



Σχολή Θετικών Επιστημών & Τεχνολογίας
Πρόγραμμα Σπουδών : Μεταπτυχιακή Εξειδίκευση στα
Πληροφοριακά Συστήματα (MSc)

Διπλωματική Εργασία
«Κανονικοποίηση χρονοσειράς και πρόγνωση καιρικών συνθηκών με
χρήση μεθόδων τεχνητής νοημοσύνης»

Προκόπης Κιούσης

Επιβλέπων καθηγητής: Ανδρέας Καναβός

Πάτρα, Ιούνιος 2026

Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή («συγγραφέας/δημιουργός») που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο ΕΑΠ, μη αποκλειστική άδεια χρήσης του δικαιώματος αναπαραγωγής, προσαρμογής, δημόσιου δανεισμού, παρουσίασης στο κοινό και ψηφιακής διάχυσής τους διεθνώς, σε ηλεκτρονική μορφή και σε οποιοδήποτε μέσο, για διδακτικούς και ερευνητικούς σκοπούς, άνευ ανταλλάγματος και για όλο το χρόνο διάρκειας των δικαιωμάτων πνευματικής ιδιοκτησίας. Η ανοικτή πρόσβαση στο πλήρες κείμενο για μελέτη και ανάγνωση δεν σημαίνει καθ' οιονδήποτε τρόπο παραχώρηση δικαιωμάτων διανοητικής ιδιοκτησίας του συγγραφέα/δημιουργού ούτε επιτρέπει την αναπαραγωγή, αναδημοσίευση, αντιγραφή, αποθήκευση, πώληση, εμπορική χρήση, μετάδοση, διανομή, έκδοση, εκτέλεση, «μεταφόρτωση» (downloading), «ανάρτηση» (uploading), μετάφραση, τροποποίηση με οποιονδήποτε τρόπο, τμηματικά ή περιληπτικά της εργασίας, χωρίς τη ρητή προηγούμενη έγγραφη συναίνεση του συγγραφέα/δημιουργού. Ο συγγραφέας/δημιουργός διατηρεί το σύνολο των ηθικών και περιουσιακών του δικαιωμάτων.



Κανονικοποίηση χρονοσειράς και πρόγνωση καιρικών συνθηκών με
χρήση μεθόδων τεχνητής νοημοσύνης

Προκόπης Κιούσης

Επιτροπή Επίβλεψης Διπλωματικής Εργασίας

Επιβλέπων Καθηγητής:

Ανδρέας Καναβός

«Ιδιότητα & Ίδρυμα Υπαγωγής»

Συν-Επιβλέπων Καθηγητής:

Παναγιώτης Μουταφής

«Ιδιότητα & Ίδρυμα Υπαγωγής»

Πάτρα, Ιούνιος 2026

*«Στους γονείς μου και στη σύντροφό μου,
που με στήριζαν σε όλη την πορεία των σπουδών»*

Περίληψη

Η παρούσα διπλωματική εργασία επικεντρώνεται στην εφαρμογή μεθοδολογιών Βαθιάς Μάθησης (Deep Learning) για τη μοντελοποίηση και την πρόγνωση μετεωρολογικών δεδομένων. Σκοπός της εργασίας, πέρα από την τεχνική διερεύνηση, είναι η εμβάθυνση στις μεθόδους της Τεχνητής Νοημοσύνης και η απόκτηση ουσιαστικής πρακτικής εξοικείωσης με τη γλώσσα προγραμματισμού Python, η οποία αποτελεί το βασικό εργαλείο υλοποίησης.

Αντικείμενο της έρευνας αποτελεί η ανάπτυξη ενός υπολογιστικού πλαισίου ικανού να αποτυπώνει τις γραμμικές και μη συσχετίσεις καθώς και την έντονη μεταβλητότητα που διέπουν τις ατμοσφαιρικές παραμέτρους. Η εργασία συνδυάζει θεωρητική τεκμηρίωση και πρακτική υλοποίηση σύγχρονων αρχιτεκτονικών, με στόχο τη βελτίωση της προγνωστικής ακρίβειας και της γενικευτικής ικανότητας των μοντέλων.

Στο στάδιο της προεπεξεργασίας, η έρευνα επικεντρώνεται στην επίδραση των τεχνικών κανονικοποίησης στη σταθερότητα και την ταχύτητα σύγκλισης των αλγορίθμων. Αξιολογούνται μέθοδοι όπως οι StandardScaler, MinMaxScaler, RobustScaler, PowerTransformer και QuantileTransformer, με στόχο τη βέλτιστη διαχείριση των ακραίων τιμών και την ομαλοποίηση των κατανομών εισόδου. Παράλληλα, εφαρμόζεται η τεχνική του sin-cos embedding για τη συνεπή μαθηματική αναπαράσταση των κυκλικών μεταβλητών, όπως η διεύθυνση του ανέμου και οι χρονικές περιοδικότητες. Η επιλογή μιας συγκροτημένης στρατηγικής κανονικοποίησης αναδεικνύεται ως θεμελιώδης παράγοντας για την αποδοτική εκπαίδευση των βαθιών αρχιτεκτονικών και τη διασφάλιση της ποιότητας των δεδομένων. Επιπρόσθετα αξιολογούνται διαφορετικοί αλγόριθμοι βαλτιστοποίησης στον κάθε αλγόριθμο κανονικοποίησης.

Στο κύριο μέρος της εργασίας υλοποιείται ένα πολυμεταβλητό μοντέλο Long Short-Term Memory (LSTM), σχεδιασμένο για την ταυτόχρονη πρόβλεψη πολλαπλών μεταβλητών, όπως η θερμοκρασία, η υγρασία και η ατμοσφαιρική πίεση. Επιπλέον, πραγματοποιείται παραγωγή προβλέψεων για χρονικούς ορίζοντες με τη χρήση auto-regressive multi-step forecasting.

Η ανάπτυξη του συστήματος υλοποιείται στη γλώσσα Python, με την αξιοποίηση των βιβλιοθηκών TensorFlow, NumPy και Pandas, διαμορφώνοντας ένα ευέλικτο και επεκτάσιμο περιβάλλον ανάλυσης χρονοσειρών.

Λέξεις Κλειδιά

Κανονικοποίηση Χρονοσειράς, Νευρωνικά Δίκτυα, LSTM, Πρόγνωση Καιρού, Python

Time series normalization and weather forecasting using artificial intelligence methods

Abstract

The present master's thesis focuses on the implementation of Deep Learning methodologies for the modeling and forecasting of meteorological data. Beyond the technical exploration of the subject, a core objective of this work is to deepen the understanding of Artificial Intelligence methods and acquire significant practical expertise in the Python programming language, which constitutes the primary tool for the implementation of the system. The scope of the research involves the development of a robust computational framework capable of capturing the complex non-linear correlations and the high degree of variability inherent in atmospheric parameters. This study integrates theoretical analysis with the practical application of modern neural network architectures, aiming to enhance both the predictive precision and the generalization performance of the models.

During the preprocessing stage, the research examines the impact of normalization techniques on the stability and convergence speed of the algorithms. Methods such as StandardScaler, MinMaxScaler, RobustScaler, PowerTransformer και QuantileTransformer are evaluated with the objective of optimally handling extreme values and normalizing input distributions. Furthermore, the sin-cos embedding technique is employed for the mathematically consistent representation of cyclical variables, such as wind direction and temporal periodicities. The determination of a coherent normalization strategy is identified as a critical factor for the efficient training of deep learning architectures and the overall preservation of data quality. In addition, optimizers evaluate for every scaler.

The core implementation of the thesis involves a multivariate Long Short-Term Memory (LSTM) model, specifically designed for the concurrent prediction of multiple meteorological variables, including temperature, humidity, and atmospheric pressure. Furthermore, forecasts over extended time horizons are generated using an auto-regressive multi-step forecasting approach.

The algorithm is developed in the Python programming language, utilizing the TensorFlow, NumPy, and Pandas libraries, resulting in a flexible and extensible time series analysis environment.

Keywords

Time Series Normalization, Neural Networks, LSTM, Forecasting, Python

Περιεχόμενα

Περίληψη	ii
Abstract	iv
Περιεχόμενα	vi
Κατάλογος Εικόνων	viii
Κατάλογος Πινάκων	ix
Συντομογραφίες & Ακρωνύμια	x
1. Στοχοθεσία Διπλωματικής Εργασίας	1
1.1 ΕΙΣΑΓΩΓΗ	1
1.2 ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ	2
1.3 ΔΟΜΗ ΕΡΓΑΣΙΑΣ	3
2. Κανονικοποίηση	4
2.1 Θεωρία Κανονικοποίησης	4
2.2 Μέθοδοι Κανονικοποίησης	4
3. Τεχνητή Νοημοσύνη	10
3.1 Εισαγωγή	10
3.2 Μηχανική Μάθηση	10
3.2.1 Κατηγορίες Μηχανικής Μάθησης	11
3.2.2 Εξαγωγή Χαρακτηριστικών (Feature Engineering)	14
3.2.3 Κλασικοί Αλγόριθμοι Μηχανικής Μάθησης	14
3.2.4 Πλεονεκτήματα και Περιορισμοί Μηχανικής Μάθησης	18
3.3 Βαθιά Μάθηση (Deep Learning)	19
3.3.1 Εισαγωγή στη Βαθιά Μάθηση (Deep Learning)	19
3.3.2 Θεωρία Νευρωνικών Δικτύων	19
3.3.3 Εκπαίδευση Νευρωνικών Δικτύων	21
3.3.4 Τύποι Νευρωνικών Δικτύων και Μοντέλα Μνήμης	22
3.4 Αλγόριθμοι Βελτιστοποίησης (Optimizers)	25
3.4.1 Εισαγωγή στους Αλγορίθμους Βελτιστοποίησης	25
3.4.2 Κατηγορίες και Τύποι Αλγορίθμων Βελτιστοποίησης	27
4. Python	30
4.1 Εισαγωγή Κεφαλαίου	30
4.2 Εισαγωγή Python	30
4.3 Βιβλιοθήκες	31
4.3.1 Pandas	31
4.3.2 NumPy	32
4.3.3 Matplotlib	32
4.3.4 Sci-Kit Learn	32
4.3.5 Tensorflow	33
5. Σύστημα Καταγραφής Δεδομένων	34
5.1 Εισαγωγή Κεφαλαίου	34
5.2 Αισθητήρες Σταθμού	35
5.3 Εμφάνιση και Αποθήκευση Δεδομένων	38

6. Υλοποίηση Κώδικα Πρόγνωσης.....	40
6.1 Εισαγωγή Κεφαλαίου	40
6.2 Διαχείριση Χρονοσειρών	40
6.3 Εύρεση Βέλτιστων Μεθόδων Κανονικοποίησης και Αλγορίθμου Βελτιστοποίησης....	43
6.3.1 Κώδικας Σύγκρισης Scalers	46
6.3.2 Αποτελέσματα Συγκρίσεων	63
6.4 Υλοποίηση Κώδικα.....	68
6.4.1 Κώδικας Καιρικής Πρόγνωσης.....	68
6.4.2 Σύγκριση Αρχιτεκτονικών LSTM.....	76
6.4.3 Παραμετροποίηση LSTM	81
6.4.4 Πρόγνωση ανά παράμετρο.....	85
7. Συμπεράσματα.....	91
Βιβλιογραφία	92

