες την πληροφορίες που θα αποτελέσουν την έξοδο για την βραχυπρόθεσμη μνήμη h_t του νευρώνα για το χρονικό βήμα t.

Επίσης, από τα τέσσερα εσωτερικά μονοεπίπεδα νευρωνικά δίκτυα, εκείνο που εφαρμόζει την υπερβολική εφαπτομένη $tanh$ ως συνάρτηση ενεργοποίησης με έξοδο την c_t', θεωρείται το «βασικό» δίκτυο του νευρώνα LSTM, διότι το αποτέλεσμά του αποτελεί την πιθανή νέα πληροφορία που θα ενσωματωθεί στη μακροπρόθεσμη μνήμη c_{t-1} για την τελική εξαγωγή της νέας μακροπρόθεσμης μνήμης c_t του νευρώνα για το χρονικό βήμα t.

Επιπλέον, οι δυο έξοδοι του νευρώνα για το χρονικό βήμα t υπολογίζονται ως εξής:

$$c_t = c_{t-1} \cdot f_t + c_t' \cdot i_t$$

$$h_t = \tanh(c_t) \cdot o_t$$

Όπου:

- c_{t-1} είναι η προηγούμενη μακροπρόθεσμη μνήμη
- c_t είναι η ενημερωμένη μακροπρόθεσμη μνήμη
- h_t είναι η έξοδος και βραχυπρόθεσμη μνήμη του νευρώνα για το χρονικό βήμα t

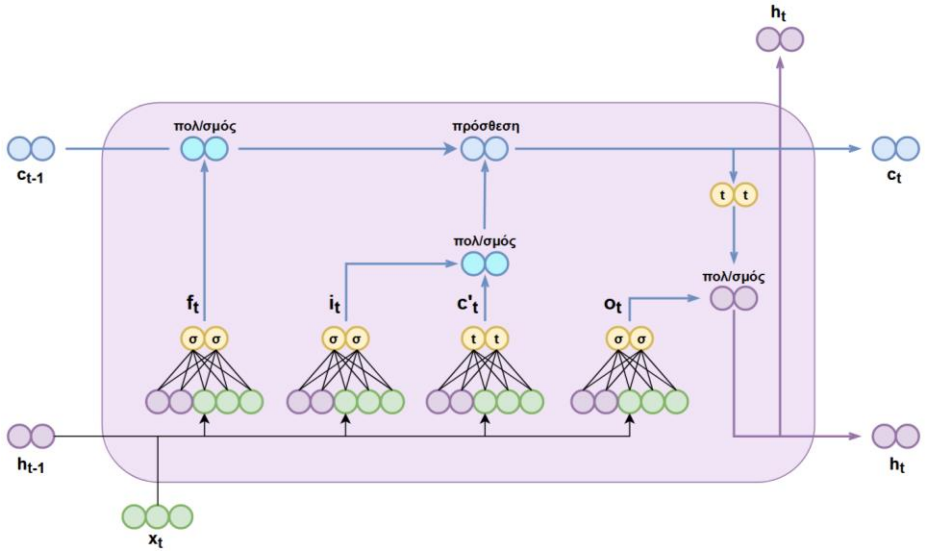
Πρέπει να τονιστεί ότι, τα αποτελέσματα των τριών πυλών f_t, i_t και o_t εξάγονται μετά από εφαρμογή της σιγμοειδούς συνάρτησης ενεργοποίησης, το οποίο σημαίνει ότι οι τελικές τιμές τους κυμαίνονται μεταξύ του μηδέν και της μονάδας. Επομένως, εφόσον η επόμενη πράξη μετά την εξαγωγή τους είναι πολλαπλασιασμοί, τότε οι τιμές κοντά στο μηδέν θα έχουν ως αποτέλεσμα την διαγραφή πληροφορίας, ενώ οι τιμές κοντά στο μηδέν εξασφαλίζουν την διατήρησή και τη διάδοσή της.

Τελικά, η πύλη απώλειας μέσω της πράξης $c_{t-1} \cdot f_t$ διαγράφει πληροφορίες από την μέχρι τότε μακροπρόθεσμη μνήμη, η πύλη εισόδου μέσω της πράξης $c_t' \cdot i_t$ διαγράφει ασήμαντες πληροφορίες από την προσωρινή είσοδο του δικτύου και διατηρεί τις πιο σημαντικές, και το άθροισμα αυτών των δυο γινομένων αποτελεί την νέα μακροπρόθεσμη μνήμη του νευρώνα. Επιπλέον, η πύλη εξόδου μέσω της πράξης $\tanh(c_t) \cdot o_t$ ελέγχει την έξοδο του νευρώνα αναλαμβάνοντας να συγκρατήσει μόνο τις πιο σχετικές πληροφορίες της νέας μακροπρόθεσμης μνήμης, η οποία υπόκειται πρώτα σε έναν επιπλέον μετασχηματισμό μέσω της συνάρτησης της υπερβολικής εφαπτομένης $tanh$.

ΑΝΑΛΥΣΗ ΤΩΝ ΔΙΑΣΤΑΣΕΩΝ

Έστω το προηγούμενο παράδειγμα των 60 εγγραφών ενός πίνακα δεδομένων με 3 χαρακτηριστικά, και έστω ο μετασχηματισμός τους σε ακολουθίες των 4 στοιχείων, ο οποίος οδηγεί στον νέο πίνακα διάστασης $(57x4x3)$.

Στο μαθηματικό παράδειγμα εκπαίδευσης προηγούμενου υποκεφαλαίου για την αρχιτεκτονική MLP, το νευρωνικό δίκτυο που χρησιμοποιήθηκε είχε ένα κρυμμένο επίπεδο με δυο ξεκάθαρα διακριτούς νευρώνες h1 και h2. Στην περίπτωση ενός δικτύου LSTM όμως, δεν είναι δυνατή αυτή η διάκριση, καθώς υπάρχει μόνο ένας νευρώνας ο οποίος επεξεργάζεται τα δεδομένα που τού εισάγονται μέσω ενός εσωτερικού βρόγχου επανάληψης. Επομένως, για την καλύτερη κατανόηση των διαστάσεων και της διάκρισης της εσωτερικής διαδικασίας του νευρώνα LSTM, δίνεται το παρακάτω περιγραφικό γράφημα, όπου ορίζεται το 2 ως μέγεθος του επιπέδου LSTM:



Εικόνα 16. Απεικόνιση Επιμέρους Αρχιτεκτονικής Νευρώνα LSTM [79]

Επομένως, το μέγεθος του LSTM αναφέρεται στη διάσταση των καταστάσεων μνήμης c_t και h_t που εξάγει ο νευρώνας, και να σημειωθεί ότι οι πράξεις του πολλαπλασιασμού και της πρόσθεση αφορούν πράξεις στοιχείων θέσης (element-wise).

Τελικά, παρά την εμφανή πολυπλοκότητα της αρχιτεκτονικής τους, οι νευρώνες LSTM, αλλά και κάθε παραλλαγή τους, προτιμώνται σε σχέση με τους νευρώνες RNN στην ανάπτυξη επαναληπτικών νευρωνικών δικτύων. Η προσθήκη της κατάστασης μακροπρόθεσμης μνήμης, σε συνδυασμό με τους εξειδικευμένους μαθηματικούς μετασχηματισμούς, καταφέρνει να λύσει το πρόβλημα των ακολουθιών μεγάλου μήκους, συγκρατώντας παρελθοντικές πληροφορίες καθ' όλη τη διάρκεια επεξεργασίας των δεδομένων, καθώς και να αντιμετωπίσει το πρόβλημα των ασταθών παραγώγων, προσφέροντας σταθερότητα στον υπολογισμό τους κατά την προς τα πίσω διάδοση του σφάλματος προς όλα τα επαναληπτικά βήματα των νευρώνων, επιτρέποντας έτσι και την ανάπτυξη πολυπλοκότερων μοντέλων με περισσότερα υπολογιστικά επίπεδα για την αναγνώριση πιο εξειδικευμένων σχέσεων και μοτίβων στα δεδομένα.

2.4 Αυτοκωδικοποιητής (Autoencoder – AE)

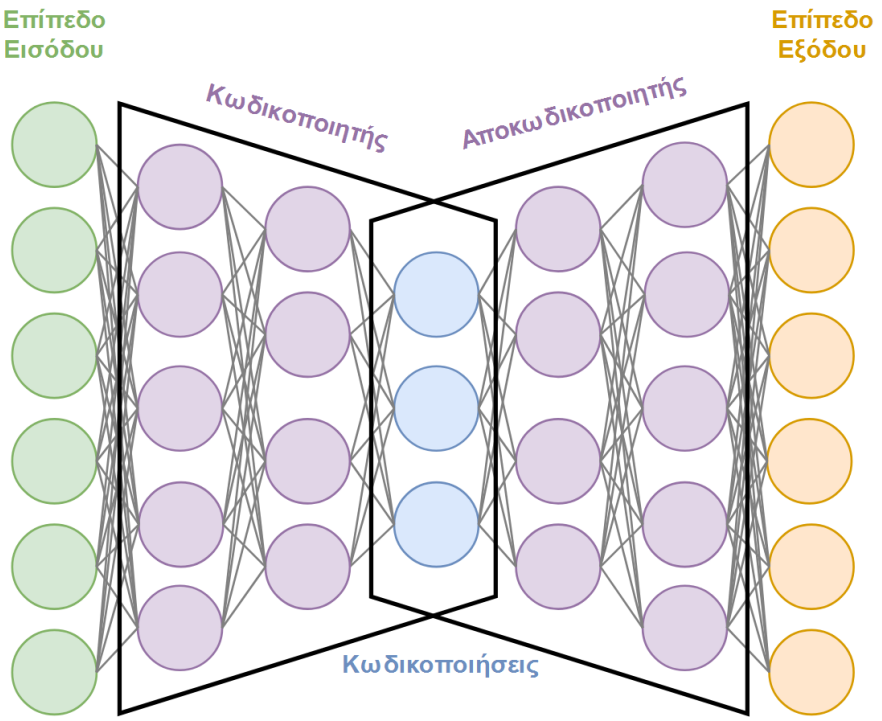
Ένας Αυτοκωδικοποιητής (Autoencoder – AE) αποτελεί εξειδικευμένη αρχιτεκτονική Τεχνητού Νευρωνικού Δικτύου, δηλαδή διασυνδεδεμένων επιπέδων εισόδου, κρυμμένων και εξόδου, με κύριο αντικείμενο την υλοποίηση ανακατασκευών από τα δεδομένα εισόδου.

Συγκεκριμένα, ένα μοντέλο AE δομείται από δυο επιμέρους νευρωνικά δίκτυα, τον κωδικοποιητή (encoder) και τον αποκωδικοποιητή (decoder). Ο κωδικοποιητής δέχεται τα αρχικά δεδομένα εισόδου, στη συνέχεια τα κωδικοποιεί σε συμπιεσμένες αναπαραστάσεις, και τελικά, ο αποκωδικοποιητής τροφοδοτείται με αυτές τις κωδικοποιήσεις και παράγει εξόδους οι οποίες αποτελούν όσο το δυνατόν καλύτερες ανακατασκευές των δεδομένων εισόδου, με όσο το δυνατόν μικρότερο σφάλμα.

Συνήθως, τα μοντέλα AE αναλαμβάνουν να κωδικοποιήσουν δεδομένα σε έναν χώρο μικρότερης διάστασης, αναγνωρίζοντας έτσι τα πιο σημαντικά χαρακτηριστικά τους. Για τον λόγο αυτόν είναι ιδιαίτερα χρήσιμα σε εργασίες μείωσης διάστασης (dimensionality reduction) ή σε περιπτώσεις Μη-Επιβλεπόμενης Μάθησης (Unsupervised Learning) όπου υπάρχει ανάγκη αναγνώρισης χαρακτηριστικών (feature detection). Επίσης, παραλλαγές των AE, όπως τα μοντέλα VAE και GAN που έχουν αναφερθεί, χρησιμοποιούν μεθόδους κωδικοποιήσεων για να παράγουν νέα δεδομένα παρόμοια με τα αρχικά, βασιζόμενα μόνο στα σημαντικότερα χαρακτηριστικά των συμπιεσμένων αναπαραστάσεων.

2.4.1 Αρχιτεκτονική AE

Ένας συμβατικός Αυτοκωδικοποιητής έχει τυπικά την ίδια αρχιτεκτονική με ένα νευρωνικό δίκτυο MLP. Αποτελείται από ένα επίπεδο εισόδου, ένα ή περισσότερα κρυμμένα επίπεδα και ένα επίπεδο εξόδου, τα οποία είναι διασυνδεμένα μεταξύ τους και εφαρμόζουν μαθηματικούς μετασχηματισμούς στα δεδομένα τα οποία περνάνε από το δίκτυο. Ειδικότερα, μια συμβατική αρχιτεκτονική AE φαίνεται παρακάτω:



Εικόνα 17. Αρχιτεκτονική Συμβατικού Δικτύου Αυτοκωδικοποιητή

Η διαφοροποίηση ενός μοντέλου ΑΕ έγκειται στο γεγονός ότι τα επίπεδα εισόδου και εξόδου πρέπει να έχουν το ίδιο πλήθος νευρώνων, ή αλλιώς την ίδια διάσταση. Αυτό είναι αναγκαίο διότι, εφόσον η ανάπτυξη του μοντέλου γίνεται με σκοπό την ανακατασκευή δεδομένων, τότε πρέπει η έξοδος να έχει ίδια διάσταση με τα δεδομένα εισόδου τα οποία καλείται να ανακατασκευάσει. Επίσης, η δομή ενός μοντέλου ΑΕ, όταν αυτό υλοποιείται με πολλαπλά κρυμμένα επίπεδα, ακολουθεί συνήθως συμμετρική κατασκευή ως προς το πλήθος των νευρώνων, και παρατηρώντας το παραπάνω παράδειγμα, εφαρμόζονται οι διαστάσεις $6-5-4-3-4-5-6$, όπου 3 είναι η διάσταση των κωδικοποιήσεων.

Επιπλέον, για τις ανάγκες της ανακατασκευής, ως χαρακτηριστικά στόχου για το τελικό επίπεδο, πρέπει να οριστούν ακριβώς τα ίδια χαρακτηριστικά που τροφοδοτούνται στο δίκτυο ως είσοδος, δηλαδή οι ετικέτες του δικτύου να είναι τα ίδια δεδομένα εισόδου, κάνοντας το γενικότερο πρόβλημα τύπου Αυτό-Επιβλεπόμενης Μάθησης (Self-Supervised Learning). Ωστόσο, αυτός ο τύπος μάθησης δεν είναι απόλυτος και εξαρτάται από το πρόβλημα που καλείται να επιλυθεί.

Επίσης, στην περίπτωση όπου οι κωδικοποιήσεις υλοποιούνται με μικρότερη διάσταση σε σχέση με εκείνη των αρχικών δεδομένων εισόδου, τότε το μοντέλο καλείται υποπλήρες ή ελλιπές (undercomplete), ενώ στην περίπτωση ίδιας ή μεγαλύτερης διάστασης, καλείται

υπερπλήρες (overcomplete). Ειδικότερα, ένα ελλιπές μοντέλο ΑΕ βασίζεται στη μείωση των διαστάσεων των αρχικών χαρακτηριστικών για να αναγνωρίσει τα σημαντικότερα στοιχεία και να τα συμπιέσει σε αντιπροσωπευτικές αναπαραστάσεις. Παρά το γεγονός ότι αποτελεί τον πιο συνηθισμένο τύπο ΑΕ, μπορεί να εμφανίσει αδυναμία, λόγω του περιορισμού της διάστασης, στο να μάθει τα πραγματικά υποκείμενα μοτίβα των δεδομένων, οδηγώντας έτσι στην παραγωγή ανεπιθύμητων ανακατασκευών. Από την άλλη, ένα υπερπλήρες μοντέλο, κωδικοποιώντας την είσοδο του δικτύου σε χώρο μεγαλύτερης διάστασης χαρακτηριστικών, έχει μεγαλύτερη πιθανότητα να αγνοήσει σημαντικά στοιχεία των δεδομένων, και απλά να λειτουργεί ως μια συνάρτηση ταυτότητας, αντιγράφοντας την είσοδο στην έξοδό του. Η επιλογή του μοντέλου που θα χρησιμοποιηθεί εξαρτάται από το εκάστοτε πρόβλημα και μετριάζεται αναλόγως για τις ανάγκες και τις καταστάσεις που πρέπει να αντιμετωπιστούν μέσω κατάλληλων τεχνικών οι οποίες εφαρμόζονται στην αρχιτεκτονική του μοντέλου κατά την ανάπτυξή του.

Η διαδικασία εκπαίδευσης ακολουθεί τα ίδια βήματα όπως έχουν περιγραφεί και στους προηγούμενους τύπους νευρωνικών δικτύων. Μόλις γίνει η εξαγωγή των προσωρινών προβλέψεων από το επίπεδο εξόδου, συγκρίνονται με τα πραγματικά δεδομένα, τα οποία έχουν δοθεί ως είσοδος στο δίκτυο, μέσω μιας κατάλληλης συνάρτησης απώλειας για να υπολογιστεί το συνολικό σφάλμα της ανακατασκευής. Στη συνέχεια, το συνολικό σφάλμα, βάσει του Αλγορίθμου Οπισθοδιάδοσης, διαδίδεται προς τα πίσω στα επίπεδα του δικτύου και υπολογίζεται η συνεισφορά κάθε παραμέτρου στο συνολικό σφάλμα μέσω των μερικών παραγώγων και του κανόνα της αλυσίδας του διαφορικού λογισμού. Αφού υπολογιστεί η συνεισφορά κάθε παραμέτρου του δικτύου, τότε όλες οι παράμετροι ενημερώνονται με μια μέθοδο βελτιστοποίησης με σκοπό να ελαχιστοποιηθεί η συνάρτηση απώλειας με κάθε νέα επανάληψη της παραπάνω διαδικασίας. Να σημειωθεί ότι, όπως και στις προηγούμενες περιπτώσεις, και εδώ η εκπαίδευση προτιμάται να γίνεται με τα δεδομένα εισόδου υπό μορφή παρτίδων στοιχείων.

Τελικά, η αξιολόγηση και η λειτουργικότητα ενός εκπαιδευμένου μοντέλου ΑΕ έγκειται στο σφάλμα των ανακατασκευών που εξάγει. Σε περίπτωση που τροφοδοτηθεί με δεδομένα που διαφέρουν σημαντικά από αυτά που έχει μάθει να ανακατασκευάζει, τότε το σφάλμα της ανακατασκευής θα είναι σημαντικά μεγαλύτερο από αυτό που αναμένεται.

Η αρχιτεκτονική ενός μοντέλου ΑΕ είναι ευέλικτη ως προς την επιλογή του είδους των επιπέδων και των νευρώνων του, δηλαδή η φύση του προβλήματος που καλείται να αντιμετωπίσει παίζει καθοριστικό ρόλο στη δομή του μοντέλου. Συγκεκριμένα, ένα μοντέλο ΑΕ στη συμβατική του μορφή μπορεί να δεχτεί δεδομένα πίνακα πολλών στηλών και μέσω

της εκπαίδευσης των ανακατασκευών να αναγνωρίσει με τις κωδικοποιήσεις τα σημαντικότερα χαρακτηριστικά. Ακόμη, ένα μοντέλο AE του οποίου τα επίπεδα είναι νευρώνες LSTM, ή γενικότερα επαναληπτικού τύπου, μπορεί να μάθει να ανακατασκευάζει ακολουθίες και χρονοσειρές, ή αν δομείται με επίπεδα τύπου CNN μπορεί τελικά να είναι ικανό να ανακατασκευάζει εικόνες ή να παράγει νέες παρόμοιες, καθώς τα δίκτυα CNN εφαρμόζονται ιδιαίτερα στην επεξεργασία και την ανάλυση δεδομένων εικόνας.

Κεφάλαιο 3

Αλγόριθμοι Μηχανικής Μάθησης

Πρακτικές Εφαρμογές

1 Λογισμικά Μηχανικής Μάθησης

Στο συγκεκριμένο Κεφάλαιο παρουσιάζονται δυο σύνολα δεδομένων τα οποία δημιουργήθηκαν από καταγεγραμμένες μετρήσεις αισθητήρων που τοποθετήθηκαν σε μηχανικά εξαρτήματα. Συγκεκριμένα, τα δεδομένα εξετάζονται στα πλαίσια της Στατιστικής Ανάλυσης και στη συνέχεια αναπτύσσονται κατάλληλα μοντέλα Μηχανικής Μάθησης στα πλαίσια της Προβλεπτικής Συντήρησης για την αναγνώριση βλαβερών καταστάσεων στη λειτουργία των εξοπλισμών.

Κάθε αναλυτική διαδικασία γίνεται με υπολογιστικό τρόπο σε ειδικό περιβάλλον και με κύρια γλώσσα προγραμματισμού την Python. Ειδικότερα, με την υποστήριξη εξειδικευμένων διαθέσιμων βιβλιοθηκών επιτυγχάνεται με εύκολο τρόπο τόσο η ανάλυση των δεδομένων, όσο και η ανάπτυξη των αλγορίθμων Μηχανικής Μάθησης. Οι βιβλιοθήκες Python που χρησιμοποιούνται είναι οι Pandas [80], Numpy [81], Matplotlib [82], Scikit-learn [83] και Tensorflow [84].

Συγκεκριμένα, η Pandas επιτρέπει τη στατιστική ανάλυση και επεξεργασία συνόλων δεδομένων, η Numpy αποτελεί τη θεμελιώδη βιβλιοθήκη για επιστημονικούς υπολογισμούς και την επεξεργασία πινάκων, και η Matplotlib υποστηρίζει την οπτικοποίηση των δεδομένων με διαγράμματα. Έπειτα, η βιβλιοθήκη Scikit-learn επιτρέπει την επεξεργασία δεδομένων και την ανάλυσή τους έχοντας ως επίκεντρο τη μοντελοποίηση μέσω κλασσικών αλγορίθμων Μηχανικής Μάθησης. Τέλος, η βιβλιοθήκη Tensorflow παρέχει, μεταξύ άλλων, τη δυνατότητα του αυτόματου υπολογισμού των παραγώγων κάθε παραγωγίσιμης συνάρτησης, κάτι που είδαμε ότι είναι απαραίτητο στη διαδικασία εκπαίδευσης ενός μοντέλου Βαθιάς Μάθησης. Ειδικότερα, η ανάπτυξη των μοντέλων Βαθιάς Μάθησης γίνεται με τη δημοφιλή διεπαφή (API - Application Programming Interface) της Tensorflow, το Keras [85]. Φυσικά, κάθε παραπάνω βιβλιοθήκη αποτελείται

από ένα μεγάλο εύρος εργαλείων, αλλά στην προκειμένη περίπτωση η λειτουργικότητά τους κινείται στα παραπάνω πλαίσια εργασιών.

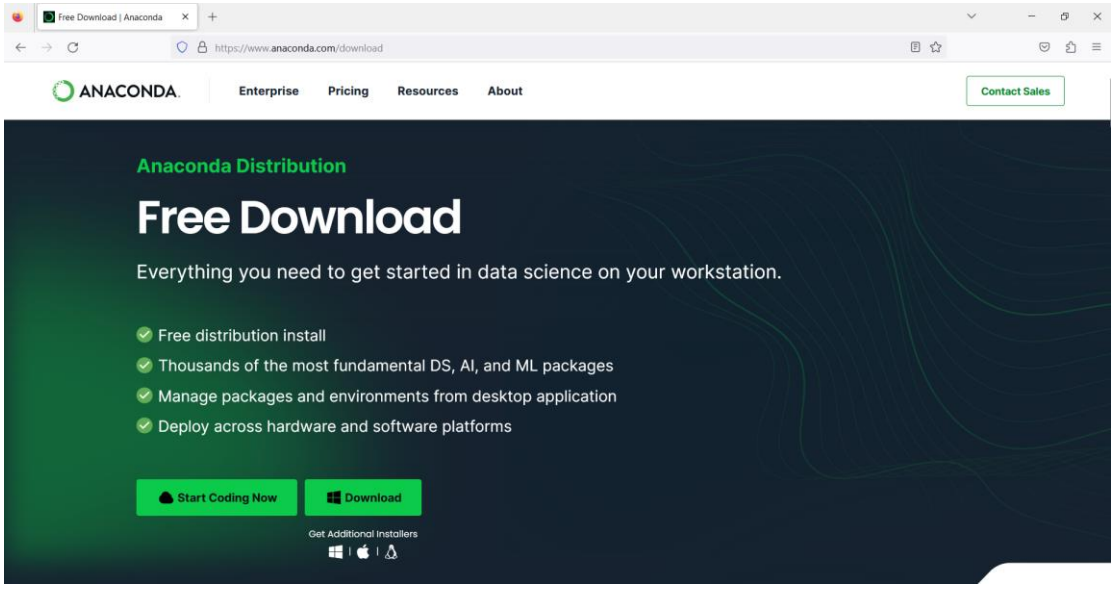
Επιπλέον, η υπολογιστική διαδικασία γίνεται με τη βοήθεια ειδικού τύπου αρχείων, τα οποία ονομάζονται notebooks (σημειωματάρια). Χρησιμοποιούνται ευρέως στην επιστήμη και ανάλυση δεδομένων και σε εργασίες Μηχανικής Μάθησης, καθώς επιτρέπουν διαδραστικές υπολογιστικές διαδικασίες, δηλαδή την εκτέλεση του προγραμματιστικού κώδικα σε ανεξάρτητα κομμάτια, σε αντίθεση με την εκτέλεση ολόκληρων αρχείων.

Παρακάτω παρουσιάζονται δυο τρόποι για να στηθεί εύκολα ένα περιβάλλον Στατιστικής Ανάλυσης και Μηχανικής Μάθησης με όλα τα παραπάνω χαρακτηριστικά.

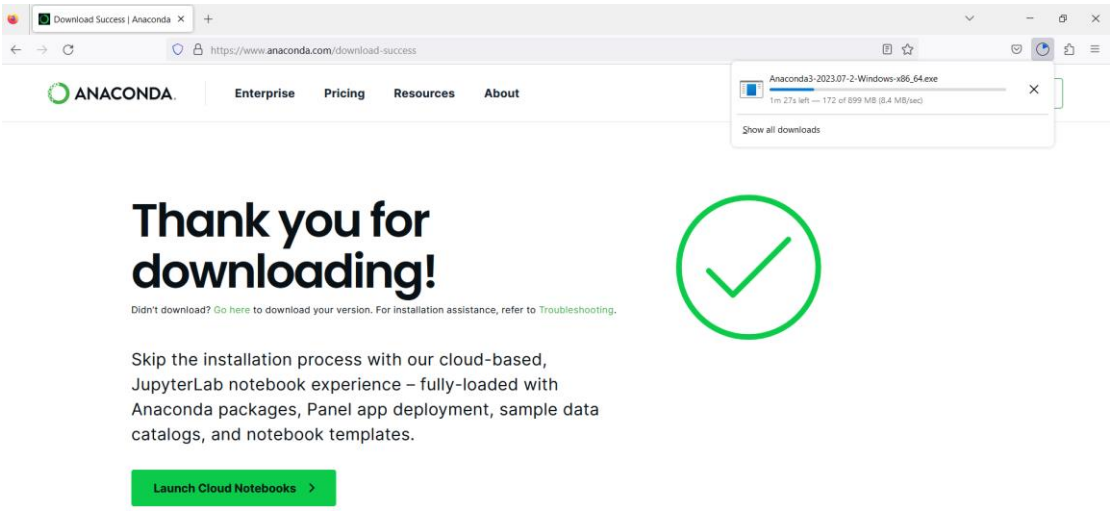
1.1 Anaconda Distribution

Ο πρώτος τρόπος για τη δημιουργία ενός κατάλληλου περιβάλλοντος ανάπτυξης είναι μέσω του Anaconda, το οποίο επιτρέπει τη διανομή και εγκατάσταση της γλώσσας προγραμματισμού Python, αλλά και πληθώρας βιβλιοθηκών της, προσφέροντας μια ευκολότερη και πιο συγκεντρωτική διαχείριση.

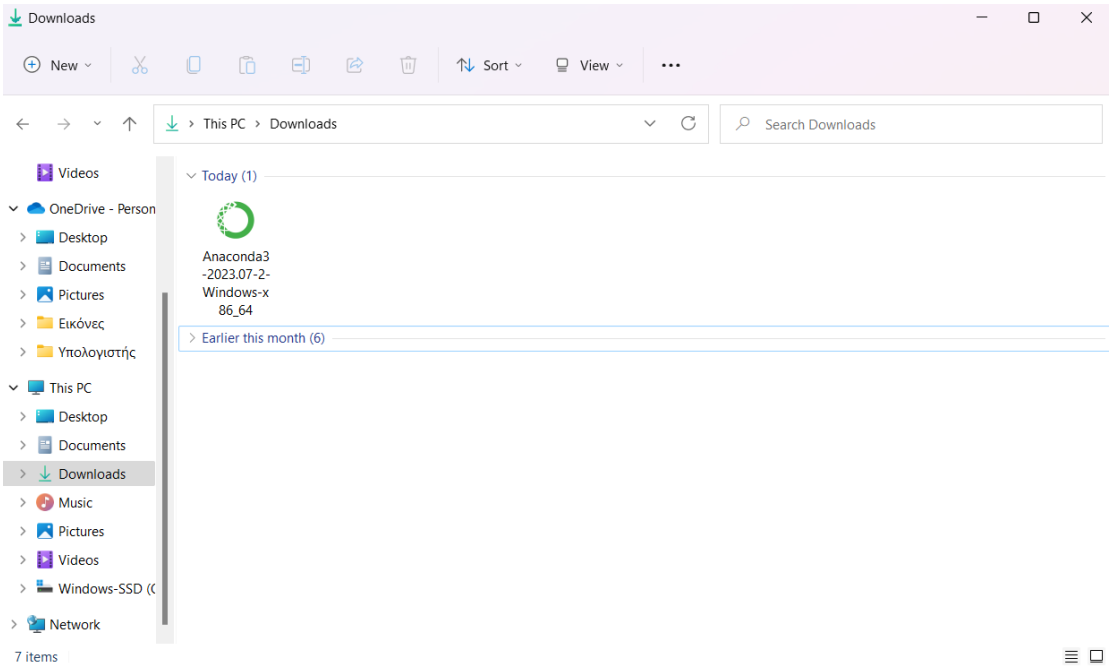
Το αρχείο εγκατάστασης του Anaconda είναι ελεύθερα διαθέσιμο μέσω της επίσημης ιστοσελίδας https://www.anaconda.com/ [86]:



Εικόνα 18. Επίσημη Ιστοσελίδα Διανομέα Anaconda

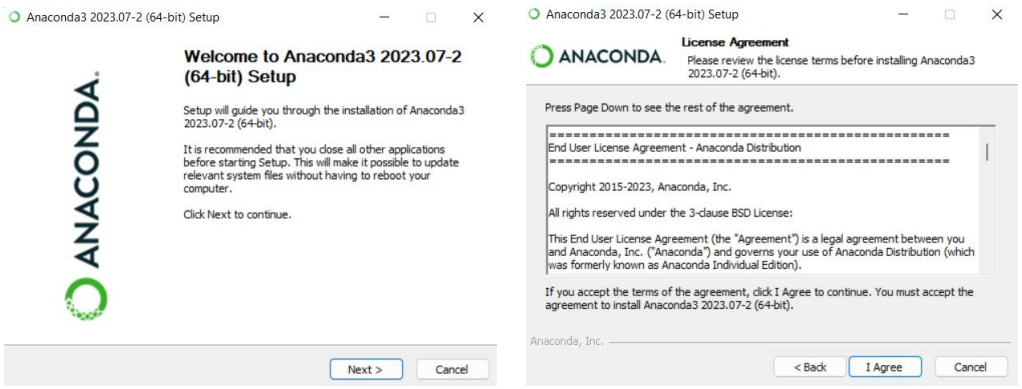


Εικόνα 19. Κατέβασμα Αρχείου Εγκατάστασης Anaconda

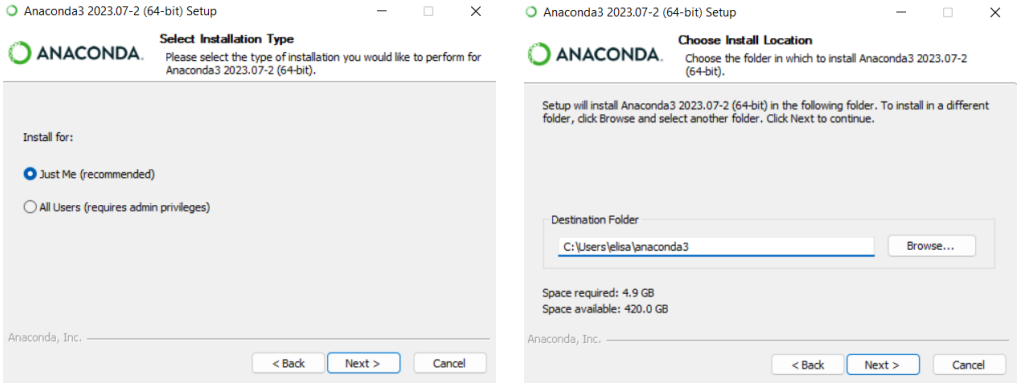


Εικόνα 20. Αρχείο Εγκατάστασης Anaconda

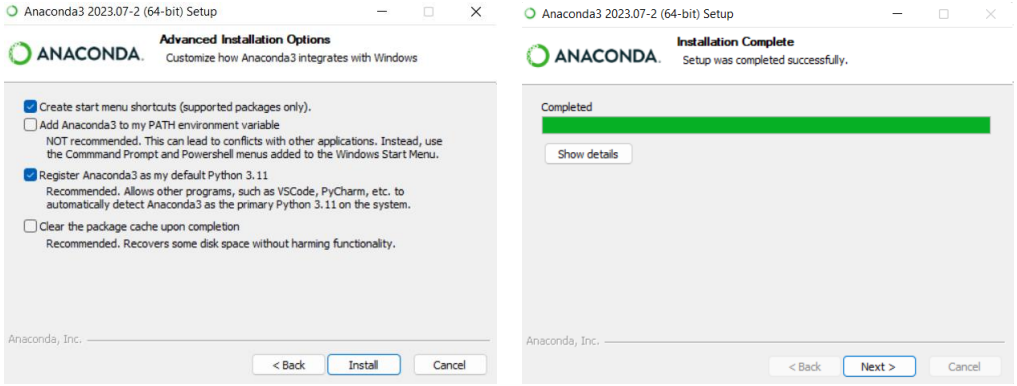
Η διαδικασία εγκατάστασης ακολουθεί τα συμβατικά βήματα εγκατάστασης προγράμματος με όλες τις προεπιλεγμένες ρυθμίσεις:



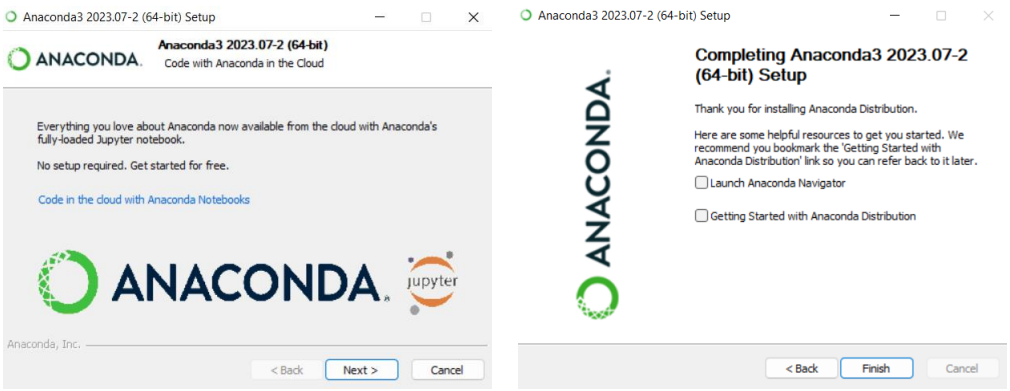
Εικόνα 21. Πρώτο και Δεύτερο Βήμα Εγκατάστασης Anaconda



Εικόνα 22. Τρίτο και Τέταρτο Βήμα Εγκατάστασης Anaconda

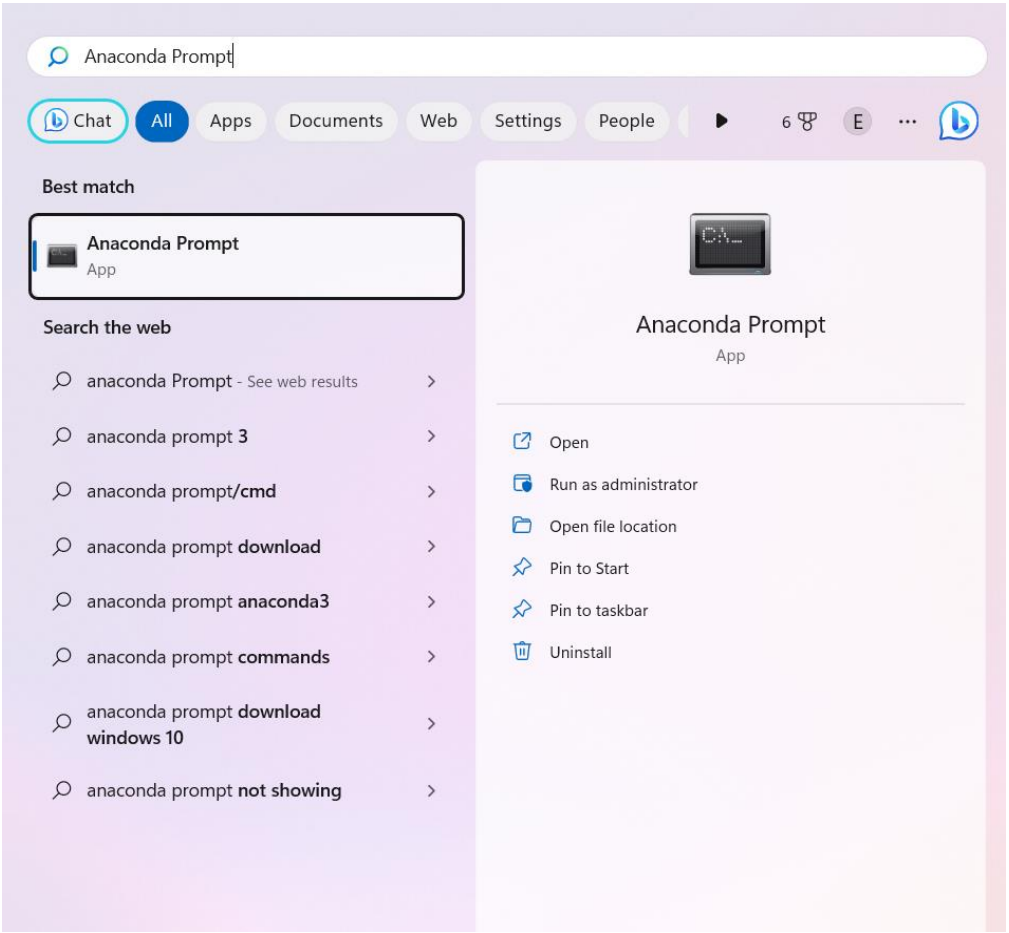


Εικόνα 23. Πέμπτο και Έκτο Βήμα Εγκατάστασης Anaconda



Εικόνα 24. Έβδομο και Όγδοο Βήμα Εγκατάστασης Anaconda

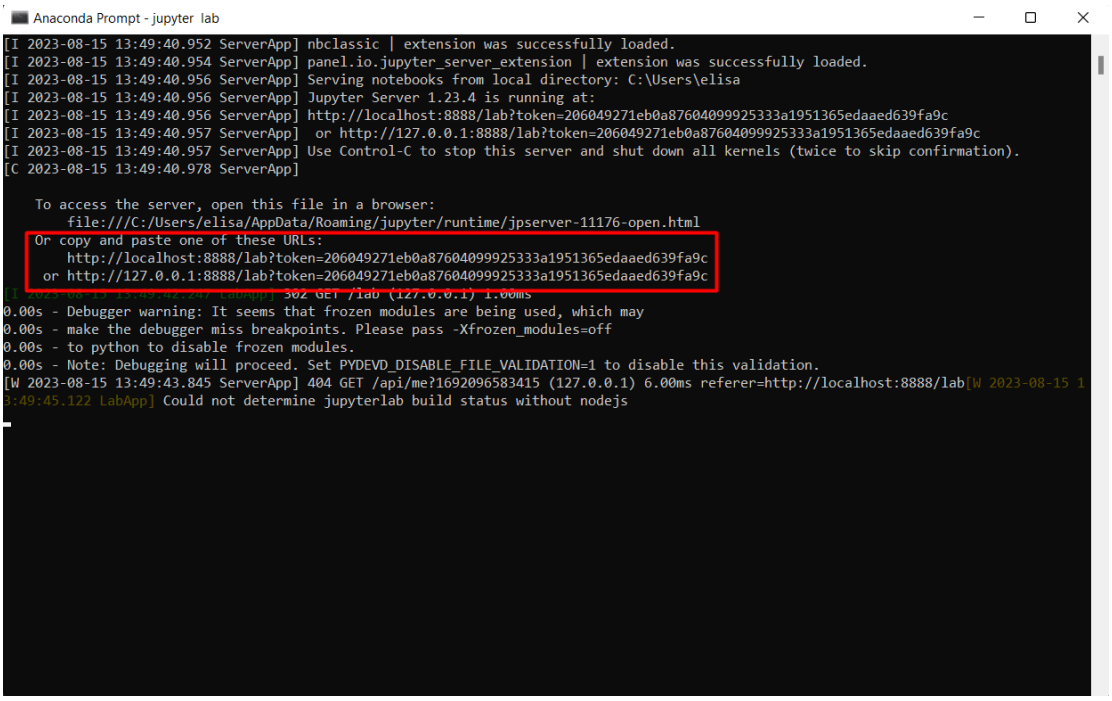
Μετά την ολοκλήρωση της εγκατάστασης, η εκκίνηση του υπολογιστικού περιβάλλοντος γίνεται γρήγορα μέσω του τερματικού εντολών Anaconda Prompt, εκτελώντας την εντολή jupyter lab:



Εικόνα 25. Τερματικό Εντολών Anaconda Prompt

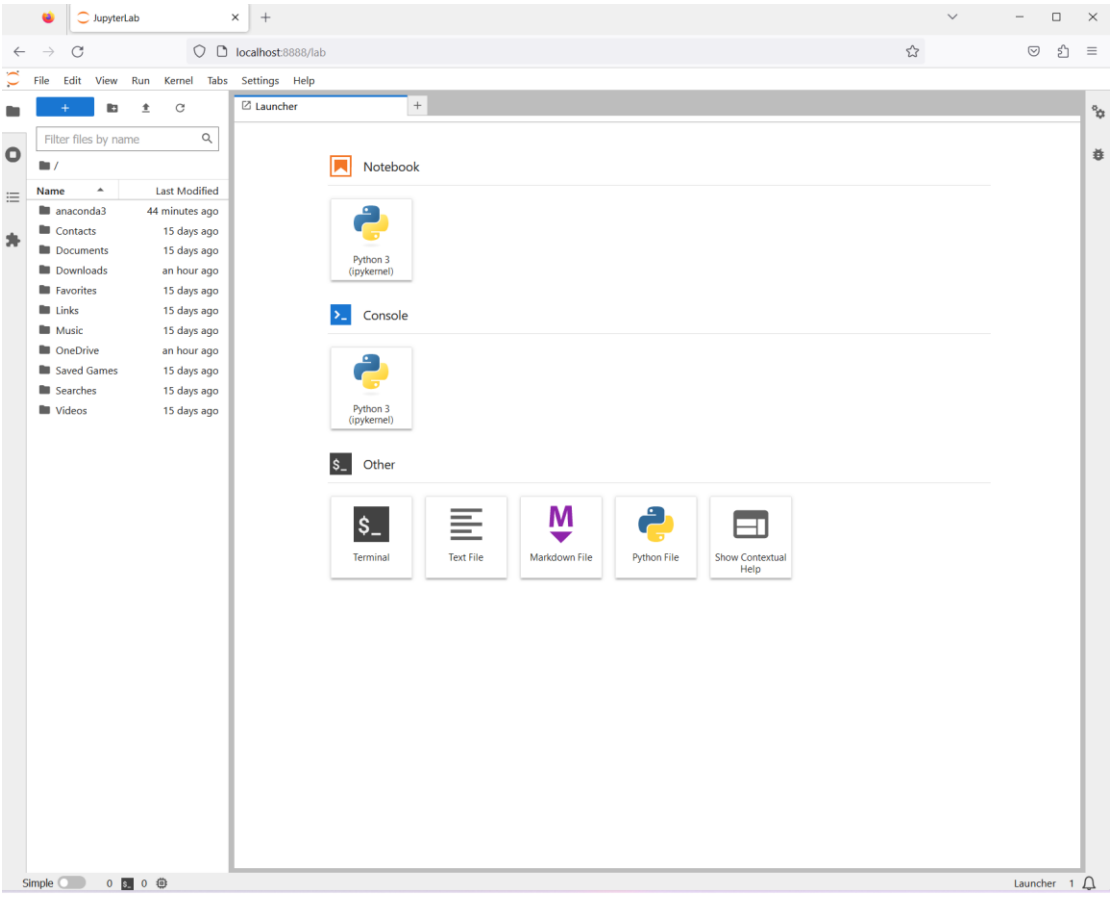
Εικόνα 26. Εκτέλεση Εντολής Έναρξης Υπολογιστικού Περιβάλλοντος JupyterLab

Με την εκτέλεση της παραπάνω εντολής θα πρέπει να ανοίξει αυτόματα το υπολογιστικό περιβάλλον JupyterLab στον προεπιλεγμένο browser του υπολογιστή. Αν δεν συμβεί αυτό, τότε προτείνεται η αντιγραφή μιας από τις παρακάτω προτεινόμενες διευθύνσεις:



Εικόνα 27. Εναλλακτική Εκκίνηση Έναρξης Υπολογιστικού Περιβάλλοντος JupyterLab

Το JupyterLab είναι ένα από τα υπολογιστικά περιβάλλοντα ανάπτυξης το οποίο επιτρέπει τη διαδικτυακή εκτέλεση των αρχείων notebook, ή των Jupyter Notebook στην προκειμένη περίπτωση, και έχει την παρακάτω μορφή:

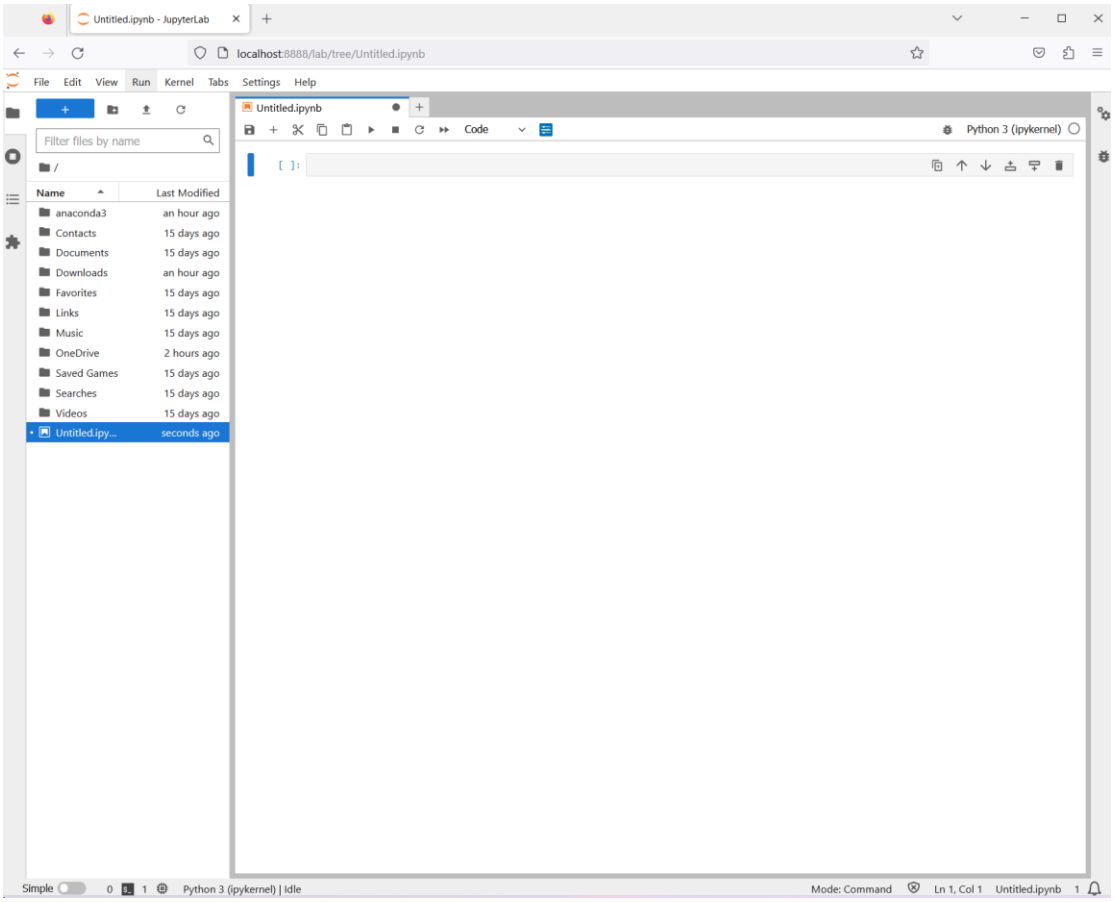


Εικόνα 28. Περιβάλλον Ανάπτυξης JupyterLab στον Browser

Χρειάζεται ιδιαίτερη προσοχή για να παραμείνει ενεργό το προηγούμενο τερματικό εντολών, καθώς με το κλείσιμό του απενεργοποιείται το περιβάλλον ανάπτυξης και παύει να ανταποκρίνεται.

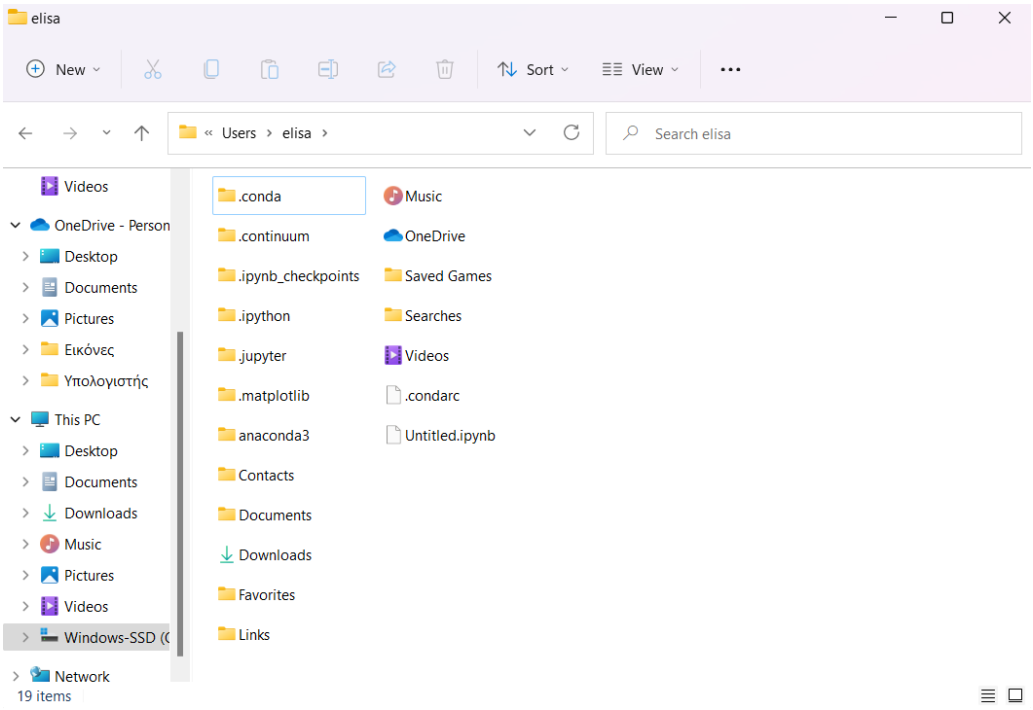
Να σημειωθεί επίσης ότι, η παραπάνω εφαρμογή δεν αποτελεί κάποια υπηρεσία Cloud, αλλά αξιοποιεί τους πόρους του μηχανήματος όπου λειτουργεί. Συγκεκριμένα, προς το παρόν βασίζεται στην υπολογιστική ισχύ του επεξεργαστή (Central Processing Unit – CPU), αλλά μπορεί να ρυθμιστεί κατάλληλα και για τη χρήση κάρτας γραφικών (Graphics Processing Unit – GPU).

Επιλέγοντας από το αρχικό μενού «Launcher», στην υποκατηγορία «Notebook», την προτίμηση «Python 3 (ipykernel)», ανοίγει αυτόματα ένα νέο αρχείο notebook με τίτλο «Untitled» και τύπο αρχείου .ipynb για κωδικοποίηση με την γλώσσα προγραμματισμού Python όπως φαίνεται παρακάτω:



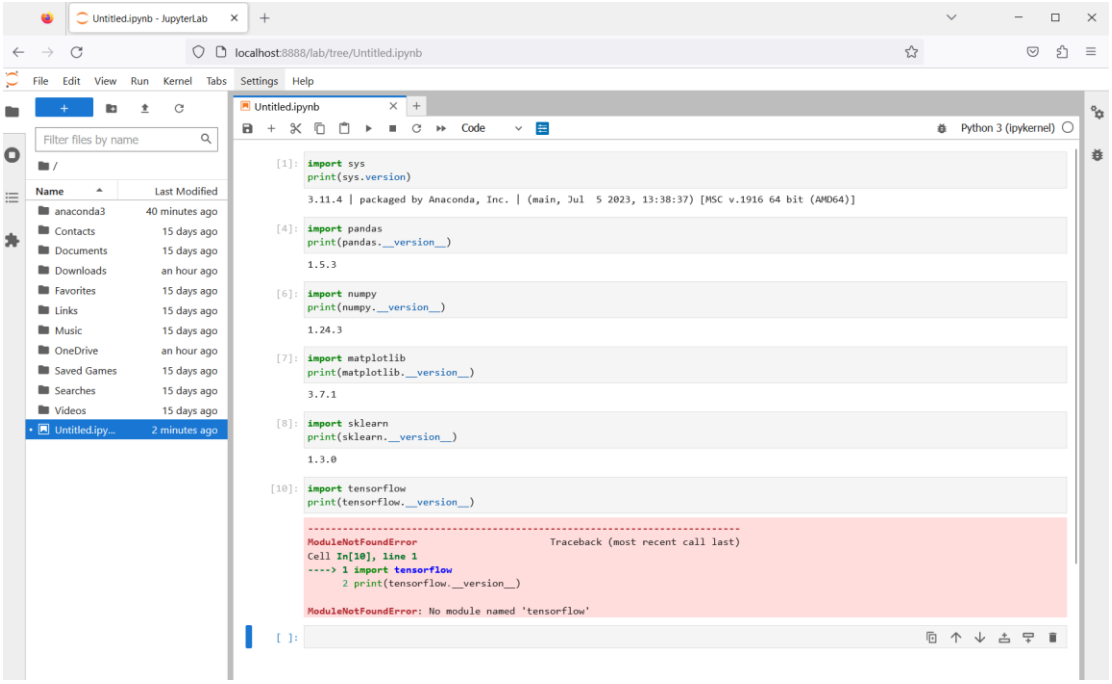
Εικόνα 29. Δημιουργία notebook στο Περιβάλλον Ανάπτυξης JupyterLab

Η εξερεύνηση αρχείων στο αριστερό μέρος του υπολογιστικού περιβάλλοντος αφορά στον φάκελο του χρήστη, όπως είχε οριστεί κατά τη στιγμή της εγκατάστασης του Anaconda, φαίνεται αντίστοιχα παρακάτω, και αφήνεται στην ευχέρεια του χρήστη να προσαρμόσει κάθε περαιτέρω διαδρομή των αρχείων όπως κρίνει αυτός:



Εικόνα 30. Εξερεύνηση Τοπικών Αρχείων Χρήστη JupyterLab

Αυτή τη στιγμή είμαστε έτοιμοι για την εκτέλεση εντολών, όπου κάθε κελί μπορεί να αποτελέσει ανεξάρτητο κομμάτι του notebook:

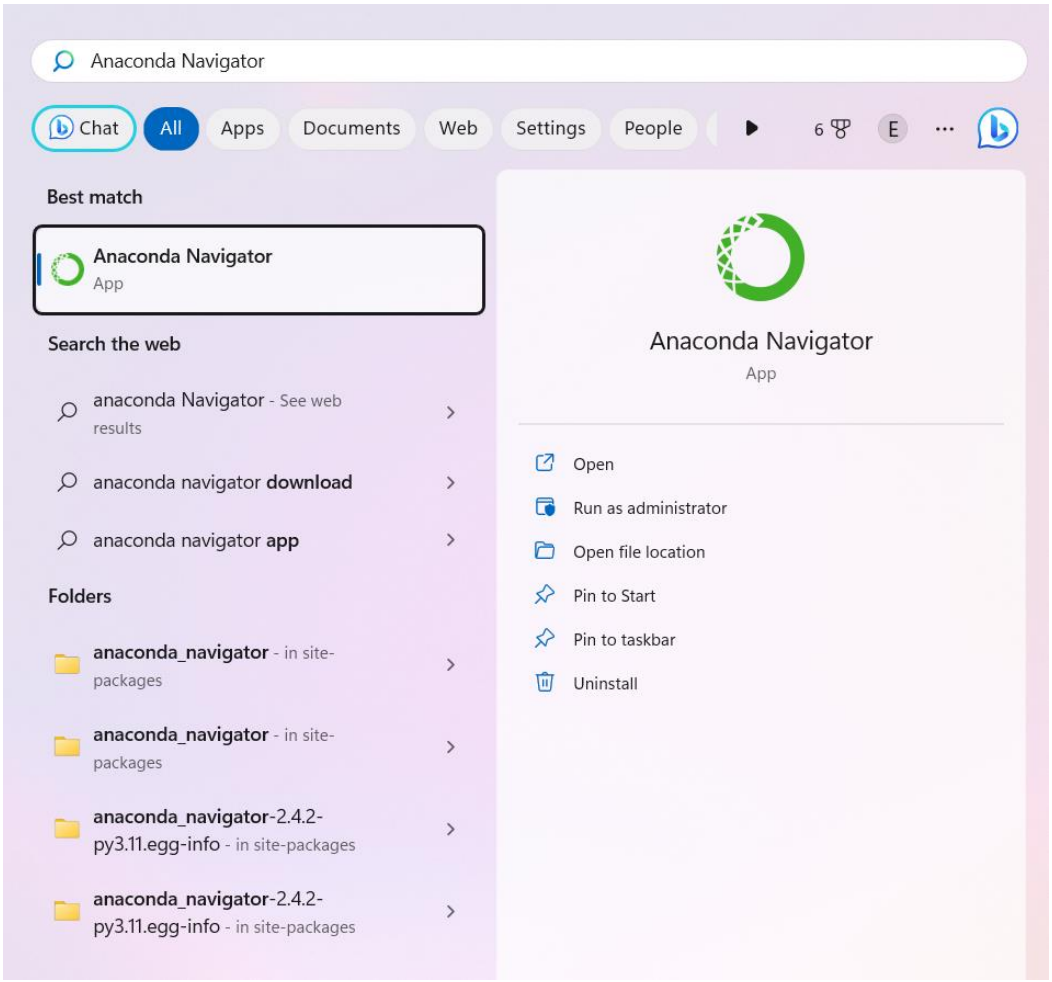


Εικόνα 31. Εκτέλεση Κελιών του Notebook (Ανεπιτυχής Είσοδος Πακέτων)

Παραπάνω έγινε έλεγχος για τη διαθέσιμη έκδοση της γλώσσας προγραμματισμού Python που εγκαταστάθηκε με το Anaconda, και έγινε τμηματικά η είσοδος των απαραίτητων βιβλιοθηκών, εκτός της Tensorflow, η οποία για το περιβάλλον όπου δουλεύουμε δεν είναι ακόμα εγκατεστημένη.

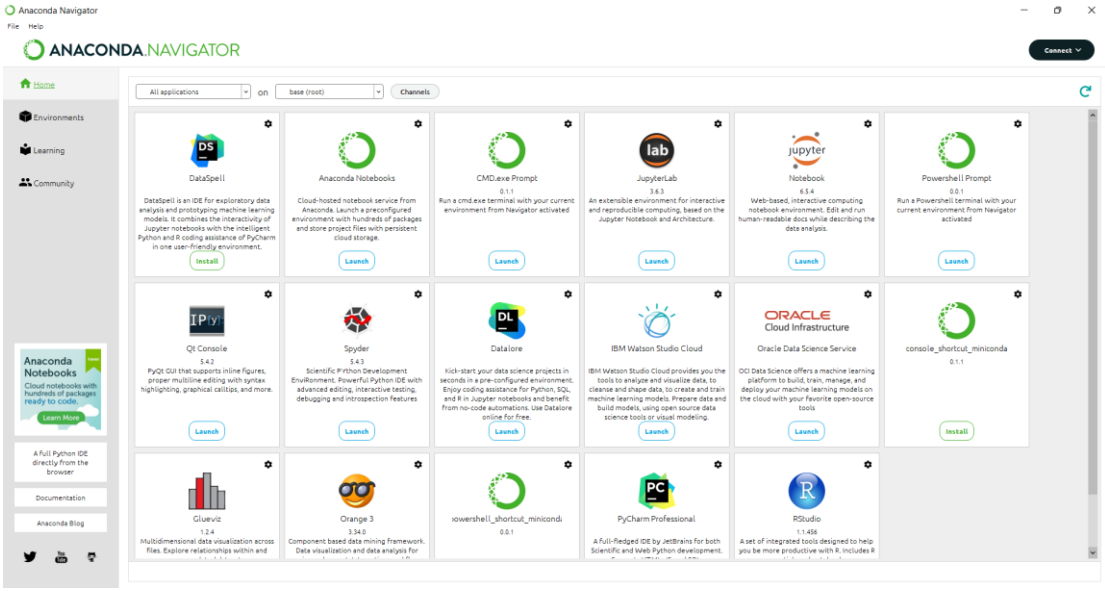
Στο κύριο υπολογιστικό περιβάλλον χρησιμοποιείται η γλώσσα Python έκδοσης 3.11, με την οποία η βιβλιοθήκη Tensorflow είναι προς το παρόν ασύμβατη. Συνεπώς, παρακάτω ακολουθεί η διαδικασία δημιουργίας ενός νέου περιβάλλοντος που χρησιμοποιεί την Python 3.10, και την εγκατάσταση των απαραίτητων βιβλιοθηκών.

Η διαδικασία μπορεί να γίνει με τερματικά εντολών, αλλά εδώ παρουσιάζεται κάνοντας χρήση της διεπαφής Anaconda Navigator:



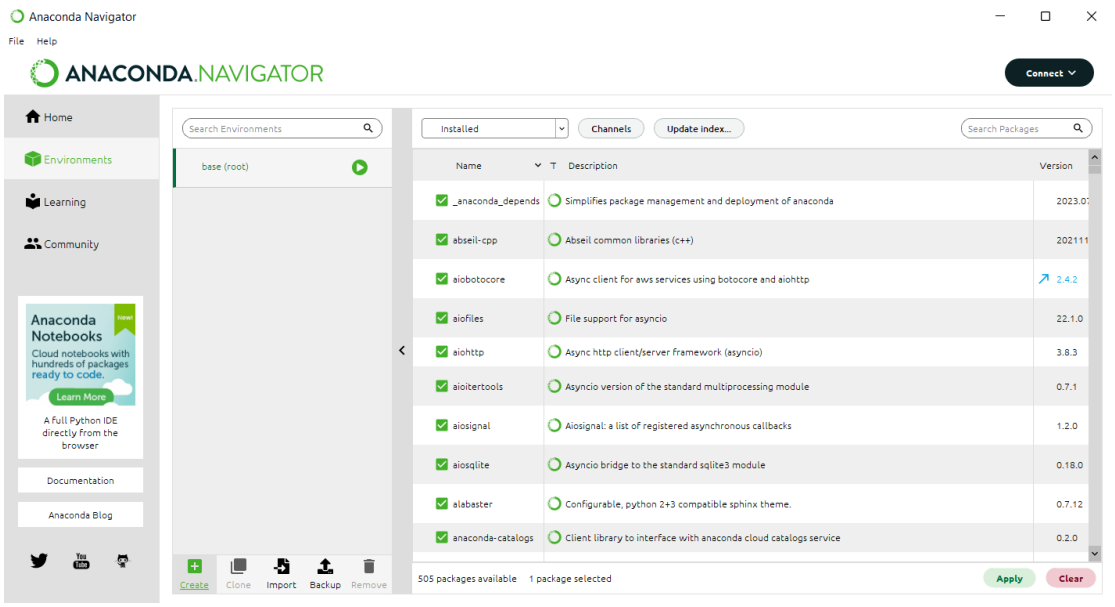
Εικόνα 32. Άνοιγμα της Εφαρμογής Anaconda Navigator

Με το άνοιγμα του Anaconda Navigator βλέπουμε στην αρχική σελίδα διάφορες προεπιλεγμένες εγκατεστημένες εφαρμογές του βασικού υπολογιστικού περιβάλλοντος «base (root)» το οποίο εγκαταστάθηκε αυτόματα με το Anaconda:



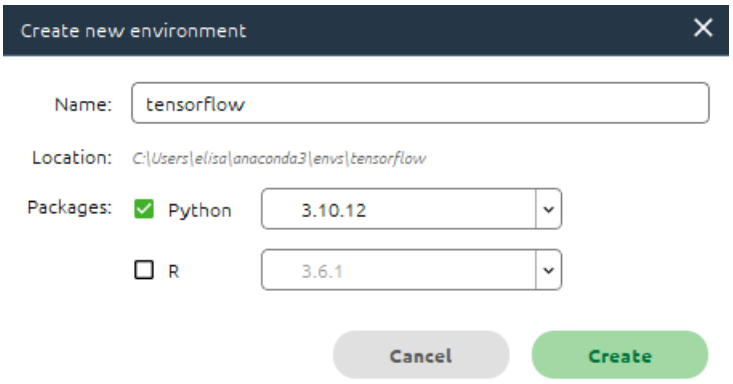
Εικόνα 33. Αρχική Σελίδα του Anaconda Navigator

Από το αριστερό μέρος επιλέγουμε «Environments» όπου μπορούμε να δούμε όλα τα διαθέσιμα περιβάλλοντα με τα εγκατεστημένα πακέτα τους, και επιλέγουμε την προτίμηση «Create» για να δημιουργήσουμε το νέο περιβάλλον:



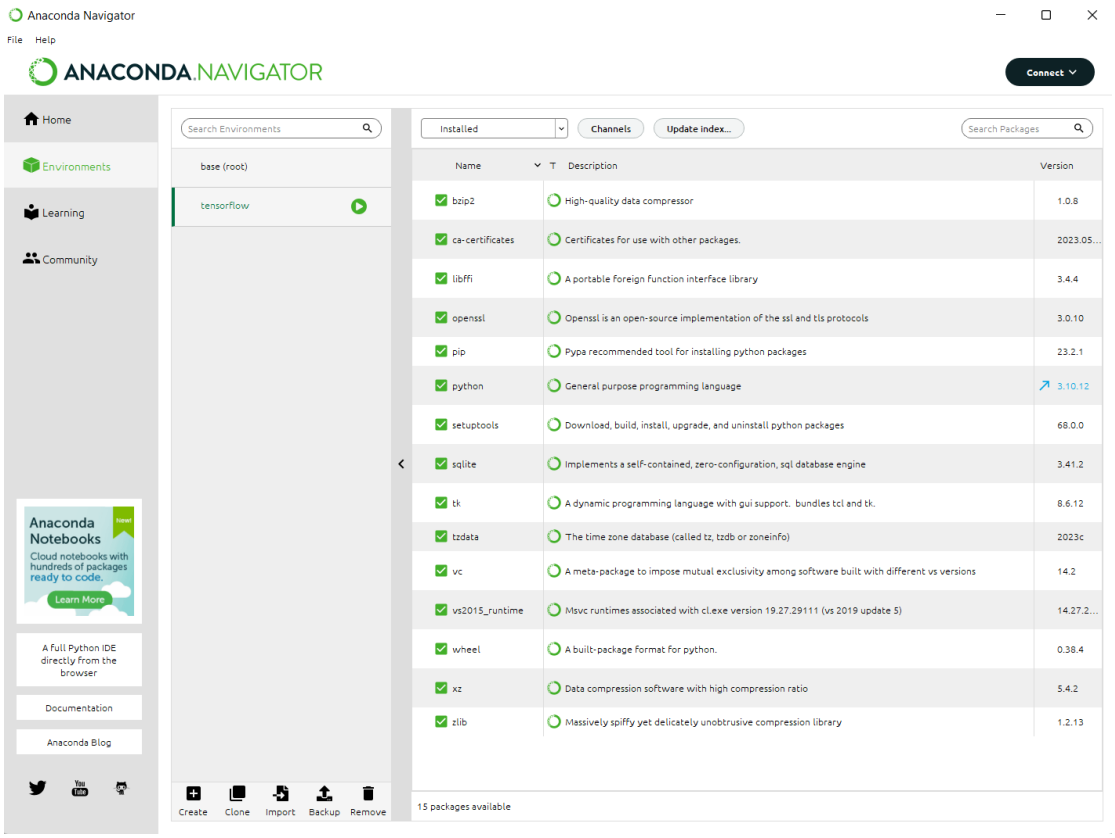
Εικόνα 34. Περιβάλλοντα του Anaconda και Δημιουργία

Στη συνέχεια δίνουμε ως όνομα του νέου περιβάλλοντος το «tensorflow» και επιλέγουμε την έκδοση Python 3.10 από όλες τις διαθέσιμες εκδόσεις:



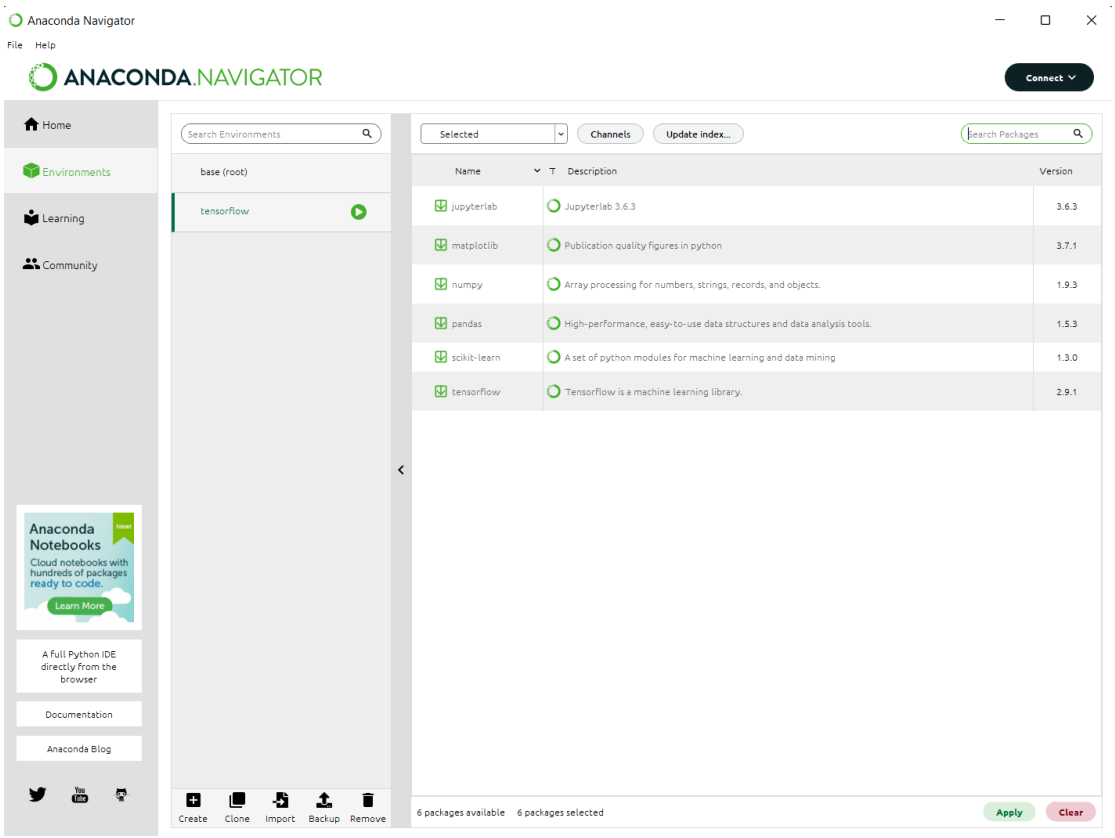
Εικόνα 35. Δημιουργία Νέου Python Περιβάλλοντος tensorflow

Το νέο περιβάλλον που δημιουργείται αποτελείται μόνο από 15 βασικά προεπιλεγμένα πακέτα python:



Εικόνα 36. Προεπισκόπηση Νέου Περιβάλλοντος tensorflow

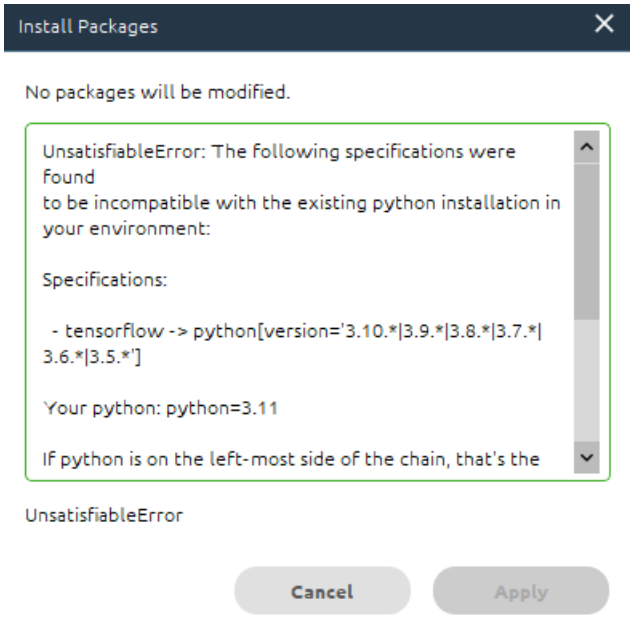
Στη συνέχεια, από την αναπτυσσόμενη λίστα (παραπάνω είναι επιλεγμένο το Installed) επιλέγουμε την προτίμηση «Not Installed» και κάνοντας αναζήτηση επιλέγουμε όλα τα πακέτα που θέλουμε να εγκαταστήσουμε, αλλά και την εφαρμογή JupyterLab που επίσης δεν ήταν στα προεγκατεστημένα πακέτα:



Εικόνα 37. Πακέτα προς Εγκατάσταση στο Περιβάλλον tensorflow

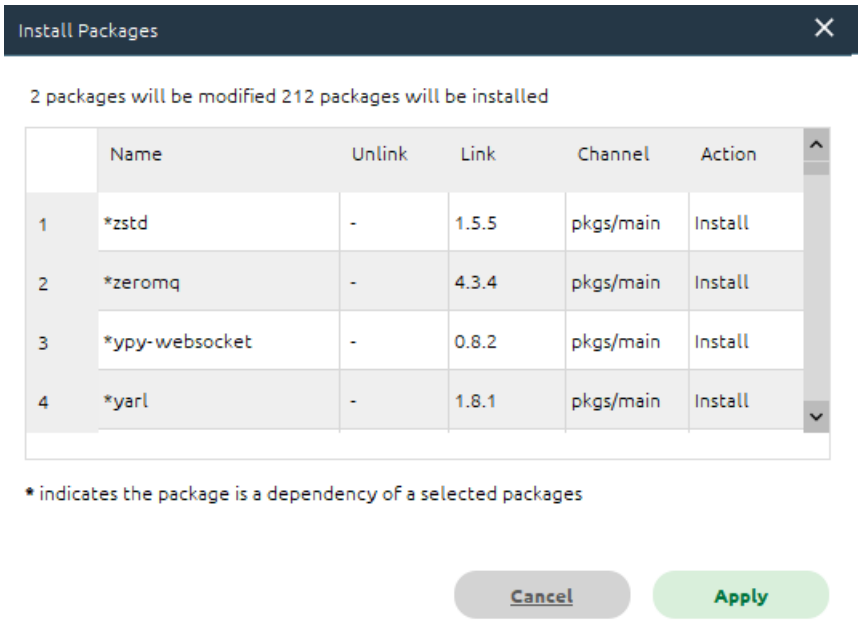
Με την εντολή «Apply» της παραπάνω εικόνας γίνεται έλεγχος για τυχόν εξαρτήσεις μεταξύ των πακέτων, ή τροποποιήσεις για λόγους ασυμβιβασίας, και βγαίνει αντίστοιχη ειδοποίηση πριν την οριστική επιλογή εγκατάστασης.

Στην περίπτωση προσπάθειας εγκατάστασης μόνο της βιβλιοθήκης Tensorflow στο βασικό υπολογιστικό περιβάλλον της Python 3.11 είχαμε την παρακάτω προειδοποίηση:



Εικόνα 38. Ασυμβιβασία Tensorflow και Python 3.11

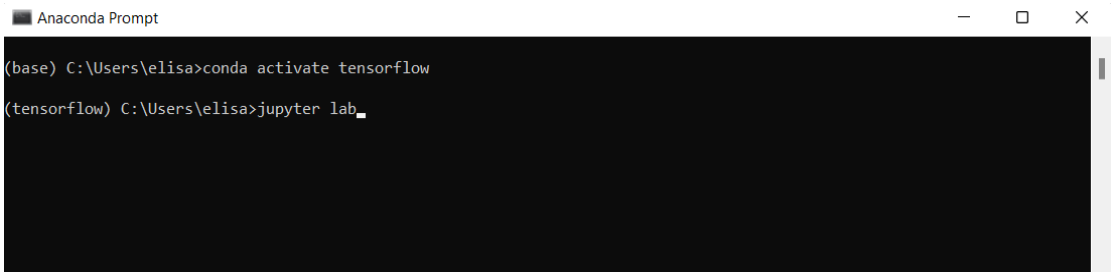
Αλλά, μετά τη δημιουργία του νέου περιβάλλοντος έχουμε τη παρακάτω σύνοψη:



Εικόνα 39. Εγκατάσταση Πακέτων στο Νέο Περιβάλλον tensorflow

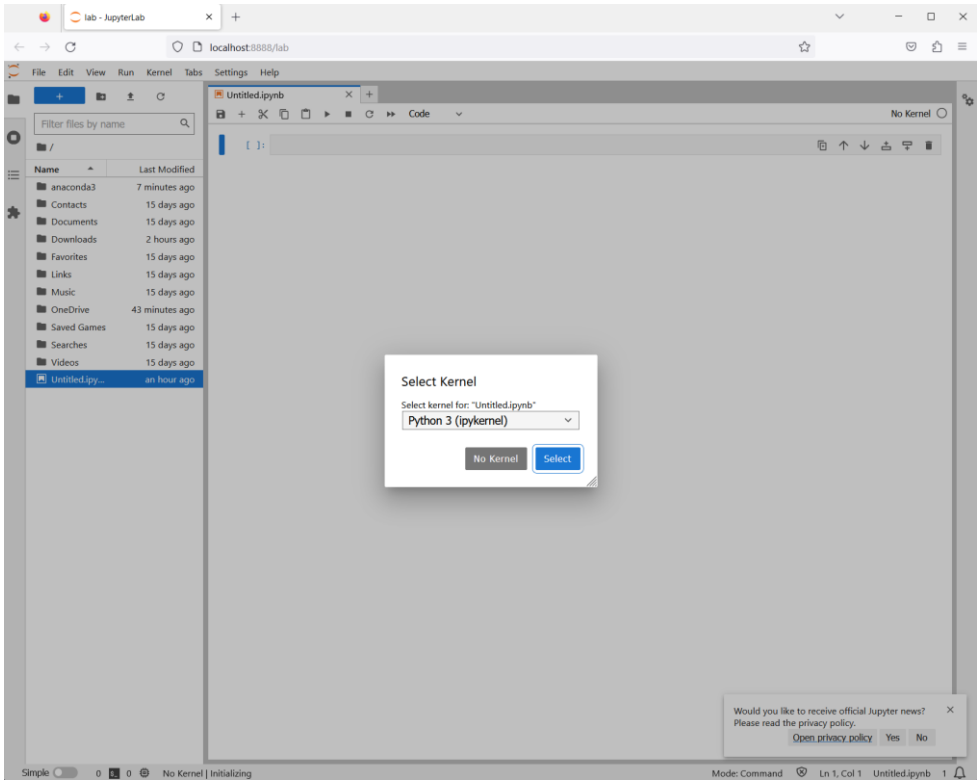
Βλέπουμε ότι για την εγκατάσταση των 6 επιλεγμένων πακέτων απαιτήθηκε η εγκατάσταση 212 πακέτων συνολικά και η τροποποίηση 2 πακέτων.

Μετά την εγκατάσταση, κλείνουμε τη διεπαφή Anaconda Navigator και μέσω ενός νέου τερματικού εντολών Anaconda Prompt ενεργοποιούμε το νέο περιβάλλον με το όνομα «tensorflow» μέσω της εντολής conda activate tensorflow, και στη συνέχεια κάνουμε εκκίνηση της εφαρμογής JupyterLab με την εντολή jupyter lab:



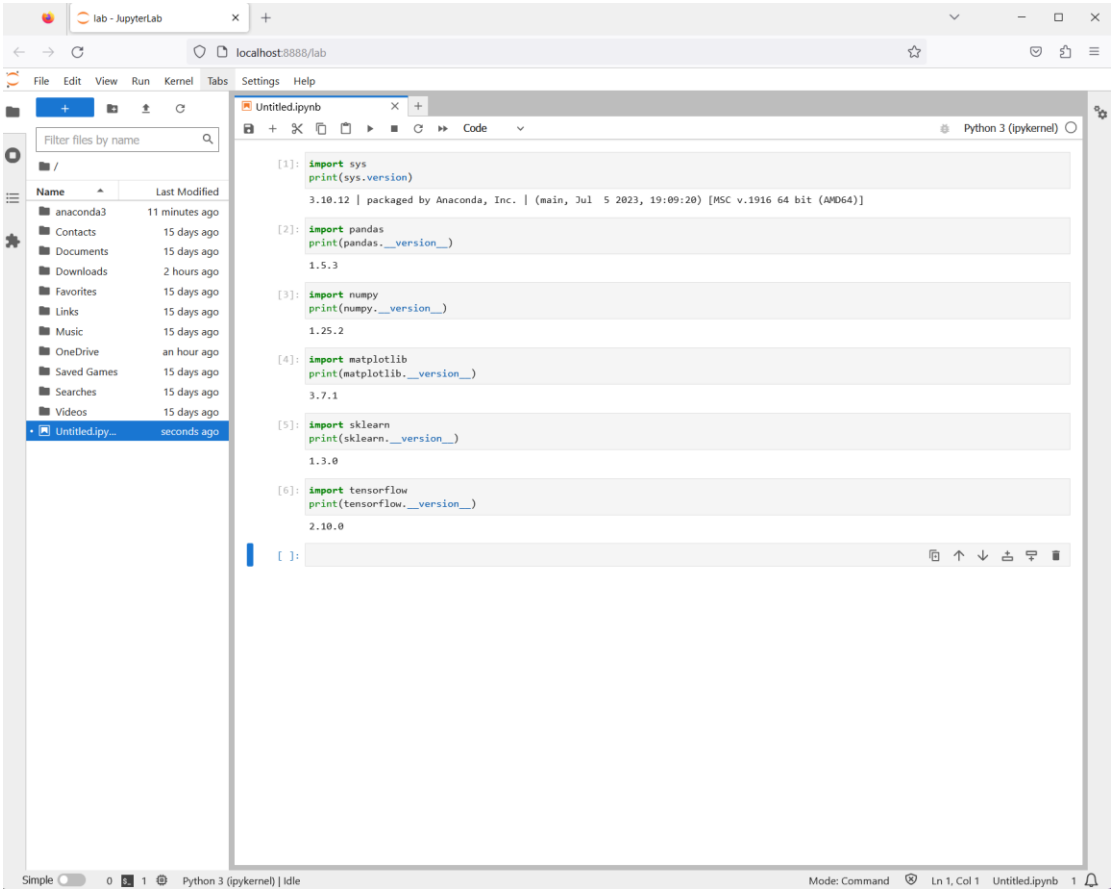
Εικόνα 40. Έναρξη Υπολογιστικού Περιβάλλοντος JupyterLab στο Νέο Περιβάλλον tensorflow

Γίνεται και πάλι εκκίνηση του υπολογιστικού περιβάλλοντος ανάπτυξης, όπου αυτή τη φορά ζητείται αυτόματα η επιλογή προτίμησης της γλώσσας προγραμματισμού «Python 3 (ipykernel)»:



Εικόνα 41. Περιβάλλον Ανάπτυξης JupyterLab στο Νέο Περιβάλλον tensorflow

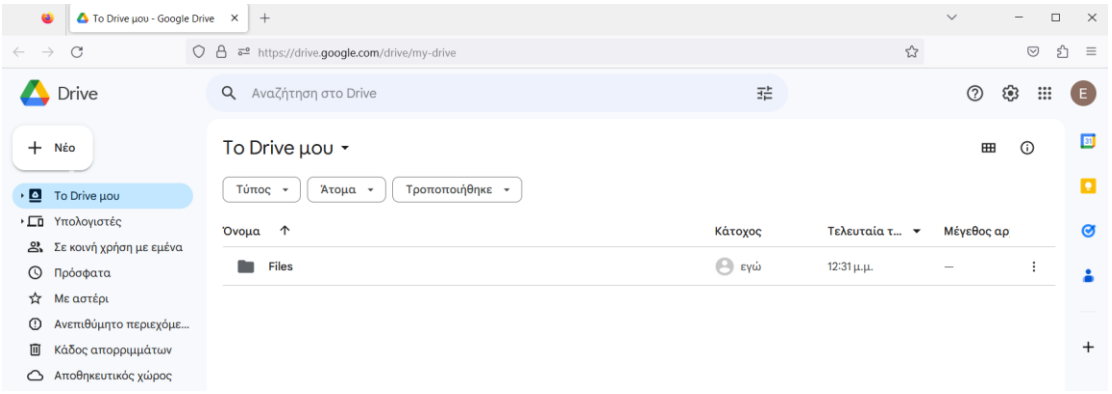
Τέλος, επιβεβαιώνεται η χρήση της έκδοσης 3.10 για τη γλώσσα προγραμματισμού Python και τα απαραίτητα πακέτα εισάγονται κανονικά χωρίς κάποια προειδοποίηση:



Εικόνα 42. Εκτέλεση Κελιών του Notebook στο Νέο Περιβάλλον tensorflow

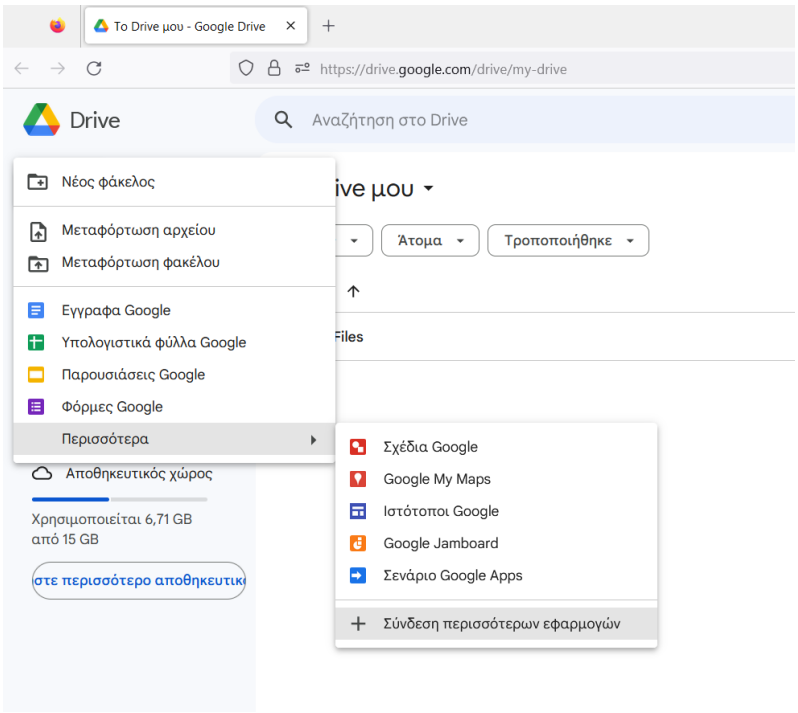
1.2 Google Colaboratory

Ο επόμενος τρόπος, ο οποίος είναι και αυτός που επιλέχτηκε για την υλοποίηση των εφαρμογών της συγκεκριμένης πτυχιακής εργασίας, χρησιμοποιεί την εφαρμογή Google Colaboratory [87], η οποία είναι μια ελεύθερη υπηρεσία εκτέλεσης notebook και λειτουργεί απομακρυσμένα στο Cloud. Επομένως, δεν χρειάζεται κάποια εγκατάσταση, παρά μόνο ένας λογαριασμός Google για πρόσβαση στο Google Drive:

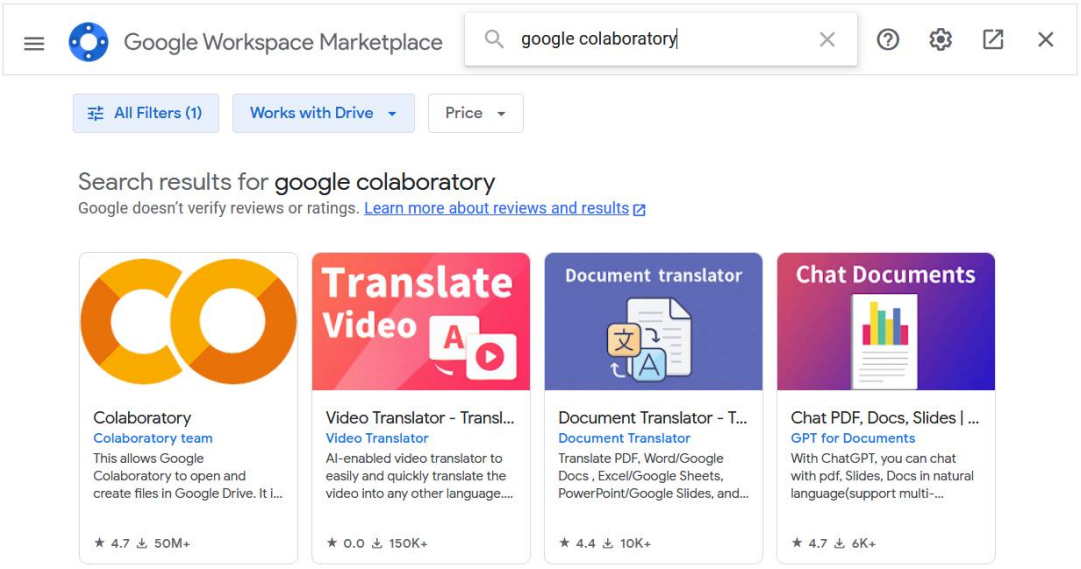


Εικόνα 43. Google Drive του Χρήστη

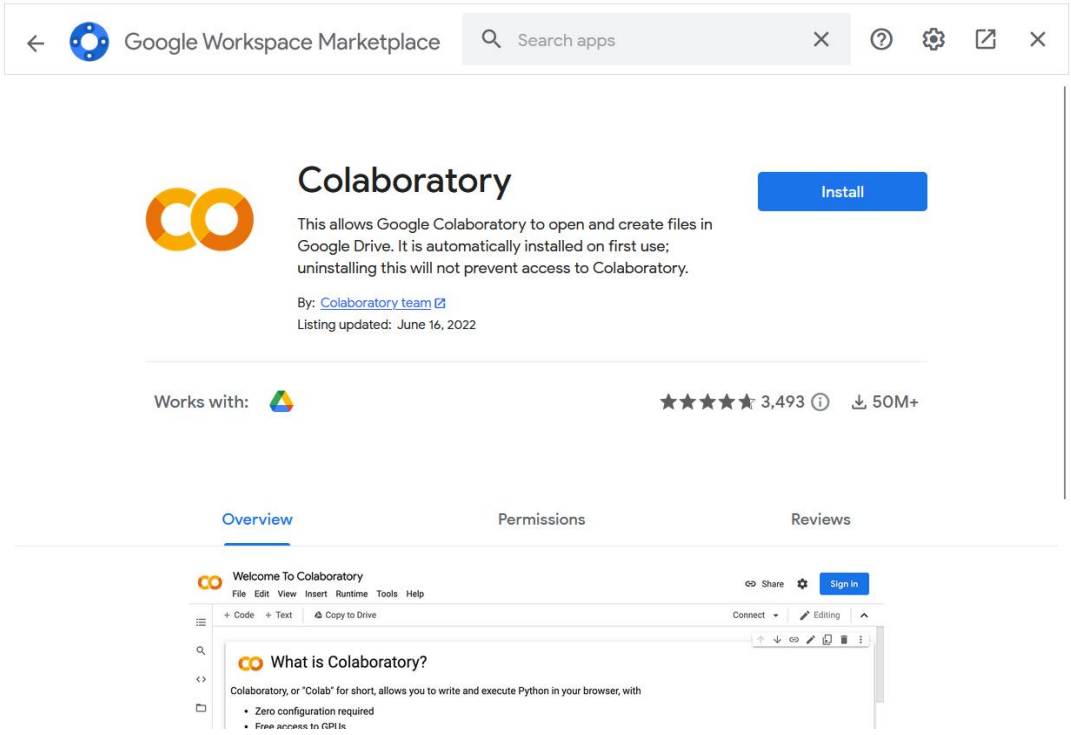
Στη συνέχεια από την επιλογή «+Νέο» κάνουμε εγκατάσταση την εφαρμογή Google Colaboratory, δίνοντας οποιαδήποτε απαραίτητη άδεια:



Εικόνα 44. Αναζήτηση Εφαρμογής Google Colaboratory

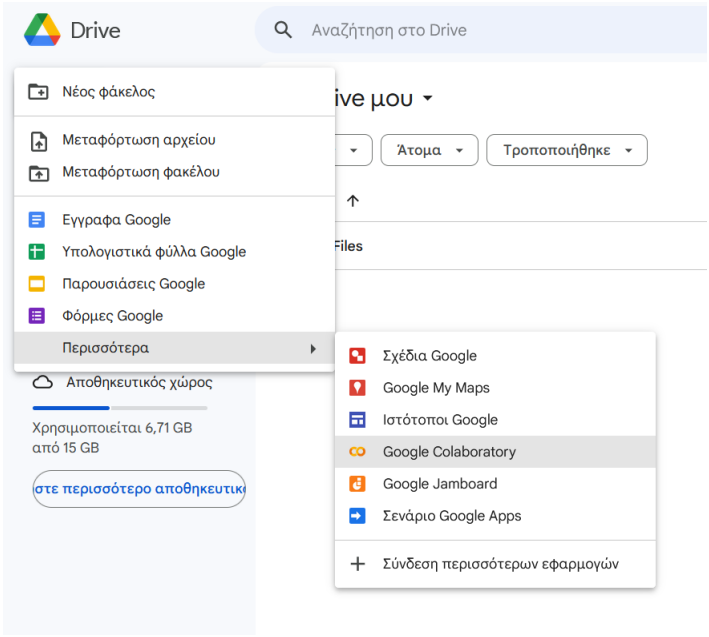


Εικόνα 45. Εύρεση Εφαρμογής Google Colaboratory



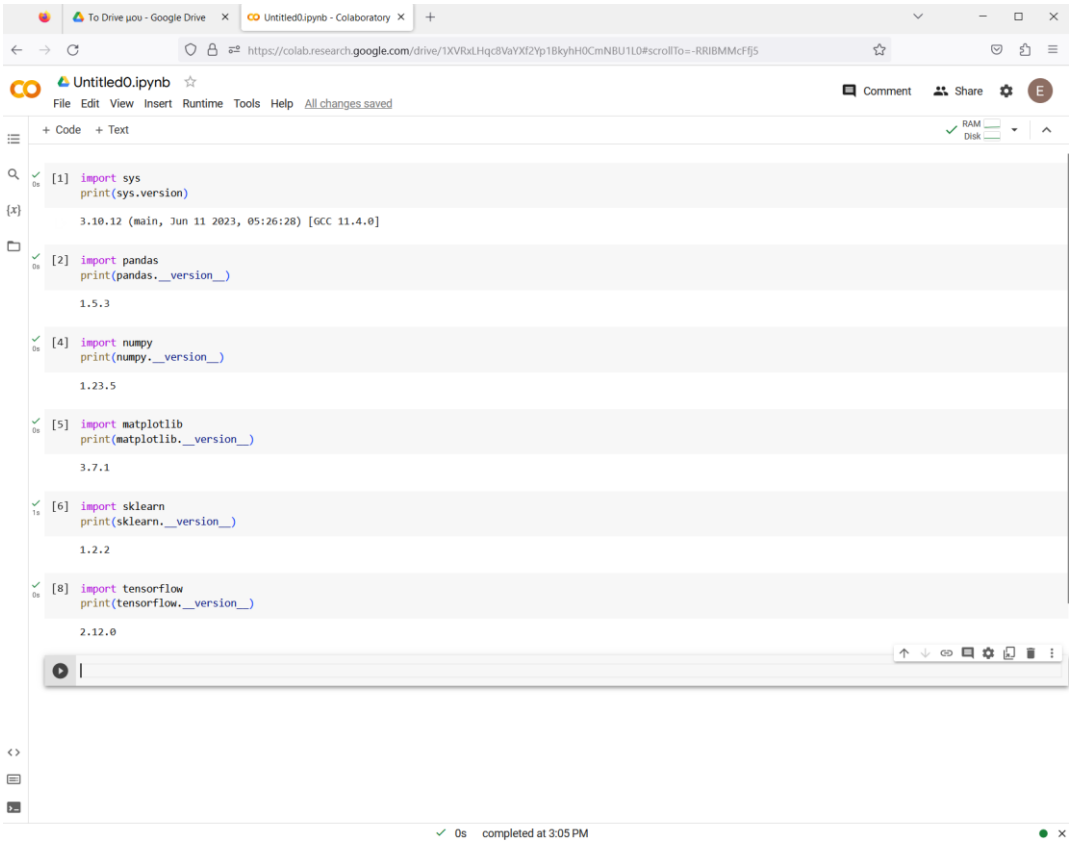
Εικόνα 46. Εγκατάσταση Εφαρμογής Google Colaboratory

Μετά την εγκατάσταση, επιστρέφοντας πίσω στο Google Drive βλέπουμε πλέον τη δυνατότητα εκκίνησης της εφαρμογής Google Colaboratory:



Εικόνα 47. Άνοιγμα Notebook με την Εφαρμογή Google Colaboratory

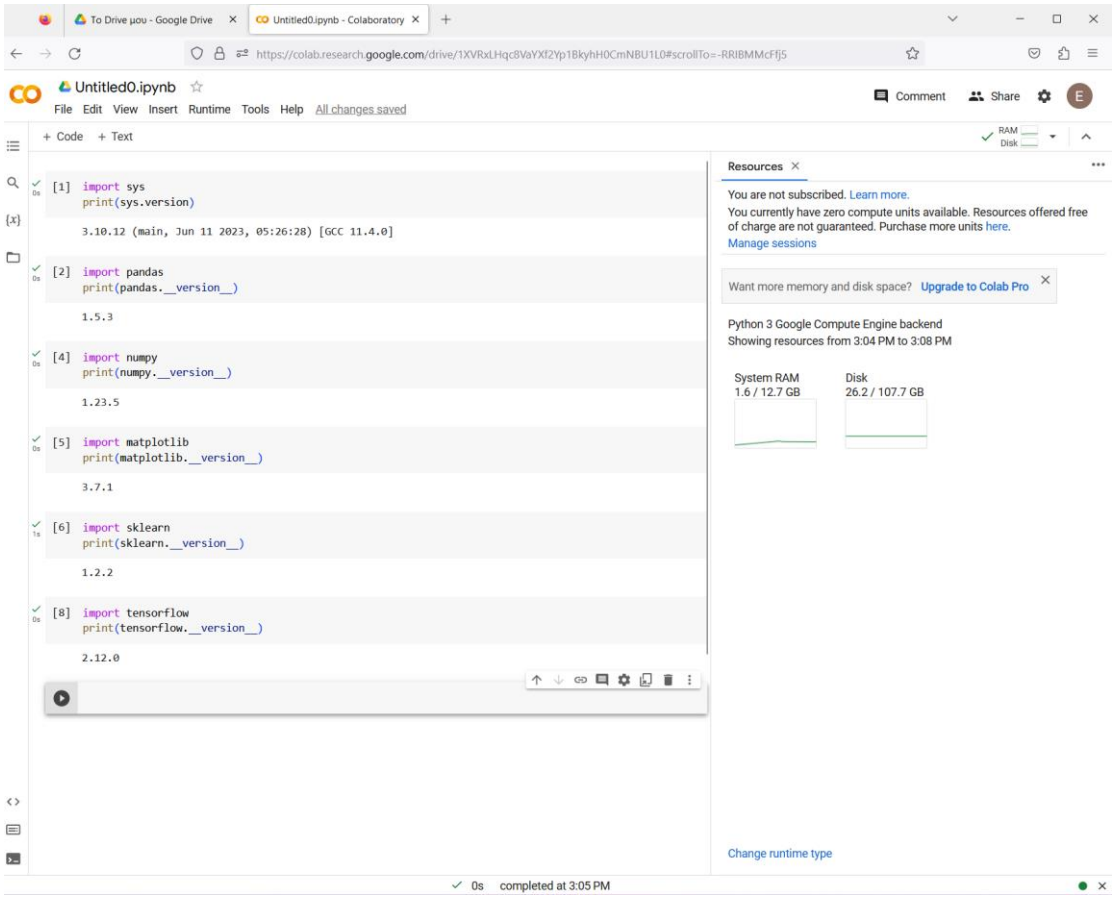
Με την εκκίνηση της εφαρμογής Google Colaboratory, ανοίγει αυτόματα σε νέα καρτέλα ένα notebook, όπου, με παρόμοιο τρόπο όπως στην εφαρμογή JupyterLab, μπορούμε να ελέγξουμε την έκδοση Python που χρησιμοποιείται και να εισάγουμε τα απαραίτητα πακέτα:



Εικόνα 48. Notebook με την Εφαρμογή Google Colaboratory

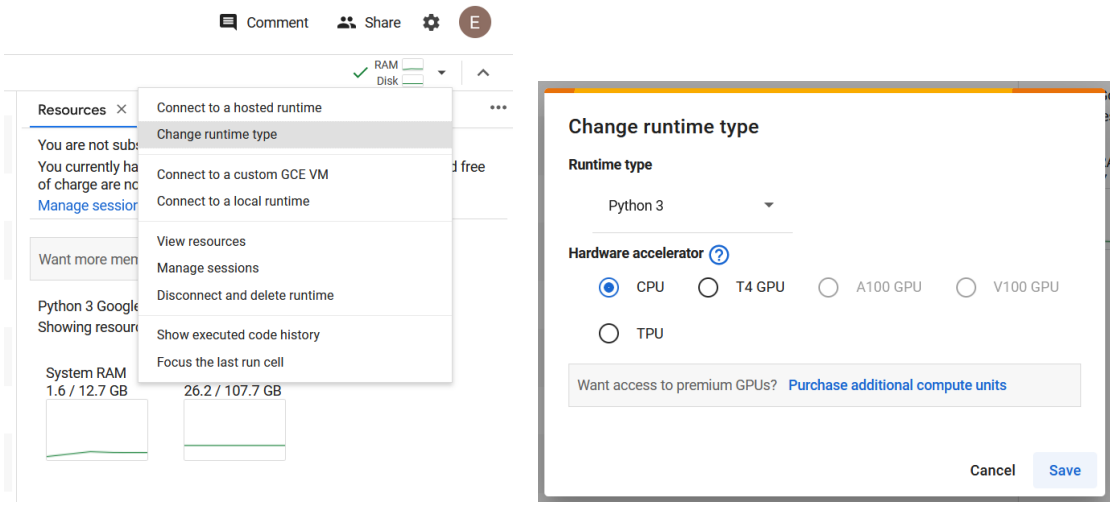
Το περιβάλλον του Google Colaboratory έχει διαθέσιμα όλα τα πακέτα που χρησιμοποιούνται σε εφαρμογές επιστημονικής φύσεως και Μηχανικής Μάθησης, συνεπώς δεν απαιτείται κάποια επιπλέον εγκατάσταση ή ρύθμιση.

Αναφέρθηκε ότι το Google Colaboratory είναι υπηρεσία Cloud, επομένως χρησιμοποιεί απομακρυσμένους πόρους, τους οποίους μπορούμε να ελέγξουμε στο δεξί μέρος του notebook:



Εικόνα 49. Διαθέσιμοι Πόροι του Google Colaboratory

Επιπλέον, δίνεται και η επιλογή από το αναπτυσσόμενο μενού να γίνει αλλαγή του περιβάλλοντος εκτέλεσης με αξιοποίηση κάρτας γραφικών:



Εικόνα 50. Αξιοποίηση Κάρτας Γραφικών στο Περιβάλλον του Google Colaboratory

Επίσης, για είναι εφικτή η διαχείριση των αρχείων δεδομένων τα οποία πρέπει να είναι αποθηκευμένα στο Google Drive, αλλά και η πρόσβασή μας σε αυτά, πρέπει να προσαρτηθεί στην αρχή το ίδιο το Google Drive στο notebook όπου δουλεύουμε κάθε στιγμή, και αυτό γίνεται με τον παρακάτω τρόπο:

```
Input []:
1  from google.colab import drive
2  drive.mount('/content/drive')

Output []:
Mounted at /content/drive
```

Το Output μάς ενημερώνει για τη διαδρομή (path) που προσαρτήσαμε, επομένως για την πρόσβαση σε κάθε αρχείο που χρειαζόμαστε για ανάγνωση, ή για την περίπτωση αποθήκευσης, πρέπει να χρησιμοποιούμε ως πρόθεμα την παραπάνω διαδρομή που έχουμε ορίσει.

Να σημειωθεί ότι κατά την προσάρτηση θα ζητηθεί επιβεβαίωση για να επιτραπεί η πρόσβαση στα αρχεία του Google Drive μέσω του Google λογαριασμού που χρησιμοποιείται από τον εκάστοτε χρήστη.

2 Εφαρμογή σε πρόβλημα Κατηγοριοποίησης

2.1 Περιγραφή Προβλήματος

Η πρώτη εφαρμογή αφορά σε ένα σύνολο δεδομένων το οποίο συλλέχθηκε από πειραματική διάταξη κινητήρα με προσαρτημένο άξονα, και περιγράφει χαρακτηριστικές μετρήσεις που λήφθηκαν από ένα ρουλεμάν τοποθετημένο στην άκρη του άξονα, κατά τη λειτουργία του υπό τρεις διαφορετικές συνθήκες λίπανσης.

Συγκεκριμένα, στο πρώτο πείραμα το ρουλεμάν βρισκόταν στην κανονική του κατάσταση με την συνιστώμενη ποσότητα εργοστασιακού λιπαντικού, στο δεύτερο πείραμα λειτουργούσε χωρίς καθόλου λιπαντικό, ενώ στο τρίτο πείραμα το ρουλεμάν λειτουργούσε με υπερβολική ποσότητα λιπαντικού.

Εφόσον υπάρχουν τρεις διαφορετικές διακριτές καταστάσεις στο πρόβλημα, τότε μπορούμε να το κατατάξουμε στην περίπτωση μιας εφαρμογής Κατηγοριοποίησης (Classification), κατά την οποία ο στόχος για το μοντέλο Βαθιάς Μάθησης που θα αναπτυχθεί θα είναι η αναγνώριση της λειτουργικής κατάστασης του ρουλεμάν, αφού φυσικά εκπαιδευτεί κατάλληλα στο παραπάνω σύνολο δεδομένων. Επιπλέον, η διαδικασία μάθησης του αλγορίθμου θα επιτευχθεί με τον τύπο της Επιβλεπόμενης Μάθησης (Supervised Learning), καθώς είναι γνωστή η πραγματική κατάσταση κάθε στοιχείου δεδομένων, δηλαδή υπάρχουν διαθέσιμες ετικέτες στο σύνολο δεδομένων.

Στα πλαίσια της Προβλεπτικής Συντήρησης, η παραπάνω αποτελεί μια εφαρμογή Ανίχνευσης Βλαβών (Fault Detection), όπου ως βλάβες θεωρούνται οι δυο καταστάσεις με την περίσσεια και την απουσία εργοστασιακού λιπαντικού.

Τα δεδομένα του προβλήματος είναι δημοσίως διαθέσιμα για κατέβασμα διαδικτυακά [88] από τη βάση δεδομένων του Zenodo.

2.2 Στατιστική Ανάλυση Δεδομένων

Σε πρώτο στάδιο, όπως έχει αναφερθεί, είναι σημαντική η ανάλυση των δεδομένων για την καλύτερη κατανόησή τους. Ελέγχονται τα δεδομένα ως προς τα διαθέσιμα χαρακτηριστικά τους, την ύπαρξη ακραίων ή ελλιπών στοιχείων, εξάγονται μέτρα θέσης και διασποράς, οπτικοποιούνται οι καταστάσεις μέσω γραφημάτων, κ.α. Γενικότερα υλοποιούνται εκείνες

οι πρακτικές οι οποίες κρίνονται απαραίτητες, έτσι ώστε να επιτευχθεί μια καλύτερη επίγνωση για τις πληροφορίες που παρέχονται από το συλλεγμένο σύνολο δεδομένων.

Τα δεδομένα τις συγκεκριμένης εφαρμογής βρίσκονται ήδη στην τελική «καθαρή» μορφή τους, και οποιαδήποτε περαιτέρω επεξεργασία είναι αναγκαία, θα γίνει στα υποκεφάλαια ανάπτυξης των μοντέλων Βαθιάς Μάθησης. Ωστόσο, οι βασικότερες τεχνικές στατιστικής ανάλυσης περιγράφονται παρακάτω αναλυτικά και προσφέρονται καθοδηγητικές διαδικασίες, τόσο σε πλαίσια προγραμματισμού όσο και ερμηνείας των δεδομένων.

2.2.1 Εισαγωγή Πακέτων

Σαν πρώτο βήμα, αμέσως μετά την προσάρτηση του Google Drive που αναφέρθηκε παραπάνω, εισάγονται τα πακέτα της Python που θα χρησιμοποιηθούν για τη στατιστική ανάλυση και την οπτικοποίηση των δεδομένων ως εξής:

```
Input []:
1  import pandas
2  import matplotlib
```

Και ο έλεγχος των εκδόσεων που χρησιμοποιούνται γίνεται με τον παρακάτω κώδικα:

```
Input []:
1  print('pandas', pandas.__version__)
2  print('matplotlib', matplotlib.__version__)

Output []:
pandas 1.5.3
matplotlib 3.7.1
```

Για καλύτερη διαχείριση του κώδικα θα γίνει η εισαγωγή των πακέτων ως εξής:

```
Input []:
1  import pandas as pd
2  import matplotlib.pyplot as plt
```

Επομένως, για να γίνει κλήση συναρτήσεων και εντολών της βιβλιοθήκης Pandas αρκεί το πρόθεμα pd, ενώ όσον αφορά στη βιβλιοθήκη Matplotlib, το module που εισάχθηκε ως plt είναι αρκετό για τα διαγράμματα που θα χρειαστούν.

2.2.2 Εισαγωγή Δεδομένων

Επόμενο βήμα είναι η εισαγωγή των δεδομένων τα οποία θα αναλυθούν και που βρίσκονται αποθηκευμένα στο Google Drive, ως εξής:

```
Input []:
1  data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
2                      thesis/data/bearing_vibration_metrics.csv')
```

Τα δεδομένα διαβάστηκαν ως αρχείο τύπου .csv, όπως είναι ήδη αποθηκευμένα στη διαδρομή του Google Drive που έχουμε ορίσει στον κώδικα, και καταχωρήθηκαν μέσω της βιβλιοθήκης Pandas ως αντικείμενο τύπου DataFrame στη μεταβλητή data, την οποία καλούμε παρακάτω:

```
Input []:
1  data
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	1
...	...	...	...	...	...	...
8564	0.000728	0.880769	7.337692	8.304615	26.866153	3
8565	0.000703	0.853846	6.926154	8.102308	26.806153	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3
8568	0.000708	0.866154	7.378462	8.506923	26.799999	3

8569 rows × 6 columns

Παρατηρούμε ότι το παραπάνω σύνολο δεδομένων αποτελείται από 8569 εγγραφές (ξεκινώντας την αύξουσα αρίθμηση των γραμμών από το μηδέν) και από 6 στήλες, οι οποίες περιγράφουν όλα τα χαρακτηριστικά των στοιχείων.

Συγκεκριμένα τα χαρακτηριστικά είναι τα V-RMS, a-RMS, a-Peak, Crest, Temperature και Bearing State. Τα πέντε πρώτα περιγράφουν τις μετρήσεις που συλλέχθηκαν από το

ρουλεμάν κατά τη λειτουργία του, ενώ το τελευταίο χαρακτηριστικό δηλώνει το πείραμα κατά το οποίο πάρθηκαν οι μετρήσεις.

2.2.3 Ανάλυση Κραδασμών

Προτού προχωρήσουμε σε περαιτέρω ανάλυση των δεδομένων, είναι σημαντικό να γίνει κατανοητή η συνεισφορά του κάθε χαρακτηριστικού για τα δεδομένα.

Αρχικά, ως κραδασμός ή δόνηση ορίζεται η ταλάντωση ενός αντικειμένου γύρω από τη θέση ισορροπίας του. Συνεπώς, ο τομέας της Ανάλυσης Κραδασμών [89] έχει ως αντικείμενο τη μελέτη αυτής της κίνησης και όλων των παράγωγων μεγεθών τα οποία προκύπτουν από αυτήν. Οι βιομηχανίες εξαρτώνται από τη λειτουργία μηχανημάτων παραγωγής τα οποία περιλαμβάνουν πληθώρα εξαρτημάτων τα οποία αναπόφευκτα παράγουν δονήσεις κατά τη λειτουργία τους, επομένως είναι προφανές ότι η Ανάλυση Κραδασμών αποτελεί βασική τεχνική που εφαρμόζεται από τις πολιτικές συντήρησης.

Ειδικότερα, ένα εξάρτημα, κατά την λειτουργία του, μετατοπίζεται κάθε χρονική στιγμή γύρω από τη θέση ισορροπίας του, πραγματοποιώντας περιοδική κίνηση γύρω από αυτό. Ο ρυθμός μεταβολής αυτής της μετατόπισης, δηλαδή η παράγωγος της μετατόπισης ως προς τον χρόνο, ορίζει την ταχύτητα του εξαρτήματος για κάθε χρονική στιγμή. Κατά τον ίδιο τρόπο, ο ρυθμός μεταβολής της ταχύτητας, δηλαδή η παράγωγος της ταχύτητας ως προς τον χρόνο, ορίζει την επιτάχυνση του εξαρτήματος για κάθε χρονική στιγμή.

Σε μια περιοδική κίνηση γύρω από ένα σημείο ισορροπίας, η ταχύτητα ενός εξαρτήματος γίνεται μέγιστη όταν επιστρέφει σε αυτό, και μηδενίζεται όταν βρίσκεται στη μεγαλύτερη ακραία απόσταση από αυτό, δηλαδή όταν η μετατόπισή του είναι η μέγιστη δυνατή. Επομένως, δεν αρκεί μία μόνο μέτρηση σε μια χρονική στιγμή λειτουργίας για να οδηγήσει σε κάποιο συμπέρασμα για την κατάσταση υγείας ενός μηχανήματος. Για τον λόγο αυτό, για την εξαγωγή μιας αντιπροσωπευτικής μέτρησης ταχύτητας, αλλά και επιτάχυνσης, συλλέγονται όλες οι στιγμιαίες τιμές μέσα σε ένα προκαθορισμένο χρονικό διάστημα δευτερολέπτων και τελικά υπολογίζεται ένα μέσο αποτέλεσμα από αυτές. Συγκεκριμένα στην Ανάλυση Κραδασμών, επιλέγεται το μέτρο της Τετραγωνικής Ρίζας του Μέσου Τετραγώνου (Root Mean Square – RMS) των στιγμιαίων τιμών όπως ορίζεται με τον παρακάτω τύπο:

$$x_{RMS} = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} x_i^2}$$

Συνεπώς, τα συνήθη μεγέθη που επιλέγονται και μελετώνται είναι αυτά της ταχύτητας-RMS (velocity-RMS) ή v-RMS, και της επιτάχυνσης-RMS (acceleration-RMS) ή a-RMS όπως υπολογίζονται παρακάτω:

$$v - RMS = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} v_i^2} \quad και \quad a - RMS = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} a_i^2}$$

Όπου:

- n είναι το συνολικό πλήθος των στιγμιαίων μετρήσεων που λήφθηκαν
- v_i είναι όλες οι στιγμιαίες ταχύτητες που λήφθηκαν στον προκαθορισμένο χρόνο
- a_i είναι όλες οι στιγμιαίες επιταχύνσεις που λήφθηκαν στον προκαθορισμένο χρόνο

Επιπλέον μεγέθη της Ανάλυσης Κραδασμών αποτελούν τα μεγέθη Αιχμής (Peak). Συγκεκριμένα, η ταχύτητα Αιχμής (velocity-Peak) ή v-Peak δηλώνει τη μέγιστη ταχύτητα που καταγράφεται κατά τη λήψη των μετρήσεων, η οποία είναι ουσιαστικά η ταχύτητα του εξαρτήματος όταν αυτό περνάει από τη θέση ισορροπίας του, προτού αρχίσει να επιβραδύνει όσο πλησιάζει τη μέγιστη μετατόπισή του από αυτό, όπου και η ταχύτητά του μηδενίζεται. Αντίστοιχα η επιτάχυνση Αιχμής (acceleration-Peak) ή a-Peak δηλώνει τη μέγιστη τιμή επιτάχυνσης που καταγράφεται κατά τον καθορισμένο χρόνο μετρήσεων, η οποία παρατηρείται στα άκρα της ταλάντωσης του εξαρτήματος, προτού ξεκινήσει και πάλι την κίνησή του για να επιστρέψει και να περάσει από το σημείο ισορροπίας.

Επίσης, σημαντικό μέγεθος της Ανάλυσης Κραδασμών, το οποίο θεωρείται και από τα πιο αντιπροσωπευτικά χαρακτηριστικά που μπορούν να χρησιμοποιηθούν για την εξαγωγή συμπερασμάτων υγείας της κατάστασης των μηχανημάτων, είναι ο Συντελεστής Κορυφής (Crest Factor). Ο Συντελεστής Κορυφής υπολογίζεται ως ο λόγος της τιμής Αιχμής προς τη μέση τιμή RMS του μεγέθους το οποίο μελετάται. Επομένως, υπάρχουν ο Συντελεστής Κορυφής για το μέγεθος της ταχύτητας και της επιτάχυνσης, τα οποία υπολογίζονται αντίστοιχα ως εξής:

$$v - Crest = \frac{v - Peak}{v - RMS} \quad και \quad a - Crest = \frac{a - Peak}{a - RMS}$$

Τελικά, ο κλάδος της Ανάλυσης Κραδασμών προσφέρει αποδοτικές και μη-παρεμβατικές μεθόδους για τη παρακολούθηση μηχανημάτων και εξαρτημάτων μέσω των παραπάνω μεγεθών, αλλά και άλλων παράγωγών τους. Σε αυτήν την προσπάθεια έχουν συμβάλει σημαντικά οι καινοτόμες τεχνολογίες της Βιομηχανίας 4.0, όπως οι «έξυπνοι» αισθητήρες και το Βιομηχανικό Διαδίκτυο των Πραγμάτων (Industrial Internet of Things – IioT), καθώς πλέον τα χαρακτηριστικά που μπορούν να μετρήσουν οι αισθητήρες τελευταίας

τεχνολογίας είναι πολύ περισσότερα, οι μετρήσεις είναι πολύ πιο ακριβείς, και με κατάλληλα εγκατεστημένα συστήματα είναι δυνατόν οι παραπάνω μετρήσεις να διαβιβαστούν κατάλληλα και να αποθηκευτούν για περεταίρω μελέτη και ανάλυση. Συνεπώς, η ανάλυση των σημάτων των κραδασμών που παράγονται κατά τη λειτουργία των βιομηχανικών συστημάτων αποτελεί πλέον αναπόσπαστο κομμάτι κάθε στρατηγικής συντήρησης, καθώς μπορούν να περιγράψουν σε πολυδιάστατο βαθμό την κατάσταση υγείας και την καταπόνηση των μηχανημάτων παραγωγής.

2.2.4 Ερμηνεία Χαρακτηριστικών

Για τα δεδομένα του προβλήματος που εξετάζεται, αναφέρθηκαν τα V-RMS, a-RMS, a-Peak, Crest, Temperature και Bearing State ως χαρακτηριστικά του συνόλου δεδομένων. Τα πέντε πρώτα περιλαμβάνουν τις μετρήσεις όπως καταγράφηκαν από κατάλληλο αισθητήρα προσαρτημένο στο ρουλεμάν της πειραματικής διάταξης, ενώ το τελευταίο δηλώνει τη συνθήκη γράσου υπό την οποία λειτουργούσε το ρουλεμάν.

Συγκεκριμένα:

- Το V-RMS εκφράζεται σε m/s και αποτελεί το μέσο αποτέλεσμα της ταχύτητας του ρουλεμάν για κάθε μέτρηση, όπως περιγράφηκε ο υπολογισμός του παραπάνω. Είναι αντιπροσωπευτικό μέγεθος των συχνότερων τύπων υπερφόρτωσης των μηχανών, καθώς μακροχρόνιες αυξημένες επιβαρύνσεις μπορούν να προκαλέσουν σημαντικές βλάβες στα εξαρτήματα, αλλά και σε πιο έντονο βαθμό να τα καταστρέψουν.
- Το a-RMS εκφράζεται σε m/s^2 και αποτελεί το μέσο αποτέλεσμα της επιτάχυνσης του ρουλεμάν για κάθε μέτρηση, όπως περιγράφηκε ο υπολογισμός του παραπάνω. Είναι αντιπροσωπευτικό μέγεθος της φθοράς των εξαρτημάτων, ειδικά όταν έρχονται σε μηχανική επαφή μεταξύ τους, αλλά και της κατάστασης λίπανσης, όπου μπορεί να υπάρχουν επιβλαβή ίχνη ξένων ουσιών ή νερού στο γράσου.
- Το a-Peak εκφράζεται σε m/s^2 και αποτελεί τη μέγιστη τιμή επιτάχυνσης του ρουλεμάν η οποία καταγράφηκε από τον αισθητήρα κατά το χρονικό διάστημα κάθε μέτρησης. Είναι αντιπροσωπευτικό μέγεθος των δυνάμεων που ασκούνται στα εξαρτήματα, και περιγράφει τυχόν ξαφνικούς κραδασμούς που μπορεί να συμβούν σε περίπτωση συγκρούσεων.
- Το Crest στο συγκεκριμένο σύνολο δεδομένων περιγράφει τον λόγο της μέγιστης τιμής επιτάχυνσης a-Peak προς τη μέση τιμή a-RMS και δεν εκφράζεται σε κάποια

μορφή μονάδων μέτρησης. Είναι αντιπροσωπευτικό μέγεθος της γενικότερης κατάστασης υγείας των εξαρτημάτων και αποτελεί χαρακτηριστική τιμή αξιολόγησης της γενικότερης ανάλυσης των σημάτων κραδασμού. Υψηλότερες τιμές Crest υποδηλώνουν μεγαλύτερες ακραίες τιμές επιτάχυνσης έναντι της μέσης υπολογιζόμενης, που σημαίνει ότι τα εξαρτήματα τα οποία έχουν υποστεί βλάβες παράγουν σήματα υψηλότερης συχνότητας σε συντομότερο χρονικό διάστημα από το κανονικό.

- Το Temperature δηλώνει τη θερμοκρασία του ρουλεμάν κατά τη λειτουργία του, εκφράζεται σε βαθμούς Κελσίου $°C$ και η χρησιμότητά του είναι προφανής.
- Το Bearing State αποτελεί πρόσθετο χαρακτηριστικό στο σύνολο δεδομένων και δεν προέρχεται από κάποια μέτρηση. Δηλώνει απλώς τους ακέραιους αναγνωριστικούς αριθμούς των πειραμάτων, κατά τα οποία καταγράφηκαν οι πέντε προηγούμενες μετρήσεις από τον αισθητήρα, και που χρησιμοποιήθηκαν κατά τη συσσωμάτωση όλων των μετρήσεων σε ένα συνολικό πίνακα δεδομένων για την εύκολη διάκριση κάθε λιπαντικής συνθήκης. Συγκεκριμένα, ο αριθμός 1 δηλώνει ότι το ρουλεμάν βρισκόταν στην κανονική του κατάσταση, το 2 ότι λειτουργούσε χωρίς εργοστασιακό λιπαντικό, ενώ το 3 ότι λειτουργούσε με υπερβολική ποσότητα λιπαντικού.

Είναι σημαντικό να αναφερθεί ότι η καταγραφή των μετρήσεων από τον αισθητήρα πραγματοποιούνταν κάθε 2 λεπτά, επομένως τα στοιχεία των δεδομένων καθενός από τα τρία πειράματα αντιπροσωπεύουν μια χρονοσειρά με βήμα 2 λεπτών.

2.2.5 Συνοπτικές Πληροφορίες Δεδομένων

Αφού έγινε κατανοητό το τί αντιπροσωπεύει κάθε χαρακτηριστικό του συνόλου δεδομένων, μπορεί να συνεχιστεί η διαδικασία της στατιστικής ανάλυσης για την εξαγωγή περαιτέρω πληροφοριών.

Γενικότερες πληροφορίες για το σύνολο δεδομένων παίρνουμε με την παρακάτω εντολή:

```
Input []:
1  data.info()

Output []:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8569 entries, 0 to 8568
Data columns (total 6 columns):
 #   Column         Non-Null Count  Dtype
```

```
---  ------         --------------  -----
 0   V-RMS          8569 non-null   float64
 1   a-RMS          8569 non-null   float64
 2   a-Peak         8569 non-null   float64
 3   Crest          8569 non-null   float64
 4   Temperature    8569 non-null   float64
 5   Bearing State  8569 non-null   int64
dtypes: float64(5), int64(1)
memory usage: 401.8 KB
```

Συνεπώς, ενημερωνόμαστε για το γεγονός ότι όντως τα δεδομένα αποτελούν αντικείμενο DataFrame της βιβλιοθήκης Pandas, και προφανώς υπακούουν σε κάθε επιμέρους εντολή που σχετίζεται με αυτόν τον τύπο αντικειμένου. Επίσης, όπως παρατηρήθηκε και κατά την εισαγωγή των δεδομένων προηγουμένως, το πλήθος των εγγραφών είναι 8569, ξεκινώντας την αύξουσα αρίθμηση από το μηδέν, σύμφωνα με το αντικείμενο ευρετηρίου RangeIndex της Pandas, ενώ το συνολικό πλήθος των στηλών, οι οποίες περιγράφονται και με τα αντίστοιχα ονόματά τους, είναι 6.

Επίσης, ειδικότερα για τις στήλες, παρατηρούμε ότι καθεμία περιλαμβάνει 8569 έγκυρες εγγραφές (non-null), που σημαίνει ότι ο πίνακας δεδομένων είναι πλήρης, χωρίς την ύπαρξη ελλιπών τιμών. Ακόμη, τα 5 βασικά χαρακτηριστικά V-RMS, a-RMS, a-Peak, Crest και Temperature, τα οποία δηλώνουν τις μετρήσεις που καταγράφηκαν κατά τα πειράματα, είναι αποθηκευμένα υπό μορφή float64, αντίστοιχη των πραγματικών αριθμητικών τιμών, ενώ το χαρακτηριστικό Bearing State είναι αποθηκευμένο υπό τη μορφή int64, η οποία υποδηλώνει ακέραιες τιμές.

Συγκεκριμένα για το χαρακτηριστικό Bearing State χρησιμοποιούμε την παρακάτω εντολή:

```
Input []:
1  data['Bearing State'].value_counts()

Output []:
1    3296
3    2637
2    2636
Name: Bearing State, dtype: int64
```

Παρατηρούμε ότι για το χαρακτηριστικό Bearing State του συνόλου δεδομένου, το οποίο έχουμε αποθηκεύσει με το όνομα data, χρησιμοποιώντας τη συνάρτηση value_counts(), εξάγουμε το πλήθος των διαφορετικών τιμών της στήλης Bearing State, όπου εδώ είναι οι ακέραιοι 1, 2 και 3, καθώς και το πλήθος των στοιχείων-γραμμών του συνόλου δεδομένων data που αντιστοιχούν σε καθεμία από αυτές τις τρεις διαφορετικές τιμές.

Συνεπώς, για την κανονική κατάσταση του ρουλεμάν έχουμε 3296 εγγραφές, για την κατάσταση χωρίς λιπαντικό έχουμε 2637, και για την κατάσταση με υπερβολική ποσότητα εργοστασιακού λιπαντικού έχουμε 2636 εγγραφές. Επίσης, επιβεβαιώνουμε ότι το πλήθος όλων των παραπάνω επιμέρους εγγραφών είναι συνολικά 8569, όσο και το συνολικό πλήθος των στοιχείων του συνόλου δεδομένων data. Προφανώς, το πρώτο πείραμα διήρκησε για περισσότερο χρόνο από τα υπόλοιπα, αν αναλογιστούμε ότι κάθε εγγραφή καταγράφεται κάθε 2 λεπτά λειτουργίας.

2.2.6 Έλεγχος Ελλιπών Τιμών

Αν και πληροφορηθήκαμε από τη συνάρτηση info() για το πλήθος των έγκυρων τιμών του συνόλου δεδομένων data, με τον παρακάτω τρόπο ελέγχουμε το ακριβές πλήθος των ελλιπών τιμών για κάθε στήλη:

```
Input []:
1  data.isna()
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	False	False	False	False	False	False
1	False	False	False	False	False	False
2	False	False	False	False	False	False
3	False	False	False	False	False	False
4	False	False	False	False	False	False
...	...	...	...	...	...	...
8564	False	False	False	False	False	False
8565	False	False	False	False	False	False
8566	False	False	False	False	False	False
8567	False	False	False	False	False	False
8568	False	False	False	False	False	False

8569 rows × 6 columns

Το αποτέλεσμα που εξάγεται είναι επίσης ένας πίνακας δεδομένων τύπου DataFrame, όπου σε κάθε κελί υπάρχει το λογικό αποτέλεσμα True (Αληθές), αν η αντίστοιχη τιμή του

αρχικού συνόλου δεδομένων data είναι όντως ελλιπής, ενώ το λογικό αποτέλεσμα False (Ψευδές), αν η τιμή του αντίστοιχου αρχικού κελιού είναι έγκυρη. Προφανώς, σε έναν πίνακα διάστασης 8569x6 είναι αδύνατο να βγάλουμε συμπέρασμα με αυτό το εξαγόμενο αποτέλεσμα, και για αυτόν τον λόγο αυτό εκτελούμε την παρακάτω εντολή:

```
Input []:
1  data.isna().sum()

Output []:
V-RMS           0
a-RMS           0
a-Peak          0
Crest           0
Temperature     0
Bearing State   0
dtype: int64
```

Συνεπώς, βλέπουμε ξεκάθαρα πως για κάθε στήλη το πλήθος των ελλιπών τιμών είναι μηδέν, καθώς η συνάρτηση sum() αντιμετωπίζει κάθε λογικό αποτέλεσμα True αντιστοιχίζοντάς το με τη μονάδα, και κάθε λογικό False με το μηδέν.

Σε περίπτωση που υπήρχαν ελλιπές τιμές, για να δούμε ακριβώς σε ποια στοιχεία υπάρχει έστω και ένα μη-έγκυρο κελί, ακολουθούμε την παρακάτω εντολή:

```
Input []:
1  data[data.isna().any(axis='columns')]

Output []:
```

V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State

Το εξαγόμενο αποτέλεσμα είναι αντικείμενο DataFrame, αλλά στην περίπτωσή μας είναι κενό, καθώς δεν υπάρχουν κελιά με μη-έγκυρες τιμές.

Επίσης, με την παρακάτω εντολή παίρνουμε το ευρετήριο των παραπάνω ελλιπών στοιχείων:

```
Input []:
1  data[data.isna().any(axis='columns')].index

Output []:
Int64Index([], dtype='int64')
```

Το αντικείμενο ευρετηρίου Int64Index είναι επίσης κενό, καθώς δεν υπάρχουν αντίστοιχες μη-έγκυρες τιμές, αλλά σε περίπτωση που υπήρχαν, το παραπάνω στοιχείο θα περιείχε όλους εκείνους τους ακέραιους που θα αντιστοιχούσαν σε κάθε γραμμή του συνόλου δεδομένων data (από το 0 μέχρι το 8568) στην οποία θα υπήρχε τουλάχιστον μια μη-έγκυρη τιμή.

2.2.9 Διαχείριση Ελλιπών Τιμών

Αφού επιβεβαιωθεί η ύπαρξη ελλιπών τιμών, η πρώτη επιλογή είναι να διαγραφούν με τον παρακάτω τρόπο:

```
Input []:
1  data.dropna(axis='index', how='any', inplace=True)
```

Με την παραπάνω εντολή αντικαθίστανται (inplace=True) το σύνολο δεδομένων data, με εκείνο το αντικείμενο DataFrame από το οποίο πλέον λείπουν εκείνα τα στοιχεία-γραμμές (axis='index') στα οποία υπάρχει έστω και ένα κελί (how='any') μη-έγκυρης τιμής.

Ωστόσο, αν η διαγραφή των μη-έγκυρων κελιών διαταράσσει κατά πολύ το σύνολο των δεδομένων, και το πλήθος των ελλιπών τιμών είναι μικρό, τότε μπορούν να αντικατασταθούν και να «γεμίσουν» τα αντίστοιχα μη-έγκυρα κελιά με τον παρακάτω τρόπο:

```
Input []:
1  values={'V-RMS': data['V-RMS'].median(),
2          'a-RMS': data['a-RMS'].median(),
3          'a-Peak': data['a-Peak'].median(),
4          'Crest': data['Crest'].median(),
5          'Temperature': data['Temperature'].median()}
6  data.fillna(value=values, axis='index', inplace=True)
```

Σύμφωνα με την παραπάνω σειρά εντολών, δημιουργήθηκε ένα λεξικό (dictionary) με λέξεις κλειδιά τις πέντε στήλες του συνόλου δεδομένων data, όπου στην καθεμία αντιστοιχίστηκε η διάμεσος από όλες τις τιμές της εκάστοτε στήλης. Στη συνέχεια, με τη συνάρτηση fillna() το σύνολο δεδομένων data αντικαθιστάται με εκείνο το αντικείμενο DataFrame όπου οι μη-έγκυρες τιμές σε κάθε στήλη αντικαθιστώνται με τις διαμέσους κάθε στήλης όπως έχουν οριστεί στο λεξικό. Προφανώς το «γέμισμα» των μη-έγκυρων κελιών μπορεί να γίνει με εκείνες τις τιμές που κρίνονται κατάλληλες για κάθε πρόβλημα που αντιμετωπίζεται, καθώς και με άλλους τρόπους εκτός της συνάρτησης fillna().

Στην συγκεκριμένη περίπτωση, η εκτέλεση των παραπάνω εντολών δεν θα έχει αντίκτυπο για το σύνολο δεδομένων data, καθώς δεν υπάρχουν μη-έγκυρες τιμές πουθενά.

ΣΗΜΑΝΤΙΚΗ ΠΑΡΑΤΗΡΗΣΗ

Προσοχή, η παραπάνω σειρά εντολών «γεμίσματος» των κελιών έχει καθοδηγητικό ρόλο, καθώς, έτσι όπως είναι ορισμένο το σύνολο δεδομένων data, ο υπολογισμός των διαμέσων ακολουθεί εσφαλμένη τακτική. Αυτό οφείλεται στο ότι το σύνολο δεδομένων data αποτελείται από τρεις διαφορετικές καταστάσεις (Bearing State) και επομένως οι διάμεσοι που υπολογίζονται λαμβάνουν υπόψη όλες τις 8569 τιμές κάθε στήλης, κάτι που είναι εσφαλμένο. Επίσης, στο παραπάνω λεξικό δεν λαμβάνονται υπόψη τυχόν μη-έγκυρες τιμές της τελευταίας στήλης Bearing State, κάτι που θα περιέπλεκε κατά πολύ την αντικατάστασή τους, καθώς αποτελείται από αναγνωριστικές τιμές ακεραίων των πειραμάτων. Επομένως, αν κάποιος ήθελε να ακολουθήσει τον δεύτερο τρόπο διαχείρισης των ελλιπών τιμών θα έπρεπε να κάνει το «γέμισμα» για κάθε κατάσταση πειράματος ξεχωριστά, και στη συνέχεια να συσσωματώσει εκ νέου τα τρία αποτελέσματα. Σε παρακάτω υποκεφάλαιο δίνονται αντίστοιχες οδηγίες που βοηθούν στο χώρισμα και τη μετέπειτα συσσωμάτωση των τριών καταστάσεων.

2.2.7 Έλεγχος Διπλότυπων Τιμών

Σε περίπτωση που πρέπει να γίνει έλεγχος διπλότυπων ή πολλαπλών εγγραφών, το οποίο συνήθως μπορεί να οφείλεται σε κάποιο σφάλμα κατά την καταγραφή ή τη διάδοση μετρήσεων από τον ίδιο τον αισθητήρα ή το εγκατεστημένο υπολογιστικό σύστημα διαχείρισης των δεδομένων, ακολουθούνται οι παρακάτω εντολές:

```
Input []:
1  data.duplicated(keep='first')

Output []:
0       False
1       False
2       False
3       False
4       False
        ...
8564    False
8565    False
8566    False
```

```
8567    False
8568    False
Length: 8569, dtype: bool
```

Όπως και στην περίπτωση των ελλιπών τιμών, κάθε λογικό αποτέλεσμα True αντιστοιχεί σε κάθε πολλαπλή εγγραφή ολόκληρης της γραμμής μετά από την πρώτη εμφάνισή της στο σύνολο δεδομένων, ενώ κάθε λογικό False αντιστοιχεί σε κάθε πρωτότυπη γραμμή στοιχείων. Επειδή, όπως και προηγουμένως, είναι δύσκολο να βγει με αυτόν τον τρόπο κάποιο αποτέλεσμα για το αν υπάρχουν διπλότυπα ή όχι, αλλά και για το ποια είναι αυτά, εκτελούμε το παρακάτω:

```
Input []:
1  data.duplicated(keep='first').sum()

Output []:
0
```

Παρατηρούμε ότι το σύνολο όλων των διπλότυπων είναι μηδενικό, αλλά σε περίπτωση που θέλαμε να δούμε όλες τις όμοιες εγγραφές, μετά την κάθε πρωτότυπη καταγραφή, θα ακολουθήσουμε τα παρακάτω:

```
Input []:
1  data[data.duplicated(keep='first')]

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State

Παραπάνω εμφανίζονται σε αντικείμενο τύπου DataFrame όλες οι πολλαπλές εγγραφές εκτός των πρωτότυπων καταγραφών του συνόλου δεδομένων data, αλλά στην περίπτωσή μας είναι κενό καθώς δεν υπάρχουν πολλαπλές όμοιες γραμμές.

```
Input []:
1  data[data.duplicated(keep='first')].index

Output []:
Int64Index([], dtype='int64')
```

Με την παραπάνω εντολή εμφανίζονται σε αντικείμενο τύπου Int64Index όλοι οι ακέραιοι αριθμοί οι οποίοι αντιστοιχούν σε εκείνες τις γραμμές του συνόλου δεδομένων data (από το 0 μέχρι το 8568), όπου εμφανίζονται εγγραφές όμοιες με κάποια προηγούμενη, η οποία

θεωρείται ως η πρωτότυπη καταγραφή του συνόλου δεδομένων data, αλλά στην περίπτωσή μας είναι και πάλι κενό καθώς δεν υπάρχουν πολλαπλές όμοιες εγγραφές.

2.2.8 Έλεγχος Αρνητικών ή Μηδενικών Τιμών

Εφόσον το σύνολο των δεδομένων που έχουν συλλεχθεί αφορά σε μεγέθη κίνησης, ταχύτητας και επιτάχυνσης, τότε όλες οι καταγεγραμμένες μετρήσεις πρέπει να είναι θετικές, επομένως ένας έλεγχος που θα ήταν καλό να γίνει αφορά στην επιβεβαίωση ύπαρξης ή όχι αρνητικών ή μηδενικών τιμών.

Παρακάτω ελέγχονται οι ζητούμενες τιμές για κάθε μέγεθος χωριστά:

```
Input []:
1  print(data[data['V-RMS']<=0])
2  print(data[data['a-RMS']<=0])
3  print(data[data['a-Peak']<=0])
4  print(data[data['Crest']<=0])
5  print(data[data['Temperature']<=0])

Output []:
Empty DataFrame
Columns: [V-RMS, a-RMS, a-Peak, Crest, Temperature, Bearing State]
Index: []
Empty DataFrame
Columns: [V-RMS, a-RMS, a-Peak, Crest, Temperature, Bearing State]
Index: []
Empty DataFrame
Columns: [V-RMS, a-RMS, a-Peak, Crest, Temperature, Bearing State]
Index: []
Empty DataFrame
Columns: [V-RMS, a-RMS, a-Peak, Crest, Temperature, Bearing State]
Index: []
Empty DataFrame
Columns: [V-RMS, a-RMS, a-Peak, Crest, Temperature, Bearing State]
Index: []
```

Τα εξαγόμενα αποτελέσματα είναι και πάλι κενά αντικείμενα DataFrame, απλώς λόγω του ότι χρησιμοποιείται η συνάρτηση print() για λόγους συντομίας, τα εξαγόμενα έχουν διαφορετική μορφή από αυτή του κενού πίνακα που εμφανιζόταν σε προηγούμενα παραδείγματα.

Ομοίως, αν θέλουμε να δούμε σε ποιες θέσεις βρίσκονται εκείνα τα στοιχεία που έχουν σε κάποιες από τις στήλες τους μη-θετικές τιμές, τότε εκτελούμε τα παρακάτω:

```
Input []:
1  print(data[data['V-RMS']<=0].index)
2  print(data[data['a-RMS']<=0].index)
3  print(data[data['a-Peak']<=0].index)
4  print(data[data['Crest']<=0].index)
5  print(data[data['Temperature']<=0].index)

Output []:
Int64Index([], dtype='int64')
Int64Index([], dtype='int64')
Int64Index([], dtype='int64')
Int64Index([], dtype='int64')
Int64Index([], dtype='int64')
```

Όπως και τα κενά αντικείμενα DataFrame, έτσι κι εδώ έχουμε κενά αντικείμενα Int64Index, καθώς τα μεγέθη του συνόλου δεδομένων data αποτελούνται μόνο από θετικές τιμές.

ΕΠΙΠΛΕΟΝ ΠΑΡΑΔΕΙΓΜΑΤΑ

Για λόγους σύγκρισης και επίδειξης των παραπάνω αποτελεσμάτων, έστω ο παρακάτω έλεγχος στην περίπτωση όπου εξετάζονται τιμές V-RMS κάτω από $0.0007 m/s$:

```
Input []:
1  data[data['V-RMS']<=0.0007]

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
3296	0.000695	0.609231	3.911539	6.477692	25.938461	2
3300	0.000699	0.602308	3.706154	6.118462	25.900000	2
3302	0.000700	0.613846	3.800769	6.173846	25.900000	2
3309	0.000699	0.576923	3.474615	6.029231	25.900000	2
3312	0.000697	0.556429	3.722857	6.729286	25.900000	2
...	...	...	...	...	...	...
7389	0.000691	0.823077	7.055385	8.565385	26.400000	3
7534	0.000698	0.750769	5.903846	7.758462	26.691539	3
7643	0.000697	0.759231	6.028462	7.965385	27.003077	3
8013	0.000698	0.800769	6.601538	8.250769	26.299999	3
8014	0.000698	0.803846	6.703846	8.296923	26.283846	3

919 rows × 6 columns

Και:

```
Input []:
1  data[data['V-RMS']<=0.0007].index

Output []:
Int64Index([3296, 3300, 3302, 3309, 3312, 3313, 3314, 3315, 3316, 3317,
            ...
            7353, 7356, 7364, 7386, 7387, 7389, 7534, 7643, 8013, 8014],
           dtype='int64', length=919)
```

Επομένως, το αντικείμενο DataFrame έχει συγκρατήσει μόνο εκείνα τα στοιχεία-γραμμές όπου το V-RMS έχει τιμή μικρότερη ή ίση του 0.0007, ενώ το αντικείμενο Int64Index αποτελείται μόνο από τους ακέραιους αριθμούς των αντίστοιχων γραμμών όπως παρουσιάζονται στο αριστερό ευρετήριο του πίνακα δεδομένων data.

Επιπλέον έλεγχος με πιο σύνθετες συνθήκες για τιμές του μεγέθους Crest πάνω από 7 με συμπερίληψη του άκρου, και κάτω από 8 χωρίς συμπερίληψη του άκρου:

```
Input []:
1  data[(data['Crest']>=7) & (data['Crest']<8)]

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
3315	0.000700	0.583846	4.080000	7.047692	25.734616	2
3963	0.000670	0.714615	5.090769	7.070000	26.900000	2
3973	0.000674	0.696154	4.899231	7.052308	26.900000	2
3974	0.000684	0.706154	4.983077	7.056154	26.900000	2
4170	0.000705	0.660000	5.080000	7.543077	26.200001	2
...	...	...	...	...	...	...
8560	0.000715	0.842857	6.382857	7.541429	26.799999	3
8561	0.000733	0.876923	6.996154	7.993846	26.799999	3
8563	0.000726	0.855385	6.603077	7.735385	26.845384	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3

1863 rows × 6 columns

Και:

```
Input []:
1  data[(data['Crest']>=7) & (data['Crest']<8)].index

Output []:
Int64Index([3315, 3963, 3973, 3974, 4170, 4171, 4177, 4178, 4180, 4181,
            ...
            8548, 8551, 8553, 8554, 8556, 8560, 8561, 8563, 8566, 8567],
           dtype='int64', length=1863)
```

Επομένως, μέσω της βιβλιοθήκης Pandas είναι δυνατή η επιλογή συγκεκριμένων τιμών για κάθε στήλη ενός συνόλου δεδομένων, μέσω κατάλληλων συνθηκών και ανάλογα το πρόβλημα που καλείται να επιλυθεί. Προφανώς, η βιβλιοθήκη Pandas προσφέρει πληθώρα συναρτήσεων, εντολών και εφαρμογών και υπάρχουν παραπάνω από ένας τρόποι για να ληφθεί κάποιο συμπέρασμα ή για να εξαχθεί κάποιο αποτέλεσμα.

2.2.9 Επίδειξη Διαγραφής Τιμών

Οι περιπτώσεις διπλότυπων εγγραφών και αρνητικών ή μηδενικών τιμών, οι οποίες πολύ πιθανόν να οφείλονται σε σφάλματα κατά τη συλλογή δεδομένων, μπορούν να

διαταράξουν κατά πολύ τα αποτελέσματα, επομένως είναι καλό να απομακρυνθούν από το σύνολο δεδομένων data το οποίο αναλύουμε.

Εφόσον έχει γίνει κατανοητός ο τρόπος εξαγωγής συγκεκριμένων στοιχείων-γραμμών μέσω του ευρετηρίου και των αντικειμένων Int64Index, ο ευκολότερος τρόπος για να διαγραφούν τα ζητούμενα στοιχεία είναι ο παρακάτω:

```
Input []:
1  data.drop(index=data[…].index, axis='index', inplace=True)
```

Με την εκτέλεση της παραπάνω εντολής διαγράφονται μόνιμα από το σύνολο δεδομένων data εκείνα τα στοιχεία-γραμμές, όπως ορίζονται ως index στη συνάρτηση drop() από τον χρήστη, βάσει εκείνων των δεδομένων που θέλει να διαγράψει.

Η παραπάνω αποτελεί εκτελέσιμη εντολή χωρίς κάποιο άμεσο Output. Ωστόσο, για να εμφανιστεί ο νέος πίνακας δεδομένων data αρκεί η απλή εκτέλεση της μεταβλητής αυτής, εφόσον πλέον έχει αντικατασταθεί πλήρως από το μειωμένο αντικείμενο DataFrame. Αν στην παραπάνω συνάρτηση drop() είχε συμπληρωθεί inplace=False, τότε η εκτέλεσης της εντολής παράγει ως άμεσο εξαγόμενο αποτέλεσμα το προσωρινό αντικείμενο DataFrame το οποίο θα πάρει τη θέση του συνόλου δεδομένων data. Συνήθως, συνιστάται ο προσωρινός έλεγχος προτού γίνει κάποια μόνιμη αλλαγή, όχι μόνο στην περίπτωση διαγραφής στοιχείων, αλλά και σε κάθε τροποποίηση των δεδομένων που αναφέρθηκε.

2.2.10 Ορισμός Νέου Ευρετηρίου

Μετά από κάθε διαγραφή, ή σε περιπτώσεις διαφόρων άλλων τροποποιήσεων, το νέο σύνολο δεδομένων που προκύπτει αποτελείται από ένα ελλιπές αντικείμενο ευρετηρίου, καθώς έχουν αφαιρεθεί γραμμές αφήνοντας αρκετές κενές ενδιάμεσες θέσεις. Κάτι τέτοιο μπορεί να δημιουργήσει προβλήματα κατά την ανάλυση των δεδομένων, επομένως αποτελεί καλή τακτική η ενημέρωση του ευρετηρίου με τον παρακάτω τρόπο:

```
Input []:
1  data.reset_index(drop=True, inplace=False)
```

Το νέο σύνολο δεδομένων data έχει πλέον ως στοιχείο ευρετηρίου διαδοχικές τιμές σε αύξουσα σειρά χωρίς ελλιπές θέσεις, αλλά στο παράδειγμα αυτό η παραπάνω εντολή δεν τροποποιεί τα αρχικά δεδομένα, καθώς δεν υπήρχε λόγος κάποιας διαγραφής εξαρχής.

Αλλά για να γίνει κατανοητή η παραπάνω διαφορά, έστω ο πίνακας δεδομένων όπως προέκυψε από το φιλτράρισμα του μεγέθους Crest για τιμές πάνω από 7 με συμπερίληψη του άκρου, και κάτω από 8 χωρίς συμπερίληψη του άκρου:

```
Input []:
1  dfsort = data[(data['Crest']>=7) & (data['Crest']<8)]
2  dfsort

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
3315	0.000700	0.583846	4.080000	7.047692	25.734616	2
3963	0.000670	0.714615	5.090769	7.070000	26.900000	2
3973	0.000674	0.696154	4.899231	7.052308	26.900000	2
3974	0.000684	0.706154	4.983077	7.056154	26.900000	2
4170	0.000705	0.660000	5.080000	7.543077	26.200001	2
...	...	...	...	...	...	...
8560	0.000715	0.842857	6.382857	7.541429	26.799999	3
8561	0.000733	0.876923	6.996154	7.993846	26.799999	3
8563	0.000726	0.855385	6.603077	7.735385	26.845384	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3

1863 rows × 6 columns

Αυτή τη φορά, αποθηκεύσαμε τον εξαγόμενο πίνακα δεδομένων τύπου DataFrame στη μεταβλητή dfsort, και επειδή η πρώτη εντολή κάνει απλή ανάθεση στη μεταβλητή, χρειάστηκε να κληθεί η δεύτερη εντολή ξεχωριστά.

Παρατηρούμε ότι τα στοιχεία του πίνακα είναι 1863, αλλά αριστερά η στήλη του ευρετηρίου έχει συγκρατήσει τις θέσεις των στοιχείων όπως παρουσιάζονται στον αρχικό πίνακα δεδομένων data, επομένως, η ενημέρωση της σειράς γίνεται ως εξής:

```
Input []:
1  dfsort.reset_index(drop=True, inplace=True)
2  dfsort

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.000700	0.583846	4.080000	7.047692	25.734616	2
1	0.000670	0.714615	5.090769	7.070000	26.900000	2
2	0.000674	0.696154	4.899231	7.052308	26.900000	2
3	0.000684	0.706154	4.983077	7.056154	26.900000	2
4	0.000705	0.660000	5.080000	7.543077	26.200001	2
...	...	...	...	...	...	...
1858	0.000715	0.842857	6.382857	7.541429	26.799999	3
1859	0.000733	0.876923	6.996154	7.993846	26.799999	3
1860	0.000726	0.855385	6.603077	7.735385	26.845384	3
1861	0.000705	0.834615	6.556923	7.822308	26.892307	3
1862	0.000713	0.862308	6.893846	7.989231	26.799999	3

1863 rows × 6 columns

Η πρώτη γραμμή εντολών, εφόσον υπάρχει η δήλωση inplace=True, κάνει απευθείας αντικατάσταση χωρίς κάποιο άμεσο εξαγόμενο αποτέλεσμα, επομένως χρησιμοποιείται στη συνέχεια και πάλι το όνομα της μεταβλητής dfsort στην οποία είναι αποθηκευμένος ο πίνακας δεδομένων τύπου DataFrame που τροποποιήθηκε. Συνεπώς, παρατηρούμε τη νέα σειρά δεδομένων με τη σωστή διαδοχική αύξουσα αρίθμηση, ξεκινώντας και πάλι από το μηδέν.

Ένας επιπλέον καλός έλεγχος για τη στήλη του ευρετηρίου είναι ο παρακάτω, όπου ελέγχεται η μοναδικότητα κάθε στοιχείου, αλλά και η κατάσταση της ταξινόμησης:

```
Input []:
1  print(dfsort.index.is_unique)
2  print(dfsort.index.is_monotonic_increasing)

Output []:
True
True
```

2.2.11 Ομαδοποίηση Βάσει Κατηγορίας

Εφόσον στο συγκεκριμένο παράδειγμα ο πίνακας δεδομένων data αποτελείται από τρεις διαφορετικές διακριτές καταστάσεις, συνιστάται, για την εξαγωγή πιο αντιπροσωπευτικών πληροφοριών και πιο κατανοητής ανάλυσης, να χωριστούν οι επιμέρους περιπτώσεις, και

να ομαδοποιηθούν τα στοιχεία σύμφωνα με τη λειτουργική κατάσταση του ρουλεμάν κατά την οποία καταγράφηκαν οι αντίστοιχες μετρήσεις. Η παραπάνω ομαδοποίηση που περιγράφηκε γίνεται με τον εξής τρόπο:

```
Input []:
1  data_grouped = data.groupby('Bearing State')
2  data_grouped

Output []:
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7d329a985ff0>
```

Το σύνολο δεδομένων data χωρίστηκε σύμφωνα με τη συνάρτηση groupby() σε κατηγορίες βάσει του χαρακτηριστικού Bearing State, το οποίο αποτελείται από τρεις διακριτές τιμές, τις 1, 2 και 3. Η συγκεκριμένη ομαδοποίηση αποθηκεύεται στη μεταβλητή data_grouped, η οποία κατά την κλήση της εξάγεται ως αντικείμενο τύπου DataFrameGroupBy της βιβλιοθήκης Pandas. Η κλήση του κάθε επιμέρους ομαδοποιημένου πίνακα δεδομένων που έχει δημιουργηθεί γίνεται με τον παρακάτω τρόπο:

```
Input []:
1  state_1 = data_grouped.get_group(1)
2  state_2 = data_grouped.get_group(2)
3  state_3 = data_grouped.get_group(3)
```

Συνεπώς, με τη συνάρτηση get_group() και τον αναγνωριστικό ακέραιο της στήλης Bearing State εξάγονται οι επιμέρους πίνακες δεδομένων του στοιχείου data_grouped. Παραπάνω έγινε η ανάθεση του κάθε πίνακα τύπου DataFrame σε ξεχωριστές μεταβλητές, και καλώντας κάποια από αυτές εμφανίζεται ο αντίστοιχος πίνακας δεδομένων:

```
Input []:
1  state_1

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	1
...	...	...	...	...	...	...
3291	0.001048	0.700000	3.230769	4.773846	27.130000	1
3292	0.001054	0.697692	3.243077	4.803846	27.200001	1
3293	0.001067	0.698462	3.220000	4.746923	27.262307	1
3294	0.001035	0.687692	3.146154	4.751538	27.299999	1
3295	0.001042	0.687692	3.196154	4.800769	27.304615	1

3296 rows × 6 columns

Παραπάνω φαίνονται μόνο τα 3296 στοιχεία του πίνακα δεδομένων data όπως έχουν ομαδοποιηθεί για τη δημιουργία του πίνακα δεδομένων state_1, ο οποίος αποτελείται μόνο από τις μετρήσεις που καταγράφηκαν κατά τη λειτουργία του ρουλεμάν με τη συνιστώμενη ποσότητα εργοστασιακού λιπαντικού. Αναλόγως καλούνται και οι πίνακες state_2 και state_3, οι οποίοι παρουσιάζονται αναλυτικά στο Παράρτημα Α', και πρέπει να σημειωθεί ότι τα ευρετήριά τους έχουν την ίδια αρίθμηση όπως και στον αρχικό πίνακα δεδομένων, δηλαδή στον πίνακα state_2 η αρίθμηση ξεκινάει από το στοιχείο 3296, ενώ στον πίνακα state_3 η αρίθμηση ξεκινάει από το στοιχείο 5932.