Κατάλογος Εικόνων

Εικόνα 1 Διάγραμμα Ροής Εποπτευόμενης Μάθησης.....	12
Εικόνα 2 Διάγραμμα Ροής Μη Εποπτευόμενης Μάθησης.....	12
Εικόνα 3 Διάγραμμα Ροής Ενισχυτικής Μάθησης.....	13
Εικόνα 4 Βιολογικά και Τεχνητά Νευρωνικά Δίκτυα	20
Εικόνα 5 RNN Δίκτυο	24
Εικόνα 6 Λογότυπο Γλώσσας Python.....	30
Εικόνα 7 Ανεμόμετρο Vantage Pro2 (6410).....	36
Εικόνα 8 Αισθητήρας Θερμοκρασίας - Σχετικής Υγρασίας.....	36
Εικόνα 9 Βροχόμετρο	37
Εικόνα 10 Configuration 1 KDE T_mean με StandarScaler	63
Εικόνα 11 Configuration 1 Time Series Overlay T_mean με StandarScaler	64
Εικόνα 12 Configuration 2 KDE T_mean με PowerTransformer(Yeo-Johnson)	64
Εικόνα 13 Configuration 2 Time Series Overlay T_mean με PowerTransformer(Yeo-Johnson).....	64
Εικόνα 14 Configuration 3 KDE T_mean με QuantileTransformer (normal).....	65
Εικόνα 15 Configuration 3 Time Series Overlay T_mean με QuantileTransformer (normal).....	65
Εικόνα 16 LSTM 128 → 64 → 32	77
Εικόνα 17 LSTM 32 → 64 → 32	78
Εικόνα 18 LSTM 64 → 32 → 64	79
Εικόνα 19 LSTM 64 → 32 → 16	79
Εικόνα 20 Απεικόνιση LSTM.....	81
Εικόνα 21 Train Loss Σύνολο No. 5	83
Εικόνα 22 Train Loss Σύνολο No. 6	83
Εικόνα 23 Train Loss Σύνολο No. 7	84
Εικόνα 24 True vs Predicted Values Θερμοκρασίας (T_mean)	85
Εικόνα 25 True vs Predicted Values Θερμοκρασίας (T_max).....	85
Εικόνα 26 True vs Predicted Values Θερμοκρασίας (T_min).....	86
Εικόνα 27 True vs Predicted Values Σχετικής Υγρασίας (RH_mean)	86
Εικόνα 28 True vs Predicted Values Σχετικής Υγρασίας (RH_max).....	86
Εικόνα 29 True vs Predicted Values Σχετικής Υγρασίας (RH_min).....	87
Εικόνα 30 True vs Predicted Values Πίεσης (Prs_mean).....	87
Εικόνα 31 True vs Predicted Values Πίεσης (Prs_max).....	87
Εικόνα 32 True vs Predicted Values Πίεσης (Prs_min)	88
Εικόνα 33 True vs Predicted Values Βροχόπτωσης (Ac_R)	88
Εικόνα 34 True vs Predicted Values Ταχύτητας Ανέμου (WS_mean).....	89
Εικόνα 35 True vs Predicted Values Ριπή Ανέμου (WG).....	89
Εικόνα 36 True vs Predicted Values Διεύθυνση Ανέμου (DWD_sin)	90
Εικόνα 37 True vs Predicted Values Διεύθυνση Ανέμου (DWD_cos)	90

Κατάλογος Πινάκων

Πίνακας 1 Μορφή Αρχείου.....	42
Πίνακας 2 Optimizer Adam MAE/MSE.....	66
Πίνακας 3 Optimizer RMSprop MAE/MSE.....	67
Πίνακας 4 Optimizer Nadam MAE/MSE	67
Πίνακας 5 Τελικοί Scalers και Optimizer.....	68
Πίνακας 6 Σύγκριση MAE ανά σύνολο παραμέτρων.....	82

Συντομογραφίες & Ακρωνύμια

Ακολουθούν κάποια παραδείγματα:

ΔΕ	Διπλωματική Εργασία
ΕΑΠ	Ελληνικό Ανοικτό Πανεπιστήμιο
ΜΜ	Μηχανική Μάθηση
ΒΜ	Βαθιά Μάθηση
ΤΝΔ	Τεχνητά Νευρωνικά Δίκτυα
ΕΑΑ	Εθνικό Αστεροσκοπείο Αθηνών

1. Στοχοθεσία Διπλωματικής Εργασίας

1.1 ΕΙΣΑΓΩΓΗ

Η ικανότητα ακριβούς πρόβλεψης μετεωρολογικών παραμέτρων, όπως η θερμοκρασία, η υγρασία και η ταχύτητα του ανέμου, έχει άμεσο αντίκτυπο σε ένα ευρύ φάσμα ανθρώπινων δραστηριοτήτων, από την αγροτική παραγωγή και τις μεταφορές μέχρι τη διαχείριση της ενέργειας και την προστασία της δημόσιας υγείας. Επίσης, ο κλάδος της μετεωρολογίας έχει συμβάλει τα μέγιστα στον ρου της ιστορίας, όπως για παράδειγμα στον Β' Παγκόσμιο Πόλεμο, όταν το γερμανικό επιτελείο στρατού δεν μελέτησε τις μετεωρολογικές προβλέψεις οι οποίες προέβλεπαν στον βαρύτερο χειμώνα του αιώνα. Έτσι παρά τις ανησυχίες των μετεωρολόγων, ο γερμανικός στρατός εισέβαλε στην Ρωσία αποκομίζοντας στο τέλος μια ολέθρια ήττα με αρκετές και βαριές απώλειες.

Ωστόσο, η ατμόσφαιρα αποτελεί ένα ογκώδες και δυναμικό σύστημα πληροφοριών, όπου μικρές μεταβολές στις αρχικές συνθήκες μπορούν να οδηγήσουν σε διαφορετικά αποτελέσματα. Αυτή η εγγενής πολυπλοκότητα καθιστά τις παραδοσιακές στατιστικές μεθόδους συχνά ανεπαρκείς για τη σύλληψη των μη γραμμικών σχέσεων που διέπουν τα καιρικά δεδομένα. Άλλο ένα ιστορικό παράδειγμα που αξίζει να αναφερθεί στο σημείο αυτό είναι η αποβίβαση της Νορμανδίας και πάλι στην περίοδο του Β' Παγκοσμίου Πολέμου. Ενώ οι Γερμανοί μετεωρολόγοι προέβλεπαν κακοκαιρία για τουλάχιστον δώδεκα ημέρες, συνεπώς δεν θα ήταν εφικτό στους Συμμάχους να πραγματοποιήσουν αποβίβαση. Έτσι το γερμανικό στρατεύμα εφησυχάστηκε και ο στρατηγός Ρόμμελ αναχώρησε ήσυχος για τον Βερολίνο. Εν αντιθέση, οι Αμερικανοί και Βρετανοί μετεωρολόγοι προέβλεπαν μια μικρή ανάπαυλα πέντε ωρών. Το συμμαχικό επιτελείο εμπιστεύτηκε τις προβλέψεις και συνάμμα με μεγάλο ρίσκο πραγματοποιήθηκε η αποβίβαση πιάνοντας τον γερμανικό στρατό σε απόλυτο αιφνιδιασμό. Καθίσταται κατανοητό πως η επιστήμη της μετεωρολογίας είναι ένας κλάδος άρρηκτα συνδεδεμένος με την ανθρώπινη ζωή και ιστορία, βάση του οποίου παίρνουμε διάφορες αποφάσεις από απλές στιγμές για την καθημερινότητα μας, για τις εργασίες μας αλλά και για σημαντικά, όπως είδαμε, γεγονότα.

Η σύγχρονη μετεωρολογία χαρακτηρίζεται από τη διαθεσιμότητα τεράστιου όγκου δεδομένων (Big Data) που προέρχονται από επίγειους σταθμούς, δορυφόρους και αισθητήρες. Παρά τον πλούτο της πληροφορίας, τα δεδομένα αυτά παρουσιάζουν σημαντικές προκλήσεις, όπως έντονη μεταβλητότητα, αστοχία, τυχόν απώλεια, θόρυβο και διαφορετικές κλίμακες μέτρησης. Η αποτελεσματική επεξεργασία τους απαιτεί προηγμένες τεχνικές κανονικοποίησης, καθώς η απόδοση των υπολογιστικών μοντέλων εξαρτάται άμεσα από την ποιότητα της εισερχόμενης πληροφορίας. Η σωστή διαχείριση ακραίων τιμών και η μαθηματική αναπαράσταση κυκλικών φαινομένων αποτελούν κρίσιμα στάδια που καθορίζουν τη σταθερότητα της εκπαιδευτικής διαδικασίας.

1.2 ΣΚΟΠΟΣ ΕΡΓΑΣΙΑΣ

Κύριος σκοπός της παρούσας διπλωματικής εργασίας είναι η κατανόηση των αρχών της Κανονικοποίησης, της Τεχνητής Νοημοσύνης, συγκεκριμένα της Βαθιάς Μάθησης (Deep Learning) και των Νευρωνικών Δικτύων μέσα από την εφαρμογή τους σε ρεαλιστικά προβλήματα ανάλυσης και πρόγνωσης καιρικών συνθηκών. Η εργασία επιδιώκει όχι μόνο τη θεωρητική κατανόηση των μεθοδολογιών αυτών, αλλά και την πρακτική τους αξιοποίηση σε πραγματικά δεδομένα, με στόχο την ανάπτυξη ενός λειτουργικού συστήματος πρόγνωσης καιρού. Επιπρόσθετα, με την δοκιμή διαφόρων μοντέλων και αλγορίθμων θα εντοπιστούν και θα καταγραφούν οι διαφορές μεταξύ των αλγορίθμων.

Μέσω της ανάπτυξης του υπολογιστικού κώδικα, επιδιώκεται η πρακτική κατανόηση του οικοσυστήματος της Python και των εξειδικευμένων βιβλιοθηκών της, ενώ παράλληλα δομείται μια ολοκληρωμένη ροή εργασιών. Αυτή η διαδικασία περιλαμβάνει την υλοποίηση τεχνικών κανονικοποίησης και μετασχηματισμού των δεδομένων ώστε να γίνει εφικτό να διαχειριστούν σωστά από τα μοντέλα των νευρωνικών δικτύων. Βάση θα δωθεί και στους αλγόριθμους βελτιστοποίησης των νευρωνικών δικτύων, προτού καταλήξουμε στην εκπαίδευση και αξιολόγηση πολυμεταβλητών μοντέλων πρόγνωσης χρονοσειρών. Τελικός στόχος είναι η ανάδειξη των διαφορών μεταξύ των μοντέλων της τεχνητής νοημοσύνης και της κρίσιμης επίδρασης που ασκεί η ορθή προετοιμασία των δεδομένων στην ευστάθεια και την τελική προγνωστική ικανότητα των μοντέλων τεχνητής νοημοσύνης.

1.3 ΔΟΜΗ ΕΡΓΑΣΙΑΣ

Η παρούσα διπλωματική εργασία τμηματοποιείται σε επτά κεφάλαια. Στο πρώτο κεφάλαιο γίνεται αναφορά του λόγου κανονικοποίησης των αρχικών δεδομένων καθώς και μοντέλα κανονικοποίησης.

Εν συνεχεία, στο επόμενο κεφάλαιο πραγματοποιείται μια έρευνα και αναφορά στην επιστήμη της Τεχνητής Νοημοσύνης. Αναπτύσσονται τμηματικά οι επιστήμες της Μηχανικής και Βαθιάς Μάθησης, η τεχνολογία των Τεχνητών Νευρωνικών Δικτύων καθώς και οι αλγόριθμοι βελτιστοποίησης.

Στο τρίτο κεφάλαιο παρουσιάζεται η γλώσσα Python και οι βιβλιοθήκες αυτής. Στο τέταρτο κεφάλαιο παρατίθεται όλη η διαδικασία καταγραφής και αρχικής επεξεργασίας των μετεωρολογικών δεδομένων που θα χρησιμοποιηθούν, τα οποία έχουν ληφθεί από το Εθνικό Αστεροσκοπείο Αθηνών.

Το πέμπτο κεφάλαιο αφορά την υλοποίηση και εκτέλεση της εργασίας και του κώδικα κανονικοποίησης και πρόγνωσης. Στο κεφάλαιο αυτό γίνεται αναφορά στην διαδικασία υλοποίησης του κώδικα. Πιο συγκεκριμένα γίνεται γίνεται σύγκριση μεταξύ διαφόρων μεθόδων κανονικοποίησης για τα εισαγόμενα δεδομένα και δοκιμάζονται διάφοροι αλγόριθμοι βελτιστοποίησης προκειμένου μέσω γραφημάτων να εντοπιστεί και να παρουσιαστεί ο βέλτιστος αλγόριθμος.

Τέλος, στο έκτο και τελευταίο κεφάλαιο αξιολογούνται τα αποτελέσματα που προέκυψαν και καταγράφονται τα συμπεράσματα της διπλωματικής εργασίας.