2.2.12 Μέτρα Θέσης και Διασποράς

Μια γρήγορη εποπτεία των βασικών μέτρων θέσης και διασποράς για κάθε πίνακα δεδομένων τύπου DataFrame μπορούν να εξαχθούν με την κλήση της συνάρτησης describe() για τον εκάστοτε πίνακα. Προφανώς, η κλήση της εντολής για τον συνολικό πίνακα δεδομένων data δεν θα έχει κάποιο ουσιαστικό νόημα, διότι ο υπολογισμών των μέτρων θα γίνει λαμβάνοντας υπόψη όλες τις τιμές, για όλα τα στοιχεία και των τριών διαφορετικών πειραμάτων, κάτι το οποίο δεν θα οδηγήσει σε αντιπροσωπευτικά αποτελέσματα. Επομένως, εξάγονται ξεχωριστά με τον παρακάτω τρόπο:

```
Input []:
```

```
1  state_1.describe()
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
count	3296.000000	3296.000000	3296.000000	3296.000000	3296.000000	3296.0
mean	0.001049	0.686683	3.212853	4.756187	26.054698	1.0
std	0.000038	0.024753	0.060114	0.087172	0.816874	0.0
min	0.000962	0.600000	2.996923	4.438462	24.230770	1.0
25%	0.001010	0.684615	3.171264	4.697692	25.496731	1.0
50%	0.001049	0.698462	3.210769	4.755934	25.857692	1.0
75%	0.001086	0.700000	3.253077	4.810769	26.777692	1.0
max	0.001193	0.799231	3.477692	5.183846	27.799999	1.0

Επομένως, για την περίπτωση 1 του ρουλεμάν, κατά την οποία λειτουργεί στην κανονική του κατάσταση με τη συνιστώμενη ποσότητα εργοστασιακού λιπαντικού, εξάγονται το πλήθος (count) των στοιχείων, ο μέσος όρος (mean), η τυπική απόκλιση (std), η ελάχιστη τιμή (min), το άνω φράγμα του 25% των εγγραφών, το άνω φράγμα του 50% των εγγραφών, το οποίο δηλώνει και τη διάμεσο (median), το άνω φράγμα του 75% των εγγραφών, και η μέγιστη τιμή (max) για καθεμία στήλη του πίνακα δεδομένων state_1. Αναλόγως εξάγονται και τα μέτρα θέσης και διασποράς για τους πίνακες state_2 και state_3, οι οποίοι παρουσιάζονται αναλυτικά στο Παράρτημα Β'.

Επιπλέον, άλλη μια οπτική των παραπάνω περιγραφικών μέτρων είναι και η παρακάτω:

Input []:

```
1  data_grouped['V-RMS'].describe()
```

Output []:

Bearing State	count	mean	std	min	25%	50%	75%	max
1	3296.0	0.001049	0.000038	0.000962	0.001010	0.001049	0.001086	0.001193
2	2636.0	0.000732	0.000030	0.000642	0.000708	0.000733	0.000755	0.000834
3	2637.0	0.000727	0.000032	0.000635	0.000705	0.000725	0.000748	0.000842

Παραπάνω εξάγονται τα βασικά μέτρα θέσης και διασποράς για κάθε επιμέρους πίνακα δεδομένων ο οποίος περιγράφει μια ξεχωριστή λειτουργική κατάσταση για το ρουλεμάν, αλλά μόνο για το μέγεθος V-RMS. Με ανάλογο τρόπο μπορούν να εξαχθούν οι αντίστοιχοι πίνακες και για τα υπόλοιπα μεγέθη a-RMS, a-Peak, Crest και Temperature, οι οποίοι παρουσιάζονται αναλυτικά στο Παράρτημα Β'.

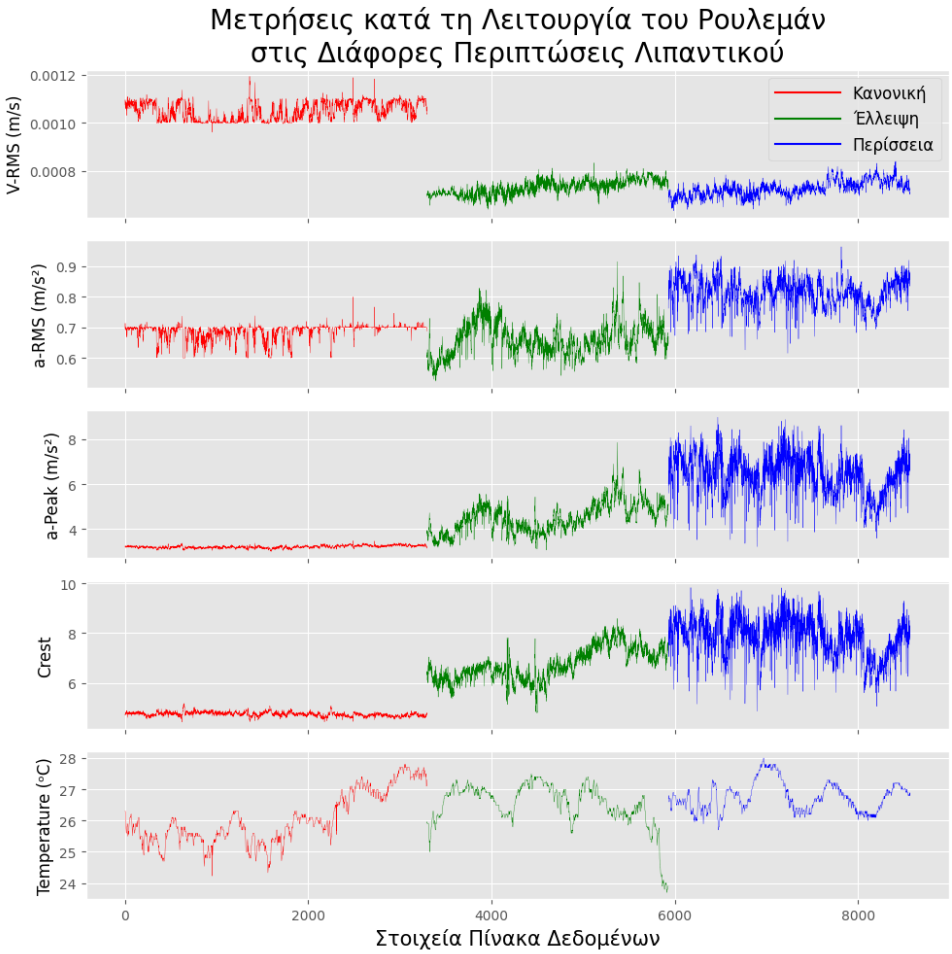
Τελικά, από τη μελέτη των παραπάνω περιγραφικών μέτρων, αλλά και της μεταξύ τους σύγκρισης, μπορούν να προκύψουν νέες πληροφορίες για τη φύση των δεδομένων, αλλά και να διαπιστωθούν τυχόν ανισορροπίες, όπως για παράδειγμα, στην περίπτωση ύπαρξης μη-θετικών τιμών, θα φαινόταν η αντίστοιχη προβληματική τιμή από την ελάχιστη τιμή των πινάκων.

2.2.13 Γραφικές Παραστάσεις

Παρακάτω δίνονται οι γραφικές παραστάσεις των δεδομένων για κάθε λειτουργική κατάσταση του ρουλεμάν ως προς τις συνθήκες λίπανσης. Για την εξαγωγή των διαγραμμάτων χρησιμοποιήθηκε το module της Python βιβλιοθήκης Matplotlib, όπως έχει οριστεί ως plt κατά την εισαγωγή των πακέτων, και τα αντίστοιχα κομμάτια κώδικα για κάθε γραφική παράσταση βρίσκονται στο Παράρτημα Δ’.

ΓΡΑΦΗΜΑ ΓΡΑΜΜΗΣ

Τα παρακάτω γραφήματα γραμμών (line plots) περιγράφουν τις χρονοσειρές που σχηματίζονται από τις μετρήσεις που συλλέχθηκαν κατά τη λειτουργία του ρουλεμάν στις τρεις διαφορετικές συνθήκες λίπανσης.

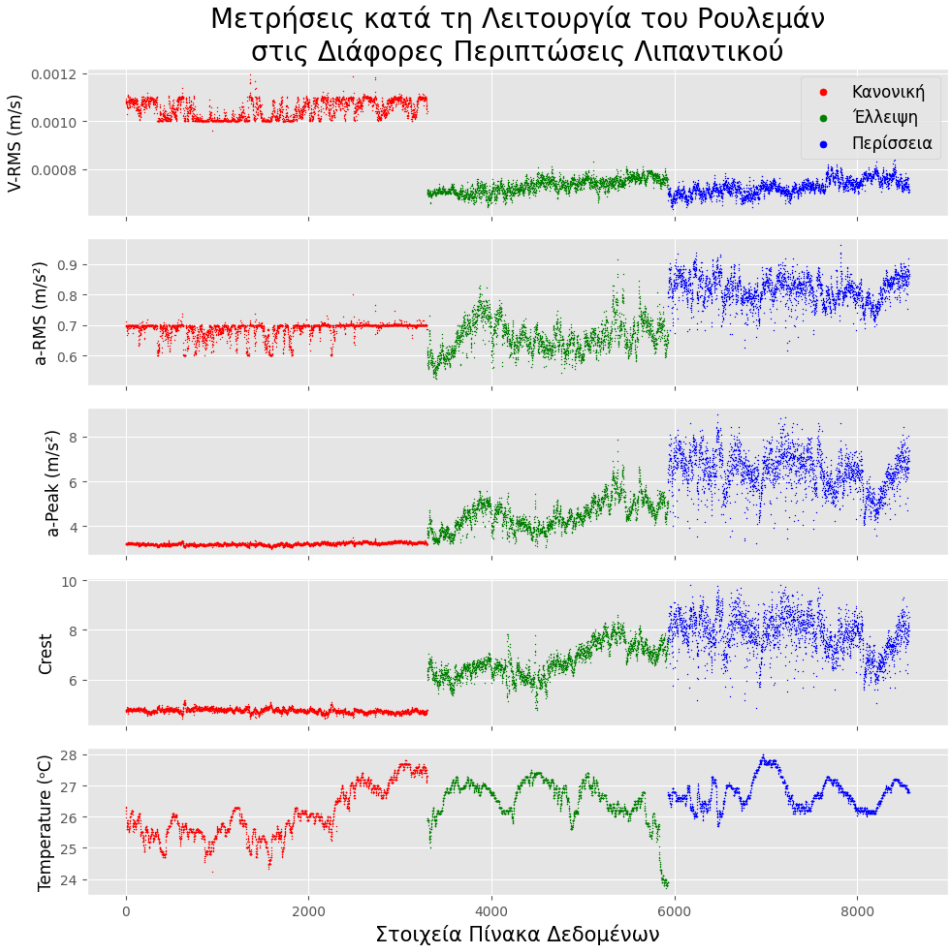


Εικόνα 51. Γραφήματα Γραμμής Καταστάσεων Ρουλεμάν Πρώτης Εφαρμογής

Όσον αφορά στην περίπτωση του V-RMS, τυπικά, μεγάλες τιμές ταχύτητας υποδηλώνουν πιο επιβαρυμένες καταστάσεις λειτουργίας. Στην περίπτωση του παραδείγματος, αν και οι καταστάσεις έλλειψης και περίσσειας λιπαντικού θεωρούνται επιβλαβής, οι τιμές της ταχύτητας είναι χαμηλότερες από αυτές που καταγράφηκαν κατά την κανονική κατάσταση, λόγω του ότι δεν ασκήθηκε κάποιο βάρος κατά τη χρονική περίοδο των μετρήσεων στο ρουλεμάν. Ωστόσο, παρατηρούμε έντονες αναταράξεις στις περιπτώσεις των επιταχύνσεων a-RMS και a-Peak κατά τις καταστάσεις έλλειψης και περίσσειας ποσότητας εργοστασιακού λιπαντικού, σε σχέση με τη λειτουργία υπό κανονικές συνθήκες. Παράλληλα, το μέγεθος Crest, το οποίο είναι και το πιο αντιπροσωπευτικό που θα μπορούσαμε να μελετήσουμε, παίρνει σαφώς υψηλότερες τιμές κατά τις επιβλαβείς περιπτώσεις. Η θερμοκρασία παρατηρείται να παραμένει σχετικά σταθερή και σε χαμηλά επίπεδα, χωρίς κάποια ιδιαίτερη διαφορά μεταξύ των πειραμάτων.

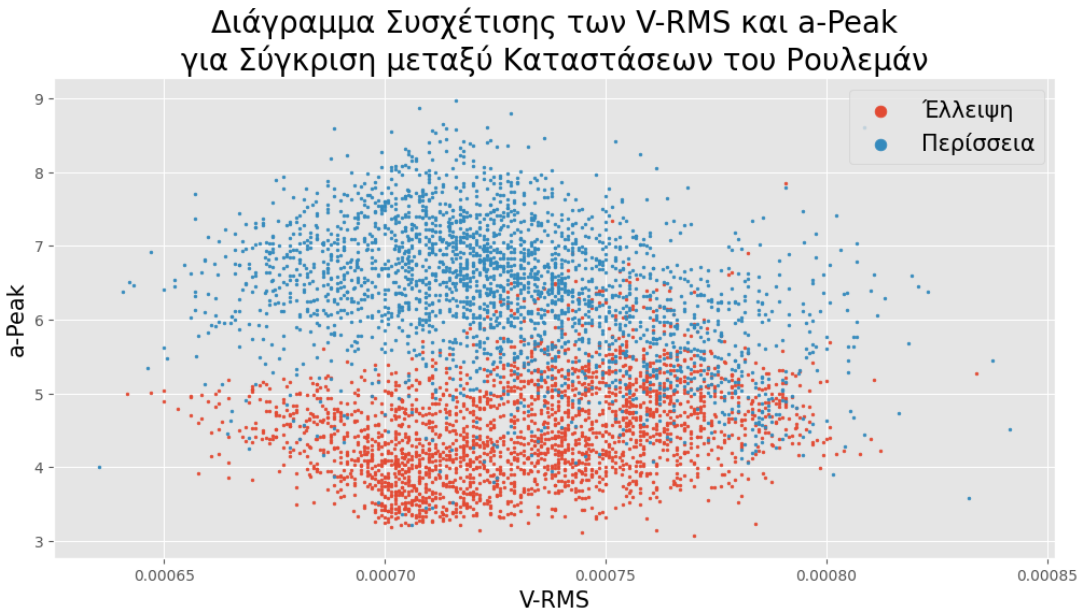
ΓΡΑΦΗΜΑ ΔΙΑΣΠΟΡΑΣ

Τα εξαγόμενα γραφήματα διασποράς (scatter plots) για τις τρεις διαφορετικές περιπτώσεις μετρήσεων παρουσιάζουν ακριβώς τις ίδιες χρονοσειρές με τα παραπάνω γραφήματα γραμμών, ωστόσο επιτρέπουν στον χρήστη την ακριβή γνώση της θέσης κάθε στοιχείου σε σχέση με τα γειτονικά του, σε αντίθεση με τη συνεχή γραμμή των παραπάνω.



Εικόνα 52. Γραφήματα Διασποράς Καταστάσεων Ρουλεμάν Πρώτης Εφαρμογής

Επιπλέον, τα διαγράμματα διασποράς είναι ιδιαίτερα χρήσιμα για τη συσχέτιση μεταξύ χαρακτηριστικών, αλλά και τη σύγκριση διαφορετικών καταστάσεων. Για παράδειγμα, παρακάτω φαίνεται η συσχέτιση μεταξύ των τιμών των μεγεθών V-RMS και a-Peak, τόσο για την κατάσταση έλλειψης λιπαντικού, όσο και για την κατάσταση υπερβολικής ποσότητας.



Εικόνα 53. Γράφημα Διασποράς V-RMS και a-Peak Καταστάσεων Βλάβης Ρουλεμάν Πρώτης Εφαρμογής

Προφανώς, μπορούν να αναλυθούν όλα τα δυνατά ζεύγη μεγεθών για όλες τις δυνατές συγκρίσεις μεταξύ των διαφορετικών καταστάσεων του ρουλεμάν.

Στο επόμενο υποκεφάλαιο παρουσιάζονται συνοπτικά όλα τα σχετικά διαγράμματα διασποράς συσχετίσεων, αλλά ο παραπάνω τρόπος επιτρέπει μια καλύτερη κατανόηση της εξάρτησης μεταξύ των τιμών διαφορετικών καταστάσεων λίπανσης.

ΘΗΚΟΓΡΑΜΜΑ

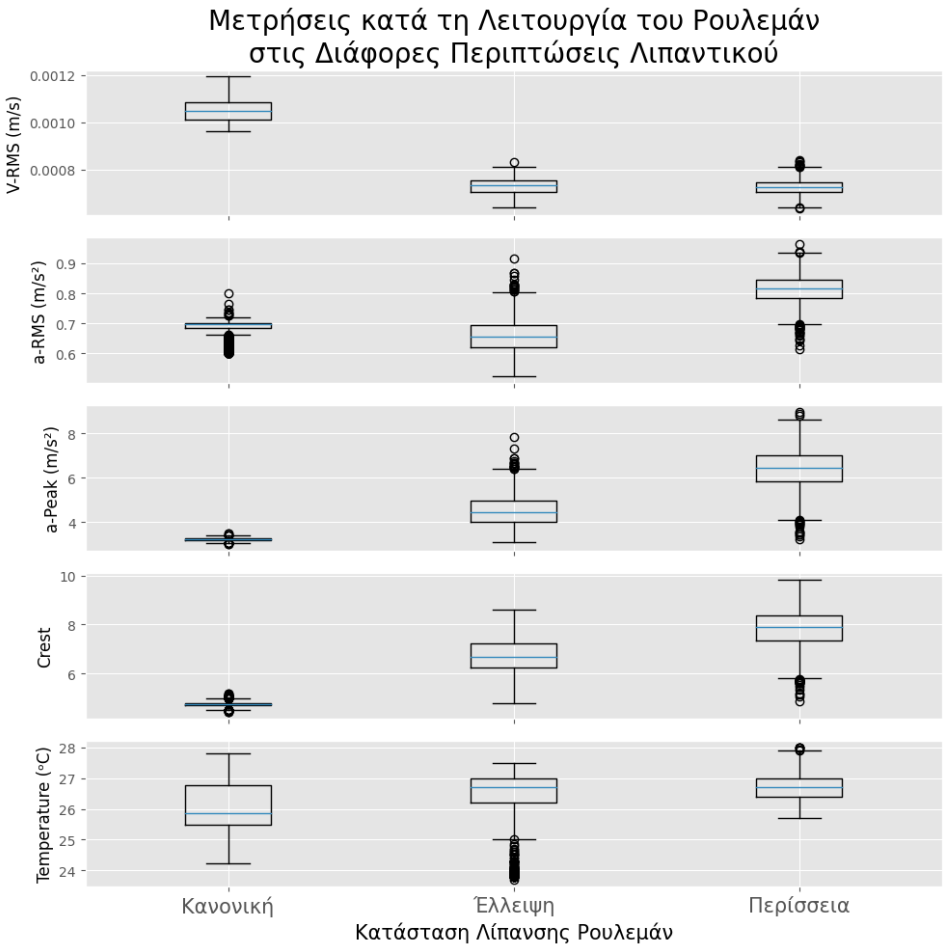
Τα θηκογράμματα (box plots) αποτελούν σημαντικό τύπο διαγράμματος, καθώς προσφέρουν μια καλύτερη εποπτεία για τη διασπορά των τιμών κάθε μεγέθους.

Συγκεκριμένα, η μεσαία γραμμή ενός θηκογράμματος δηλώνει την τιμή της διαμέσου, και τα άνω και κάτω άκρα της «θήκης» (box) που σχηματίζεται δηλώνουν αντίστοιχα εκείνες τις τιμές των δεδομένων που έχουν τον ρόλο του άνω φράγματος για το 25% των παρατηρήσεων και το 75% των παρατηρήσεων αντίστοιχα. Αυτό σημαίνει ότι η διάμεσος είναι εκείνη η τιμή που έχει τον ρόλο του άνω φράγματος για το 50% των στοιχείων, όπως αναλύθηκε και κατά τον υπολογισμό των μέτρων θέσης και διασποράς. Επιπλέον, οι τιμές-φράγματα των 25%, 50% και 75% ονομάζονται και αλλιώς τεταρτημόρια και συμβολίζονται Q1, Q2 και Q3 αντίστοιχα (από το αγγλικό quartile).

Επίσης, τα τελικά όρια ενός θηκογράμματος είναι δυο οριζόντιες γραμμές, πέρα των ορίων της «θήκης», και σε αυτήν την περίπτωση υπολογίζονται ως $Q1 - 1.5 * (Q3 - Q1)$ και

$Q3 + 1.5 * (Q3 - Q1)$, όπου η διαφορά των τεταρτημορίων $Q3 - Q1$ ονομάζεται αλλιώς ενδοτεταρτημοριακό εύρος (interquartile range - IQR). Ο σχεδιασμός ενός θηκογράμματος μέσω της βιβλιοθήκης Matplotlib είναι παραμετροποιήσιμος και ποικίλει ανάλογα με τα δεδομένα που εξετάζονται.

Τα θηκογράμματα του παραδείγματος που μελετάμε, για όλες τις καταστάσεις του ρουλεμάν, και για όλα τα χαρακτηριστικά του συνόλου δεδομένων φαίνονται παρακάτω:



Εικόνα 54. Θηκογράμματα Καταστάσεων Ρουλεμάν Πρώτης Εφαρμογής

Παρατηρούμε ότι υπάρχουν τιμές εκτός των ορίων του θηκογράμματος, κάτι που γενικότερα παραπέμπει σε ακραίες τιμές. Ωστόσο, οι παραπάνω περιπτώσεις δεν αποτελούν ούτε μεμονωμένες καταστάσεις, ούτε αποκλίνουν κατά πολύ από τα επιλεγμένα όρια των γραφημάτων.

2.2.14 Συσχετίσεις Μεγεθών

Ενδιαφέρον παρουσιάζουν και τυχόν εξαρτήσεις μεταξύ των μεγεθών, καθώς προσφέρουν περαιτέρω πληροφορίες για τη σχέση μεταξύ των δεδομένων. Συγκεκριμένα, υπολογίζονται παρακάτω οι Συντελεστές Συσχέτισης Πίρσον (Pearson Correlation Coefficient) για όλα τα ζεύγη στηλών της κανονικής κατάστασης λίπανσης του ρουλεμάν, και προφανώς με αντίστοιχο τρόπο υπολογίζονται και για τις υπόλοιπες καταστάσεις, οι οποίες παρουσιάζονται αναλυτικά στο Παράρτημα Ε’.

```
Input []:
1  state_1_corr = state_1.corr(method='pearson')
2  state_1_corr
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
V-RMS	1.000000	0.595561	0.704940	-0.693446	0.429080	NaN
a-RMS	0.595561	1.000000	0.695501	-0.716310	0.423715	NaN
a-Peak	0.704940	0.695501	1.000000	-0.538803	0.633294	NaN
Crest	-0.693446	-0.716310	-0.538803	1.000000	-0.391334	NaN
Temperature	0.429080	0.423715	0.633294	-0.391334	1.000000	NaN
Bearing State	NaN	NaN	NaN	NaN	NaN	NaN

Ο συντελεστής συσχέτισης Πίρσον αναγνωρίζει κυρίως γραμμικές εξαρτήσεις μεταξύ των δεδομένων, και η τιμή κάθε συντελεστή κυμαίνεται από -1 μέχρι και 1. Συνεπώς, οι αρνητικές τιμές υποδηλώνουν αρνητική γραμμική συσχέτιση, ενώ αντίστοιχα, οι θετικές τιμές υποδηλώνουν θετική γραμμική συσχέτιση. Όσο πιο κοντά στα δυο άκρα είναι μια τιμή, τόσο πιο ισχυρή είναι η γραμμική εξάρτηση των δυο μεγεθών, και αυτό μεταφράζεται γραφικά με τα διαγράμματα διασποράς όπου τα αντίστοιχα στοιχεία των δυο μεγεθών συγκεντρώνονται σχηματίζοντας ολοένα και πιο ξεκάθαρη ευθεία.

Προφανώς, κάθε μέγεθος είναι πλήρως εξαρτημένο από τον εαυτό του, γι’ αυτό και ο συντελεστής Πίρσον είναι καθαρή θετική μονάδα, η οποία φαίνεται παντού στην κύρια διαγώνιο του παραπάνω πίνακα.

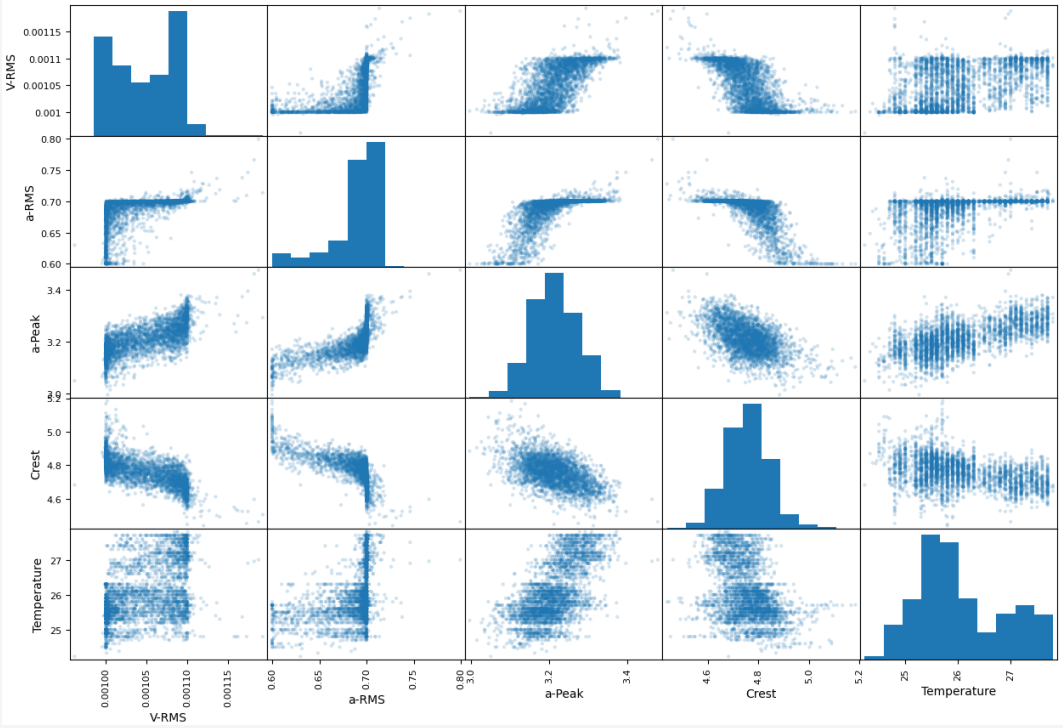
Επιπλέον, η στήλη Bearing State αποτελείται από τη μονάδα σε όλα τα στοιχεία του συνόλου δεδομένων για την κανονική κατάσταση, επομένως είναι αδύνατον, αλλά και δεν έχει ουσιαστικό νόημα, να υπολογιστεί ο συντελεστής συσχέτισης με καθένα από τα υπόλοιπα ζεύγη, και παρουσιάζεται ως NaN στον παραπάνω πίνακα.

Τα διαγράμματα διασποράς για όλα τα ζεύγη μεγεθών της κανονικής κατάστασης του ρουλεμάν, αντίστοιχα εκείνου που παρουσιάστηκε στο προηγούμενο υποκεφάλαιο, μπορούμε να τα εξάγουμε συγκεντρωτικά με τον παρακάτω τρόπο, ενώ οι υπόλοιπες καταστάσεις παρουσιάζονται αναλυτικά στο Παράρτημα Ε’.

Input []:

```
plt.style.use('default')
pd.plotting.scatter_matrix(state_1[['V-RMS', 'a-RMS', 'a-Peak',
                                    'Crest', 'Temperature']],
                           figsize=(15, 10), alpha=0.2,
                           diagonal='hist')
plt.show()
```

Output []:



Μπορούν εύκολα να διακριθούν οι πιο ισχυρές αρνητικές και θετικές συσχετίσεις, οι οποίες επιβεβαιώνουν και τους αντίστοιχους συντελεστές συσχέτισης Πίρσον που υπολογίστηκαν παραπάνω. Για παράδειγμα τα μεγέθη a-RMS και a-Peak έχουν συντελεστή 0.695501 και όντως παρατηρούμε ότι τα στοιχεία συγκεντρώνονται γύρω από μια ευθεία με θετική κλίση, ενώ τα μεγέθη a-RMS και Crest έχουν συντελεστή -0.716310 ο οποίος επιβεβαιώνεται από την αρνητική κλίση της ευθείας που σχηματίζεται.

Επιπλέον, η παραπάνω εντολή παραμετροποιήθηκε έτσι ώστε η κύρια διαγώνιος του εξαγόμενου διαγράμματος να παρουσιάζει τα ιστογράμματα κάθε μεγέθους για καλύτερη κατανόηση της κατανομής τους.

2.3 Πρώτο Μοντέλο Νευρωνικού Δικτύου

Παραπάνω πραγματοποιήθηκε μια βασική στατιστική ανάλυση του πίνακα δεδομένων data και παρουσιάστηκαν κάποιες μέθοδοι για την εξαγωγή χρήσιμων πληροφοριών και τη διαχείριση μη-αντιπροσωπευτικών δεδομένων. Προφανώς, μπορεί να γίνει πιο εκτενής μελέτη, με πιο ενδελεχείς ελέγχους, και η βιβλιοθήκη Pandas προσφέρει πληθώρα συναρτήσεων για τον σκοπό αυτό.

Επομένως, με μια καλύτερη πλέον κατανόηση των διαφορετικών καταστάσεων του ρουλεμάν, αλλά και κάποιων μεταξύ τους διαφοροποιήσεων, μπορούμε να προχωρήσουμε στη διαδικασία ανάπτυξης του μοντέλου Βαθιάς Μάθησης.

Η γενικότερη αρχιτεκτονική των Τεχνητών Νευρωνικών Δικτύων η οποία επιλέχθηκε για την αντιμετώπιση του συγκεκριμένου προβλήματος είναι αυτή του Πολυεπίπεδου Αντιλήπτρου (Multilayer Perceptron - MLP).

2.3.1 Εισαγωγή Πακέτων

Για τις ανάγκες των παρακάτω διαδικασιών εισάγονται και τα επιπλέον πακέτα της Python:

```
Input []:
1  import numpy
2  from tensorflow import keras
3  import sklearn
```

Ο έλεγχος των εκδόσεων που χρησιμοποιούνται γίνεται με τον παρακάτω κώδικα:

```
Input []:
1  print('numpy', numpy.__version__)
2  print('keras', keras.__version__)
3  print('sklearn', sklearn.__version__)

Output []:
numpy 1.22.4
keras 2.12.0
sklearn 1.2.2
```

Για καλύτερη διαχείριση του κώδικα και των αναγκών των παρακάτω διαδικασιών θα γίνει η εισαγωγή των επιπλέον πακέτων και των απαραίτητων module ως εξής:

```
Input []:
```

```
1  import numpy as np
2  from tensorflow import keras
3  from sklearn.model_selection import train_test_split
4  from sklearn.metrics import ConfusionMatrixDisplay,
5                              classification_report
```

2.3.2 Ετικέτες Δεδομένων

Αναφέρθηκε προηγουμένως ότι το συγκεκριμένο πρόβλημα ανάπτυξης και εκπαίδευσης μοντέλου κατατάσσεται στην κατηγορία της Επιβλεπόμενης Μάθησης, καθώς υπάρχουν διαθέσιμες ετικέτες στο σύνολο δεδομένων, οι οποίες περιγράφουν τη φύση κάθε στοιχείου-γραμμής.

Συγκεκριμένα, οι αναγνωριστικοί ακέραιοι 1, 2 και 3 στη στήλη Bearing State του συνόλου δεδομένων data, περιγράφουν αντίστοιχα την κατάσταση του ρουλεμάν της πειραματικής διάταξης κατά τη λειτουργία του με τη συνιστώμενη ποσότητα εργοστασιακού λιπαντικού, με έλλειψη και με περίσσεια αυτής.

Ωστόσο, για την αρχιτεκτονική του μοντέλου που θα εκπαιδευτεί είναι απαραίτητο η παραπάνω αρίθμηση να ξεκινάει από το μηδέν, οπότε, προτού προχωρήσουμε στις παρακάτω διαδικασίες, θα γίνει η αλλαγή τους με τον παρακάτω τρόπο:

```
Input []:
1  data['Bearing State'] = data['Bearing State'] - 1
2  print(data)
3  print(data['Bearing State'].value_counts())

Output []:
         V-RMS     a-RMS    a-Peak     Crest  Temperature  Bearing State
0     0.001074  0.695385  3.198462  4.740000    26.308461              0
1     0.001095  0.713077  3.244615  4.716923    26.299999              0
2     0.001079  0.697143  3.227857  4.795000    26.299999              0
3     0.001080  0.700000  3.213077  4.793077    26.208462              0
4     0.001088  0.700000  3.241538  4.783077    26.200001              0
...        ...       ...       ...       ...          ...            ...
8564  0.000728  0.880769  7.337692  8.304615    26.866153              2
8565  0.000703  0.853846  6.926154  8.102308    26.806153              2
8566  0.000705  0.834615  6.556923  7.822308    26.892307              2
8567  0.000713  0.862308  6.893846  7.989231    26.799999              2
8568  0.000708  0.866154  7.378462  8.506923    26.799999              2

[8569 rows x 6 columns]
0    3296
```

```
2    2637
1    2636
Name: Bearing State, dtype: int64
```

Παρατηρούμε στο αντικείμενο DataFrame την τροποποίηση της τελευταίας στήλης, αλλά και πιο συγκεκριμένα τις μοναδικές της τιμές 0, 1 και 2 με το πλήθος των στοιχείων που αντιστοιχεί στην καθεμία μέσω της συνάρτησης value_counts().

2.3.3 Δεδομένα Εκπαίδευσης

Το μοντέλο που θα αναπτυχθεί θα πρέπει να εκπαιδευτεί στα δεδομένα του προβλήματος, έτσι ώστε όταν τροφοδοτείται με νέα δεδομένα, να μπορεί να αναγνωρίσει με όσο το δυνατόν μεγαλύτερη ακρίβεια, τη σωστή κατάσταση στην οποία ανήκουν.

Ωστόσο, δεν αρκεί μόνο η εκπαίδευση, αλλά και η αξιολόγηση για το αν το συγκεκριμένο μοντέλο έχει εκπαιδευτεί σωστά. Μια εκπαιδευτική διαδικασία θεωρείται επιτυχημένη όταν το αντίστοιχο μοντέλο είναι σε θέση να γενικεύσει και να αποδώσει ικανοποιητικά αποτελέσματα όταν έρχεται αντιμέτωπο με στοιχεία τα οποία βλέπει για πρώτη φορά και δεν τα έχει συναντήσει κατά την εκπαίδευσή του.

Για τον λόγο αυτό, συνηθισμένη τακτική αποτελεί ο διαχωρισμός του αρχικού πίνακα δεδομένων σε επιμέρους σύνολα δεδομένων. Συγκεκριμένα, το πρώτο επιμέρους κομμάτι αναφέρεται ως Δεδομένα Εκπαίδευσης (Training Dataset), το δεύτερο ως Δεδομένα Επιβεβαίωσης (Validation Dataset) και το τρίτο ως Δεδομένα Αξιολόγησης (Testing Dataset). Τα δεδομένα Εκπαίδευσης περιλαμβάνουν τις εγγραφές πάνω στις οποίες εκπαιδεύεται το μοντέλο, τα δεδομένα Επιβεβαίωσης χρησιμοποιούνται κατά τη διάρκεια της εκπαιδευτικής διαδικασίας για τον έλεγχο της προόδου του, χωρίς το μοντέλο να εκπαιδεύεται πάνω σε αυτά. Τα δεδομένα Αξιολόγησης χρησιμοποιούνται για την αξιολόγηση της επίδοσης και της γενίκευσης του τελικού εκπαιδευμένου μοντέλου, έχοντας τον ρόλο των «νέων» στοιχείων, τα οποία βλέπει για πρώτη φορά το μοντέλο.

Για την περίπτωση που εξετάζεται θα γίνει πρώτα η διαίρεση των δεδομένων στα σύνολα Εκπαίδευσης και Αξιολόγησης, σε ένα ποσοστό 80% για το πρώτο και 20% για το δεύτερο ως προς το μέγεθος του αρχικού συνόλου data, και το σύνολο Επιβεβαίωσης θα οριστεί τη στιγμή που θα ξεκινήσει η διαδικασία εκπαίδευσης του μοντέλου.

2.3.4 Διαίρεση Δεδομένων Εκπαίδευσης και Αξιολόγησης

Με τη βοήθεια της βιβλιοθήκης Scikit-learn, και συγκεκριμένα της μεθόδου train_test_split(), όπως έχει γίνει η εισαγωγή της από το module του model_selection, θα γίνει ο διαχωρισμός των ζητούμενων συνόλων δεδομένων ως εξής:

```
Input []:
1  X_train, X_test, y_train, y_test = train_test_split(
2                                     data['Bearing State'],
3                                     data.iloc[:,:-1],
4                                     test_size = 0.2,
5                                     shuffle=True)
```

Με την παραπάνω εντολή όλες οι εγγραφές του συνόλου δεδομένων data ανακατεύονται (shuffle=True) και δημιουργούνται τέσσερα επιμέρους σύνολα δεδομένων.

Συγκεκριμένα, τα X_train και X_test είναι αντικείμενα τύπου DataFrame τα οποία αποτελούνται από τις 5 στήλες του πίνακα data (data.iloc[:,:-1]) οι οποίες περιγράφουν τις μετρήσεις του ρουλεμάν, ενώ τα y_train και y_test είναι αντικείμενα τύπου Series της βιβλιοθήκης Pandas, καθώς αποτελούνται μόνο από μια στήλη η οποία περιλαμβάνει τις αντίστοιχες ετικέτες (data['Bearing State']) των δυο προηγούμενων συνόλων X_train και X_test.

Προφανώς τα σύνολα με κατάληξη _train αποτελούν τα δεδομένα Εκπαίδευσης που θα χρησιμοποιήσει το μοντέλο, ενώ τα σύνολα με κατάληξη _test αποτελούν τα δεδομένα Αξιολόγησης που θα χρησιμοποιηθούν για τον τελικό έλεγχο. Επιπλέον, η δήλωση test_size=0.2 σημαίνει ότι το σύνολο Αξιολόγησης θα περιλαμβάνει το 20% των στοιχείων του συνόλου data.

Ωστόσο, είναι σημαντικό να αναφερθεί η κατανομή των τριών περιπτώσεων λειτουργικής κατάστασης του ρουλεμάν, ως προς το πλήθος των στοιχείων που αντιστοιχούν στην καθεμία. Για τον λόγο αυτόν εκτελούνται οι παρακάτω εντολές:

```
Input []:
1  print(data['Bearing State'].value_counts() / len(data))
2  print(y_train.value_counts() / len(y_train))
3  print(y_test.value_counts() / len(y_test))

Output []:
0    0.384642
2    0.307737
1    0.307620
Name: Bearing State, dtype: float64
0    0.386579
```

```
1    0.310284
2    0.303136
Name: Bearing State, dtype: float64
0    0.376896
2    0.326138
1    0.296966
Name: Bearing State, dtype: float64
```

Τα παραπάνω αποτελέσματα δηλώνουν το ποσοστό που καταλαμβάνει κάθε περίπτωση στα διάφορα σύνολα δεδομένων τα οποία μελετώνται.

Ειδικότερα, 38.4642% των μετρήσεων αφορούν την κανονική κατάσταση λίπανσης, 30.7620% την κατάσταση έλλειψης και 30.7737% την κατάσταση υπερβολικής λίπανσης, έτσι όπως παρουσιάζονται στο πραγματικό σύνολο δεδομένων data.

Τα αντίστοιχα ποσοστά για τα επιμέρους σύνολα δεδομένων _train και _test, τα οποία προέκυψαν σύμφωνα με τον παραπάνω τρόπο της τυχαίας διαίρεσης, είναι 38.6579%, 31.0284%, 30.3136% για τα δεδομένα Εκπαίδευσης, και 37.6896%, 29.6966%, 32.6138% για τα δεδομένα Αξιολόγησης.

Αν και ο διαχωρισμός των συνόλων έγινε με πάρα πολύ μικρή απόκλιση από τις πραγματικές κατανομές των στοιχείων, καλό θα ήταν να διατηρηθούν τα αρχικά ποσοστά και στα επιμέρους σύνολα δεδομένων Εκπαίδευσης και Αξιολόγησης. Αυτό επιτυγχάνεται με τον παρακάτω τρόπο:

```
Input []:
1  X_train, X_test, y_train, y_test = train_test_split(
2                                      data.iloc[:,:-1],
3                                      data['Bearing State'],
4                                      test_size = 0.2,
5                                      shuffle=True,
6                                      stratify=data['Bearing State'])
7  print(data['Bearing State'].value_counts() / len(data))
8  print(y_train.value_counts() / len(y_train))
9  print(y_test.value_counts() / len(y_test))

Output []:
0    0.384642
2    0.307737
1    0.307620
Name: Bearing State, dtype: float64
0    0.384683
2    0.307659
1    0.307659
Name: Bearing State, dtype: float64
```

```
0    0.384481
2    0.308051
1    0.307468
Name: Bearing State, dtype: float64
```

Επομένως, με την επιπλέον δήλωση stratify=data['Bearing State'], ο διαχωρισμός των δεδομένων data στα σύνολα Εκπαίδευσης και Αξιολόγησης ακολουθεί τη στρωματοποίηση και τις ποσοστιαίες κατανομές των αρχικών καταστάσεων, όπως φαίνεται τελικά και με τα εξαγόμενα αποτελέσματα.

Ακόμη, για περαιτέρω επιβεβαίωση, ελέγχουμε τη διάσταση καθενός από τα τέσσερα επιμέρους σύνολα δεδομένων ως εξής:

```
Input []:
1  print(X_train.shape)
2  print(y_train.shape)
3  print(X_test.shape)
4  print(y_test.shape)

Output []:
(6855, 5)
(6855,)
(1714, 5)
(1714,)
```

Συνεπώς, 6855 (80%) στοιχεία των 5 χαρακτηριστικών με τις αντίστοιχες ετικέτες τους θα χρησιμοποιηθούν για την εκπαίδευση του μοντέλου, ενώ 1714 (20%) για την τελική αξιολόγησή του.

2.3.5 Αλλαγή Τύπου Δεδομένων

Αναφέρθηκε ότι τα X_train και X_test είναι τύπου DataFrame, ενώ τα y_train και y_test είναι τύπου Series. Όμως, τα παραπάνω αποτελούν συγκεκριμένα μοναδικά αντικείμενα της βιβλιοθήκης Pandas και δεν μπορούν να τροφοδοτηθούν στο μοντέλο για επεξεργασία σε αυτή τη μορφή.

Συγκεκριμένα, για τα X_train και y_train έχουμε:

```
Input []:
1  print(X_train)
2  print(y_train)

Output []:
```

```
         V-RMS     a-RMS    a-Peak     Crest  Temperature
7214  0.000694  0.801429  6.710000  8.339286    26.774285
7861  0.000740  0.797692  6.125385  7.659231    26.799999
5388  0.000717  0.641538  5.353077  8.276923    26.333846
7189  0.000715  0.851538  7.193846  8.410000    27.100000
8429  0.000735  0.792143  5.780000  7.217143    27.200001
...        ...       ...       ...       ...          ...
2132  0.001015  0.700000  3.201538  4.733846    26.000000
562   0.001070  0.698462  3.171538  4.721538    25.799999
4213  0.000724  0.639231  4.097692  6.363077    26.299999
8374  0.000757  0.842308  6.228462  7.402308    27.009231
13    0.001071  0.691538  3.186923  4.790769    25.600000

[6855 rows x 5 columns]
7214    2
7861    2
5388    1
7189    2
8429    2
       ..
2132    0
562     0
4213    1
8374    2
13      0
Name: Bearing State, Length: 6855, dtype: int64
```

Αξίζει να παρατηρήσουμε παραπάνω και τη σωστή αντιστοιχία των ευρετηρίων των δυο επιμέρους συνόλων δεδομένων.

Με τον παρακάτω τρόπο γίνεται ο μετασχηματισμός των επιμέρους συνόλων δεδομένων σε πίνακες (arrays) μέσω της βιβλιοθήκης Numpy, οι οποίοι θα μπορούν να αναλυθούν από το μοντέλο που θα αναπτυχθεί αργότερα με τη βιβλιοθήκη Keras:

```
Input []:
1  X_train = np.asarray(X_train)
2  X_test = np.asarray(X_test)
3  y_train = np.asarray(y_train)
4  y_test = np.asarray(y_test)
```

Τελικά, η μετασχηματισμένη μορφή των X_train και y_train είναι:

```
Input []:
1  print(X_train)
2  print(y_train)
```

```
Output []:
[[6.93571417e-04 8.01428573e-01 6.71000000e+00 8.33928558e+00
  2.67742853e+01]
 [7.40000002e-04 7.97692299e-01 6.12538466e+00 7.65923071e+00
  2.67999992e+01]
 [7.16923073e-04 6.41538464e-01 5.35307697e+00 8.27692307e+00
  2.63338456e+01]
 ...
 [7.23846149e-04 6.39230774e-01 4.09769232e+00 6.36307694e+00
  2.62999992e+01]
 [7.56923082e-04 8.42307691e-01 6.22846156e+00 7.40230766e+00
  2.70092308e+01]
 [1.07076923e-03 6.91538458e-01 3.18692310e+00 4.79076925e+00
  2.56000004e+01]]
[2 2 1 ... 1 2 0]
```

Τα αντίστοιχα αποτελέσματα για τα σύνολα δεδομένων X_test και y_test πριν και μετά τον μετασχηματισμό τους παρουσιάζονται στο Παράρτημα Ζ'.

ΠΑΡΑΤΗΡΗΣΗ

Μέσα από δοκιμές παρατηρήθηκε ότι ήταν δυνατή η ανάγνωση και διαχείριση των αντικειμένων DataFrame από το μοντέλο για την εκπαίδευσή του. Ωστόσο, θα γίνεται η παραπάνω αλλαγή των στοιχείων σε πίνακες τύπου array για λόγους συνοχής και διαδικασιών που θα χρησιμοποιηθούν παρακάτω.

2.3.6 Αρχικοποίηση του Μοντέλου

Η αρχιτεκτονική του μοντέλου Βαθιάς Μάθησης που επιλέχθηκε είναι αυτή των Τεχνητών Νευρωνικών Δικτύων, και συγκεκριμένα εκείνη του Πολυεπίπεδου Αντιλήπτρου (MLP). Επομένως το μοντέλο θα αποτελείται από ένα επίπεδο εισόδου, ένα ή περισσότερα ενδιάμεσα επίπεδα και ένα επίπεδο εξόδου. Υπενθυμίζεται ότι τα συμβατικά επίπεδα αποτελούνται από ένα πλήθος νευρώνων, οι οποίοι εφαρμόζουν μαθηματικές πράξεις με πολλαπλασιασμό παραμέτρων βαρών και με την ενσωμάτωση παραμέτρων σταθερών όρων, προτού μετασχηματίσουν το τελικό αποτέλεσμα μέσω μια συνάρτησης ενεργοποίησης.

Βάσει αυτών, μέσω της βιβλιοθήκης Tensorflow, και συγκεκριμένα του δημοφιλές module της, Keras, ορίζουμε το παρακάτω μοντέλο MLP με όνομα «model_1»:

```
Input []:
1  model_1 = keras.models.Sequential(
2      name='model_1',
3      layers=[
4      keras.Input(shape=(5,)),
5      keras.layers.Dense(10, activation='relu',
6                         kernel_initializer='he_normal',
7                         bias_initializer='zeros'),
8      keras.layers.Dense(10, activation='relu',
9                         kernel_initializer='he_normal',
10                        bias_initializer='zeros'),
11     keras.layers.Dense(10, activation='relu',
12                        kernel_initializer='he_normal',
13                        bias_initializer='zeros'),
14     keras.layers.Dense(3, activation='softmax',
15                        kernel_initializer='glorot_uniform',
16                        bias_initializer='zeros')
17 ])
```

Να σημειωθεί ότι η συμβατική ονομασία Dense ορίζεται για τα επίπεδα τα οποία εφαρμόζουν τις πράξεις όπως περιγράφηκαν παραπάνω, αλλά και πιο αναλυτικά στο μαθηματικό παράδειγμα του προηγούμενο Κεφαλαίου.

Η παραπάνω αρχιτεκτονική αποτελείται από ένα επίπεδο εισόδου (Input) των 5 νευρώνων, αφού τα δεδομένα εκπαίδευσης, τα οποία πλέον είναι μετασχηματισμένα σε πίνακες, περιλαμβάνουν 5 τιμές για κάθε στοιχείο-γραμμή.

Επίσης, το επίπεδο εξόδου, το οποίο ορίζεται στο τέλος, αποτελείται από 3 νευρώνες, εφόσον οι ετικέτες είναι αποθηκευμένες στα σύνολα δεδομένων y_train και y_test ως τρεις διακριτές τιμές 0, 1 και 2. Ακόμη, για το επίπεδο εξόδου έχει οριστεί ως συνάρτηση ενεργοποίησης η softmax, η οποία εξάγει τρία αποτελέσματα πρόβλεψης σε μορφή πιθανοτήτων, από τις οποίες επιλέγεται η κατάσταση μεγαλύτερης πιθανότητας ως η τελική πρόβλεψη του μοντέλου, βάσει του παρακάτω μαθηματικού τύπου:

$$softmax(x) = \frac{e^{x}}{\sum_{i=1}^{K} e^{x_i}}$$

Όπου:

- x είναι η διανυσματική έξοδος του επιπέδου εξόδου μετά τον μετασχηματισμό του από την ενσωμάτωση των παραμέτρων βαρών και σταθερών όρων από τους νευρώνες του επιπέδου
- x_i είναι καθεμία από τις τιμές του διανύσματος εξόδου x

- K είναι το πλήθος των διαφορετικών κλάσεων (ή αλλιώς των νευρώνων του επιπέδου εξόδου)

Τέλος, το μοντέλο αποτελείται από 3 ενδιάμεσα επίπεδα των 10 νευρώνων, το πλήθος τον οποίων επιλέγεται από τον χρήστη, ανάλογα με το πρόβλημα που καλείται να επιλύσει, και τροποποιείται καταλλήλως. Ακόμη, ως συνάρτηση ενεργοποίησης για καθένα από τα επίπεδα επιλέχτηκε η ReLU (Rectified Linear Unit), η οποία εισάγει μη-γραμμικότητα στα αποτελέσματα των πράξεων για την αναγνώριση πολυπλοκότερων μοτίβων στα δεδομένα, με τον παρακάτω μαθηματικό τύπο:

$$ReLU(x) = \max(0, x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

Όπου:

- x είναι η διανυσματική έξοδος του επιπέδου εξόδου μετά τον μετασχηματισμό του από την ενσωμάτωση των παραμέτρων βαρών και σταθερών όρων από τους νευρώνες του επιπέδου

Ακόμη, η δήλωση kernel_initializer αρχικοποιεί τυχαία τις τιμές των παραμέτρων των βαρών (weights) σύμφωνα με κάποια κατανομή, και στη συγκεκριμένη περίπτωση επιλέχτηκε η κατανομή he_normal για τα ενδιάμεσα επίπεδα, ενώ η glorot_uniform για τα βάρη του επιπέδου εξόδου. Επίσης, οι παράμετροι των σταθερών όρων (biases) αρχικοποιούνται ως μηδέν σε κάθε επίπεδο μέσω της δήλωσης bias_initializer='zeros'. Η τυχαία επιλογή των αρχικών τιμών βάσει κατανομών, ειδικά για τα βάρη, μπορεί να οδηγήσει σε πιο γρήγορη διαδικασία εκπαίδευσης, αλλά και σε πιο σταθερή διάδοση του σήματος τόσο προς τα εμπρός, όσο και προς τα πίσω, κατά τον υπολογισμό των μερικών παραγώγων του σφάλματος ως προς τις παραμέτρους.

Η επιλογή του πλήθους των επιπέδων, του πλήθους των νευρώνων, των συναρτήσεων ενεργοποίησης, του τρόπου αρχικοποίησης των παραμέτρων, αλλά και άλλων παραγόντων για τον ορισμό της αρχιτεκτονικής ενός μοντέλου, κάποιοι από τους οποίους θα σχολιαστούν και παρακάτω, δεν είναι πάντα απόλυτη. Χρειάζεται πειραματισμός για να βρεθεί ο κατάλληλος συνδυασμός αυτών των υπερπαραμέτρων (hyperparameters), όπως ονομάζονται, ο οποίος θα οδηγήσει σε μια επιτυχημένη εκπαιδευτική διαδικασία, αλλά και σε ένα μοντέλο που θα είναι ικανό να γενικεύει, και να βγάζει συμπεράσματα με ακρίβεια, αλλά και με μικρή απόκλιση σφάλματος.

Μια γενικότερη εποπτεία της αρχιτεκτονικής του μοντέλου καλείται ως εξής:

```
Input []:
1  model_1.summary()
```

```
Output []:
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                60

 dense_1 (Dense)             (None, 10)                110

 dense_2 (Dense)             (None, 10)                110

 dense_3 (Dense)             (None, 3)                 33

=================================================================
Total params: 313
Trainable params: 313
Non-trainable params: 0
_________________________________________________________________
```

Παραπάνω φαίνεται μια σύνοψη της αρχιτεκτονικής του μοντέλου, όπως έχει αρχικοποιηθεί προηγουμένως. Ειδικότερα για το μοντέλο με όνομα «model_1», όπως ορίστηκε κατά την αρχικοποίηση (name='model_1'), στην πρώτη στήλη εμφανίζονται τα ονόματα των επιπέδων από το χαμηλότερο επίπεδο εισόδου (input layer) προς το υψηλότερο επίπεδο εξόδου (output layer), καθώς και ο τύπος τους (Dense). Στη δεύτερη στήλη δηλώνεται η διάσταση εξόδου του κάθε επιπέδου, η οποία λειτουργεί ως είσοδος για το επόμενο επίπεδο στη σειρά, και στην τρίτη στήλη δηλώνεται το πλήθος των συνολικών παραμέτρων του κάθε επιπέδου, βαρών, σταθερών όρων, κ.α. Τέλος, δίνεται το συνολικό πλήθος όλων των παραμέτρων (Total params), καθώς και αυτών που ενημερώνονται (Trainable params) ή όχι (Non-trainable params) κατά τη διαδικασία της εκπαίδευσης για την ελαχιστοποίηση του σφάλματος.

2.3.7 Σύνταξη του Μοντέλου

Αφού αρχικοποιήθηκε το μοντέλο, πρέπει να οριστούν η συνάρτηση απώλειας (loss function), την οποία θα προσπαθήσει να ελαχιστοποιήσει το μοντέλο κατά την εκπαίδευσή του, και ο αλγόριθμος βελτιστοποίησης βάσει του οποίου θα ενημερώνονται οι παράμετροι των επιπέδων σε κάθε επιμέρους επανάληψη μάθησης. Η σύνταξη των παραπάνω στοιχείων γίνεται ως εξής:

```
Input []:
1  loss = keras.losses.SparseCategoricalCrossentropy()
2  optim = keras.optimizers.Nadam(learning_rate=0.001)
3  metr = keras.metrics.SparseCategoricalAccuracy()
4
5  model_1.compile(loss=loss, optimizer=optim, metrics=metr)
```

Για το αρχικοποιημένο μοντέλο model_1 έχουν οριστεί ως συνάρτηση απώλειας η SparseCategoricalCrossentropy (SCC), και ως αλγόριθμος βελτιστοποίησης ο Nadam με ρυθμό μάθησης 0.001.

Ειδικά για τη συνάρτηση απώλειας, ο αντίστοιχος μαθηματικός τύπος ο οποίος αφορά στην περίπτωση διαφορετικών κλάσεων είναι ο παρακάτω:

$$SCC = -\frac{1}{N} \cdot \sum_{i=1}^{N} \sum_{k=1}^{K} y_k^{(i)} \cdot \log(p_k^{(i)})$$

Όπου:

- N είναι το πλήθος των στοιχείων
- K είναι το πλήθος των διαφορετικών κλάσεων
- $y_k^{(i)}$ είναι η πραγματική πιθανότητα για το στοιχείο i να ανήκει ή οχι στην κλάση k, η οποία θα είναι είτε 1, είτε 0
- $p_k^{(i)}$ είναι η προβλεπόμενη πιθανότητα για το στοιχείο i να ανήκει στην κλάση k

Σε αυτό το σημείο μπορούμε να ορίσουμε προαιρετικά και μια επιπλέον μετρική, τη SparseCategoricalAccuracy (SCA), η οποία προσφέρει επιπλέον αξιολόγηση της εκπαιδευτικής διαδικασίας του μοντέλου, ως προς την πρόοδο της ακρίβειάς του, δηλαδή ως προς την ικανότητά του να αναγνωρίζει σωστά κάθε διαφορετική κλάση.

Η συνάρτηση απώλειας και ο αλγόριθμος βελτιστοποίησης αποτελούν επίσης σημαντικό κομμάτι του συνδυασμού των υπερπαραμέτρων του μοντέλου, και όπως με τις προηγούμενες, δεν υπάρχουν αυστηροί κανόνες για την επιλογή τους. Πρακτικά, η προγραμματιστική διαδικασία δεν θα μπορέσει να πραγματοποιηθεί χωρίς πρώτα να οριστούν αυτές οι δυο υπερπαράμετροι, ενώ η χρήση των επιπλέον μετρικών δεν είναι απαραίτητη για την εκτέλεσή της.

2.3.8 Εκπαίδευση του Μοντέλου

Αφού έγινε η απαραίτητη προεπεξεργασία στα δεδομένα, καθώς και η αρχικοποίηση του μοντέλου και η σύνταξή του με τις απαραίτητες υπερπαραμέτρους, το επόμενο βήμα είναι η διαδικασία της εκπαίδευσης, η οποία με τη βοήθεια του Keras γίνεται με τη μέθοδο fit() ως εξής:

```
Input []:
1  history = model_1.fit(X_train, y_train, epochs=300,
2                        batch_size=32, validation_split=0.2,
3                        shuffle=True, verbose=1)

Output []:
Epoch 1/300
172/172 [==============================] - 2s 4ms/step - loss: 2.7363 -
sparse_categorical_accuracy: 0.6173 - val_loss: 0.3537 -
val_sparse_categorical_accuracy: 0.8724
Epoch 2/300
172/172 [==============================] - 1s 4ms/step - loss: 0.2924 -
sparse_categorical_accuracy: 0.8919 - val_loss: 0.2961 -
val_sparse_categorical_accuracy: 0.8775
Epoch 3/300
172/172 [==============================] - 1s 3ms/step - loss: 0.2520 -
sparse_categorical_accuracy: 0.9035 - val_loss: 0.2627 -
val_sparse_categorical_accuracy: 0.8920
Epoch 4/300
.........
Epoch 298/300
172/172 [==============================] - 1s 4ms/step - loss: 0.1163 -
sparse_categorical_accuracy: 0.9593 - val_loss: 0.1173 -
val_sparse_categorical_accuracy: 0.9613
Epoch 299/300
172/172 [==============================] - 1s 4ms/step - loss: 0.1171 -
sparse_categorical_accuracy: 0.9573 - val_loss: 0.1196 -
val_sparse_categorical_accuracy: 0.9606
Epoch 300/300
172/172 [==============================] - 1s 4ms/step - loss: 0.1167 -
sparse_categorical_accuracy: 0.9588 - val_loss: 0.1284 -
val_sparse_categorical_accuracy: 0.9548
```

Αρχικά, στη μέθοδο fit() δηλώνονται τα δεδομένα Εκπαίδευσης, τα οποία είναι οι τιμές των χαρακτηριστικών X_train με τις αντίστοιχες ετικέτες τους y_train.

Στη συνέχεια, δηλώνεται το πλήθος των εποχών για το οποίο θέλουμε να εκπαιδευτεί το μοντέλο, το οποίο εδώ είναι 300, δηλαδή όλα τα δεδομένα Εκπαίδευσης θα περάσουν 300

φορές από το μοντέλο. Να σημειωθεί ότι κάθε επαναληπτική διαδικασία μάθησης περιλαμβάνει την προς τα εμπρός διάδοση των δεδομένων, για την εξαγωγή μιας πρόβλεψης και του σφάλματος αυτής, καθώς και την προς τα πίσω διάδοση, για την ενημέρωση των παραμέτρων του μοντέλου. Επομένως, στο τέλος της εκπαίδευσης, όλα τα δεδομένα θα έχουν πραγματοποιήσει 300 προς τα εμπρός διαδόσεις και 300 οπισθοδιαδόσεις συνολικά.

Έπειτα, δηλώνεται το πλήθος των στοιχείων κάθε παρτίδας για κάθε εποχή. Αναφέρθηκε ότι σε κάθε εποχή είναι ιδιαίτερα απαιτητική η υπολογιστική διαδικασία όλου του συνόλου δεδομένων ταυτόχρονα, επομένως η εκπαίδευση γίνεται σε παρτίδες όπου το πλήθος των στοιχείων είναι διαχειρίσιμο χωρίς μεγάλες απαιτήσεις μνήμης. Στη συγκεκριμένη περίπτωση, όλα τα δεδομένα θα περνάνε μέσα από το μοντέλο σε παρτίδες των 32 στοιχείων, μέχρι να ολοκληρωθεί κάθε εποχή.

Κατόπιν, ορίζεται το σύνολο δεδομένων Επιβεβαίωσης. Παραπάνω χρησιμοποιήθηκε η μέθοδος train_test_split() για να διαιρεθεί το αρχικό σύνολο δεδομένων data στα υποσύνολα Εκπαίδευσης και Αξιολόγησης με ποσοστά διαίρεσης 80% και 20% αντίστοιχα. Αναφέρθηκε όμως και η ανάγκη ενός συνόλου δεδομένων Επιβεβαίωσης κατά τη διαδικασία της εκπαίδευσης για να αξιολογεί την πρόοδο του μοντέλου. Ένας τρόπος θα ήταν να ξαναχρησιμοποιηθεί η μέθοδος train_test_split(), αλλά αυτή τη φορά πάνω στο σύνολο Εκπαίδευσης, για να γίνει μια επιμέρους ποσοστιαία διαίρεση σε ένα νέο σύνολο Εκπαίδευσης, σαφώς μικρότερης διάστασης, και σε ένα σύνολο Επιβεβαίωσης. Ωστόσο, στη συγκεκριμένη περίπτωση, η δήλωση validation_split=0.2 σημαίνει ότι από το ήδη δηλωθέν σύνολο Εκπαίδευσης (X_train και y_train) θα επιλεχθεί τυχαία ένα 20% των στοιχείων του, το οποίο θα έχει τον ρόλο του συνόλου Επιβεβαίωσης και δεν θα χρησιμοποιηθεί στην εκπαιδευτική διαδικασία.

Τέλος, με τη δήλωση shuffle=True τα δεδομένα Εκπαίδευσης ανακατεύονται για άλλη μια φορά (η μέθοδος train_test_split() διαίρεσε προηγουμένως και αυτή τα δεδομένα τυχαία), ενώ η εντολή verbose=1 επηρεάζει μόνο την εμφάνιση της εκπαιδευτικής διαδικασίας όπως αυτή παρουσιάζεται ως εξαγόμενο αποτέλεσμα μετά την κλήση της μεθόδου fit().

Ειδικότερα για την εκπαιδευτική διαδικασία όπως παρουσιάζεται στο Output, ενημερωνόμαστε για την εποχή στην οποία βρίσκεται το μοντέλο κάθε φορά (Epoch 1/300, κλπ), αλλά και για το βήμα, ή αλλιώς την παρτίδα την οποία επεξεργάζεται (172/172 κλπ). Υπενθυμίζεται ότι το αρχικό σύνολο Εκπαίδευσης αποτελούνταν από 6855 στοιχεία, από τα οποία το 20% παρακρατήθηκε ως σύνολο Επιβεβαίωσης, αφήνοντας το υπόλοιπο 80% ως το τελικό σύνολο Εκπαίδευσης, το οποίο πλέον έχει μέγεθος 5484. Συνεπώς, με μέγεθος

παρτίδας τα 32 στοιχεία, απαιτούνται 5484/32=171.375=172 βήματα για να περάσουν όλα τα δεδομένα από το μοντέλο σε μια εποχή.

Επίσης, ενημερωνόμαστε για την τιμή της συνάρτησης απώλειας (loss) και την επιπλέον μετρική ακρίβειας (sparse_categorical_accuracy), όπως υπολογίζονται στο τέλος κάθε εποχής, τόσο για το σύνολο Εκπαίδευσης, όσο και για το σύνολο Επιβεβαίωσης, για το οποίο οι αντίστοιχες τιμές έχουν το πρόθεμα val_.

Να σημειωθεί ότι, η μαθηματική σημασία του εκπαιδευμένου μοντέλου είναι ότι πλέον οι παράμετροι των επιπέδων του (βάρη και σταθεροί όρου) έχουν σταθεροποιηθεί σε συγκεκριμένες τιμές, όπως ενημερώθηκαν στο τελευταίο επαναληπτικό βήμα της εκπαιδευτικής διαδικασίας. Επομένως, για την απόκτηση κάθε πρόβλεψης, τα νέα δεδομένα που τροφοδοτούνται στο μοντέλο θα μετασχηματίζονται πάντα βάσει αυτών των τιμών των παραμέτρων.

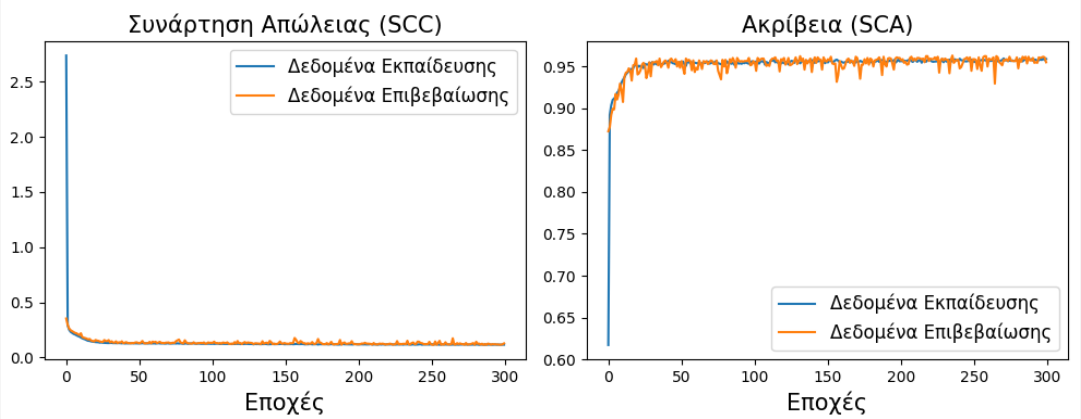
2.3.9 Διαγράμματα Προόδου

Οι τιμές συνάρτησης απώλειας και ακρίβειας οι οποίες παρουσιάζονται παραπάνω, αποθηκεύονται μέσω της εκπαιδευτικής διαδικασίας ως αντικείμενο τύπου history, και μπορούν να κληθούν με τον παρακάτω τρόπο για να οπτικοποιηθεί η πρόοδος της εκπαίδευσης του μοντέλου:

```
Input []:
1  plt.style.use('default')
2
3  fig, axs = plt.subplots(1, 2, figsize=(10,4))
4
5  axs[0].plot(history.history['loss'],
6              label='Δεδομένα Εκπαίδευσης')
7  axs[0].plot(history.history['val_loss'],
8              label='Δεδομένα Επιβεβαίωσης')
9  axs[0].set_title('Συνάρτηση Απώλειας (SCC)', fontsize='15')
10 axs[0].set_xlabel('Εποχές', fontsize='15')
11 axs[0].legend(fontsize='12')
12
13 axs[1].plot(history.history['sparse_categorical_accuracy'],
14             label='Δεδομένα Εκπαίδευσης')
15 axs[1].plot(history.history['val_sparse_categorical_accuracy'],
16             label='Δεδομένα Επιβεβαίωσης')
17 axs[1].set_title('Ακρίβεια (SCA)', fontsize='15')
18 axs[1].set_xlabel('Εποχές', fontsize='15')
19 axs[1].legend(fontsize='12')
```

```
20
21 plt.tight_layout()
```

Output []:



Η συμπεριφορά των παραπάνω διαγραμμάτων υποδηλώνει μια καλή εκπαιδευτική διαδικασία, καθώς η συνάρτηση απώλειας ελαχιστοποιείται τόσο για τα δεδομένα Εκπαίδευσης, όσο και για τα δεδομένα Επιβεβαίωσης κατά το πέρασμα των εποχών. Σε περίπτωση που υπήρχε αποκλίνουσα συμπεριφορά των γραμμών θα ήταν ενδεικτικό κακής εκπαίδευσης. Επίσης, το γεγονός ότι η ακρίβεια του μοντέλου μεγιστοποιείται παράλληλα με την ελαχιστοποίηση του σφάλματος επιβεβαιώνει περαιτέρω την ικανοποιητική πρόοδο στη διαδικασία μάθησης.

2.3.10 Αξιολόγηση

Αφού διαπιστώθηκε μια επιτυχημένη εκπαιδευτική διαδικασία, μπορούμε να περάσουμε στο επόμενο βήμα, το οποίο είναι η αξιολόγηση του μοντέλου πάνω σε νέα δεδομένα, τα οποία δεν έχει συναντήσει προηγουμένως. Η αξιολόγηση του εκπαιδευμένου μοντέλου θα γίνει στα δεδομένα Αξιολόγησης X_test των χαρακτηριστικών και y_test των αντίστοιχων ετικετών τους, έτσι όπως έγινε η διαίρεσή τους κατά το στάδιο της προεπεξεργασίας των δεδομένων

Εναλλακτική τακτική θα ήταν η αξιολόγηση να γίνει στο σύνολο Επιβεβαίωσης, σε περίπτωση φυσικά που είχε γίνει ξεχωριστή διαίρεση, όπως συζητήθηκε παραπάνω, και υπήρχε διαθέσιμο, διότι στην περίπτωσή μας επιλέχτηκε τυχαία κατά την εκπαίδευση του μοντέλου. Τα δεδομένα Αξιολόγησης θα χρησιμοποιούνταν την τελευταία στιγμή, όπου θα ήμασταν ευχαριστημένοι με την αρχιτεκτονική του μοντέλου, καθώς αυτή μπορεί να αλλάξει μέχρι την τελική εκδοχή που θα μας βόλευε και θα επηρεαζόταν από κάποια σημεία της αξιολόγησης.

Η παραπάνω ανησυχία έχει να κάνει με το γεγονός ότι όσο περισσότερο ο χρήστης αναπτύσσει και παραμετροποιεί το μοντέλο του, τόσο πιο επηρεασμένος είναι από τα δεδομένα που βλέπει και επεξεργάζεται, και τελικά το μοντέλο φτάνει σε σημείο να ικανοποιεί τις ανάγκες όλων των διαιρεμένων συνόλων δεδομένων. Επομένως, υπάρχει η ανάγκη να ακολουθούνται εκείνες οι διαδικασίες που συμβάλουν όσο το δυνατόν περισσότερο σε ένα πιο αμερόληπτο μοντέλο.

Μια πρώτη εποπτεία ως προς την ακρίβεια και την επίδοση του εκπαιδευμένου μοντέλου μπορεί να γίνει με τη μέθοδο evaluate() του Keras με τον παρακάτω τρόπο:

```
Input []:
1  model_1.evaluate(X_test, y_test)

Output []:
54/54 [==============================] - 0s 2ms/step - loss: 0.1285 -
sparse_categorical_accuracy: 0.9562
[0.12845464050769806, 0.9562426805496216]
```

Κατά τη διαδικασία της παραπάνω αξιολόγησης, όλα τα δεδομένα του συνόλου Αξιολόγησης, το οποίο αποτελείται από 1714 στοιχεία, περνάνε σε παρτίδες των 32 στοιχείων, επομένως σε 54 βήματα, μέσα από τα επίπεδα του μοντέλου, πραγματοποιώντας μόνο μια προς τα εμπρός διάδοση, στο τέλος της οποίας υπολογίζεται το συνολικό σφάλμα και η ακρίβεια.

Τα τελικά αποτελέσματα σφάλματος και ακρίβειας για το σύνολο Αξιολόγησης είναι αντίστοιχα 0.1258 και 0.9562 (οι μη στρογγυλοποιημένες τιμές δίνονται στο τέλος σε μορφή λίστας), ενώ, κοιτώντας τα αναλυτικά βήματα εκπαίδευσης του μοντέλου, οι τελικές αντίστοιχες τιμές για τα δεδομένα Εκπαίδευσης στο τέλος της τελευταίας εποχής (Epoch 300/300) είναι 0.1167 και 0.9588.

Είναι λογικό να υπάρχει μικρή απόκλιση, ιδιαίτερα προς τα κάτω, καθώς το μοντέλο είναι εκπαιδευμένο, και ειδικά παραμετροποιημένο, για τα δεδομένα Εκπαίδευσης. Ωστόσο, το γεγονός ότι οι τιμές σφάλματος και ακρίβειας κυμαίνονται σχετικά στα ίδια πλαίσια είναι καλή ένδειξη για το ότι το εκπαιδευμένο μοντέλο έχει μάθει να γενικεύει ικανοποιητικά όταν τροφοδοτείται με νέα δεδομένα, παρόμοιου μοτίβου με εκείνα πάνω στα οποία έχει εκπαιδευτεί.

2.3.11 Λήψη Προβλέψεων

Προτού συνεχιστεί το στάδιο της αξιολόγησης του μοντέλου, είναι καλό να εξηγηθεί και η διαδικασία λήψης των προβλέψεων από το μοντέλο.

Ειδικά για τις προβλέψεις του συνόλου Αξιολόγησης έχουμε:

```
Input []:
1  y_test_pred = model_1.predict(X_test)

Output []:
54/54 [==============================] - 0s 4ms/step
```

Ως εξαγόμενο αποτέλεσμα παρουσιάζεται και πάλι η πρόοδος των 54 βημάτων, ανάλογο των μεθόδων fit() και evaluate(). Πλέον, στον πίνακα (array) y_test_pred είναι αποθηκευμένες οι προβλέψεις των δεδομένων εισαγωγής (input) X_test, όπως προέκυψαν ως έξοδοι (output) από το μοντέλο, μετά την προς τα εμπρός διάδοσή τους μέσω των επιπέδων του, και είναι τα παρακάτω:

```
Input []:
1  print(y_test_pred)
2  print(y_test_pred.shape)

Output []:
[[9.98784542e-01 1.01687305e-03 1.98457055e-04]
 [9.99160349e-01 7.33431662e-04 1.06090185e-04]
 [1.12544277e-23 2.97740996e-02 9.70225871e-01]
 ...
 [5.12133842e-29 8.78455956e-03 9.91215527e-01]
 [4.79564712e-14 9.91492569e-01 8.50743335e-03]
 [1.41280176e-09 9.99159455e-01 8.40574969e-04]]
(1714, 3)
```

Υπενθυμίζεται ότι τα δεδομένα Αξιολόγησης αποτελούνται από 1714 στοιχεία, και ότι ο πίνακας των ετικετών του κάθε στοιχείου έχει επίσης διάσταση 1714, όπως φαίνεται και παρακάτω:

```
Input []:
1  print(y_test.shape)

Output []:
(1714,)
```

Αντιθέτως, η έξοδος y_test_pred του μοντέλου είναι πίνακας διάστασης (1714x3), διότι το τελευταίο επίπεδο εξόδου είχε μέγεθος τριών νευρώνων από την αρχικοποίηση της

αρχιτεκτονικής του, και μάλιστα τού εφαρμοζόταν η συνάρτηση ενεργοποίησης softmax. Επομένως, κάθε γραμμή του εξαγόμενου πίνακα αποτελείται από τρία στοιχεία τα οποία εκφράζουν την πιθανότητα να είναι σωστή κάθε αντίστοιχη κατάσταση του ρουλεμάν, έτσι όπως έχουν οριστεί (κανονική, έλλειψη και περίσσεια ποσότητα λιπαντικού).

Συνεπώς, με τον παρακάτω τρόπο επιλέγεται η θέση του στοιχείου του πίνακα στην οποία η πιθανότητα είναι η μεγαλύτερη:

```
Input []:
1  y_test_pred = np.argmax(y_test_pred, axis=1)
2  print(y_test_pred)
3  print(y_test_pred.shape)

Output []:
[0 0 2 ... 2 1 1]
(1714,)
```

Επιπλέον, για εποπτικούς λόγους, παρακάτω παρουσιάζονται οι πρώτες 15 πραγματικές ετικέτες του συνόλου Αξιολόγησης X_test με τις αντίστοιχες προβλέψεις του μοντέλου:

```
Input []:
1  print(y_test[:15])
2  print(y_test_pred[:15])

Output []:
[0 0 1 1 1 0 0 1 0 0 0 0 0 0 1]
[0 0 2 1 2 0 0 1 0 0 0 0 0 0 1]
```

Ενδεικτικά, παρατηρούμε ότι στις πρώτες 15 προβλέψεις, η $3^{η}$ και η $5^{η}$, ενώ στην πραγματικότητα αναφέρονται ως καταστάσεις λειτουργίας με έλλειψη λιπαντικού, αναγνωρίστηκαν ως καταστάσεις με περίσσεια λιπαντικού από το εκπαιδευμένο μοντέλο.

2.3.12 Αξιολόγηση με Πίνακα Σύγχυσης

Από την πρώτη συμβατική αξιολόγηση των νέων δεδομένων Αξιολόγησης παρατηρήθηκε σφάλμα 0.1258 από την συνάρτηση απώλειας, και ακρίβεια 0.9562 ή 95.62%. Οι δυο τιμές αναφέρονται στο συγκεντρωτικό αποτέλεσμα των δεδομένων και δεν προσφέρουν περαιτέρω πληροφορίες για το πώς κατανέμεται το σφάλμα και η ακρίβεια στις τρεις διαφορετικές καταστάσεις του ρουλεμάν.

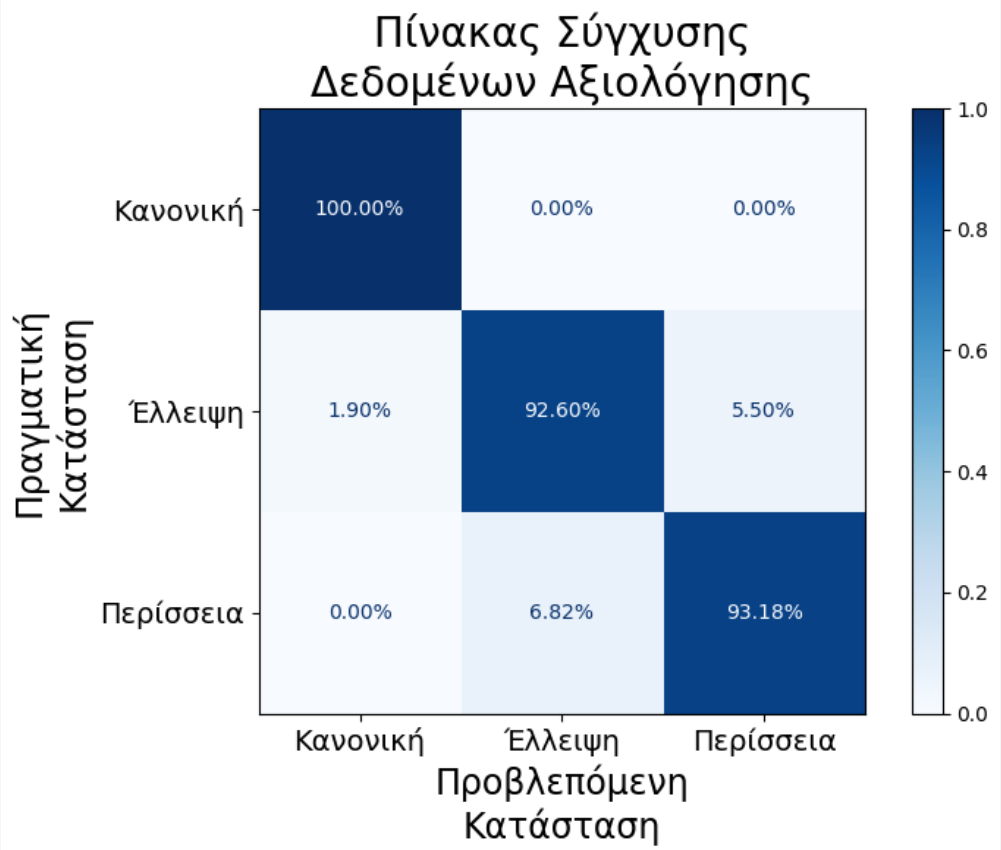
Για τον λόγο αυτό, και επειδή το συγκεκριμένο πρόβλημα είναι τύπου Κατηγοριοποίησης και περιλαμβάνει διακριτές καταστάσεις, ένας πιο αντιπροσωπευτικός τρόπος αξιολόγησης

είναι μέσω του Πίνακα Σύγχυσης (Confusion Matrix), ο οποίος παράγεται για το σύνολο Αξιολόγησης σύμφωνα με τον παρακάτω κώδικα:

Input []:

```
# y_test_pred = model_1.predict(X_test)
# y_test_pred = np.argmax(y_test_pred, axis=1)

fig, ax = plt.subplots(figsize=(8, 6))
labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
ConfusionMatrixDisplay.from_predictions(
          y_test, y_test_pred, display_labels=labels,
          normalize='true', values_format='.2%',
          include_values=True, cmap='Blues',
          xticks_rotation='horizontal',
          ax=ax, colorbar=True)
_ = ax.set_title('Πίνακας Σύγχυσης\n
                  Δεδομένων Αξιολόγησης', fontsize='20')
plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
plt.tick_params(labelsize='14')
plt.tight_layout()
plt.show()
```

Output []:



Συνεπώς, παρατηρούμε ειδικότερα για κάθε κατάσταση λίπανσης του ρουλεμάν τα αντίστοιχα ποσοστά επιτυχίας, τα οποία παρουσιάζονται στην κύρια διαγώνιο του πίνακα, αλλά και τα ποσοστά όπως κατανέμονται σε περιπτώσεις λάθος πρόβλεψης.

Συγκεκριμένα για τις δυο επιβλαβείς καταστάσεις, η πιθανότητα για μια κατάσταση έλλειψης λιπαντικού να αναγνωριστεί σωστά από το μοντέλο αποτελεί το 92.60%, ενώ για την κατάσταση υπερβολικής λίπανσης το αντίστοιχο ποσοστό είναι 93.18%. Επίσης, μια κατάσταση έλλειψης έχει 5.50% πιθανότητα να προβλεφθεί ως κατάσταση περίσσειας λίπανσης, ενώ αντίθετα, μια κατάσταση υπερβολικού λιπαντικού προβλέπεται κατά 6.82% ως κατάσταση έλλειψης.

Ωστόσο, ιδιαίτερης σημασίας κρίνονται οι περιπτώσεις όπου μια κατάσταση φυσιολογικής λειτουργίας μπορεί να αναγνωριστεί ως επιβλαβής, ή αντίθετα, μια προβληματική κατάσταση να συσχετιστεί με μια κανονική και να προβλεφθεί αναλόγως.

Ειδικότερα, παρατηρούμε ότι στο σύνολο δεδομένων που αξιολογήθηκε, μια κανονική κατάσταση αναγνωρίστηκε σωστά κατά το 100% των περιπτώσεων, χωρίς την ύπαρξη πιθανότητας να συσχετιστεί με μια επιβλαβή. Για τον λόγο αυτό, αν θεωρηθεί η φυσιολογική κατάσταση ως «θετική» (positive) και ο συνδυασμός των δυο επιβλαβών ως «αρνητική» (negative), τότε δεν υπάρχει περίπτωση να παρουσιαστεί κάποιο Ψευδές Αρνητικό (False Negative - FN) σφάλμα, ή όπως είναι και αλλιώς γνωστό, σφάλμα Τύπου II. Από την άλλη, παρατηρήθηκε ότι το 1.90% των περιπτώσεων έλλειψης λιπαντικού χαρακτηρίστηκε ως φυσιολογική κατάσταση, δηλαδή παρουσιάστηκαν στα αποτελέσματα σφάλματα Τύπου I, ή αλλιώς Ψευδή Θετικά (False Positive - FP), ενώ καμία πραγματική περίπτωση υπερβολικού λιπαντικού δεν αναγνωρίστηκε ως φυσιολογική.

Παρά το γεγονός ότι το σφάλμα Τύπου I είναι ιδιαίτερα μικρό, σε σχέση με τα υπόλοιπα, αξίζει να σημειωθεί ότι στην περίπτωση μιας πραγματικής βιομηχανικής παραγωγικής διαδικασίας, το να παραβλεφθεί μια προβληματική λειτουργική κατάσταση ως φυσιολογική, μπορεί να οδηγήσει σε βάθος χρόνου σε σημαντικά προβλήματα φθοράς ή ακόμα και σε μεγάλη ζημιά του εξοπλισμού. Αντίθετα, τα σφάλματα Τύπου II, δεν αποτελούν ιδιαίτερο κίνδυνο για τα βιομηχανικά εξαρτήματα, και στην χειρότερη περίπτωση μπορεί να προκαλέσουν κάποια καθυστέρηση στην παραγωγή μέχρι να γίνει κάποιος έλεγχος συντήρησης και να επιβεβαιωθεί ο λάθος συναγερμός.

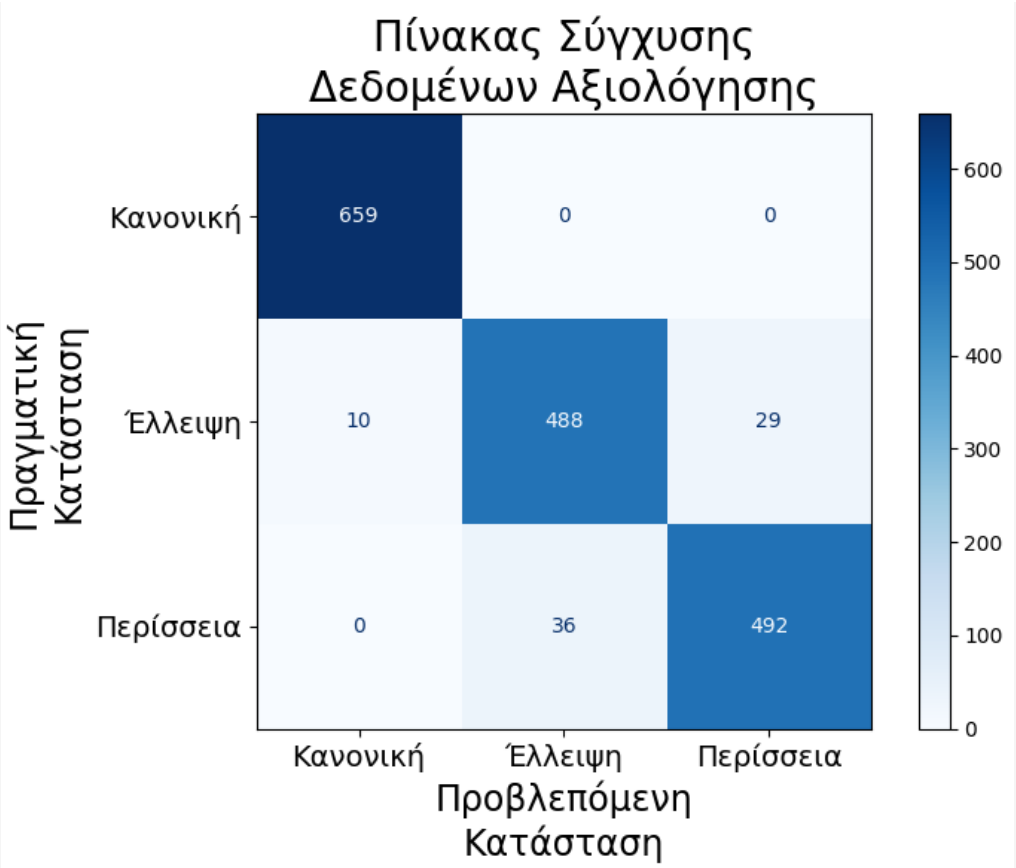
2.3.13 Μέτρα Αξιολόγησης του Πίνακα Σύγχυσης

Επιπλέον μέτρα αξιολόγησης των δυνατοτήτων πρόβλεψης καταστάσεων από το μοντέλο μπορούν να παραχθούν από τον Πίνακα Σύγχυσης και είναι τα παρακάτω:

- Precision (Ακρίβεια)
- Recall (Ανάκληση)
- f1-score

Για τον υπολογισμό των μέτρων αυτών χρειάζονται οι ακέραιες τιμές του Πίνακα Σύγχυσης, σε αντίθεση με τα κανονικοποιημένα ποσοστά που παρουσιάστηκαν, δηλαδή είναι απαραίτητα τα πλήθη των Αληθών Θετικών (True Positive - TP), Αληθών Αρνητικών (True Negative - TN), Ψευδών Θετικών (False Positive - FP) και Ψευδών Αρνητικών (False Negative - FN) περιπτώσεων. Γενικότερα, για τον υπολογισμό των παραπάνω μέτρων αξιολόγησης, μια κατάσταση θεωρείται «θετική» όταν τα μέτρα υπολογίζονται ως προς αυτήν, ενώ ο συνδυασμός των δυο υπολοίπων θεωρείται ως «αρνητική» κατάσταση. Ο ζητούμενος Πίνακας Σύγχυσης με τα ακέραια πλήθη καλείται ως εξής:

```
Input []:
1       # y_test_pred = model_1.predict(X_test)
2       # y_test_pred = np.argmax(y_test_pred, axis=1)
3
4       fig, ax = plt.subplots(figsize=(8, 6))
5       labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
6       ConfusionMatrixDisplay.from_predictions(
7                 y_test, y_test_pred, display_labels=labels,
9                 include_values=True, cmap='Blues',
10                xticks_rotation='horizontal',
11                ax=ax, colorbar=True)
12      _ = ax.set_title('Πίνακας Σύγχυσης\n
13                        Δεδομένων Αξιολόγησης', fontsize='20')
14      plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
15      plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
16      plt.tick_params(labelsize='14')
17      plt.tight_layout()
18      plt.show()

Output []:
```



Επίσης, οι μαθηματικοί τύποι των παραπάνω μέτρων αξιολόγησης είναι οι εξής:

- $precision = TP\ /\ (TP + FP)$
- $recall = TP\ /\ (TP + FN)$
- $f1 =\ 2\ /\ (\ 1/precision\ + 1/recall)$

Τελικά, τα παραπάνω μέτρα αξιολόγησης για κάθε κατάσταση μπορούν να ληφθούν γρήγορα και σε συνοπτική μορφή με τον παρακάτω τρόπο:

```
Input []:
1  # y_test_pred = model_1.predict(X_test)
2  # y_test_pred = np.argmax(y_test_pred, axis=1)
3
4  targets = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
5  cr_test = classification_report( y_test, y_test_pred,
6                                   target_names=targets)
7  print(cr_test)

Output []:
              precision    recall  f1-score   support

    Κανονική       0.99      1.00      0.99       659
     Έλλειψη       0.93      0.93      0.93       527
   Περίσσεια       0.94      0.93      0.94       528
```

```
              precision    recall  f1-score   support

    accuracy                         0.96      1714
   macro avg       0.95      0.95      0.95      1714
weighted avg       0.96      0.96      0.96      1714
```

Ειδικότερα, όσον αφορά στη σημασία του κάθε μέτρου, το precision αναφέρεται στην ικανότητα του μοντέλου να μην αναγνωρίζει κάποια «αρνητική» κατάσταση ως «θετική», το recall αναφέρεται στην ικανότητά του να βρίσκει όλες τις «θετικές» καταστάσεις, ενώ το f1-score, το οποίο είναι ο αρμονικός μέσος των δυο προηγούμενων, παίρνει τιμές μεταξύ 0 και 1, και χρησιμοποιείται γενικότερα ως μεμονωμένο ισοδύναμο μέτρο των προηγούμενων για τη σύγκριση και την αξιολόγηση μεταξύ διαφορετικών μοντέλων. Η σημαντικότητα μεταξύ των precision και recall κρίνεται κάθε φορά σύμφωνα με το πρόβλημα που εξετάζεται, και πολλές φορές πρέπει να εξεταστεί το αντάλλαγμα (trade-off) μεταξύ των τιμών τους, ώστε να δοθεί προτεραιότητα σε εκείνο το μέτρο που αρμόζει καλύτερα στα δεδομένα του προβλήματος και τα συμπεράσματα που προτιμώνται. Για παράδειγμα, στην περίπτωσή μας θα θέλαμε μεγαλύτερο ποσοστό precision στην κανονική κατάσταση, για να μην παραβλεφθούν περιπτώσεις βλαβών, ενώ το recall περιλαμβάνει τους λάθος συναγερμούς που αναφέρθηκαν, οι οποίοι δεν είναι σχετικά επικίνδυνοι για τη βιομηχανική παραγωγή.

2.3.14 Αποθήκευση Μοντέλου και Αποτελεσμάτων

Αφού εκπαιδευτεί και αξιολογηθεί το μοντέλο, αποτελεί συνιστώμενη πρακτική η αποθήκευσή του, έτσι ώστε να γίνει αμέσως η κλήση του την επόμενη φορά χωρίς να χρειαστεί εκ νέου εκπαίδευση, κάτι που θα άλλαζε εντελώς την εσωτερική παραμετροποίηση των επιπέδων του, άλλα όχι απαραίτητα σε μεγάλο βαθμό.

Η αποθήκευση του μοντέλου γίνεται με τον παρακάτω τρόπο:

```
Input []:
1  keras.models.save_model(model_1, '/content/drive/MyDrive/
2                    Colab Notebooks/thesis/saved_models/model_1')
```

Συνεπώς, το μοντέλο αποθηκεύεται στο Google Drive, στον φάκελο saved_models, στη διαδρομή που έχουμε ορίσει, και αν δεν υπάρχουν οι επιμέρους φάκελοι, τότε δημιουργούνται αυτόματα με την παραπάνω εντολή.

Η εκ νέου κλήση του αποθηκευμένου μοντέλου γίνεται ως εξής:

```
Input []:
```

```
1  loaded_model_1 = keras.models.load_model('/content/drive/MyDrive/
2                    Colab Notebooks/thesis/saved_models/model_1')
```

Με τον παραπάνω τρόπο φορτώνεται το αποθηκευμένο μοντέλο στη νέα μεταβλητή loaded_model_1, και κάθε επιμέρους μέθοδος, ανάλογη των παραπάνω που χρησιμοποιεί το μοντέλο, όπως τις evaluate() και predict(), γίνεται με το συγκεκριμένο πρόθεμα.

Επίσης σημαντική τακτική αποτελεί και η αποθήκευση των τιμών σφάλματος και ακρίβειας, όπως υπολογίστηκαν σε κάθε εποχή κατά την εκπαιδευτική διαδικασία, και βρίσκονται προσωρινά αποθηκευμένες στο αντικείμενο history.

Η μόνιμη αποθήκευση των τιμών σφάλματος και ακρίβειας γίνεται ως εξής:

```
Input []:
1  hist_df = pd.DataFrame(history.history)
2  hist_df.to_csv('/content/drive/MyDrive/Colab Notebooks/
3                  thesis/saved_models/model_1.csv' , index=False)
```

Συνεπώς, πρώτα δημιουργείται ένα αντικείμενο τύπου DataFrame με τις τιμές, κυρίως για λόγους ευκολίας στην ανάγνωση, και στη συνέχεια αποθηκεύεται στη δοσμένη διαδρομή ως αρχείο .csv, ενώ η δήλωση index=False υπάρχει για να μη γίνει αποθήκευση του ευρετηρίου σε νέα επιπλέον στήλη.

Το παραπάνω αρχείο καλείται και πάλι, με τον ίδιο τρόπο όπως εισάχθηκαν στην αρχή τα δεδομένα του προβλήματος με τη συνάρτηση read_csv(), ως εξής:

```
Input []:
1  datadf = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
2                        thesis/saved_models/model_1.csv')
3  datadf

Output []:
```

	loss	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	2.736342	0.617250	0.353663	0.872356
1	0.292385	0.891867	0.296139	0.877462
2	0.252013	0.903538	0.262685	0.892050
3	0.233419	0.909920	0.248003	0.899344
4	0.224547	0.911561	0.240120	0.898614
...	...	...	...	...
295	0.116994	0.958242	0.118886	0.959883
296	0.116429	0.957513	0.116256	0.960613
297	0.116331	0.959336	0.117288	0.961342
298	0.117120	0.957330	0.119601	0.960613
299	0.116672	0.958789	0.128429	0.954778

300 rows × 4 columns

Οι παραπάνω αποθηκεύσεις επιτρέπουν τη φόρτωση των ίδιων μοντέλων και των αποτελεσμάτων τους για περαιτέρω επεξεργασία και ανάλυση χωρίς τον φόβο να χαθεί τυχόν πρόοδος της ανάλυσης και αξιολόγησης του μοντέλου.

Με ανάλογο τρόπο μπορεί κανείς να αποθηκεύσει και τα σύνολα δεδομένων, έτσι ώστε να χρησιμοποιούνται τα ίδια δεδομένα Εκπαίδευσης και Αξιολόγησης τόσο στην ανάπτυξη πολλαπλών μοντέλων με διαφορετικές αρχιτεκτονικές, όσο και στη σύγκριση μεταξύ τους, προσφέροντας έτσι μια πιο αμερόληπτη οπτική και πιο αντιπροσωπευτική συγκριτική διαδικασία.

2.4 Δεύτερο Μοντέλο Νευρωνικού Δικτύου

Υπενθυμίζεται ότι το προηγούμενο εκπαιδευμένο μοντέλο MLP, κατά τον έλεγχο της γενίκευσης και των επιδόσεών του πάνω στα δεδομένα Αξιολόγησης X_test, κατάφερε να πετύχει συνολική τιμή σφάλματος (loss) και ακρίβειας (accuracy) 0.1258 και 0.9562 αντίστοιχα.

Η τιμή της ακρίβειας υποδηλώνει το πόσο σωστά κατάφερε το μοντέλο να προβλέψει τις πραγματικές καταστάσεις και εκφράζει τον μέσο όρο της συνολικής επιτυχίας για όλες τις καταστάσεις, κάτι που δεν είναι πάντα αντιπροσωπευτικό, όπως είδαμε από τον Πίνακα Σύγχυσης προηγουμένως, αλλά είναι μια καλή πρώτη οπτική προσέγγιση. Από την άλλη, το σφάλμα που υπολογίζεται μέσω της συνάρτησης απώλειας εκφράζει τη σοβαρότητα των λάθος προβλέψεων, επομένως είναι σημαντική η προσπάθεια να μειωθεί όσο το δυνατόν περισσότερο, καθώς μεγάλες τιμές σφάλματος είναι ενδεικτικές σοβαρών λαθών στις

προβλέψεις, όπως για παράδειγμα η παράβλεψη μια επιβλαβούς κατάστασης ως φυσιολογικής που παρατηρήθηκε επίσης από τον Πίνακα Σύγχυσης.

Συνεπώς, αν και τα τωρινά αποτελέσματα οδήγησαν σε σχετικά μεγάλες τιμές ακρίβειας, και παρά το γεγονός ότι είναι παρόμοια με αυτά που καταγράφηκαν κατά το τέλος της εκπαιδευτικής διαδικασίας (0.1167 σφάλμα και 0.9588 ακρίβεια στην τελευταία εποχή), πρέπει να γίνει προσπάθεια μείωσής τους, ειδικά της τιμής σφάλματος.

Η ανάπτυξη ενός μοντέλου για την επίτευξη μιας καλύτερης επίδοσης απαιτεί προσεγμένες μικρές προσαρμογές (fine-tuning), τόσο στην αρχιτεκτονική του υπάρχοντος μοντέλου, όσο και στην προεπεξεργασία των δεδομένων.

Στη συγκεκριμένη περίπτωση, μια επιπλέον διαδικασία που θα ενσωματωθεί στην εκπαίδευση είναι η αλλαγή κλίμακας των δεδομένων. Το συγκεκριμένο βήμα είναι ιδιαίτερα σημαντικό στην εκπαίδευση ενός μοντέλου, και εσκεμμένα παραβλέφθηκε προηγουμένως για να τονιστεί η σημασία του.

2.4.1 Εισαγωγή Πακέτων

Εισάγονται εξ αρχής όλα τα πακέτα που είναι απαραίτητα για την προεπεξεργασία των δεδομένων και την ανάπτυξη του μοντέλου:

```
Input []:
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4  from tensorflow import keras
5  from sklearn.model_selection import train_test_split
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.metrics import ConfusionMatrixDisplay,
8                              classification_report
```

Η επιπλέον προσθήκη αφορά στο αντικείμενο StandardScaler().

2.4.2 Εισαγωγή και Διαίρεση Δεδομένων

Γίνεται η εισαγωγή των δεδομένων από τη διαδρομή όπου βρίσκονται αποθηκευμένα στο Google Drive ως αρχείο .csv, και έπειτα γίνεται η αλλαγή στην αρίθμηση των ετικετών από 1,2,3 σε 0,1,2 για να μπορέσει να τις επεξεργαστεί ο τύπος και η αρχιτεκτονική του μοντέλου που έχουμε επιλέξει:

```
Input []:
1  data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
2                      thesis/data/bearing_vibration_metrics.csv')
3  data['Bearing State'] = data['Bearing State'] - 1
4  data
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	0
1	0.001095	0.713077	3.244615	4.716923	26.299999	0
2	0.001079	0.697143	3.227857	4.795000	26.299999	0
3	0.001080	0.700000	3.213077	4.793077	26.208462	0
4	0.001088	0.700000	3.241538	4.783077	26.200001	0
...	...	...	...	...	...	...
8564	0.000728	0.880769	7.337692	8.304615	26.866153	2
8565	0.000703	0.853846	6.926154	8.102308	26.806153	2
8566	0.000705	0.834615	6.556923	7.822308	26.892307	2
8567	0.000713	0.862308	6.893846	7.989231	26.799999	2
8568	0.000708	0.866154	7.378462	8.506923	26.799999	2

8569 rows × 6 columns

Το εξαγόμενο DataFrame είναι το ίδιο με εκείνο του υποκεφαλαίου 2.3.2, όπου επεξηγείται αναλυτικότερα η αλλαγή των ετικετών, με μόνη διαφορά την οπτικοποίηση των στοιχείων λόγω της εντολής print() που εφαρμόζεται στην προηγούμενη περίπτωση.

Στη συνέχεια, το σύνολο των στοιχείων διαιρείται στα δεδομένα Εκπαίδευσης και Αξιολόγησης, όπως προηγουμένως, βάσει της ποσοστιαίας κατανομής των τριών διαφορετικών περιπτώσεων:

```
Input []:
1  X_train, X_test, y_train, y_test = train_test_split(
2                                      data.iloc[:,:-1],
3                                      data['Bearing State'],
4                                      test_size = 0.2,
5                                      shuffle=True,
6                                      stratify=data['Bearing State'])
```

Έπειτα τα παράγωγα σύνολα δεδομένων μετασχηματίζονται σε στοιχεία τύπου πίνακα (arrays) για καλύτερη διαχείριση:

```
Input []:
1  X_train = np.asarray(X_train)
2  X_test = np.asarray(X_test)
3  y_train = np.asarray(y_train)
4  y_test = np.asarray(y_test)
```

2.4.3 Αλλαγή Κλίμακας Δεδομένων

Αναφέρθηκε ότι, ένα από τα αναγκαία μέτρα προετοιμασίας των δεδομένων για μια πιο ομαλή και επιτυχημένη εκπαιδευτική διαδικασία είναι η αλλαγή των τιμών των δεδομένων σε μια πιο ομοιόμορφη κλίμακα. Είναι προφανές, κοιτώντας τον πίνακα δεδομένων data, ότι οι τιμές για τα πέντε χαρακτηριστικά κυμαίνονται σε ένα εύρος μεταξύ χιλιοστών (V-RMS) έως και δεκάδων (Temperature), και μια τέτοια ανισορροπία στα δεδομένα μπορεί να προκαλέσει δυσκολίες στη διαχείρισή τους από το μοντέλο κατά τη διαδικασία μάθησης.

Επομένως, για το συγκεκριμένο πρόβλημα επιλέχθηκε η αλλαγή κλίμακας των δεδομένων με τη μέθοδο της Κανονικοποίησης (Standardization). Σύμφωνα με τη μέθοδο αυτή, οι τιμές κάθε χαρακτηριστικού-στήλης μετασχηματίζονται έτσι ώστε οι τελικές τιμές τους να ακολουθούν την Κανονική Κατανομή (Normal Distribution), δηλαδή να έχουν μέσο όσο μηδέν και τυπική απόκλιση τη μονάδα. Ειδικότερα, η αλλαγή των τιμών εφαρμόζεται με τον παρακάτω μαθηματικό τύπο:

$$X_{scaled} = \frac{X - X_{mean}}{X_{std}}$$

Όπου:

- X είναι οι αρχικές τιμές της στήλης του εκάστοτε χαρακτηριστικού
- X_{mean} είναι ο μέσος όρος των αρχικών τιμών X
- X_{std} είναι η τυπική απόκλιση των αρχικών τιμών X
- X_{scaled} είναι οι νέες μετασχηματισμένες τιμές του εκάστοτε χαρακτηριστικού

Ο παραπάνω μετασχηματισμός επιτυγχάνεται με τη βοήθεια της βιβλιοθήκης Scikit-learn ως εξής:

```
Input []:
1  scaler = StandardScaler()
2  scaler.fit(X_train)
3  X_train_scaled = scaler.transform(X_train)
4  X_test_scaled = scaler.transform(X_test)
```

Αρχικά, ορίζεται ο μετασχηματιστής StandardScaler() και στη συνέχεια εφαρμόζεται πάνω στο σύνολο Εκπαίδευσης για τον υπολογισμό του μέσου όρου και της τυπικής απόκλισης του κάθε χαρακτηριστικού-στήλης που θα χρησιμοποιηθούν για την αλλαγή των τιμών. Έπειτα, με τη μέθοδο transform() τα σύνολα Εκπαίδευσης και Αξιολόγησης μετασχηματίζονται σύμφωνα με τον παραμετροποιημένο μετασχηματιστή και τον παραπάνω μαθηματικό τύπο.

Πρέπει να σημειωθεί ότι όλοι οι μετασχηματισμοί γίνονται βάσει των μέσων όρων και των τυπικών αποκλίσεων του συνόλου πάνω στο οποίο θα εκπαιδευτεί το μοντέλο, και δεν εφαρμόζονται ξεχωριστοί μετασχηματιστές StandardScaler() για κάθε νέο σύνολο που θα περάσει από το πλέον εκπαιδευμένο μοντέλο για πρόβλεψη ή για αξιολόγηση.

Για μια εποπτεία των μετασχηματισμών σε σχέση με τις αρχικές τιμές καλούμε την πρώτη γραμμή του συνόλου Εκπαίδευσης:

```
Input []:
1  print(X_train[0])
2  print(X_train_scaled[0])

Output []:
[1.08076924e-03 6.99999988e-01 3.25923078e+00 4.79538466e+00
 2.61000004e+01]
[ 1.43505648 -0.22983724 -0.91031646 -1.0660772  -0.41801012]
```

Συνεπώς, βλέπουμε ότι στα μετασχηματισμένα χαρακτηριστικά υπάρχει μεγαλύτερη ομοιογένεια ως προς τις τιμές τους. Υπενθυμίζεται ότι η διαίρεση του αρχικού συνόλου δεδομένων έγινε τυχαία, επομένως η παραπάνω είναι επίσης μια τυχαία εγγραφή.

Ακόμη, να σημειωθεί ότι η επιστημονική γραφή $ae-xx$ ή $ae+xx$ δηλώνει τον πολλαπλασιασμό του αριθμού a με τις δυνάμεις του δέκα 10^{-xx} ή 10^{+xx} αντίστοιχα, το οποίο σημαίνει ότι η υποδιαστολή του αριθμού a μετακινείται αριστερά ή δεξιά κατά xx θέσεις, αναλόγως του πρόσημου του εκθέτη της δύναμης.

Για επιπλέον εποπτεία και επιβεβαίωση ότι η νέα κατανομή των μετασχηματισμένων τιμών είναι η Κανονική, παρουσιάζονται οι μέσοι όροι και οι τυπικές αποκλίσεις των χαρακτηριστικών, δηλαδή ανά στήλη, πριν και μετά την αλλαγή κλίμακας:

```
Input []:
1  print(X_train[:,0].mean(), X_train[:,0].std())
2  print(X_train[:,1].mean(), X_train[:,1].std())
3  print(X_train[:,2].mean(), X_train[:,2].std())
4  print(X_train[:,3].mean(), X_train[:,3].std())
5  print(X_train[:,4].mean(), X_train[:,4].std())
```

```
Output []:
0.0008523683088742336 0.00015908255923220035
0.7179629148031346 0.07733879756288244
4.582620557588386 1.4529545910017905
6.300689002825331 1.4125651814196625
26.414589657025143 0.7485958693628189
```

Τα παραπάνω μέτρα θέσης και διασποράς των αρχικών τιμών του κάθε χαρακτηριστικού είναι ανάλογα με αυτά που υπολογίζονται από τον συνολικό πίνακα δεδομένων data, όπως παρουσιάζονται στο Παράρτημα Β', και η μικρή απόκλιση των παραπάνω οφείλεται στο ότι το σύνολο Εκπαίδευσης αποτελεί κομμάτι του συνόλου data.

Παρακάτω παρουσιάζονται οι αντίστοιχες τιμές των μέσων τιμών και τυπικών αποκλίσεων του συνόλου Εκπαίδευσης μετά τον μετασχηματισμό των χαρακτηριστικών σύμφωνα με τη μέθοδο της Κανονικοποίησης:

```
Input []:
1  print(X_train_scaled[:,0].mean(), X_train_scaled[:,0].std())
2  print(X_train_scaled[:,1].mean(), X_train_scaled[:,1].std())
3  print(X_train_scaled[:,2].mean(), X_train_scaled[:,2].std())
4  print(X_train_scaled[:,3].mean(), X_train_scaled[:,3].std())
5  print(X_train_scaled[:,4].mean(), X_train_scaled[:,4].std())

Output []:
2.9230204447023816e-16 1.0
-1.570346089973088e-16 0.9999999999999999
2.487676974214793e-17 0.9999999999999999
-1.8450270892093048e-16 1.0
-3.496222680844373e-15 1.0
```

Συνεπώς, κάθε στήλη τιμών των μετασχηματισμένων χαρακτηριστικών του συνόλου Εκπαίδευσης ακολουθεί όντως την Κανονική Κατανομή με μέση τιμή μηδέν και τυπική απόκλιση μονάδα.

Αντίστοιχα για τις μετασχηματισμένες τιμές των χαρακτηριστικών του συνόλου Αξιολόγησης έχουμε:

```
Input []:
1  print(X_test_scaled[:,0].mean(), X_test_scaled[:,0].std())
2  print(X_test_scaled[:,1].mean(), X_test_scaled[:,1].std())
3  print(X_test_scaled[:,2].mean(), X_test_scaled[:,2].std())
4  print(X_test_scaled[:,3].mean(), X_test_scaled[:,3].std())
5  print(X_test_scaled[:,4].mean(), X_test_scaled[:,4].std())
```

```
Output []:
0.0011735956733567629 0.9980019268585533
0.017140565635991194 0.9989279363059248
0.013388317812892962 1.0008535083704728
0.009043999248804419 1.001821244134054
0.013483866078974295 1.0080841495754993
```

Παρατηρείται ότι και οι νέες τιμές του συνόλου Αξιολόγησης ακολουθούν την Κανονική Κατανομή, αλλά παρουσιάζουν μια μικρή απόκλιση δεκαδικών ψηφίων, και δεν είναι τόσο «ξεκάθαρες» όσο οι μετασχηματισμένες τιμές του συνόλου Εκπαίδευσης, διότι ο μετασχηματιστής StandardScaler() παραμετροποιήθηκε σύμφωνα με τις μέσες τιμές και τις τυπικές αποκλίσεις των χαρακτηριστικών του αρχικού συνόλου Εκπαίδευσης για την εφαρμογή του μαθηματικού τύπου. Το γεγονός ότι βλέπουμε για το μετασχηματισμένο σύνολο Αξιολόγησης μέσες τιμές κοντά στο μηδέν, και τυπικές αποκλίσεις κοντά στη μονάδα σημαίνει ότι τα δεδομένα ακολουθούν παρόμοιο μοτίβο και είναι αντιπροσωπευτικά με αυτά των δεδομένων Εκπαίδευσης.

2.4.4 Ανάπτυξη Μοντέλου

Η αρχιτεκτονική του μοντέλου δεν θα αλλάξει, επομένως τα παρακάτω βήματα είναι ίδια με αυτά του προηγούμενου υποκεφαλαίου 2.3.

Αρχικοποιούμε το νέο μοντέλο «model_2» και εξάγουμε τη συνοπτική εικόνα των επιπέδων και των παραμέτρων:

```
Input []:
1  model_2 = keras.models.Sequential(
2            name='model_2',
3            layers=[
4            keras.Input(shape=(5,)),
5            keras.layers.Dense(10, activation='relu',
6                               kernel_initializer='he_normal',
7                               bias_initializer='zeros'),
8            keras.layers.Dense(10, activation='relu',
9                               kernel_initializer='he_normal',
10                              bias_initializer='zeros'),
11           keras.layers.Dense(10, activation='relu',
12                              kernel_initializer='he_normal',
13                              bias_initializer='zeros'),
14           keras.layers.Dense(3, activation="softmax",
15                              kernel_initializer='glorot_uniform',
16                              bias_initializer='zeros')
```

```
17 ])
18 model_2.summary()

Output []:
Model: "model_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_40 (Dense)            (None, 10)                60

 dense_41 (Dense)            (None, 10)                110

 dense_42 (Dense)            (None, 10)                110

 dense_43 (Dense)            (None, 3)                 33

=================================================================
Total params: 313
Trainable params: 313
Non-trainable params: 0
_________________________________________________________________
```

Παρακάτω γίνεται η σύνταξη του μοντέλου για να ενσωματωθούν η συνάρτηση απώλειας, ο αλγόριθμος βελτιστοποίησης και η μετρική ακρίβειας:

```
Input []:
1  loss = keras.losses.SparseCategoricalCrossentropy()
2  optim = keras.optimizers.Nadam(learning_rate=0.001)
3  metr = keras.metrics.SparseCategoricalAccuracy()
4
5  model_2.compile(loss=loss, optimizer=optim, metrics=metr)
```

Παρακάτω γίνεται η κλήση για να ξεκινήσει η εκπαιδευτική διαδικασία, όπου αυτή τη φορά χρησιμοποιούμε τις μετασχηματισμένες τιμές των χαρακτηριστικών X_train_scaled:

```
Input []:
1  history = model_2.fit(X_train_scaled, y_train, epochs=300,
2                        batch_size=32, validation_split=0.2,
3                        shuffle=True, verbose=1)

Output []:
Epoch 1/300
172/172 [==============================] - 4s 4ms/step - loss: 1.0616 -
sparse_categorical_accuracy: 0.4807 - val_loss: 0.4185 -
val_sparse_categorical_accuracy: 0.8687
Epoch 2/300
```

```
172/172 [==============================] - 0s 3ms/step - loss: 0.2437 -
sparse_categorical_accuracy: 0.9209 - val_loss: 0.1528 -
val_sparse_categorical_accuracy: 0.9519
Epoch 3/300
172/172 [==============================] - 1s 3ms/step - loss: 0.1409 -
sparse_categorical_accuracy: 0.9530 - val_loss: 0.1217 -
val_sparse_categorical_accuracy: 0.9562
Epoch 4/300
.........
Epoch 298/300
172/172 [==============================] - 1s 6ms/step - loss: 0.0374 -
sparse_categorical_accuracy: 0.9860 - val_loss: 0.0618 -
val_sparse_categorical_accuracy: 0.9774
Epoch 299/300
172/172 [==============================] - 1s 4ms/step - loss: 0.0362 -
sparse_categorical_accuracy: 0.9869 - val_loss: 0.0587 -
val_sparse_categorical_accuracy: 0.9825
Epoch 300/300
172/172 [==============================] - 1s 4ms/step - loss: 0.0366 -
sparse_categorical_accuracy: 0.9865 - val_loss: 0.0622 -
val_sparse_categorical_accuracy: 0.9796
```

Επομένως, το νέο μοντέλο παρουσίασε με τη λήξη την εκπαιδευτικής διαδικασίας σφάλμα 0.0366 και ακρίβεια 0.9865, έναντι των 0.1167 σφάλματος και 0.9588 ακρίβειας του προηγούμενου μοντέλου.

Συνεπώς, ήδη από τα αποτελέσματα της εκπαίδευσης βλέπουμε σημαντική μείωση του σφάλματος σε τιμή μικρότερη της πρώτης περίπτωσης, που σημαίνει πλέον ότι οι λάθος προβλέψεις είναι μικρότερης σοβαρότητας, ενώ η συνολική ακρίβεια του μοντέλου για όλες τις περιπτώσεις φαίνεται να βελτιώνεται.

Ωστόσο, καλύτερη οπτική της επίδοσης του νέου εκπαιδευμένου μοντέλου θα υπάρξει κατά της αξιολόγησή του πάνω στα νέα δεδομένα Αξιολόγησης.

Επίσης, με τον παρακάτω κώδικα γίνονται η αποθήκευση του ίδιου του μοντέλου και των παραπάνω τιμών σφάλματος και ακρίβειας, αλλά και φόρτωσή τους:

```
Input []:
1  # save model_2
2  keras.models.save_model(model_2, '/content/drive/MyDrive/
3                  Colab Notebooks/thesis/saved_models/model_2')
4
5  # load model_2
6  model_2 = keras.models.load_model('/content/drive/MyDrive/
7                  Colab Notebooks/thesis/saved_models/model_2')
```

```
8
9  # save history
10 hist_df = pd.DataFrame(history.history)
11 hist_df.to_csv('/content/drive/MyDrive/Colab Notebooks/
12                  thesis/saved_models/model_2.csv', index=False)
13
14 # load history
15 datadf = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
16                       thesis/saved_models/model_2.csv')
17 datadf
```

Output []:

	loss	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	1.061649	0.480671	0.418528	0.868709
1	0.243741	0.920861	0.152782	0.951860
2	0.140868	0.952954	0.121665	0.956236
3	0.121748	0.956601	0.111683	0.958425
4	0.112987	0.959154	0.106210	0.956966
...	...	...	...	...
295	0.036241	0.985777	0.056515	0.979577
296	0.036405	0.986871	0.055650	0.981036
297	0.037432	0.985959	0.061820	0.977389
298	0.036200	0.986871	0.058709	0.982495
299	0.036599	0.986506	0.062179	0.979577

300 rows × 4 columns

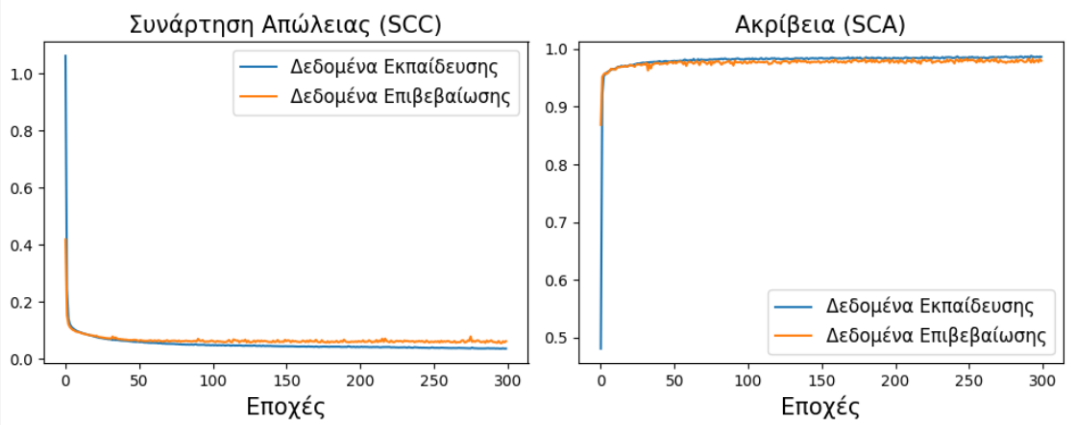
2.4.5 Διαγράμματα Προόδου

Για να ελέγξουμε αν όντως η εκπαιδευτική διαδικασία ήταν επιτυχής, πρέπει να γίνει ο οπτικός έλεγχος μέσω των διαγραμμάτων απώλειας και ακρίβειας, όσον αφορά στην πρόοδο των αντίστοιχων τιμών κατά το πέρασμα των εποχών:

```
Input []:
1  plt.style.use('default')
2
3  fig, axs = plt.subplots(1, 2, figsize=(10,4))
4
5  axs[0].plot(history.history['loss'],
6              label='Δεδομένα Εκπαίδευσης')
7  axs[0].plot(history.history['val_loss'],
8              label='Δεδομένα Επιβεβαίωσης')
```

```
axs[0].set_title('Συνάρτηση Απώλειας (SCC)', fontsize='15')
axs[0].set_xlabel('Εποχές', fontsize='15')
axs[0].legend(fontsize='12')

axs[1].plot(history.history['sparse_categorical_accuracy'],
            label='Δεδομένα Εκπαίδευσης')
axs[1].plot(history.history['val_sparse_categorical_accuracy'],
            label='Δεδομένα Επιβεβαίωσης')
axs[1].set_title('Ακρίβεια (SCA)', fontsize='15')
axs[1].set_xlabel('Εποχές', fontsize='15')
axs[1].legend(fontsize='12')

plt.tight_layout()
```

Output []:



Η ελαχιστοποίηση της συνάρτησης απώλειας και η παράλληλη μεγιστοποίηση της ακρίβειας, τόσο για τα δεδομένα Εκπαίδευσης, όσο και για τα δεδομένα Επιβεβαίωσης, υποδεικνύουν ότι το μοντέλο έχει εκπαιδευτεί επιτυχώς με μεγάλη πιθανότητα ικανοποιητικής γενίκευσης σε νέα δεδομένα.

Αξίζει να σημειωθεί ότι οι τελικές τιμές που καταγράφηκαν για την τελευταία εποχή ήταν 0.0622 και 0.9796 για το σφάλμα και την ακρίβεια των δεδομένων Επιβεβαίωσης.

2.4.6 Αξιολόγηση Μοντέλου

Εφόσον το εκπαιδευμένο μοντέλο παρουσίασε ικανοποιητική συμπεριφορά κατά την εκπαίδευση, θα προχωρήσουμε στο επόμενο βήμα, αυτό της αξιολόγησης με αντίστοιχο τρόπο όπως την προηγούμενη περίπτωση.

Μια πρώτη συμβατική εποπτεία για τις συγκεντρωτικές τιμές σφάλματος και ακρίβειας των πλέον μετασχηματισμένων δεδομένων Αξιολόγησης X_test_scaled παρουσιάζεται παρακάτω:

```
Input []:
1  model_2.evaluate(X_test_scaled, y_test)

Output []:
54/54 [==============================] - 0s 1ms/step - loss: 0.0538 -
sparse_categorical_accuracy: 0.9825
[0.0538439117372036, 0.9824970960617065]
```

Παρατηρούμε ότι από τις προβλέψεις του εκπαιδευμένου μοντέλου πάνω στα δεδομένα Αξιολόγησης προκύπτει το συνολικό σφάλμα 0.0538 και η γενική ακρίβεια 0.9825.

Υπενθυμίζεται ότι στο τελευταίο βήμα της εκπαίδευσης το μοντέλο παρουσίασε σφάλμα 0.0366 και ακρίβεια 0.9865 στα δεδομένα Εκπαίδευσης, και σφάλμα 0.0622 και ακρίβεια 0.9796 στα δεδομένα Επιβεβαίωσης.

Συνεπώς, τα δεδομένα Αξιολόγησης παρουσιάζουν προφανές αυξημένο σφάλμα, και μάλιστα πιο κοντά σε αυτό των δεδομένων Επιβεβαίωσης. Αυτό είναι κάτι αναμενόμενο, καθώς το μοντέλο δεν έχει τροφοδοτηθεί, κατά την εκπαίδευσή του, με αυτά τα δεδομένα, και είναι παραμετροποιημένο για τα δεδομένα Εκπαίδευσης.

Από την άλλη, όσον αφορά στις τιμές της ακρίβειας, αυτή των δεδομένων Αξιολόγησης πλησιάζει περισσότερο σε αυτή των δεδομένων Εκπαίδευσης, παρά σε αυτή των δεδομένων Επιβεβαίωσης, αν αναλογιστούμε ότι το εκπαιδευμένο μοντέλο δεν έχει «δει» ποτέ τα δεδομένα Επιβεβαίωσης και Αξιολόγησης, ούτε έχει εκπαιδευτεί πάνω σε αυτά, παρά μόνο σε παρεμφερή τους.

Σε κάθε περίπτωση, έχουν εξαχθεί σημαντικά βελτιωμένα αποτελέσματα, σε σχέση με αυτά του προηγούμενου μοντέλου.

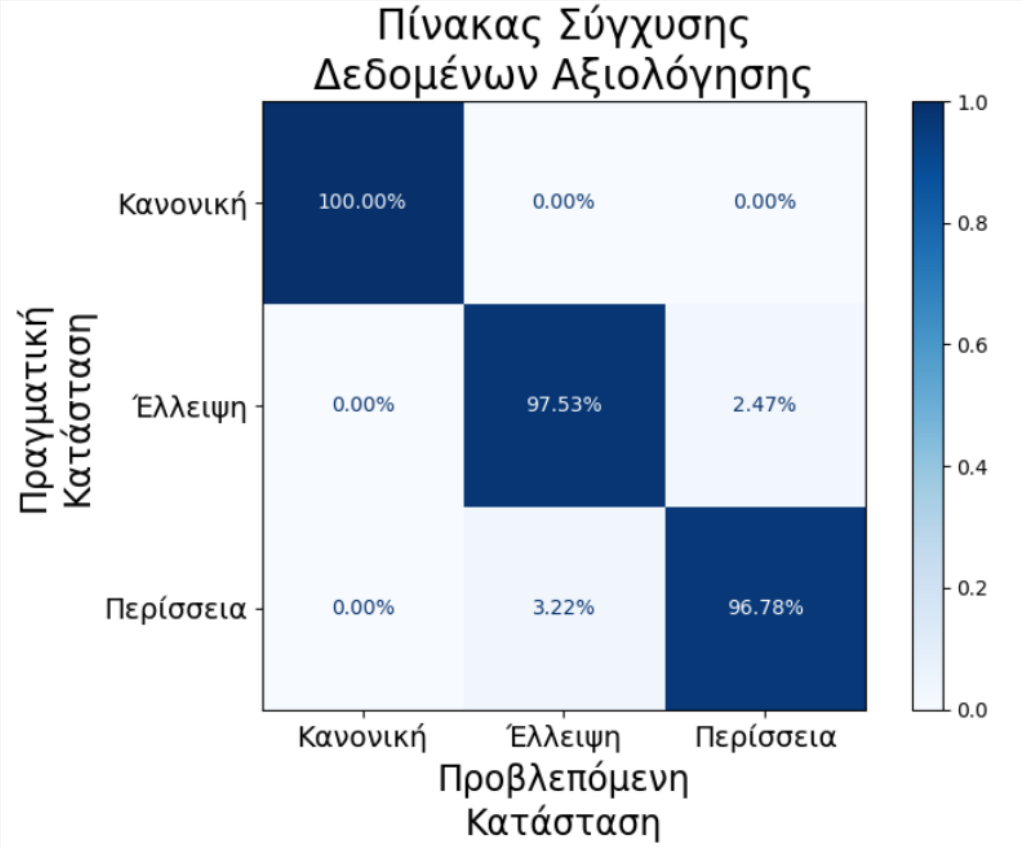
Περισσότερες πληροφορίες για την ακρίβεια των προβλέψεων και την γενικότερη επίδοση του εκπαιδευμένου μοντέλου, παίρνουμε μέσω του Πίνακα Σύγχυσης, αλλά και των παραγόμενων μεγεθών του, συγκεκριμένα των precision, recall και f1-score:

```
Input []:
1  y_test_pred = model_2.predict(X_test_scaled)
2  y_test_pred = np.argmax(y_test_pred, axis=1)
3
4  fig, ax = plt.subplots(figsize=(8, 6))
5  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
6  ConfusionMatrixDisplay.from_predictions(
```

```
                  y_test, y_test_pred, display_labels=labels,
                  normalize='true', values_format='.2%',
                  include_values=True, cmap='Blues',
                  xticks_rotation='horizontal',
                  ax=ax, colorbar=True)
_ = ax.set_title('Πίνακας Σύγχυσης\n
                     Δεδομένων Αξιολόγησης', fontsize='20')
plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
plt.tick_params(labelsize='14')
plt.tight_layout()
plt.show()

targets = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
cr_test = classification_report(y_test, y_test_pred,
  target_names=targets)
print(cr_test)
```

Output []:

```
54/54 [==============================] - 0s 2ms/step
```




```
              precision    recall  f1-score   support

    Κανονική       1.00      1.00      1.00       659
```

```
     Έλλειψη       0.97      0.98      0.97       527
   Περίσσεια       0.98      0.97      0.97       528

    accuracy                           0.98      1714
   macro avg       0.98      0.98      0.98      1714
weighted avg       0.98      0.98      0.98      1714
```

Είναι σημαντικό να αναφερθεί ότι, το πρώτο βήμα της απόκτησης όλων των προβλέψεων είναι αναγκαίο, καθώς απαιτείται για την εξαγωγή του Πίνακα Σύγχυσης και των επιπλέον μέτρων αξιολόγησης.

Η πρώτη σημαντική παρατήρηση είναι πως υπάρχει αύξηση σε όλα τα ποσοστά μεταξύ του Πίνακα Σύγχυσης του προηγούμενου μοντέλου και αυτού που εξετάζεται εδώ. Ειδικότερα, για την κύρια διαγώνιο, η οποία εκφράζει την επιτυχημένη πρόβλεψη καθεμίας κατάσταση είναι 100%, 97.53% και 96.78% σε αντίθεση με τα 100%, 92.60% και 93.18%. Αντίστοιχα, τα ποσοστά λάθους μεταξύ των δυο επιβλαβών καταστάσεων είναι 2.47% και 3.22%, έναντι των 5.50% και 6.82%, τα οποία βλέπουμε ότι σχεδόν υποδιπλασιάστηκαν στην περίπτωση του νέου μοντέλου.

Η δεύτερη σημαντική παρατήρηση είναι η πλήρης απουσία σφαλμάτων Τύπου Ι, δηλαδή μια πραγματικά επιβλαβής κατάσταση να αναγνωριστεί ως φυσιολογική και να παραβλεφθεί από το μοντέλο. Αυτό βέβαια δεν σημαίνει ότι το συγκεκριμένο μοντέλο δεν θα μπερδέψει ποτέ μια επιβλαβή κατάσταση με μια κανονική, αλλά το ότι υπάρχουν πολύ μικρότερες πιθανότητες στο να συμβεί κάτι τέτοιο, σε σχέση με το αρχικό μοντέλο, όπου η αντίστοιχη πιθανότητα σφάλματος ήταν στο 1.90%.

Τα επιπλέον μέτρα αξιολόγησης precision, recall και f1-score παρουσιάζουν επίσης βελτίωση, και μάλιστα η τιμή του precision, του οποίου η σημασία για το πρόβλημά μας αναλύθηκε στο υποκεφάλαιο 2.3.13, είναι το μέγιστο δυνατό που θα μπορούσαμε να έχουμε.

Τελικά, βλέπουμε τη σημασία της αλλαγής κλίμακας των δεδομένων και την αναγκαιότητα να υπάρχει ως καθιερωμένο βήμα σε μια διαδικασία ανάπτυξης και εκπαίδευσης μοντέλων Βαθιάς Μάθησης. Οι μέθοδοι μετασχηματισμών ποικίλουν, δεν υπάρχουν αυστηροί κανόνες για την επιλογή της καθεμίας, αλλά επιλέγεται εκείνη που κρίνεται καταλληλότερη για το εκάστοτε πρόβλημα, και που επιτρέπει μια όσο το δυνατόν ομαλότερη εκπαιδευτική διαδικασία και βέλτιστες δυνατότητες πρόβλεψης.

2.5 Τρίτο Μοντέλο Νευρωνικού Δικτύου

Κατά το δεύτερο μοντέλο MLP είδαμε σημαντική βελτίωση τόσο στην τιμή του σφάλματος, όσο και στην τιμή της ακρίβειας, και η καλύτερη επίδοση του μοντέλου επιβεβαιώθηκε από την αξιολόγησή του μέσω του Πίνακα Σύγχυσης.

Ωστόσο, πρέπει να αναλογιστούμε ότι τα δεδομένα που συλλέχθηκαν αποτελούν μια χρονοσειρά της λειτουργικής κατάστασης ενός εξαρτήματος σε διάστημα λίγων ημερών. Επομένως, υπάρχει μεγάλη πιθανότητα να υπάρχουν εξαρτήσεις μεταξύ διαδοχικών μετρήσεων. Για παράδειγμα, μια μεγάλη τιμή Crest, ακολουθούμενη από μια ακόμα μεγαλύτερη, θα μπορούσε να σημαίνει ότι η κατάσταση του εξαρτήματος εκείνη τη στιγμή έπαιξε σημαντικό ρόλο, ή ήταν προάγγελος, για την επόμενη κατάσταση μεγαλύτερης τιμής Crest. Το παραπάνω παράδειγμα αναφέρεται μόνο σε δυο διαδοχικές τιμές, αλλά μια χρονοσειρά μπορεί να κρύβει πολλά μοτίβα παρόμοιας εξάρτησης και σε βάθος χρόνου.

Αναφέρθηκε ότι η αρχιτεκτονική των νευρωνικών δικτύων MLP είναι ιδιαίτερα χρήσιμη στην επεξεργασία και την ανάλυση δεδομένων πίνακα (tabular data) και είναι ικανή να βρει μαθηματικούς και στατιστικούς κανόνες για όλα τα δεδομένα του. Όμως, υστερεί στην διαχείριση ακολουθιών, όπως είναι οι χρονοσειρές, και στην αναγνώριση χρονικών εξαρτήσεων, και για τον λόγο αυτό προτιμώνται επαναληπτικά νευρωνικά δίκτυα όπως τα RNN, LSTM και GRU.

Στην περίπτωσή μας, διατηρώντας την αρχιτεκτονική MLP, ένας τρόπος για να γνωστοποιηθεί η χρονική εξάρτηση στο μοντέλο είναι μέσω της Μηχανικής Χαρακτηριστικών (Feature Engineering).

Ειδικότερα, υπενθυμίζεται ότι ο χρόνος μεταξύ δυο καταγραφών, δηλαδή δυο γραμμών του συνόλου δεδομένων, ήταν δυο λεπτά. Συνεπώς, τρεις γραμμές του πίνακα αντιστοιχούν σε ένα χρονικό παράθυρο τεσσάρων λεπτών. Με τη γνώση αυτή, ο αρχικός πίνακας δεδομένων των πέντε χαρακτηριστικών (χωρίς τη συμπερίληψη της στήλης των ετικετών Bearing State), μπορεί να μετασχηματιστεί σε έναν νέο πίνακα δεκαπέντε χαρακτηριστικών, όπου κάθε γραμμή θα αποτελείται από κάθε τρεις διαδοχικές γραμμές του αρχικού πίνακα και θα περιγράφει τις μετρήσεις τεσσάρων λεπτών.

Προφανώς, ο τελικός τροποποιημένος πίνακας θα είναι κατά 6 εγγραφές μικρότερος, διότι από κάθε επιμέρους πίνακα δεδομένων των ξεχωριστών καταστάσεων, οι δυο τελευταίες γραμμές θα πρέπει να ομαδοποιηθούν με την τρίτη τελευταία, για να σχηματίσουν τελικά την τελευταία εγγραφή του νέου μετασχηματισμένου πίνακα των 15 χαρακτηριστικών.

Ειδικότερα, ο αρχικός πίνακας δεδομένων είναι ο παρακάτω:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	1
...	...	...	...	...	...	...
8564	0.000728	0.880769	7.337692	8.304615	26.866153	3
8565	0.000703	0.853846	6.926154	8.102308	26.806153	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3
8568	0.000708	0.866154	7.378462	8.506923	26.799999	3

8569 rows × 6 columns

Εικόνα 55. Αρχικός Πίνακας Δεδομένων Καταστάσεων Ρουλεμάν Πρώτης Εφαρμογής

Ενώ, ο μετασχηματισμένος πίνακας δεδομένων που θα χρησιμοποιηθεί στο συγκεκριμένο παράδειγμα είναι ο παρακάτω:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	V-RMS_2	a-RMS_2	a-Peak_2	Crest_2	Temperature_2	V-RMS_3	a-RMS_3	a-Peak_3	Crest_3	Temperature_3	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	0.001095	0.713077	3.244615	4.716923	26.299999	0.001079	0.697143	3.227857	4.795000	26.299999	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	0.001079	0.697143	3.227857	4.795000	26.299999	0.001080	0.700000	3.213077	4.793077	26.208462	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	0.001080	0.700000	3.213077	4.793077	26.208462	0.001088	0.700000	3.241538	4.783077	26.200001	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	0.001088	0.700000	3.241538	4.783077	26.200001	0.001085	0.700000	3.200769	4.737692	26.099231	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	0.001085	0.700000	3.200769	4.737692	26.099231	0.001079	0.696154	3.204615	4.761538	25.963846	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8558	0.000719	0.863846	7.023846	8.129231	26.799999	0.000726	0.855385	6.603077	7.735385	26.845384	0.000728	0.880769	7.337692	8.304615	26.866153	3
8559	0.000726	0.855385	6.603077	7.735385	26.845384	0.000728	0.880769	7.337692	8.304615	26.866153	0.000703	0.853846	6.926154	8.102308	26.806153	3
8560	0.000728	0.880769	7.337692	8.304615	26.866153	0.000703	0.853846	6.926154	8.102308	26.806153	0.000705	0.834615	6.556923	7.822308	26.892307	3
8561	0.000703	0.853846	6.926154	8.102308	26.806153	0.000705	0.834615	6.556923	7.822308	26.892307	0.000713	0.862308	6.893846	7.989231	26.799999	3
8562	0.000705	0.834615	6.556923	7.822308	26.892307	0.000713	0.862308	6.893846	7.989231	26.799999	0.000708	0.866154	7.378462	8.506923	26.799999	3

8563 rows × 16 columns

Εικόνα 56. Μετασχηματισμένος Πίνακας Δεδομένων Καταστάσεων Ρουλεμάν Πρώτης Εφαρμογής

Η διαδικασία και ο κώδικας μετασχηματισμού του αρχικού πίνακα (data) των 5 στηλών, στον τελικό πίνακα (data_shifted) των 15 στηλών, χωρίς την στήλη των ετικετών, παρουσιάζεται αναλυτικά στο Παράρτημα Η’.

2.5.1 Προεπεξεργασία

Με ανάλογο τρόπο όπως στις προηγούμενες περιπτώσεις, εισάγονται τα απαραίτητα πακέτα της Python και τα νέα μετασχηματισμένα δεδομένα, τα οποία στη συνέχεια διαιρούνται σε σύνολα Εκπαίδευσης και Αξιολόγησης, και μετασχηματίζονται περαιτέρω σύμφωνα με τη μέθοδο της Κανονικοποίησης για την αλλαγή κλίμακας:

```
Input []:
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4  from tensorflow import keras
5  from sklearn.model_selection import train_test_split
6  from sklearn.preprocessing import StandardScaler
7  from sklearn.metrics import ConfusionMatrixDisplay,
8                              classification_report
9
10 data = pd.read_csv('/content/drive/MyDrive/
11                      Colab Notebooks/thesis/data/
12                      bearing_vibration_metrics_shifted.csv')
13 data['Bearing State'] = data['Bearing State'] - 1
14
15 X_train, X_test, y_train, y_test = train_test_split(
16                                     data.iloc[:,:-1],
17                                     data['Bearing State'],
18                                     test_size = 0.2,
19                                     shuffle=True,
20                                     stratify=data['Bearing State'])
21
22 X_train = np.asarray(X_train)
23 X_test = np.asarray(X_test)
24 y_train = np.asarray(y_train)
25 y_test = np.asarray(y_test)
26
27 scaler = StandardScaler()
28 scaler.fit(X_train)
29 X_train_scaled = scaler.transform(X_train)
30 X_test_scaled = scaler.transform(X_test)
```

2.5.2 Ανάπτυξη Μοντέλου

Αρχικοποιούμε το νέο μοντέλο «model_3» και εξάγουμε τη συνοπτική εικόνα των επιπέδων και των παραμέτρων:

```
Input []:
1  model_3 = keras.models.Sequential(
2            name='model_3',
3            layers=[
4            keras.Input(shape=(15,)),
5            keras.layers.Dense(30, activation='relu',
6                               kernel_initializer='he_normal',
7                               bias_initializer='zeros'),
8            keras.layers.Dense(30, activation='relu',
9                               kernel_initializer='he_normal',
10                              bias_initializer='zeros'),
11           keras.layers.Dense(30, activation='relu',
12                              kernel_initializer='he_normal',
13                              bias_initializer='zeros'),
14           keras.layers.Dense(3, activation="softmax",
15                              kernel_initializer='glorot_uniform',
16                              bias_initializer='zeros')
17 ])
18 model_3.summary()

Output []:
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 30)                480

 dense_1 (Dense)             (None, 30)                930

 dense_2 (Dense)             (None, 30)                930

 dense_3 (Dense)             (None, 3)                 93

=================================================================
Total params: 2,433
Trainable params: 2,433
Non-trainable params: 0
_________________________________________________________________
```

Κάποιες αλλαγές που έγιναν κατά την αρχικοποίηση του μοντέλου είναι η αύξηση των νευρώνων του επιπέδου εισόδου από 5 σε 15, εφόσον τα νέα δεδομένα έχουν 15 μεταβλητές-στήλες σε κάθε γραμμή, αλλά και η αύξηση των νευρώνων των ενδιάμεσων επιπέδων από 10 σε 30, διότι με την αύξηση των χαρακτηριστικών απαιτείται και αύξηση των υπολογιστικών νευρώνων για καλύτερη ανάλυση των δεδομένων. Αυτό οδήγησε το

συνολικό πλήθος παραμέτρων βαρών και σταθερών όρων να αυξηθεί σε 2.433, από 313 των προηγούμενων συμβατικών μοντέλων.

Παρακάτω γίνεται η σύνταξη του μοντέλου για να ενσωματωθούν η συνάρτηση απώλειας, ο αλγόριθμος βελτιστοποίησης και η μετρική ακρίβειας:

```
Input []:
1  loss = keras.losses.SparseCategoricalCrossentropy()
2  optim = keras.optimizers.Nadam(learning_rate=0.001)
3  metr = keras.metrics.SparseCategoricalAccuracy()
4
5  model_3.compile(loss=loss, optimizer=optim, metrics=metr)
```

Παρακάτω γίνεται η κλήση για να ξεκινήσει η εκπαιδευτική διαδικασία:

```
Input []:
1  history = model_3.fit(X_train_scaled, y_train, epochs=300,
2                        batch_size=32, validation_split=0.2,
3                        shuffle=True, verbose=1)

Output []:
Epoch 1/300
172/172 [==============================] - 2s 4ms/step - loss: 0.1923 -
sparse_categorical_accuracy: 0.9325 - val_loss: 0.0940 -
val_sparse_categorical_accuracy: 0.9620
Epoch 2/300
172/172 [==============================] - 0s 3ms/step - loss: 0.0884 -
sparse_categorical_accuracy: 0.9673 - val_loss: 0.0800 -
val_sparse_categorical_accuracy: 0.9723
Epoch 3/300
172/172 [==============================] - 0s 3ms/step - loss: 0.0770 -
sparse_categorical_accuracy: 0.9719 - val_loss: 0.0681 -
val_sparse_categorical_accuracy: 0.9745
Epoch 4/300
.........
Epoch 298/300
172/172 [==============================] - 1s 5ms/step - loss: 1.9436e-04 -
sparse_categorical_accuracy: 1.0000 - val_loss: 0.0462 -
val_sparse_categorical_accuracy: 0.9905
Epoch 299/300
172/172 [==============================] - 1s 4ms/step - loss: 2.2393e-04 -
sparse_categorical_accuracy: 1.0000 - val_loss: 0.0549 -
val_sparse_categorical_accuracy: 0.9920
Epoch 300/300
```

```
172/172 [==============================] - 1s 5ms/step - loss: 3.9491e-04 -
sparse_categorical_accuracy: 1.0000 - val_loss: 0.0571 -
val_sparse_categorical_accuracy: 0.9912
```

Με μια πρώτη ματιά βλέπουμε ότι το σφάλμα για τα δεδομένα Εκπαίδευσης είναι 0.00039491, δηλαδή πάρα πολύ κοντά στο μηδέν, ενώ η ακρίβεια έχει τιμή 1, που σημαίνει 100% επιτυχία στο να προβλέπει το μοντέλο σωστά κάθε περίπτωση του συνόλου Εκπαίδευσης. Οι αντίστοιχες τιμές σφάλματος και ακρίβειας για το σύνολο Επιβεβαίωσης, όπως υπολογίστηκαν στο τέλος της εκπαιδευτικής διαδικασίας είναι 0.0571 και 0.9912.

Συνεπώς, παρατηρούμε μια πολύ μεγάλη διαφορά στα αποτελέσματα μεταξύ των δεδομένων Εκπαίδευσης και των θεωρητικά «άγνωστων» δεδομένων Επιβεβαίωσης, ασχέτως του ότι η ακρίβεια υποδηλώνει μια γενικότερη επιτυχία προβλέψεων. Η μεγάλη ανησυχία προκαλείται από το γεγονός ότι στα δεδομένα Εκπαίδευσης οι τιμές του σφάλματος και ακρίβειας είναι σχεδόν τέλειες, και μάλιστα, με το σφάλμα 145 φορές μικρότερο από αυτό των δεδομένων επιβεβαίωσης, ενώ γενικότερα θα θέλαμε οι τιμές να προοδεύουν σχετικά παράλληλα.

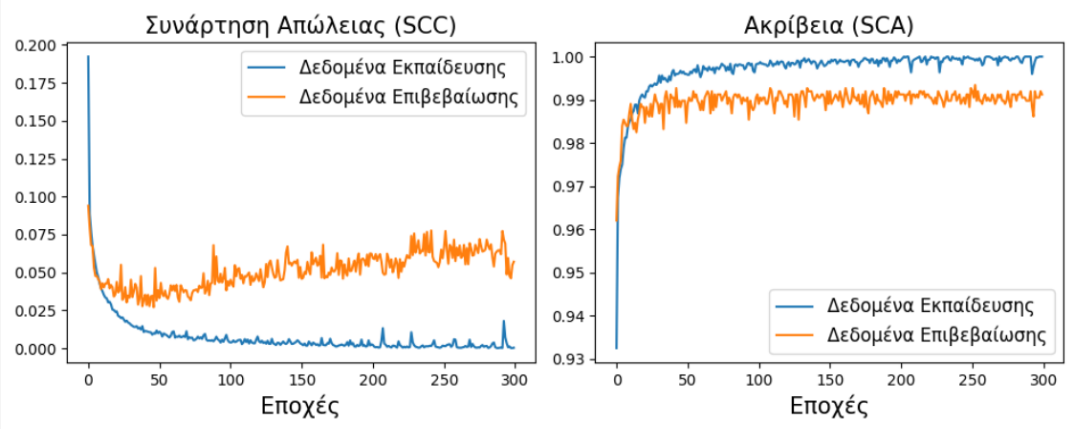
Επομένως, για μια καλύτερη εικόνα της προόδου της εκπαιδευτικής διαδικασίας του μοντέλου, πρέπει να οπτικοποιήσουμε τις παραπάνω τιμές με διαγράμματα προόδου της συνάρτησης απώλειας και της ακρίβειας:

Input []:

```
1  plt.style.use('default')
2
3  fig, axs = plt.subplots(1, 2, figsize=(10,4))
4
5  axs[0].plot(history.history['loss'],
6              label='Δεδομένα Εκπαίδευσης')
7  axs[0].plot(history.history['val_loss'],
8              label='Δεδομένα Επιβεβαίωσης')
9  axs[0].set_title('Συνάρτηση Απώλειας (SCC)', fontsize='15')
10 axs[0].set_xlabel('Εποχές', fontsize='15')
11 axs[0].legend(fontsize='12')
12
13 axs[1].plot(history.history['sparse_categorical_accuracy'],
14             label='Δεδομένα Εκπαίδευσης')
15 axs[1].plot(history.history['val_sparse_categorical_accuracy'],
16             label='Δεδομένα Επιβεβαίωσης')
17 axs[1].set_title('Ακρίβεια (SCA)', fontsize='15')
18 axs[1].set_xlabel('Εποχές', fontsize='15')
19 axs[1].legend(fontsize='12')
20
```

```
21 plt.tight_layout()
```

Output []:



Επομένως, παρατηρούμε ότι, όσο η τιμή της συνάρτησης απώλειας για τα δεδομένα Εκπαίδευσης ελαχιστοποιείται, τόσο η αντίστοιχη τιμή για τα δεδομένα Επιβεβαίωσης τείνει να μεγιστοποιηθεί, και μάλιστα αν συνεχιζόταν η εκπαίδευση για περισσότερες εποχές, πολύ πιθανόν το σφάλμα Επιβεβαίωσης να μεγάλωνε ακόμη περισσότερο.

Μια τέτοια απόκλιση των γραμμών σφάλματος είναι ενδεικτικό του ότι το μοντέλο έχει υπερπροσαρμόσει (overfit) τις παραμέτρους των επιπέδων του έτσι ώστε να αναγνωρίζει τέλεια τα μοτίβα των δεδομένων Εκπαίδευσης και μόνον αυτά. Συνεπώς, κάθε προσπάθεια πρόβλεψης από το μοντέλο πάνω σε νέα δεδομένα, και μάλιστα παρόμοιας συμπεριφοράς με αυτή των δεδομένων Εκπαίδευσης, έχει μεγάλη πιθανότητα να είναι λανθασμένη.

Η αντιμετώπιση του προβλήματος της υπερπροσαρμογής (overfitting) του μοντέλου πάνω στο σύνολο των δεδομένων με τα οποία εκπαιδεύεται, γίνεται με πληθώρα τεχνικών κανονικοποίησης (regularization), αλλά και με αλλαγές στην αρχιτεκτονική του ίδιου του μοντέλου ως προς το πλήθος και τον τύπο των επιπέδων, τον αριθμό των νευρώνων, τη συνάρτηση απώλειας, τον αλγόριθμο βελτιστοποίησης, την αρχικοποίηση των παραμέτρων, τις συναρτήσεις ενεργοποίησης κ.α.

Για παράδειγμα, ένας τρόπος για να αποφευχθεί η περίπτωση της υπερπροσαρμογής, είναι να σταματήσει η διαδικασία εκπαίδευσης σε κάποιο σημείο πριν από αυτό όπου αρχίζει να παρατηρείται η απόκλιση των γραμμών της συνάρτησης απώλειας. Η τεχνική του παραπάνω παραδείγματος είναι γνωστή ως «πρόωρη διακοπή» (early stopping), ενώ άλλες γνωστές τεχνικές είναι η κανονικοποίηση σφάλματος με βάσει των παραμέτρων L1 και L2 (L1 regularization και L2 regularization ή και συνδυασμός των δύο), η παράλειψη νευρώνων (dropout), η Κανονικοποίηση των παρτίδων (Batch Normalization) κατά τη διάδοσή τους στα επίπεδα του μοντέλου κατά την εκπαίδευση, κ.α.

Για ακόμη μια φορά, δεν υπάρχουν αυστηροί κανόνες για την ανάπτυξη ενός μοντέλου, μόνο καθοδηγητικές τεχνικές, οι οποίες τελικά πρέπει να προσαρμοστούν από τον ίδιο τον χρήστη έτσι ώστε να εκπαιδευτεί ένα μοντέλο κατάλληλα για το πρόβλημα που πρέπει να αντιμετωπιστεί.

2.5.3 Προσαρμογή Μοντέλου

Η αρχικοποίηση του τελικού μοντέλου «model_4», όπως τελικά προσαρμόστηκε, και η εξαγωγή της συνοπτικής εικόνας των επιπέδων και των παραμέτρων του φαίνεται παρακάτω:

```
Input []:
1  model_4 = keras.models.Sequential(
2            name='model_4',
3            layers=[
4            keras.Input(shape=(15,)),
5            keras.layers.BatchNormalization(),
6            keras.layers.Dense(30, activation='relu',
7                               kernel_initializer='he_normal',
8                               bias_initializer='zeros'),
9            keras.layers.BatchNormalization(),
10           keras.layers.Dense(30, activation='relu',
11                              kernel_initializer='he_normal',
12                              bias_initializer='zeros'),
13           keras.layers.BatchNormalization(),
14           keras.layers.Dense(30, activation='relu',
15                              kernel_initializer='he_normal',
16                              bias_initializer='zeros'),
17           keras.layers.BatchNormalization(),
18           keras.layers.Dense(3, activation="softmax",
19                              kernel_initializer='glorot_uniform',
20                              bias_initializer='zeros')
21 ])
22 model_4.summary()

Output []:
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 batch_normalization (BatchN  (None, 15)               60
 ormalization)
```

```
 dense_4 (Dense)             (None, 30)                480

 batch_normalization_1 (Batc  (None, 30)               120
 hNormalization)

 dense_5 (Dense)             (None, 30)                930

 batch_normalization_2 (Batc  (None, 30)               120
 hNormalization)

 dense_6 (Dense)             (None, 30)                930

 batch_normalization_3 (Batc  (None, 30)               120
 hNormalization)

 dense_7 (Dense)             (None, 3)                 93

=================================================================
Total params: 2,853
Trainable params: 2,643
Non-trainable params: 210
_________________________________________________________________
```

Συνεπώς, για την ανάπτυξη του νέου μοντέλου «model_4», ενσωματώθηκαν στο προηγούμενο μοντέλο «model_3» τα επίπεδα BatchNormalization() πριν από κάθε επίπεδο Dense().

Η λειτουργικότητα του επιπέδου BatchNormalization() έγκειται στο γεγονός ότι εφαρμόζει επιπλέον μετασχηματισμό σε κάθε παρτίδα στοιχείων (στην περίπτωσή μας είναι 32) που περνάει σε κάθε βήμα από το μοντέλο κατά την εκπαιδευτική διαδικασία. Ειδικότερα, κάθε παρτίδα, προτού περάσει σε κάποιο επόμενο επίπεδο Dense(), μετασχηματίζεται και πάλι με τη μέθοδο της Κανονικοποίησης, έτσι ώστε όλα τα στοιχεία της να έχουν μέσο όρο μηδέν και τυπική απόκλιση τη μονάδα. Επιπλέον, καθ' όλη τη διαδικασία της εκπαίδευσης αποθηκεύονται οι κινητοί μέσοι όροι και τυπικές αποκλίσεις από όλες τις παρτίδες που τροφοδοτούνται στο μοντέλο, έτσι ώστε κατά τη διαδικασία των προβλέψεων να χρησιμοποιούνται για την Κανονικοποίηση των νέων δεδομένων όταν περνάνε από το εκπαιδευμένο μοντέλο για την εξαγωγή συμπερασμάτων.

Παρακάτω γίνεται η σύνταξη του μοντέλου για να ενσωματωθούν η συνάρτηση απώλειας, ο αλγόριθμος βελτιστοποίησης και η μετρική ακρίβειας, και γίνεται η κλήση για να ξεκινήσει η εκπαιδευτική διαδικασία:

```
Input []:
```

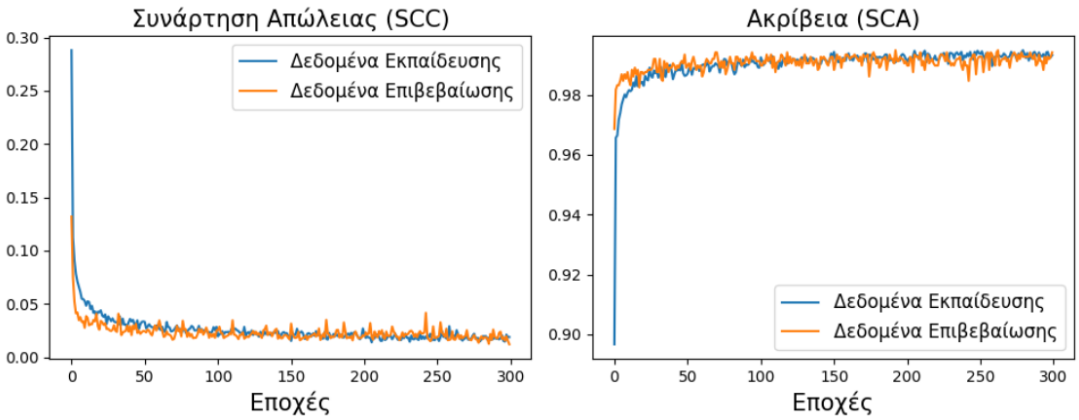
```
loss = keras.losses.SparseCategoricalCrossentropy()
optim = keras.optimizers.Nadam(learning_rate=0.001)
metr = keras.metrics.SparseCategoricalAccuracy()
model_4.compile(loss=loss, optimizer=optim, metrics=metr)

history = model_4.fit(X_train_scaled, y_train, epochs=300,
                      batch_size=32, validation_split=0.2,
                      shuffle=True, verbose=1)
```

Output []:

```
Epoch 1/300
172/172 [==============================] - 4s 5ms/step - loss: 0.2879 -
sparse_categorical_accuracy: 0.8967 - val_loss: 0.1318 -
val_sparse_categorical_accuracy: 0.9686
Epoch 2/300
172/172 [==============================] - 1s 4ms/step - loss: 0.1123 -
sparse_categorical_accuracy: 0.9657 - val_loss: 0.0746 -
val_sparse_categorical_accuracy: 0.9818
Epoch 3/300
172/172 [==============================] - 1s 4ms/step - loss: 0.0905 -
sparse_categorical_accuracy: 0.9664 - val_loss: 0.0522 -
val_sparse_categorical_accuracy: 0.9832
Epoch 4/300
.........
Epoch 298/300
172/172 [==============================] - 1s 5ms/step - loss: 0.0211 -
sparse_categorical_accuracy: 0.9923 - val_loss: 0.0195 -
val_sparse_categorical_accuracy: 0.9920
Epoch 299/300
172/172 [==============================] - 1s 4ms/step - loss: 0.0188 -
sparse_categorical_accuracy: 0.9925 - val_loss: 0.0154 -
val_sparse_categorical_accuracy: 0.9927
Epoch 300/300
172/172 [==============================] - 1s 5ms/step - loss: 0.0188 -
sparse_categorical_accuracy: 0.9932 - val_loss: 0.0121 -
val_sparse_categorical_accuracy: 0.9942
```

Παρατηρούμε ότι οι τιμές σφάλματος και ακρίβειας, για τα σύνολα δεδομένων Εκπαίδευσης και Αξιολόγησης είναι αντίστοιχα 0.0188, 0.9932 και 0.0121, 0.9942.

Η διαφορά με τα αποτελέσματα του προηγούμενου μοντέλου είναι εμφανής, και για περαιτέρω επιβεβαίωση της προόδου έχουμε τα παρακάτω διαγράμματα:



Εικόνα 57. Διαγράμματα Προόδου Σφάλματος και Ακρίβειας Τελικού Μοντέλου Πρώτης Εφαρμογής

Επομένως, είναι προφανής η ομαλοποίηση της εκπαιδευτικής διαδικασίας μετά την προσαρμογή του μοντέλου. Η συνάρτηση απώλειας για τα δεδομένα Επιβεβαίωσης ελαχιστοποιείται παράλληλα με αυτήν των δεδομένων Εκπαίδευσης, ενώ η ακρίβεια και για τα δυο σύνολα μεγιστοποιείται με παρόμοιο τρόπο.

Η αποθήκευση του μοντέλου και των αντίστοιχων τιμών σφάλματος και ακρίβειας κατά τη διαδικασία εκπαίδευσης παρουσιάζεται παρακάτω:

```
Input []:
1  # save model_4
2  keras.models.save_model(model_4, '/content/drive/MyDrive/
3                  Colab Notebooks/thesis/saved_models/model_4')
4
5  # load model_4
6  model_4 = keras.models.load_model('/content/drive/MyDrive/
7                  Colab Notebooks/thesis/saved_models/model_4')
8
9  # save history
10 hist_df = pd.DataFrame(history.history)
11 hist_df.to_csv('/content/drive/MyDrive/Colab Notebooks/
12                  thesis/saved_models/model_4.csv', index=False)
13
14 # load history
15 datadf = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
16                       thesis/saved_models/model_4.csv')
17 datadf

Output []:
```

	loss	sparse_categorical_accuracy	val_loss	val_sparse_categorical_accuracy
0	0.287933	0.896715	0.131824	0.968613
1	0.112268	0.965693	0.074555	0.981752
2	0.090518	0.966423	0.052217	0.983212
3	0.078083	0.971715	0.041192	0.983212
4	0.070965	0.973905	0.041883	0.984672
...	...	...	...	...
295	0.015264	0.994161	0.018674	0.990511
296	0.014565	0.994343	0.015424	0.992701
297	0.021069	0.992336	0.019510	0.991971
298	0.018812	0.992518	0.015446	0.992701
299	0.018790	0.993248	0.012064	0.994161

300 rows × 4 columns

2.5.4 Αξιολόγηση του Προσαρμοσμένου Μοντέλου

Από τη στιγμή που η συμπεριφορά μάθησης του μοντέλου ήταν η επιθυμητή, μπορούμε να προχωρήσουμε στη διαδικασία της αξιολόγησης του νέου εκπαιδευμένου μοντέλου πάνω στα δεδομένα του συνόλου Αξιολόγησης, με ανάλογο τρόπο όπως τις προηγούμενες περιπτώσεις.

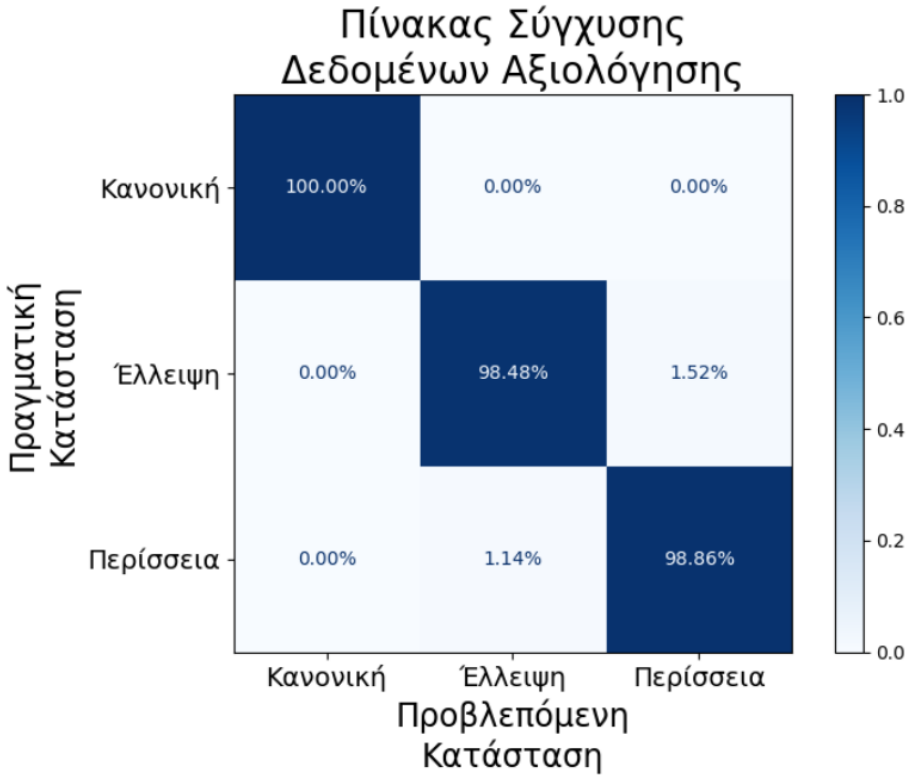
Ο πρώτος συμβατικός έλεγχος για τις γενικές τιμές σφάλματος και ακρίβειας του συνόλου Αξιολόγησης παρουσιάζεται παρακάτω:

```
Input []:
1  model_4.evaluate(X_test_scaled, y_test)

Output []:
54/54 [==============================] - 0s 2ms/step - loss: 0.0258 -
sparse_categorical_accuracy: 0.9918
[0.02575114369392395, 0.9918271899223328]
```

Στα εξαγόμενα αποτελέσματα βλέπουμε τιμή σφάλματος 0.0258 και τιμή ακρίβειας 0.9918. Υπενθυμίζεται ότι, οι τιμές σφάλματος και ακρίβειας στο σύνολο Αξιολόγησης για το προηγούμενο μοντέλο «model_2», όπου έγινε για πρώτη φορά η Κανονικοποίηση των δεδομένων σύμφωνα με τη μέθοδο StandardScaler(), ήταν αντίστοιχα 0.0538 και 0.9825, ενώ για το πρώτο συμβατικό μοντέλο «model_1» ήταν 0.1258 και 0.9562.

Συνεπώς, είναι προφανής η κλιμάκωση βελτίωσης του μοντέλου καθ' όλα τα στάδια ανάπτυξής του, με την τελική εκδοχή του να παρουσιάζει σαφώς τις βέλτιστες τιμές σφάλματος και ακρίβειας.

Παρακάτω παρουσιάζεται και ο αντίστοιχος Πίνακας Σύγχυσης, μαζί με τα παράγωγα μέτρα αξιολόγησης precision, recall και f1-score:



Εικόνα 58. Πίνακας Σύγχυσης Τελικού Μοντέλου Πρώτης Εφαρμογής

	precision	recall	f1-score	support
Κανονική	1.00	1.00	1.00	659
Έλλειψη	0.99	0.98	0.99	527
Περίσσεια	0.98	0.99	0.99	527
accuracy			0.99	1713
macro avg	0.99	0.99	0.99	1713
weighted avg	0.99	0.99	0.99	1713

Εικόνα 59. Μέτρα Αξιολόγησης Τελικού Μοντέλου Πρώτης Εφαρμογής

Επομένως, επιβεβαιώνεται περαιτέρω η τελική αρχιτεκτονική, παραμετροποίηση και εκπαίδευση του μοντέλου ως η βέλτιστη από όλες τις περιπτώσεις που εξετάστηκαν.

ΠΑΡΑΤΗΡΗΣΗ

Σε μια διαδικασία ανάπτυξης ενός μοντέλου Βαθιάς Μάθησης, μόλις αποφασιστεί η τελική εκδοχή του μοντέλου που θα χρησιμοποιηθεί, πρέπει το ίδιο το μοντέλο να αρχικοποιηθεί και να συνταχθεί εκ νέου, για να εκπαιδευτεί μια τελευταία φορά πάνω σε όλα τα διαθέσιμα δεδομένα. Αυτό συμβαίνει γιατί έγινε διαίρεση του αρχικού πίνακα δεδομένων με σκοπό την εκπαίδευση και αξιολόγησή του, όπως ορίζει η συμβατική διαδικασία ανάπτυξης αλγορίθμων μάθησης. Ωστόσο, τα δεδομένα που δεν χρησιμοποιήθηκαν για την εκπαίδευση του μοντέλου, δηλαδή τα σύνολα Επιβεβαίωσης και Αξιολόγησης, αποτελούν αποδεκτά και αντιπροσωπευτικά δεδομένα του προβλήματος που εξετάζεται, τα οποία μπορούν να προσφέρουν περαιτέρω γνώση για τα υποκείμενα μοτίβα του συνόλου δεδομένων.

3 Εφαρμογή σε πρόβλημα Ανίχνευσης Ανωμαλίας

3.1 Περιγραφή Προβλήματος

Η δεύτερη εφαρμογή αφορά σε ένα σύνολο δεδομένων το οποίο συλλέχθηκε από έναν βιομηχανικό ανεμιστήρα κατά τη διάρκεια της παραγωγική διαδικασίας. Βασική λειτουργία του μηχανήματος αποτελεί η επανακυκλοφορία του αέρα, έτσι ώστε να ρυθμίζεται η θερμοκρασία και να ψύχεται ο εξοπλισμός τον οποίο υποβοηθά.

Συγκεκριμένα, αισθητήρες ακρίβειας τοποθετήθηκαν στις δυο προσαρτημένες μονάδες ρουλεμάν του βιομηχανικού ανεμιστήρα και κατέγραφαν για ένα διάστημα μηνών μετρήσεις ταχύτητας και θερμοκρασίας. Μάλιστα, οι παραπάνω μετρήσεις σημειώθηκαν κατά τη διάρκειας ήπιας, αλλά και επιβαρυμένης λειτουργίας του μηχανήματος. Συνεπώς, για κάθε μονάδα ρουλεμάν υπάρχουν δυο διαθέσιμα σύνολα δεδομένων φυσιολογικής και επιβαρυμένης κατάστασης σε μορφή χρονοσειρών.

Η αρχιτεκτονική του μοντέλου νευρωνικού δικτύου Βαθιάς Μάθησης που επιλέχτηκε για το συγκεκριμένο πρόβλημα είναι αυτή του Αυτοκωδικοποιητή (Autoencoder - AE) με την ενσωμάτωση επαναληπτικών επιπέδων LSTM, ακολουθώντας τη προσέγγιση της Ανακατασκευής (Reconstruction). Επιπλέον, η διαδικασία μάθησης του μοντέλου θα επιτευχθεί με τον τύπο της Αυτό-Επιβλεπόμενης Μάθησης (Self-Supervised Learning), καθώς τα ίδια τα δεδομένα εισόδου, τα οποία θα τροφοδοτηθούν στο μοντέλο, θα αποτελούν και την επιθυμητή έξοδό του.

Στα πλαίσια της Προβλεπτικής Συντήρησης, η παραπάνω αποτελεί μια εφαρμογή Ανίχνευσης Ανωμαλιών (Anomaly Detection), όπου το εκπαιδευμένο μοντέλο, μετά την εκπαίδευσή του πάνω στα δεδομένα φυσιολογικής κατάστασης, θα είναι σε θέση να αναγνωρίζει τυχόν μη-φυσιολογικές συμπεριφορές σε νέα δεδομένα.

Για καλύτερη κατανόηση όλων των διαδικασιών ανάλυσης και του διαχωρισμού των δυο μονάδων ρουλεμάν, ακολουθείται η σύμβαση, ως προς την ονοματολογία τους, σε «αριστερή» και «δεξιά» προσαρτημένη μονάδα ρουλεμάν, καθ’ όλη τη διάρκεια εξέτασης της συγκεκριμένης εφαρμογής.

Τα δεδομένα του προβλήματος είναι δημοσίως διαθέσιμα για κατέβασμα διαδικτυακά [90] από τη βάση δεδομένων του Zenodo.

3.2 Εισαγωγή Πακέτων

Οι απαραίτητες βιβλιοθήκες Python που χρησιμοποιήθηκαν, καθώς και οι εκδόσεις τους τη στιγμή της εξέτασης του προβλήματος, παρουσιάζονται παρακάτω:

```
Input []:
1  import pandas
2  import matplotlib
3  import numpy
4  from tensorflow import keras
5  import sklearn
6
7  print('pandas', pandas.__version__)
8  print('matplotlib', matplotlib.__version__)
9  print('numpy', numpy.__version__)
10 print('keras', keras.__version__)
11 print('sklearn', sklearn.__version__)

Output []:
pandas 1.5.3
matplotlib 3.7.1
numpy 1.22.4
keras 2.12.0
sklearn 1.2.2
```

Για τις ανάγκες μιας πιο ευέλικτης ανάλυσης και διαχείρισης του προγραμματιστικού κώδικα, η εισαγωγή των απαραίτητων πακέτων γίνεται ως εξής:

```
Input []:
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4  from tensorflow import keras
5  from sklearn.preprocessing import MinMaxScaler
```

3.3 Εισαγωγή Δεδομένων

Με την προϋπόθεση ότι υπάρχει προσαρτημένο το Google Drive στο notebook όπου εργαζόμαστε, αλλά και τα αντίστοιχα αρχεία ανεβασμένα σε κατάλληλο φάκελο, μπορούμε να φορτώσουμε, ως αντικείμενα DataFrame, όλα τα απαραίτητα αρχεία ως εξής:

```
Input []:
1  left_norm = pd.read_csv('/content/drive/MyDrive/
2                           Colab Notebooks/thesis/data/
```