Επίσης, γίνεται χρήση σχημάτων, εξισώσεων και εικόνων για την καλύτερη κατανόηση του αναγνώστη.

2. Κανονικοποίηση

2.1 Θεωρία Κανονικοποίησης

Η κανονικοποίηση (normalization) ή αλλιώς ανακλιμάκωση (feature scaling) αποτελεί το στάδιο της αρχικής επεξεργασίας δεδομένων και μετατρέπει τις αρχικές μεταβλητές σε κοινή κλίμακα, χωρίς να αλλοιώνει τη σχετική πληροφορία που αυτές περιέχουν. Οι μεταβλητές συχνά εμφανίζουν διαφορετικές μονάδες μέτρησης, εύρη τιμών ή στατιστικές ιδιότητες, γεγονός που επηρεάζει αρνητικά τους αλγορίθμους μηχανικής μάθησης. Η κανονικοποίηση επιλύει αυτό το πρόβλημα εφαρμόζοντας μαθηματικούς μετασχηματισμούς όπως z-score standardization, min-max scaling ή robust scaling, ώστε όλες οι μεταβλητές να αποκτήσουν συγκρίσιμη αριθμητική συμπεριφορά (Han, Pei & Kamber, 2011). Οι αρχικές τιμές, μετατρέπονται σε διανύσματα τα οποία μπορούν να αξιοποιηθούν μετέπειτα στα διάφορα μοντέλα νευρωνικών δικτύων. Με αυτόν τον τρόπο, μειώνεται η επίδραση των διαφορετικών κλιμάκων και βελτιώνεται η σταθερότητα των αλγορίθμων.

Για τη διασφάλιση της ευστάθειας και της αποδοτικότητας των αλγορίθμων Βαθιάς Μάθησης (Deep Learning), η κανονικοποίηση θεωρείται θεμελιώδη προϋπόθεση. Η απουσία της, είναι αρκετά πιθανό να προκαλέσει αστάθεια στις κλίσεις ή και πλήρη αδυναμία εκπαίδευσης του μοντέλου, καθώς τα νευρωνικά δίκτυα παρουσιάζουν υψηλή ευαισθησία στις αποκλίσεις κλίμακας στα δεδομένα εισόδου. Η ομογενοποίηση των εισόδων βελτιστοποιεί τη λειτουργία του αλγορίθμου καθόδου κλίσης (gradient descent) και περιορίζει κινδύνους όπως η εξαφάνιση (vanishing) ή η έκρηξη (exploding) των κλίσεων (Goodfellow, Bengio & Courville, 2016).

2.2 Μέθοδοι Κανονικοποίησης

Καθώς οι μεταβλητές ποικίλουν ως προς τα χαρακτηριστικά τους όπως για παράδειγμα η μονάδα μέτρησης, η κλίμακα και το εύρος τους, απαιτείται διαφορετική προσέγγιση προκειμένου να γίνει η κανονικοποίηση τους. Έτσι υπάρχει μια ποικιλία από μεθόδους από την οποία μπορεί να επιλεγεί η καταλληλότερη για κάθε τύπο δεδομένων, βελτιώνοντας έτσι τη σταθερότητα και την απόδοση των μοντέλων βαθιάς μάθησης.

Μερικές από τις μεθόδους κανονικοποίησης είναι οι εξής:

Z-score Scaling (Standardization)

Η μέθοδος μετατρέπει τα δεδομένα εισόδου ώστε να παρουσιάζουν μηδενική $\mu = 0$ μέση τιμή και μοναδιαία $\sigma = 1$ τυπική απόκλιση. Για Gause δεδομένα, είναι απαραίτητη καθώς διασφαλίζει τη διατήρηση των αναλογικών αποστάσεων μεταξύ των τιμών.

$$x' = \frac{x - \mu}{\sigma}$$

Εφαρμόζοντάς την σε νευρωνικό δίκτυο, η μέθοδος του Z-score scaling βελτιστοποιεί τη διαδρομή του αλγορίθμου καθόδου κλίσης (gradient descent), αποτρέποντας τις διακυμάνσεις και επιτυγχάνοντας ταχύτερη σύγκλιση του μοντέλου.

Min–Max Scaling

Με το Min–Max Scaling, όλες οι τιμές ενός συνόλου δεδομένων «συμπιέζονται» ώστε να χωρέσουν σε ένα διάστημα $[0,1]$ ή $[-1,1]$, χωρίς ωστόσο να αλλάζει η μεταξύ τους αναλογία.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Είναι ιδανική για δεδομένα με προκαθορισμένα όρια μιας και το κυριότερο πλεονέκτημα της είναι η διατήρησης της δομής κατανομής και των σχετικών αποστάσεων καθώς ένα και μόνο ακραίο δείγμα μπορεί να συμπίπτει το σύνολο της πληροφορίας σε ένα αρκετά μικρό υποδιάστημα.

Unit Range Scaling

Πρόκειται για μια παραλλαγή του Min - MaxMax scaling η οποία επιτρέπει προσαρμοσμένα εύρη $[-1,1]$. Χρησιμοποιείται σε συγκεκριμένο αριθμητικό εύρος για λόγους σταθερότητας ή συμβατότητας με μοντέλα. Διατηρεί τις αναλογίες των τιμών αλλά παραμένει ευαίσθητη σε outliers.

Robust Scaling

Βασίζεται στους στατικούς δείκτες διάμεσου και IQR (interquartile range), καθιστώντας τη κατάλληλη για ακραίες τιμές. Είναι ιδανική για δεδομένα με έντονη μεταβλητότητα όπως η ταχύτητα ανέμου ή οικονομικές χρονοσειρές.

$$x' = \frac{x - \text{median}}{IQR}$$

MaxAbs Scaling

Στη μέθοδο αυτή, οι τιμές μιας παραμέτρου αναπροσαρμόζονται με βάση τη μέγιστη απόλυτη τιμή τους, ώστε να κυμαίνονται στο εύρος $[-1,1]$. Πλεονέκτημα της είναι πως διατηρεί την αραιότητα (sparsity) των δεδομένων καθώς δεν μετατοπίζει τις τιμές και διατηρεί τις μηδενικές τιμές.

$$x' = \frac{x}{|x|_{\max}}$$

Ιδανική μέθοδος για επεξεργασία αραιών πινάκων (sparse matrices) καθώς και δεδομένα τα οποία τείνουν στο μηδέν.

L1 / L2 Normalization (Vector Normalization)

Η μέθοδος κανονικοποίησης διανυσμάτων (Vector Normalization), διαφέρει από τις άλλες μεθόδους γιατί δεν εξετάζει κάθε χαρακτηριστικό ξεχωριστά, ως ένα ενιαίο διάνυσμα στο χώρο. Η μέθοδος L1 Normalization διαιρεί τα στοιχεία με το άθροισμα των απόλυτων τιμών τους, μετατρέποντας τις τιμές σε αναλογίες που αθροίζουν στο 1.

$$x'_i = \frac{x_i}{\sum_{j=1}^n |x_j|}$$

Αντίστοιχα στην L2 Normalization, κάθε τιμή διαιρείται με την Ευκλείδεια απόσταση του δείγματος, με αποτέλεσμα το τελικό διάνυσμα να έχει μήκος ίσο με μονάδα. Είναι ιδανική για

συγκρίσεις ομοιότητας σε εφαρμογές όπως η αναγνώριση κειμένου ή προσώπων. Και οι δύο μέθοδοι εξασφαλίζουν ότι τα διανύσματα θα έχουν την ίδια βαρύτητα στο μοντέλο, ανεξάρτητα από το αρχικό τους μέγεθος, εστιάζοντας περισσότερο στη "γωνία" και τη σχέση των χαρακτηριστικών μεταξύ τους.

Logarithmic Transforms (log, log10, log1p)

Οι αλγοριθμικοί μετασχηματισμοί, χρησιμοποιούνται για να μειώσουν την ασυμμετρία και τυχόν απότομες κορυφές (spikes) στις τιμές των δεδομένων, εξασφαλίζοντας μια σταθερή διακύμανση. Είναι κατάλληλα σε περιπτώσεις δεδομένων με μακρές ουρές (heavy tails), όπως για παράδειγμα στις μετρήσεις βροχόπτωσης ή τα οικονομικά μεγέθη, όπου λίγες ακραίες τιμές μπορούν να επηρεάσουν τη συνολική εικόνα. Πιο συγκεκριμένα ο μετασχηματισμός log1p είναι απαραίτητος όταν το σύνολο δεδομένων περιλαμβάνει μηδενικά, καθώς επιτρέπει τον υπολογισμό του λογαρίθμου χωρίς μαθηματικά σφάλματα, προσθέτοντας προληπτικά μονάδα σε κάθε τιμή.

$$x' = \log(1+x)$$

Power Transforms (Box–Cox, Yeo–Johnson)

Οι μετασχηματισμοί ισχύος, στοχεύουν στην μετατροπή των δεδομένων ώστε να προσεγγίζουν όσο το δυνατόν περισσότερο την κανονική (Gauss) κατανομή. Η μέθοδος εφαρμόζει σταθεροποίηση σε τυχόν διακυμάνσεις των τιμών της χρονοσειράς και ταυτόχρονα μειώνεται όποια ασυμμετρία (skewness) υπάρχει. Προκειμένου να πραγματοποιηθούν αυτές οι μετατροπές, χρησιμοποιούνται προηγμένα μαθηματικά μοντέλα τα power transformation, εξ' ου και το όνομα της μεθόδου. Η μέθοδος Box–Cox περιορίζεται αποκλειστικά σε σύνολα με θετικές τιμές, ενώ στη μέθοδο Yeo–Johnson γίνεται επεξεργασία τόσο θετικών όσο και αρνητικών αριθμών. Οι μέθοδοι αυτοί αξιοποιούνται κατά κύριο λόγο σε δεδομένα έντονης στατιστικής ασυμμετρίας, καθώς ομαλοποιούν τη διακύμανση και βοηθούν τους αλγορίθμους να εντοπίσουν μοτίβα που σε άλλη περίπτωση θα επισκιάζονταν από τη μη κανονικότητα των δεδομένων.

Quantile Transformation

Ο μετασχηματισμός ποσοστημορίων (**Quantile Transformation**) τροποποιεί τα δεδομένα σε ομοιόμορφη ή κανονική μορφή, εκμηδενίζοντας τις ακραίες τιμές (outliers). Είναι κατάλληλη για σύνολα δεδομένων με σύνθετες ή εξαιρετικά μη κανονικές κατανομές, καθώς ομαλοποιεί τις αποκλίσεις. Αν και είναι αποτελεσματικός, χρήζει ιδιαίτερης προσοχής, καθώς κατά τη διαδικασία είναι πιθανό να αλλοιωθούν οι πραγματικές αποστάσεις μεταξύ των τιμών και να παραμορφωθεί η φυσική δομή και οι σχέσεις των αρχικών δεδομένων.

Rank Transformation

Ο μετασχηματισμός κατάταξης (Rank Transformation) μετατρέπει τα αρχικά δεδομένα με σειρά προτεραιότητας που αυτά καταλαμβάνουν στο σύνολο, διασφαλίζοντας τη διατήρηση της ιεραρχίας αλλά όχι των ακριβών αποστάσεων μεταξύ των τιμών. Η μέθοδος είναι ιδιαίτερα ανθεκτική σε ακραίες τιμές (outliers) και η οποία διευκολύνει τη διαχείριση μη γραμμικών σχέσεων. Κύριο μειονέκτημα της είναι η απώλεια της ποσοτικής πληροφορίας, καθώς η απόλυτη διαφορά μεταξύ δύο τιμών θυσιάζεται υπέρ της σχετικής τους θέσης.

Decimal Scaling

Η μέθοδος αυτή μετασχηματίζει τα δεδομένα, μετακινώντας την υποδιαστολή της τιμής τους στο εύρος $[-1,1]$. Αν και πρόκειται για απλή και γρήγορη μέθοδο, δεν πραγματοποιείται κατανομή δεδομένων. Χρησιμοποιείται κυρίως σε εκπαιδευτικά παραδείγματα ή σε πολύ απλά datasets.