```
3                          left_bearing_unit_normal_state.csv',
4                          index_col=[0])
5  left_abnorm = pd.read_csv('/content/drive/MyDrive/
6                            Colab Notebooks/thesis/data/
7                            left_bearing_unit_encumbered_state.csv',
8                            index_col=[0])
9  right_norm = pd.read_csv('/content/drive/MyDrive/
10                          Colab Notebooks/thesis/data/
11                          right_bearing_unit_normal_state.csv',
12                          index_col=[0])
13 right_abnorm = pd.read_csv('/content/drive/MyDrive/
14                            Colab Notebooks/thesis/data/
15                            right_bearing_unit_encumbered_state.csv',
16                            index_col=[0])
```

Τα DataFrame left_norm και left_abnorm αντιστοιχούν στις μετρήσεις που καταγράφηκαν για την αριστερή προσαρτημένη μονάδα ρουλεμάν σε φυσιολογικές και σε επιβαρυμένες λειτουργικές συνθήκες, ενώ τα DataFrame right_norm και right_abnorm αποτελούν τους αντίστοιχους πίνακες δεδομένων για τη δεξιά προσαρτημένη μονάδα ρουλεμάν.

Η δήλωση index_col=[0] αποτελεί ειδική περίπτωση εισαγωγής δεδομένων, των οποίων τα πρωτότυπα αρχεία .csv τα οποία φορτώνονται έχουν επιπλέον αποθηκευμένο το ευρετήριο των γραμμών στην πρώτη τους στήλη.

Με ξεχωριστή κλήση της καθεμίας από τις παραπάνω τέσσερις μεταβλητές παρουσιάζονται οι αντίστοιχοι πίνακες δεδομένων και κάθε μονάδα ρουλεμάν:

	vrms_left	temp_left
0	0.758	74.1
1	0.086	95.2
2	0.093	102.5
3	0.087	86.5
4	0.089	86.4
...	...	...
5255	0.099	117.1
5256	0.103	119.1
5257	0.099	121.0
5258	0.305	88.3
5259	0.249	81.1

5260 rows × 2 columns

	vrms_left	temp_left
0	0.149	103.1
1	0.138	104.2
2	0.140	111.4
3	0.151	114.8
4	0.149	123.2
...	...	...
2636	0.359	95.3
2637	0.121	87.5
2638	0.133	86.6
2639	0.130	85.0
2640	0.128	82.9

2641 rows × 2 columns

Εικόνα 60. Πίνακες Δεδομένων Αριστερής Μονάδας Ρουλεμάν για τη Φυσιολογική (αριστερά) και Επιβαρυμένη (δεξιά) κατάσταση

	vrms_right	temp_right
0	0.323	74.4
1	0.083	89.1
2	0.083	94.2
3	0.073	84.0
4	0.077	83.0
...	...	...
5255	0.103	110.0
5256	0.102	114.8
5257	0.096	116.7
5258	0.553	98.2
5259	0.523	94.1

5260 rows × 2 columns

	vrms_right	temp_right
0	0.132	102.8
1	0.151	103.7
2	0.159	105.2
3	0.169	107.0
4	0.196	110.3
...	...	...
2636	0.501	81.6
2637	0.161	93.9
2638	0.163	92.0
2639	0.144	90.3
2640	0.160	88.0

2641 rows × 2 columns

Εικόνα 61. Πίνακες Δεδομένων Δεξιάς Μονάδας Ρουλεμάν για τη Φυσιολογική (αριστερά) και Επιβαρυμένη (δεξιά) κατάσταση

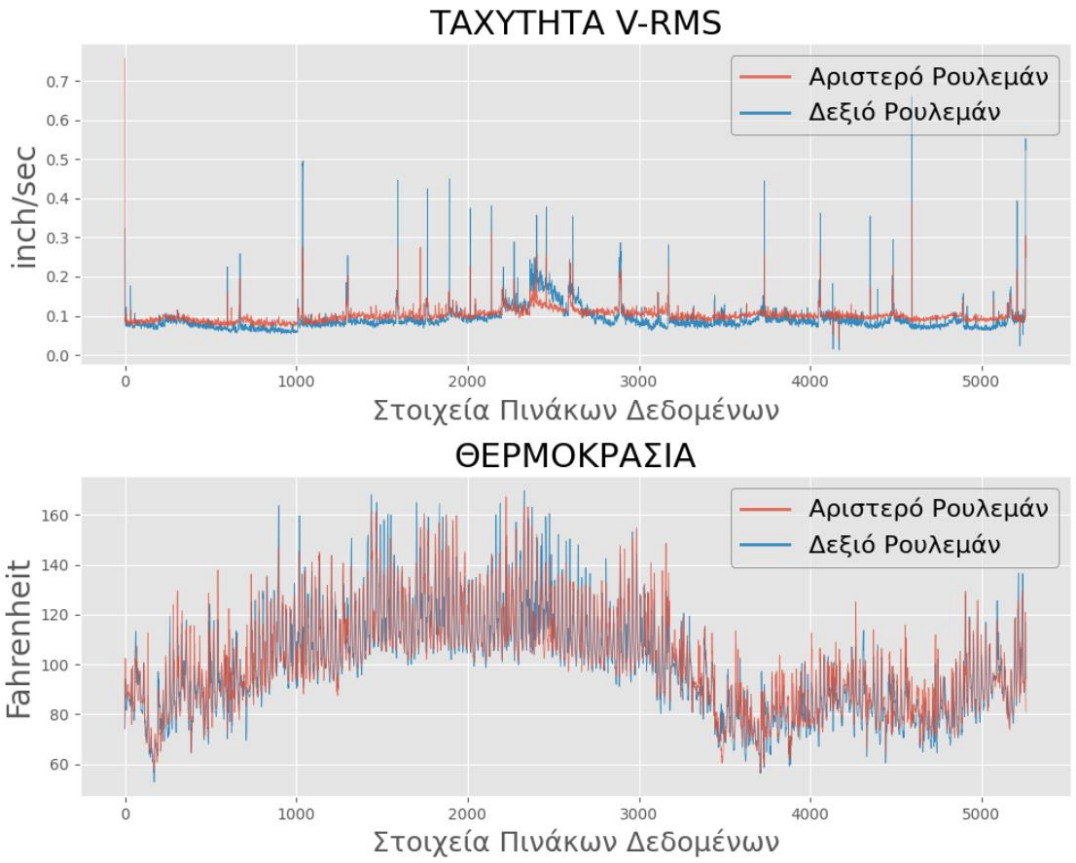
Επομένως, παρατηρούμε ότι, τα χαρακτηριστικά κάθε συνόλου δεδομένων είναι η ταχύτητα V-RMS, όπως περιγράφηκε στο υποκεφάλαιο 2.2.3 από την Ανάλυση Κραδασμών, και η θερμοκρασία. Επίσης, οι μονάδες μέτρησης που χρησιμοποιούνται για τα παραπάνω μεγέθη είναι $inch/sec$ (ίντσες ανά δευτερόλεπτο) για την ταχύτητα V-RMS, και βαθμοί Φαρενάιτ $°F$ (Fahrenheit) για τη θερμοκρασία.

Επιπλέον πληροφορίες που εξάγουμε αφορούν τις διαστάσεις κάθε συνόλου δεδομένων, όπου για τις φυσιολογικές καταστάσεις των ρουλεμάν υπάρχουν 5260 εγγραφές, ενώ για τις επιβαρυμένες λειτουργικές καταστάσεις υπάρχουν 2641 καταγραφές μετρήσεων ταχύτητας V-RMS και θερμοκρασίας.

3.4 Οπτικοποίηση Δεδομένων

Όπως και στην πρώτη εφαρμογή, έτσι και εδώ, τα δεδομένα βρίσκονται στην τελική μορφή τους και είναι ήδη έτοιμα για οποιαδήποτε απαραίτητη προεπεξεργασία, πριν την τροφοδότησή τους στο μοντέλο Βαθιάς Μάθησης. Συνεπώς, δεν θα επαναληφθεί η προηγούμενη εκτενής στατιστική ανάλυση, αλλά θα οπτικοποιηθούν οι χρονοσειρές όλων των περιπτώσεων, στα πλαίσια σύγκρισης και ερμηνείας συμπεριφορών των δεδομένων.

Ο κώδικας Python για την εξαγωγή καθενός από τα παρακάτω διαγράμματα χρονοσειρών βρίσκεται στο Παράρτημα Θ’.



Εικόνα 62. Χρονοσειρές Ταχύτητας και Θερμοκρασίας Φυσιολογικής Λειτουργικής Κατάστασης για τις Μονάδες Ρουλεμάν

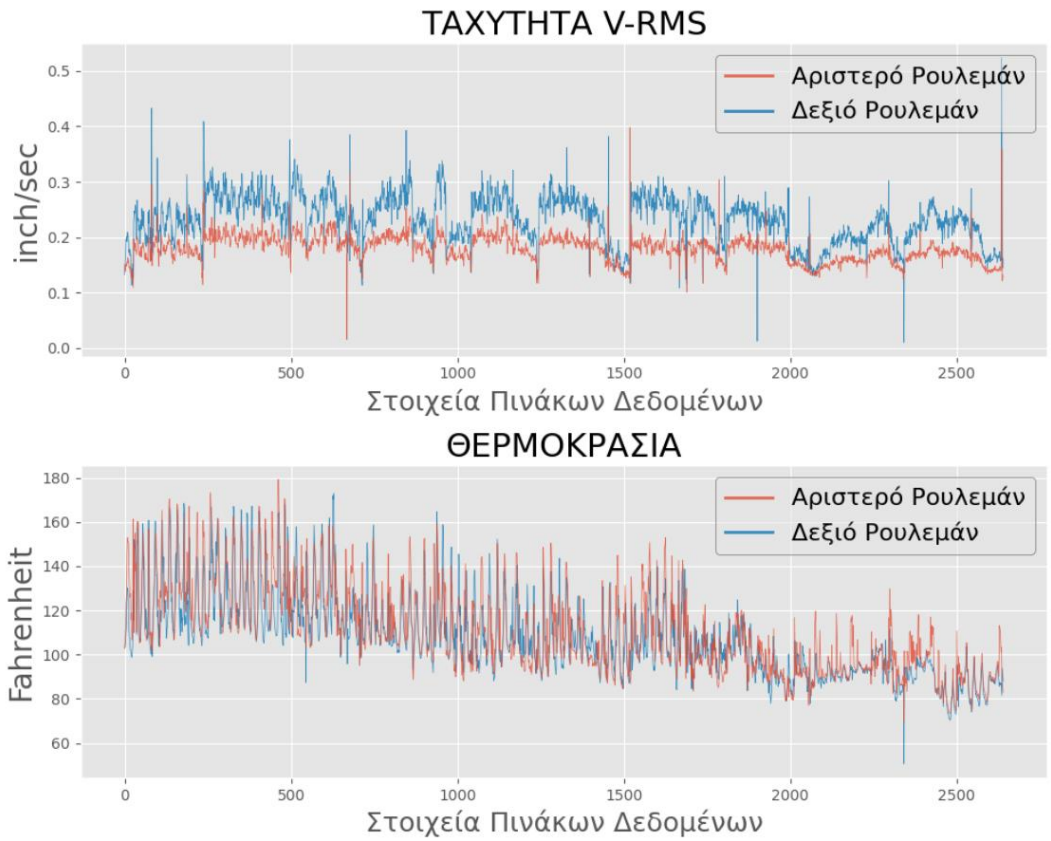
Παραπάνω παρουσιάζονται οι μετρήσεις της ταχύτητας V-RMS και της θερμοκρασίας των δυο προσαρτημένων μονάδων ρουλεμάν, όπως καταγράφηκαν κατά τη φυσιολογική κατάσταση λειτουργίας του βιομηχανικού ανεμιστήρα.

Γενικότερα, η συμπεριφορά των δυο ρουλεμάν κυμαίνεται στα ίδια πλαίσια μετρήσεων, ωστόσο, ειδικότερα για την περίπτωση των ταχυτήτων V-RMS, παρατηρείται μια πιο επιβαρυμένη κατάσταση στη δεξιά μονάδα κατά τη μέση της χρονοσειράς.

Επίσης, είναι σημαντικό να γίνει αναφορά στις μεμονωμένες απότομες υψηλές τιμές της ταχύτητας που καταγράφηκαν καθ’ όλη τη διάρκεια λειτουργίας του μηχανήματος. Από τη στιγμή που τα δεδομένα συλλέχθηκαν από πραγματική βιομηχανική διάταξη, είναι λογικό να συνοδεύονται και να επηρεάζονται από ποικίλους παράγοντες που έχουν να κάνουν με την παραγωγική διαδικασία.

Συγκεκριμένα, για τον εξοπλισμό που εξετάζεται, παραδείγματα των παραπάνω παραγόντων ήταν οι ξαφνικές πτώσης τάσης ή διακοπής του ρεύματος, αλλά και οι σκόπιμες παύσεις λειτουργίας για τυχόν εργασίες συντήρησης. Σε κάθε περίπτωση, η επανεκκίνηση του μηχανήματος προκαλεί απότομη αύξηση των δονήσεων μέχρι να σταθεροποιηθεί η λειτουργία του, και τα γεγονότα αυτά καταγράφηκαν από τους αισθητήρες και μεταφράστηκαν στις παραπάνω απότομες υψηλές μετρήσεις. Επιπλέον, αξίζει να σημειωθεί ότι οι απότομες υψηλές μετρήσεις της δεξιάς μονάδας ρουλεμάν είναι υψηλότερες από τις αντίστοιχες της αριστερής μονάδας, λόγω της θέσης της δεξιάς μονάδας πιο κοντά στον ανεμιστήρα του μηχανήματος.

Παρακάτω παρουσιάζονται οι μετρήσεις της ταχύτητας V-RMS και της θερμοκρασίας των δυο προσαρτημένων μονάδων ρουλεμάν, όπως καταγράφηκαν κατά την επιβαρυμένη κατάσταση λειτουργίας του βιομηχανικού ανεμιστήρα:



Εικόνα 63. Χρονοσειρές Ταχύτητας και Θερμοκρασίας Επιβαρυμένης Λειτουργικής Κατάστασης για τις Μονάδες Ρουλεμάν

Παρατηρείται και πάλι ότι οι μετρήσεις και από τις δυο μονάδες ρουλεμάν ακολουθούν την ίδια συμπεριφορά, ωστόσο είναι εμφανές ότι οι τιμές της ταχύτητας V-RMS για τη δεξιά μονάδα είναι γενικά υψηλότερες από τις αντίστοιχες της αριστερής μονάδας, κάτι το οποίο επίσης είναι πιθανό να οφείλεται και πάλι στη θέση της δεξιάς μονάδας και στην εγγύτητά του με τον ανεμιστήρα της βιομηχανικής διάταξης.

Για περαιτέρω σύγκριση των μετρήσεων μεταξύ όλων των περιπτώσεων παρουσιάζεται παρακάτω ο μέσος όρος για κάθε μέγεθος που καταγράφηκε:

Πίνακας 3. Μέσος Όρος Χρονοσειρών

	Φυσιολογική Κατάσταση	Επιβαρυμένη Κατάσταση	
V-RMS αριστερής μονάδας	0.101	0.179	$inch/sec$
V-RMS δεξιάς μονάδας	0.092	0.233	

ΘΕΡΜΟΚΡΑΣΙΑ αριστερής μονάδας	100.27	110.45	$°F$
ΘΕΡΜΟΚΡΑΣΙΑ δεξιάς μονάδας	98.88	107.24	

Επομένως, είναι ιδιαίτερα προφανής η διαφορά στις τιμές ταχύτητας και θερμοκρασίας μεταξύ των φυσιολογικών και επιβαρυμένων καταστάσεων λειτουργίας.

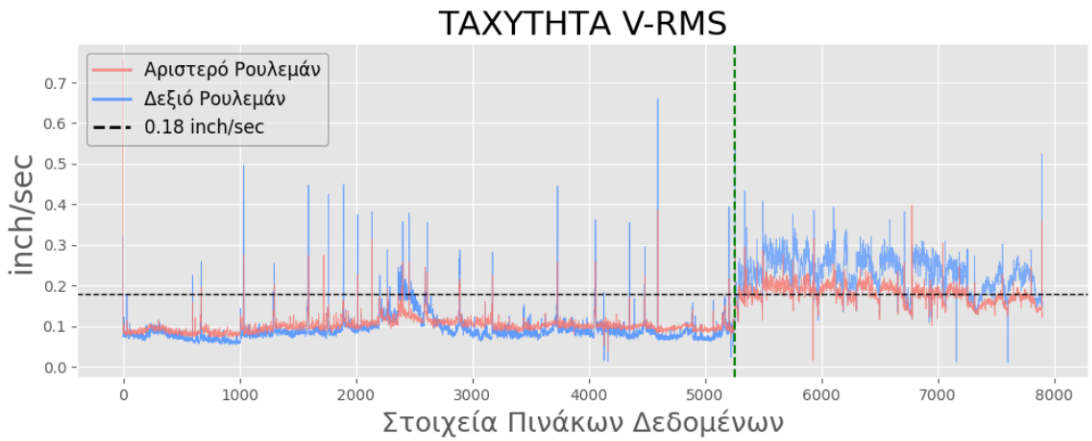
ΠΑΡΑΤΗΡΗΣΗ

Το διεθνές πρότυπο ISO 20816-3:2022, το οποίο αφορά στις μετρήσεις των μηχανικών κραδασμών, συστήνει για τον συγκεκριμένο βιομηχανικό εξοπλισμό, ως επιθυμητά άνω όρια τιμών, τα $0.03\ inch/sec$, $0.07\ inch/sec$ και $0.18\ inch/sec$, τα οποία αντιστοιχούν σε καλή, μέτρια και κακή λειτουργική κατάσταση, ενώ οποιαδήποτε μεγαλύτερη μέτρηση θεωρείται χαρακτηριστικό επικίνδυνης περίπτωσης για το μηχάνημα. Ωστόσο, η συγκεκριμένη οριοθέτηση έχει κυρίως κατευθυντήριο σκοπό για τις βιομηχανίες και τον έλεγχο των εξαρτημάτων τους, και για τον λόγο αυτόν δεν είναι αρκετή για να αναγνωρίσει υποκείμενες επιβλαβής συμπεριφορές.

Για παράδειγμα, οι παραπάνω στιγμιαίες απότομες μετρήσεις ταχύτητας V-RMS, βάσει των ορίων του ISO θα θεωρούνταν επικίνδυνες, παρά το γεγονός ότι πρόκειται για φυσιολογική συμπεριφορά της παραγωγικής διαδικασίας, και θα οδηγούσαν σε τυχόν παύσεις που τελικά θα επιβεβαιώνανε τον λάθος συναγερμό. Για τον λόγο αυτό, μια τεχνική Βαθιάς Μάθησης θα ήταν πολύ πιο αξιόπιστη, συνοδευόμενη με τις μετρήσεις ISO ως μια επιπλέον μετρική σε υποβοηθητικό ρόλο.

Επιπλέον, οι υψηλότερες μετρήσεις κατά την επιβαρυμένη κατάσταση λειτουργίας δεν αποτελούν απαραίτητα λόγο ανησυχίας για κάποια σοβαρή βλάβη, καθώς προφανώς ο βιομηχανικός εξοπλισμός λειτουργούσε κανονικά και καταγράφονταν οι μετρήσεις από τους αισθητήρες. Οι κατευθυντήριες των πρωτόκολλων και των προτύπων υπάρχουν για να προσφέρουν μια γενικότερη εικόνα κατά την παραγωγική διαδικασία και δεν αποτελούν αυστηρούς κανόνες διαχείρισης των εξαρτημάτων.

Παρ’ όλα αυτά, μια πιο συγκεντρωτική εικόνα για όλες τις μετρήσεις της ταχύτητας V-RMS δίνεται στο παρακάτω διάγραμμα, όπου παρουσιάζεται η σχέση κάθε μέτρησης ταχύτητας για τις δυο μονάδες ρουλεμάν με το ανώτατο όριο του προτύπου ISO:



Εικόνα 64. Χρονοσειρές Ταχύτητας Φυσιολογικής και Επιβαρυμένης Λειτουργικής Κατάστασης για τις Μονάδες Ρουλεμάν

Συνεπώς, με την προσπάθεια ανάπτυξης του μοντέλου Βαθιάς Μάθησης δεν είναι σκοπός μας να εκπαιδεύσουμε ένα μοντέλο να αναγνωρίζει όλες τις επιβαρυμένες καταστάσεις ως ανώμαλες συμπεριφορές, αλλά περισσότερο να ξεχωρίζει τις πιο ακραίες περιπτώσεις στα δεδομένα.

3.5 Προεπεξεργασία Δεδομένων

Αρχικά, για να γίνει ξεκάθαρη η όλη παρακάτω διαδικασία, υπενθυμίζεται ότι τα διαθέσιμα σύνολα δεδομένων είναι τέσσερα. Συγκεκριμένα, η φυσιολογική και επιβαρυμένη κατάσταση της αριστερής μονάδας ρουλεμάν, και η φυσιολογική και επιβαρυμένη κατάσταση της δεξιάς μονάδας ρουλεμάν.

Για την ανάπτυξη και εκπαίδευση του μοντέλου Βαθιάς Μάθησης θα χρησιμοποιηθεί η φυσιολογική κατάσταση της αριστερής μονάδας, λόγω της πιο σταθερής συμπεριφοράς της χρονοσειράς (το δεξιό ρουλεμάν εμφάνισε διαταραχή στη μέση της χρονοσειράς ταχύτητας).

Ειδικότερα, το σύνολο δεδομένων της φυσιολογικής κατάστασης του αριστερού ρουλεμάν, θα διαιρεθεί και πάλι σε σύνολα Εκπαίδευσης και Αξιολόγησης, και μετά την εκπαίδευση του μοντέλου στα δεδομένα Εκπαίδευσης της φυσιολογικής κατάστασης του αριστερού ρουλεμάν, η γενικότερη αξιολόγηση θα γίνει στα δεδομένα Αξιολόγησης της φυσιολογικής κατάστασης του αριστερού ρουλεμάν, στα δεδομένα της επιβαρυμένης κατάστασης του αριστερού ρουλεμάν, στα δεδομένα της φυσιολογικής κατάστασης του δεξιού ρουλεμάν και στα δεδομένα της επιβαρυμένης κατάστασης του δεξιού ρουλεμάν.

Επομένως, κάθε απαραίτητη προεπεξεργασία των χρονοσειρών και των τιμών τους θα γίνεται συνολικά σε πέντε σύνολα δεδομένων, αλλά τα τέσσερα πρωτότυπα σύνολα με τις διαστάσεις τους, όπως παρουσιάστηκαν και προηγουμένως κατά την εισαγωγή τους, είναι τα παρακάτω:

```
Input []:
1  print(left_norm.shape)
2  print(left_abnorm.shape)
3  print(right_norm.shape)
4  print(right_abnorm.shape)

Output []:
(5260, 2)
(2641, 2)
(5260, 2)
(2641, 2)
```

3.5.1 Διαίρεση Δεδομένων Εκπαίδευσης και Αξιολόγησης

Σε αντίθεση με τη διαίρεση της πρώτης εφαρμογής, όπου τα δεδομένα χωρίστηκαν τυχαία με τη βοήθεια της μεθόδου train_test_split() της βιβλιοθήκης Scikit-learn, στη συγκεκριμένη περίπτωση, η διαίρεση των δεδομένων θα γίνει χρονικά, ακολουθώντας και πάλι τον συμβατικό ποσοστιαίο διαχωρισμό των 80% και 20% σε σύνολα Εκπαίδευσης και Αξιολόγησης αντίστοιχα.

Επαναλαμβάνεται ότι η παραπάνω διαίρεση γίνεται στις φυσιολογικές μετρήσεις της αριστερής μονάδας ρουλεμάν, και μάλιστα όχι στους πρωτότυπους πίνακες δεδομένων, με τον παρακάτω τρόπο:

```
Input []:
1  left_norm_train = left_norm.copy()
2  left_norm_test = left_norm.copy()
3
4  left_norm_train.drop(index=left_norm_train.loc[4208:].index,
5                       inplace=True)
6  left_norm_test.drop(index=left_norm_test.loc[:4207].index,
7                       inplace=True)
```

Συνεπώς, έγινε η διαίρεση των 5260 φυσιολογικών τιμών του αριστερού ρουλεμάν στις πρώτες 4208 εγγραφές, για τη δημιουργία του συνόλου Εκπαίδευσης, και στις υπόλοιπες 1052 εγγραφές, για τη δημιουργία του συνόλου Αξιολόγησης.

Επομένως, τα πέντε σύνολα δεδομένων με τις αντίστοιχες διαστάσεις τους για αυτήν τη στιγμή είναι τα παρακάτω:

```
Input []:
1  print(left_norm_train.shape)
2  print(left_norm_test.shape)
3  print(left_abnorm.shape)
4  print(right_norm.shape)
5  print(right_abnorm.shape)

Output []:
(4208, 2)
(1052, 2)
(2641, 2)
(5260, 2)
(2641, 2)
```

Προφανώς, με την κλήση της καθεμίας μεταβλητής, ο χρήστης μπορεί να εμφανίσει και να εξετάσει κάθε πίνακα δεδομένων ως αντικείμενο DataFrame ξεχωριστά, τόσο αυτήν τη στιγμή, όσο και καθ' όλη τη διάρκεια της παρακάτω επεξεργασίας των δεδομένων.

3.5.2 Φιλτράρισμα Δεδομένων

Αναφέρθηκε ότι οι απότομες ξαφνικές υψηλές μετρήσεις στην ταχύτητα V-RMS αποτελούν φυσιολογική και αναμενόμενη συμπεριφορά ενός βιομηχανικού μηχανήματος σε μια πραγματική παραγωγική διαδικασία, συνεπώς η διαγραφή τους ως ακραίες τιμές θα προκαλούσε πιθανώς διατάραξη στα δεδομένα.

Για τον λόγο αυτό αποφασίστηκε το φιλτράρισμα κάθε χρονοσειράς με τη μέθοδο της κινούμενης διαμέσου. Σύμφωνα με τη μέθοδο αυτή, για κάθε διαδοχικό πλήθος τιμών της χρονοσειράς, το οποίο καθορίζεται από τον χρήστη, επιλέγεται η διάμεσός τους, με αποτέλεσμα να δημιουργηθεί με τον τρόπο αυτό μια νέα χρονοσειρά, η οποία θα χρησιμοποιηθεί, αντί της πρωτότυπης, για κάθε επόμενη διαδικασία.

Το παραπάνω φιλτράρισμα, για κάθε ένα από τα πέντε σύνολα δεδομένων, γίνεται προγραμματιστικά με τον παρακάτω τρόπο:

```
Input []:
1  left_norm_train_median = left_norm_train.copy()
2  left_norm_test_median = left_norm_test.copy()
3  left_abnorm_median = left_abnorm.copy()
4  right_norm_median = right_norm.copy()
```

```
5  right_abnorm_median = right_abnorm.copy()
6
7  left_norm_train_median =left_norm_train_median.
8                                  rolling(24).median().dropna()
9  left_norm_test_median = left_norm_test_median.
10                                 rolling(24).median().dropna()
11 left_abnorm_median = left_abnorm_median.
12                                 rolling(24).median().dropna()
13 right_norm_median = right_norm_median.
14                                 rolling(24).median().dropna()
15 right_abnorm_median = right_abnorm_median.
16                                 rolling(24).median().dropna()
17
18 left_norm_train_median.reset_index(drop=True, inplace = True)
19 left_norm_test_median.reset_index(drop=True, inplace = True)
20 left_abnorm_median.reset_index(drop=True, inplace = True)
21 right_norm_median.reset_index(drop=True, inplace = True)
22 right_abnorm_median.reset_index(drop=True, inplace = True)
```

Αρχικά, με τη μέθοδο .copy(), δημιουργούνται αντίγραφα των πέντε συνόλων δεδομένων, τα οποία διαχωρίζονται με την κατάληξη _median.

Στη συνέχεια, γίνεται αντικατάσταση των αντιγραμμένων χρονοσειρών σύμφωνα με τη μέθοδο της κινούμενης διαμέσου, μέσω των επαναληπτικών μεθόδων .rolling(24).median(). Συνεπώς, κάθε 24 διαδοχικές μετρήσεις αντικαθίστανται με τη διάμεσό τους, και το πλήθος των 24 επιλέχτηκε πειραματικά κατά τη διαδικασία ανάπτυξης του μοντέλου Βαθιάς Μάθησης. Ταυτόχρονα, στους νέους πίνακες δεδομένων των διαμέσων, διαγράφονται με τη μέθοδο .dropna() οι πρώτες 23 γραμμές καθώς είναι κενές (NaN), διότι η αντικατάσταση της διαμέσου προϋποθέτει την ύπαρξη 24 διαδοχικών μετρήσεων, κάτι που δεν υφίσταται για τις πρώτες 23 γραμμές των αρχικών πινάκων δεδομένων.

Έπειτα, διορθώνεται η αρίθμηση του ευρετηρίου για καθένα από τα νέα σύνολα δεδομένων των διαμέσων, έτσι ώστε να ξεκινάει από το μηδέν.

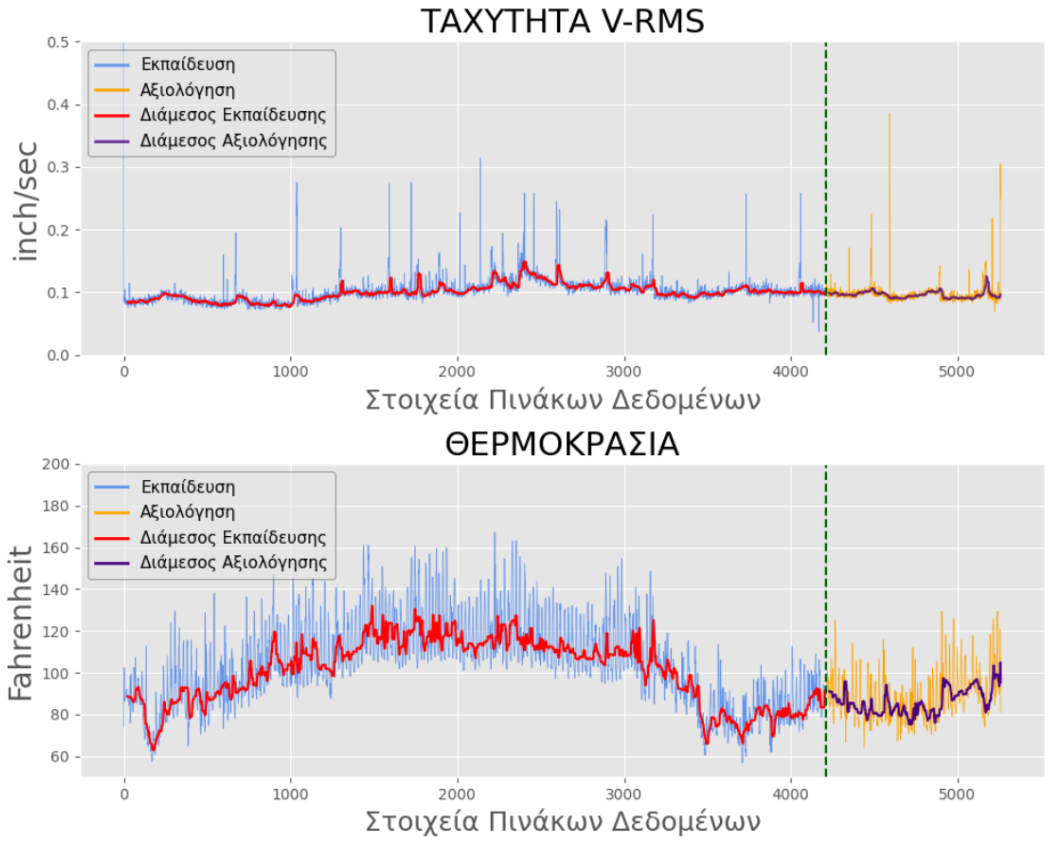
Τελικά, τα πέντε σύνολα δεδομένων των κινούμενων διαμέσων με τις αντίστοιχες διαστάσεις τους είναι τα παρακάτω:

```
Input []:
1  print(left_norm_train_median.shape)
2  print(left_norm_test_median.shape)
3  print(left_abnorm_median.shape)
4  print(right_norm_median.shape)
5  print(right_abnorm_median.shape)
```

```
Output []:
(4185, 2)
(1029, 2)
(2618, 2)
(5237, 2)
(2618, 2)
```

Επαναλαμβάνεται ότι, τα πέντε νέα σύνολα δεδομένων των διαμέσων είναι πλέον κατά 23 εγγραφές μικρότερα από τα προηγούμενα, λόγω του ότι δεν μπορεί να υπολογιστεί διάμεσος για τις 23 πρώτες μετρήσεις.

Παρακάτω παρουσιάζονται, μόνο για τη φυσιολογική κατάσταση της αριστερής μονάδας ρουλεμάν, τα πρωτότυπα και τα φιλτραρισμένα δεδομένα Εκπαίδευσης και Αξιολόγησης:



Εικόνα 65. Διαίρεση Χρονοσειρών Ταχύτητας και Θερμοκρασίας σε Σύνολα Δεδομένων Εκπαίδευσης και Αξιολόγησης με Φιλτράρισμα Διαμέσου

Επομένως, είναι προφανές ότι με τη μέθοδο της κινούμενης διαμέσου οι χρονοσειρές γίνονται περισσότερο διαχειρίσιμες, καθώς εξομαλύνονται οι ακραίες τιμές των απότομων επανεκκινήσεων του μηχανήματος.

Ο κώδικας Python για την εξαγωγή των παραπάνω διαγραμμάτων βρίσκεται επίσης στο Παράρτημα Θ'.

3.5.3 Αλλαγή Κλίμακας Δεδομένων

Ήδη από την πρώτη εφαρμογή αναγνωρίστηκε η σημασία της αλλαγής κλίμακας των χαρακτηριστικών για μια καλύτερη εκπαιδευτική διαδικασία. Συνεπώς, είναι λογικό να εφαρμοστεί αντίστοιχος μετασχηματισμός και για τα σύνολα δεδομένων τους προβλήματος που εξετάζεται, μιας και υπάρχει σημαντική διαφορά της τάξης των αριθμών μεταξύ της ταχύτητας και της θερμοκρασίας, κάτι που μπορεί να οδηγήσει το μοντέλο να παραμετροποιηθεί υπέρ των μεγαλύτερων τιμών.

Ειδικότερα, για την επίτευξη της ομοιογένειας μεταξύ των μετρήσεων, η μέθοδος που επιλέχθηκε είναι αυτή της Ομαλοποίησης (Normalization) η οποία χρησιμοποιείται συχνά ως εναλλακτική της Κανονικοποίησης που εφαρμόστηκε στο προηγούμενο πρόβλημα. Μέσω της Ομαλοποίησης, οι τιμές κάθε χαρακτηριστικού (δηλαδή κάθε στήλης) μετασχηματίζονται έτσι ώστε να περιορίζονται σε ένα προκαθορισμένο εύρος τιμών, και η αλλαγή των μεγεθών βασίζεται στον παρακάτω μαθηματικό τύπο:

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \cdot (max - \min) + min$$

Όπου:

- X είναι οι αρχικές τιμές της στήλης του εκάστοτε χαρακτηριστικού
- X_{min} είναι η ελάχιστη τιμή των αρχικών τιμών X
- X_{max} είναι η μέγιστη τιμή των αρχικών τιμών X
- min είναι το κάτω όριο του νέου επιθυμητού εύρους τιμών
- max είναι το πάνω όριο του νέου επιθυμητού εύρους τιμών
- X_{scaled} είναι οι νέες μετασχηματισμένες τιμές του εκάστοτε χαρακτηριστικού

Ειδικά για τα μεγέθη του προβλήματος που εξετάζεται, επιλέχθηκε ο μετασχηματισμός των μετρήσεων σε νέες τιμές ανάμεσα στο συμβατικό εύρος μεταξύ του μηδέν και του ένα, συνεπώς για τον παραπάνω μαθηματικό τύπο ισχύει $min = 0$ και $max = 1$.

Ο παραπάνω μετασχηματισμός και η αλλαγή κλίμακας όλων των συνόλων δεδομένων επιτυγχάνεται με τη βοήθεια της βιβλιοθήκης Scikit-learn ως εξής:

```
Input []:
1  scaler = MinMaxScaler(feature_range=(0, 1))
2  scaler.fit(left_norm_train_median)
3
```

```
4  ds_left_norm_train = scaler.transform(left_norm_train_median)
5  ds_left_norm_test = scaler.transform(left_norm_test_median)
6  ds_left_abnorm = scaler.transform(left_abnorm_median)
7
8  right_norm_median.rename(columns={"vrms_right": "vrms_left",
9                                    "temp_right": "temp_left"},
10                           inplace=True)
11 right_abnorm_median.rename(columns={"vrms_right": "vrms_left",
12                                     "temp_right": "temp_left"},
13                            inplace=True)
14
15 ds_right_norm = scaler.transform(right_norm_median)
16 ds_right_abnorm = scaler.transform(right_abnorm_median)
```

Αρχικά, ορίζεται ο μετασχηματιστής MinMaxScaler() για το εύρος τιμών μεταξύ του μηδέν και του ένα, και στη συνέχεια εφαρμόζεται, μέσω της μεθόδου .fit(), πάνω στο σύνολο Εκπαίδευσης της φυσιολογικής κατάστασης της αριστερής προσαρτημένης μονάδας ρουλεμάν, έτσι ώστε να υπολογιστούν όλες οι ελάχιστες και μέγιστες τιμές για όλα τα χαρακτηριστικά-στήλες του συνόλου, οι οποίες θα χρησιμοποιηθούν για την αλλαγή κλίμακας όλων των υπόλοιπων δεδομένων.

Μετά την εφαρμογή του μετασχηματιστή στο σύνολο Εκπαίδευσης, μετασχηματίζονται και αποθηκεύονται όλα τα σύνολα που αφορούν το αριστερό ρουλεμάν, και για τον διαχωρισμό τους χρησιμοποιείται το πρόθεμα ds_.

Έπειτα, είναι απαραίτητη η μετονομασία των στηλών των συνόλων δεδομένων του δεξιού ρουλεμάν, λόγω ιδιοτροπίας του κώδικα που απαιτεί ίδια ονομασία στηλών με το σύνολο όπου εφαρμόστηκε ο μετασχηματιστής MinMaxScaler(). Άλλος τρόπος για να αποφευχθεί αυτό είναι να μετονομαστούν από την αρχή όλες οι στήλες σύμφωνα με αύξουσα αρίθμηση. Τελικά, μετασχηματίζονται και αποθηκεύονται με πρόθεμα ds_ όλα τα σύνολα που αφορούν το δεξί ρουλεμάν.

Να σημειωθεί ότι, μέσω της διαδικασίας αλλαγής κλίμακας, όλα τα σύνολα δεδομένων κατάληξης _median τα οποία είναι αντικείμενα τύπου DataFrame, μετασχηματίζονται αυτομάτως σε αντικείμενα τύπου πίνακα (array) με πρόθεμα ds_, και εμφανίζονται συγκεντρωτικά παρακάτω, όπως αποθηκεύτηκαν, με τις αντίστοιχες διαστάσεις τους, οι οποίες δεν παρουσιάζουν κάποια διαφορά:

```
Input []:
1  print(ds_left_norm_train.shape)
2  print(ds_left_norm_test.shape)
3  print(ds_left_abnorm.shape)
```

```
4  print(ds_right_norm.shape)
5  print(ds_right_abnorm.shape)

Output []:
(4185, 2)
(1029, 2)
(2618, 2)
(5237, 2)
(2618, 2)
```

Ως παράδειγμα του μετασχηματισμού παρατίθεται το σύνολο Εκπαίδευσης:

```
Input []:
1  ds_left_norm_train

Output []:
array([[0.11188811, 0.37281977],
       [0.11188811, 0.37281977],
       [0.1048951 , 0.36918605],
       ...,
       [0.30769231, 0.41133721],
       [0.31468531, 0.41424419],
       [0.32167832, 0.41424419]])
```

3.5.4 Μετασχηματισμός Ακολουθιών

Αναφέρθηκε, κατά την περιγραφή του προβλήματος, ότι το μοντέλο της Βαθιάς Μάθησης το οποίο θα εφαρμοστεί, θα ακολουθεί την αρχιτεκτονική του Αυτοκωδικοποιητή (Autoencoder - AE) με την ενσωμάτωση επαναληπτικών επιπέδων LSTM. Επίσης αναφέρθηκε ότι, οι επαναληπτικοί νευρώνες επιπέδων τύπου RNN ή LSTM είναι φτιαγμένοι έτσι ώστε να μπορούν να αναλύουν ακολουθίες δεδομένων σε διαδοχικές χρονικές στιγμές μέσω βρόγχων. Τελικά, αυτό σημαίνει ότι τα πέντε σύνολα των δεδομένων του προβλήματος πρέπει να μετασχηματιστούν κατάλληλα σε ακολουθίες, έτσι ώστε να μπορούν τα τροφοδοτηθούν στο νευρωνικό δίκτυο για επεξεργασία και ανάλυση από τους νευρώνες LSTM.

Για το μέγεθος των ακολουθιών επιλέχθηκε να εφαρμοστεί το πλήθος των 24 διαδοχικών στοιχείων, και πάλι μέσα από πειραματισμούς. Συνεπώς, τα πέντε σύνολα δεδομένων προθέματος ds_, όπως μετασχηματίστηκαν μέχρι τώρα, μέσω της μεθόδου της κινούμενης διαμέσου, και της ομαλοποίησης για την αλλαγή κλίμακας, μετασχηματίζονται για ακόμα μια φορά σε ακολουθίες των 24 στοιχείων σύμφωνα με τον παρακάτω τρόπο:

```
Input []:
1  seq = 24
2  ds1_seq = []
3  ds2_seq = []
4  ds3_seq = []
5  ds4_seq = []
6  ds5_seq = []
7  for i in range(len(ds_left_norm_train)-24+1):
8      ds1_seq.append(ds_left_norm_train[i:i+seq])
9  for i in range(len(ds_left_norm_test)-24+1):
10     ds2_seq.append(ds_left_norm_test[i:i+seq])
11 for i in range(len(ds_left_abnorm)-24+1):
12     ds3_seq.append(ds_left_abnorm[i:i+seq])
13 for i in range(len(ds_right_norm)-24+1):
14     ds4_seq.append(ds_right_norm[i:i+seq])
15 for i in range(len(ds_right_abnorm)-24+1):
16     ds5_seq.append(ds_right_abnorm[i:i+seq])
17 ds_left_norm_train_seq = np.array(ds1_seq)
18 ds_left_norm_test_seq = np.array(ds2_seq)
19 ds_left_abnorm_seq = np.array(ds3_seq)
20 ds_right_norm_seq = np.array(ds4_seq)
21 ds_right_abnorm_seq = np.array(ds5_seq)
```

Τελικά, τα οριστικά σύνολα δεδομένων Εκπαίδευσης και Αξιολόγησης της φυσιολογικής λειτουργικής κατάστασης της αριστερής μονάδας ρουλεμάν, της επιβαρυμένης κατάστασης της αριστερής μονάδας ρουλεμάν, της φυσιολογικής κατάστασης της δεξιάς μονάδας ρουλεμάν και της επιβαρυμένης κατάστασης της δεξιάς μονάδας ρουλεμάν, παρουσιάζονται παρακάτω με τις αντίστοιχες διαστάσεις τους, και για ο διαχωρισμός τους από τα προηγούμενα σύνολα επιτυγχάνεται με την κατάληξη _seq:

```
Input []:
1  print(ds_left_norm_train_seq.shape)
2  print(ds_left_norm_test_seq.shape)
3  print(ds_left_abnorm_seq.shape)
4  print(ds_right_norm_seq.shape)
5  print(ds_right_abnorm_seq.shape)

Output []:
(4162, 24, 2)
(1006, 24, 2)
(2595, 24, 2)
(5214, 24, 2)
(2595, 24, 2)
```

Η διαφορά στην διάσταση των συνόλων δεδομένων είναι πλέον εμφανής, καθώς οι προηγουμένως δισδιάστατοι πίνακες έχουν μετασχηματιστεί σε τρισδιάστατους. Επίσης, παρατηρούμε ότι το πλήθος των στοιχείων (πρώτη διάσταση) σε κάθε σύνολο μειώθηκε για ακόμη μια φορά κατά 23 εγγραφές. Αυτό συνέβη διότι επιλέχτηκε προηγουμένως η ομαδοποίηση των στοιχείων σε ακολουθίες μήκους 24. Τελικά, κάθε ένα στοιχείο των πέντε νέων μετασχηματισμένων συνόλων δεδομένων αποτελεί μια ακολουθία 24 εγγραφών, όπου κάθε εγγραφή περιλαμβάνει 2 μετρήσεις, αυτές της ταχύτητας και της θερμοκρασίας.

Ως παράδειγμα του μετασχηματισμού παρατίθεται το σύνολο Εκπαίδευσης:

```
Input []:
1  ds_left_norm_train_seq

Output []:
array([[[0.11188811, 0.37281977],
        [0.11188811, 0.37281977],
        [0.1048951 , 0.36918605],
        ...,
        [0.09090909, 0.3619186 ],
        [0.09090909, 0.3619186 ],
        [0.0979021 , 0.35901163]],
```

```
       [[0.11188811, 0.37281977],
        [0.1048951 , 0.36918605],
        [0.09090909, 0.36700581],
        ...,
        [0.09090909, 0.3619186 ],
        [0.0979021 , 0.35901163],
        [0.09090909, 0.35610465]],

       [[0.1048951 , 0.36918605],
        [0.09090909, 0.36700581],
        [0.08391608, 0.36700581],
        ...,
        [0.0979021 , 0.35901163],
        [0.09090909, 0.35610465],
        [0.09090909, 0.35537791]],

       ...,

       [[0.3006993 , 0.29505814],
        [0.29370629, 0.29505814],
        [0.29370629, 0.29505814],
        ...,
```

```
        [0.29370629, 0.36555233],
        [0.3006993 , 0.40188953],
        [0.30769231, 0.41133721]],

       [[0.29370629, 0.29505814],
        [0.29370629, 0.29505814],
        [0.29370629, 0.29505814],
        ...,
        [0.3006993 , 0.40188953],
        [0.30769231, 0.41133721],
        [0.31468531, 0.41424419]],

       [[0.29370629, 0.29505814],
        [0.29370629, 0.29505814],
        [0.29370629, 0.29505814],
        ...,
        [0.30769231, 0.41133721],
        [0.31468531, 0.41424419],
        [0.32167832, 0.41424419]]])
```

3.6 Μοντέλο Αυτοκωδικοποιητή

Το επιλεγμένο μοντέλο Βαθιάς Μάθησης το οποίο εφαρμόστηκε, ακολουθεί την προσέγγιση της Ανακατασκευής (Reconstruction), μέσω της αρχιτεκτονικής του Αυτοκωδικοποιητή (Autoencoder - AE), με την ενσωμάτωση επαναληπτικών επιπέδων Μακράς Βραχυπρόθεσμης Μνήμης (Long Short-Term Memory - LSTM).

Συγκεκριμένα, ο ζητούμενος στόχος για την εκπαίδευση του μοντέλου AE-LSTM (Autoencoder-LSTM) είναι να τροφοδοτηθεί με τις φυσιολογικές ακολουθίες του συνόλου Εκπαίδευσης, έτσι ώστε να μάθει να τις ανακατασκευάζει με όσο το δυνατόν μικρότερο σφάλμα, αναγνωρίζοντας τα σημαντικότερα συστατικά των χαρακτηριστικών, μέσω της διαδικασίας κωδικοποίησης των δεδομένων εισόδου σε συμπιεσμένες αναπαραστάσεις.

3.6.1 Αρχικοποίηση Μοντέλου

Έχει αναφερθεί ότι, ένα μοντέλο Αυτοκωδικοποιητή ακολουθεί τη συμβατική αρχιτεκτονική των διασυνδεδεμένων επιπέδων των Τεχνητών Νευρωνικών Δικτύων και αποτελείται από δυο επιμέρους Τεχνητά Νευρωνικά Δίκτυα, τον «Κωδικοποιητή» και τον «Αποκωδικοποιητή».

Αρχικά, ορίζουμε τον «Κωδικοποιητή» με όνομα «Encoder» και εξάγουμε τη συνοπτική εικόνα των επιπέδων του:

```
Input []:
1  encoder = keras.models.Sequential(
2      name='Encoder',
3      layers=[
4      keras.Input(shape=(24,2)),
5      keras.layers.LSTM(8, activation='tanh', return_sequences=True),
6      keras.layers.LSTM(4, activation='tanh', return_sequences=True),
7      keras.layers.LSTM(2, activation='tanh',
8                        activity_regularizer=
9                        keras.regularizers.L1(l1=0.00015),
10                       return_sequences=False)
11 ])
12 encoder.summary()

Output []:
Model: "Encoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 24, 8)             352

 lstm_1 (LSTM)               (None, 24, 4)             208

 lstm_2 (LSTM)               (None, 2)                 56

=================================================================
Total params: 616
Trainable params: 616
Non-trainable params: 0
_________________________________________________________________
```

Πρώτον, παρατηρούμε ότι το επίπεδο εισόδου έχει οριστεί με διάσταση $(24,2)$, καθώς όλα τα οριστικά μετασχηματισμένα σύνολα δεδομένων αποτελούνται πλέον από εγγραφές οι οποίες είναι ακολουθίες 24 στοιχείων και 2 χαρακτηριστικών.

Δεύτερον, τα δομικά στοιχεία όλων των υπόλοιπων επιπέδων είναι νευρώνες τύπου LSTM, με συνάρτηση ενεργοποίησης την υπερβολική εφαπτομένη $tanh$, και η αρχικοποίηση όλων των παραμέτρων βαρών και σταθερών όρων αφήνεται να είναι η προεπιλεγμένη της βιβλιοθήκης Keras. Επιπλέον, όσον αφορά στις διαστάσεις χαρακτηριστικών κάθε επιπέδου LSTM, το πρώτο ενδιάμεσο επίπεδο έχει διάσταση 8, το δεύτερο ενδιάμεσο επίπεδο έχει διάσταση 4, και το τελικό επίπεδο εξόδου έχει διάσταση χαρακτηριστικών 2.

Τρίτον, για τα δυο ενδιάμεσα επίπεδα υπάρχει η δήλωση return_sequences=True, ενώ για το επίπεδο εξόδου η δήλωση return_sequences=False. Υπενθυμίζεται ότι η έξοδος ενός επαναληπτικού επιπέδου (RNN ή LSTM ή GRU κ.α.) μπορεί να είναι μια ολόκληρη ακολουθία, ή μπορεί να είναι μόνο το τελευταίο στοιχείο της τελικής ακολουθίας. Συνεπώς, με την παραπάνω δήλωση True, το πρώτο επίπεδο LSTM δέχεται τα δεδομένα εισόδου, τα οποία είναι ακολουθίες των 24 στοιχείων και 2 χαρακτηριστικών, τα αναλύει σύμφωνα με τρόπο που επεξηγήθηκε στο Κεφάλαιο 2, και στη συνέχεια εξάγει και πάλι ακολουθίες 24 στοιχείων, αλλά αυτή τη φορά στον εκτεταμένο χώρο των 8 χαρακτηριστικών. Έπειτα, το επόμενο επίπεδο LSTM, δέχεται ως είσοδο τις νέες ακολουθίες των 24 στοιχείων και 8 χαρακτηριστικών και εξάγει επίσης ακολουθίες 24 στοιχείων αλλά στον μειωμένο χώρο των 4 χαρακτηριστικών. Τελικά, το επίπεδο εξόδου LSTM τροφοδοτείται με τις ακολουθίες των 24 στοιχείων και 4 χαρακτηριστικών, τις αναλύει, και εξάγει μονάχα το τελευταίο στοιχείο των επεξεργασμένων ακολουθιών σε επίσης μειωμένο χώρο 2 χαρακτηριστικών, καθώς κατά τον ορισμό του χρησιμοποιήθηκε η δήλωση False. Η δήλωση return_sequences=True είναι αναγκαία όταν το επόμενο επίπεδο του δικτύου είναι επαναληπτικής φύσεως, καθώς απαιτεί ως είσοδο δεδομένα υπό μορφή ακολουθιών.

Τέταρτον, ειδικά για το τελευταίο επίπεδο εξόδου LSTM του «Κωδικοποιητή», πρέπει να σημειωθεί ότι αποτελεί το επίπεδο των κωδικοποιήσεων του μοντέλου AE-LSTM, δηλαδή εξάγει τις συμπιεσμένες αναπαραστάσεις των ακολουθιών εισόδου, με σκοπό να αναγνωρίσει τα σημαντικότερα συστατικά τους τα οποία αρκούν για να οδηγήσουν σε καλύτερες ανακατασκευές. Επιπλέον, αναφέρθηκε ότι έχει διάσταση χαρακτηριστικών 2, η οποία είναι ίδια με τη διάσταση των χαρακτηριστικών των ακολουθιών εισόδου, συνεπώς οι κωδικοποιήσεις έχουν ίδια διάσταση με τα δεδομένα εισόδου και το μοντέλο AE-LSTM ακολουθεί την αρχιτεκτονική ενός υπερπλήρους μοντέλου Αυτοκωδικοποιητή.

Πέμπτον, υπενθυμίζεται ότι στην περίπτωση των υπερπλήρων μοντέλων υπάρχει ο κίνδυνος το μοντέλο να μάθει απλά να αντιγράφει την είσοδο στην έξοδό του, χωρίς να εξάγει καμία χρήσιμη πληροφορία για τα δεδομένα, καθώς μπορεί να αγνοηθούν σημαντικά στοιχεία των δεδομένων και να μην αναγνωριστούν τα ουσιαστικά συστατικά των χαρακτηριστικών κατά την εκπαίδευσή του, λόγω της ελευθερίας όσον αφορά στον χώρο των διαστάσεων. Για να μετριαστεί αυτή η αδυναμία των υπερπλήρων μοντέλων, μια τεχνική που χρησιμοποιείται είναι η κανονικοποίηση (regularization) του επίπεδου των κωδικοποιήσεων. Με την παραπάνω τεχνική, εφαρμόζεται ένας επιπλέον μαθηματικός όρος στη συνάρτηση απώλειας για το συγκεκριμένο επίπεδο, έτσι ώστε να περιορίσει την «ελευθερία» των παραμέτρων του κατά τη διαδικασία της εκπαίδευσης, προσδίδοντας

σποραδικότητα (sparsity) στο χώρο των κωδικοποιήσεων, δηλαδή μειώνεται η διαθέσιμη διάσταση χαρακτηριστικών σε επίπεδο παραμέτρων. Ειδικότερα, για το μοντέλο AE-LSTM το οποίο αναπτύσσεται για το συγκεκριμένο πρόβλημα, εφαρμόζεται στο επίπεδο εξόδου του «Κωδικοποιητή» κανονικοποίηση παραμέτρου $L1 = 0.00015$, η τιμή της οποίας επίσης προέκυψε μέσω πειραματισμού.

Στη συνέχεια, ορίζουμε τον «Αποκωδικοποιητή» με όνομα «Decoder» και εξάγουμε τη συνοπτική εικόνα των επιπέδων του:

```
Input []:
1  decoder = keras.models.Sequential(
2      name='Decoder',
3      layers=[
4      keras.Input(shape=(2)),
5      keras.layers.RepeatVector(24),
6      keras.layers.LSTM(4, activation='tanh', return_sequences=True),
7      keras.layers.LSTM(8, activation='tanh', return_sequences=True),
8      keras.layers.TimeDistributed(keras.layers.Dense(2))
9  ])
10 decoder.summary()

Output []:
Model: "Decoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 repeat_vector (RepeatVector  (None, 24, 2)            0
 )

 lstm_3 (LSTM)               (None, 24, 4)             112

 lstm_4 (LSTM)               (None, 24, 8)             416

 time_distributed (TimeDistr  (None, 24, 2)            18
 ibuted)

=================================================================
Total params: 546
Trainable params: 546
Non-trainable params: 0
_________________________________________________________________
```

Πρώτον, υπενθυμίζεται ότι η έξοδος του «Κωδικοποιητή» είναι κωδικοποιήσεις υπό μορφή διανυσμάτων των 2 χαρακτηριστικών, επομένως, το επίπεδο εισόδου (Input) του «Αποκωδικοποιητή» δέχεται την αντίστοιχη διάσταση δεδομένων.

Δεύτερον, από τη στιγμή που ενσωματώνονται επίπεδα LSTM στην αρχιτεκτονική του δικτύου, είναι απαραίτητο τα δεδομένα να τροφοδοτούνται σε αυτά υπό μορφή ακολουθιών. Συνεπώς, το πρώτο ενδιάμεσο επίπεδο RepeatVector() αποτελεί «γέφυρα» μεταξύ του «Κωδικοποιητή» και το «Αποκωδικοποιητή», καθώς αντιγράφει το προηγούμενο διάνυσμα των 2 χαρακτηριστικών 24 φορές, παράγοντας έτσι μια ακολουθία 24 στοιχείων και 2 χαρακτηριστικών.

Τρίτον, το επόμενο επίπεδο είναι τύπου LSTM, το οποίο δέχεται τις παραπάνω ακολουθίες 24 στοιχείων και 2 χαρακτηριστικών, τις επεξεργάζεται, και εξάγει ακολουθίες (return_sequences=True) των 24 στοιχείων στον εκτεταμένο χώρο των 4 χαρακτηριστικών. Στη συνέχεια, το επόμενο επίπεδο είναι επίσης τύπου LSTM, το οποίο δέχεται τις ακολουθίες των 24 στοιχείων και 4 χαρακτηριστικών, τις αναλύει, και εξάγει ακολουθίες των 24 στοιχείων στον ευρύτερο εκτεταμένο χώρο των 8 χαρακτηριστικών. Παρατηρείται ότι το πλήθος των διαστάσεων ακολουθεί συμμετρικά το αντίστοιχο πλήθος διαστάσεων των επιπέδων του «Κωδικοποιητή».

Τέταρτον, το επίπεδο εξόδου TimeDistributed() δέχεται τις ακολουθίες των 24 στοιχείων (γραμμών) και 8 χαρακτηριστικών (στηλών), και για κάθε χρονικό βήμα, δηλαδή για κάθε στοιχείο της ακολουθίας, εφαρμόζει ένα πλήρως διασυνδεδεμένο γραμμικό επίπεδο διάστασης 2 μέσω της δήλωσης Dense(2). Συνεπώς, η τελική έξοδος του «Αποκωδικοποιητή» είναι μια ακολουθία 24 στοιχείων και 2 χαρακτηριστικών, δηλαδή εξάγει δεδομένα ίδιας διάστασης με αυτή της εισόδου του «Κωδικοποιητή».

Τελικά, ορίζουμε το οριστικό μοντέλο του Αυτοκωδικοποιητή με όνομα «Autoencoder» συνδυάζοντας τα παραπάνω δίκτυα του «Κωδικοποιητή» (encoder) και του «Αποκωδικοποιητή» (decoder) με τον παρακάτω τρόπο:

```
Input []:
1  autoencoder = keras.Sequential(
2      name='Autoencoder',
3      layers=[encoder, decoder]
4  )
```

Προς το παρόν, δεν είναι δυνατόν να εξαχθεί η συνοπτική εικόνα των επιπέδων του τελικού μοντέλου «Autoencoder», καθώς είναι αναγκαίος ο ορισμός του επιπέδου εισόδου και η διάσταση των δεδομένων που θα τροφοδοτηθούν στο μοντέλο, κάτι που είναι ήδη

ορισμένο στα επιμέρους δίκτυά του, encoder και decoder, όπου και έγινε η εξαγωγή των συνόψεών τους. Ωστόσο, μετά την εκπαίδευση του μοντέλου, εφόσον θα έχει πλέον αναγνωριστεί η διάσταση των δεδομένων εισόδου, μπορεί να γίνει κλήση της συνοπτικής εικόνας των επιπέδων του.

Εν κατακλείδι, έχει αρχικοποιηθεί ένα υπερπλήρες μοντέλο Αυτοκωδικοποιητή με ενσωματωμένα επαναληπτικά επίπεδα Μακράς Βραχυπρόθεσμης Μνήμης (AE-LSTM), του οποίου η δομή, ως προς τη διάσταση των επιμέρους επιπέδων του «Κωδικοποιητή» και του «Αποκωδικοποιητή» ακολουθεί τη συμμετρία $2-8-4-2-4-8-2$. Συγκεκριμένα, δέχεται ακολουθίες διάστασης $(24,2)$, στη συνέχεια τις αναλύει σε κωδικοποιήσεις διανυσμάτων διάστασης 2, και τελικά εξάγει ακολουθίες επίσης διάστασης $(24,2)$, οι οποίες αποτελούν ανακατασκευές των αρχικών. Επιπλέον, στις κωδικοποιήσεις του μοντέλου εφαρμόζεται η μέθοδος κανονικοποίησης παραμέτρου $L1$, οδηγώντας σε ένα «Σποραδικό» μοντέλο Αυτοκωδικοποιητή (Sparse Autoencoder).

3.6.2 Σύνταξη Μοντέλου

Έπειτα, ακολουθεί η σύνταξη του μοντέλου AE-LSTM έτσι ώστε να ενσωματωθούν στη διαδικασία της εκπαίδευσης η συνάρτηση απώλειας, ο αλγόριθμος βελτιστοποίησης των παραμέτρων και η επιπλέον προαιρετική μετρική ακρίβειας:

```
Input []:
1  loss = keras.losses.MeanSquaredError()
2  optim = keras.optimizers.Adam(learning_rate=0.001)
3  metr = keras.metrics.RootMeanSquaredError()

4  autoencoder.compile(loss=loss, optimizer=optim, metrics=metr)
```

Αρχικά, επιλέγεται ως συνάρτηση απώλειας η συνάρτηση του Μέσου Τετραγωνικού Σφάλματος (Mean Squared Error - MSE), η οποία υπολογίζει το σφάλμα μεταξύ των προσωρινών προβλέψεων κατά την εκπαίδευση και των πραγματικών ακολουθιών που πρέπει να μάθει να ανακατασκευάζει το μοντέλο, με τον παρακάτω μαθηματικό τύπο:

$$MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} (y - \hat{y})^2$$

Όπου:

- y είναι η πραγματική αρχική είσοδος
- $\hat{y}$ είναι η εξαγόμενη ανακατασκευή του μοντέλου

- n είναι το πλήθος των στοιχείων για τα οποία υπολογίζεται το συνολικό σφάλμα

Πρέπει να σημειωθεί ότι, από τη στιγμή που τα δεδομένα τα οποία επεξεργάζονται από το μοντέλο είναι πολυδιάστατα, ο υπολογισμός του σφάλματος γίνεται από την τελευταία διάσταση προς την πρώτη διάσταση. Υπενθυμίζεται ότι, η εκπαίδευση του μοντέλου γίνεται με παρτίδες στοιχείων, συνεπώς για την περίπτωσή μας, τα δεδομένα έχουν διάσταση $(32{,}24{,}2)$, δηλαδή σε κάθε επαναληπτικό βήμα αναλύονται 32 ακολουθίες, όπου κάθε ακολουθία έχει 24 στοιχεία, και κάθε στοιχείο αναλύεται σε 2 χαρακτηριστικά. Συνεπώς, με αντίστροφο τρόπο γίνεται και η εξαγωγή του σφάλματος, δηλαδή υπολογίζεται, βάσει του μαθηματικού τύπου MSE, το μέσο σφάλμα για τα 2 χαρακτηριστικά κάθε στοιχείου για κάθε ακολουθία, στη συνέχεια υπολογίζεται το μέσο σφάλμα για ολόκληρη την ακολουθία των 24 στοιχείων, για κάθε ακολουθία της παρτίδας ξεχωριστά, και τελικά υπολογίζεται το μέσο σφάλμα ολόκληρης της παρτίδας των 32 ακολουθιών.

Ωστόσο, ο παραπάνω τύπος δεν αποτελεί την τελική συνάρτηση απώλειας του μοντέλου, καθώς κατά την αρχικοποίησή του εφαρμόστηκε ένας επιπλέον όρος παραμέτρου $L1$ στο επίπεδο των κωδικοποιήσεων μέσω της μεθόδου κανονικοποίησης. Αυτό έχει ως αποτέλεσμα η παραπάνω συνάρτηση MSE να τροποποιείται στην παρακάτω Συνάρτηση Απώλειας (Loss Function - LF) η οποία ακολουθεί τον γενικότερο μαθηματικό τύπο:

$$LF = \frac{1}{n} \cdot \sum_{i=1}^{n} (y - \hat{y})^2 + \lambda \cdot \sum_{i=1}^{m} |z_i|$$

Όπου:

- y είναι η πραγματική αρχική είσοδος
- $\hat{y}$ είναι η προβλεπόμενη έξοδος του μοντέλου
- n είναι το πλήθος των στοιχείων για τα οποία υπολογίζεται το συνολικό σφάλμα
- λ είναι η παράμετρος κανονικοποίησης
- m είναι το πλήθος των στοιχείων του επιπέδου που κανονικοποιείται
- z_i είναι οι τιμές των στοιχείων του επιπέδου που κανονικοποιείται

Ειδικότερα, το μοντέλο AE-LSTM του προβλήματος κανονικοποιείται ως προς τις κωδικοποιήσεις του επιπέδου εξόδου του «Κωδικοποιητή» με παράμετρο κανονικοποίησης $\lambda = 0.00015$. Υπενθυμίζεται ότι, κατά την επαναληπτική εκπαίδευση του μοντέλου, εκτιμάται η συνεισφορά της κάθε παραμέτρου του δικτύου στο συνολικό σφάλμα, μέσω του υπολογισμού των μερικών παραγώγων του συνολικού σφάλματος ως προς κάθε παράμετρο. Συνεπώς, το επίπεδο των κωδικοποιήσεων επηρεάζεται κατά τον υπολογισμό των παραγώγων από τον πρόσθετο όρο στη συνάρτηση απώλειας, και έτσι το μοντέλο

τείνει να ρυθμίσει κατάλληλα τις παραμέτρους του επιπέδου αυτού έτσι ώστε να ελαχιστοποιηθεί το σφάλμα, με αποτέλεσμα οι τιμές των κωδικοποιήσεων να περιορίζονται. Επομένως, η υπερπλήρης φύση του μοντέλου μετριάζεται με την εξαγωγή «σποραδικών» αναπαραστάσεων των χαρακτηριστικών, έτσι ώστε να αναγνωριστούν μονάχα τα ουσιώδη στοιχεία τους, τα οποία αρκούν για να οδηγήσουν σε καλύτερες ανακατασκευές των δεδομένων.

Στη συνέχεια, ο αλγόριθμος βελτιστοποίησης για την ενημέρωση των παραμέτρων σε κάθε επαναληπτικό βήμα της εκπαίδευσης, επιλέχθηκε να είναι ο Adam με ρυθμό μάθησης $\eta = 0.001$, ενώ ως επιπλέον μετρική για την παρακολούθηση της προόδου εκπαίδευσης του μοντέλου, ορίστηκε το Ριζικό Μέσο Τετραγωνικό Σφάλμα (Root Mean Squared Error - RMSE), το οποίο υπολογίζεται με ανάλογο τρόπο όπως το σφάλμα MSE, ως προς τις διαστάσεις των στοιχείων κάθε παρτίδας δεδομένων, σύμφωνα με τον παρακάτω μαθηματικό τύπο:

$$RMSE = \sqrt{\frac{1}{n} \cdot \sum_{i=1}^{n} (y - \hat{y})^2}$$

Όπου:

- y είναι η πραγματική αρχική είσοδος
- $\hat{y}$ είναι η εξαγόμενη ανακατασκευή του μοντέλου
- n είναι το πλήθος των στοιχείων για τα οποία υπολογίζεται το συνολικό σφάλμα

Μια κύρια διαφορά μεταξύ των MSE και RMSE είναι πως τα αποτέλεσμα του RMSE έχουν ίδια τάξη μονάδων μέτρησης με αυτές των δεδομένων εισόδου, ενώ τα αποτελέσματα του MSE υπολογίζονται για το τετράγωνο της μονάδας μέτρησής τους. Συνεπώς, το RMSE είναι μια καλή μετρική για να εξεταστεί, η οποία αναμένεται να ελαχιστοποιηθεί παράλληλα με την κανονικοποιημένη συνάρτηση απώλειας MSE.

Η προτίμηση των παραπάνω τριών υπερπαραμέτρων του μοντέλου επιτεύχθηκε μέσα από πειραματισμούς και σαφώς δεν αποτελούν απόλυτη επιλογή για την ανάπτυξη αντίστοιχων μοντέλων, καθώς κάθε πρόβλημα και περίπτωση δεδομένων είναι διαφορετική, και ορισμένες τεχνικές λειτουργούν καλύτερα από κάποιες άλλες.

3.6.3 Εκπαίδευση Μοντέλου

Το τελικό βήμα, μετά την αρχικοποίηση και τη σύνταξη του μοντέλου AE-LSTM είναι η κατάλληλη κλήση για να ξεκινήσει η εκπαιδευτική διαδικασία, ως εξής:

```
Input []:
1  history = autoencoder.fit(ds_left_norm_train_seq,
2                            ds_left_norm_train_seq,
3                            epochs=100, batch_size=32,
4                            validation_split=0.2,
5                            shuffle=True, verbose=1)

Output []:
Epoch 1/100
105/105 [==============================] - 23s 84ms/step - loss: 0.1284 -
root_mean_squared_error: 0.3583 - val_loss: 0.0672 -
val_root_mean_squared_error: 0.2590
Epoch 2/100
105/105 [==============================] - 7s 67ms/step - loss: 0.0510 -
root_mean_squared_error: 0.2256 - val_loss: 0.0721 -
val_root_mean_squared_error: 0.2683
Epoch 3/100
105/105 [==============================] - 6s 58ms/step - loss: 0.0270 -
root_mean_squared_error: 0.1638 - val_loss: 0.0409 -
val_root_mean_squared_error: 0.2020
Epoch 4/100
.........
Epoch 98/100
105/105 [==============================] - 6s 57ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0409 - val_loss: 9.5984e-04 -
val_root_mean_squared_error: 0.0305
Epoch 99/100
105/105 [==============================] - 8s 72ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0409 - val_loss: 9.6345e-04 -
val_root_mean_squared_error: 0.0306
Epoch 100/100
105/105 [==============================] - 6s 62ms/step - loss: 0.0017 -
root_mean_squared_error: 0.0409 - val_loss: 9.4062e-04 -
val_root_mean_squared_error: 0.0302
```

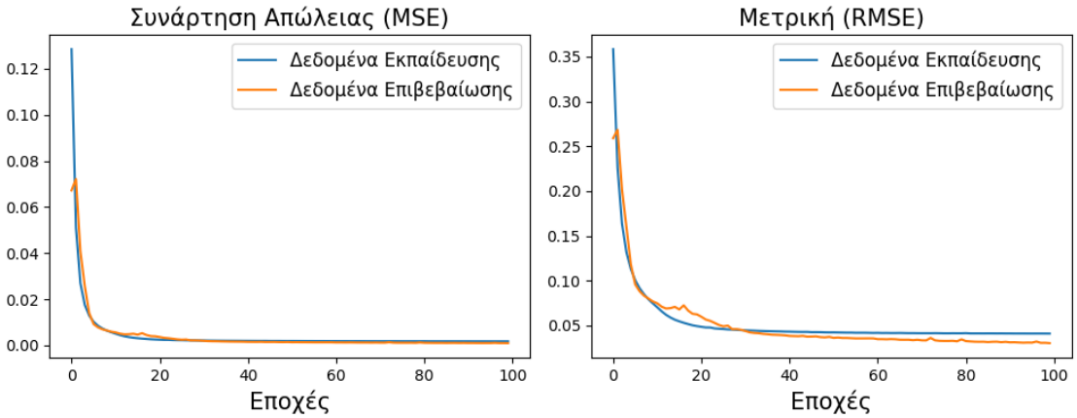
Αρχικά, να σημειωθεί ότι, ως σύνολο χαρακτηριστικών και ως σύνολο ετικετών ορίστηκε ολόκληρο το σύνολο δεδομένων Εκπαίδευσης της φυσιολογικής κατάστασης της αριστερής μονάδας ρουλεμάν, καθώς το ζητούμενο είναι η επιτυχημένη ανακατασκευή των ακολουθιών εισόδου, δηλαδή οι ακολουθίες εξόδου να είναι όσο το δυνατόν ίδιες με τις ακολουθίες εισόδου.

Επίσης, επιλέγεται και πάλι, ως σύνολο Επιβεβαίωσης για την παρακολούθηση της προόδου της εκπαιδευτικής διαδικασίας, ένα τυχαίο ποσοστό του συνόλου Εκπαίδευσης της τάξης του 20%. Επιπλέον, τα δεδομένα ανακατεύονται τυχαία, χωρίζονται σε παρτίδες

των 32 ακολουθιών και με διάρκεια εκπαίδευσης τις 100 εποχές, περνάνε διαδοχικά από τα επίπεδα του δικτύου για ανάλυση και επεξεργασία.

Τελικά, στο τελευταίο βήμα της εκπαίδευσης παρατηρείται ότι για το σύνολο Εκπαίδευσης, η κανονικοποιημένη συνάρτηση απώλειας MSE (loss) έχει ελαχιστοποιηθεί στο 0.0017 και το σφάλμα RMSE στο 0.0409, ενώ για το σύνολο Επιβεβαίωσης, οι αντίστοιχες τιμές είναι 0.00094062 και 0.0302. Οι μικρότερες τιμές του συνόλου Επιβεβαίωσης είναι πιθανόν πως οφείλονται στην κανονικοποίηση του μοντέλου, καθώς αποτελεί γενικότερα τεχνική για την αντιμετώπιση της υπερπροσαρμογής του στα δεδομένα Εκπαίδευσης, για τα οποία λαμβάνεται υπόψη και ο επιπλέον όρος κανονικοποίησης στον υπολογισμό του σφάλματος, κάτι που δεν συμβαίνει για τα δεδομένα Επιβεβαίωσης.

Για μια καλύτερη εποπτεία των παραπάνω τιμών, καθ' όλη τη διάρκεια της εκπαιδευτικής διαδικασίας του μοντέλου AE-LSTM, παρουσιάζονται παρακάτω τα διαγράμματα προόδου για την κανονικοποιημένη συνάρτηση απώλειας MSE και τη μετρική συνάρτηση σφάλματος RMSE:



Εικόνα 66. Διαγράμματα Προόδου Σφάλματος και Ακρίβειας Μοντέλου Δεύτερης Εφαρμογής

Επομένως, παρατηρείται μια επιθυμητή συμπεριφορά των τιμών σφάλματος, αλλά και των τιμών RMSE, οι οποίες ελαχιστοποιούνται παράλληλα, τόσο για τα δεδομένα Εκπαίδευσης, όσο και για τα δεδομένα Επιβεβαίωσης. Συνεπώς, η εκπαίδευση του μοντέλου AE-LSTM θεωρείται επιτυχής και μπορούμε να προχωρήσουμε στην αξιολόγησή του για τον έλεγχο των επιδόσεών του.

Επίσης, από τη στιγμή που πραγματοποιήθηκε η εκπαίδευση του μοντέλου «Autoencoder», μπορεί να γίνει η εξαγωγή της συνοπτικής εικόνας των επιπέδων του ως εξής:

```
Input []:
1  autoencoder.summary()

Output []:
Model: "Autoencoder"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Encoder (Sequential)        (None, 2)                 616

 Decoder (Sequential)        (None, 24, 2)             546

=================================================================
Total params: 1,162
Trainable params: 1,162
Non-trainable params: 0
_________________________________________________________________
```

Οι πληροφορίες που παίρνουμε από την παραπάνω σύνοψη είναι περιορισμένες καθώς η αρχικοποίηση του μοντέλου AE-LSTM έγινε σε βήματα, και τα ονόματα των επιπέδων «Encoder» και «Decoder» είναι αυτά που ορίστηκαν κατά την αρχικοποίηση του «Κωδικοποιητή» και «Αποκωδικοποιητή» αντίστοιχα.

Τέλος, το εκπαιδευμένο μοντέλο, αλλά και οι τιμές σφάλματος και μετρικής μπορούν να αποθηκευτούν με τον παρακάτω τρόπο:

```
Input []:
1  # save autoencoder
2  keras.models.save_model(autoencoder, '/content/drive/MyDrive/
3                          Colab Notebooks/thesis/
4                          saved_models/autoencoder')
5
6  # load autoencoder
7  autoencoder = keras.models.load_model('/content/drive/MyDrive/
8                          Colab Notebooks/thesis/
9                          saved_models/autoencoder')
10
11 # save history
12 hist_df = pd.DataFrame(history.history)
13 hist_df.to_csv('/content/drive/MyDrive/Colab Notebooks/
14                 thesis/saved_models/autoencoder.csv', index=False)
15
16 # load history
17 datadf = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
18                       thesis/saved_models/autoencoder.csv')
```

```
19 datadf
```

Output []:

	loss	root_mean_squared_error	val_loss	val_root_mean_squared_error
0	0.128440	0.358281	0.067204	0.258992
1	0.051037	0.225607	0.072106	0.268297
2	0.026966	0.163790	0.040854	0.201962
3	0.017513	0.131795	0.025671	0.160110
4	0.012904	0.113029	0.014021	0.118285
...	...	...	...	...
95	0.001723	0.040928	0.000976	0.030815
96	0.001724	0.040947	0.001054	0.032042
97	0.001723	0.040944	0.000960	0.030548
98	0.001721	0.040924	0.000963	0.030622
99	0.001718	0.040895	0.000941	0.030237

100 rows × 4 columns

3.7 Αξιολόγηση Μοντέλου

Το επόμενο βήμα, μετά την επιτυχημένη εκπαίδευση του μοντέλου AE-LSTM, είναι η αξιολόγησή του για να επιβεβαιωθούν η ικανότητα της γενίκευσής του και οι επιδόσεις του, όταν καλείται να ανακατασκευάσει νέες ακολουθίες τις οποίες δεν έχει συναντήσει προηγουμένως.

Αναφέρθηκε ότι, η αξιολόγηση της λειτουργικότητας ενός μοντέλου το οποίο ακολουθεί την προσέγγιση των ανακατασκευών, εξαρτάται από το σφάλμα που προκύπτει από τη σύγκριση των αρχικών δεδομένων εισόδου με τις ανακατασκευασμένες εξόδους του μοντέλου. Σαφώς, κατά την εκπαίδευσή του, αυτό το γενικότερο σφάλμα ανακατασκευής ελαχιστοποιείται. Επομένως, δεδομένα ίδιας συμπεριφοράς με αυτά πάνω στα οποία εκπαιδεύτηκε θα παρουσιάζουν παρόμοια τάξη σφάλματος, ενώ δεδομένα διαφορετικών συμπεριφορών θα έχουν ως αποτέλεσμα μεγαλύτερο σφάλμα.

Πάνω στην παραπάνω λογική στηρίζεται και το μοντέλο AE-LSTM του προβλήματος το οποίο αναπτύχθηκε και εκπαιδεύτηκε. Συνεπώς, αναμένουμε ακολουθίες φυσιολογικής κατάστασης να εξάγουν μικρό σφάλμα, ενώ ακολουθίες μη-φυσιολογικής η επιβαρυμένης κατάστασης του ρουλεμάν να παρουσιάζουν μεγαλύτερο σφάλμα. Επομένως, κατά την

αξιολόγηση υπάρχει ανάγκη να βρεθεί εκείνο το όριο (threshold) το οποίο θα διαχωρίζει τα αποδεκτά από τα μη-αποδεκτά σφάλματα ανακατασκευής, αναγνωρίζοντας έτσι τις περιπτώσεις ανωμαλίας του βιομηχανικού εξοπλισμού στα πλαίσια της Προβλεπτικής Συντήρησης.

Για την παρακάτω διαδικασία αξιολόγησης που ακολουθεί, ορίζεται αρχικά το όριο ανακατασκευής με έναν συμβατικό τρόπο μέσα από το ίδιο το σύνολο Εκπαίδευσης (στο οποίο συμπεριλαμβάνεται και το σύνολο Επιβεβαίωσης) των φυσιολογικών ακολουθιών της αριστερής μονάδας ρουλεμάν (ds_left_norm_train_seq), και στη συνέχεια το όριο αυτό χρησιμοποιείται για την αναγνώριση μη-φυσιολογικών συμπεριφορών στα υπόλοιπα τέσσερα σύνολα δεδομένων τα οποία επεξεργάστηκαν προηγουμένως. Δηλαδή, το σύνολο Αξιολόγησης των φυσιολογικών ακολουθιών της αριστερής μονάδας ρουλεμάν (ds_left_norm_test_seq), το σύνολο των επιβαρυμένων ακολουθιών της αριστερής μονάδας ρουλεμάν (ds_left_abnorm_seq), το σύνολο των φυσιολογικών ακολουθιών της δεξιάς μονάδας ρουλεμάν (ds_right_norm_seq), και το σύνολο των επιβαρυμένων ακολουθιών της δεξιάς μονάδας ρουλεμάν (ds_right_abnorm_seq).

Προτού ξεκινήσει η διαδικασία της αξιολόγησης ορίζουμε και πάλι τη συνάρτηση απώλειας που θα χρησιμοποιηθεί για την εξαγωγή των σφαλμάτων, αλλά και τη μετρική σε περίπτωση που χρειαστεί:

```
Input []:
1  loss = keras.losses.MeanSquaredError()
2  metr = keras.metrics.RootMeanSquaredError()
```

Να σημειωθεί ότι, η εκπαίδευση του μοντέλου AE-LSTM έγινε σύμφωνα με την κανονικοποιημένη συνάρτηση απώλειας MSE, η οποία οδηγεί σε μεγαλύτερες τιμές σφάλματος κατά την εκπαίδευση, ενώ για την εξαγωγή των σφαλμάτων θα χρησιμοποιηθεί ο κλασικός μαθηματικός τύπος MSE, όπως περιγράφηκε κατά τη σύνταξη του μοντέλου.

3.7.1 Συμβατική Αξιολόγηση

Αρχικά παρουσιάζεται η συμβατική γενική αξιολόγηση για καθένα από τα πέντε σύνολα δεδομένων που έχουν προεπεξεργαστεί σε ακολουθίες:

```
Input []:
1  print(autoencoder.evaluate(ds_left_norm_train_seq,
2                             ds_left_norm_train_seq))
3  print(autoencoder.evaluate(ds_left_norm_test_seq,
4                             ds_left_norm_test_seq))
```

```
print(autoencoder.evaluate(ds_left_abnorm_seq,
                           ds_left_abnorm_seq))
print(autoencoder.evaluate(ds_right_norm_seq,
                           ds_right_norm_seq))
print(autoencoder.evaluate(ds_right_abnorm_seq,
                           ds_right_abnorm_seq))
```

Output []:

```
131/131 [==============================] - 2s 17ms/step - loss: 0.0016 -
root_mean_squared_error: 0.0390
[0.0015612947754561901, 0.03898079693317413]
32/32 [==============================] - 0s 12ms/step - loss: 9.9016e-04 -
root_mean_squared_error: 0.0312
[0.0009901570156216621, 0.031224820762872696]
82/82 [==============================] - 1s 12ms/step - loss: 0.0568 -
root_mean_squared_error: 0.2379
[0.05679503083229065, 0.23793266713619232]
163/163 [==============================] - 2s 12ms/step - loss: 0.0035 -
root_mean_squared_error: 0.0589
[0.0035133520141243935, 0.05885517969727516]
82/82 [==============================] - 1s 12ms/step - loss: 0.5767 -
root_mean_squared_error: 0.7593
[0.57672518491745, 0.7592799067497253]
```

Συνεπώς, με μια πρώτη ματιά, βλέπουμε ότι για το σύνολο Εκπαίδευσης με τα δεδομένα της φυσιολογικής λειτουργίας του αριστερού ρουλεμάν, οι τιμές σφάλματος και μετρικής είναι αντίστοιχα 0.0016 και 0.0390, για το σύνολο Αξιολόγησης με τα δεδομένα της φυσιολογικής λειτουργίας του αριστερού ρουλεμάν είναι 0.00099016 ή 0.0010 και 0.0312, για το σύνολο με τα δεδομένα επιβαρυμένης κατάστασης του αριστερού ρουλεμάν είναι 0.0568 και 0.2379, για το σύνολο με τα δεδομένα φυσιολογικής κατάστασης του δεξιού ρουλεμάν είναι 0.0035 και 0.0589, και για το σύνολο με τα δεδομένα επιβαρυμένης κατάστασης του δεξιού ρουλεμάν είναι 0.5767 και 0.7593.

Επομένως, παρατηρούμε ότι όλα τα σύνολα της φυσιολογικής κατάσταση των μονάδων ρουλεμάν, συμπεριλαμβανομένου και του συνόλου Εκπαίδευσης, σε δεδομένα του οποίου εκπαιδεύτηκε το μοντέλο AE-LSTM, παρουσιάζουν τιμές σφάλματος (loss) στην τάξη του χιλιοστού, ενώ τα δεδομένα επιβάρυνσης έχουν γενικότερο σφάλμα ανακατασκευής σαφώς μεγαλύτερο. Ειδικά, για την αριστερή επιβαρυμένη κατάσταση έχουμε σφάλμα στην τάξη του εκατοστού, ενώ στην δεξιά επιβαρυμένη κατάσταση στην τάξη του δεκάτου, κάτι που δεν αποτελεί έκπληξη, καθώς αν δούμε και πάλι τις συγκριτικές χρονοσειρές της επιβαρυμένης λειτουργικής κατάστασης, οι τιμές της δεξιάς μονάδας ρουλεμάν είναι

εμφανώς υψηλότερες από αυτές της αριστερής μονάδας, συνεπώς ήταν αναμενόμενο το μεγαλύτερο σφάλμα ανακατασκευής για την περίπτωση του δεξιού ρουλεμάν. Ανάλογη συγκριτική συμπεριφορά παρουσιάζεται και στις τιμές της μετρικής, καθώς είναι εμφανής η διαφορά μεταξύ φυσιολογικής και επιβαρυμένης κατάστασης.

Ειδικότερα, για να πάρουμε τα «καθαρά» σφάλματα MSE των ανακατασκευών, αλλά και τις τιμές RMSE, χρειάζεται να πάρουμε πρώτα τις ίδιες τις ανακατασκευές του εκπαιδευμένου μοντέλου AE-LSTM για όλα τα σύνολα δεδομένων, και αυτό γίνεται ως εξής:

```
Input []:
ds_left_norm_train_seq_pred =
        autoencoder.predict(ds_left_norm_train_seq)
ds_left_norm_test_seq_pred =
        autoencoder.predict(ds_left_norm_test_seq)
ds_left_abnorm_seq_pred =
        autoencoder.predict(ds_left_abnorm_seq)
ds_right_norm_seq_pred =
        autoencoder.predict(ds_right_norm_seq)
ds_right_abnorm_seq_pred =
        autoencoder.predict(ds_right_abnorm_seq)

Output []:
131/131 [==============================] - 2s 17ms/step
32/32 [==============================] - 1s 16ms/step
82/82 [==============================] - 1s 14ms/step
163/163 [==============================] - 2s 12ms/step
82/82 [==============================] - 1s 12ms/step
```

Στη συνέχεια, με τη συνάρτηση MSE που έχουμε ορίσει ως loss στο τέλος του προηγούμενου υποκεφαλαίου, παίρνουμε τα «καθαρά» σφάλματα ανακατασκευών:

```
Input []:
print(loss(ds_left_norm_train_seq,
           ds_left_norm_train_seq_pred).numpy())
print(loss(ds_left_norm_test_seq,
           ds_left_norm_test_seq_pred).numpy())
print(loss(ds_left_abnorm_seq,
           ds_left_abnorm_seq_pred).numpy())
print(loss(ds_right_norm_seq,
           ds_right_norm_seq_pred).numpy())
print(loss(ds_right_abnorm_seq,
           ds_right_abnorm_seq_pred).numpy())

Output []:
0.0015195027
```

```
0.0009749895
0.056611948
0.0034639332
0.57650596
```

Παρατηρούμε ότι τα «καθαρά» σφάλματα ανακατασκευών είναι λίγο μικρότερα από αυτά που υπολογίστηκαν κατά την μέθοδο evaluate(), κάτι που πιθανόν οφείλεται σε εσωτερικούς υπολογισμούς και διαδικασίες της μεθόδου. Να σημειωθεί επίσης ότι, η μέθοδος evaluate() δεν λαμβάνει υπόψη τον πρόσθετο όρο κανονικοποίησης.

Έπειτα, με τη συνάρτηση RMSE που έχουμε ορίσει ως metr στο τέλος του προηγούμενου υποκεφαλαίου, υπολογίζουμε και τα σφάλματα της μετρικής:

```
Input []:
1  metr.reset_state()
2  metr.update_state(ds_left_norm_train_seq,
3                    ds_left_norm_train_seq_pred)
4  print(metr.result().numpy())
5
6  metr.reset_state()
7  metr.update_state(ds_left_norm_test_seq,
8                    ds_left_norm_test_seq_pred)
9  print(metr.result().numpy())
10
11 metr.reset_state()
12 metr.update_state(ds_left_abnorm_seq,
13                   ds_left_abnorm_seq_pred)
14 print(metr.result().numpy())
15
16 metr.reset_state()
17 metr.update_state(ds_right_norm_seq,
18                   ds_right_norm_seq_pred)
19 print(metr.result().numpy())
20
21 metr.reset_state()
22 metr.update_state(ds_right_abnorm_seq,
23                   ds_right_abnorm_seq_pred)
24 print(metr.result().numpy())

Output []:
0.0389808
0.031224823
0.23793265
0.05885519
```

```
0.7592799
```

Βλέπουμε ότι τα αποτελέσματα είναι ίδια με αυτά που εξάγει η μέθοδος evaluate(). Επίσης, οι μέθοδοι reset_state() και update_state() είναι αναγκαίες, καθώς η απλή κλήση της συνάρτησης metr, παρόμοια με αυτή της συνάρτησης loss παραπάνω, έχει ως αποτέλεσμα νέα αποτελέσματα σε κάθε επανάληψη του κελιού εντολών.

3.7.2 Ορισμός Ορίου Ανακατασκευής

Μια πρώτη οπτική στα σφάλματα ανακατασκευής προήλθε από τη συμβατική αξιολόγηση του μοντέλου AE-LSTM πάνω σε όλα τα σύνολα δεδομένων, ωστόσο δεν είναι αρκετή για να εξετάσουμε την ικανότητα γενίκευσης, τις επιδόσεις, και τις δυνατότητές του στα πλαίσια των ανακατασκευών.

Όπως αναφέρθηκε, παρακάτω θα εξεταστεί το ίδιο το σύνολο Εκπαίδευσης των φυσιολογικών δεδομένων της αριστερής μονάδας ρουλεμάν για να οριστεί ένα συμβατικό όριο σφάλματος ανακατασκευής για την αναγνώριση μη-φυσιολογικών συμπεριφορών, και κατ' επέκταση ανωμαλιών στη λειτουργία του βιομηχανικού εξοπλισμού. Επίσης, υπενθυμίζεται ότι όλες οι ανακατασκευές των ακολουθιών (κατάληξη ονομάτων μεταβλητών _pred) έχουν καλεστεί στο προηγούμενο υποκεφάλαιο μέσω της μεθόδου predict().

Το πρώτο βήμα είναι η δημιουργία μιας λίστας (πρόθεμα loss_) με όλα τα 4162 σφάλματα ανακατασκευής κάθε ακολουθίας του συνόλου Εκπαίδευσης των φυσιολογικών δεδομένων της αριστερής μονάδας ρουλεμάν, σύμφωνα με τον παρακάτω τρόπο:

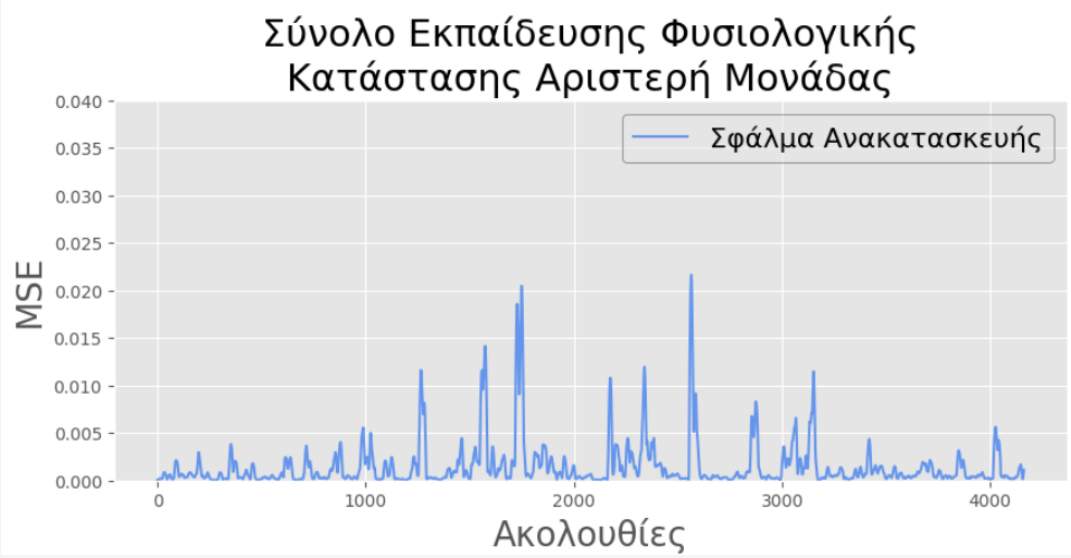
```
Input []:
1  print(ds_left_norm_train_seq.shape)
2  loss_ds_left_norm_train_seq = []
3  for i in range(len(ds_left_norm_train_seq)):
4    x = loss(ds_left_norm_train_seq[i],
5             ds_left_norm_train_seq_pred[i]).numpy()
6    loss_ds_left_norm_train_seq.append(x)
7  print(len(loss_ds_left_norm_train_seq))

Output []:
(4162, 24, 2)
4162
```

Επίσης, μπορεί να γίνει και οπτικοποίηση όλων των σφαλμάτων ως εξής:

Input [ ]:

```
plt.style.use('ggplot')
plt.figure(figsize=(10,4))

plt.plot(loss_ds_left_norm_train_seq,
         label='Σφάλμα Ανακατασκευής',
         color='cornflowerblue')
plt.ylim(0,0.04)
plt.title('Σύνολο Εκπαίδευσης Φυσιολογικής\n
           Κατάστασης Αριστερή Μονάδας', fontsize='22')
plt.xlabel('Ακολουθίες', fontsize='20')
plt.ylabel('MSE', fontsize='20')
plt.legend(loc='upper right', fontsize='16',
           handlelength=2, framealpha=0.5, edgecolor='black')

fig.tight_layout()

plt.show()
```

Output [ ]:



Στη συνέχεια, η λίστα των σφαλμάτων ανακατασκευών μετασχηματίζεται σε αντικείμενο DataFrame της βιβλιοθήκης Pandas (κατάληξη _df) για την εξαγωγή των μεγεθών θέσης και διασποράς, σύμφωνα με τον παρακάτω τρόπο:

Input [ ]:

```
loss_ds_left_norm_train_seq_df = pd.DataFrame(
                                    loss_ds_left_norm_train_seq)
loss_ds_left_norm_train_seq_df.describe()
```

Output [ ]:

	0
count	4162.000000
mean	0.001520
std	0.002545
min	0.000016
25%	0.000278
50%	0.000576
75%	0.001675
max	0.021633

Ως συμβατικό όριο ανακατασκευής θα επιλεχθεί η μέγιστη τιμή 0.021633 από το σύνολο σφαλμάτων των δεδομένων Εκπαίδευσης των φυσιολογικών τιμών της αριστερής μονάδας ρουλεμάν. Ένα πιο αυστηρό όριο ανακατασκευής θα ήταν η μέση τιμή 0.001520 αυξημένη κατά μια μονάδα τυπικής απόκλισης 0.002545, δηλαδή το όριο 0.004065. Η επιλογή του κατάλληλου ορίου γίνεται γενικότερα και πάλι βάσει του προβλήματος που εξετάζεται κάθε φορά, αλλά και της περαιτέρω μελέτης της αξιολόγησης, η οποία μπορεί να οδηγήσει σε αναπροσαρμογή του ορίου για να ταιριάζει καλύτερα στις συνθήκες της περίπτωσης που εξετάζεται.

Θα μείνουμε στην επιλογή της μέγιστης τιμής 0.021633 ως το συμβατικό όριο ανακατασκευής, βασιζόμενοι στη λογική του ότι τα δεδομένα Εκπαίδευσης θεωρούνται τα αποδεκτά φυσιολογικά δεδομένα λειτουργίας, επομένως όλα τα επιμέρους σφάλματα ανακατασκευών των ακολουθιών είναι επίσης αποδεκτά.

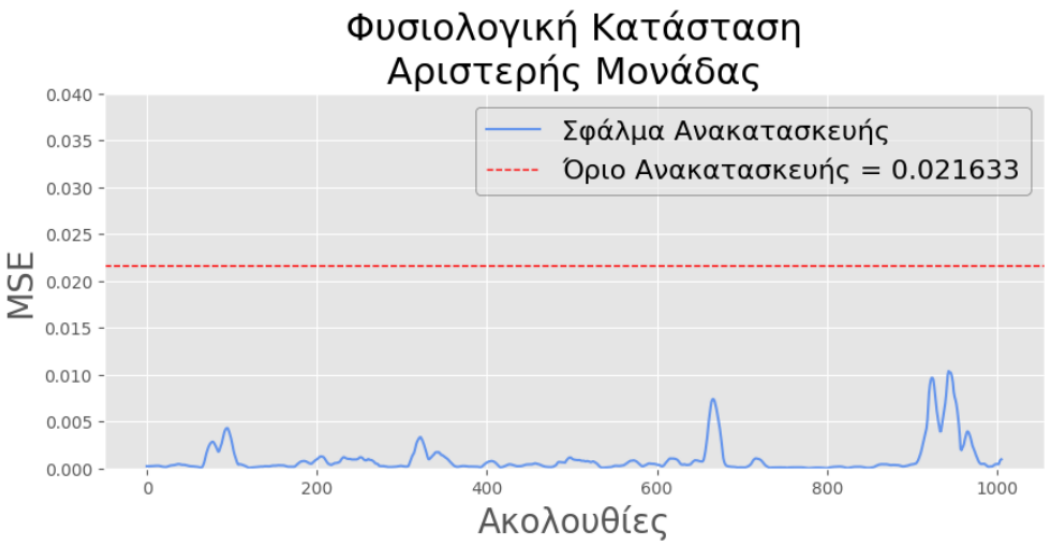
Αξίζει να αναφερθεί ότι, η παραπάνω μέση τιμή είναι το «καθαρό» στρογγυλοποιημένο αποτέλεσμα του γενικότερου σφάλματος του συνόλου δεδομένων Εκπαίδευσης, όπως υπολογίστηκε στο προηγούμενο υποκεφάλαιο μέσω της συνάρτησης MSE.

3.7.3 Αξιολόγηση Φυσιολογικής Κατάσταση Αριστερής Μονάδας

Παρακάτω ακολουθεί η αξιολόγηση, για την αναγνώριση τυχόν ανώμαλων ακολουθιών, του πρώτου από τα τέσσερα σύνολα δεδομένων, αυτό της φυσιολογικής λειτουργικής κατάστασης της αριστερής μονάδας ρουλεμάν, ή αλλιώς το σύνολο Αξιολόγησης, όπως χωρίστηκε κατά την προεπεξεργασία των δεδομένων, και το τελικό μοντέλο AE-LSTM δεν εκπαιδεύτηκε με αυτά.

Ο κώδικας Python για κάθε επόμενη διαδικασία βρίσκεται στο Παράρτημα Ι', αλλά μόνο για τη συγκεκριμένη περίπτωση. Η αξιολόγηση όλων των υπόλοιπων συνόλων δεδομένων επιτυγχάνεται με ανάλογο τρόπο με αυτόν που παρουσιάζεται στο Παράτημα Ι', όπου χρειάζεται προσοχή στην προσαρμογή των ονομάτων μεταβλητών, των διαστάσεων αξόνων σε κάποια διαγράμματα και στην αλλαγή αριθμών διάστασης σε ειδικές περιπτώσεις.

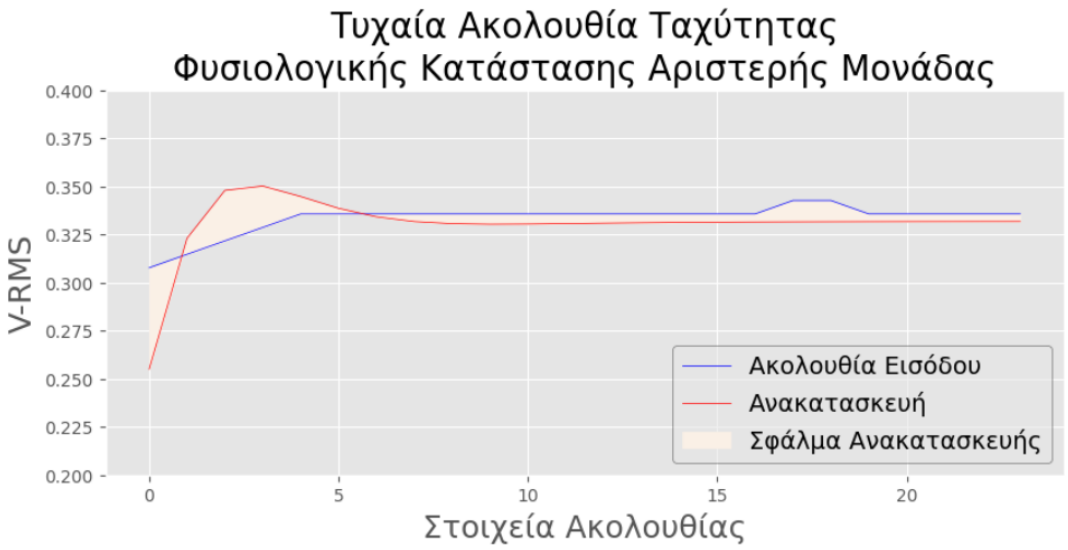
Αρχικά, υπολογίζονται για όλες τις 1006 ακολουθίες του συνόλου δεδομένων τα αντίστοιχα σφάλματα ανακατασκευής και οπτικοποιούνται παρακάτω:



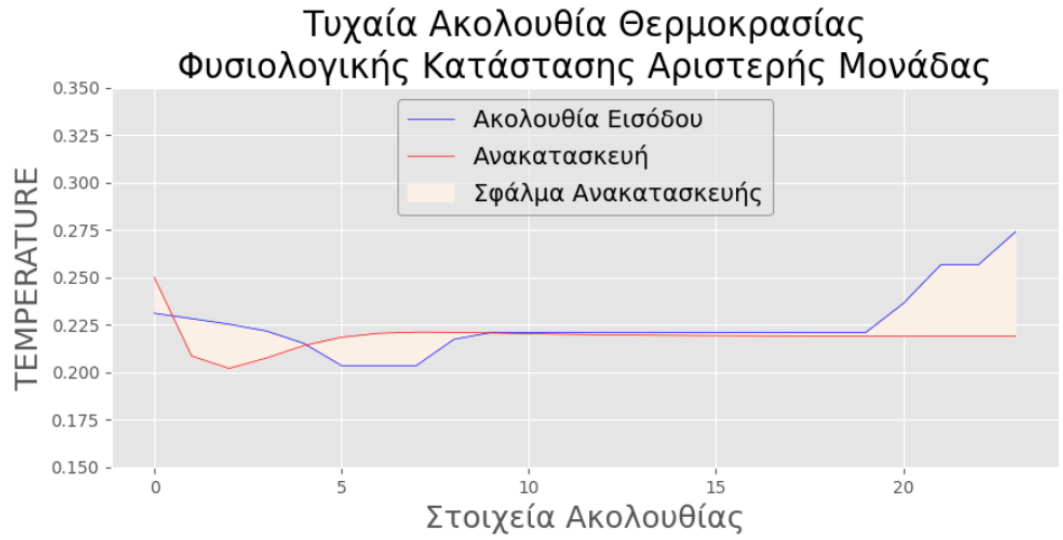
Εικόνα 67. Σφάλματα Ανακατασκευής Φυσιολογικής Κατάστασης Αριστερής Μονάδας Ρουλεμάν

Παρατηρούμε ότι κάθε ακολουθία φυσιολογικής κατάστασης αναγνωρίστηκε ως αποδεκτή, και μάλιστα τα αντίστοιχα σφάλματα ανακατασκευών έχουν τιμές αρκετά χαμηλότερες από το ορισμένο όριο ανακατασκευής 0.021633.

Στη συνέχεια, επιλέγεται τυχαία μια ακολουθία του συνόλου δεδομένων, ειδικότερα η 585η στη σειρά, παρουσιάζονται οι τιμές της ταχύτητας V-RMS και της θερμοκρασίας TEMPERATURE της ακολουθίας αυτής, και εξετάζονται μαζί με τις ανακατασκευές τους:



Εικόνα 68. Ανακατασκευή Ταχύτητας Τυχαίας Ακολουθίας Φυσιολογικής Κατάστασης Αριστερής Μονάδας Ρουλεμάν

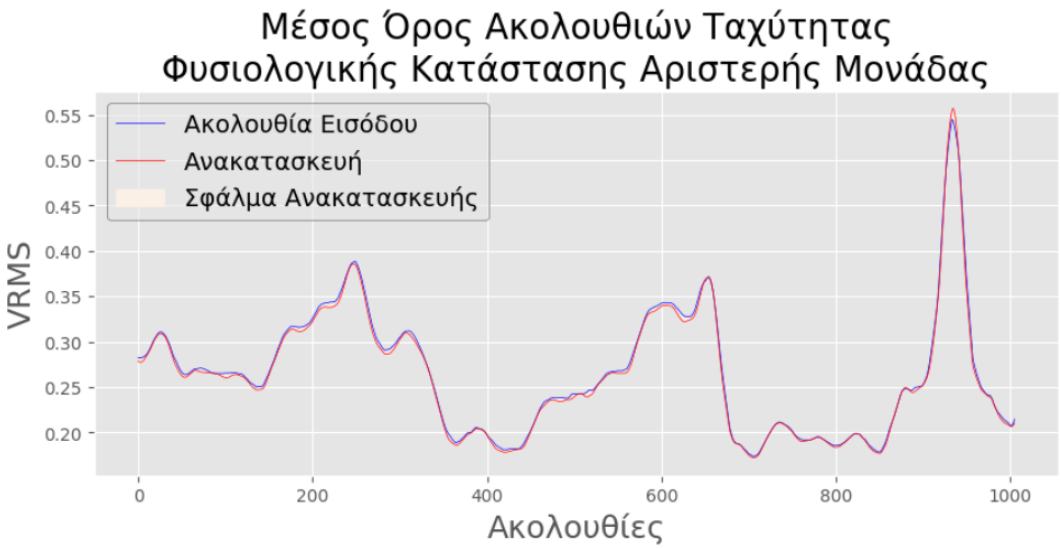


Εικόνα 69. Ανακατασκευή Θερμοκρασίας Τυχαίας Ακολουθίας Φυσιολογικής Κατάστασης Αριστερής Μονάδας Ρουλεμάν

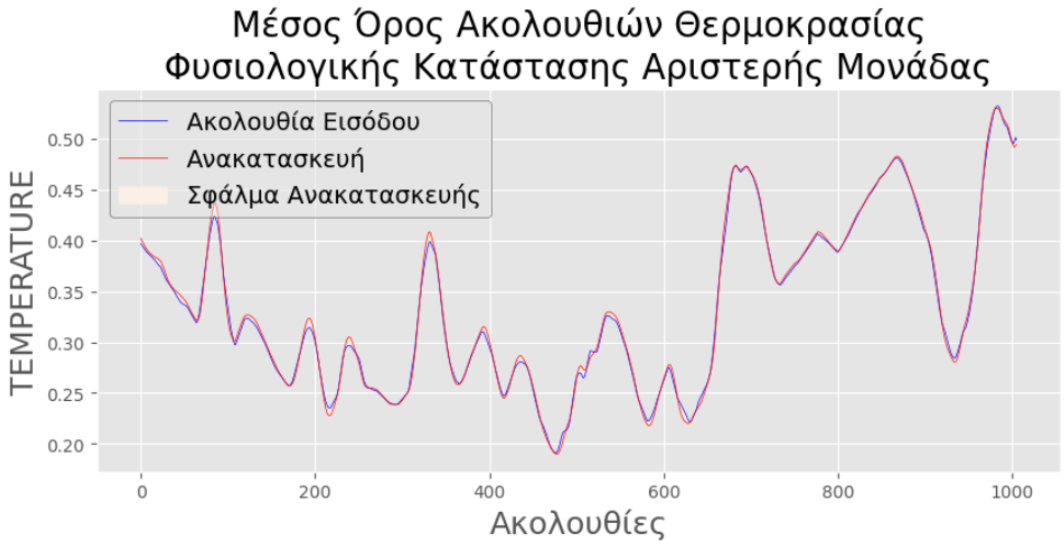
Τα αντίστοιχα επιμέρους σφάλματα για τις παραπάνω ανακατασκευές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE είναι 0.00019302306 και 0.0003537076, ωστόσο δεν μπορούν αυτά τα επιμέρους σφάλματα να συγκριθούν απόλυτα με το γενικό ορισμένο όριο ανακατασκευής 0.021633, καθώς το τελευταίο υπολογίζεται από τον συνδυασμό ταχύτητας και θερμοκρασίας. Συνεπώς ο μέσος όρος των δυο παραπάνω επιμέρους τιμών, δηλαδή η τιμή 0.00027336533, αντιστοιχεί στο πραγματικό συνολικό σφάλμα ανακατασκευής της ακολουθίας.

Βλέπουμε επομένως ότι οι ανακατασκευές των φυσιολογικών ακολουθιών ταχύτητας και θερμοκρασίας ακολουθούν όσο το δυνατόν καλύτερα τις αρχικές ακολουθίες εισόδου, και κατ' επέκταση το αντίστοιχο σφάλμα ανακατασκευής ανήκει στα πλαίσια των αποδεκτών σφαλμάτων.

Για μια γενικότερη εποπτεία των ανακατασκευών, παρουσιάζονται παρακάτω οι χρονοσειρές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE, όπου κάθε στοιχείο των χρονοσειρών αντιστοιχεί στον μέσο όρο των 24 στοιχείων κάθε επιμέρους ακολουθίας ταχύτητας και θερμοκρασίας:



Εικόνα 70. Μέσος Όρος Ακολουθιών Ανακατασκευής Ταχύτητας Φυσιολογικής Κατάστασης Αριστερής Μονάδας Ρουλεμάν



Εικόνα 71. Μέσος Όρος Ακολουθιών Ανακατασκευής Θερμοκρασίας Φυσιολογικής Κατάστασης Αριστερής Μονάδας Ρουλεμάν

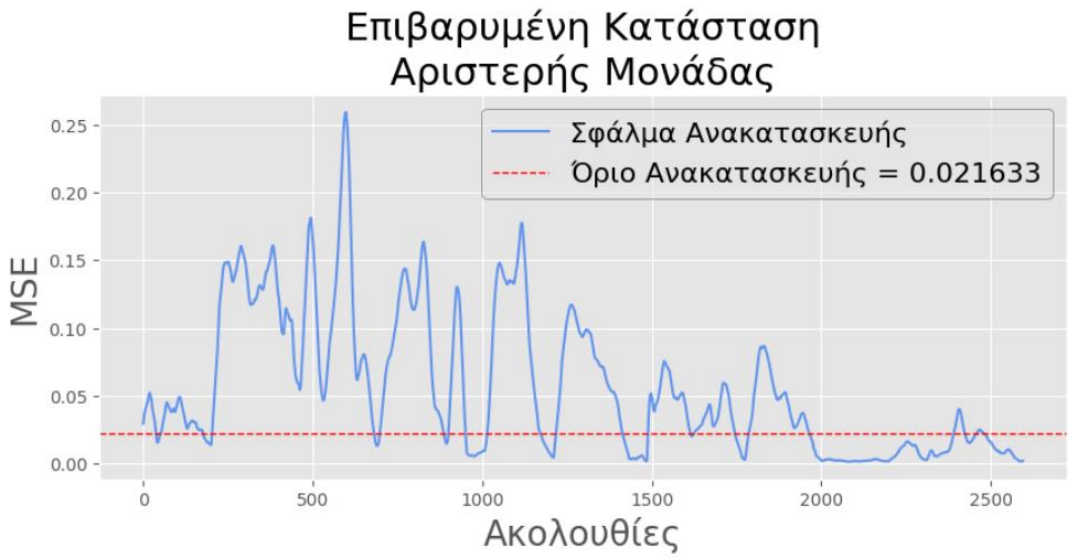
Επομένως, παρατηρείται μια γενικότερη συμφωνία μεταξύ των ακολουθιών και των ανακατασκευών τους, τόσο για την ταχύτητα, όσο και για τη θερμοκρασία. Ειδικότερα, για τις χρονοσειρές V-RMS και TEMPERATURE των μέσων όρων τον ακολουθιών, τα σφάλματα ανακατασκευής μέσω του υπολογισμού της συνάρτησης MSE είναι 0.000014088119 και 0.000016369568 αντίστοιχα.

Φυσικά, τα παραπάνω σφάλματα δεν είναι συγκρίσιμα με το όριο του σφάλματος ανακατασκευής 0.021633, λόγω του διαφορετικού τρόπου υπολογισμού τους, αλλά επιτρέπουν μια γενικότερη εικόνα των μέσων επιδόσεων του μοντέλου AE-LSTM.

3.7.4 Αξιολόγηση Επιβαρυμένης Κατάσταση Αριστερής Μονάδας

Παρακάτω ακολουθεί η παρόμοια αξιολόγηση του δεύτερου από τα τέσσερα σύνολα δεδομένων, αυτό της επιβαρυμένης λειτουργικής κατάστασης της αριστερής μονάδας ρουλεμάν, για την αναγνώριση τυχόν ανώμαλων ακολουθιών.

Αρχικά, υπολογίζονται για όλες τις 2595 ακολουθίες του συνόλου δεδομένων τα αντίστοιχα σφάλματα ανακατασκευής και οπτικοποιούνται παρακάτω:



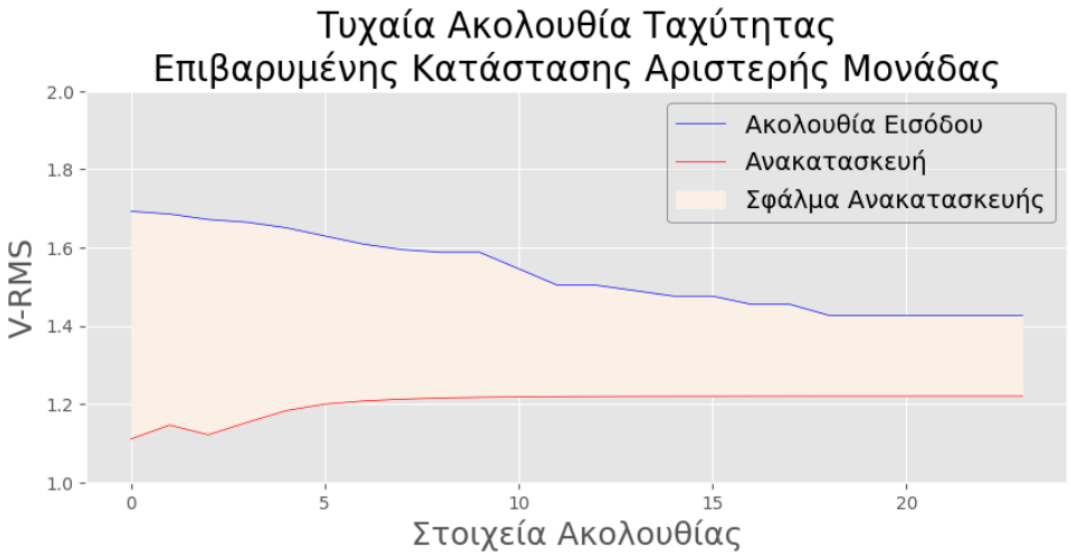
Εικόνα 72. Σφάλματα Ανακατασκευής Επιβαρυμένης Κατάστασης Αριστερής Μονάδας Ρουλεμάν

Παρατηρούμε ότι μεγάλο πλήθος των ακολουθιών αναγνωρίστηκαν ως ανωμαλίες, ως προς τη συμπεριφορά τους, σε σχέση με τις ακολουθίες που έχει μάθει να αναγνωρίζει και να ανακατασκευάζει το εκπαιδευμένο μοντέλο. Συγκεκριμένα, σε 1721 ακολουθίες, τα σφάλματα ανακατασκευής ξεπέρασαν το προκαθορισμένο όριο 0.021633, ενώ οι υπόλοιπες 874 φαίνεται πως παρουσίασαν συμπεριφορά ανάλογη των φυσιολογικών δεδομένων.

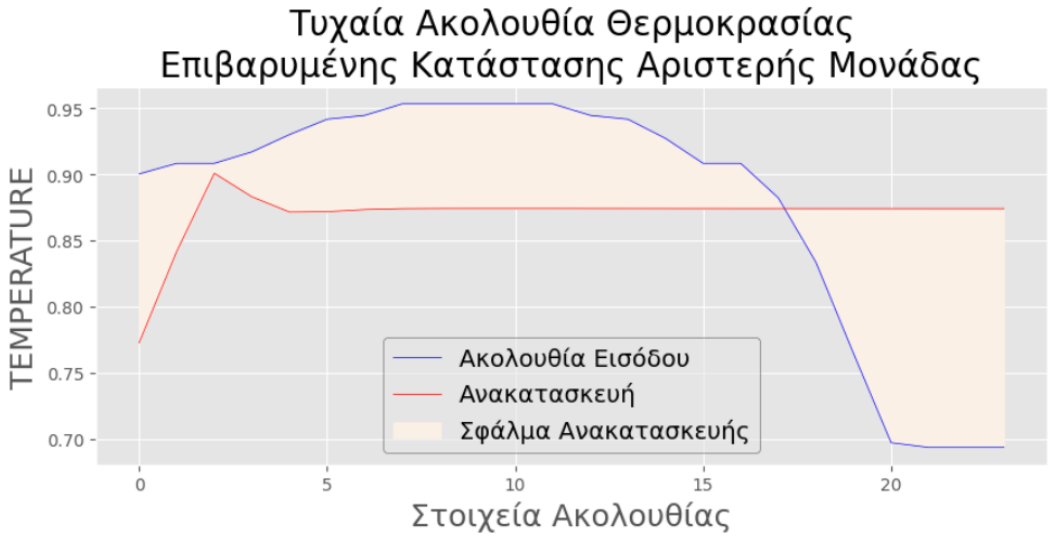
Υπενθυμίζεται ότι, παρά τις υψηλές μετρήσεις των επιβαρυμένων καταστάσεων, δεν αποτελούν απαραίτητα ένδειξη βλάβης ή απόλυτα επικίνδυνης λειτουργίας, καθώς συνέχιζαν να λειτουργούν σε χρονικό διάστημα μηνών. Επομένως, επιτρέπεται στο μοντέλο AE-LSTM ο παραπάνω βαθμός ελευθερίας, και αξίζει να σημειωθεί ότι κατά τη διάρκεια της ανάπτυξής του υπήρξαν περιπτώσεις εκπαιδευμένων μοντέλων με ακριβώς τα ίδια χαρακτηριστικά, όπου κατά την αξιολόγησή τους, οι ανώμαλες ακολουθίες έφτασαν μονάχα την τάξη των διακοσίων.

Επίσης, ενδιαφέρον αποτελεί το γεγονός ότι το παραπάνω διάγραμμα παρουσιάζει ιδιαίτερη ομοιότητα και με τη συμπεριφορά της χρονοσειράς της επιβαρυμένης κατάστασης του αριστερού ρουλεμάν της Εικόνας 64, όπου γίνεται και η σύγκριση με το ανώτατο όριο ISO.

Στη συνέχεια, επιλέγεται τυχαία μια ακολουθία του συνόλου δεδομένων, ειδικότερα η 520^{η} στη σειρά, παρουσιάζονται οι τιμές της ταχύτητας V-RMS και της θερμοκρασίας TEMPERATURE της ακολουθίας αυτής, και εξετάζονται μαζί με τις ανακατασκευές τους:



Εικόνα 73. Ανακατασκευή Ταχύτητας Τυχαίας Ακολουθίας Επιβαρυμένης Κατάστασης Αριστερής Μονάδας Ρουλεμάν

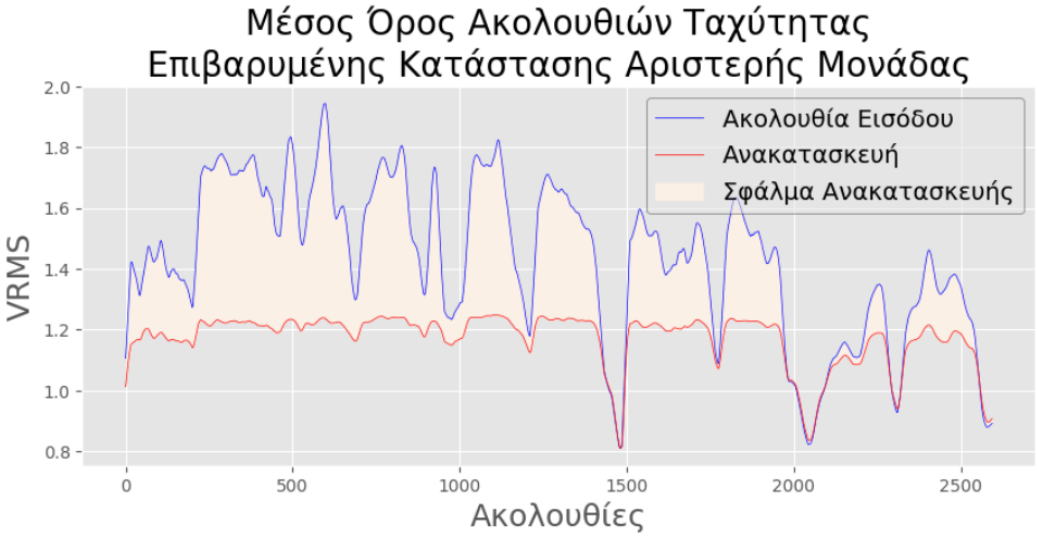


Εικόνα 74. Ανακατασκευή Θερμοκρασίας Τυχαίας Ακολουθίας Επιβαρυμένης Κατάστασης Αριστερής Μονάδας Ρουλεμάν

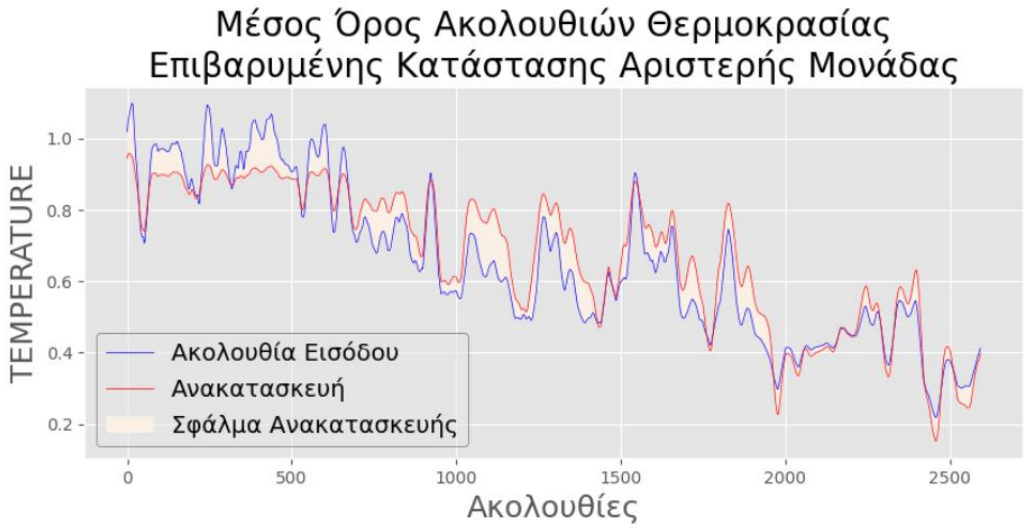
Τα επιμέρους σφάλματα για τις παραπάνω ανακατασκευές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE είναι 0.12578154 και 0.009365617 αντίστοιχα, ενώ ο μέσος όρος τους, ο οποίος και καθορίζει το αν θα αναγνωριστούν ως αποδεκτές ακολουθίες είναι 0.0675735785, το οποίο ξεπερνάει κατά πολύ το προκαθορισμένο όριο ανακατασκευής 0.021633.

Παρατηρούμε επίσης, ότι ενώ για τη θερμοκρασία το επιμέρους σφάλμα ανακατασκευής είναι πολύ μικρό, το αντίστοιχο σφάλμα για την ταχύτητα είναι κατά πολύ μεγαλύτερο, και αυτό αποτέλεσε τελικά τον κύριο παράγοντα απόρριψης της συγκεκριμένης ακολουθίας.

Για μια γενικότερη εποπτεία των ανακατασκευών, παρουσιάζονται παρακάτω οι χρονοσειρές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE, όπου κάθε στοιχείο των χρονοσειρών αντιστοιχεί στον μέσο όρο των 24 στοιχείων κάθε επιμέρους ακολουθίας ταχύτητας και θερμοκρασίας:



Εικόνα 75. Μέσος Όρος Ακολουθιών Ανακατασκευής Ταχύτητας Επιβαρυμένης Κατάστασης Αριστερής Μονάδας Ρουλεμάν



Εικόνα 76. Μέσος Όρος Ακολουθιών Ανακατασκευής Θερμοκρασίας Επιβαρυμένης Κατάστασης Αριστερής Μονάδας Ρουλεμάν

Παρατηρείται μια μεγάλη ασυμφωνία όσον αφορά στις μέσες ανακατασκευές της ταχύτητας V-RMS, ειδικότερα για ακολουθίες υψηλότερων τιμών, κάτι που ήταν αναμενόμενο. Ωστόσο, οι πιο χαμηλές τιμές ακολουθιών αναγνωρίζονται γενικότερα ως αποδεκτές ως προς τη συμπεριφορά τους, και μάλιστα συνάδουν με τα αντίστοιχα μεμονωμένα σφάλματα της Εικόνας 72, καθώς και με το διάγραμμα της χρονοσειράς της επιβαρυμένης κατάστασης του αριστερού ρουλεμάν της Εικόνας 64, όπου γίνεται και η σύγκριση με το ανώτατο όριο ISO. Το παραπάνω γενικότερο σφάλμα ανακατασκευής της συνάρτησης MSE των μέσων τιμών των ακολουθιών είναι 0.09906463.

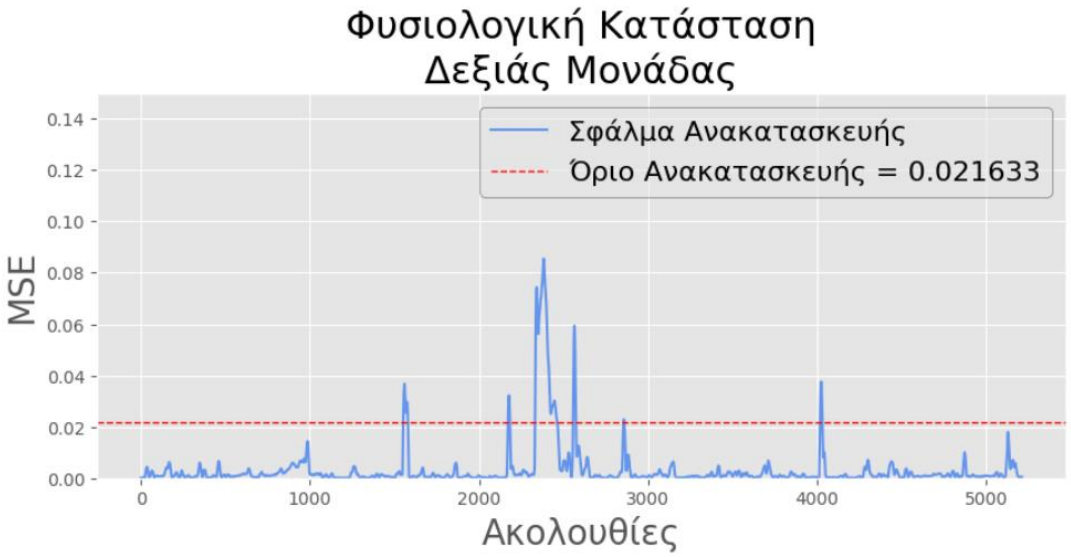
Αντίστοιχα, οι μέσες ανακατασκευές θερμοκρασίας TEMPERATURE, αν και δεν βρίσκονται σε πλήρη συμφωνία, όπως στην περίπτωση των φυσιολογικών τιμών του προηγούμενου υποκεφαλαίου, ακολουθούν ικανοποιητικά τη γενικότερη τάση των πραγματικών μέσων ακολουθιών, και το γενικότερο σφάλμα ανακατασκευής υπολογίστηκε με τη συνάρτηση MSE στο 0.005247884.

3.7.5 Αξιολόγηση Φυσιολογικής Κατάσταση Δεξιάς Μονάδας

Παρακάτω ακολουθεί η αξιολόγηση του τρίτου από τα τέσσερα σύνολα δεδομένων, αυτό της φυσιολογικής λειτουργικής κατάστασης της δεξιάς μονάδας ρουλεμάν, για την αναγνώριση τυχόν ανώμαλων ακολουθιών.

Παρά το γεγονός ότι πρόκειται για άλλο εξάρτημα από αυτό που χρησιμοποιήθηκε για την εκπαίδευση του μοντέλου AE-LSTM (το μοντέλο εκπαιδεύτηκε πάνω σε λειτουργικά δεδομένα της αριστερής μονάδας ρουλεμάν), υπάρχει ενδιαφέρον στο κατά πόσο μπορεί να ανταπεξέλθει σε εξοπλισμό παρόμοιων χαρακτηριστικών.

Αρχικά, υπολογίζονται για όλες τις 5214 ακολουθίες του συνόλου δεδομένων τα αντίστοιχα σφάλματα ανακατασκευής και οπτικοποιούνται παρακάτω:



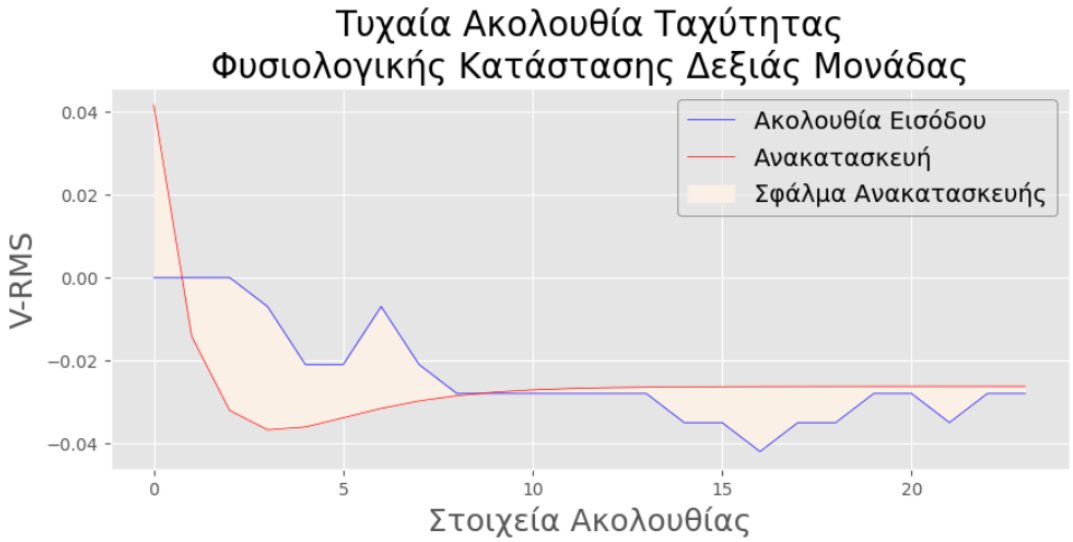
Εικόνα 77. Σφάλματα Ανακατασκευής Φυσιολογικής Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Επομένως, παρατηρούμε ότι η πλειοψηφία των 5005 ακολουθιών του συγκεκριμένου συνόλου δεδομένων αναγνωρίζονται ως φυσιολογικές καταστάσεις, ενώ μόνο 209 περιπτώσεις σφάλματος ξεπερνούν το προκαθορισμένο όριο ανακατασκευής, και συνεπώς αποτελούν ανωμαλίες, ως προς τη συμπεριφορά των ακολουθιών τους, σε σχέση με τα δεδομένα πάνω στα οποία εκπαιδεύτηκε το μοντέλο.

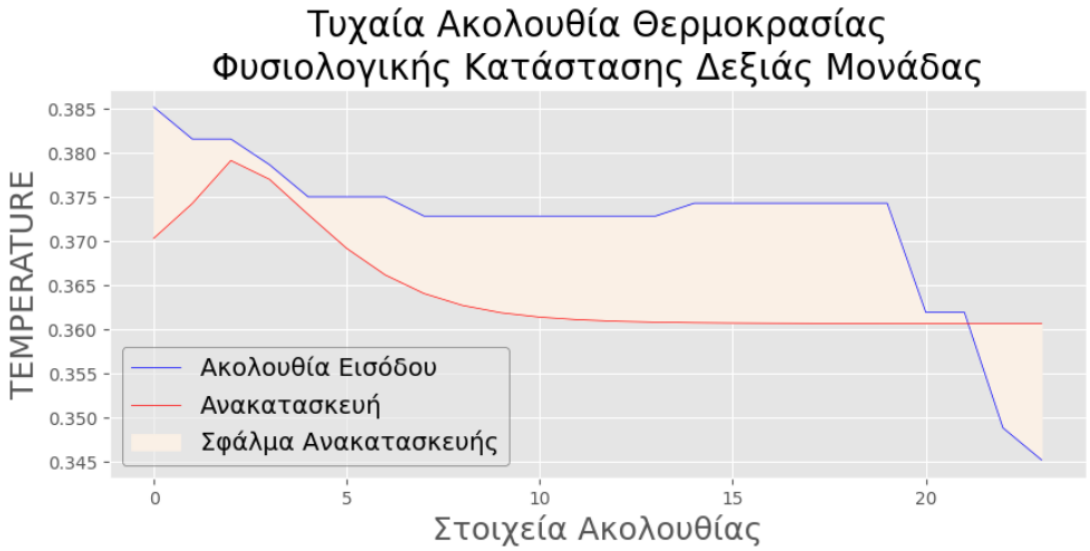
Ειδικότερα, τα σφάλματα γύρω από την 2500η περίπτωση ήταν αναμενόμενα, καθώς και οι αντίστοιχες μετρήσεις ταχύτητας V-RMS των αρχικών χρονοσειρών εμφάνισαν υψηλές τιμές στο σημείο αυτό, κατά τη μέση της χρονοσειράς.

Το γεγονός που αποτελεί ιδιαίτερο ενδιαφέρον είναι τα υψηλά σφάλματα γύρω από τη 1500η και 4000η ακολουθία. Δεδομένου ότι το σύνολο δεδομένων που μελετάται αφορά στη φυσιολογική κατάσταση της δεξιάς μονάδας ρουλεμάν, και ότι οι αντίστοιχες μετρήσεις στην αρχική χρονοσειρά ταχύτητας V-RMS της Εικόνας 62 ακολουθούν ήπια συμπεριφορά, υπάρχει πιθανότητα τα συγκεκριμένα υψηλά σφάλματα ανακατασκευής να αποτελούν περιπτώσεις Ψευδών Θετικών (False Positive) αποτελεσμάτων. Συνεπώς, είτε όντως πρόκειται για μη-φυσιολογικές συμπεριφορές που αναγνώρισε το μοντέλο, είτε πρόκειται για περιπτώσεις αναγνώρισης ανωμαλιών, ενώ στην πραγματικότητα δεν υπάρχουν. Σε κάθε περίπτωση, στα πλαίσια μιας πραγματικής βιομηχανικής διαδικασίας, θα χρειαζόταν έλεγχος συντήρησης μέσω του ανθρώπινου παράγοντα για την επιβεβαίωση ύπαρξης βλάβης.

Στη συνέχεια, επιλέγεται τυχαία μια ακολουθία του συνόλου δεδομένων, ειδικότερα η 410η στη σειρά, παρουσιάζονται οι τιμές της ταχύτητας V-RMS και της θερμοκρασίας TEMPERATURE της ακολουθίας αυτής, και εξετάζονται μαζί με τις ανακατασκευές τους:



Εικόνα 78. Ανακατασκευή Ταχύτητας Τυχαίας Ακολουθίας Φυσιολογικής Κατάστασης Δεξιάς Μονάδας Ρουλεμάν



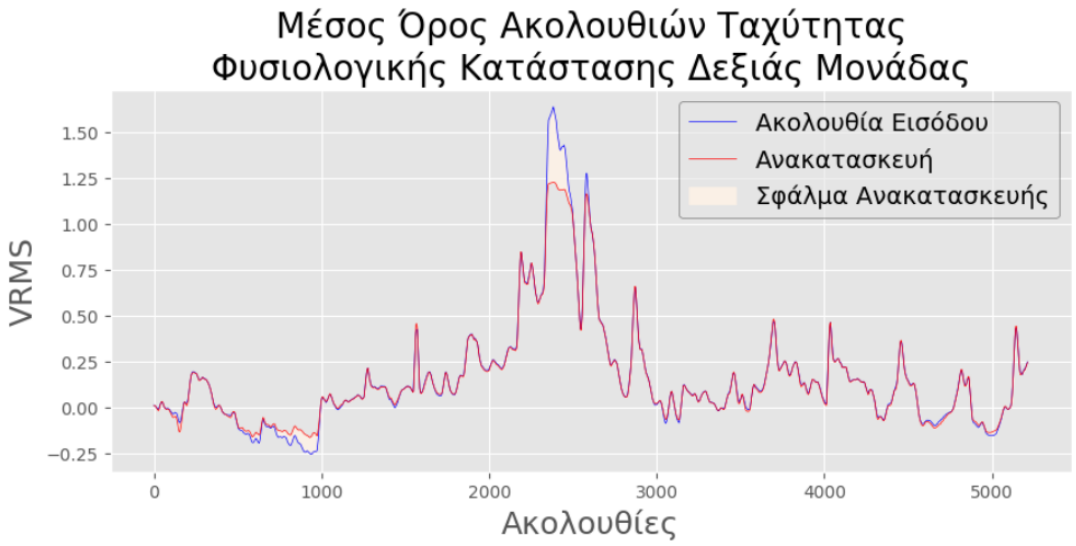
Εικόνα 79. Ανακατασκευή Θερμοκρασίας Τυχαίας Ακολουθίας Φυσιολογικής Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Τα επιμέρους σφάλματα για τις παραπάνω ανακατασκευές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE είναι 0.00023128318 και 0.00011423493 αντίστοιχα, ενώ ο μέσος όρος τους, ο οποίος και καθορίζει το αν θα αναγνωριστούν ως αποδεκτές ακολουθίες

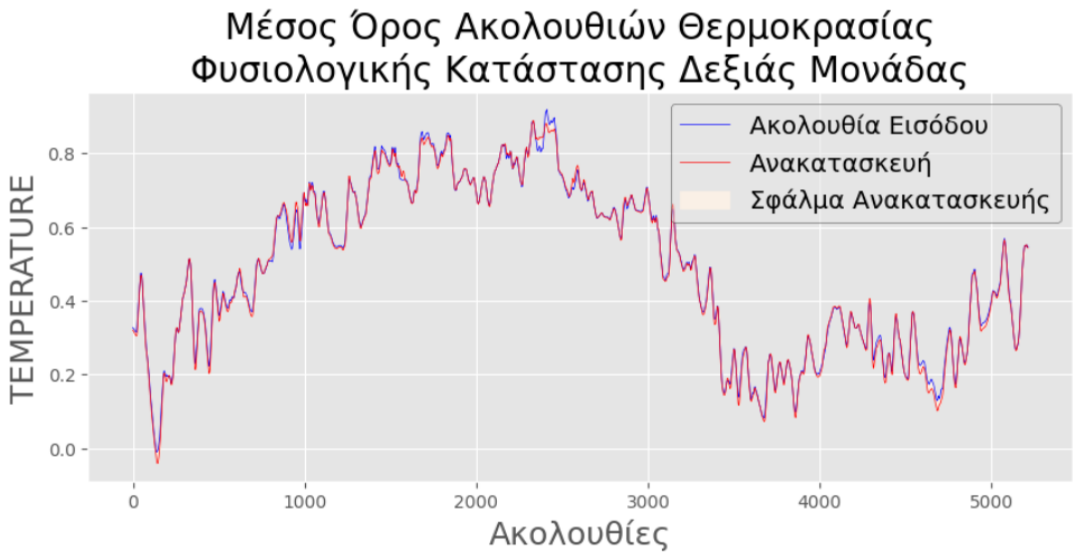
είναι 0.000172759055, το οποίο είναι προφανώς μικρότερο από το προκαθορισμένο όριο ανακατασκευής 0.021633.

Παρατηρούμε για τις τιμές των παραπάνω σφαλμάτων ανακατασκευής, τα οποία υπολογίζονται με τη βοήθεια του κλασσικού τύπου της συνάρτησης MSE, μια συμπεριφορά ανάλογη με αυτή που εξετάστηκε για την τυχαία ακολουθία της φυσιολογικής κατάστασης του αριστερού ρουλεμάν. Επίσης, ακόμη και οι ίδιες οι ανακατασκευές ακολουθούν όσο το δυνατόν καλύτερα τις επιμέρους ακολουθίες εισόδου, και κατ' επέκταση το αντίστοιχο σφάλμα ανακατασκευής ανήκει στα πλαίσια των αποδεκτών σφαλμάτων.

Για μια γενικότερη εποπτεία των ανακατασκευών, παρουσιάζονται παρακάτω οι χρονοσειρές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE, όπου κάθε στοιχείο των χρονοσειρών αντιστοιχεί στον μέσο όρο των 24 στοιχείων κάθε επιμέρους ακολουθίας ταχύτητας και θερμοκρασίας:



Εικόνα 80. Μέσος Όρος Ακολουθιών Ανακατασκευής Ταχύτητας Φυσιολογικής Κατάστασης Δεξιάς Μονάδας Ρουλεμάν



Εικόνα 81. Μέσος Όρος Ακολουθιών Ανακατασκευής Θερμοκρασίας Φυσιολογικής Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Παρατηρείται και πάλι συμφωνία ως προς τις μέσες τιμές των ακολουθιών εισόδου και των αντίστοιχων ανακατασκευών τους, ανάλογη με αυτή στην περίπτωση της φυσιολογικής κατάστασης της αριστερής μονάδας ρουλεμάν. Επίσης, για το σημείο γύρω από την 2500η περίπτωση στη χρονοσειρά ταχύτητας V-RMS, επιβεβαιώνεται η εμφάνιση των μεγαλύτερων μεμονωμένων σφαλμάτων ανακατασκευής της Εικόνας 77.

Προς το παρόν, σύμφωνα με τα παραπάνω διαγράμματα, η γενίκευση του μοντέλου AE-LSTM σε εξαρτήματα ανάλογων χαρακτηριστικών με αυτά πάνω στα οποία εκπαιδεύτηκε είναι ικανοποιητική, καθώς παρατηρούνται παρόμοιες συμπεριφορές τόσο στην αξιολόγηση των φυσιολογικών δεδομένων του αριστερού ρουλεμάν, όσο και στην αξιολόγηση των φυσιολογικών δεδομένων του δεξιού ρουλεμάν.

Επιπλέον, για τις χρονοσειρές V-RMS και TEMPERATURE των μέσων όρων των ακολουθιών τα σφάλματα ανακατασκευής μέσω του υπολογισμού της συνάρτησης MSE είναι 0.002493882 και 0.00010563642 αντίστοιχα.

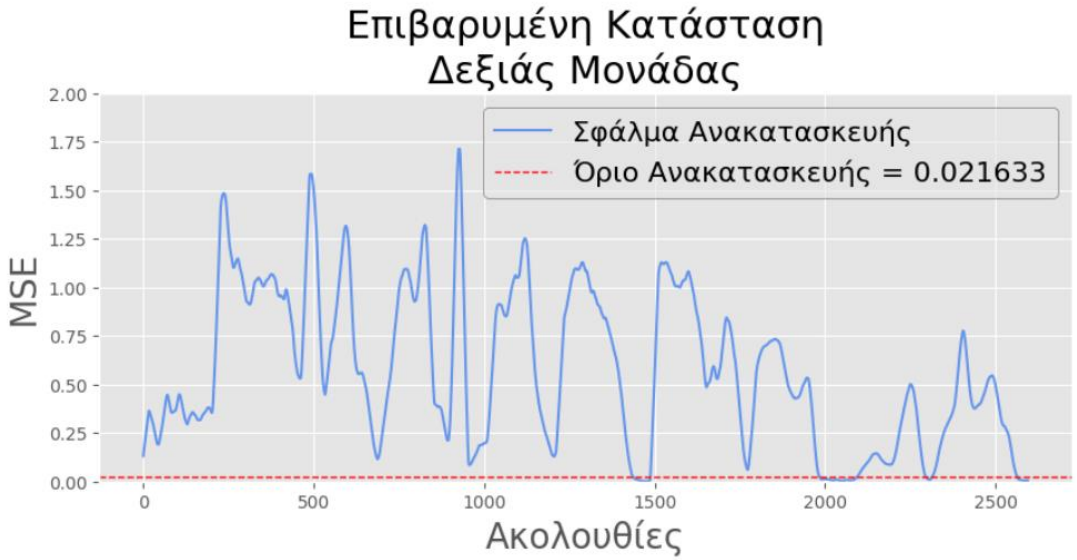
3.7.6 Αξιολόγηση Επιβαρυμένης Κατάσταση Δεξιάς Μονάδας

Τελικά, παρακάτω ακολουθεί η αξιολόγηση και του τελευταίου από τα τέσσερα σύνολα δεδομένων, αυτό της επιβαρυμένης λειτουργικής κατάστασης της δεξιάς μονάδας ρουλεμάν, για την αναγνώριση τυχόν ανώμαλων ακολουθιών.

Ειδικότερα, αναμένουμε υψηλότερα σφάλματα, λόγω και των υψηλότερων τιμών ταχύτητας V-RMS στις αρχικές ακολουθίες εισόδου, σε σχέση με αυτές των πιο ήπιων

συμπεριφορών, όπου εκπαιδεύτηκε το μοντέλο AE-LSTM. Επίσης, υπενθυμίζεται ότι, από τη συμβατική αξιολόγηση των συνόλων δεδομένων, το συγκεκριμένο σύνολο σημείωσε το υψηλότερο σφάλμα ανακατασκευής.

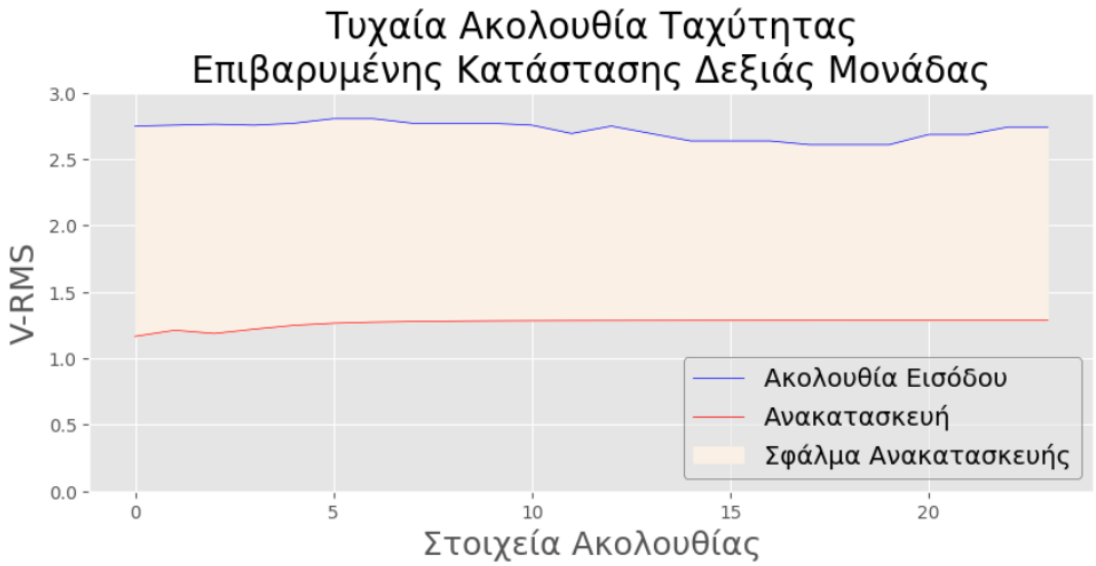
Αρχικά, υπολογίζονται για όλες τις 2595 ακολουθίες του συνόλου δεδομένων τα αντίστοιχα σφάλματα ανακατασκευής και οπτικοποιούνται παρακάτω:



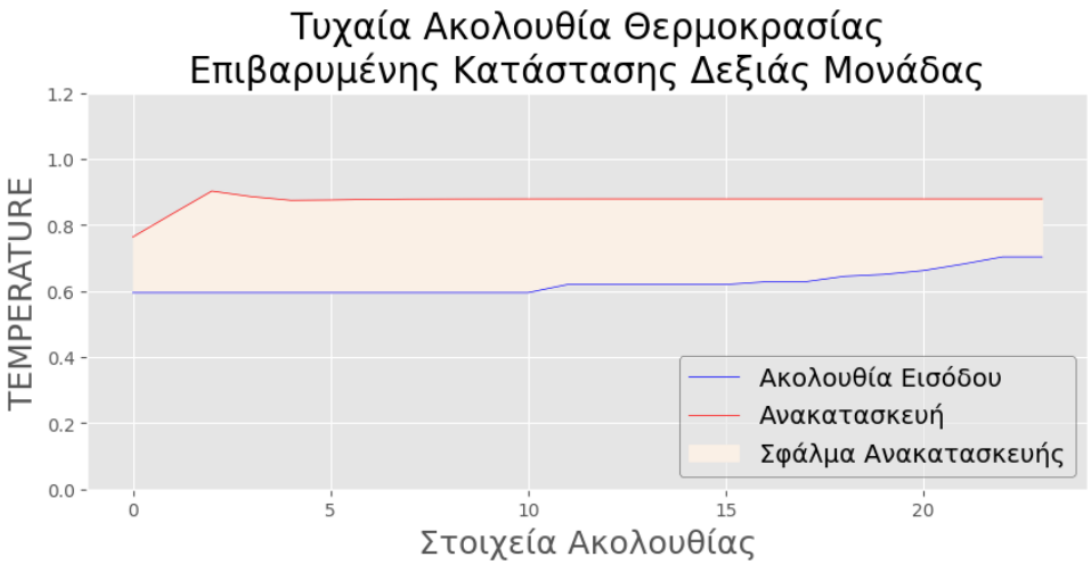
Εικόνα 82. Σφάλματα Ανακατασκευής Επιβαρυμένης Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Επομένως, όπως περιμέναμε, είναι εμφανές ότι η πλειοψηφία των ακολουθιών, συγκεκριμένα 2391 στο σύνολό τους, αναγνωρίζονται βάσει του σφάλματος ανακατασκευής ως ανωμαλίες, ενώ μόνο οι 204 θεωρούνται ανάλογες της αποδεκτής κατάστασης λειτουργίας.

Στη συνέχεια, επιλέγεται τυχαία μια ακολουθία του συνόλου δεδομένων, ειδικότερα η 1300^{η} στη σειρά, παρουσιάζονται οι τιμές της ταχύτητας V-RMS και της θερμοκρασίας TEMPERATURE της ακολουθίας αυτής, και εξετάζονται μαζί με τις ανακατασκευές τους:



Εικόνα 83. Ανακατασκευή Ταχύτητας Τυχαίας Ακολουθίας Επιβαρυμένης Κατάστασης Δεξιάς Μονάδας Ρουλεμάν



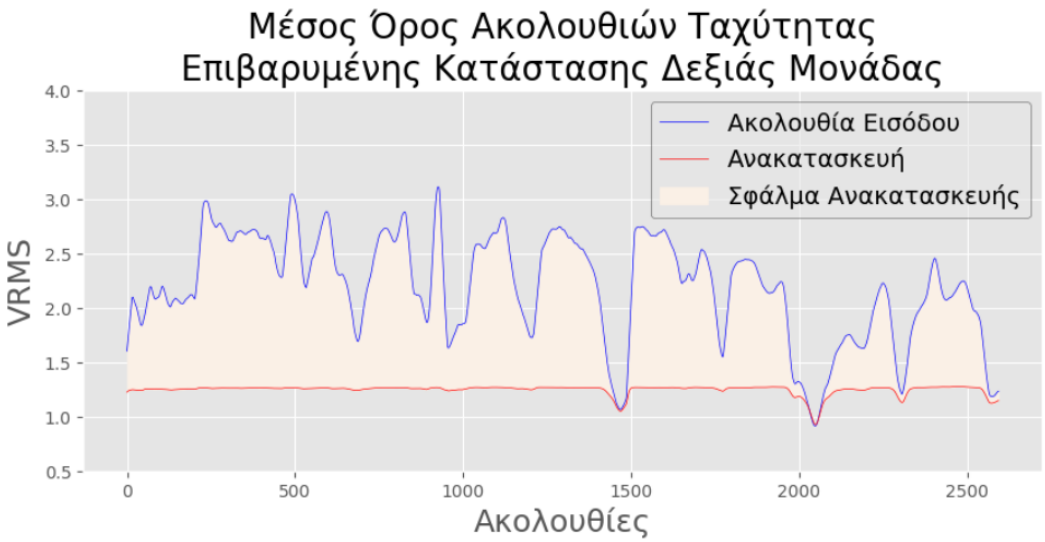
Εικόνα 84. Ανακατασκευή Θερμοκρασίας Τυχαίας Ακολουθίας Επιβαρυμένης Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Τα επιμέρους σφάλματα για τις παραπάνω ανακατασκευές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE είναι 2.1071713 και 0.06414094 αντίστοιχα, ενώ ο μέσος όρος τους, ο οποίος και καθορίζει το αν θα αναγνωριστούν ως αποδεκτές ακολουθίες είναι 1.0856561200000001, το οποίο ξεπερνάει κατά πολύ το προκαθορισμένο όριο ανακατασκευής 0.021633.

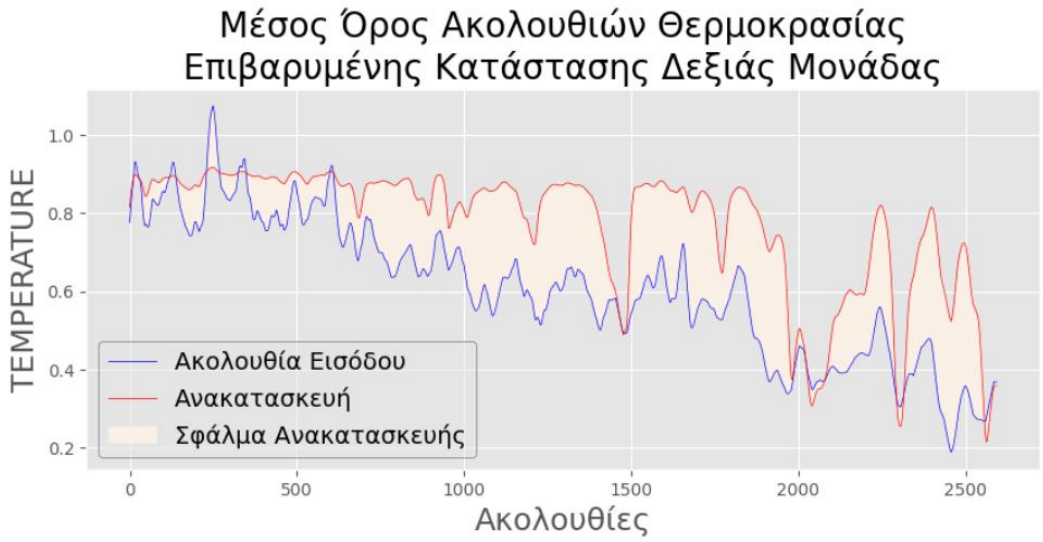
Επίσης, παρατηρούμε και πάλι τη μεγάλη διαφορά σφάλματος μεταξύ ταχύτητας και θερμοκρασίας, με την ταχύτητα να επηρεάζει κατά πολύ τον μέσο όρο του σφάλματος

ανακατασκευής, με τρόπο ανάλογο της επιβαρυμένης κατάστασης της αριστερής μονάδας, ωστόσο σε μεγαλύτερο βαθμό.

Για μια γενικότερη εποπτεία των ανακατασκευών, παρουσιάζονται παρακάτω οι χρονοσειρές ταχύτητας V-RMS και θερμοκρασίας TEMPERATURE, όπου κάθε στοιχείο των χρονοσειρών αντιστοιχεί στον μέσο όρο των 24 στοιχείων κάθε επιμέρους ακολουθίας ταχύτητας και θερμοκρασίας:



Εικόνα 85. Μέσος Όρος Ακολουθιών Ανακατασκευής Ταχύτητας Επιβαρυμένης Κατάστασης Δεξιάς Μονάδας Ρουλεμάν



Εικόνα 86. Μέσος Όρος Ακολουθιών Ανακατασκευής Θερμοκρασίας Επιβαρυμένης Κατάστασης Δεξιάς Μονάδας Ρουλεμάν

Παρατηρείται και πάλι η αναμενόμενη μεγάλη ασυμφωνία ως προς τις μέσες τιμές των ακολουθιών εισόδου και των αντίστοιχων ανακατασκευών τους, ανάλογη με αυτή στην περίπτωση της επιβαρυμένης κατάστασης της αριστερής μονάδας ρουλεμάν, ωστόσο σε αρκετά μεγαλύτερο βαθμό. Επίσης, μπορούμε να αναγνωρίσουμε ότι, οι περιπτώσεις των αποδεκτών σφαλμάτων ανακατασκευής στην Εικόνα 82, γύρω από τα σημεία 1500 και 2000, επιβεβαιώνονται από τους κοντινούς μέσους όρους των ακολουθιών εισόδου και των ανακατασκευών τους και στα δυο παραπάνω διαγράμματα.

Επιπλέον, για τις χρονοσειρές V-RMS και TEMPERATURE των μέσων όρων των ακολουθιών τα σφάλματα ανακατασκευής μέσω του υπολογισμού της συνάρτησης MSE είναι 1.0894212 και 0.040070117 αντίστοιχα, με την ταχύτητα και πάλι να παρουσιάζει το κατά πολύ μεγαλύτερο γενικό σφάλμα.

Παρακάτω, δίνεται μια συγκριτική επισκόπηση όλων των σφαλμάτων ανακατασκευής, όπως παρουσιάστηκαν παραπάνω για όλες τις περιπτώσεις, και όπως υπολογίστηκαν με τον μαθηματικό τύπο της συνάρτησης MSE:

Πίνακας 4. Σφάλματα Ανακατασκευών Αξιολόγησης

			V-RMS	TEMPERATURE
Αριστερή Μονάδα Ρουλεμάν	Φυσιολογική Κατάσταση	Τυχαία Ακολουθία	0.00019302306	0.0003537076
		Μέση Ακολουθία	0.000014088119	0.000016369568
	Επιβαρυμένη Κατάσταση	Τυχαία Ακολουθία	0.12578154	0.009365617
		Μέση Ακολουθία	0.09906463	0.005247884
Δεξιά Μονάδα Ρουλεμάν	Φυσιολογική Κατάσταση	Τυχαία Ακολουθία	0.00023128318	0.00011423493
		Μέση Ακολουθία	0.002493882	0.00010563642
	Επιβαρυμένη Κατάσταση	Τυχαία Ακολουθία	2.1071713	0.06414094
		Μέση Ακολουθία	1.0894212	0.040070117

Συνεπώς, παρατηρούμε ότι, όσον αφορά στη φυσιολογική κατάσταση και των δυο μονάδων ρουλεμάν, τα σφάλματα ανακατασκευών ακολουθούν ανάλογη συμπεριφορά, ωστόσο για την περίπτωση του αριστερού ρουλεμάν, οι ανακατασκευές των ακολουθιών επιτεύχθηκαν με λίγο μικρότερα σφάλματα. Το γεγονός αυτό οφείλεται πιθανότατα στο ότι το μοντέλο AE-LSTM είναι εκπαιδευμένο πάνω σε δεδομένα φυσιολογικής κατάστασης που συλλέχθηκαν από την αριστερή μονάδα, συνεπώς είναι αναμενόμενο να παρουσιάζουν παρόμοιες συμπεριφορές και κατ' επέκταση μικρότερα σφάλματα ανακατασκευής.

Όσον αφορά στις περιπτώσεις των επιβαρυμένων καταστάσεων, σαφώς παρατηρούνται μεγαλύτερες τιμές σφαλμάτων ανακατασκευής, ωστόσο για την αριστερή μονάδα ρουλεμάν τα σφάλματα αυτά είναι σημαντικά μικρότερα από αυτά της δεξιάς μονάδας, κάτι που ήταν αναμενόμενο, λόγω των υψηλότερων μετρήσεων που καταγράφηκαν για τη δεξιά μονάδα στην χρονοσειρά της ταχύτητας V-RMS.

Επίσης, για όλα τα σφάλματα της ταχύτητας V-RMS σε όλες τις παραπάνω περιπτώσεις του πίνακα, τόσο στις τυχαίες, όσο και στις μέσες ακολουθίες, παρατηρούμε ότι είναι εμφανώς μεγαλύτερα από τα αντίστοιχα σφάλματα της θερμοκρασίας TEMPERATURE, με εξαίρεση την περίπτωση της φυσιολογικής κατάστασης της αριστερής μονάδας ρουλεμάν όπου η διαφορά αυτή παρατηρείται σε μικρότερο βαθμό. Κάτι τέτοιο είναι ενδεικτικό της σημαντικότητας του χαρακτηριστικού της ταχύτητας σε σχέση με αυτό της θερμοκρασίας, καθώς αυτό φαίνεται να επηρεάζει κατά κύριο λόγο το συνολικό σφάλμα ανακατασκευής, και τελικά την αναγνώριση της εκάστοτε ακολουθίας ως αποδεκτής ή ως ανωμαλίας.

Προς το παρόν, σύμφωνα με τα παραπάνω διαγράμματα και αποτελέσματα, η γενίκευση του μοντέλου AE-LSTM σε εξαρτήματα ανάλογων χαρακτηριστικών με αυτά πάνω στα οποία εκπαιδεύτηκε είναι ικανοποιητική, καθώς παρατηρούνται παρόμοιες συμπεριφορές τόσο στην αξιολόγηση των φυσιολογικών δεδομένων του αριστερού ρουλεμάν, όσο και στην αξιολόγηση των φυσιολογικών δεδομένων του δεξιού ρουλεμάν.

4 Εφαρμογή σε πρόβλημα Ομαδοποίησης

4.1 Περιγραφή Προβλήματος

Η τρίτη εφαρμογή αφορά και πάλι το σύνολο δεδομένων που συλλέχθηκαν από το ρουλεμάν της πειραματικής διάταξης της πρώτης εφαρμογής, κατά τη λειτουργία του υπό τις τρεις διαφορετικές συνθήκες λίπανσης (κανονική ποσότητα, απώλεια λιπαντικού, περίσσεια λιπαντικού).

Συγκεκριμένα, παρουσιάζονται τεχνικές αλγορίθμων Μηχανικής Μάθησης της βιβλιοθήκης Scikit-learn, οι οποίες χρησιμοποιούνται για την ομαδοποίηση παρόμοιων στοιχείων ενός πίνακα δεδομένων. Συνεπώς, θα εξεταστούν αλγόριθμοι τύπου Μη-Επιβλεπόμενης Μάθησης (Unsupervised Learning), ωστόσο σε πιο ελεγχόμενο επίπεδο, καθώς υπάρχουν διαθέσιμες στο σύνολο δεδομένων οι ετικέτες για την πραγματική λειτουργική κατάσταση του ρουλεμάν.

Επισημαίνεται ότι, στόχος του κεφαλαίου είναι η υπολογιστική εφαρμογή των τεχνικών ομαδοποίησης, χωρίς τη μαθηματική εμβάθυνση των προηγούμενων κεφαλαίων, και κυρίως η οπτικοποίηση των δεδομένων και των αποτελεσμάτων των αλγορίθμων μάθησης, σε εισαγωγικό βαθμό και κατευθυντήριων πλαισίων.

4.2 Προεπεξεργασία Δεδομένων

Αρχικά, εισάγονται τα απαραίτητα πακέτα και συναρτήσεις:

```
Input []:
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4
5  from sklearn.preprocessing import StandardScaler
6  from sklearn.cluster import KMeans
7  from sklearn.mixture import GaussianMixture, BayesianGaussianMixture
8  from sklearn.decomposition import PCA
9  from sklearn.metrics import ConfusionMatrixDisplay
```

Στη συνέχεια, εισάγεται το σύνολο των δεδομένων με τις τρεις διαφορετικές καταστάσεις του ρουλεμάν, γίνεται τροποποίηση της αύξουσας αρίθμησης των τριών καταστάσεων με αρχή το μηδέν, γίνεται διαχωρισμός των δεδομένων στα πέντε χαρακτηριστικά X και τη

στήλη των ετικετών y, τα οποία έπειτα μετασχηματίζονται σε στοιχεία πινάκων και τελικά το σύνολο των χαρακτηριστικών μετασχηματίζεται περαιτέρω για να γίνει αλλαγή κλίμακας των τιμών με τη μέθοδο της Κανονικοποίησης, έτσι ώστε κάθε στήλη να έχει μέση τιμή μηδέν και τυπική απόκλιση τη μονάδα:

```
Input []:
1  data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
2                      thesis/data/bearing_vibration_metrics.csv')
3  data['Bearing State'] = data['Bearing State'] - 1
4
5  X = data.iloc[:,:-1]
6  y = data['Bearing State']
7
8  X = np.asarray(X)
9  y = np.asarray(y)
10
11 scaler = StandardScaler()
12 scaler.fit(X)
13 X_scaled = scaler.transform(X)
```

Επομένως το τελικό μετασχηματισμένο σύνολο δεδομένων που θα τροφοδοτηθεί στα μοντέλα για εκπαίδευση θα είναι το X_scaled.

Να σημειωθεί ότι τα μοντέλα που εξετάζονται είναι απλοί αλγόριθμοι Μηχανικής Μάθησης και βασίζονται σε τεχνικές πιθανοτήτων ή σε κλασσικούς αλγόριθμους επανάληψης, σε αντίθεση με την επαναληπτική διαδικασία μάθησης των Τεχνητών Νευρωνικών Δικτύων της μεθόδου Οπισθοδιάδοσης.

Επίσης, τα δεδομένα δεν ανακατεύτηκαν (shuffle) ως προς τις εγγραφές τους, και δεν έγινε διαχωρισμός σε σύνολα Εκπαίδευσης, Επιβεβαίωσης ή Αξιολόγησης (train_test_split), καθώς θέλουμε να ελέγξουμε τις δυνατότητες διαχωρισμού των μοντέλων για ολόκληρο το σύνολο δεδομένων.

4.3 Τεχνικές Ομαδοποίησης

Αρχικά, οι τεχνικές που εξετάστηκαν ήταν ο αλγόριθμος K-Means, το μοντέλο Gaussian Mixture και το εναλλακτικό μοντέλο Bayesian Gaussian Mixture.

Ο αλγόριθμος K-Means ομαδοποιεί τα δεδομένα γύρω από ένα κεντρικό σημείο για κάθε δυνατή κλάση, το οποίο ανανεώνεται με επαναληπτικό τρόπο σύμφωνα με την απόστασή του από όλα τα υπόλοιπα σημεία του συνόλου δεδομένων. Από την άλλη, οι τεχνικές

Gaussian Mixture είναι μοντέλα πιθανοτήτων οι οποίες ομαδοποιούν τα δεδομένα βάσει της κατανομής τους, και κυρίως σε δεδομένα τα οποία προέρχονται από ελλειψοειδής κατανομές.

4.3.1 K-Means

Γίνεται αρχικοποίηση ενός μοντέλου K-Means από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα δεδομένα X_scaled και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή:

```
Input []:
1  kmeans = KMeans(n_clusters=3, n_init=10)
2  kmeans.fit(X_scaled)
3  y_pred = kmeans.predict(X_scaled)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα κέντρα που αναγνώρισε, όλες τις ετικέτες, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των επαναλήψεων, το πλήθος των χαρακτηριστικών κ.α.:

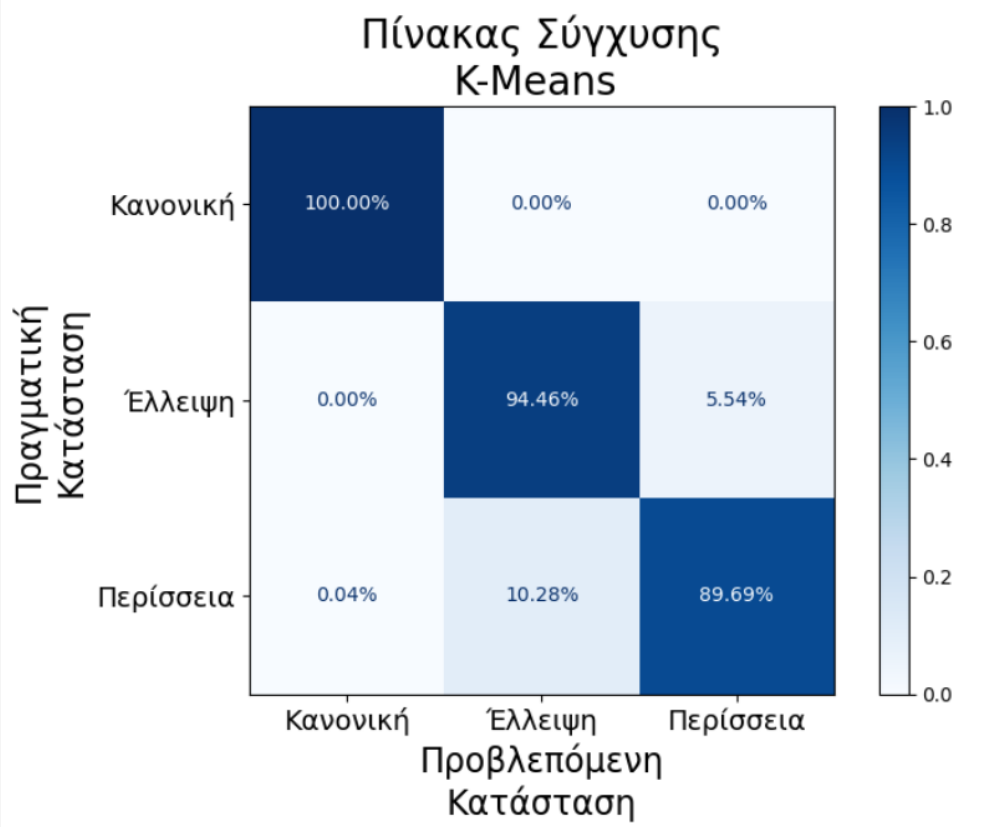
```
Input []:
1  print('Parameters: ', kmeans.get_params(deep=True))
2  print('Cluster Centers: \n', kmeans.cluster_centers_)
3  print('Labels: ', kmeans.labels_)
4  print('Predicted Clusters: ', np.unique(y_pred))
5  print('Iterations: ', kmeans.n_iter_)
6  print('Features: ', kmeans.n_features_in_)

Output []:
Parameters:  {'algorithm': 'lloyd', 'copy_x': True, 'init': 'k-means++',
              'max_iter': 300, 'n_clusters': 3, 'n_init': 10,
              'random_state': None, 'tol': 0.0001, 'verbose': 0}
Cluster Centers:
 [[ 1.2353473  -0.4052996  -0.94458454 -1.09508297 -0.4806018 ]
 [-0.75240378 -0.72162291 -0.0937895   0.25549187  0.15189687]
 [-0.79470766  1.32482167  1.34281618  1.15658082  0.46389681]]
Labels:  [0 0 0 ... 2 2 2]
Predicted Clusters:  [0 1 2]
Iterations:  5
Features:  5
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

Input []:

```
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5             y, y_pred, display_labels=labels, normalize='true',
6             values_format='.2%', include_values=True, cmap='Blues',
7             xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\nK-Means ', fontsize='20')
9  plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
10 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
11 plt.tick_params(labelsize='14')
12 plt.tight_layout()
13 plt.show()
```

Output []:



Παρατηρούμε ότι υπάρχει αρκετά μεγάλη επιτυχία στην αναγνώριση των καταστάσεων, ωστόσο σε μικρότερο βαθμό από τα μοντέλα MLP Βαθιάς Μάθησης της πρώτης εφαρμογής. Επίσης, σημειώθηκε και ένα ποσοστό σφάλματος Τύπου I της τάξης 0.04%, κάτι το οποίο είναι προτιμότερο να αποφεύγεται.

Επιπλέον, ο αλγόριθμος K-Means επηρεάζεται ιδιαίτερα από τις τιμές των δεδομένων, και σε περίπτωση που δεν είχε προηγηθεί Κανονικοποίηση μέσης τιμής μηδέν και τυπικής

απόκλισης μονάδας, τα ποσοστά επιτυχίας θα σημαντικά μειωμένα και οι πιθανότητες σφάλματος αρκετά αυξημένες.

4.3.2 Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα δεδομένα X_scaled και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

```
Input []:
1  gm = GaussianMixture(n_components=3, n_init=10)
2  gm.fit(X_scaled)
3  y_pred = gm.predict(X_scaled)
4  y_pred_prob = gm.predict_proba(X_scaled)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', gm.get_params(deep=True))
2  print('Weights: ', gm.weights_)
3  print('Converged: ', gm.converged_)
4  print('Iterations to Converge: ', gm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', gm.n_features_in_)

Output []:
Parameters:  {'covariance_type': 'full', 'init_params': 'kmeans',
              'max_iter': 100, 'means_init': None, 'n_components': 3,
              'n_init': 10, 'precisions_init': None,
              'random_state': None, 'reg_covar': 1e-06, 'tol': 0.001,
              'verbose': 0, 'verbose_interval': 10,
              'warm_start': False, 'weights_init': None}
Weights:  [0.38464232 0.33674879 0.27860889]
Converged:  True
Iterations to Converge:  4
Predicted Propabilities:
 [[1.00000000e+00 4.39285551e-41 9.20360185e-46]
```

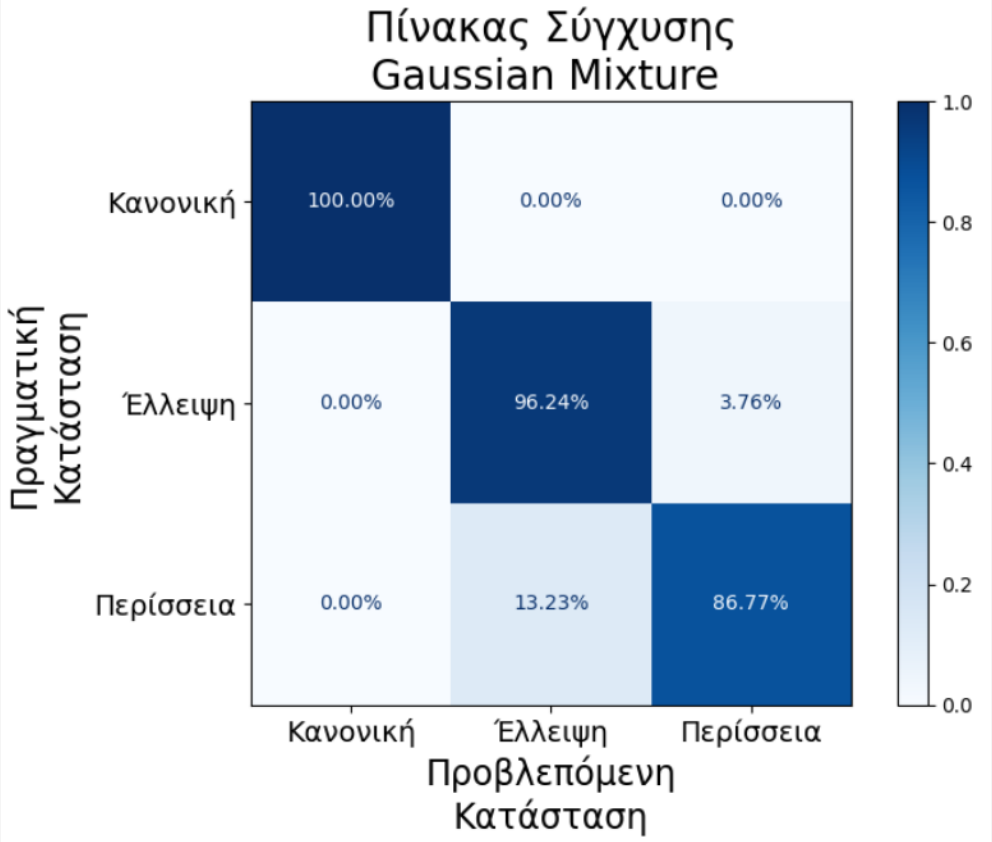
```
 [1.00000000e+00 9.14324800e-49 2.56521835e-43]
 [1.00000000e+00 8.28619336e-43 4.47050420e-44]]
Predicted Clusters:  [0 1 2]
Features:  5
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

Input []:

```
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5             y, y_pred, display_labels=labels, normalize='true',
6             values_format='.2%', include_values=True, cmap='Blues',
7             xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\nGaussian Mixture ',
9                   fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13 plt.tight_layout()
14 plt.show()
```

Output []:



Συγκρίνοντας τον Πίνακα Σύγχυσης με αυτόν του μοντέλου K-Means, βλέπουμε ότι στη συγκεκριμένη περίπτωση, αν και δεν υπάρχουν σφάλματα Τύπου I, υπάρχει μεγαλύτερη δυσκολία στην αναγνώριση της κατάστασης με την περίσσεια ποσότητα λιπαντικού.