Clipping / Winsorization

Η λειτουργία της μεθόδου είναι να μειώνει την επίδραση των outliers και περιορίζει τις ακραίες τιμές σε προκαθορισμένα όρια. Συχνά χρησιμοποιείται πριν από άλλες μεθόδους, καθώς δεν αποτελεί ολοκληρωμένη μέθοδο κανονικοποίησης.

Sin-Cos Embedding

Μετατρέπει κυκλικές μεταβλητές (όπως η διεύθυνση ανέμου, η ώρα ή οι εποχές) σε δύο συνεχείς συνιστώσες: $\sin[f_0](\theta)$ και $\cos[f_0](\theta)$. Έτσι, χαρτογραφούνται οι τιμές σε ένα ζεύγος συντεταγμένων ημιτόνου και συνημιτόνου πάνω σε έναν μοναδιαίο κύκλο. Διατηρεί δηλαδή τη γεωμετρική κυκλικότητα και αποτρέπει ασυνέχειες όπως η μετάβαση από το σημείο 0° στο 360° . Με αυτόν τον τρόπο, διατηρείται η γεωμετρική συνέχεια και η σχετική απόσταση μεταξύ των χρονικών ή γωνιακών σημείων, επιτρέποντας στο μοντέλο να αντιληφθεί τη φυσική περιοδικότητα των δεδομένων χωρίς την απώλεια πληροφορίας που προκαλεί η κλασική κανονικοποίηση.

3. Τεχνητή Νοημοσύνη

3.1 Εισαγωγή

Η Τεχνητή Νοημοσύνη - TX (Artificial Intelligence) αποτελεί πλέον έναν από τους ισχυρότερους κλάδους της επιστήμης των υπολογιστών, οδηγώντας σε έναν θεμελιώδη μετασχηματισμό στη προσέγγιση της μεθοδολογίας της ανάλυσης δεδομένων, των μοντέλων πρόγνωσης και της αυτοματοποίησης των συστημάτων. Επίκεντρο της ερευνητικής ανάπτυξης είναι η σχεδίαση και υλοποίηση υπολογιστικών αρχιτεκτονικών, ικανών να προσομοιώνουν ανώτερες γνωστικές λειτουργίες όπως η λήψη αποφάσεων βάση στοχαστικών προτύπων, η επαγωγική μάθηση, ο ορθολογικός συλλογισμός και η δυναμική προσαρμογή σε μεταβαλλόμενα σύνολα δεδομένων.

Η εξέλιξη της TX ακολουθεί μια μη γραμμική πορεία, με εναλλασσόμενες περιόδους έντονης ανάπτυξης και σχετικής ύφεσης. Ωστόσο, στη σύγχρονη εποχή σηματοδοτείται μια πρωτοφανής αναγέννηση, η οποία στηρίζεται στην ραγδαία αύξηση των διαθέσιμων δεδομένων μεγάλης κλίμακας (Big Data), την δυναμική ενίσχυση της υπολογιστικής ισχύος και δομών καθώς και τη βελτιστοποίηση των αλγορίθμων Μηχανικής και Βαθιάς Μάθησης (Machine and Deep Learning). Στο παρόν τεχνολογικό οικοσύστημα, η TX αποτελεί βάση σε ένα ευρύ φάσμα εφαρμογών υψηλής πολυπλοκότητας όπως τα συστήματα λήψης αποφάσεων, η αυτόνομη πλοήγηση, η επεξεργασία της φυσικής γλώσσας καθώς και ανάλυσης των χρηματοοικονομικών και περιβαλλοντικών δεικτών σε πραγματικό χρόνο.

3.2 Μηχανική Μάθηση

Η Μηχανική Μάθηση (Machine Learning - ML) αποτελεί ένα από τα κυριότερα τμήματα της TX, εστιάζοντας στη σχεδίαση και ανάπτυξη αλγορίθμων που διαθέτουν τη δυνατότητα αυτοβελτίωσης μέσω ανάλυσης και επεξεργασίας εμπειρικών δεδομένων. Ο όρος Μηχανική Μάθηση επινοήθηκε από τον Arthur Samuel το 1959, υπάλληλο της IBM και πρωτοπόρο στον κλάδο των παιχνιδιών για ηλεκτρονικό υπολογιστή. Η MX τοποθετείται στο πεδίο μελέτης που

ερευνά την μελέτη και ανάπτυξη αλγορίθμων μάθησης από είσοδο δεδομένων και να πραγματοποιούν προβλέψεις με αυτά. Πρόκειται για μια κατηγορία της επιστήμης των υπολογιστών και η ανάπτυξη της οφείλεται στην ανάπτυξη της τεχνητής νοημοσύνης. Η ειδοποιός διαφορά της από τον κλασικό προγραμματισμό, έγκειται στην μετατόπιση της ευθύνης ορισμού των κανόνων λειτουργίας. Καθώς στον προγραμματισμό η λογική επιβάλλεται ρητά από τους χρήστες, στην MX το σύστημα λαμβάνει αυτόνομα υποκείμενα πρότυπα, σχέσεις και συσχετίσεις αυτών. Η συνεχή βελτίωση των εσωτερικών παραμέτρων των μοντέλων επιταχύνει αυτή την διαδικασία, συμβάλλοντας έτσι στην μείωση σφάλματος ή κόστους αλλά και στην βελτιστοποίηση της απόδοσής τους.

Στη σύγχρονη επιστημονική βιβλιογραφία η MX θεωρείται ως μια απαραίτητη τεχνολογία για την αντιμετώπιση υψηλής πολυπλοκότητας προβλημάτων όπως η πρόβλεψη χρονοσειρών, η επεξεργασία φυσικής γλώσσας (Natural Language Processing - NLP), η χρηματοοικονομική μοντελοποίηση υπό αβέβαιες συνθήκες κ.α. Η επιστημονική της αξία έγκειται στην αποτελεσματική της ικανότητα προσέγγισης και μοντελοποίησης σύνθετων μη γραμμικών φαινομένων, ακόμα και σε περιπτώσεις όπου οι φυσικοί, οι λειτουργικοί ή οι αιτιακοί μηχανισμοί δεν είναι πλήρως κατανοητοί ή μαθηματικός περιγράψιμοι.

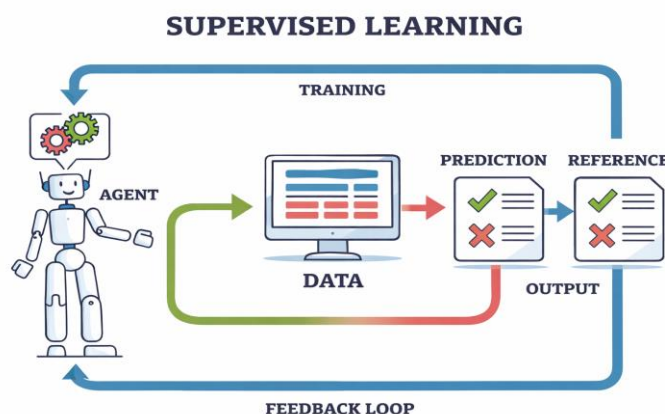
3.2.1 Κατηγορίες Μηχανικής Μάθησης

Η MX διαφοροποιείται σε τρεις βασικές κατηγορίες ανάλογα με το είδος των δεδομένων και τον τρόπο μάθησης του μοντέλου. Αυτές είναι η **Εποπτευόμενη**, η **Μη Εποπτευόμενη** και η **Ενισχυτική Μάθηση**. Κάθε μια από αυτές τις κατηγορίες χρησιμοποιεί διαφορετικούς αλγόριθμους και εξυπηρετεί ξεχωριστούς στόχους.

Εποπτευόμενη Μάθηση (Supervised Learning)

Πρόκειται για την πιο ευρέως διαδεδομένη κατηγορία της MX η οποία στηρίζεται στη χρήση δεδομένων που συνδέονται από γνωστές εξόδους. Κατά την προσέγγιση αυτή, γίνεται εκπαίδευση του μοντέλου με ζεύγη εισόδων και εξόδων, μαθαίνοντας έτσι να συσχετίζει τα χαρακτηριστικά των δεδομένων με την επιθυμητή απάντηση. Στα πλαίσια της εκπαίδευσης του, το μοντέλο συγκρίνει τις προβλέψεις με τις πραγματικές τιμές, υπολογίζει το σφάλμα και

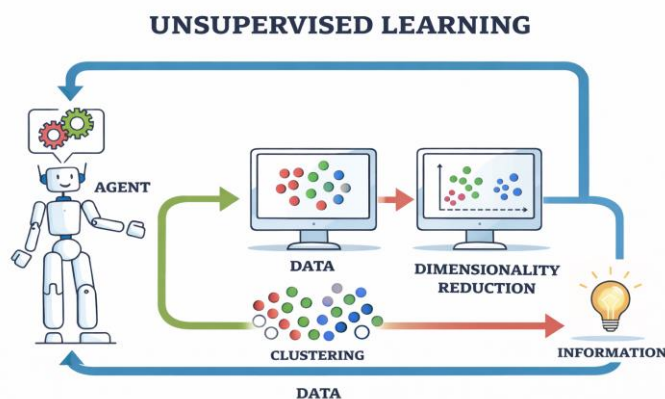
προσαρμόζει τις μεταβλητές τους ώστε να το ελαχιστοποιήσει. Αυτού του είδους μάθησης χρησιμοποιείται τόσο για προβλήματα ταξινόμησης όσο και για προβλήματα παλινδρόμησης, όταν δηλαδή ζητείται η πρόβλεψη συνεχών τιμών ή όταν υπάρχει απαίτηση ανάθεσης μια κατηγορίας σε κάθε δείγμα. Όταν υπάρχουν επαρκή ιστορικά δεδομένα με γνωστές εξόδους, η μάθηση αυτή είναι η καταλληλότερη.



Εικόνα 1 Διάγραμμα Ροής Εποπτευόμενης Μάθησης

Μη Εποπτευόμενη Μάθηση (Unsupervised Learning)

Σε αντίθεση με την Εποπτευόμενη Μάθηση που διατίθενται ετικέτες εξόδου από τα δεδομένα, όταν το σύστημα καλείται να ανακαλύψει μόνο του τη δομή τους, χρησιμοποιείται η Μη Εποπτευόμενη Μάθηση. Σκοπός της είναι η ομαδοποίηση δειγμάτων ως προς την ομοιότητα τους, η αναγνώριση προτύπων, ο εντοπισμός ανωμαλιών ή η μείωση της διάστασης των δεδομένων.

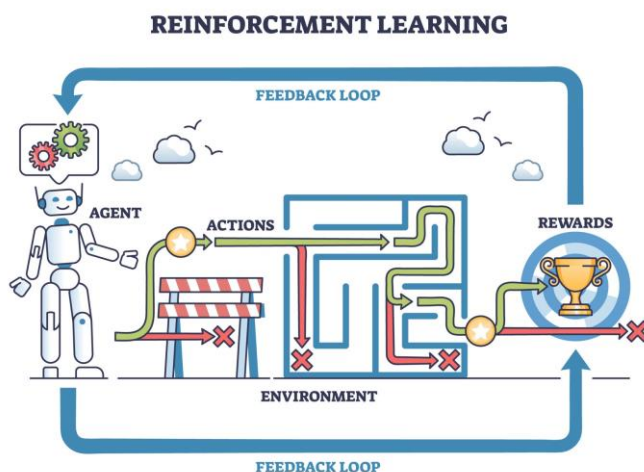


Εικόνα 2 Διάγραμμα Ροής Μη Εποπτευόμενης Μάθησης

Σε αντίθεση με το προηγούμενο μοντέλο μάθησης, εδώ δεν υπάρχει σωστή απάντηση προς σύγκριση με το αποτέλεσμα, συνεπώς γίνεται αξιολόγηση με κριτήρια βασισμένα στη συνοχή και τη διαχωρισιμότητα των ομάδων ή στην ποιότητα της αναπαράστασης. Όταν στοχεύουμε στην εξερεύνηση άγνωστων συνόλων δεδομένων, στον εντοπισμό κρυφών δομών, στην λήψη χρήσιμης πληροφορίας και στην προεπεξεργασία των δεδομένων πριν από εφαρμογή σύνθετων μοντέλων, η Μη Εποπτευόμενη Μάθηση κρίνεται ως η βέλτιστη επιλογή.