4.3.3 Bayesian Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Bayesian Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα δεδομένα X_scaled και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

```
Input []:
1  bgm = BayesianGaussianMixture(n_components=3, n_init=10)
2  bgm.fit(X_scaled)
3  y_pred = bgm.predict(X_scaled)
4  y_pred_prob = bgm.predict_proba(X_scaled)
```

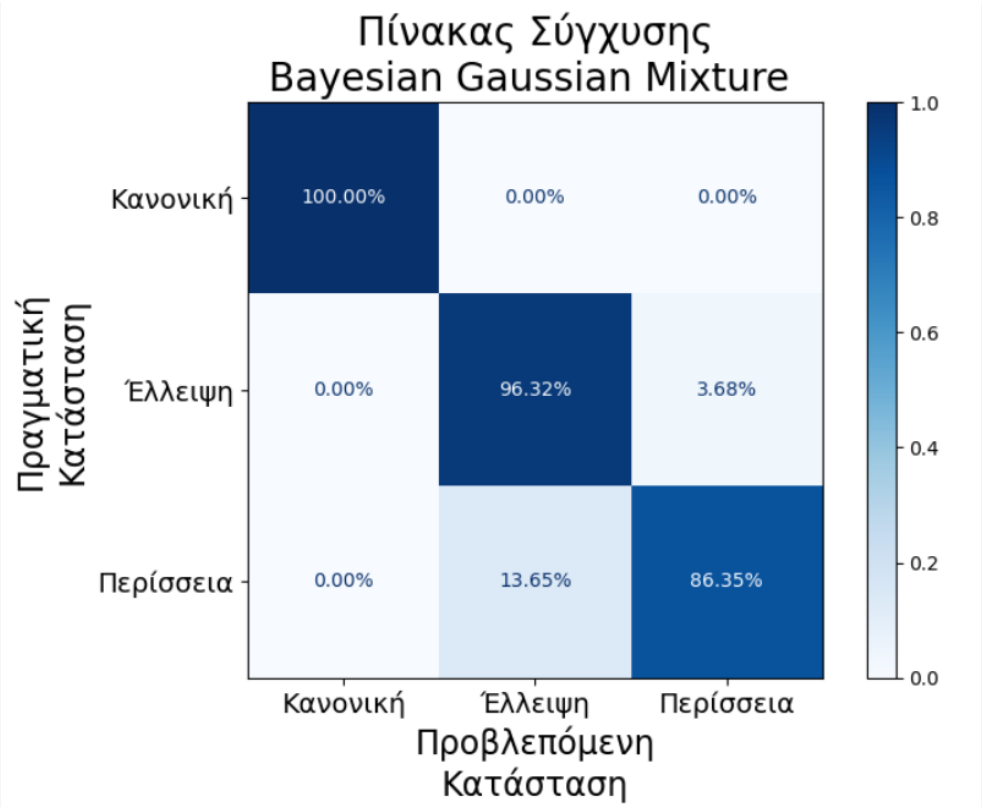
Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', bgm.get_params(deep=True))
2  print('Weights: ', bgm.weights_)
3  print('Converged: ', bgm.converged_)
4  print('Iterations to Converge: ', bgm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', bgm.n_features_in_)

Output []:
Parameters:  {'covariance_prior': None, 'covariance_type': 'full',
              'degrees_of_freedom_prior': None, 'init_params': 'kmeans',
              'max_iter': 100, 'mean_precision_prior': None,
              'mean_prior': None, 'n_components': 3, 'n_init': 10,
              'random_state': None, 'reg_covar': 1e-06, 'tol': 0.001,
              'verbose': 0, 'verbose_interval': 10, 'warm_start': False,
              'weight_concentration_prior': None,
              'weight_concentration_prior_type': 'dirichlet_process'}
Weights:  [0.38471411 0.34044384 0.27484205]
```

```
Converged:  True
Iterations to Converge:  17
Predicted Propabilities:
 [[1.00000000e+00 7.08550281e-40 2.86781363e-45]
 [1.00000000e+00 3.62774183e-47 1.15838674e-42]
 [1.00000000e+00 2.12638300e-41 1.94443900e-43]]
Predicted Clusters:  [0 1 2]
Features:  5
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5             y, y_pred, display_labels=labels, normalize='true',
6             values_format='.2%', include_values=True, cmap='Blues',
7             xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\nBayesian Gaussian Mixture ',
9                   fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13 plt.tight_layout()
14 plt.show()

Output []:
```



Παρατηρούμε παρόμοια αποτελέσματα με αυτά του μοντέλου Gaussian Mixture, κάτι που δεν προκαλεί ιδιαίτερη έκπληξη, μιας και πρόκειται για μοντέλα ίδιας τάξης.

4.4 Principal Component Analysis - PCA

Οι παραπάνω τεχνικές ομαδοποίησης, στα πλαίσια Μη-Επιβλεπόμενης Μάθησης έδωσαν ικανοποιητικά αποτελέσματα ως προς την ομαδοποίηση των δεδομένων.

Ωστόσο, λόγω του πλήθους των χαρακτηριστικών, είναι αδύνατον να έχουμε μια γενικότερη οπτικοποίηση ολόκληρου του συνόλου δεδομένων, πέρα από τα διάφορα γραφήματα που παρουσιάστηκαν για κάθε χαρακτηριστικό ξεχωριστά κατά την πρώτη εφαρμογή.

Επομένως, επιλέχτηκε η δημοφιλής μέθοδος PCA για να γίνει μείωση των πέντε διαστάσεων σε δυο διαστάσεις με τον παρακάτω τρόπο:

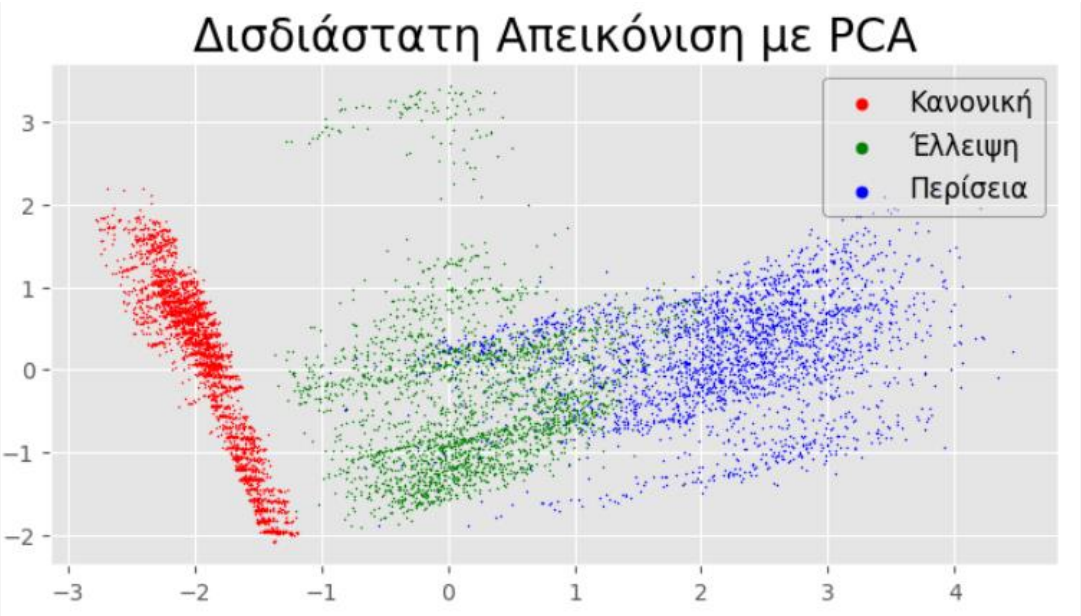
```
Input []:
1  pca = PCA(n_components=2)
2  pca.fit(X_scaled)
3  X_new = pca.transform(X_scaled)
```

Έπειτα, δημιουργείται ένα αντικείμενο τύπου DataFrame για να γίνει κατηγοριοποίηση των νέων δισδιάστατων δεδομένων:

```
Input []:
1  X_df = pd.DataFrame(X_new)
2  y_df = pd.DataFrame(y, columns=['Bearing State'])
3  df = pd.concat([X_df, y_df], axis=1)
4
5  data_grouped = df.groupby('Bearing State')
6  state_0 = data_grouped.get_group(0)
7  state_1 = data_grouped.get_group(1)
8  state_2 = data_grouped.get_group(2)
```

Τελικά, οπτικοποιούνται τα νέα μετασχηματισμένα δεδομένα:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(8,4))
3  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
4              label='Κανονική', color='red')
5  plt.scatter(state_1[0], state_1[1], s=1, marker='.',
6              label='Έλλειψη', color='green')
7  plt.scatter(state_2[0], state_2[1], s=1, marker='.',
8              label='Περίσεια', color='blue')
9  plt.title('Δισδιάστατη Απεικόνιση μέσω PCA', fontsize='20')
10 leg = plt.legend(loc='upper right', fontsize='12',
11                  markerscale=10, framealpha=0.5)
12 leg.get_frame().set_edgecolor('black')
13 fig.tight_layout()
14 plt.show()

Output []:
```



Με μια πρώτη ματιά, παρατηρούμε την πρώτη κανονική κατάσταση να ξεχωρίζει, κάτι που επιβεβαιώνεται και από τις γραφικές παραστάσεις της πρώτης εφαρμογής, ενώ οι δυο καταστάσεις βλάβης φαίνεται να έχουν ένα μεγάλο κομμάτι όπου επικαλύπτονται, και πολύ πιθανόν να περιπλέξει τις διαδικασίες μάθησης των κλασσικών μοντέλων.

4.4.1 K-Means

Γίνεται αρχικοποίηση ενός μοντέλου K-Means από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή:

```
Input []:
1  kmeans = KMeans(n_clusters=3, n_init=10)
2  kmeans.fit(X_new)
3  y_pred = kmeans.predict(X_new)
```

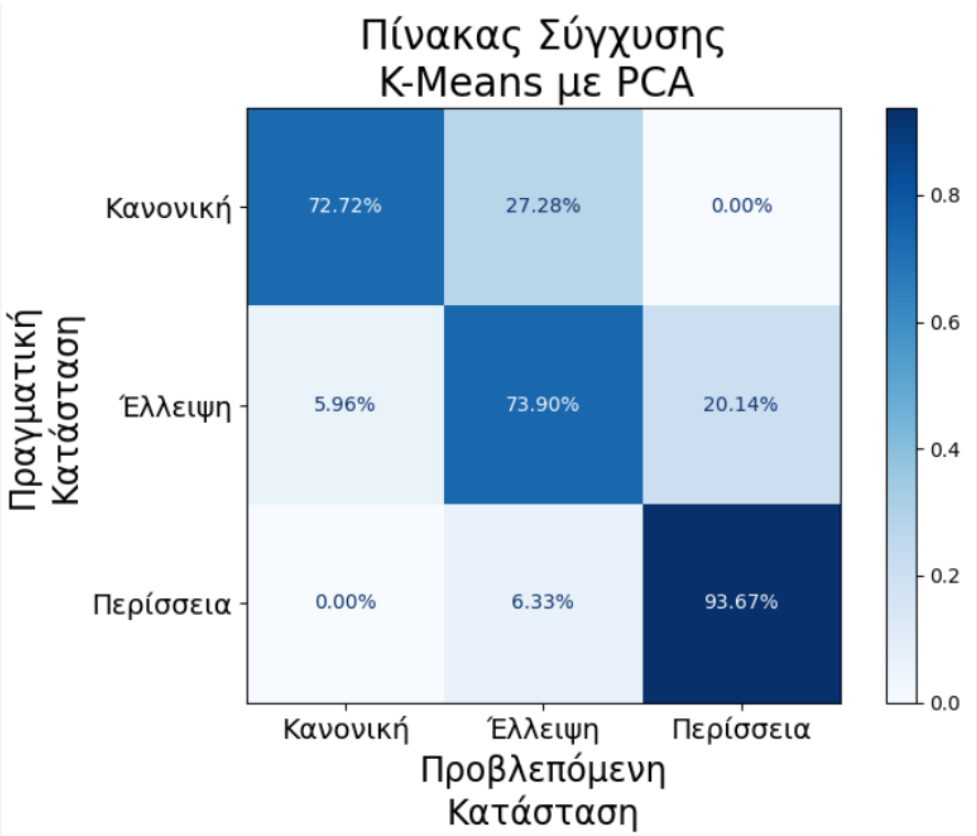
Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα κέντρα που αναγνώρισε, όλες τις ετικέτες, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των επαναλήψεων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', kmeans.get_params(deep=True))
2  print('Cluster Centers: \n', kmeans.cluster_centers_)
3  print('Labels: ', kmeans.labels_)
```

```
4  print('Predicted Clusters: ', np.unique(y_pred))
5  print('Iterations: ', kmeans.n_iter_)
6  print('Features: ', kmeans.n_features_in_)

Output []:
Parameters:  {'algorithm': 'lloyd', 'copy_x': True, 'init': 'k-means++',
              'max_iter': 300, 'n_clusters': 3, 'n_init': 10,
              'random_state': None, 'tol': 0.0001, 'verbose': 0}
Cluster Centers:
 [[-1.97302955  0.71336961]
 [-0.43455084 -0.80940708]
 [ 2.11234404  0.20664551]]
Labels:  [0 0 0 ... 2 2 2]
Predicted Clusters:  [0 1 2]
Iterations:  7
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5             y, y_pred, display_labels=labels, normalize='true',
6             values_format='.2%', include_values=True, cmap='Blues',
7             xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\nK-Means με PCA ',
9                   fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13
14 plt.tight_layout()
15 plt.show()

Output []:
```



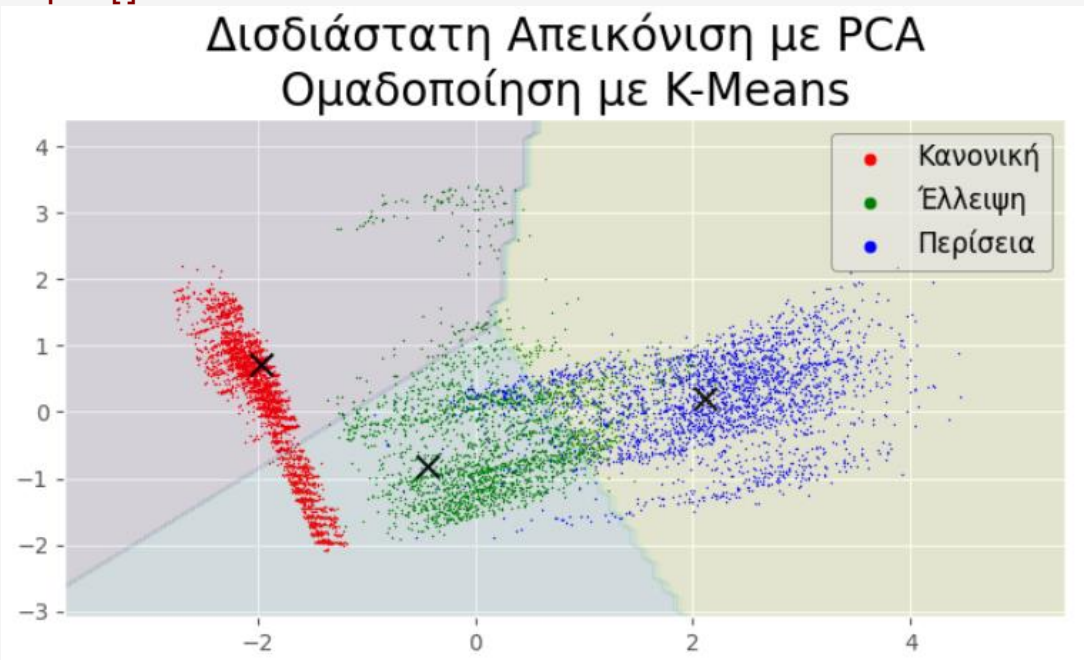
Παρατηρούνται αμέσως οι χειρότερες επιδόσεις του μοντέλου, κάτι το οποίο ήταν αναμενόμενο, από τη δισδιάστατη αναπαράσταση του νέου μετασχηματισμένου συνόλου δεδομένων, και μάλιστα παρουσιάζονται και μεγάλα ποσοστά πιθανοτήτων σφαλμάτων Τύπου I και Τύπου II.

Παρακάτω απεικονίζονται και πάλι τα δισδιάστατα δεδομένα μαζί με τα κέντρα και τα όρια των τριών ομάδων τα οποία αναγνώρισε το μοντέλο K-Means:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(8,4))
3  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
4              label='Κανονική', color='red')
5  plt.scatter(state_1[0], state_1[1], s=1, marker='.',
6              label='Έλλειψη', color='green')
7  plt.scatter(state_2[0], state_2[1], s=1, marker='.',
8              label='Περίσεια', color='blue')
9  plt.scatter(kmeans.cluster_centers_[:,0],
10             kmeans.cluster_centers_[:,1],
11             s=100, marker='x', color='black')
12 plt.title('Δισδιάστατη Απεικόνιση μέσω PCA\n
13           Ομαδοποίηση με K-Means', fontsize='20')
14 leg = plt.legend(loc='upper right', fontsize='12',
```

```
                  markerscale=10, framealpha=0.5)
leg.get_frame().set_edgecolor('black')

# Create a meshgrid for visualization
x_min, x_max = X_new[:, 0].min() - 1, X_new[:, 0].max() + 1
y_min, y_max = X_new[:, 1].min() - 1, X_new[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Predict cluster labels for each point in the meshgrid
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundaries
plt.contourf(xx, yy, Z, alpha=0.1, cmap='viridis')

fig.tight_layout()
plt.show()
```

Output []:



Μπορούμε επομένως να επιβεβαιώσουμε και τα αποτελέσματα του Πίνακα Σύγχυσης από την παραπάνω ομαδοποίηση.

4.4.2 Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια

προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

```
Input []:
1  gm = GaussianMixture(n_components=3, n_init=10)
2  gm.fit(X_new)
3  y_pred = gm.predict(X_new)
4  y_pred_prob = gm.predict_proba(X_new)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

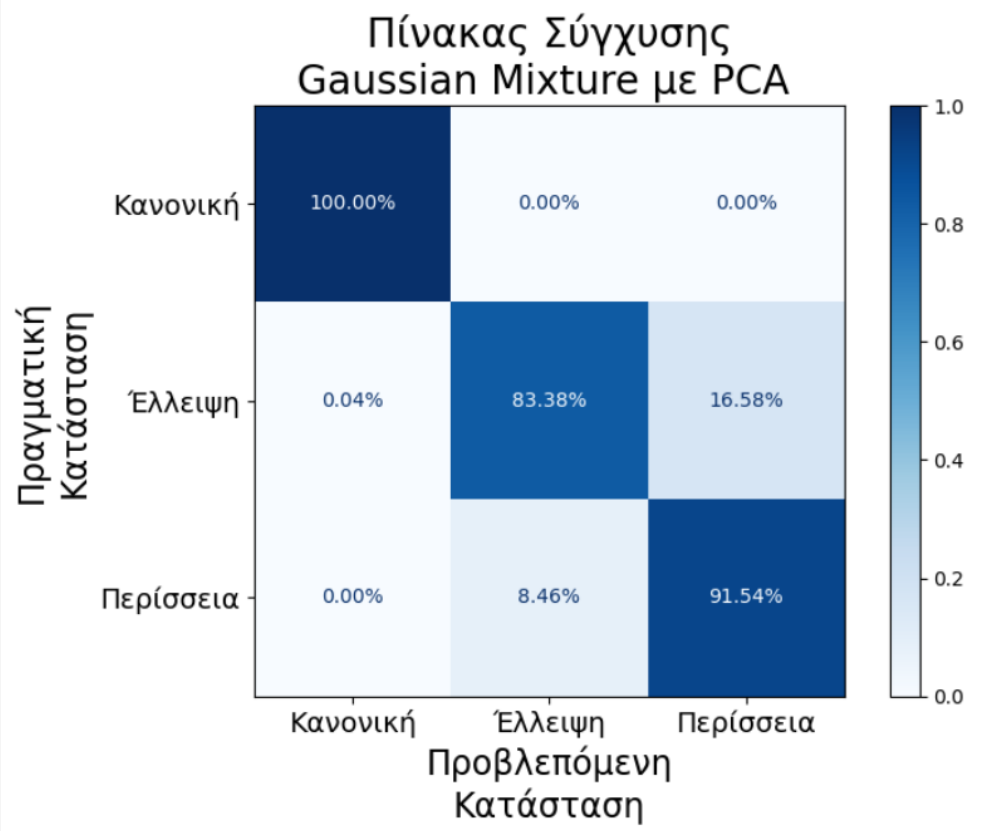
```
Input []:
1  print('Parameters: ', gm.get_params(deep=True))
2  print('Weights: ', gm.weights_)
3  print('Converged: ', gm.converged_)
4  print('Iterations to Converge: ', gm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', gm.n_features_in_)

Output []:
Parameters:  {'covariance_type': 'full', 'init_params': 'kmeans',
              'max_iter': 100, 'means_init': None, 'n_components': 3,
              'n_init': 10, 'precisions_init': None, 'random_state': None,
              'reg_covar': 1e-06, 'tol': 0.001, 'verbose': 0,
              'verbose_interval': 10, 'warm_start': False,
              'weights_init': None}
Weights:  [0.38376712 0.27689762 0.33933526]
Converged:  True
Iterations to Converge:  12
Predicted Propabilities:
 [[9.99660467e-01 3.38419741e-04 1.11279283e-06]
 [9.99594173e-01 4.04663982e-04 1.16347743e-06]
 [9.99630036e-01 3.68814575e-04 1.14991712e-06]]
Predicted Clusters:  [0 1 2]
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
```

```
plt.style.use('default')
fig, ax = plt.subplots(figsize=(8, 6))
labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
ConfusionMatrixDisplay.from_predictions(
           y, y_pred, display_labels=labels, normalize='true',
           values_format='.2%', include_values=True, cmap='Blues',
           xticks_rotation='horizontal', ax=ax, colorbar=True)
_ = ax.set_title('Πίνακας Σύγχυσης\nGaussian Mixture με PCA ',
                 fontsize='20')
plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
plt.tick_params(labelsize='14')
plt.tight_layout()
plt.show()
```

Output []:



Σε σχέση με το προηγούμενο μοντέλο K-Means βλέπουμε σημαντική βελτίωση, ωστόσο λίγο χειρότερη από τα μοντέλα πριν τη μείωση των διαστάσεων με τη μέθοδο PCA.

Πρέπει να αναφερθεί ότι, τα μοντέλα τύπου Gaussian Mixture, με την προσαρμογή τους στο σύνολο δεδομένων, αναγνωρίζουν μέσες τιμές και πίνακες συνδιακύμανσης για κάθε μια ομάδα που ζητείται να αναγνωρίσουν, καθώς διαχωρίζουν τα διάφορα δεδομένα ανάλογα

με την κατανομή τους. Οι μέσες τιμές και οι πίνακες συνδιακύμανσης παρουσιάζονται παρακάτω:

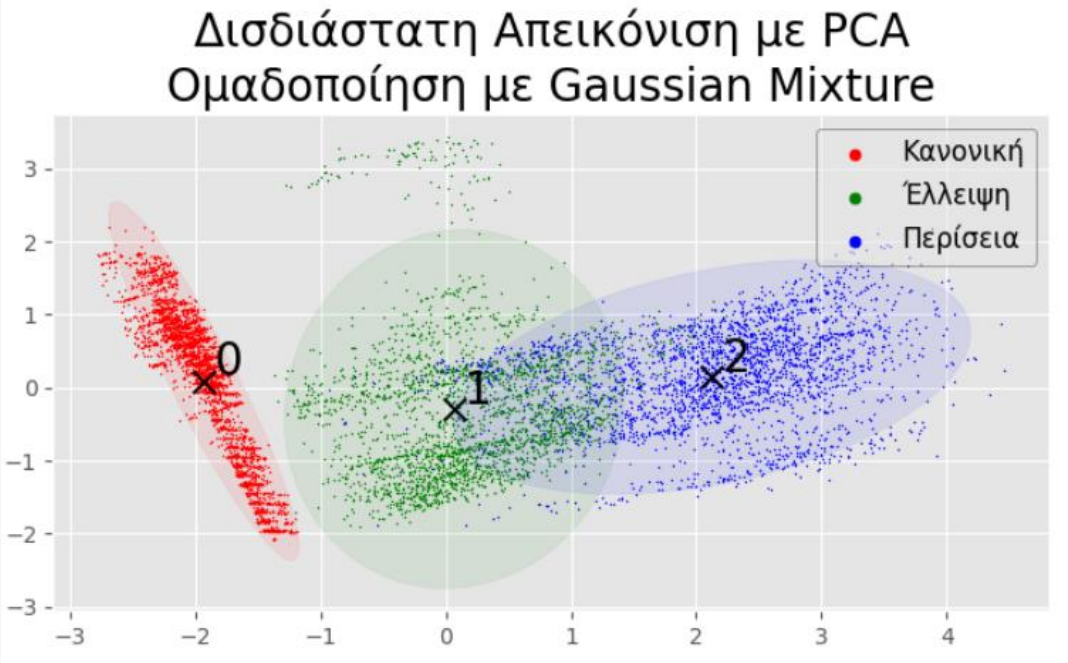
```
Input []:
1  print(gm.means_)
2  print(gm.covariances_)

Output []:
[[-1.93027677  0.08796886]
 [ 0.06710939 -0.29535789]
 [ 2.12826199  0.14152476]]
[[[ 0.09605066 -0.28716715]
  [-0.28716715  1.00808771]]

 [[ 0.30623009  0.02938661]
  [ 0.02938661  1.00688415]]

 [[ 0.70995967  0.20779924]
  [ 0.20779924  0.42839525]]]
```

Συνεπώς, με αυτήν την πληροφορία μπορούμε να απεικονίσουμε και πάλι τα δισδιάστατα δεδομένα μαζί με τις μέσες τιμές και τις γενικές κατευθύνσεις των ελλειψοειδών που αναγνωρίστηκαν:

```
Input []:
1  from matplotlib.patches import Ellipse
2
3  means = gm.means_
4  covariances = gm.covariances_
5
6  plt.style.use('ggplot')
7  plt.figure(figsize=(8,4))
8  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
9              label='Κανονική', color='red')
10 plt.scatter(state_1[0], state_1[1], s=1, marker='.',
11             label='Έλλειψη', color='green')
12 plt.scatter(state_2[0], state_2[1], s=1, marker='.',
13             label='Περίσεια', color='blue')
14 plt.scatter(gm.means_[0,0], gm.means_[0,1],
15             s=100, marker='x', color='black')
16 plt.text(gm.means_[0,0]+.08, gm.means_[0,1]+.08, '0', fontsize=20)
17 plt.scatter(gm.means_[1,0], gm.means_[1,1],
18             s=100, marker='x', color='black')
19 plt.text(gm.means_[1,0]+.08, gm.means_[1,1]+.08, '1', fontsize=20)
20 plt.scatter(gm.means_[2,0], gm.means_[2,1],
21             s=100, marker='x', color='black')
```

```
22 plt.text(gm.means_[2,0]+.08, gm.means_[2,1]+.08, '2', fontsize=20)
23 plt.title('Δισδιάστατη Απεικόνιση με PCA\n
24           Ομαδοποίηση με Gaussian Mixture', fontsize='20')
25 leg = plt.legend(loc='upper right', fontsize='12',
26                  markerscale=10, framealpha=0.5)
27 leg.get_frame().set_edgecolor('black')
28
29 for i, (mean, cov_matrix) in enumerate(zip(means, covariances)):
30     eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
31     angle = np.degrees(np.arctan2(eigenvectors[1, 0],
32                                   eigenvectors[0, 0]))
33     width = 2 * np.sqrt(5.991 * eigenvalues[0])
34                                         # 95% confidence interval
35     height = 2 * np.sqrt(5.991 * eigenvalues[1])
36                                         # 95% confidence interval
37     if i==0: ellipse = Ellipse(mean, width, height, angle,
38                                color='red', alpha=0.08)
39     if i==1: ellipse = Ellipse(mean, width, height, angle,
40                                color='green', alpha=0.08)
41     if i==2: ellipse = Ellipse(mean, width, height, angle,
42                                color='blue', alpha=0.08)
43     plt.gca().add_patch(ellipse)
44
45 fig.tight_layout()
46 plt.show()
```

Output []:



Βλέπουμε τις επικαλύψεις των δυο καταστάσεων βλαβών οι οποίες επιβεβαιώνουν και πάλι τα αποτελέσματα του Πίνακα Σύγχυσης.

4.4.3 Bayesian Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Bayesian Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

```
Input []:
1  bgm = BayesianGaussianMixture(n_components=3, n_init=10)
2  bgm.fit(X_new)
3  y_pred = bgm.predict(X_new)
4  y_pred_prob = bgm.predict_proba(X_new)
```

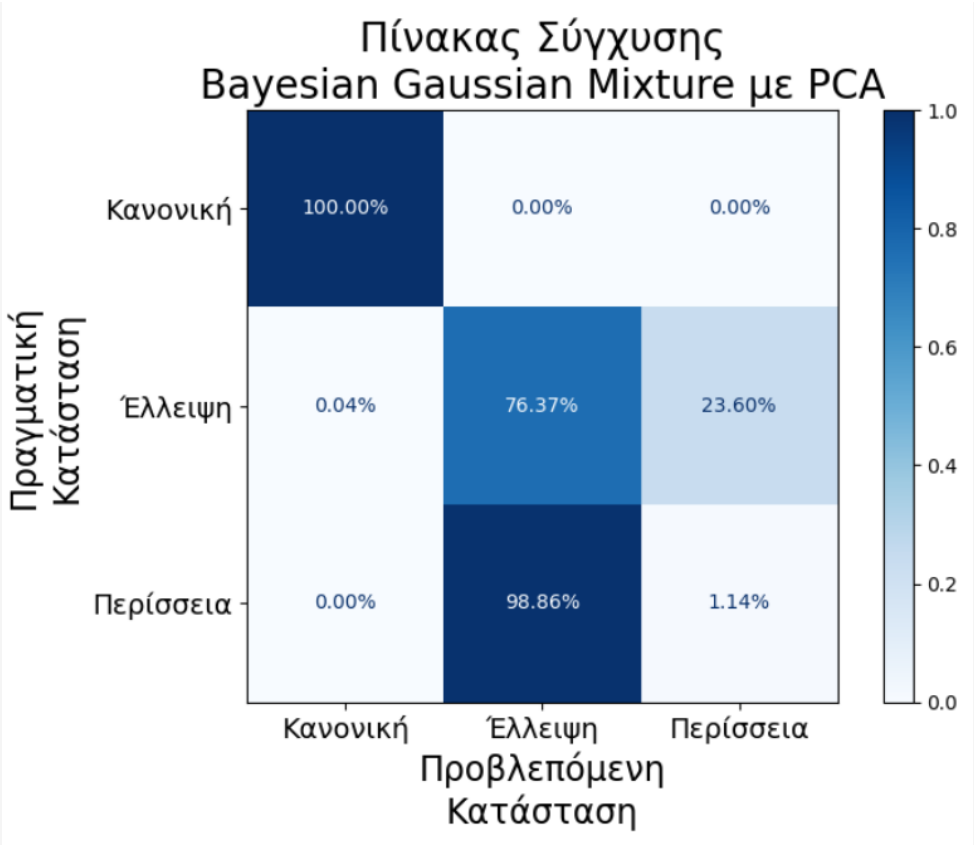
Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', bgm.get_params(deep=True))
2  print('Weights: ', bgm.weights_)
3  print('Converged: ', bgm.converged_)
4  print('Iterations to Converge: ', bgm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', bgm.n_features_in_)

Output []:
Parameters:  {'covariance_prior': None, 'covariance_type': 'full',
              'degrees_of_freedom_prior': None, 'init_params': 'kmeans',
              'max_iter': 100, 'mean_precision_prior': None,
              'mean_prior': None, 'n_components': 3, 'n_init': 10,
              'random_state': None, 'reg_covar': 1e-06, 'tol': 0.001,
              'verbose': 0, 'verbose_interval': 10, 'warm_start': False,
              'weight_concentration_prior': None,
              'weight_concentration_prior_type': 'dirichlet_process'}
Weights:  [0.38163775 0.5071448  0.11121745]
```

```
Converged:  True
Iterations to Converge:  100
Predicted Propabilities:
 [[9.98465485e-01 1.52608913e-03 8.42611079e-06]
 [9.98596933e-01 1.39106532e-03 1.20020615e-05]
 [9.98524262e-01 1.46557053e-03 1.01674100e-05]]
Predicted Clusters:  [0 1 2]
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5              y, y_pred, display_labels=labels, normalize='true',
6              values_format='.2%',include_values=True, cmap='Blues',
7              xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\n Bayesian
9                    Gaussian Mixture με PCA', fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13 plt.tight_layout()
14 plt.show()

Output []:
```



Παρατηρούμε μια πάρα πολύ κακή ομαδοποίηση, όπου και οι δυο καταστάσεις βλάβης έχουν συγχωνευτεί σε μια ομάδα, και συγκεκριμένα στην προβλεπόμενη ομάδα της έλλειψης λιπαντικού.

Παρακάτω παρουσιάζονται οι μέσες τιμές και οι πίνακες συνδιακύμανσης της κάθε ομάδας:

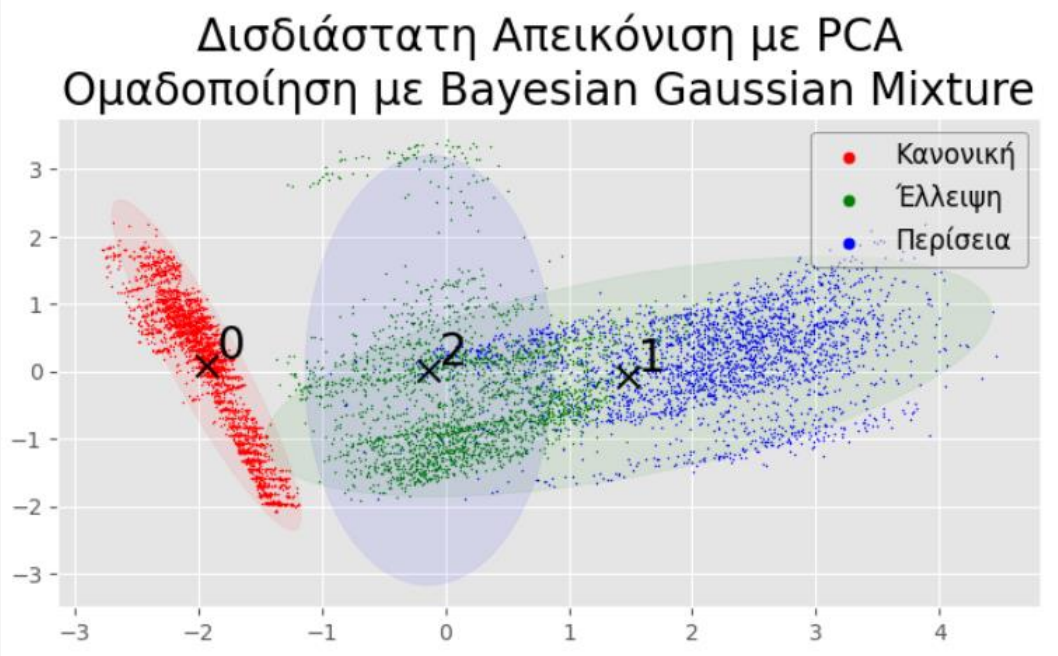
```
Input []:
1  print(bgm.means_)
2  print(bgm.covariances_)

Output []:
[[-1.93211792  0.09659719]
 [ 1.48228139 -0.07664045]
 [-0.13025007  0.01805042]]
[[[ 0.0975697  -0.28480085]
  [-0.28480085  0.9996259 ]]

 [[ 1.44765532  0.46773732]
  [ 0.46773732  0.52828961]]

 [[ 0.1680268   0.00473293]
  [ 0.00473293  1.69499044]]]
```

Συνεπώς, η τελική δισδιάστατη απεικόνιση με τα ελλειψοειδή παρουσιάζεται παρακάτω:

```
Input []:
from matplotlib.patches import Ellipse

means = bgm.means_
covariances = bgm.covariances_

plt.style.use('ggplot')
plt.figure(figsize=(8,4))
plt.scatter(state_0[0], state_0[1], s=1, marker='.',
            label='Κανονική', color='red')
plt.scatter(state_1[0], state_1[1], s=1, marker='.',
            label='Έλλειψη', color='green')
plt.scatter(state_2[0], state_2[1], s=1, marker='.',
            label='Περίσεια', color='blue')
plt.scatter(bgm.means_[0,0], bgm.means_[0,1],
            s=100, marker='x', color='black')
plt.text(bgm.means_[0,0]+.08, bgm.means_[0,1]+.08, '0', fontsize=20)
plt.scatter(bgm.means_[1,0], bgm.means_[1,1],
            s=100, marker='x', color='black')
plt.text(bgm.means_[1,0]+.08, bgm.means_[1,1]+.08, '1', fontsize=20)
plt.scatter(bgm.means_[2,0], bgm.means_[2,1],
            s=100, marker='x', color='black')
plt.text(bgm.means_[2,0]+.08, bgm.means_[2,1]+.08, '2', fontsize=20)
plt.title('Δισδιάστατη Απεικόνιση με PCA\n
           Ομαδοποίηση με Bayesian Gaussian Mixture', fontsize='20')
leg = plt.legend(loc='upper right', fontsize='12',
                 markerscale=10, framealpha=0.5)
leg.get_frame().set_edgecolor('black')

for i, (mean, cov_matrix) in enumerate(zip(means, covariances)):
    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
    angle = np.degrees(np.arctan2(eigenvectors[1, 0],
                                  eigenvectors[0, 0]))
    width = 2 * np.sqrt(5.991 * eigenvalues[0])
                                     # 95% confidence interval
    height = 2 * np.sqrt(5.991 * eigenvalues[1])
                                     # 95% confidence interval
    if i==0: ellipse = Ellipse(mean, width, height, angle,
                               color='red', alpha=0.08)
    if i==1: ellipse = Ellipse(mean, width, height, angle,
                               color='green', alpha=0.08)
    if i==2: ellipse = Ellipse(mean, width, height, angle,
                               color='blue', alpha=0.08)
    plt.gca().add_patch(ellipse)

fig.tight_layout()
```

```
46 plt.show()
```

Output []:



Παρατηρείται μια πολύ πιο έντονη επικάλυψη μεταξύ των δυο καταστάσεων βλαβών, σε σχέση με το αντίστοιχο διάγραμμα στην περίπτωση του μοντέλου Gaussian Mixture του προηγούμενου υποκεφαλαίου, που για ακόμη μια φορά επιβεβαιώνει το αποτέλεσμα του Πίνακα Σύγχυσης.

4.5 PCA Τεσσάρων Χαρακτηριστικών

Παρατηρήθηκε ότι η μείωση των διαστάσεων από τις πέντε στις δύο οδήγησε σε μια κακή περίπτωση δισδιάστατης αναπαράστασης με έντονο το στοιχείο της επικάλυψης μεταξύ των καταστάσεων βλαβών, και μάλιστα με μεγάλη διασπορά στα στοιχεία της κάθε κατάστασης.

Συνεπώς, μέσα από δοκιμές, αφαιρέθηκε από το σύνολο δεδομένων X_scaled η στήλη που αντιστοιχεί στο χαρακτηριστικό της θερμοκρασίας και έγινε επανάληψη της παραπάνω διαδικασίας. Δηλαδή το νέο σύνολο των τεσσάρων χαρακτηριστικών είναι αυτό το οποίο μετασχηματίστηκε μέσω της μεθόδου PCA στη δισδιάστατη αναπαράσταση, πριν γίνει η προσαρμογή στα τρία μοντέλα Μηχανικής Μάθησης:

```
Input []:
1  pca = PCA(n_components=2)
2  pca.fit(X_scaled[:,[0,1,2,3]])
```

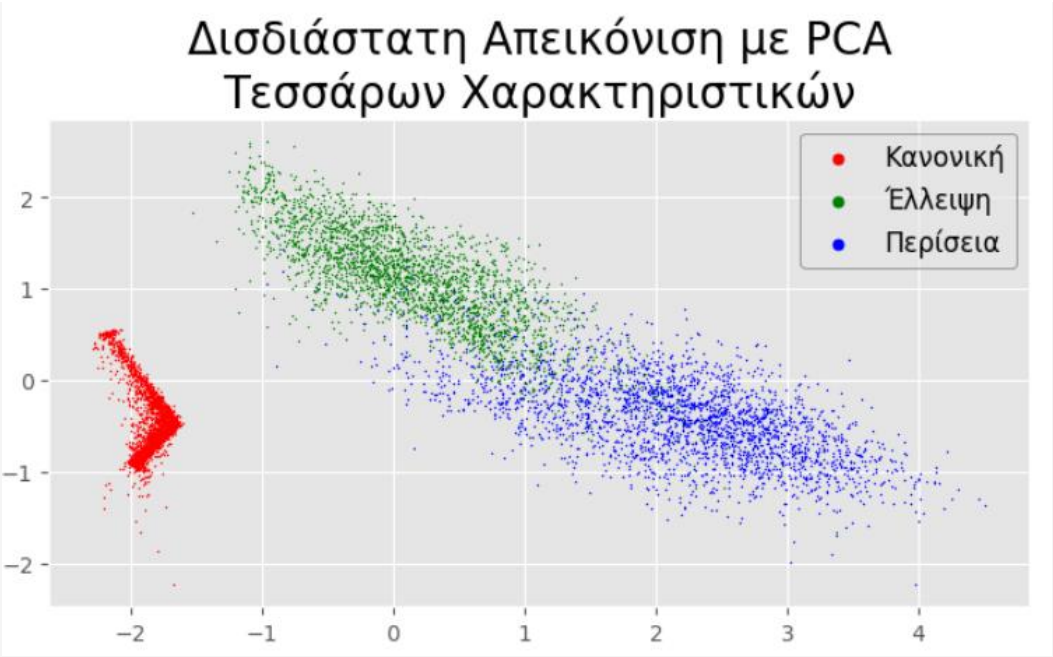
```
3  X_new = pca.transform(X_scaled[:,[0,1,2,3]])
```

Στη συνέχεια, δημιουργείται και πάλι ένα αντικείμενο τύπου DataFrame για να γίνει κατηγοριοποίηση των νέων δισδιάστατων δεδομένων:

```
Input []:
1  X_df = pd.DataFrame(X_new)
2  y_df = pd.DataFrame(y, columns=['Bearing State'])
3  df = pd.concat([X_df, y_df], axis=1)
4
5  data_grouped = df.groupby('Bearing State')
6  state_0 = data_grouped.get_group(0)
7  state_1 = data_grouped.get_group(1)
8  state_2 = data_grouped.get_group(2)
```

Τελικά, οπτικοποιούνται τα νέα μετασχηματισμένα δεδομένα:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(8,4))
3  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
4              label='Κανονική', color='red')
5  plt.scatter(state_1[0], state_1[1], s=1, marker='.',
6              label='Έλλειψη', color='green')
7  plt.scatter(state_2[0], state_2[1], s=1, marker='.',
8              label='Περίσεια', color='blue')
9  plt.title('Δισδιάστατη Απεικόνιση με PCA\n
10           Τεσσάρων Χαρακτηριστικών', fontsize='20')
11 leg = plt.legend(loc='upper right', fontsize='12',
12                  markerscale=10, framealpha=0.5)
13 leg.get_frame().set_edgecolor('black')
14 fig.tight_layout()
15 plt.show()

Output []:
```



Επομένως, παρατηρούμε ότι, πέρα από την κανονική λιπαντική κατάσταση η οποία και πάλι ξεχωρίζει, τα σημεία των δυο καταστάσεων βλάβης δεν παρουσιάζουν τόσο έντονη διασπορά όσο στην προηγούμενη περίπτωση, και φαίνεται να είναι συσσωρευμένα σε διακριτά ελλειψοειδή όρια. Ωστόσο, παραμένουν κάποια στοιχεία επικάλυψης, αλλά όχι σε τόσο σημαντικό βαθμό.

4.5.1 K-Means

Γίνεται αρχικοποίηση του μοντέλου K-Means από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new τα οποία πλέον προήλθαν από τον μετασχηματισμό των τεσσάρων χαρακτηριστικών, και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή:

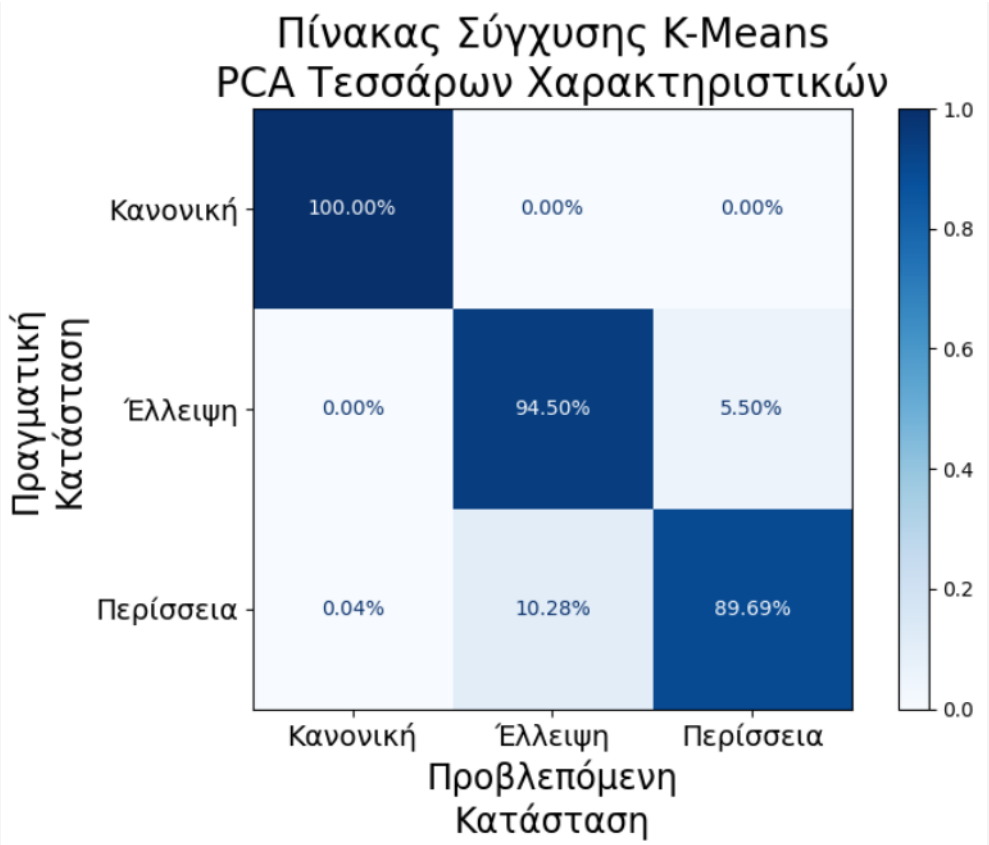
```
Input []:
1  kmeans = KMeans(n_clusters=3, n_init=10)
2  kmeans.fit(X_new)
3  y_pred = kmeans.predict(X_new)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα κέντρα που αναγνώρισε, όλες τις ετικέτες, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των επαναλήψεων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', kmeans.get_params(deep=True))
2  print('Cluster Centers: \n', kmeans.cluster_centers_)
3  print('Labels: ', kmeans.labels_)
4  print('Predicted Clusters: ', np.unique(y_pred))
5  print('Iterations: ', kmeans.n_iter_)
6  print('Features: ', kmeans.n_features_in_)

Output []:
Parameters:  {'algorithm': 'lloyd', 'copy_x': True, 'init': 'k-means++',
              'max_iter': 300, 'n_clusters': 3, 'n_init': 10,
              'random_state': None, 'tol': 0.0001, 'verbose': 0}
Cluster Centers:
 [[-1.86690351 -0.54169221]
 [ 0.13533371  1.06654818]
 [ 2.30853721 -0.46575326]]
Labels:  [0 0 0 ... 2 2 2]
Predicted Clusters:  [0 1 2]
Iterations:  6
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5              y, y_pred, display_labels=labels, normalize='true',
6              values_format='.2%', include_values=True, cmap='Blues',
7              xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης K-Means\n
9                   PCA Τεσσάρων Χαρακτηριστικών', fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13
14 plt.tight_layout()
15 plt.show()

Output []:
```



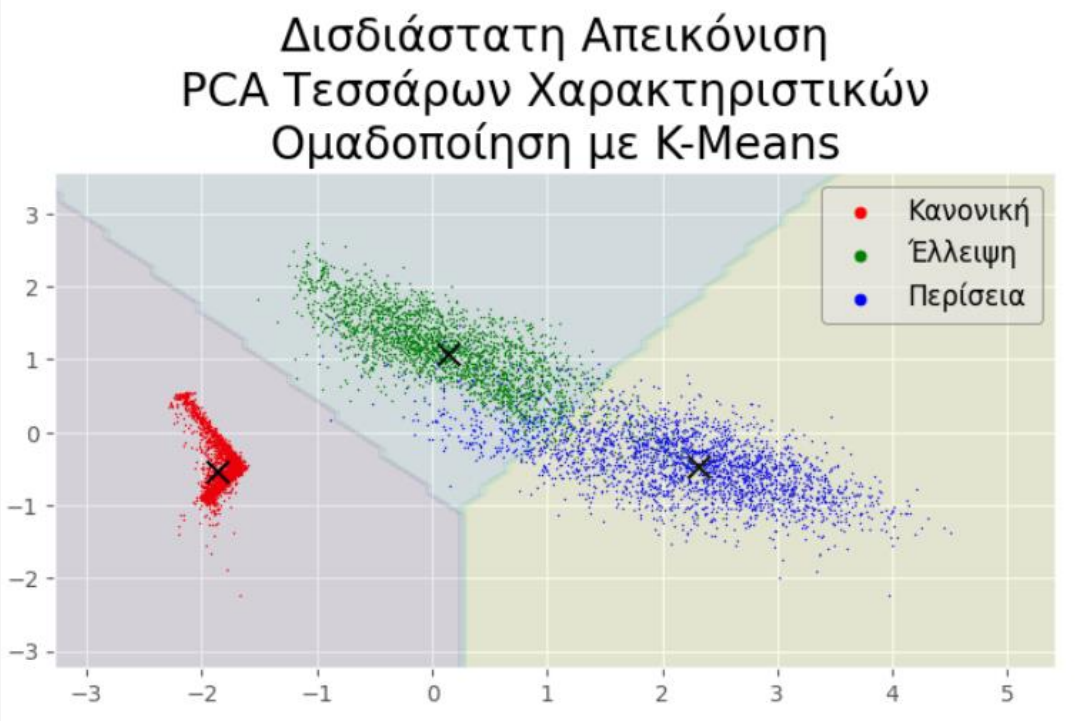
Παρατηρείται αμέσως η καλύτερη επίδοση του αλγορίθμου, σε σχέση με τις επιδόσεις κατά την περίπτωση όπου εφαρμόστηκε η μέθοδος PCA στα πέντε χαρακτηριστικά. Μάλιστα, τα αποτελέσματα είναι παρόμοια με αυτά του πρώτου μοντέλου K-Means όπου δεν έγινε κάποιος ιδιαίτερος μετασχηματισμός στα δεδομένα, πέρα από την τυπική αλλαγή κλίμακας των τιμών.

Παρακάτω απεικονίζονται και πάλι τα νέα δισδιάστατα δεδομένα που προέκυψαν από τη μέθοδο PCA στα τέσσερα χαρακτηριστικά, μαζί με τα κέντρα και τα όρια των τριών ομάδων τα οποία αναγνώρισε το μοντέλο K-Means:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(8,4))
3  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
4              label='Κανονική', color='red')
5  plt.scatter(state_1[0], state_1[1], s=1, marker='.',
6              label='Έλλειψη', color='green')
7  plt.scatter(state_2[0], state_2[1], s=1, marker='.',
8              label='Περίσεια', color='blue')
9  plt.scatter(kmeans.cluster_centers_[:,0],
10             kmeans.cluster_centers_[:,1],
11             s=100, marker='x', color='black')
12 plt.title('Δισδιάστατη Απεικόνιση\n
```

```
          PCA Τεσσάρων Χαρακτηριστικών\n
          Ομαδοποίηση με K-Means', fontsize='20')
leg = plt.legend(loc='upper right', fontsize='12',
                 markerscale=10, framealpha=0.5)
leg.get_frame().set_edgecolor('black')

# Create a meshgrid for visualization
x_min, x_max = X_new[:, 0].min() - 1, X_new[:, 0].max() + 1
y_min, y_max = X_new[:, 1].min() - 1, X_new[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Predict cluster labels for each point in the meshgrid
Z = kmeans.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundaries
plt.contourf(xx, yy, Z, alpha=0.1, cmap='viridis')

fig.tight_layout()
plt.show()
```

Output []:



Μπορούμε επομένως να επιβεβαιώσουμε και τα αποτελέσματα του Πίνακα Σύγχυσης από την παραπάνω ομαδοποίηση, και ειδικότερα φαίνεται καθαρά το 10.28% σφάλματος της κατάστασης υπερβολικής λίπανσης το οποίο αναγνωρίστηκε ως κατάσταση έλλειψης.

4.5.2 Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new τα οποία πλέον προήλθαν από τον μετασχηματισμό των τεσσάρων χαρακτηριστικών, και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

```
Input []:
1  gm = GaussianMixture(n_components=3, n_init=10)
2  gm.fit(X_new)
3  y_pred = gm.predict(X_new)
4  y_pred_prob = gm.predict_proba(X_new)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

```
Input []:
1  print('Parameters: ', gm.get_params(deep=True))
2  print('Weights: ', gm.weights_)
3  print('Converged: ', gm.converged_)
4  print('Iterations to Converge: ', gm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', gm.n_features_in_)

Output []:
Parameters:  {'covariance_type': 'full', 'init_params': 'kmeans',
              'max_iter': 100, 'means_init': None, 'n_components': 3,
              'n_init': 10, 'precisions_init': None, 'random_state': None,
              'reg_covar': 1e-06, 'tol': 0.001, 'verbose': 0,
              'verbose_interval': 10, 'warm_start': False,
              'weights_init': None}
Weights:  [0.38464226 0.31364224 0.3017155 ]
Converged:  True
Iterations to Converge:  4
Predicted Propabilities:
 [[1.00000000e+00 3.20071289e-20 1.45563975e-11]
```

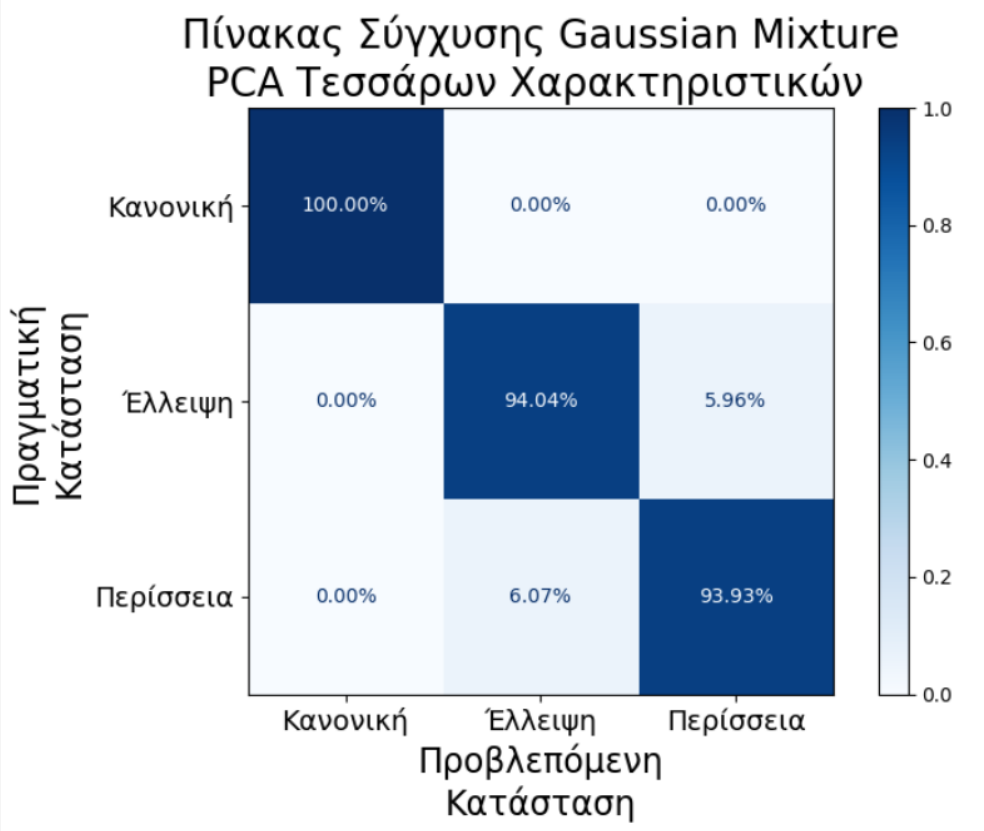
```
 [1.00000000e+00 1.71133218e-22 1.22994023e-12]
 [1.00000000e+00 2.35823576e-20 1.23492034e-11]]
Predicted Clusters:  [0 1 2]
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

Input []:

```
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5             y, y_pred, display_labels=labels, normalize='true',
6             values_format='.2%', include_values=True, cmap='Blues',
7             xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης Gaussian Mixture\n
9                   PCA Τεσσάρων Χαρακτηριστικών ', fontsize='20')
10 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
11 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
12 plt.tick_params(labelsize='14')
13 plt.tight_layout()
14 plt.show()
```

Output []:



Το συγκεκριμένα αποτελέσματα αποτελούν την πρώτη περίπτωση όπου τα ποσοστά επιτυχίας βρίσκονται στην ίδια τάξη τιμών, δηλαδή πάνω από 93%. Υπενθυμίζεται ότι η αντίστοιχη εφαρμογή χωρίς τη μέθοδο PCA εμφάνισε λίγο μεγαλύτερο ποσοστό επιτυχίας για την κατάσταση της έλλειψης (96.24%) και πολύ μικρότερο για την κατάσταση περίσσειας (86.77%), και παρά την μεγαλύτερη τιμή της δεύτερη κατάστασης, είναι προτιμότερη η παραπάνω επίδοση.

Επιπλέον, αξίζει να σημειωθεί ότι, τα παραπάνω αποτελέσματα είναι λίγο καλύτερα από αυτά της πρώτης προσπάθειας ανάπτυξης Τεχνητού Νευρωνικού Δικτύου, κάτι που φέρνει το παραπάνω μοντέλο Μη-Επιβλεπόμενης Μάθησης, στο ίδιο πλαίσιο σύγκρισης με ένα συμβατικό μοντέλο Επιβλεπόμενης Μάθησης, και πιθανότητα με λίγο πιο εξειδικευμένη παραμετροποίηση να επιτυγχάνονταν ακόμα καλύτερα αποτελέσματα.

Επίσης, τα καλύτερα αποτελέσματα του μοντέλου Gaussian Mixture ήταν αναμενόμενα, μιας και η ίδια η διαδικασία μάθησης στηρίζεται στην αναγνώριση κατανομών, και σε αυτήν την περίπτωση είναι εμφανή τα ελλειψοειδή τα οποία σχηματίζονται από τον δισδιάστατο μετασχηματισμό των δεδομένων.

Παρακάτω παρουσιάζονται οι μέσες τιμές και οι πίνακες συνδιακύμανσης της κάθε ομάδας:

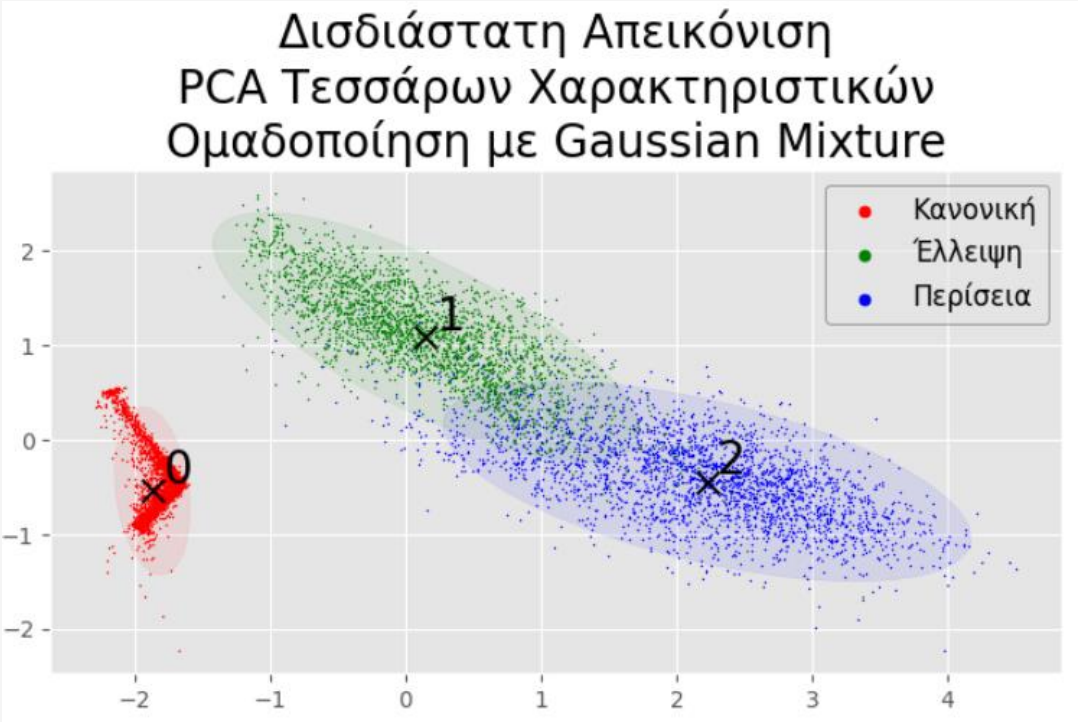
```
Input []:
1  print(gm.means_)
2  print(gm.covariances_)

Output []:
[[-1.86719979 -0.54190498]
 [ 0.14857248  1.08933858]
 [ 2.22595573 -0.44155185]]
[[[ 0.01287347 -0.009348  ]
  [-0.009348    0.13088323]]

 [[ 0.41053998 -0.25887635]
  [-0.25887635  0.2901267 ]]

 [[ 0.63224723 -0.19429289]
  [-0.19429289  0.18642023]]]
```

Συνεπώς, με αυτήν την πληροφορία μπορούμε να απεικονίσουμε και πάλι νέα δισδιάστατα δεδομένα που προέκυψαν από τη μέθοδο PCA στα τέσσερα χαρακτηριστικά, μαζί με τις μέσες τιμές και τις γενικές κατευθύνσεις των ελλειψοειδών που αναγνωρίστηκαν:

```
Input []:
1  from matplotlib.patches import Ellipse
```

```
2
3  means = gm.means_
4  covariances = gm.covariances_
5
6  plt.style.use('ggplot')
7  plt.figure(figsize=(8,4))
8  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
9              label='Κανονική', color='red')
10 plt.scatter(state_1[0], state_1[1], s=1, marker='.',
11             label='Έλλειψη', color='green')
12 plt.scatter(state_2[0], state_2[1], s=1, marker='.',
13             label='Περίσεια', color='blue')
14 plt.scatter(gm.means_[0,0], gm.means_[0,1],
15             s=100, marker='x', color='black')
16 plt.text(gm.means_[0,0]+.08, gm.means_[0,1]+.08, '0', fontsize=20)
17 plt.scatter(gm.means_[1,0], gm.means_[1,1],
18             s=100, marker='x', color='black')
19 plt.text(gm.means_[1,0]+.08, gm.means_[1,1]+.08, '1', fontsize=20)
20 plt.scatter(gm.means_[2,0], gm.means_[2,1],
21             s=100, marker='x', color='black')
22 plt.text(gm.means_[2,0]+.08, gm.means_[2,1]+.08, '2', fontsize=20)
23 plt.title('Δισδιάστατη Απεικόνιση\n
24            PCA Τεσσάρων Χαρακτηριστικών\n
25            Ομαδοποίηση με Gaussian Mixture', fontsize='20')
26 leg = plt.legend(loc='upper right', fontsize='12',
27                  markerscale=10, framealpha=0.5)
28 leg.get_frame().set_edgecolor('black')
29
30 for i, (mean, cov_matrix) in enumerate(zip(means, covariances)):
31     eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
32     angle = np.degrees(np.arctan2(eigenvectors[1, 0],
33                                   eigenvectors[0, 0]))
34     width = 2 * np.sqrt(5.991 * eigenvalues[0])
35                                         # 95% confidence interval
36     height = 2 * np.sqrt(5.991 * eigenvalues[1])
37                                         # 95% confidence interval
38     if i==0: ellipse = Ellipse(mean, width, height, angle,
39                                color='red', alpha=0.08)
40     if i==1: ellipse = Ellipse(mean, width, height, angle,
41                                color='green', alpha=0.08)
42     if i==2: ellipse = Ellipse(mean, width, height, angle,
43                                color='blue', alpha=0.08)
44     plt.gca().add_patch(ellipse)
45
46 fig.tight_layout()
47 plt.show()
```

Output []:



Επομένως, παρατηρούμε την ελλειψοειδή αναγνώριση της κατανομής των δεδομένων και από το προσαρμοσμένο πλέον μοντέλο. Παρά το γεγονός ότι υπάρχει και πάλι επικάλυψη των κατανομών, δεν εμφανίζεται σε τόσο έντονο βαθμό ώστε να διαταράσσει τα αποτελέσματα και τα σφάλματα στις προβλέψεις.

4.5.3 Bayesian Gaussian Mixture

Γίνεται αρχικοποίηση ενός μοντέλου Gaussian Mixture από το οποίο ζητείται να αναγνωρίσει 3 κλάσεις για τα δεδομένα που θα τού παρουσιαστούν, στη συνέχεια προσαρμόζεται το μοντέλο στα νέα δισδιάστατα δεδομένα X_new τα οποία πλέον προήλθαν από τον μετασχηματισμό των τεσσάρων χαρακτηριστικών, και τελικά εξάγονται οι προβλεπόμενες κλάσεις για κάθε εγγραφή, αλλά και οι πιθανότητες που οδήγησαν στις προβλέψεις:

Input []:

```
1  bgm = BayesianGaussianMixture(n_components=3, n_init=10)
2  bgm.fit(X_new)
3  y_pred = bgm.predict(X_new)
4  y_pred_prob = bgm.predict_proba(X_new)
```

Στη συνέχεια, μπορούμε να πάρουμε κάποια επιπλέον στοιχεία από το προσαρμοσμένο πλέον μοντέλο, όπως τις παραμέτρους, τα βάρη των τριών κλάσεων, το αν υπήρχε σύγκλιση του μοντέλου σε κάποιο αποτέλεσμα και πόσες επαναλήψεις χρειάστηκαν, τα αποτελέσματα των εξαγόμενων πιθανοτήτων, τις προβλεπόμενες ομάδες του συνόλου δεδομένων, το πλήθος των χαρακτηριστικών κ.α.:

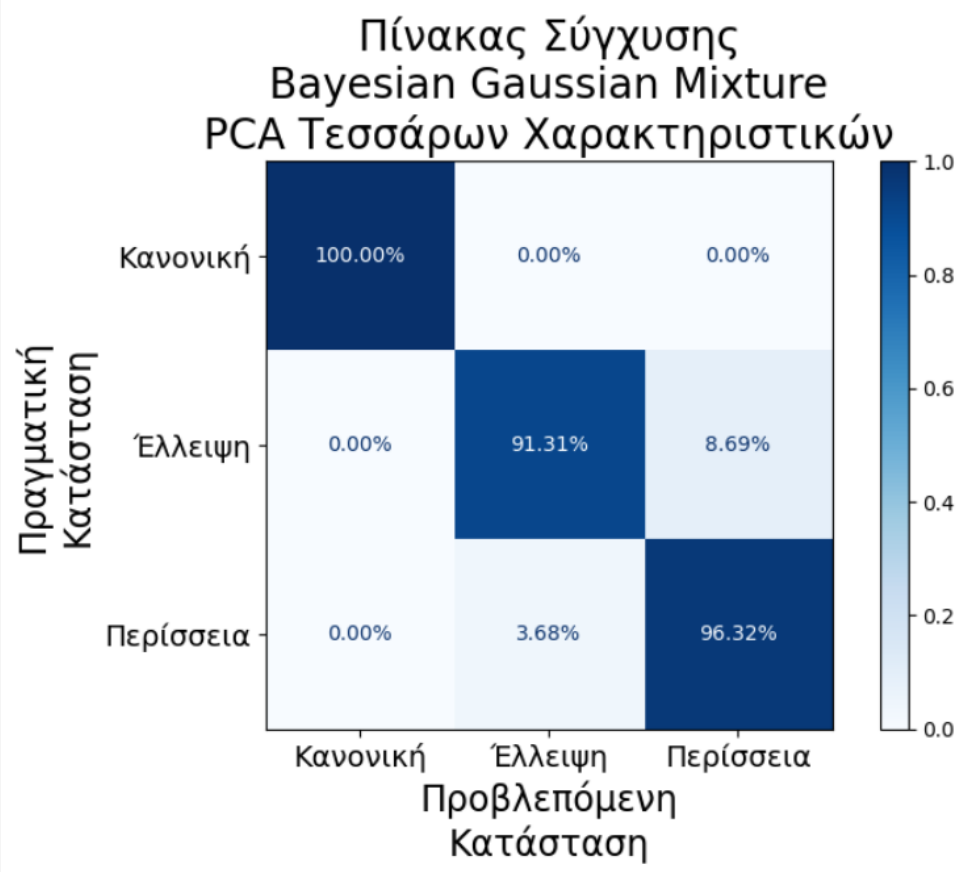
```
Input []:
1  print('Parameters: ', bgm.get_params(deep=True))
2  print('Weights: ', bgm.weights_)
3  print('Converged: ', bgm.converged_)
4  print('Iterations to Converge: ', bgm.n_iter_)
5  print('Predicted Propabilities: \n', y_pred_prob[:3])
6  print('Predicted Clusters: ', np.unique(y_pred))
7  print('Features: ', bgm.n_features_in_)

Output []:
Parameters:  {'covariance_prior': None, 'covariance_type': 'full',
              'degrees_of_freedom_prior': None, 'init_params': 'kmeans',
              'max_iter': 100, 'mean_precision_prior': None,
              'mean_prior': None, 'n_components': 3, 'n_init': 10,
              'random_state': None, 'reg_covar': 1e-06, 'tol': 0.001,
              'verbose': 0, 'verbose_interval': 10, 'warm_start': False,
              'weight_concentration_prior': None,
              'weight_concentration_prior_type': 'dirichlet_process'}
Weights:  [0.38471344 0.32332248 0.29196407]
Converged:  True
Iterations to Converge:  33
Predicted Propabilities:
 [[1.00000000e+00 4.53330824e-11 4.68726979e-22]
 [1.00000000e+00 3.11137142e-12 1.08786665e-24]
 [1.00000000e+00 3.74172647e-11 3.21684360e-22]]
Predicted Clusters:  [0 1 2]
Features:  2
```

Παρακάτω παρουσιάζεται και ο Πίνακας Σύγχυσης για το παραπάνω μοντέλο:

```
Input []:
1  plt.style.use('default')
2  fig, ax = plt.subplots(figsize=(8, 6))
3  labels = ['Κανονική', 'Έλλειψη', 'Περίσσεια']
4  ConfusionMatrixDisplay.from_predictions(
5              y, y_pred, display_labels=labels, normalize='true',
6              values_format='.2%', include_values=True, cmap='Blues',
7              xticks_rotation='horizontal', ax=ax, colorbar=True)
8  _ = ax.set_title('Πίνακας Σύγχυσης\n
```

```
9                   Bayesian Gaussian Mixture\n
10                  PCA Τεσσάρων Χαρακτηριστικών', fontsize='20')
11 plt.xlabel('Προβλεπόμενη\nΚατάσταση', fontsize='17')
12 plt.ylabel('Πραγματική\nΚατάσταση', fontsize='17')
13 plt.tick_params(labelsize='14')
14 plt.tight_layout()
15 plt.show()
```

Output []:



Από τη στιγμή που το μοντέλο Bayesian Gaussian Mixture αποτελεί εναλλακτική του μοντέλου Gaussian Mixture, και βασίζεται στην αναγνώριση κατανομών, τα παραπάνω αποτελέσματα ήταν επίσης αναμενόμενα, και παρόμοια με αυτά του μοντέλου Gaussian Mixture του προηγούμενου υποκεφαλαίου.

Υπενθυμίζεται και εδώ για λόγους σύγκρισης ότι, η αντίστοιχη εφαρμογή χωρίς τη μέθοδο PCA εμφάνισε μεγαλύτερο ποσοστό επιτυχίας για την κατάσταση της έλλειψης (96.32%) και πολύ μικρότερο για την κατάσταση περίσσειας (86.35%).

Παρακάτω παρουσιάζονται οι μέσες τιμές και οι πίνακες συνδιακύμανσης της κάθε ομάδας:

```
Input []:
1  print(bgm.means_)
2  print(bgm.covariances_)
```

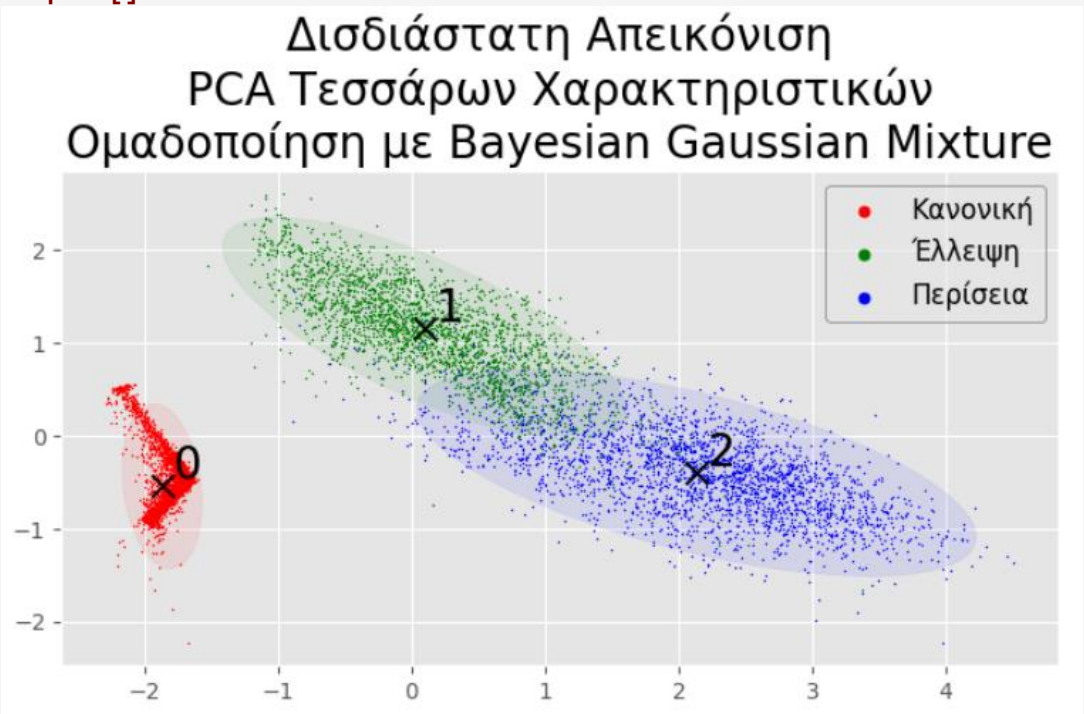
```
Output []:
[[-1.86663298 -0.54174223]
 [ 2.13321114 -0.39913741]
 [ 0.09677984  1.15524948]]
[[[ 0.01488287 -0.00903515]
  [-0.00903515  0.13111631]]

 [[ 0.73110218 -0.24100593]
  [-0.24100593  0.20678402]]

 [[ 0.38016033 -0.22210796]
  [-0.22210796  0.23972479]]]
```

Συνεπώς, η τελική δισδιάστατη απεικόνιση με τα ελλειψοειδή παρουσιάζεται παρακάτω:

```
Input []:
1  from matplotlib.patches import Ellipse
2
3  means = bgm.means_
4  covariances = bgm.covariances_
5
6  plt.style.use('ggplot')
7  plt.figure(figsize=(8,4))
8  plt.scatter(state_0[0], state_0[1], s=1, marker='.',
9              label='Κανονική', color='red')
10 plt.scatter(state_1[0], state_1[1], s=1, marker='.',
11             label='Έλλειψη', color='green')
12 plt.scatter(state_2[0], state_2[1], s=1, marker='.',
13             label='Περίσεια', color='blue')
14 plt.scatter(bgm.means_[0,0], bgm.means_[0,1],
15             s=100, marker='x', color='black')
16 plt.text(bgm.means_[0,0]+.08, bgm.means_[0,1]+.08, '0', fontsize=20)
17 plt.scatter(bgm.means_[1,0], bgm.means_[1,1],
18             s=100, marker='x', color='black')
19 plt.text(bgm.means_[1,0]+.08, bgm.means_[1,1]+.08, '2', fontsize=20)
20 plt.scatter(bgm.means_[2,0], bgm.means_[2,1],
21             s=100, marker='x', color='black')
22 plt.text(bgm.means_[2,0]+.08, bgm.means_[2,1]+.08, '1', fontsize=20)
23 plt.title('Δισδιάστατη Απεικόνιση\n
24           PCA Τεσσάρων Χαρακτηριστικών\n
25           Ομαδοποίηση με Bayesian Gaussian Mixture', fontsize='20')
26 leg = plt.legend(loc='upper right', fontsize='12',
27                  markerscale=10, framealpha=0.5)
28 leg.get_frame().set_edgecolor('black')
29
```

```
for i, (mean, cov_matrix) in enumerate(zip(means, covariances)):
    eigenvalues, eigenvectors = np.linalg.eigh(cov_matrix)
    angle = np.degrees(np.arctan2(eigenvectors[1, 0],
                                  eigenvectors[0, 0]))
    width = 2 * np.sqrt(5.991 * eigenvalues[0])
                                   # 95% confidence interval
    height = 2 * np.sqrt(5.991 * eigenvalues[1])
                                   # 95% confidence interval
    if i==0: ellipse = Ellipse(mean, width, height, angle,
                               color='red', alpha=0.08)
    if i==2: ellipse = Ellipse(mean, width, height, angle,
                               color='green', alpha=0.08)
    if i==1: ellipse = Ellipse(mean, width, height, angle,
                               color='blue', alpha=0.08)
    plt.gca().add_patch(ellipse)

fig.tight_layout()
plt.show()
```

Output []:



Επομένως, παρατηρούμε και πάλι την ελλειψοειδή αναγνώριση της κατανομής των δεδομένων και από το προσαρμοσμένο μοντέλο, αλλά και την επιβεβαίωση των αποτελεσμάτων του Πίνακα Σύγχυσης και των σφαλμάτων λόγω της επικάλυψης των κατανομών.

ΠΑΡΑΤΗΡΗΣΗ

Αναφέρθηκε ότι το παραπάνω πρόβλημα ομαδοποίησης ακολουθεί τεχνικές Μη-Επιβλεπόμενης Μάθησης, αλλά σε ένα πιο ελεγχόμενο περιβάλλον, καθώς μπορούμε να εξετάσουμε την επιτυχία των προβλέψεων γνωρίζοντας τις πραγματικές καταστάσεις.

Όλες οι παραπάνω εφαρμογές ομαδοποιούν τα δεδομένα σε κλάσεις, όπου οι ετικέτες ορίζονται με τυχαίο τρόπο ως προς τον αναγνωριστικό ακέραιο 0, 1 ή 2. Συνεπώς, για να επιτευχθεί εκείνη η περίπτωση όπου οι ετικέτες συμπίπτουν με τις πραγματικές, υπήρξε η ανάγκη πολλαπλών αρχικοποιήσεων μέχρι να εμφανιστεί η ζητούμενη. Κάτι τέτοιο δεν επηρέασε τα αποτελέσματα των μοντέλων, καθώς είναι αλγόριθμοι πιθανοτήτων και σταθερών βημάτων, οι οποίοι οδηγούν πάντοτε στα ίδια συμπεράσματα.

Ωστόσο, στην ακριβώς προηγούμενη περίπτωση ομαδοποίησης με Bayesian Gaussian Mixture μετά από τη μέθοδο PCA τεσσάρων χαρακτηριστικών, ήταν αναγκαία για τον παραπάνω λόγο η χειροκίνητη αλλαγή στις προβλεπόμενες ετικέτες αμέσως μετά το κομμάτι της αρχικοποίησης, καθώς υπήρχε δυσκολία στο να επιτευχθούν οι ζητούμενες ετικέτες μετά από πολλαπλές επαναλήψεις.

Η αλλαγή αυτή έγινε με τον παρακάτω τρόπο:

```
Input []:
1  for i in range(8569):
2      if y_pred[i]==0:
3          y_pred[i]=0
4      elif y_pred[i]==2:
5          y_pred[i]=1
6      elif y_pred[i]==1:
7          y_pred[i]=2
```

Συζήτηση και Συμπεράσματα

Εν κατακλείδι, στα πλαίσια μιας συνοπτική επανάληψης, αξίζει η αναφορά όλων των κύριων διαδικασιών που παρουσιάστηκαν στη διπλωματική εργασία.

Αρχικά, έγινε αναφορά στις βασικές αρχές του τομέα της Μηχανικής Μάθησης, συγκεκριμένα στους τύπους προβλημάτων που καλείται να επιλύσει μέσω εξειδικευμένων αλγορίθμων, αλλά και στη σημαντικότητα των δεδομένων και της ανάλυσή τους. Ακόμη, παρουσιάστηκε ο διαχωρισμός μεταξύ Μηχανικής και Βαθιάς Μάθησης, και δόθηκε ιδιαίτερη έμφαση στη μαθηματική επεξήγηση της εκπαιδευτικής διαδικασίας των σύγχρονων αλγορίθμων. Επίσης, παρουσιάστηκαν οι κύριες αρχιτεκτονικές Τεχνητών Νευρωνικών Δικτύων οι οποίες εφαρμόστηκαν στα δυο σύνολα δεδομένων.

Έπειτα, παρουσιάστηκε το πρώτο σύνολο δεδομένων με τις μετρήσεις ρουλεμάν της πειραματικής διάταξης, το οποίο χρησιμοποιήθηκε για την επίδειξη επίλυσης ενός προβλήματος Κατηγοριοποίησης και Ανίχνευσης Βλαβών μέσω ενός μοντέλου Πολυεπίπεδου Αντιλήπτρου. Το σύνολο αυτό αποτέλεσε γνώμονα για να επεξηγηθούν βασικές τεχνικές στατιστικής ανάλυσης, ενώ το τελικό μοντέλο Βαθιάς Μάθησης επιτεύχθηκε μέσα από τέσσερις διαφορετικές εκδοχές, έτσι ώστε να γίνει κατανοητή η σημασία των διάφορων τεχνικών που χρησιμοποιήθηκαν, αλλά και η γενικότερη διαδικασία ανάπτυξης ενός μοντέλου Μηχανικής Μάθησης. Τα τελικά αποτελέσματα παρουσιάστηκαν αναλυτικά και ήταν ιδιαίτερα ικανοποιητικά στα πλαίσια της διάγνωσης βλάβης εξαρτημάτων, ενώ ενδιαφέρον παρουσίασε και η κλιμακούμενη αύξηση της επίδοσης μέσα από τις διαφορετικές εκδοχές του μοντέλου.

Στη συνέχεια, το δεύτερο σύνολο δεδομένων του πραγματικού βιομηχανικού εξοπλισμού χρησιμοποιήθηκε για την πιο εξειδικευμένη περίπτωση ανάλυσης ενός προβλήματος Ανίχνευσης Ανωμαλιών μέσω ενός μοντέλου Αυτοκωδικοποιητή με τη χρήση νευρώνων Μακράς Βραχυπρόθεσμης Μνήμης, ακολουθώντας την προσέγγιση των Ανακατασκευών. Τα κύρια δεδομένα διαχωρίζονταν σε δυο διαφορετικές λειτουργικές καταστάσεις δυο διαφορετικών μονάδων ρουλεμάν, καθώς για κάθε μονάδα ρουλεμάν υπήρχαν μετρήσεις φυσιολογικής και επιβαρυμένης κατάστασης. Στα πλαίσια ανάπτυξης του μοντέλου Βαθιάς Μάθησης, χρησιμοποιήθηκαν οι φυσιολογικές μετρήσεις μόνο του ενός ρουλεμάν για την εκπαίδευσή του, το οποίο στη συνέχεια αξιολογήθηκε πάνω σε όλα τα διαθέσιμα δεδομένα. Το εκπαιδευμένο μοντέλο κατάφερε να αναγνωρίσει επιτυχώς φυσιολογικές και μη-φυσιολογικές λειτουργικές καταστάσεις στην παραγωγική διαδικασία, ακόμα και στα δεδομένα του ρουλεμάν στο οποίο δεν εκπαιδεύτηκε, επιβεβαιώνοντας έτσι την ικανότητά

του να προσαρμόζεται και σε δεδομένα παρόμοιων εξαρτημάτων. Τα τελικά αποτελέσματα σε κάθε περίπτωση παρουσιάστηκαν αναλυτικά και δόθηκε έμφαση στη γραφική οπτικοποίησή τους, καθώς και στη σύγκριση των διαφόρων περιπτώσεων.

Τελικά, σε εγκυκλοπαιδικά πλαίσια, έγινε μια αναφορά σε κλασσικές μεθόδους ομαδοποίησης δεδομένων και παρουσιάστηκε η υλοποίησή τους πάνω στα δεδομένα της πειραματικής διάταξης. Υπήρξαν περιπτώσεις όπου τα αποτελέσματα αυτών των μεθόδων παρουσίασαν ιδιαίτερο ενδιαφέρον, ωστόσο δεν μπόρεσαν να φτάσουν σε επιδόσεις το αντίστοιχο τελικό μοντέλο Βαθιάς Μάθησης, τουλάχιστον στη συμβατική μορφή τους.

Δεν μπορεί να αμφισβητηθεί το γεγονός ότι οι σύγχρονες βιομηχανίες προσπαθούν συνεχώς να διατηρούν σημαντική την παρουσία τους σε ένα ανταγωνιστικό περιβάλλον, και στην περίοδο της Βιομηχανίας 4.0 την οποία διανύουμε, ένας τρόπος για να επιτευχθεί αυτός ο σκοπός είναι η εκμετάλλευση των νέων τεχνολογιών και καινοτομιών που αυτή προσφέρει. Με την εισαγωγή του Βιομηχανικού Διαδικτύου των Πραγμάτων, των συστημάτων Τεχνητής Νοημοσύνης, των συνεχώς αναβαθμιζόμενων υπολογιστικών συστημάτων κ.α., είναι πλέον δυνατή η συλλογή μεγάλου όγκου δεδομένων και η ανάλυσή τους για την καλύτερη εκτίμηση της κατάστασης υγείας του βιομηχανικού εξοπλισμού, κάτι που επιβεβαιώνεται και από το αυξημένο ερευνητικό ενδιαφέρον. Από αυτές τις τεχνολογικές εξελικτικές συνθήκες προέκυψε η στρατηγική της Προβλεπτικής Συντήρησης, δίνοντας στις βιομηχανίες ακόμα έναν τρόπο για να βελτιώσουν την παραγωγική διαδικασία, αλλά και τη συνολική αποδοτικότητά τους. Με την υλοποίηση ενός προγράμματος Προβλεπτικής Συντήρησης, οι επιχειρήσεις είναι σε θέση να λαμβάνουν αποφάσεις για επιδιορθωτικές ενέργειες του εξοπλισμού την κατάλληλη στιγμή, προλαμβάνοντας έτσι τυχόν ξαφνικές διακοπές της παραγωγής, αντίθετες των προγραμματισμών τους, οι οποίες έχουν σαφώς σημαντικό οικονομικό αντίκτυπο. Περαιτέρω πλεονεκτήματα αυτής της πολιτικής είναι η μεγιστοποίηση του χρόνου ζωής των εξαρτημάτων, η ελαχιστοποίηση του χρόνου διακοπής για λόγους συντήρησης και κατ' επέκταση η μεγιστοποίηση του χρόνου αδιάλειπτης λειτουργίας, αλλά και ελαχιστοποίηση περιττών ενεργειών συντήρησης. Η αθροιστική συνεισφορά των παραπάνω αποσκοπεί στην ελαχιστοποίηση του συνολικού κόστους των βιομηχανικών και τελικά στη βιωσιμότητά τους στη σύγχρονη εποχή.

Τελικά, αυτή η διπλωματική εργασία έρχεται να επιβεβαιώσει τα πλεονεκτήματα των μεθόδων της Προβλεπτικής Συντήρησης, μέσα από τη μελέτη δεδομένων μηχανολογικών εξαρτημάτων, τη στατιστική τους ανάλυση και την ανάπτυξη μοντέλων Βαθιάς Μάθησης.

Αναφορές - Βιβλιογραφία

[1] "industrial sector | Encyclopedia.com." https://www.encyclopedia.com/social-sciences/dictionaries-thesauruses-pictures-and-press-releases/industrial-sector (accessed Aug. 21, 2023).

[2] P. P. Groumpos, "A Critical Historical and Scientific Overview of all Industrial Revolutions," *IFAC-PapersOnLine*, vol. 54, no. 13, pp. 464–471, 2021, doi: 10.1016/j.ifacol.2021.10.492.

[3] "Industrial Revolution and Technology." https://education.nationalgeographic.org/resource/industrial-revolution-and-technology (accessed Aug. 18, 2023).

[4] "Industrial Revolution - Technology, Factories, Change | Britannica Money." https://www.britannica.com/money/topic/Industrial-Revolution/The-first-Industrial-Revolution (accessed Aug. 21, 2023).

[5] "assembly line | Automation, Efficiency & Productivity Definition | Britannica Money." https://www.britannica.com/money/assembly-line (accessed Aug. 21, 2023).

[6] H. K. Mohajan, "Third Industrial Revolution Brings Global Development," vol. 7, no. 4, 2021.

[7] "Industrial Revolution: Definition, Inventions & Dates," *HISTORY*, Mar. 27, 2023. https://www.history.com/topics/industrial-revolution/industrial-revolution (accessed Aug. 21, 2023).

[8] "The Fourth Industrial Revolution | Essay by Klaus Schwab | Britannica." https://www.britannica.com/topic/The-Fourth-Industrial-Revolution-2119734 (accessed Aug. 21, 2023).

[9] C. Bai, P. Dallasega, G. Orzes, and J. Sarkis, "Industry 4.0 technologies assessment: A sustainability perspective," *International Journal of Production Economics*, vol. 229, p. 107776, Nov. 2020, doi: 10.1016/j.ijpe.2020.107776.

[10] W. S. Alaloul, M. S. Liew, N. A. W. A. Zawawi, and I. B. Kennedy, "Industrial Revolution 4.0 in the construction industry: Challenges and opportunities for stakeholders," *Ain Shams Engineering Journal*, vol. 11, no. 1, pp. 225–230, Mar. 2020, doi: 10.1016/j.asej.2019.08.010.

[11] "Free Vector | Smart industry 4.0 flat infographics representing four industrial revolutions in engineering and manufacturing vector illustration," *Freepik*. https://www.freepik.com/free-vector/smart-industry-4-0-flat-infographics-representing-four-industrial-revolutions-engineering-manufacturing-vector-illustration_40275187.htm (accessed Aug. 21, 2023).

[12] O. Serradilla, E. Zugasti, J. Rodriguez, and U. Zurutuza, "Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects," *Appl Intell*, vol. 52, no. 10, pp. 10934–10964, Aug. 2022, doi: 10.1007/s10489-021-03004-y.

[13] R. K. Mobley, *An introduction to predictive maintenance*, 2nd ed. Amsterdam ; New York: Butterworth-Heinemann, 2002.

[14] Z. M. Çınar, A. Abdussalam Nuhu, Q. Zeeshan, O. Korhan, M. Asmael, and B. Safaei, "Machine Learning in Predictive Maintenance towards Sustainable Smart Manufacturing in Industry 4.0," *Sustainability*, vol. 12, no. 19, p. 8211, Oct. 2020, doi: 10.3390/su12198211.

[15] M. Achouch *et al.*, "On Predictive Maintenance in Industry 4.0: Overview, Models, and Challenges," *Applied Sciences*, vol. 12, no. 16, p. 8081, Aug. 2022, doi: 10.3390/app12168081.

[16] J. Levitt, *Complete Guide to Preventive and Predictive Maintenance*, Second edition. New York, NY: Industrial Press Inc, 2011.

[17] D. Tran Anh, K. Dąbrowski, and K. Skrzypek, "The Predictive Maintenance Concept in the Maintenance Department of the 'Industry 4.0' Production Enterprise," *Foundations of Management*, vol. 10, no. 1, pp. 283–292, Dec. 2018, doi: 10.2478/fman-2018-0022.

[18] T. Zonta, C. A. Da Costa, R. Da Rosa Righi, M. J. De Lima, E. S. Da Trindade, and G. P. Li, "Predictive maintenance in the Industry 4.0: A systematic literature review," *Computers & Industrial Engineering*, vol. 150, p. 106889, Dec. 2020, doi: 10.1016/j.cie.2020.106889.
[19] M. Pech, J. Vrchota, and J. Bednář, "Predictive Maintenance and Intelligent Sensors in Smart Factory: Review," *Sensors*, vol. 21, no. 4, p. 1470, Feb. 2021, doi: 10.3390/s21041470.
[20] J. Lee, H.-A. Kao, and S. Yang, "Service Innovation and Smart Analytics for Industry 4.0 and Big Data Environment," *Procedia CIRP*, vol. 16, pp. 3–8, 2014, doi: 10.1016/j.procir.2014.02.001.
[21] J. Yang, S. Li, Z. Wang, H. Dong, J. Wang, and S. Tang, "Using Deep Learning to Detect Defects in Manufacturing: A Comprehensive Survey and Current Challenges," *Materials*, vol. 13, no. 24, p. 5755, Dec. 2020, doi: 10.3390/ma13245755.
[22] T. P. Carvalho, F. A. A. M. N. Soares, R. Vita, R. D. P. Francisco, J. P. Basto, and S. G. S. Alcalá, "A systematic literature review of machine learning methods applied to predictive maintenance," *Computers & Industrial Engineering*, vol. 137, p. 106024, Nov. 2019, doi: 10.1016/j.cie.2019.106024.
[23] A. Barr and E. A. Feigenbaum, Eds., "Chapter I - Introduction," in *The Handbook of Artificial Intelligence*, Butterworth-Heinemann, 1981, pp. 1–17. doi: 10.1016/B978-0-86576-089-9.50006-5.
[24] "152 Fun AI Tools You've Never Heard of (Fresh Update!)." https://www.synthesia.io/post/ai-tools (accessed Aug. 21, 2023).
[25] "A Brief History of AI." https://aitopics.org/misc/brief-history (accessed Aug. 22, 2023).
[26] "Antiquity," *A History of Artificial Intelligence*. https://ahistoryofai.com/antiquity/ (accessed Aug. 21, 2023).
[27] "Heron of Alexandria | Ancient Greek Engineer & Mathematician | Britannica," Jul. 31, 2023. https://www.britannica.com/biography/Heron-of-Alexandria (accessed Aug. 21, 2023).
[28] "Ismail al-Jazari, the Muslim inventor whom some call the 'Father of Robotics,'" *History*, Jul. 30, 2020. https://www.nationalgeographic.com/history/history-magazine/article/ismail-al-jazari-muslim-inventor-called-father-robotics (accessed Aug. 21, 2023).
[29] R. Guinness, "What is Artificial Intelligence? Part 1," *Medium*, Mar. 21, 2018. https://towardsdatascience.com/what-is-artificial-intelligence-part-1-75a6de110141 (accessed Aug. 22, 2023).
[30] G. J. Massey, D. A. Boyle, and University of Arkansas Press, "Descartes's Tests for (Animal) Mind:," *Philosophical Topics*, vol. 27, no. 1, pp. 87–146, 1999, doi: 10.5840/philtopics199927119.
[31] "Jacques de Vaucanson | Biography, Automata, Inventions, & Facts | Britannica." https://www.britannica.com/biography/Jacques-de-Vaucanson (accessed Aug. 22, 2023).
[32] "The Mechanical Turk | The Engines of Our Ingenuity." https://engines.egr.uh.edu/episode/2765 (accessed Aug. 22, 2023).
[33] B. G. Buchanan, "A (Very) Brief History of Artificial Intelligence".
[34] "Jacquard loom | Definition, HIstory, Computer, & Facts | Britannica," Jul. 25, 2023. https://www.britannica.com/technology/Jacquard-loom (accessed Aug. 22, 2023).
[35] "The story of the Jacquard loom | Science and Industry Museum." https://www.scienceandindustrymuseum.org.uk/objects-and-stories/jacquard-loom (accessed Aug. 22, 2023).
[36] "Charles Babbage | Biography, Computers, Inventions, & Facts | Britannica," Aug. 13, 2023. https://www.britannica.com/biography/Charles-Babbage (accessed Aug. 22, 2023).
[37] "Difference Engine | Calculating Machine, Charles Babbage, 19th Century | Britannica." https://www.britannica.com/technology/Difference-Engine (accessed Aug. 22, 2023).
[38] "Analytical Engine | Description & Facts | Britannica." https://www.britannica.com/technology/Analytical-Engine (accessed Aug. 22, 2023).

[39] "Sketch of The Analytical Engine." https://www.fourmilab.ch/babbage/sketch.html (accessed Aug. 22, 2023).
[40] "Ada Lovelace | Biography, Computer, & Facts | Britannica," Aug. 14, 2023. https://www.britannica.com/biography/Ada-Lovelace (accessed Aug. 22, 2023).
[41] B. Linsky and A. D. Irvine, "Principia Mathematica," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Spring 2022.Metaphysics Research Lab, Stanford University, 2022. Accessed: Aug. 22, 2023. [Online]. Available: https://plato.stanford.edu/archives/spr2022/entries/principia-mathematica/
[42] "Turing_Paper_1936.pdf." Accessed: Aug. 22, 2023. [Online]. Available: https://www.cs.virginia.edu/~robins/Turing_Paper_1936.pdf
[43] "Turing machine | Definition & Facts | Britannica," Jun. 30, 2023. https://www.britannica.com/technology/Turing-machine (accessed Aug. 22, 2023).
[44] L. De Mol, "Turing Machines," in *The Stanford Encyclopedia of Philosophy*, E. N. Zalta, Ed., Winter 2021.Metaphysics Research Lab, Stanford University, 2021. Accessed: Aug. 22, 2023. [Online]. Available: https://plato.stanford.edu/archives/win2021/entriesuring-machine/
[45] "McCulloch-Pitts Neurons." https://mind.ilstu.edu/curriculum/mcp_neurons/index.html (accessed Aug. 22, 2023).
[46] "Neural network | Computing & Machine Learning | Britannica," Jul. 19, 2023. https://www.britannica.com/technology/neural-network (accessed Aug. 22, 2023).
[47] "Claude E. Shannon | IEEE Information Theory Society." https://www.itsoc.org/about/shannon (accessed Aug. 22, 2023).
[48] "A Brief History of Artificial Intelligence." https://www.atariarchives.org/deli/artificial_intelligence.php (accessed Aug. 22, 2023).
[49] A. M. TURING, "I.—COMPUTING MACHINERY AND INTELLIGENCE," *Mind*, vol. LIX, no. 236, pp. 433–460, Oct. 1950, doi: 10.1093/mind/LIX.236.433.
[50] R. Guinness, "What is Artificial Intelligence? Part 2," *Medium*, May 09, 2019. https://towardsdatascience.com/what-is-artificial-intelligence-part-2-bad0cb97e330 (accessed Aug. 22, 2023).
[51] SITNFlash, "The History of Artificial Intelligence," *Science in the News*, Aug. 28, 2017. https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/ (accessed Aug. 21, 2023).
[52] "Artificial intelligence (AI) | Definition, Examples, Types, Applications, Companies, & Facts | Britannica," Aug. 21, 2023. https://www.britannica.com/technology/artificial-intelligence (accessed Aug. 22, 2023).
[53] Maad M. Mijwel, "History of Artificial Intelligence," 2015, doi: 10.13140/RG.2.2.16418.15046.
[54] "Logic Theorist," *A History of Artificial Intelligence*. https://ahistoryofai.com/logic-theorist/ (accessed Aug. 22, 2023).
[55] "A PROPOSAL FOR THE DARTMOUTH SUMMER RESEARCH PROJECT ON ARTIFICIAL INTELLIGENCE," Sep. 30, 2008. https://web.archive.org/web/20080930164306/http://www-formal.stanford.edu/jmc/history/dartmouth/dartmouth.html (accessed Aug. 22, 2023).
[56] "LISP | Artificial Intelligence, Machine Learning & Programming | Britannica," Jul. 11, 2023. https://www.britannica.com/technology/LISP-computer-language (accessed Aug. 22, 2023).
[57] "PARRY encounters the DOCTOR," Internet Engineering Task Force, Request for Comments RFC 439, Jan. 1973. doi: 10.17487/RFC0439.
[58] "Humanoid History -WABOT-." https://www.humanoid.waseda.ac.jp/booklet/kato_2.html (accessed Aug. 22, 2023).
[59] "Project MAC | MIT, Artificial Intelligence, Computer Science | Britannica," Jul. 08, 2023. https://www.britannica.com/topic/Project-Mac (accessed Aug. 22, 2023).
[60] O. Kaynak, "The golden age of Artificial Intelligence," *Discov Artif Intell*, vol. 1, no. 1, p. 1, Sep. 2021, doi: 10.1007/s44163-021-00009-x.
[61] "The History of Artificial Intelligence".

[62] "Timeline of artificial intelligence - Timelines." https://timelines.issarice.com/wiki/Timeline_of_artificial_intelligence (accessed Aug. 22, 2023).
[63] M. Haenlein and A. Kaplan, "A Brief History of Artificial Intelligence: On the Past, Present, and Future of Artificial Intelligence," *California Management Review*, vol. 61, no. 4, pp. 5–14, Aug. 2019, doi: 10.1177/0008125619864925.
[64] Z. Saleh, "Artificial Intelligence Definition, Ethics and Standards," Apr. 2019.
[65] V. Jokanović, "Artificial Intelligence," 2022.
[66] E. Escott, "What are the 3 types of AI? A guide to narrow, general, and super artificial intelligence," *Codebots*, Oct. 24, 2017. https://codebots.com/artificial-intelligence/the-3-types-of-ai-is-the-third-even-possible (accessed Aug. 22, 2023).
[67] H. Yamaura, J. Igarashi, and T. Yamazaki, "Simulation of a Human-Scale Cerebellar Network Model on the K Computer," *Frontiers in Neuroinformatics*, vol. 14, 2020, Accessed: Aug. 22, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fninf.2020.00016
[68] *Stephen Hawking: "AI could spell end of the human race,"* (Dec. 02, 2014). Accessed: Aug. 22, 2023. [Online Video]. Available: https://www.youtube.com/watch?v=fFLVyWBDTfo
[69] "4 Types of AI: Getting to Know Artificial Intelligence," *Coursera*, Jun. 16, 2023. https://www.coursera.org/articles/types-of-ai (accessed Aug. 22, 2023).
[70] A. Hintze, "Understanding the four types of AI, from reactive robots to self-aware beings," *The Conversation*, Nov. 14, 2016. http://theconversation.com/understanding-the-four-types-of-ai-from-reactive-robots-to-self-aware-beings-67616 (accessed Aug. 22, 2023).
[71] F. Weber, *Artificial Intelligence for Business Analytics: Algorithms, Platforms and Application Scenarios*. Wiesbaden: Springer Fachmedien Wiesbaden, 2023. doi: 10.1007/978-3-658-37599-7.
[72] "AIMA.pdf." Accessed: Aug. 22, 2023. [Online]. Available: https://web.cs.ucla.edu/~srinath/static/pdfs/AIMA.pdf
[73] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[74] F. Chollet, *Deep learning with Python*, Second edition. Shelter Island: Manning Publications, 2021.
[75] A. Geron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*, 2nd ed. O'Reilly Media, Inc., 2019.
[76] O. Theobald, "Machine Learning For Absolute Beginners".
[77] F. Lazzeri, *Machine Learning for Time Series Forecasting with Python®*, 1st ed. Wiley, 2020. doi: 10.1002/9781119682394.
[78] C. Ranjan, *Understanding Deep Learning: Application in Rare Event Prediction*. Connaissance Publishing, 2020. doi: 10.13140/RG.2.2.34297.49765.
[79] R. Karim, "Animated RNN, LSTM and GRU," *Medium*, Jul. 04, 2020. https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45 (accessed Aug. 23, 2023).
[80] "pandas documentation — pandas 2.0.3 documentation." https://pandas.pydata.org/docs/ (accessed Aug. 23, 2023).
[81] "NumPy documentation — NumPy v1.25 Manual." https://numpy.org/doc/stable/index.html (accessed Aug. 23, 2023).
[82] "Matplotlib documentation — Matplotlib 3.7.2 documentation." https://matplotlib.org/stable/index.html (accessed Aug. 23, 2023).
[83] "scikit-learn: machine learning in Python — scikit-learn 1.3.0 documentation." https://scikit-learn.org/stable/index.html (accessed Aug. 23, 2023).
[84] "Module: tf | TensorFlow v2.13.0," *TensorFlow*. https://www.tensorflow.org/api_docs/python/tf (accessed Aug. 23, 2023).
[85] K. Team, "Keras documentation: Keras API reference." https://keras.io/api/ (accessed Aug. 23, 2023).

[86] "Anaconda | The World's Most Popular Data Science Platform," *Anaconda*. https://www.anaconda.com/ (accessed Aug. 23, 2023).
[87] "Google Colaboratory." https://colab.research.google.com/ (accessed Aug. 23, 2023).
[88] E. Karapalidou, A. Efraimidis, S. Vologiannidis, and E. Antoniou, "Vibration analysis metrics of a ball bearing during different operational states." Zenodo, Jan. 31, 2023. doi: 10.5281/zenodo.7589904.
[89] P. G. B. E. (MechEng) Cornelius Scheffer Ph.D MEng, *Practical machinery vibration analysis and predictive maintenance*. Elsevier; Newnes.
[90] E. Karapalidou, N. Alexandris, E. Antoniou, S. Vologiannidis, J. Kalomiros, and D. Varsamis, "V-RMS and Temperature metrics of the ball bearing units of an industrial blower." Zenodo, Jun. 01, 2023. doi: 10.5281/zenodo.7994124.

Παράρτημα Α’

Στοιχεία πίνακα δεδομένων state_2, ο οποίος περιλαμβάνει μόνο τις μετρήσεις που καταγράφηκαν κατά τη λειτουργία του ρουλεμάν με μηδενική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  state_2

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
3296	0.000695	0.609231	3.911539	6.477692	25.938461	2
3297	0.000704	0.600769	3.803846	6.330769	25.933077	2
3298	0.000703	0.607692	3.946923	6.457692	25.959231	2
3299	0.000705	0.617692	3.942308	6.349231	25.900000	2
3300	0.000699	0.602308	3.706154	6.118462	25.900000	2
...	...	...	...	...	...	...
5927	0.000720	0.641538	4.846923	7.443077	23.900000	2
5928	0.000759	0.650769	5.014615	7.589231	23.900000	2
5929	0.000761	0.666923	5.040000	7.503846	23.900000	2
5930	0.000735	0.670000	5.122308	7.546923	23.900000	2
5931	0.000737	0.646154	5.036154	7.666154	23.900000	2

2636 rows × 6 columns

Στοιχεία πίνακα δεδομένων state_3, ο οποίος περιλαμβάνει μόνο τις μετρήσεις που καταγράφηκαν κατά τη λειτουργία του ρουλεμάν με υπερβολική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  state_3

Output []:
```

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
5932	0.000669	0.762308	5.940769	7.719231	26.700001	3
5933	0.000694	0.811538	6.573846	8.096154	26.700001	3
5934	0.000700	0.803077	6.410000	7.982308	26.715385	3
5935	0.000704	0.790769	6.002308	7.600769	26.799999	3
5936	0.000718	0.806923	6.087692	7.539231	26.799999	3
...	...	...	...	...	...	...
8564	0.000728	0.880769	7.337692	8.304615	26.866153	3
8565	0.000703	0.853846	6.926154	8.102308	26.806153	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3
8568	0.000708	0.866154	7.378462	8.506923	26.799999	3

2637 rows × 6 columns

Παράρτημα Β’

Μέτρα θέσης και διασποράς για τον πίνακα δεδομένων data, ο οποίος περιλαμβάνει όλες τις καταστάσεις λειτουργίας του ρουλεμάν ως προς τις συνθήκες λίπανσης, επομένως δεν είναι ιδιαίτερα αντιπροσωπευτικός ως προς τα αποτελέσματα που εξάγει:

```
Input []:
1  data.describe()
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
count	8569.000000	8569.000000	8569.000000	8569.000000	8569.000000	8569.000000
mean	0.000852	0.718060	4.588954	6.308177	26.413778	1.923095
std	0.000159	0.077410	1.456797	1.417231	0.747099	0.828580
min	0.000635	0.523846	2.996923	4.438462	23.700001	1.000000
25%	0.000721	0.670714	3.233571	4.789231	25.901538	1.000000
50%	0.000759	0.700000	4.204615	6.393077	26.500000	2.000000
75%	0.001022	0.779231	5.736923	7.533846	27.000000	3.000000
max	0.001193	0.963077	8.972308	9.825385	28.000000	3.000000

Μέτρα θέσης και διασποράς για τον πίνακα δεδομένων state_2, ο οποίος περιλαμβάνει μόνο τις μετρήσεις που καταγράφηκαν κατά τη λειτουργία του ρουλεμάν με μηδενική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
2  state_2.describe()
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
count	2636.000000	2636.000000	2636.000000	2636.000000	2636.000000	2636.0
mean	0.000732	0.660787	4.498802	6.741758	26.522649	2.0
std	0.000030	0.053565	0.668678	0.653450	0.689415	0.0
min	0.000642	0.523846	3.066923	4.799231	23.700001	2.0
25%	0.000708	0.622308	3.997692	6.254533	26.200001	2.0
50%	0.000733	0.656154	4.448462	6.665000	26.700001	2.0
75%	0.000755	0.695385	4.958462	7.231731	27.000000	2.0
max	0.000834	0.914615	7.849231	8.588462	27.500000	2.0

Μέτρα θέσης και διασποράς για τον πίνακα δεδομένων state_3, ο οποίος περιλαμβάνει μόνο τις μετρήσεις που καταγράφηκαν κατά τη λειτουργία του ρουλεμάν με υπερβολική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  state_3.describe()
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
count	2637.000000	2637.000000	2637.000000	2637.000000	2637.000000	2637.0
mean	0.000727	0.814529	6.399069	7.814599	26.753765	3.0
std	0.000032	0.044970	0.890498	0.788849	0.465074	0.0
min	0.000635	0.615385	3.216154	4.873077	25.700001	3.0
25%	0.000705	0.785714	5.845385	7.336154	26.400000	3.0
50%	0.000725	0.816923	6.457143	7.911538	26.700001	3.0
75%	0.000748	0.845385	7.020000	8.364615	27.000000	3.0
max	0.000842	0.963077	8.972308	9.825385	28.000000	3.0

Μέτρα θέσης και διασποράς για το μέγεθος a-RMS ως προς κάθε κατάσταση λειτουργίας του ρουλεμάν (κανονική, έλλειψη, περίσσεια):

```
Input []:
1  data_grouped['a-RMS'].describe()
```

Output []:

Bearing State	count	mean	std	min	25%	50%	75%	max
1	3296.0	0.686683	0.024753	0.600000	0.684615	0.698462	0.700000	0.799231
2	2636.0	0.660787	0.053565	0.523846	0.622308	0.656154	0.695385	0.914615
3	2637.0	0.814529	0.044970	0.615385	0.785714	0.816923	0.845385	0.963077

Μέτρα θέσης και διασποράς για το μέγεθος a-Peak ως προς κάθε κατάσταση λειτουργίας του ρουλεμάν (κανονική, έλλειψη, περίσσεια):

```
Input []:
1  data_grouped['a-Peak'].describe()
```

Output []:

Bearing State	count	mean	std	min	25%	50%	75%	max
1	3296.0	3.212853	0.060114	2.996923	3.171264	3.210769	3.253077	3.477692
2	2636.0	4.498802	0.668678	3.066923	3.997692	4.448462	4.958462	7.849231
3	2637.0	6.399069	0.890498	3.216154	5.845385	6.457143	7.020000	8.972308

Μέτρα θέσης και διασποράς για το μέγεθος Crest ως προς κάθε κατάσταση λειτουργίας του ρουλεμάν (κανονική, έλλειψη, περίσσεια):

Input []:

```
1  data_grouped['Crest'].describe()
```

Output []:

Bearing State	count	mean	std	min	25%	50%	75%	max
1	3296.0	4.756187	0.087172	4.438462	4.697692	4.755934	4.810769	5.183846
2	2636.0	6.741758	0.653450	4.799231	6.254533	6.665000	7.231731	8.588462
3	2637.0	7.814599	0.788849	4.873077	7.336154	7.911538	8.364615	9.825385

Μέτρα θέσης και διασποράς για το μέγεθος Temperature ως προς κάθε κατάσταση λειτουργίας του ρουλεμάν (κανονική, έλλειψη, περίσσεια):

Input []:

```
1  data_grouped['Temperature'].describe()
```

Output []:

Bearing State	count	mean	std	min	25%	50%	75%	max
1	3296.0	26.054698	0.816874	24.230770	25.496731	25.857692	26.777692	27.799999
2	2636.0	26.522649	0.689415	23.700001	26.200001	26.700001	27.000000	27.500000
3	2637.0	26.753765	0.465074	25.700001	26.400000	26.700001	27.000000	28.000000

Παράρτημα Γ’

Παράδειγμα ομαδοποίησης στοιχείων του πίνακα δεδομένων data, τροποποίησης των επιμέρους πινάκων δεδομένων state_1, state_2 και state_3, και συσσωμάτωσης των νέων στοιχείων:

```
Input []:
1  # inspect data
2  print(data)
3
4  # group by 'Bearing State'
5  data_grouped = data.groupby('Bearing State')
6  state_1 = data_grouped.get_group(1)
7  state_2 = data_grouped.get_group(2)
8  state_3 = data_grouped.get_group(3)
9
10 # assign and edit new DataFrames
11 s1 = state_1[(state_1['Temperature']>26.5) &
                (state_1['Temperature']<27)].copy()
12 s2 = state_2[(state_2['Temperature']>26.5) &
                (state_2['Temperature']<27)].copy()
13 s3 = state_3[(state_3['Temperature']>26.5) &
                (state_3['Temperature']<27)].copy()
14 s3.drop(index=s3[s3['Crest']>8].index, inplace=True)
15
16 # concatenate edited datasets
17 new_data = pd.concat([s1,s2,s3], axis='index', join='outer',
                       ignore_index=True)
18
19 # inspect new data
20 print(new_data)

Output []:
         V-RMS     a-RMS    a-Peak     Crest  Temperature  Bearing State
0     0.001074  0.695385  3.198462  4.740000    26.308461              1
1     0.001095  0.713077  3.244615  4.716923    26.299999              1
2     0.001079  0.697143  3.227857  4.795000    26.299999              1
3     0.001080  0.700000  3.213077  4.793077    26.208462              1
4     0.001088  0.700000  3.241538  4.783077    26.200001              1
...        ...       ...       ...       ...          ...            ...
8564  0.000728  0.880769  7.337692  8.304615    26.866153              3
8565  0.000703  0.853846  6.926154  8.102308    26.806153              3
8566  0.000705  0.834615  6.556923  7.822308    26.892307              3
8567  0.000713  0.862308  6.893846  7.989231    26.799999              3
8568  0.000708  0.866154  7.378462  8.506923    26.799999              3
```

```
[8569 rows x 6 columns]
         V-RMS     a-RMS    a-Peak     Crest  Temperature  Bearing State
0     0.001098  0.698462  3.296154  4.743077    26.588462              1
1     0.001088  0.700000  3.278462  4.761538    26.600000              1
2     0.001098  0.700000  3.285385  4.726923    26.600000              1
3     0.001090  0.700000  3.268462  4.745385    26.566154              1
4     0.001088  0.699231  3.230769  4.694615    26.531539              1
...        ...       ...       ...       ...          ...            ...
1476  0.000715  0.842857  6.382857  7.541429    26.799999              3
1477  0.000733  0.876923  6.996154  7.993846    26.799999              3
1478  0.000726  0.855385  6.603077  7.735385    26.845384              3
1479  0.000705  0.834615  6.556923  7.822308    26.892307              3
1480  0.000713  0.862308  6.893846  7.989231    26.799999              3

[1481 rows x 6 columns]
```

Να σημειωθεί ότι, οι παραπάνω εντολές μπορούν να εκτελεστούν μεμονωμένα και με συχνές κλήσεις των στοιχείων που επεξεργάζονται και των μεθόδων που υλοποιούνται για καλύτερη εποπτεία των τροποποιήσεων. Επίσης, συνιστάται, ειδικά όταν πρόκειται για επιμέρους κομμάτια ενός μεγαλύτερου συνόλου δεδομένων, οι τροποποιήσεις να γίνονται σε αντίγραφα των δεδομένων, για τον λόγο αυτό χρησιμοποιείται η συνάρτηση copy() για την αρχικοποίηση των νέων πινάκων s1, s2 και s3. Στο παραπάνω παράδειγμα παρουσιάζεται η επιλογή στοιχείων μέσω φιλτραρίσματος βάσει της θερμοκρασίας, αλλά και η επιλογή στοιχείων, ειδικά για τον πίνακα s3, μέσω της διαγραφής. Επιπλέον, τα εξαγόμενα αποτελέσματα είναι στοιχεία τύπου DataFrame, αλλά επειδή προέρχονται από εκτέλεση της εντολής print() παρουσιάζονται με διαφορετικό τρόπο.

Παράρτημα Δ'

Γράφημα γραμμών Εικόνας 51:

```
Input []:
1  plt.style.use('ggplot')
2
3  fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(
4                                    nrows=5, ncols=1,
5                                    sharex=True, sharey=False,
6                                    figsize=(10, 10))
7
8  ax1.plot(state_1['V-RMS'], label='Κανονική',
9          linewidth=0.25, color='red')
10 ax1.plot(state_2['V-RMS'], label='Έλλειψη',
11         linewidth=0.25, color='green')
12 ax1.plot(state_3['V-RMS'],   linewidth=0.25, color='blue')
13 ax1.legend(loc='upper right', fontsize='12',  handlelength=4)
14 ax1.set_ylabel('V-RMS (m/s)', fontsize='12', color='black')
15 ax1.set_title('Μετρήσεις κατά τη Λειτουργία του Ρουλεμάν
16               \nστις Διάφορες Περιπτώσεις Λιπαντικού',
17               fontsize='20')
18 leg = ax1.legend(loc='upper right', fontsize='12',  handlelength=4)
19 for line in leg.get_lines():
20         line.set_linewidth(1.5)
21
22 ax2.plot(state_1['a-RMS'], linewidth=0.25, color='red')
23 ax2.plot(state_2['a-RMS'], linewidth=0.25, color='green')
24 ax2.plot(state_3['a-RMS'], linewidth=0.25, color='blue')
25 ax2.set_ylabel('a-RMS (m/s²)', fontsize='12', color='black')
26
27 ax3.plot(state_1['a-Peak'], linewidth=0.25, color='red')
28 ax3.plot(state_2['a-Peak'], linewidth=0.25, color='green')
29 ax3.plot(state_3['a-Peak'], linewidth=0.25, color='blue')
30 ax3.set_ylabel('a-Peak (m/s²)', fontsize='12', color='black')
31
32 ax4.plot(state_1['Crest'], linewidth=0.25, color='red')
33 ax4.plot(state_2['Crest'], linewidth=0.25, color='green')
34 ax4.plot(state_3['Crest'], linewidth=0.25, color='blue')
35 ax4.set_ylabel('Crest', fontsize='12', color='black')
36
37 ax5.plot(state_1['Temperature'], linewidth=0.25, color='red')
38 ax5.plot(state_2['Temperature'], linewidth=0.25, color='green')
39 ax5.plot(state_3['Temperature'], linewidth=0.25, color='blue')
40 ax5.set_ylabel('Temperature (°C)', fontsize='12', color='black')
41 ax5.set_xlabel('Στοιχεία Πίνακα Δεδομένων',
```

```
42                fontsize='15', color='black')
43
44 plt.tight_layout()
45
46 plt.show()
```

Γράφημα Διασποράς Εικόνας 52:

```
Input []:
1  plt.style.use('ggplot')
2
3  fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(
4                                    nrows=5, ncols=1,
5                                    sharex=True, sharey=False,
6                                    figsize=(10, 10))
7
8  ax1.scatter(state_1.index, state_1['V-RMS'],
9              label='Κανονική', marker='.', s=1, color='red')
10 ax1.scatter(state_2.index, state_2['V-RMS'],
11             label='Έλλειψη', marker='.', s=1, color='green')
12 ax1.scatter(state_3.index, state_3['V-RMS'],
13             label='Περίσσεια', marker='.', s=1, color='blue')
14 ax1.legend(loc='upper right', fontsize='12',  markerscale=10)
15 ax1.set_ylabel('V-RMS (m/s)', fontsize='12', color='black')
16 ax1.set_title('Μετρήσεις κατά τη Λειτουργία του Ρουλεμάν
17               \nστις Διάφορες Περιπτώσεις Λιπαντικού',
18               fontsize='20')
19
20 ax2.scatter(state_1.index, state_1['a-RMS'],
21             marker='.', s=1, color='red')
22 ax2.scatter(state_2.index, state_2['a-RMS'],
23             marker='.', s=1, color='green')
24 ax2.scatter(state_3.index, state_3['a-RMS'],
25             marker='.', s=1, color='blue')
26 ax2.set_ylabel('a-RMS (m/s²)', fontsize='12', color='black')
27
28 ax3.scatter(state_1.index, state_1['a-Peak'],
29             marker='.', s=1, color='red')
30 ax3.scatter(state_2.index, state_2['a-Peak'],
31             marker='.', s=1, color='green')
32 ax3.scatter(state_3.index, state_3['a-Peak'],
33             marker='.', s=1, color='blue')
34 ax3.set_ylabel('a-Peak (m/s²)', fontsize='12', color='black')
35
36 ax4.scatter(state_1.index, state_1['Crest'],
37             marker='.', s=1, color='red')
```

```
ax4.scatter(state_2.index, state_2['Crest'],
            marker='.', s=1, color='green')
ax4.scatter(state_3.index, state_3['Crest'],
            marker='.', s=1, color='blue')
ax4.set_ylabel('Crest', fontsize='12', color='black')

ax5.scatter(state_1.index, state_1['Temperature'],
            marker='.', s=1, color='red')
ax5.scatter(state_2.index, state_2['Temperature'],
            marker='.', s=1, color='green')
ax5.scatter(state_3.index, state_3['Temperature'],
            marker='.', s=1, color='blue')
ax5.set_ylabel('Temperature (°C)', fontsize='12', color='black')
ax5.set_xlabel('Στοιχεία Πίνακα Δεδομένων',
               fontsize='15', color='black')

plt.tight_layout()

plt.show()
```

Γράφημα Διασποράς Εικόνας 53:

```
Input []:
plt.style.use('ggplot')
plt.figure(figsize=(10, 5))
# plt.scatter(state_1['V-RMS'], state_1['a-Peak'],
#             marker='.', s=15, label='s1 v-t')
plt.scatter(state_2['V-RMS'], state_2['a-Peak'],
            marker='.', s=15, label='Έλλειψη')
plt.scatter(state_3['V-RMS'], state_3['a-Peak'],
            marker='.', s=15, label='Περίσσεια')
plt.tight_layout()
plt.title('Διάγραμμα Συσχέτισης των V-RMS και a-Peak
          \ηγια Σύγκριση μεταξύ Καταστάσεων του Ρουλεμάν',
          fontsize='20')
plt.ylabel('a-Peak', fontsize='15', color='black')
plt.xlabel('V-RMS', fontsize='15', color='black')
plt.legend(loc='upper right', fontsize='15',  markerscale=4)
plt.show()
```

Γράφημα θηκογράμματος Εικόνας 54:

```
Input []:
plt.style.use('ggplot')

box_dict_1 = {
```

```
4  'Κανονική' : state_1['V-RMS'],
5  'Έλλειψη' : state_2['V-RMS'],
6  'Περίσσεια' : state_3['V-RMS']
7  }
8  box_dict_2 = {
9  'Κανονική' : state_1['a-RMS'],
10 'Έλλειψη' : state_2['a-RMS'],
11 'Περίσσεια' : state_3['a-RMS']
12 }
13 box_dict_3 = {
14 'Κανονική' : state_1['a-Peak'],
15 'Έλλειψη' : state_2['a-Peak'],
16 'Περίσσεια' : state_3['a-Peak']
17 }
18 box_dict_4 = {
19 'Κανονική' : state_1['Crest'],
20 'Έλλειψη' : state_2['Crest'],
21 'Περίσσεια' : state_3['Crest']
22 }
23 box_dict_5 = {
24 'Κανονική' : state_1['Temperature'],
25 'Έλλειψη' : state_2['Temperature'],
26 'Περίσσεια' : state_3['Temperature']
27 }
28
29 fig, (ax1, ax2, ax3, ax4, ax5) = plt.subplots(
30                                    nrows=5, ncols=1,
31                                    sharex=True, sharey=False,
32                                    figsize=(10, 10))
33
34 ax1.boxplot(box_dict_1.values())
35 ax1.set_ylabel('V-RMS (m/s)', fontsize='12', color='black')
36 ax1.set_title('Μετρήσεις κατά τη Λειτουργία του Ρουλεμάν
37               \nστις Διάφορες Περιπτώσεις Λιπαντικού',
38               fontsize='20')
39
40 ax2.boxplot(box_dict_2.values())
41 ax2.set_ylabel('a-RMS (m/s²)', fontsize='12', color='black')
42
43 ax3.boxplot(box_dict_3.values())
44 ax3.set_ylabel('a-Peak (m/s²)', fontsize='12', color='black')
45
46 ax4.boxplot(box_dict_4.values())
47 ax4.set_ylabel('Crest', fontsize='12', color='black')
48
49 ax5.boxplot(box_dict_5.values())
```

```
50 ax5.set_ylabel('Temperature (°C)', fontsize='12', color='black')
51 ax5.set_xticks([1, 2, 3], box_dict_1.keys(), fontsize='15')
52 ax5.set_xlabel('Κατάσταση Λίπανσης Ρουλεμάν',
53                fontsize='15', color='black')
54
55 plt.tight_layout()
56
57 plt.show()
```

Παράρτημα Ε’

Πίνακας συντελεστών Πίρσον για όλα τα ζεύγη στηλών της λειτουργικής κατάστασης του ρουλεμάν με μηδενική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  state_2_corr = state_2.corr(method='pearson')
2  state_2_corr
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
V-RMS	1.000000	0.044496	0.285397	0.370879	-0.207972	NaN
a-RMS	0.044496	1.000000	0.767004	0.346754	-0.036521	NaN
a-Peak	0.285397	0.767004	1.000000	0.864846	-0.255512	NaN
Crest	0.370879	0.346754	0.864846	1.000000	-0.344040	NaN
Temperature	-0.207972	-0.036521	-0.255512	-0.344040	1.000000	NaN
Bearing State	NaN	NaN	NaN	NaN	NaN	NaN

Πίνακας συντελεστών Πίρσον για όλα τα ζεύγη στηλών της λειτουργικής κατάστασης του ρουλεμάν με υπερβολική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  state_3_corr = state_3.corr(method='pearson')
2  state_3_corr
```

Output []:

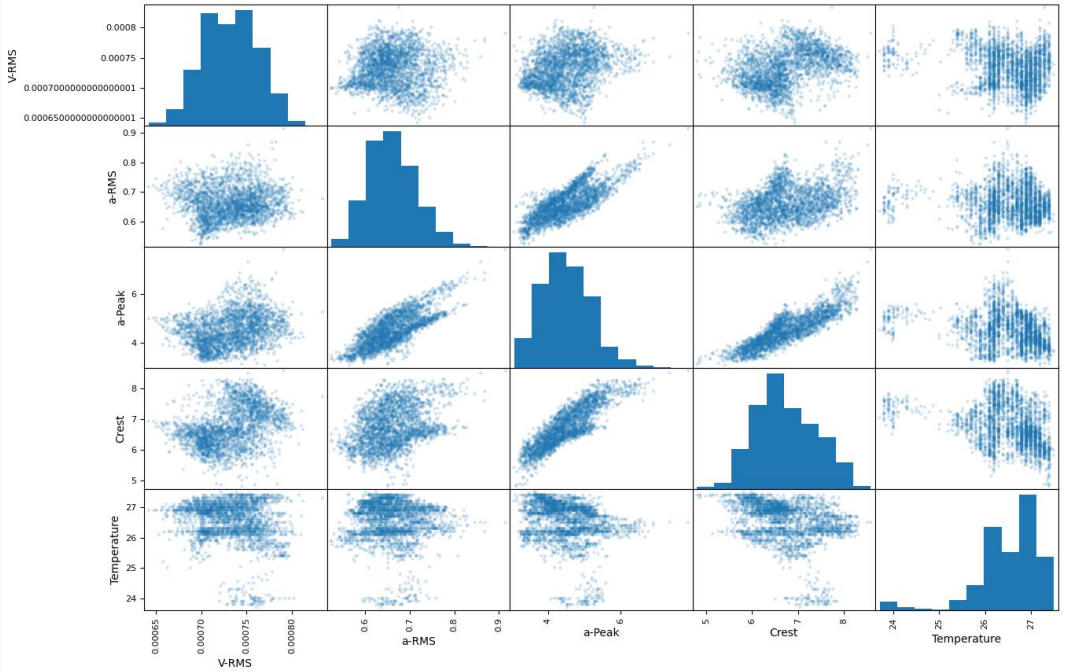
	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
V-RMS	1.000000	0.001067	-0.342288	-0.484516	0.061044	NaN
a-RMS	0.001067	1.000000	0.823762	0.615943	0.062433	NaN
a-Peak	-0.342288	0.823762	1.000000	0.951584	-0.021342	NaN
Crest	-0.484516	0.615943	0.951584	1.000000	-0.053441	NaN
Temperature	0.061044	0.062433	-0.021342	-0.053441	1.000000	NaN
Bearing State	NaN	NaN	NaN	NaN	NaN	NaN

Διαγράμματα διασποράς για όλα τα ζεύγη στηλών της λειτουργικής κατάστασης του ρουλεμάν με μηδενική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1  plt.style.use('default')
2  pd.plotting.scatter_matrix(state_2[['V-RMS', 'a-RMS', 'a-Peak',
3                                      'Crest', 'Temperature']],
```

```
4                                  figsize=(15, 10), alpha=0.2,
5                                  diagonal='hist')
6   plt.show()
```

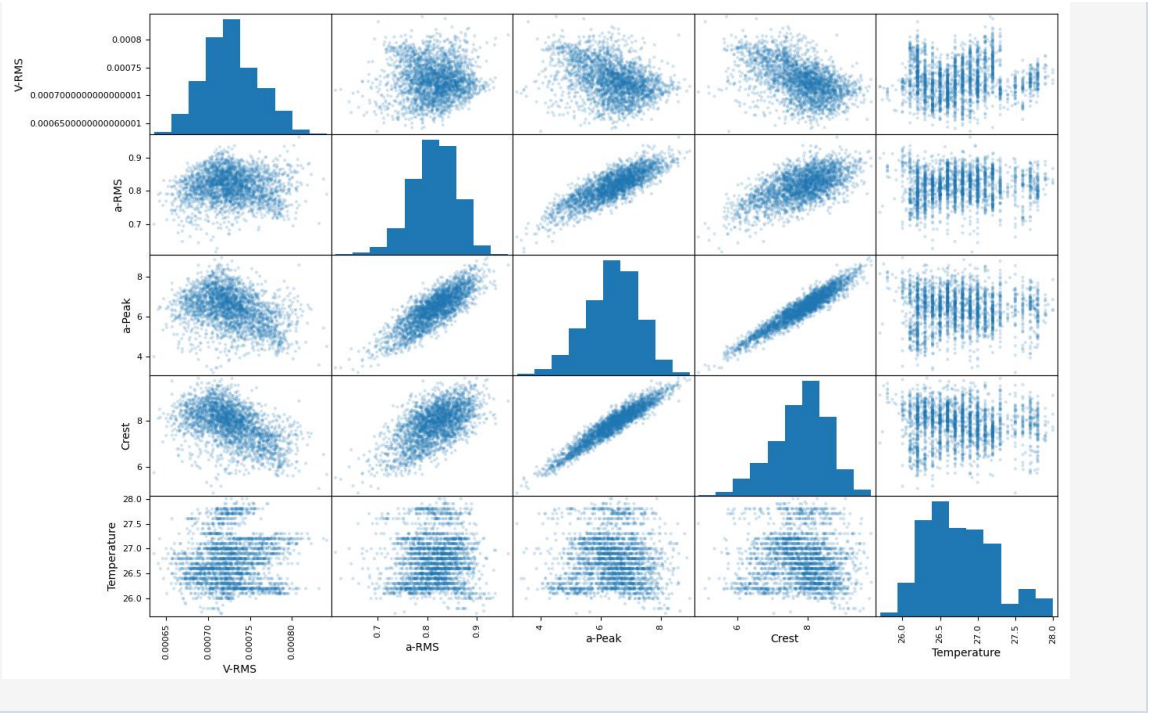
Output []:



Διαγράμματα διασποράς για όλα τα ζεύγη στηλών της λειτουργικής κατάστασης του ρουλεμάν με υπερβολική ποσότητα εργοστασιακού λιπαντικού:

```
Input []:
1   plt.style.use('default')
2   pd.plotting.scatter_matrix(state_3[['V-RMS', 'a-RMS', 'a-Peak',
3                                       'Crest', 'Temperature']],
4                                  figsize=(15, 10), alpha=0.2,
5                                  diagonal='hist')
6   plt.show()
```

Output []:


V-RMS
a-RMS
a-Peak
Crest
Temperature

Παράρτημα Ζ’

Τα σύνολα δεδομένων X_test και y_test μετά τη διαίρεση του αρχικού συνόλου δεδομένων data στα επιμέρους σύνολα Εκπαίδευσης και Αξιολόγησης, όπου το πρώτο εμφανίζεται σε μορφή DataFrame και το δεύτερο σε μορφή Series της βιβλιοθήκης Pandas:

```
Input []:
1  print(X_test)
2  print(y_test)

Output []:
        V-RMS     a-RMS    a-Peak     Crest  Temperature
416   0.001025  0.699231  3.150769  4.692308    24.799999
1296  0.001001  0.679231  3.186154  4.833846    25.894615
5811  0.000781  0.807857  5.625714  6.942143    25.387142
4080  0.000704  0.737857  4.745000  6.426429    26.500000
3947  0.000702  0.765385  5.156154  6.780769    26.900000
...        ...       ...       ...       ...          ...
199   0.001056  0.675714  3.173571  4.832857    25.000000
3589  0.000725  0.633846  3.606923  5.649231    26.915384
6611  0.000728  0.850000  6.480000  7.602308    27.000000
4874  0.000751  0.673846  4.307692  6.376154    25.600000
3476  0.000719  0.595385  3.746923  6.187692    26.600000

[1714 rows x 5 columns]
416     0
1296    0
5811    1
4080    1
3947    1
       ..
199     0
3589    1
6611    2
4874    1
3476    1
Name: Bearing State, Length: 1714, dtype: int64
```

Τα σύνολα X_test και y_test μετά τον μετασχηματισμό τους σε πίνακες (arrays) μέσω της βιβλιοθήκης Numpy:

```
Input []:
1  print(X_test)
2  print(y_test)
```

```
Output []:
[[1.02461540e-03 6.99230758e-01 3.15076925e+00 4.69230766e+00
  2.47999992e+01]
 [1.00076928e-03 6.79230773e-01 3.18615382e+00 4.83384609e+00
  2.58946150e+01]
 [7.81428564e-04 8.07857143e-01 5.62571427e+00 6.94214290e+00
  2.53871425e+01]
 ...
 [7.27692301e-04 8.50000001e-01 6.48000006e+00 7.60230769e+00
  2.70000000e+01]
 [7.50769221e-04 6.73846158e-01 4.30769227e+00 6.37615391e+00
  2.56000004e+01]
 [7.19230774e-04 5.95384621e-01 3.74692310e+00 6.18769231e+00
  2.66000004e+01]]
[0 0 1 ... 2 1 1]
```

Παράρτημα Η’

Πρώτον, εισαγωγή μόνο της βιβλιοθήκης pandas και επισκόπηση αρχικού πίνακα δεδομένων των πέντε χαρακτηριστικών:

```
Input []:
1  import pandas as pd
2
3  data = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/
4                      thesis/data/bearing_vibration_metrics.csv')
5  data
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	1
...	...	...	...	...	...	...
8564	0.000728	0.880769	7.337692	8.304615	26.866153	3
8565	0.000703	0.853846	6.926154	8.102308	26.806153	3
8566	0.000705	0.834615	6.556923	7.822308	26.892307	3
8567	0.000713	0.862308	6.893846	7.989231	26.799999	3
8568	0.000708	0.866154	7.378462	8.506923	26.799999	3

8569 rows × 6 columns

Δεύτερον, ομαδοποίηση των δεδομένων βάσει των καταστάσεων Bearing State, διότι ο μετασχηματισμός σε έναν πίνακα 15 χαρακτηριστικών πρέπει να γίνει ξεχωριστά ανά κατηγορία, και στη συνέχεια να γίνει η συσσωμάτωσή τους:

```
Input []:
1  data_grouped = data.groupby('Bearing State')
2  state_1 = data_grouped.get_group(1)
3  state_2 = data_grouped.get_group(2)
4  state_3 = data_grouped.get_group(3)
```

Τρίτον, δημιουργία αντιγράφων των παραπάνω πινάκων δεδομένων, καθώς δεν συνιστάται η επεξεργασία σε πρωτότυπα σύνολα δεδομένων, αφαίρεση της στήλης των ετικετών, διότι

θέλουμε τη μετατόπιση τιμών σε 15 στήλες μόνο για τα πέντε βασικά χαρακτηριστικά, και ενημέρωση των ευρετηρίων για να ξεκινάει η αύξουσα αρίθμηση από το μηδέν:

```
Input []:
1  state_1_s1 = state_1.copy()
2  state_1_s1.drop(columns='Bearing State', inplace=True)
3  state_1_s1.reset_index(drop=True, inplace=True)
4  # print(state_1_s1)
5
6  state_2_s1 = state_2.copy()
7  state_2_s1.drop(columns='Bearing State', inplace=True)
8  state_2_s1.reset_index(drop=True, inplace=True)
9  # print(state_2_s1)
10
11 state_3_s1 = state_3.copy()
12 state_3_s1.drop(columns='Bearing State', inplace=True)
13 state_3_s1.reset_index(drop=True, inplace=True)
14 # print(state_3_s1)
```

Τα στοιχεία state_1_s1, state_2_s1, state_3_s1 είναι αντικείμενα τύπου DataFrame και αντιπροσωπεύουν τις πέντε πρώτες στήλες, ή την πρώτη πεντάδα, των τελικών πινάκων που προσπαθούμε να δημιουργήσουμε. Επίσης, οι εντολές print(), ή κλήση των DataFrame ξεχωριστά για λόγους πιο κατανοητής εμφάνισής, προσφέρουν μια εποπτεία κατά τη διαδικασία του μετασχηματισμού.

Τέταρτον, δημιουργούνται από δυο αντίγραφα (με κατάληξη _s2 και _s3) για καθένα από τα παραπάνω τρία DataFrame, και μετασχηματίζονται κατάλληλα έτσι ώστε να παίξουν τον ρόλο της δεύτερης και τρίτης πεντάδας χαρακτηριστικών αντίστοιχα:

```
Input []:
1  col2 = {'V-RMS': 'V-RMS_2',
2          'a-RMS': 'a-RMS_2',
3          'a-Peak': 'a-Peak_2',
4          'Crest': 'Crest_2',
5          'Temperature': 'Temperature_2',
6          }
7  col3 = {'V-RMS': 'V-RMS_3',
8          'a-RMS': 'a-RMS_3',
9          'a-Peak': 'a-Peak_3',
10         'Crest': 'Crest_3',
11         'Temperature': 'Temperature_3',
12         }
13
14 state_1_s2 = state_1_s1.copy()
15 state_1_s3 = state_1_s1.copy()
```

```
state_1_s2.set_index(pd.RangeIndex(start=-1, stop=3295, step=1),
                     inplace=True)
state_1_s3.set_index(pd.RangeIndex(start=-2, stop=3294, step=1),
                     inplace=True)
state_1_s2.rename(columns=col2, inplace=True)
state_1_s3.rename(columns=col3, inplace=True)
# print(state_1_s2)
# print(state_1_s3)

state_2_s2 = state_2_s1.copy()
state_2_s3 = state_2_s1.copy()
state_2_s2.set_index(pd.RangeIndex(start=-1, stop=2635, step=1),
                     inplace=True)
state_2_s3.set_index(pd.RangeIndex(start=-2, stop=2634, step=1),
                     inplace=True)
state_2_s2.rename(columns=col2, inplace=True)
state_2_s3.rename(columns=col3, inplace=True)
# print(state_2_s2)
# print(state_2_s3)

state_3_s2 = state_3_s1.copy()
state_3_s3 = state_3_s1.copy()
state_3_s2.set_index(pd.RangeIndex(start=-1, stop=2636, step=1),
                     inplace=True)
state_3_s3.set_index(pd.RangeIndex(start=-2, stop=2635, step=1),
                     inplace=True)
state_3_s2.rename(columns=col2, inplace=True)
state_3_s3.rename(columns=col3, inplace=True)
# print(state_3_s2)
# print(state_3_s3)
```

Πέμπτον, οριζόντια συσσωμάτωση των πεντάδων _s1, _s2, _s3 για την κάθε κατάσταση, και προσθήκη νέας στήλης Bearing State με τον αντίστοιχο αναγνωριστικό ακέραιο:

```
Input []:
state_1_s = pd.concat([state_1_s1, state_1_s2, state_1_s3], axis=1)
state_1_s.dropna(inplace=True)
state_1_s['Bearing State'] = 1
# print(state_1_s)

state_2_s = pd.concat([state_2_s1, state_2_s2, state_2_s3], axis=1)
state_2_s.dropna(inplace=True)
state_2_s['Bearing State'] = 2
# print(state_2_s)

state_3_s = pd.concat([state_3_s1, state_3_s2, state_3_s3], axis=1)
```

```
12 state_3_s.dropna(inplace=True)
13 state_3_s['Bearing State'] = 3
14 # print(state_3_s)
```

Έκτον, κατακόρυφη συσσωμάτωση των τριών καταστάσεων, αποθήκευση ως αρχείο .csv, και εκ νέου φόρτωση από το drive για έλεγχο του αποτελέσματος:

Input []:

```
1  data_s = pd.concat([state_1_s, state_2_s, state_3_s],
2                     ignore_index=True)
3  data_s.to_csv('/content/drive/MyDrive/
4                 Colab Notebooks/thesis/data/
5                 bearing_vibration_metrics_shifted.csv', index=False)
6  data_shifted = pd.read_csv('/content/drive/MyDrive/
7                              Colab Notebooks/thesis/data/
8                              bearing_vibration_metrics_shifted.csv')
9  data_shifted
```

Output []:

	V-RMS	a-RMS	a-Peak	Crest	Temperature	V-RMS_2	a-RMS_2	a-Peak_2	Crest_2	Temperature_2	V-RMS_3	a-RMS_3	a-Peak_3	Crest_3	Temperature_3	Bearing State
0	0.001074	0.695385	3.198462	4.740000	26.308461	0.001095	0.713077	3.244615	4.716923	26.299999	0.001079	0.697143	3.227857	4.795000	26.299999	1
1	0.001095	0.713077	3.244615	4.716923	26.299999	0.001079	0.697143	3.227857	4.795000	26.299999	0.001080	0.700000	3.213077	4.793077	26.208462	1
2	0.001079	0.697143	3.227857	4.795000	26.299999	0.001080	0.700000	3.213077	4.793077	26.208462	0.001088	0.700000	3.241538	4.783077	26.200001	1
3	0.001080	0.700000	3.213077	4.793077	26.208462	0.001088	0.700000	3.241538	4.783077	26.200001	0.001085	0.700000	3.200769	4.737692	26.099231	1
4	0.001088	0.700000	3.241538	4.783077	26.200001	0.001085	0.700000	3.200769	4.737692	26.099231	0.001079	0.696154	3.204615	4.761538	25.963846	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8558	0.000719	0.863846	7.023846	8.129231	26.799999	0.000726	0.855385	6.603077	7.735385	26.845384	0.000728	0.880769	7.337692	8.304615	26.866153	3
8559	0.000726	0.855385	6.603077	7.735385	26.845384	0.000728	0.880769	7.337692	8.304615	26.866153	0.000703	0.853846	6.926154	8.102308	26.806153	3
8560	0.000728	0.880769	7.337692	8.304615	26.866153	0.000703	0.853846	6.926154	8.102308	26.806153	0.000705	0.834615	6.556923	7.822308	26.892307	3
8561	0.000703	0.853846	6.926154	8.102308	26.806153	0.000705	0.834615	6.556923	7.822308	26.892307	0.000713	0.862308	6.893846	7.989231	26.799999	3
8562	0.000705	0.834615	6.556923	7.822308	26.892307	0.000713	0.862308	6.893846	7.989231	26.799999	0.000708	0.866154	7.378462	8.506923	26.799999	3

8563 rows × 16 columns

Παράρτημα Θ’

Εικόνα 62:

```
Input []:
1  plt.style.use('ggplot')
2
3  fig, axs = plt.subplots(2,1, figsize=(10,8))
4
5  axs[0].plot(left_norm['vrms_left'], linewidth=0.5,
6              label='Αριστερό Ρουλεμάν', zorder=2, alpha=0.8)
7  axs[0].plot(right_norm['vrms_right'], linewidth=0.5,
8              label='Δεξιό Ρουλεμάν', zorder=1)
9  axs[0].set_title('ΤΑΧΥΤΗΤΑ V-RMS', fontsize='22')
10 axs[0].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
11 axs[0].set_ylabel('inch/sec', fontsize='20')
12 leg = axs[0].legend(loc='upper right', fontsize='16',
13                     handlelength=2, framealpha=0.5)
14 leg.get_frame().set_edgecolor('black')
15 for line in leg.get_lines():
16     line.set_linewidth(2)
17
18 axs[1].plot(left_norm['temp_left'], linewidth=0.5,
19              label='Αριστερό Ρουλεμάν', zorder=2, alpha=0.8)
20 axs[1].plot(right_norm['temp_right'], linewidth=0.5,
21              label='Δεξιό Ρουλεμάν', zorder=1)
22 axs[1].set_title('ΘΕΡΜΟΚΡΑΣΙΑ', fontsize='22')
23 axs[1].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
24 axs[1].set_ylabel('Fahrenheit', fontsize='20')
25 leg = axs[1].legend(loc='upper right', fontsize='16',
26                     handlelength=2, framealpha=0.5)
27 leg.get_frame().set_edgecolor('black')
28 for line in leg.get_lines():
29     line.set_linewidth(2)
30
31 fig.tight_layout()
32
33 plt.show()
```

Εικόνα 63:

```
Input []:
1  plt.style.use('ggplot')
2
3  fig, axs = plt.subplots(2,1, figsize=(10,8))
4
5  axs[0].plot(left_abnorm['vrms_left'], linewidth=0.5,
```

```
            label='Αριστερό Ρουλεμάν', zorder=2, alpha=0.8)
axs[0].plot(right_abnorm['vrms_right'], linewidth=0.5,
            label='Δεξιό Ρουλεμάν', zorder=1)
axs[0].set_title('ΤΑΧΥΤΗΤΑ V-RMS', fontsize='22')
axs[0].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
axs[0].set_ylabel('inch/sec', fontsize='20')
leg = axs[0].legend(loc='upper right', fontsize='16',
                    handlelength=2, framealpha=0.5)
leg.get_frame().set_edgecolor('black')
for line in leg.get_lines():
    line.set_linewidth(2)

axs[1].plot(left_abnorm['temp_left'], linewidth=0.5,
            label='Αριστερό Ρουλεμάν', zorder=2, alpha=0.8)
axs[1].plot(right_abnorm['temp_right'], linewidth=0.5,
            label='Δεξιό Ρουλεμάν', zorder=1)
axs[1].set_title('ΘΕΡΜΟΚΡΑΣΙΑ', fontsize='22')
axs[1].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
axs[1].set_ylabel('Fahrenheit', fontsize='20')
leg = axs[1].legend(loc='upper right', fontsize='16',
                    handlelength=2, framealpha=0.5)
leg.get_frame().set_edgecolor('black')
for line in leg.get_lines():
    line.set_linewidth(2)

fig.tight_layout()

plt.show()
```

Εικόνα 64:

```
Input []:
plt.style.use('ggplot')
plt.figure(figsize=(12,4))

plt.plot(left_norm['vrms_left'], linewidth=0.5,
         label='Αριστερό Ρουλεμάν', zorder=2,
         alpha=0.8, color='#F8766D')
plt.plot(left_abnorm.set_index(
                               pd.RangeIndex(start=5260, stop=7901),
                               inplace=False
                               )['vrms_left'],
         linewidth=0.5, zorder=2,
         alpha=0.8, color='#F8766D')
plt.plot(right_norm['vrms_right'], linewidth=0.5,
         label='Δεξιό Ρουλεμάν', zorder=1, color='#619CFF')
```

```
plt.plot(right_abnorm.set_index(
                                pd.RangeIndex(start=5260, stop=7901),
                                inplace=False
                                )['vrms_right'],
          linewidth=0.5, zorder=1,
          alpha=0.8, color='#619CFF')
plt.axvline(x=5260, color='green',
            linestyle='--', linewidth=1.5)
plt.axhline(y=0.18, color='black',
            linestyle='--', linewidth=1, label='0.18 inch/sec')
plt.title('ΤΑΧΥΤΗΤΑ V-RMS', fontsize='22')
plt.xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
plt.ylabel('inch/sec', fontsize='20')
leg = plt.legend(loc='upper left', fontsize='12',
                 handlelength=2, framealpha=0.5)
leg.get_frame().set_edgecolor('black')
for line in leg.get_lines():
    line.set_linewidth(2)

fig.tight_layout()

plt.show()
```

Εικόνα 65:

```
Input []:
plt.style.use('ggplot')

fig, axs = plt.subplots(2,1, figsize=(10,8))

axs[0].plot(left_norm_train['vrms_left'], linewidth=0.5,
            label='Εκπαίδευση', color='cornflowerblue', zorder=1)
axs[0].plot(left_norm_test['vrms_left'], linewidth=0.5,
            label='Αξιολόγηση', color='orange', zorder=1)
axs[0].plot(left_norm_train_median['vrms_left'], linewidth=1.5,
            label='Διάμεσος Εκπαίδευσης', color='red',
            zorder=2, alpha=1)
axs[0].plot(left_norm_test_median['vrms_left'], linewidth=1.5,
            label='Διάμεσος Αξιολόγησης', color='indigo',
            zorder=2, alpha=0.8)
axs[0].axvline(x=4208, color='darkgreen',
               linestyle='--', linewidth=1.5)
axs[0].set_title('ΤΑΧΥΤΗΤΑ V-RMS', fontsize='22')
axs[0].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
axs[0].set_ylim([0, 0.5])
axs[0].set_ylabel('inch/sec', fontsize='20')
```

```
leg = axs[0].legend(loc='upper left', fontsize='11',
                    handlelength=2, framealpha=0.5)
leg.get_frame().set_edgecolor('black')
for line in leg.get_lines():
    line.set_linewidth(2)

axs[1].plot(left_norm_train['temp_left'], linewidth=0.5,
            label='Εκπαίδευση', color='cornflowerblue', zorder=1)
axs[1].plot(left_norm_test['temp_left'], linewidth=0.5,
            label='Αξιολόγηση', color='orange', zorder=1)
axs[1].plot(left_norm_train_median['temp_left'], linewidth=1.5,
            label='Διάμεσος Εκπαίδευσης', color='red',
            zorder=2, alpha=1)
axs[1].plot(left_norm_test_median['temp_left'], linewidth=1.5,
            label='Διάμεσος Αξιολόγησης', color='indigo',
            zorder=2, alpha=1)
axs[1].axvline(x=4208, color='darkgreen',
               linestyle='--', linewidth=1.5)
axs[1].set_title('ΘΕΡΜΟΚΡΑΣΙΑ', fontsize='22')
axs[1].set_xlabel('Στοιχεία Πινάκων Δεδομένων', fontsize='18')
axs[1].set_ylim([50, 200])
axs[1].set_ylabel('Fahrenheit', fontsize='20')
leg = axs[1].legend(loc='upper left', fontsize='11',
                    handlelength=2, framealpha=0.5)
leg.get_frame().set_edgecolor('black')
for line in leg.get_lines():
    line.set_linewidth(2)

fig.tight_layout()

plt.show()
```

Παράρτημα Ι’

ΣΦΑΛΜΑΤΑ ΑΝΑΚΑΤΑΣΚΕΥΗΣ

Απόκτηση των σφαλμάτων ανακατασκευής όλων των ακολουθιών του συνόλου δεδομένων σε μορφή λίστας:

```
Input []:
1  print(ds_left_norm_test_seq.shape)
2
3  loss_ds_left_norm_test_seq = []
4  for i in range(len(ds_left_norm_test_seq)):
5    x = loss(ds_left_norm_test_seq[i],
6           ds_left_norm_test_seq_pred[i]).numpy()
7    loss_ds_left_norm_test_seq.append(x)
8
9  print(len(loss_ds_left_norm_test_seq))

Output []:
(1006, 24, 2)
1006
```

Οπτικοποίηση σφαλμάτων ανακατασκευής:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(10,4))
3
4  plt.plot(loss_ds_left_norm_test_seq,
5           label='Σφάλμα Ανακατασκευής',
6           color='cornflowerblue')
7  plt.ylim(0,0.04)
8  plt.axhline(y=0.021633, color='red', linestyle='--', linewidth=1,
9              label='Όριο Ανακατασκευής = 0.021633')
10 plt.title('Φυσιολογική Κατάσταση\nΑριστερής Μονάδας', fontsize='22')
11 plt.xlabel('Ακολουθίες', fontsize='20')
12 plt.ylabel('MSE', fontsize='20')
13 plt.legend(loc='upper right', fontsize='16', handlelength=2,
14            framealpha=0.5, edgecolor='black')
15
16 fig.tight_layout()
17
18 plt.show()
```

Δημιουργία αντικειμένου DataFrame για την παραπάνω λίστα των σφαλμάτων και υπολογισμός του πλήθους των ανώμαλων ακολουθιών:

```
Input []:
1  loss_ds_left_norm_test_seq_df =
2           pd.DataFrame(loss_ds_left_norm_test_seq)
3
4  loss_ds_left_norm_test_seq_df[
5           loss_ds_left_norm_test_seq_df[0] > 0.021633
6           ].count()

Output []:
0    0
dtype: int64
```

ΤΥΧΑΙΑ ΑΚΟΛΟΥΘΙΑ ΤΑΧΥΤΗΤΑΣ V-RMS

Δημιουργία λιστών με τις πραγματικές και τις ανακατασκευασμένες τιμές ταχύτητας V-RMS, και μετασχηματισμός τους σε αντικείμενα τύπου DataFrame όπου κάθε στήλη αντιστοιχεί στις τιμές ταχύτητας μιας ακολουθίας:

```
Input []:
1  print(ds_left_norm_test_seq.shape)
2
3  a = ds_left_norm_test_seq.reshape(1006*24,2)[:,0]
4  b = ds_left_norm_test_seq_pred.reshape(1006*24,2)[:,0]
5  seq = 24
6  a_seq = []
7  b_seq = []
8  for i in range(0,1006*24,24):
9  a_seq.append(a[i:i+seq])
10 b_seq.append(b[i:i+seq])
11 ds_left_norm_test_seq_vrms = np.array(a_seq)
12 ds_left_norm_test_seq_pred_vrms = np.array(b_seq)
13
14 ds_left_norm_test_seq_vrms = pd.DataFrame(
15                 ds_left_norm_test_seq_vrms.reshape(1006,24)
16                 ).transpose()
17 ds_left_norm_test_seq_pred_vrms = pd.DataFrame(
18                 ds_left_norm_test_seq_pred_vrms.reshape(1006,24)
19                 ).transpose()
20
21 print(ds_left_norm_test_seq_vrms.shape)
```

```
Output []:
(1006, 24, 2)
(24, 1006)
```

Οπτικοποίηση τυχαίας ακολουθίας ταχύτητας V-RMS και της ανακατασκευής της, και υπολογισμός του επιμέρους σφάλματος ανακατασκευής:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(10,4))
3
4  i_seq = 585
5  plt.plot(ds_left_norm_test_seq_vrms[i_seq],
6           linewidth=0.5, color='blue',
7           label='Ακολουθία Εισόδου')
8  plt.plot(ds_left_norm_test_seq_pred_vrms[i_seq],
9           linewidth=0.5,  color='red',
10          label='Ανακατασκευή')
11 plt.fill_between(
12          np.arange(ds_left_norm_test_seq_vrms.shape[0]),
13          ds_left_norm_test_seq_vrms[i_seq],
14          ds_left_norm_test_seq_pred_vrms[i_seq],
15          color='linen', label='Σφάλμα Ανακατασκευής')
16 plt.ylim(0.2,0.4)
17 plt.title('Τυχαία Ακολουθία Ταχύτητας\n
18            Φυσιολογικής Κατάστασης Αριστερής Μονάδας',
19            fontsize='20')
20 plt.xlabel('Στοιχεία Ακολουθίας', fontsize='18')
21 plt.ylabel('V-RMS', fontsize='18')
22 plt.legend(loc='lower right', fontsize='14',
23            handlelength=2, framealpha=0.5, edgecolor='black')
24
25 fig.tight_layout()
26
27 plt.show()
28
29 print('Σφάλμα Ανακατασκευής ',i_seq,'ης ακολουθίας = ',
30       loss(ds_left_norm_test_seq_vrms[i_seq],
31            ds_left_norm_test_seq_pred_vrms[i_seq]).numpy(),
32       sep='')

Output []:
...
Σφάλμα Ανακατασκευής 585ης ακολουθίας = 0.00019302306
```

ΤΥΧΑΙΑ ΑΚΟΛΟΥΘΙΑ ΘΕΡΜΟΚΡΑΣΙΑΣ TEMPERATURE

Δημιουργία των λιστών με τις πραγματικές και τις ανακασκευασμένες τιμές θερμοκρασίας TEMPERATURE, και μετασχηματισμός τους σε αντικείμενα τύπου DataFrame όπου κάθε στήλη αντιστοιχεί στις τιμές θερμοκρασίας μιας ακολουθίας:

```
Input []:
print(ds_left_norm_test_seq.shape)

a = ds_left_norm_test_seq.reshape(1006*24,2)[:,1]
b = ds_left_norm_test_seq_pred.reshape(1006*24,2)[:,1]
seq = 24
a_seq = []
b_seq = []
for i in range(0,1006*24,24):
a_seq.append(a[i:i+seq])
b_seq.append(b[i:i+seq])
ds_left_norm_test_seq_temp = np.array(a_seq)
ds_left_norm_test_seq_pred_temp = np.array(b_seq)

ds_left_norm_test_seq_temp = pd.DataFrame(
                ds_left_norm_test_seq_temp.reshape(1006,24)
                ).transpose()
ds_left_norm_test_seq_pred_temp = pd.DataFrame(
                ds_left_norm_test_seq_pred_temp.reshape(1006,24)
                ).transpose()

print(ds_left_norm_test_seq_temp.shape)

Output []:
(1006, 24, 2)
(24, 1006)
```

Οπτικοποίηση τυχαίας ακολουθίας θερμοκρασίας TEMPERATURE και της ανακατασκευής της, και υπολογισμός του επιμέρους σφάλματος ανακατασκευής:

```
Input []:
plt.style.use('ggplot')
plt.figure(figsize=(10,4))

i_seq = 585
plt.plot(ds_left_norm_test_seq_temp[i_seq],
         linewidth=0.5, color='blue',
         label='Ακολουθία Εισόδου')
plt.plot(ds_left_norm_test_seq_pred_temp[i_seq],
         linewidth=0.5,  color='red',
```

```
10          label='Ανακατασκευή')
11 plt.fill_between(
12          np.arange(ds_left_norm_test_seq_temp.shape[0]),
13          ds_left_norm_test_seq_temp[i_seq],
14          ds_left_norm_test_seq_pred_temp[i_seq],
15          color='linen', label='Σφάλμα Ανακατασκευής')
16 plt.ylim(0.15,0.35)
17 plt.title('Τυχαία Ακολουθία Θερμοκρασίας\n
18            Φυσιολογικής Κατάστασης Αριστερής Μονάδας',
19            fontsize='20')
20 plt.xlabel('Στοιχεία Ακολουθίας', fontsize='18')
21 plt.ylabel('TEMPERATURE', fontsize='18')
22 plt.legend(loc='upper center', fontsize='14',
23            handlelength=2, framealpha=0.5, edgecolor='black')
24
25 fig.tight_layout()
26
27 plt.show()
28
29 print('Σφάλμα Ανακατασκευής ',i_seq,'ης ακολουθίας = ',
30        loss(ds_left_norm_test_seq_temp[i_seq],
31             ds_left_norm_test_seq_pred_temp[i_seq]).numpy(),
32       sep='')
```

Output []:

```
...
Σφάλμα Ανακατασκευής 585ης ακολουθίας = 0.0003537076
```

ΜΕΣΟΣ ΌΡΟΣ ΑΚΟΛΟΥΘΙΩΝ ΤΑΧΥΤΗΤΑΣ V-RMS

Δημιουργία λίστας με τον μέσο όρο κάθε πραγματικής και ανακατασκευασμένης ακολουθίας ταχύτητας V-RMS από τον προηγούμενο πίνακα δεδομένων DataFrame:

Input []:

```
1  print(ds_left_norm_test_seq_vrms.shape)
2
3  ds_left_norm_test_seq_vrms_mean = []
4  ds_left_norm_test_seq_pred_vrms_mean = []
5  for i in range(ds_left_norm_test_seq_vrms.shape[1]):
6      x = ds_left_norm_test_seq_vrms[i].mean()
7      ds_left_norm_test_seq_vrms_mean.append(x)
8      y = ds_left_norm_test_seq_pred_vrms[i].mean()
9      ds_left_norm_test_seq_pred_vrms_mean.append(y)
10
```

```
11 print(len(ds_left_norm_test_seq_vrms_mean))

Output []:
(24, 1006)
1006
```

Οπτικοποίηση του μέσου όρου κάθε ακολουθίας ταχύτητας V-RMS και της ανακατασκευής της, και υπολογισμός του επιμέρους σφάλματος ανακατασκευής:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(10,4))
3
4  plt.plot(ds_left_norm_test_seq_vrms_mean,
5           linewidth=0.5, color='blue',
6           label='Ακολουθία Εισόδου')
7  plt.plot(ds_left_norm_test_seq_pred_vrms_mean,
8           linewidth=0.5,  color='red',
9           label='Ανακατασκευή')
10 plt.fill_between(
11          np.arange(ds_left_norm_test_seq_vrms.shape[1]),
12          ds_left_norm_test_seq_vrms_mean,
13          ds_left_norm_test_seq_pred_vrms_mean,
14          color='linen', label='Σφάλμα Ανακατασκευής')
15 plt.title('Μέσος Όρος Ακολουθιών Ταχύτητας\n
16            Φυσιολογικής Κατάστασης Αριστερής Μονάδας',
17            fontsize='20')
18 plt.xlabel('Ακολουθίες', fontsize='18')
19 plt.ylabel('VRMS', fontsize='18')
20 plt.legend(loc='upper left', fontsize='14',
21            handlelength=2, framealpha=0.5, edgecolor='black')
22
23 fig.tight_layout()
24
25 plt.show()
26
27 print('Σφάλμα Ανακατασκευής = ',
28       loss(ds_left_norm_test_seq_vrms_mean,
29            ds_left_norm_test_seq_pred_vrms_mean).numpy())

Output []:
...
Σφάλμα Ανακατασκευής =  1.4088119e-05
```

ΜΕΣΟΣ ΌΡΟΣ ΑΚΟΛΟΥΘΙΩΝ ΘΕΡΜΟΚΡΑΣΙΑΣ TEMPERATURE

Δημιουργία λίστας με τον μέσο όρο κάθε πραγματικής και ανακατασκευασμένης ακολουθίας θερμοκρασίας TEMPERATURE από τον προηγούμενο πίνακα δεδομένων DataFrame:

```
Input []:
1  print(ds_left_norm_test_seq_temp.shape)
2
3  ds_left_norm_test_seq_temp_mean = []
4  ds_left_norm_test_seq_pred_temp_mean = []
5  for i in range(ds_left_norm_test_seq_temp.shape[1]):
6      x = ds_left_norm_test_seq_temp[i].mean()
7      ds_left_norm_test_seq_temp_mean.append(x)
8      y = ds_left_norm_test_seq_pred_temp[i].mean()
9      ds_left_norm_test_seq_pred_temp_mean.append(y)
10
11 print(len(ds_left_norm_test_seq_temp_mean))

Output []:
(24, 1006)
1006
```

Οπτικοποίηση του μέσου όρου κάθε ακολουθίας θερμοκρασίας TEMPERATURE και της ανακατασκευής της για την Εικόνα 37, και υπολογισμός του επιμέρους σφάλματος ανακατασκευής:

```
Input []:
1  plt.style.use('ggplot')
2  plt.figure(figsize=(10,4))
3
4  plt.plot(ds_left_norm_test_seq_temp_mean,
5           linewidth=0.5, color='blue',
6           label='Ακολουθία Εισόδου')
7  plt.plot(ds_left_norm_test_seq_pred_temp_mean,
8           linewidth=0.5,  color='red',
9           label='Ανακατασκευή')
10 plt.fill_between(
11          np.arange(ds_left_norm_test_seq_temp.shape[1]),
12          ds_left_norm_test_seq_temp_mean,
13          ds_left_norm_test_seq_pred_temp_mean,
14          color='linen', label='Σφάλμα Ανακατασκευής')
15 plt.title('Μέσος Όρος Ακολουθιών Θερμοκρασίας\n
16            Φυσιολογικής Κατάστασης Αριστερής Μονάδας',
17            fontsize='20')
```

```
18 plt.xlabel('Ακολουθίες', fontsize='18')
19 plt.ylabel('TEMPERATURE', fontsize='18')
20 plt.legend(loc='upper left', fontsize='14',
21            handlelength=2, framealpha=0.5, edgecolor='black')
22
23 fig.tight_layout()
24
25 plt.show()
26
27 print('Σφάλμα Ανακατασκευής = ',
28        loss(ds_left_norm_test_seq_temp_mean,
29             ds_left_norm_test_seq_pred_temp_mean).numpy())
```

Output []:

```
...
Σφάλμα Ανακατασκευής =  1.6369568e-05
```