Ενισχυτική Μάθηση (Reinforcement Learning)

Τέλος, στην Ενισχυτική Μάθηση το μοντέλο δεν μαθαίνει από σύνολα δεδομένων, αλλά από την αλληλεπίδραση του με ένα περιβάλλον. Στην μέθοδο αυτή ένα σύστημα εκτελεί ενέργειες, μελετά τις συνέπειες τους και δέχεται τυχόν ποινές ή ανταμοιβές, μαθαίνοντας έτσι σταδιακά ποια στρατηγική που θα μεγιστοποιήσει τις ανταμοιβές. Η διαδικασία ουσιαστικά στηρίζεται στη δοκιμή και το λάθος με το σύστημα να ανακαλύπτει την βέλτιστη ακολουθία ενεργειών προκειμένου να οδηγηθεί σε καλύτερα αποτελέσματα.



Εικόνα 3 Διάγραμμα Ροής Ενισχυτικής Μάθησης

Αυτή η μέθοδος μάθησης, χρησιμοποιείται κυρίως σε προβλήματα όπου η λήψη αποφάσεων εξελίσσεται χρονικά για παράδειγμα σε εφαρμογές ρομποτικής, στα αυτόνομα οχήματα και σε συστήματα παιχνιδιών.

3.2.2 Εξαγωγή Χαρακτηριστικών (Feature Engineering)

Ένα από τα σημαντικότερα τμήματα της Μηχανικής Μάθησης είναι η Εξαγωγή Χαρακτηριστικών (Feature Engineering) καθώς καθορίζει συχνά την επιτυχία ή αποτυχία των μοντέλων. Βάση της ακαδημαϊκής βιβλιογραφίας η εξαγωγή χαρακτηριστικών ορίζεται ως η διαδικασία μείωσης της διαστασιμότητας, όπου μια αρχική ομάδα ακατέργαστων μεταβλητών περιορίζεται σε πιο διαχειρίσιμες ομάδες/σύνολα χρησιμοποιηθούν από τους αλγόριθμους μάθησης. Η Μηχανική Μάθηση στηρίζεται σε σημαντικό βαθμό στην παρέμβαση του χρήστη για την επιλογή, σχεδιασμό και μετασχηματισμό των χαρακτηριστικών του μοντέλου, εν αντιθέση με την Βαθιά Μάθηση στην οποία το μοντέλο μαθαίνει αυτόματα τις αναπαραστάσεις των δεδομένων με την χρήση των νευρωνικών δικτύων.

Η εξαγωγή χαρακτηριστικών περιλαμβάνει απλούς στατιστικούς μετασχηματισμούς όπως για παράδειγμα η κανονικοποίηση και η τυποποίηση ή λογαριθμικούς μετασχηματισμούς. Επιπρόσθετα, περιέχει σύνθετες τεχνικές όπως είναι η δημιουργία νέων μεταβλητών αποτύπωσης σχέσεων, αλληλεπιδράσεις ή χρονικές εξαρτήσεις. Τα σύνολα των δεδομένων χωρίζονται σε δύο κατηγορίες, τα Δομημένα και Μη Δομημένα δεδομένα. Στην πρώτη κατηγορία ανήκουν οι συνδυασμοί μεταβλητών προκειμένου να δημιουργηθούν νέες μεταβλητές οι οποίες περιγράφουν ένα φαινόμενο καλύτερα, όπως για παράδειγμα ο υπολογισμός ενός δείκτη από δύο ή περισσότερες μετρήσεις. Στην δεύτερη κατηγορία ανήκουν δεδομένα των οποίων η εξαγωγή είναι πιο σύνθετη όπως είναι τα δεδομένα ήχου, εικόνας και κειμένου. Αυτού του είδους τα δεδομένα κρίνεται απαραίτητο να επεξεργαστούν προκειμένου να έχουν αναγνωρίσιμη δομή και να μπορούν να καταχωρηθούν σε βάσεις δεδομένων.

3.2.3 Κλασικοί Αλγόριθμοι Μηχανικής Μάθησης

Οι κλασικοί αλγόριθμοι της Μηχανικής Μάθησης αποτελούν τον πυρήνα στον οποίο αναπτύχθηκαν οι σύγχρονες τεχνικές μάθησης οι οποίες εξακολουθούν να χρησιμοποιούνται ευρέως σε πληθώρα εφαρμογών. Αυτού του είδους οι αλγόριθμοι παραμένουν ιδιαίτερα αποτελεσματικοί σε προβλήματα με περιορισμένο όγκο δεδομένων και χαμηλές υπολογιστικές απαιτήσεις, παρά την συνεχή και ραγδαία ανάπτυξη των αλγορίθμων της Βαθιάς Μάθησης.

Όλοι οι αλγόριθμοι της Μηχανικής Μάθησης έχουν ως κοινό χαρακτηριστικό τις σχετικά απλές μαθηματικές αρχές οι οποίες απαιτούν προσεκτική επιλογή και επεξεργασία των μεταβλητών εισόδου και των αποτελεσμάτων τους. Συνεπώς η ποιότητα των δεδομένων καθίσταται καθοριστική για την απόδοσή τους.

Οι αλγόριθμοι χωρίζονται στους Αλγόριθμους Επιβλεπόμενης Μάθησης και στους Αλγόριθμους Μη Επιβλεπόμενης Μάθησης. Στην πρώτη κατηγορία ανήκουν οι αλγόριθμοι όπου το σύνολο δεδομένων εκπαίδευσης αποτελείται από δύο ζεύγη εισόδου εξόδου. Οι Μη Επιβλεπόμενης Μάθησης αλγόριθμοι, καλούνται να αναγνωρίσουν μοτίβα σε δεδομένα για τα οποία δεν υπάρχει αντιστοίχιση με ετικέτες.

Βασικοί αλγόριθμοι Επιβλεπόμενης Μάθησης είναι οι εξής:

Γραμμική και Λογιστική Παλινδρόμηση (Linear and Logistic Regression)

Πρόκειται για τους κυριότερους αλγόριθμους για την πρόβλεψη συνεχών τιμών και την ταξινόμηση τους. Η Γραμμική Παλινδρόμηση μοντελοποιεί τη σχέση μεταξύ εξαρτημένης μεταβλητής και ενός συνόλου ανεξάρτητων μεταβλητών μέσω γραμμικής συνάρτησης. Η Λογιστική Παλινδρόμηση χρησιμοποιεί την σιγμοειδή συνάρτηση προκειμένου να χαρτογραφήσει την έξοδο στο διάστημα $[0,1]$.

Μηχανές Διανυσμάτων Υποστήριξης (Support Vector Machines - SVM)

Ο αλγόριθμος ερευνά το βέλτιστο υπερεπίπεδο (hyperplane) που διαχωρίζει τις κλάσεις με το βέλτιστο περιθώριο (margin). Η χρήση πυρήνων (kernels) επιτρέπει σε περιπτώσεις μη γραμμικού διαχωρισμού στις οποίες μεταφέρονται δεδομένα σε χώρους υψηλότερης διαστασιμότητας.

Δέντρα Απόφασης και Τυχαία Δάση (Decision Trees and Random Forests)

Τα Δέντρα Απόφασης είναι κατανοητά και οπτικοποιήσιμα, χρησιμοποιώντας ιεραρχικές δομές κανόνων, βασισμένα σε κριτήρια κέρδους (Entropy, Gini Impurity). Λόγο της ευχρηστίας τους, εφαρμόζονται σε υπερβολικό βαθμό στα μοντέλα εκπαίδευσης. Για να αντιμετωπιστεί η υπερεφαρμογή τους, δημιουργήθηκαν μέθοδοι που συνδυάζουν πολλά δέντρας όπως είναι οι αλγόριθμοι Τυχαίων Δασών. Οι αλγόριθμοι αυτοί βελτιώνουν τη σταθερότητα και την ακρίβεια της ενσωμάτωσης πολλαπλών μοντέλων.

K-Nearest Neighbors (k-NN)

Θεωρείται ένας από τους πιο απλούς αλγόριθμους ο οποίος χαρακτηρίζεται ως μη παραμετρικός (non-parametric) και σκηνρός (lazy learner). Υπολογίζει την Ευκλείδεια απόσταση μεταξύ των σημείων του συνόλου και του νέου δείγματος. Η πρόβλεψη του βασίζεται στην πλειοψηφία των k πλησιέστερων γειτόνων (ταξινόμηση) ή στο μέσο όρο τους (παλινδρόμηση).

Ρηχά Νευρωνικά Δίκτυα (Shallow Neural Networks)

Αν και αποτελούν τμήμα της Βαθιάς Μάθησης, λόγω της περιορισμένης αρχιτεκτονικής τους, θεωρούνται τμήμα της Μηχανικής Μάθησης. Συνήθως απαρτίζονται από ένα επίπεδο εισόδου, ένα κρυφό επίπεδο και το επίπεδο εξόδου. Χρησιμοποιούν τον αλγόριθμο οπισθοδιάδοσης (backpropagation) ώστε να προσαρμόσουν τα βάρη και τις πολώσεις των νευρώνων, ελαχιστοποιώντας έτσι το σφάλμα.

Οι παρακάτω αλγόριθμοι ανήκουν σε αυτούς της Μη Επιβλεπόμενης Μάθησης και είναι οι εξής:

K-Means

Πρόκειται για τον πιο γνωστό αλγόριθμο, οποίος ομαδοποιεί τα δεδομένα βάση την ομοιότητα των χαρακτηριστικών τους. Η λειτουργία του βασίζεται στην επαναληπτική επιλογή των κέντρων των ομάδων, μέχρις ότου επιτευχθεί η σταθερότητα τους. Κρίνεται ιδανικός για γρήγορες επεξεργασίες μεγάλων συνόλων δεδομένων των οποίων ο αριθμός είναι εξ αρχής εκτιμήσιμος. Αν και απλός σαν αλγόριθμος, χρησιμοποιείται ευρέως σε εφαρμογές όπως η τμηματοποίηση και ομαδοποίηση χρονοσειρών.

Density Based Spatial Clustering (DBSCAN)

Εξίσου σημαντικός είναι και ο αλγόριθμος DBSCAN καθώς εντοπίζει τα σύνολα με βάση την πυκνότητα των δεδομένων και όχι την απόστασή τους από κάποιο κέντρο όπως γίνεται με τον k-Means. Εκτός του εντοπισμού περιοχών με αυξημένη συγκέντρωση, μπορεί και να διαχωρίσει τα σημεία που θεωρούνται ως θόρυβος. Με αυτόν τον τρόπο ο αλγόριθμος καθίσταται χρήσιμος σε ακανόνιστες δομές δεδομένων και στην ανίχνευση τυχόν ανωμαλιών.

Ιεραρχική Ομαδοποίηση (Hierarchical Clustering)

Ο αλγόριθμος αυτός ακολουθεί συνήθως μια συσσωρευτικής προσέγγιση καθώς δημιουργεί μια δένδροειδή δομή που απεικονίζει τις σχέσεις μεταξύ των δεδομένων, σε διαφορετικά επίπεδα ανάλυσης. Η ικανότητα της να αναπαριστά σχέσεις σε διαφορετικά επίπεδα, επιτρέποντας στον χρήστη να αποφασίσει για τον τελικό αριθμό ομάδων μετά την ανάλυση της δομής, κρίνεται κύριο πλεονέκτημα αυτής. Οι πιο σύνηθες χρήσεις της είναι στην βιοπληροφορική, στην ανάλυση κειμένου και αναγνώριση προτύπων.

Ανάλυση Κύριων Συνιστωσών (Principal Component Analysis - PCA)

Πρόκειται για αλγόριθμο μείωσης διάστασης, ο οποίος μετασχηματίζει τα δεδομένα σε έναν νέο μικρότερης διάστασης χώρο, διατηρώντας την μέγιστη δυνατή διακύμανση. Ο αλγόριθμος

εντοπίζει τους άξονες που διατηρούν την μέγιστη πληροφορία από τα αρχικά δεδομένα, αναλύοντας τον πίνακα διακύμανσης/συνδιακύμανσης. Διευκολύνοντας την οπτικοποίηση, την μείωση θορύβου και βελτιώνει την απόδοση άλλων αλγορίθμων.

Κανόνες Συσχέτισης (Apriori Algorithm)

Ο αλγόριθμος Apriori χρησιμοποιείται στην εξόρυξη κανόνων συσχέτισης σε ογκώδεις βάσεις δεδομένων. Η αρχή του βασίζεται στο ότι κάθε υποσύνολο ενός συχνού συνόλου στοιχείων πρέπει να είναι εξίσου συχνό. Η μέθοδος επιτρέπει την εξαγωγή στρατηγικών συμπερασμάτων για την σχέση μεταξύ διαφορετικών μεταβλητών.

3.2.4 Πλεονεκτήματα και Περιορισμοί Μηχανικής Μάθησης

Κύριο πλεονέκτημα της Μηχανικής Μάθησης είναι η δυνατότητά της να διαχειρίζεται και να αναλύσει τεράστιους όγκους δεδομένων υψηλής πολυπλοκότητας τους οποίους είναι αδύνατον να επεξεργαστούν αποτελεσματικά από τους χρήστες ή από τις κλασικές μεθόδους στατιστικής. Άλλο ένα βασικό της πλεονέκτημα είναι ικανότητά της να βελτιώνει συνεχώς την απόδοση μαθαίνοντας από νέα δεδομένα, όσο αυξάνεται συνεχώς η διαθέσιμος όγκος και πληροφορία τους. Καθώς εφαρμόζεται σε ένα ευρύ φάσμα προβλημάτων, θεωρείται αρκετά ευέλικτη. Τέλος, πολλά μοντέλα είναι σχετικά απλά με χαμηλές απαιτήσεις σε υπολογιστική ισχύ, το οποίο τα καθιστά κατάλληλα σε εφαρμογές με περιορισμένους πόρους.

Ωστόσο, πέρα από τα πλεονεκτήματα αυτής, απαιτείται να ληφθούν υπόψη και ορισμένοι περιορισμοί. Ο κυριότερος περιορισμός της είναι πως όλα τα μοντέλα της εξαρτώνται από την ποιότητα και την ποσότητα των εισαγόμενων δεδομένων. Η πιθανή ύπαρξη στρέβλωσης και η έλλειψη των δεδομένων, οδηγεί σε λανθασμένα συμπεράσματα και καθιστά τα αποτελέσματα των μοντέλων ανακριβή. Επιπρόσθετα, ορισμένοι αλγόριθμοι είναι πιθανό να προσαρμοστούν σε υψηλό βαθμό ώστε το μοντέλο να μάθει υπερβολικά καλά τα δεδομένα εκπαίδευσης, σε βάρος της γενίκευσης στα νέα δεδομένα. Σε αντίθεση με τους αλγόριθμους της MM τα οποία απαιτούν από τον χρήστη τον προσδιορισμό των σημαντικών μεταβλητών (feature engineering), οι αλγόριθμοι της BM εισάγουν την αυτόνομη μάθηση αναπαραστάσεων

(representation learning) καθιστώντας στο μοντέλο την δυνατότητα να ανακαλύπτει αυτόματα τα κρίσιμα μοτίβα από τα αρχικά και μη δομημένα δεδομένα.

Τα τελευταία χρόνια και λόγω της ραγδαίας ανάπτυξης της υπολογιστικής ισχύος μέσω των υπολογιστικών μονάδων επεξεργασίας γραφικών (GPU), την διαθεσιμότητα και χρήση μεγάλων όγκων δεδομένων (big data) αλλά και την συνεχή ανάπτυξη νέων αλγορίθμων και τεχνικών, δημιουργείται και τεράστια εξέλιξη στον κλάδο της BM.

3.3 Βαθιά Μάθηση (Deep Learning)

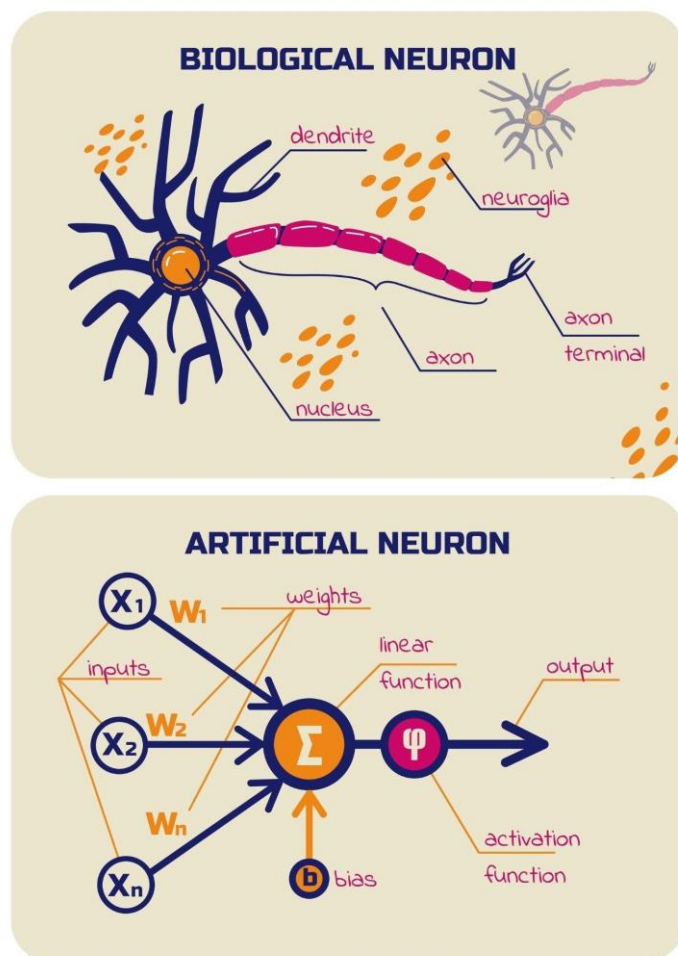
3.3.1 Εισαγωγή στη Βαθιά Μάθηση (Deep Learning)

Ο κλάδος της επιστήμης που καλύπτει την Βαθιά Μάθηση - BM (Deep Learning), είναι ένας από τους πιο αναπτυσσόμενους κλάδους της τεχνικής νοημοσύνης, όντας μια υποεπένδυση της μηχανικής μάθησης ο οποίος είναι βασισμένος στην επιστήμη των τεχνητών νευρωνικών δικτύων. Ο όρος Βαθιά Μάθηση πρωτοεμφανίστηκε το 1986 από την Rina Dechter και σχετίζεται με μια συγκεκριμένη κατηγορία θεωριών σχετικά με την ανάπτυξη του εγκεφάλου. Η κυριότερη διαφορά μεταξύ της BM και της MM έγκειται στην δομή των μοντέλων της, κατά τα οποία χρησιμοποιούνται πολλαπλά μη γραμμικής επεξεργασίας επίπεδα για την εξαγωγή πληροφοριών από τα δεδομένα. Η βασική σκέψη για την υλοποίηση της βασίζεται στο ότι τα δεδομένα μπορούν μέσα από τα διαδοχικά επίπεδα να επεξεργαστούν και να μετασχηματιστούν σταδιακά. Έτσι τα δίκτυα αυτά μπορούν να εντοπίζουν μοτίβα και δομές που δεν θα μπορούσαν να εντοπιστούν με την χρήση άλλων μοντέλων.

3.3.2 Θεωρία Νευρωνικών Δικτύων

Τα Τεχνητά Νευρωνικά Δίκτυα (Artificial Neural Networks - ANN) αποτελούν μια από τις πιο κύριες και επιδραστικές κατηγορίες μοντέλων στη Μηχανική και Βαθιά Μάθηση. Η κεντρική τους ιδέα εμπνέεται από τα αντίστοιχα Νευρωνικά Δίκτυα του ανθρώπινου εγκεφάλου στον οποίο μεγάλο πλήθος των βιολογικών νευρώνων συνεργάζονται για την επεξεργασία

πληροφοριών. Αντίστοιχα, τα Τεχνητά Νευρωνικά Δίκτυα απαρτίζονται από υπολογιστικές μονάδες (τεχνητοί νευρώνες) οι οποίοι οργανώνονται σε επίπεδα και συνδέονται μεταξύ τους μέσω βαρών που καθορίζουν την ένταση και τη σημασία κάθε σύνδεσης. Τα μοντέλα αυτά έχουν την δυνατότητα να χειρίζονται πολύπλοκες μη γραμμικές συναρτήσεις και να μαθαίνουν πρότυπα τα οποία δεν είναι ανιχνεύσιμα με την χρήση των κλασικών μέσων μάθησης.



Εικόνα 4 Βιολογικά και Τεχνητά Νευρωνικά Δίκτυα

Κάθε τεχνητός νευρώνας λαμβάνει σύνολα εισόδων είτε από τους νευρώνες εισόδου είτε από τους νευρώνες επεξεργασίας, τα πολλαπλασιάζει με μια τιμή συσχετισμένη με αυτούς η οποία ονομάζεται βάρος (weight). Στο αποτέλεσμα του πολλαπλασιασμού αυτού εφαρμόζεται μια συνάρτηση ενεργοποίησης η οποία είτε μεταδίδεται στην έξοδο του νευρώνα είτε μεταδίδεται στο επίπεδο επεξεργασία ενός άλλου νευρώνα. Η συνάρτηση αυτή είναι μη γραμμική στα νευρωνικά δίκτυα και ονομάζεται σιγμοειδής συνάρτηση.

$$f(x) = 1 / (1 + e^{-x})$$

Το μοντέλο μαθαίνει πολύπλοκες σχέσεις μεταξύ των δεδομένων χάρις της εφαρμογής της μη γραμμικότητας, έτσι καθίσταται κρίσιμη για τα νευρωνικά δίκτυα. Οι νευρώνες χωρίζονται σε επίπεδα, το επίπεδο εισόδου, ένα ή περισσότερα κρυφά επίπεδα και τέλος το επίπεδο εξόδου. Η πληροφορία μεταφέρεται από το ένα επίπεδο στο επόμενο, δημιουργώντας μια διαδικασία μετασχηματισμού των δεδομένων η οποία όσο προχωράει προς τα επόμενα και βαθύτερα επίπεδα του δικτύου γίνεται και πιο αφηρημένη.

Επί της ουσίας ο κάθε νευρώνας λαμβάνει τα δεδομένα των προηγούμενων νευρώνων επεξεργασμένα με βάση το συναπτικό βάρος (synaptic weight) της σύνδεσης του κάθε νευρώνα. Εν συνεχεία, αθροίζονται όλα τα δεδομένα στο δίκτυο και με βάση την συνάρτηση ενεργοποίησης, διαδίδεται η πληροφορία στα επόμενα επίπεδα. Έστω ότι η x_{ki} είναι η i -οστή είσοδος, το w_{ki} το i -οστό συναπτικό βάρος του νευρώνα k και f η συνάρτηση ενεργοποίησης του δικτύου, τότε για την έξοδο y_k του νευρώνα ισχύει η εξίσωση:

$$y_k = f(\sum_{i=0}^n x_{ki} * w_{ki})$$

Το συναπτικό βάρος που βρίσκεται στον k -οστό νευρώνα ονομάζεται πόλωση ή περιθώριο (bias/threshold). Εάν το άθροισμα της παραπάνω εξίσωσης είναι μεγαλύτερο ή μικρότερο από την τιμή της πόλωσης, τότε η έξοδος κάθε νευρώνα είναι 0 ή 1. Έτσι καθορίζεται πότε ο νευρώνας είναι ανενεργός και πότε ενεργοποιείται. Με αντίστοιχο τρόπο λειτουργούν και τα βιολογικά νευρικά δίκτυα.

3.3.3 Εκπαίδευση Νευρωνικών Δικτύων

Ο μηχανισμός μάθησης των νευρωνικών δικτύων βασίζεται σε επαναληπτική διαδικασία προσπαθώντας το μοντέλο να βελτιώσει την απόκλιση της τιμής εισόδου του με την προβλεπόμενη τιμή. Η διαδικασία μπορεί να χωριστεί σε 3 κύρια μέρη. Αρχικά γίνεται μια εμπρόσθια διάδοση των δεδομένων εισόδου του προς τα επόμενα επίπεδα. Αρχικά, σε κάθε νευρώνα υπολογίζεται το σταθμισμένο άθροισμα των εισόδων και εφαρμόζεται η συνάρτηση

ενεργοποίησης. Η πληροφορία προχωράει μέχρι το επίπεδο εξόδου στο οποίο γίνεται η πρόβλεψη. Αυτή η διάδοση ονομάζεται Εμπρόσθια Διάδοση (Forward Propagation). Εν συνεχεία μόλις ολοκληρωθεί μια πρόβλεψη, το δίκτυο την αξιολογεί με την χρήση της συνάρτησης κόστους ή απώλειας (loss/miss function). Επί της ουσίας, η συνάρτηση εντοπίζει την απόκλιση της πρόβλεψης με την πραγματική τιμή. Μόλις μετρηθεί η απόκλιση, ξεκινάει η τρίτη διαδικασία και με την χρήση του αλγορίθμου Οπισθοδιάδοσης σφάλματος (Backpropagation) υπολογίζεται η κλίση (gradient) της συνάρτησης κόστους ως προς κάθε βάρους του δικτύου, χρησιμοποιώντας τον κανόνα αλυσίδα (chain rule) από τον απειροστικό λογισμό. Επί της ουσίας, το δίκτυο μεταφέρει προς τα πίσω, από την έξοδο προς την είσοδο, το σφάλμα περνώντας το από κάθε σύνδεση. Τέλος, χρησιμοποιείται ένας βελτιστοποιητής (optimizer) ο οποίος αναλαμβάνει να ενημερώσει όλα τα βάρη. Η ενημέρωση γίνεται προς την αντίθετη κατεύθυνση της κλίσης προκειμένου στο επόμενο βήμα η τιμή της απόκλισης να είναι μικρότερη. Πολύ σημαντική είναι η παράμετρος του ρυθμού μάθησης (learning rate), η οποία καθορίζει το μέγεθος του βήματος που εκτελεί το δίκτυο για την ελαχιστοποίηση της απόκλισης. Κατ'αυτόν τον τρόπο, το δίκτυο μαθαίνει να προσαρμόζει τις παραμέτρους του ώστε να ελαχιστοποιείται η απόκλιση δίνοντας σταδιακά μεγαλύτερη ακρίβεια στις προβλέψεις του.

3.3.4 Τύποι Νευρωνικών Δικτύων και Μοντέλα Μνήμης

Αναλόγως τον τύπο του δικτύου αλλάζει και ο τρόπος σύνδεσης αυτού αλλά και ο τρόπος μετάδοσης των δεδομένων μεταξύ των νευρώνων του. Επίσης, στα νευρωνικά δίκτυα χρησιμοποιούνται μοντέλα μνήμης τα οποία είναι αρχιτεκτονικές σχεδιασμένες ώστε το δίκτυο να διατηρεί στο πέρασμα του χρόνου την πληροφορία, να καλεί προηγούμενες καταστάσεις και να τις χρησιμοποιεί ώστε να βελτιώσει τις προβλέψεις του.

Λυαδικός Ταξινομητής Percetron

Το απλούστερο και θεμελιώδες μοντέλο νευρωνικό δίκτυο είναι το Percetron. Αν και είναι απλό σαν δίκτυο, αποτελεί το θεμέλιο της επιστήμης των νευρωνικών δικτύων και της βαθιάς μάθησης. Η λειτουργία του είναι απλή και διαχωρίζει ένα σύνολο δεδομένων σε δύο διακριτές

τιμές 0 και 1, χρησιμοποιώντας μια γραμμική συνάρτηση απόφασης. Όπως αναφέρθηκε και νωρίτερα ο κανόνας λειτουργίας του αναφέρει πως αν ο σταθμισμένος συνδυασμός των εισόδων ξεπερνά την τιμή πόλωσης, το δίκτυο ενεργοποιείται, αν όχι το δίκτυο παραμένει ανενεργό.

Η αλγεβρική αναπαράσταση του μοντέλου είναι:

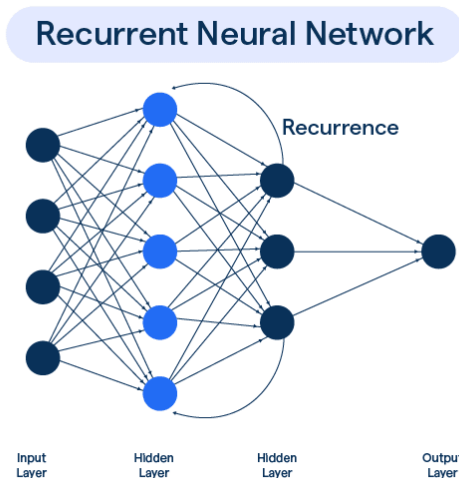
$$y = \sum_{i=1}^n w_i * x_i + b$$

Όπου x το διάνυσμα εισόδου ($x = x_1, x_2, x_3, \dots, x_i$), w το αντίστοιχο βάρος της κάθε εισόδου και b η τιμή της πόλωσης (bias).

Ο αλγόριθμος Perceptron περιορίζεται μόνο σε γραμμικά διαχωρίσιμα δεδομένα καθώς η ανάγκη για επεξεργασία δεδομένων και προβλημάτων που είναι μη γραμμικά, όπως πολύπλοκες σχέσεις μεταξύ χαρακτηριστικών, έδωσε έναυσμα στην ανάπτυξη πολυεπίπεδων νευρωνικών δικτύων.

Αναδρομικά Νευρωνικά Δίκτυα (Recurrent Neural Networks - RNN)

Τα RNN αποτελούν την βασική αρχιτεκτονική στα μοντέλα μνήμης. Διατηρούν σε κάθε βρόχο του δικτύου εσωτερική κατάσταση που ενημερώνεται σε κάθε χρονικό βήμα, επιτρέποντας έτσι το δίκτυο να λαμβάνει υπόψη προηγούμενα δεδομένα. Η έξοδος ενός νευρώνα τροφοδοτείται ξανά σε ένα χρονικά βήμα ως είσοδος του ίδιου νευρώνα και για το επόμενο βήμα, λειτουργώντας έτσι σαν μια βραχυπρόθεσμη μνήμη. Παρόλα αυτά, λόγω του προβλήματος της εξαφάνισης της κλίσης (vanishing gradient), η μνήμη τους είναι περιορισμένη και χάνεται γρήγορα. Κατά το φαινόμενο αυτό οι κλίσεις της συνάρτησης σφάλματος γίνονται υπερβολικά μικρές καθώς διαδίδονται προς τα πίσω, δηλαδή από το επίπεδο εξόδου προς το επίπεδο εισόδου. Για τον λόγο αυτό, τα Αναδρομικά Νευρωνικά Δίκτυα θεωρούνται πρωτόγονα μοντέλα μνήμης.



Εικόνα 5 RNN Δίκτυο

Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης (Long Short Term Memory - LSTM)

Για να επιλυθεί το ζήτημα της βραχυπρόθεσμης μνήμης των RNN, αναπτύχθηκαν τα Δίκτυα Μακράς Βραχυπρόθεσμης Μνήμης - LSTM. Η δομή τους είναι πιο σύνθετη από τα RNN, καθώς χρησιμοποιούν ένα κύτταρο μνήμης (cell state) λειτουργώντας σαν διάδρομος για την μεταφορά πληροφοριών. Διαθέτουν έναν μηχανισμό από πύλες, την πύλη εισόδου (input gate), την πύλη διαγραφής (forget gate) και την πύλη εξόδου (output gate). Με την χρήση των πυλών, τα LSTM ελέγχου ποιές πληροφορίες είναι σημαντικές και θα διατηρηθούν, ποιές θα διαγραφούν και ποιές θα διαδοθούν στην έξοδο. Με τον τρόπο αυτό το δίκτυο “θυμάται” δεδομένα και πληροφορίες από μακρινά χρονικά βήματα.

Μονάδες Αναδρομικών Πυλών (gated Recurrent Units - GRU)

Μια εξέλιξη και πιο απλοποιημένη εκδοχή των LSTM είναι οι Μονάδες Αναδρομικών Πυλών - GRU. Η κύρια διαφορά τους με τα LSTM είναι πως συνδυάζουν την πύλη διαγραφής και την πύλη εισόδου σε μια κοινή, την πύλη ενημέρωσης (update gate). Για την λειτουργία τους απασχολούνται λιγότεροι υπολογιστικοί πόροι και εκπαιδεύονται ταχύτερα λόγω της σχετικά απλής δομής τους.

Μηχανισμοί Προσοχής και Μετασχηματιστές (Attention Mechanisms and Transformers)

Πρόκειται για την εξέλιξη των μοντέλων μνήμης αν και διαφέρουν από τα προηγούμενα καθώς δεν είναι αναδρομικά. Χρησιμοποιείται ένας μηχανισμός προσοχής (self-attention) και δεν συμπίεζουν τα δεδομένα προηγούμενων βημάτων σε ένα διάνυσμα μνήμης. Αντί να επεξεργάζονται ταυτόχρονα όλα τα δεδομένα μιας ακολουθίας, το μοντέλο υπολογίζει βαθμούς σημαντικότητας για κάθε δεδομένο.

Δίκτυα Νευρωνικής Μνήμης (Neural Turing Machines - NTM)

Τέλος, τα Δίκτυα Νευρωνικής Μνήμης είναι μια αρχιτεκτονική που συνδυάζει τα νευρωνικά δίκτυα με εξωτερικές μονάδες μνήμης. Το δίκτυο πέρα από την μάθηση και επεξεργασία δεδομένων καλείται να αποθηκεύει και να διαβάζει τα δεδομένα από την εξωτερική μνήμη. Με τον τρόπο αυτό, γίνεται εφικτό το μοντέλο να εκτελεί πολύπλοκες αλγοριθμικές εργασίες με υψηλές απαιτήσεις σε δομημένη μνήμη.

3.4 Αλγόριθμοι Βελτιστοποίησης (Optimizers)

3.4.1 Εισαγωγή στους Αλγορίθμους Βελτιστοποίησης

Οι Αλγόριθμοι Βελτιστοποίησης είναι ίσως το πιο βασικό κομμάτι στην επιστήμη των Νευρωνικών Δικτύων και της Βαθιάς Μάθησης καθώς αποτελούν το κυριότερο σημείο στην εκπαίδευση του μοντέλου. Κύριο μέλημα των αλγορίθμων αυτών είναι η προσαρμογή των παραμέτρων του μοντέλου και πιο συγκεκριμένα, των βαρών και των περιθωρίων, με τέτοιο τρόπο ώστε να γίνεται η βέλτιστη ελαχιστοποίηση στη συνάρτηση κόστους. Ολόκληρη η διαδικασία θεωρείται αρκετά κρίσιμη, καθώς στα πλαίσια αυτής οριοθετείται η σύγκληση, η σταθερότητα στην εκπαίδευση του δικτύου και συνολικά την απόδοση του μοντέλου. Είναι αδύνατον για οποιοδήποτε είδος νευρωνικού δικτύου, να μάθει είτε πολύπλοκα είτε απλά πρότυπα, είτε ακόμα και να γενικευτούν σωστά τα νέα δεδομένα χωρίς να γίνει χρήση αυτών των αλγορίθμων.

Η εκπαίδευση και η μάθηση των νευρωνικών δικτύων, βασίζεται στη χρήση των παραγώγων της συνάρτησης κόστους (loss function), ως προς τις παραμέτρους του εκάστοτε μοντέλου. Κατά την διάρκεια της εκπαίδευσης, το μοντέλο παράγει μια πρόγνωση της οποίας η τιμή, με την χρήση της συνάρτησης κόστους, συγκρίνεται με την πραγματική τιμή. Ο αλγόριθμος παίρνει την πληροφορία για την τιμή και το μέγεθος του σφάλματος κρίνοντας πως πρέπει να τροποποιηθούν τα βάρη ώστε από την επόμενη επανάληψη, να υπάρχει μείωση στο σφάλμα.

Οι αλγόριθμοι βελτιστοποίησης τροποποιούν τόσο την ταχύτητα εκπαίδευσης των μοντέλων όσο και την ποιότητα της τελικής τους λύσης. Η επιλογή ενός ακατάλληλου αλγορίθμου για το νευρωνικό δίκτυο, μπορεί να οδηγήσει σε με επαρκή και ασταθή εκπαίδευση, υπερ εφαρμογή των τιμών πρόγνωσης και αστοχία στις ελαχιστοποιήσεις. Εν αντιθέση, η χρήση του κατάλληλου αλγορίθμου, επιταχύνει την σύγκλιση και βελτιώνει την αποδοτικότητα του μοντέλου. Για τους πάρα πάνω λόγους, η κατανόηση της λειτουργίας, των βασικών αρχών και διαφορών των αλγορίθμων βελτιστοποίησης, αποτελεί το πιο κρίσιμο σημείο για την επιλογή αυτού προτού εφαρμοστεί στο μοντέλο, ώστε να έχουμε την βέλτιστη και επιτυχημένη ανάπτυξη του νευρωνικού δικτύου.

Ένας από τους πιο απλούς και πρώτους αλγορίθμους βελτιστοποίησης είναι ο κατιόντας κατάβασης (gradient descent). Ο συγκεκριμένος αλγόριθμος ενημερώνει τα βάρη για την κατεύθυνση και την μέγιστη ελαχιστοποίηση της τιμής του σφάλματος. Παρόλα αυτά, ο αλγόριθμος gradient descent, εμφανίζει σημαντικά μειονεκτήματα, για παράδειγμα η αργή του σύγκλιση, η ευαισθησία ως προς την επιλογή του ρυθμού μάθησης και η δυσκολία αντιμετώπισης πολύπλοκων τιμών κόστους.

Προκειμένου να αντιμετωπιστούν αποδοτικότερα όλα τα ζητήματα των μοντέλων και των αλγορίθμων μάθησης, έχουν αναπτυχθεί αρκετοί αλγόριθμοι οι οποίοι βελτιώνουν την απόδοση της εκπαίδευσης των μοντέλων με την χρήση τεχνικών όπως η προσαρμοστική ρύθμιση του ρυθμού μάθησης, την χρήση ορμής (momentum), την εξομάλυνση στις διακυμάνσεις των παραγώγων τιμών αλλά και την αξιοποίηση προηγούμενων και ιστορικών τιμών και πληροφοριών. Μερικά παραδείγματα αλγορίθμων είναι οι SGD, RMSProp, Adam, Adagrad κ.α. ο οποίοι αποτελούν τα βασικότερα εργαλεία στην επιστήμη των Νευρωνικών

Δικτύων και Βαθιάς Μάθησης καθώς επιτρέπουν στον βέλτιστο βαθμό την εκπαίδευση πολύπλοκων αρχιτεκτονικών.

3.4.2 Κατηγορίες και Τύποι Αλγορίθμων Βελτιστοποίησης

Οι αλγόριθμοι βελτιστοποίησης, κατηγοριοποιούνται σε τρεις κύριες κατηγορίες. Στην πρώτη και πιο βασική κατηγορία, είναι οι τύποι αλγορίθμων που αποτελούν Βασικά Μοντέλα Βελτιστοποίησης (Gradient Descent Family). Επόμενη κατηγορία είναι τα Μοντέλα Βελτιστοποίησης με Ορμή (Momentum-Based Optimizers) και τελευταία κατηγορία είναι τα Προσαρμοστικά Μοντέλα Βελτιστοποίησης (Adaptive Learning Rate Optimizers).

Βασικά Μοντέλα Βελτιστοποίησης (Gradient Descent Family)

Πρόκειται για την πρώτη και θεμελιώδη κατηγορία των αλγορίθμων βελτιστοποίησης η οποία βασίζεται στον αλγόριθμο, στον οποίο έχουμε αναφερθεί νωρίτερα, τον Gradient Descent. Η βασική σκέψη και πράξη των αλγορίθμων αυτών είναι ότι τα βάρη του δικτύου, ενημερώνονται ως προς την κατεύθυνση της μέγιστης ελαχιστοποίησης της συνάρτησης κόστους. Τρεις από τους κυριότερους τύπους στην κατηγορία αυτή είναι οι Batch, Stochastic και Mini-Batch Gradient Descent.

Στον τύπο Batch Gradient Descent ο αλγόριθμος, υπολογίζει την ολίσθηση (gradient) χρησιμοποιώντας ολόκληρο το σύνολο δεδομένων με αποτέλεσμα να γίνεται σταθερή αλλά ταυτόχρονα και αργή σύγκλιση, ιδίως όταν υπάρχει μεγάλος όγκος εισαγόμενων δεδομένων.

Ο δεύτερος τύπος ο Stochastic Gradient Descent, ενημερώνει τις τιμές των βαρών με την χρήση ενός δείγματος την φορά. Αφενός είναι πιο γρήγορος συγκριτικά με τον προηγούμενο τύπο, αλλά μπορεί να προκαλέσει θόρυβο στις ενημερώσεις τιμών.

Τέλος, ο τρίτος τύπος είναι ο Mini-Batch Gradient Descent ο οποίος χωρίζει τα δεδομένα σε μικρά σύνολα, επιτυγχάνοντας έτσι σταθερότητα αλλά και ταχύτητα στην εκπαίδευση του μοντέλου. Οι σύγχρονες GPU χρησιμοποιούν αποτελεσματικά στις λειτουργίες τους, αυτού του τύπου αλγορίθμους.

Μοντέλα Βελτιστοποίησης με Ορμή (Momentum - Based Optimizers)

Σε αυτή την κατηγορία, περιλαμβάνονται αλγόριθμοι ο οποίοι ενσωματώνουν την έννοια της ορμής (momentum) στην διαδικασία ενημέρωσης των βαρών. Σε αυτή τη κατηγορία ο αλγόριθμος χρησιμοποιεί τις τιμές από προηγούμενα ιστορικά gradients προκειμένου να γίνει η πρόβλεψη και να ελαχιστοποιηθεί το σφάλμα.

Κύριος τύπος της κατηγορίας αυτής είναι ο Standard Momentum στον οποίο αν η ολίσθηση (gradient) είναι κινείται με σταθερό ρυθμό προς μια κατεύθυνση, η σύγκλιση επιτυγχάνεται μέσω της ορμής (momentum). Ενώ σε αντίθετη περίπτωση, όταν υπάρχει αστάθεια στον ρυθμό της ολίσθησης αλλάζοντας αρκετά συχνά κατεύθυνση, η ορμή βοηθά στην μείωση των ταλαντώσεων.

Ένας εξελιγμένος αλγόριθμος είναι ο Nesterov Accelerated Gradient, ο οποίος προσπαθεί να προβλέψει την ολίσθηση όχι ως προς το παρόν σημείο, αλλά σε μια προβλεπόμενη τιμή με βάση την κίνηση που πραγματοποιεί το μοντέλο.

Προσαρμοστικά Μοντέλα Βελτιστοποίησης (Adaptive Learning Rate Optimizers)

Πρόκειται για την πιο σύγχρονη κατηγορία στην επιστήμη των Νευρωνικών Δικτύων. Οι αλγόριθμοι της κατηγορίας αυτής, προσαρμόζουν δυναμικά βάση κάθε τιμής των παραμέτρων του μοντέλου, τον ρυθμό μάθησης του.

Ο πρώτος αλγόριθμος ο οποίος εισήγαγε την σκέψη ότι η κάθε τιμή της παραμέτρου είναι πιθανό να έχει τον δικό της ρυθμό μάθησης, ο οποίος μειώνεται σταδιακά όσο αυξάνεται η λαμβανόμενη πληροφορία από τις ολισθήσεις. Αν και είναι αρκετά αποτελεσματικός ακόμα και αν τα δεδομένα είναι αραιά, μειονεκτεί στο γεγονός ότι ο ρυθμός μάθησης μειώνεται αρκετά γρήγορα.

Ο αλγόριθμος RMSProp αναπτύχθηκε προκειμένου να ξεπεραστεί το μειονέκτημα του αλγορίθμου Adagrad. Έτσι, ο RMSProp χρησιμοποιεί εκθετική εξομάλυνση των τετραγώνων τιμών των ολισθήσεων διατηρώντας ταυτόχρονα έναν σταθερό και λειτουργικό ρυθμό μάθησης.

Σε αυτή την κατηγορία ανήκει και ο πιο γνωστός αλγόριθμος ο Adam. Ο αλγόριθμος αυτός συνδυάζει τα πλεονεκτήματα του αλγορίθμου Momentum και του RMSProp. Κατά την λειτουργία του Adam, υπολογίζεται τόσο ο μέσος όρος των ολισθήσεων, όσο και ο μέσος όρος των τιμών των τετραγώνων τους. Έτσι καθίσταται εφικτό η γρήγορη και σταθερή σύγκλιση σε μια μεγάλη πληθώρα προβλημάτων.

Τέλος, έχει αναπτυχθεί και μια βελτιωμένη έκδοση του αλγορίθμου Adam, ο αλγόριθμος AdamW, ο οποίος βελτιστοποιεί και διορθώνει τη χρήση και εφαρμογή της απόσβεσης των βαρών (weight decay), προκαλώντας έτσι καλύτερη γενίκευση και περισσότερο αξιόπιστη εκπαίδευση και σε μεγάλα μοντέλα.

4. Python



Εικόνα 6 Λογότυπο Γλώσσας Python

4.1 Εισαγωγή Κεφαλαίου

Στο παρόν κεφάλαιο γίνεται αναφορά στην γλώσσα προγραμματισμού Python, η οποία χρησιμοποιήθηκε για την ανάπτυξη της κωδικοποίησης χρονοσειράς και εκπαίδευσης μοντέλων καιρικής πρόγνωσης. Η συγγραφή πραγματοποιήθηκε στο Google Colab με χρήση της γλώσσας Python σε συνδυασμό με επιπλέον βιβλιοθήκες όπως Pandas, NumPy, SCikit-learn και TensorFlow Object Detection.

4.2 Εισαγωγή Python

Η γλώσσα προγραμματισμού Python είναι μια από τις πλέον δημοφιλές και ευέλικτες στο χώρο των ηλεκτρονικών υπολογιστών. Κατηγοριοποιείται στις γλώσσες προστακτικού προγραμματισμού και υποστηρίζει τον διαδικαστικό και τον αντικειμενοστραφή προγραμματισμό. Σχεδιάστηκε από τον Ολλανδό Guido van Rossum το 1981 ενώ κυκλοφόρησε το 1991. Είναι αρκετά απλή στη χρήση της και ευανάγνωστη καθιστώντας την κατάλληλη για να υλοποιηθούν σύνθετες ιδέες με λιγότερες γραμμές κώδικα σε σύγκριση με άλλες γλώσσες προγραμματισμού. Καθώς είναι γλώσσα γενικής χρήσης, έχει κυριαρχήσει σε διάφορους τομείς όπως η ανάπτυξη ιστοσελίδων, η ανάλυση δεδομένων και η χρήση τεχνητής νοημοσύνης κ.α.

Επιπρόσθετα, είναι ελεύθερη προς χρήση για κάθε προγραμματιστή και χρησιμοποιείται σε όλα τα λειτουργικά συστήματα (Windows, Linux και Mac OS X). Είναι ανοιχτού κώδικα (open source) λογισμικό, επεκτάσιμη και μπορεί να αλληλεπιδράσει με διάφορες άλλες γλώσσες προγραμματισμού και τέλος διαθέτει εύχρηστο περιβάλλον ανάπτυξης κώδικα με ονομασία IDLE.

Ένα από τα πλεονεκτήματα της είναι το τεράστιο οικοσύστημα από βιβλιοθήκες και πακέτα που επιτρέπουν στους χρήστες να υλοποιούν εφαρμογές χωρίς να χειρίζονται οι ίδιοι χαμηλού επιπέδου λεπτομέρειες όπως είναι τα πρωτόκολλα, sockets ή λοιπές διεργασίες.

4.3 Βιβλιοθήκες

Οι βιβλιοθήκες που χρησιμοποιήθηκαν για την ολοκλήρωση της παρούσας εργασίας είναι οι εξής:

- Pandas: χρησιμοποιήθηκε για ανάλυση και χειρισμό δεδομένων.
- NumPy (Numerical Python): χρησιμοποιήθηκε για την αριθμητική ανάλυση.
- Matplotlib: χρησιμοποιήθηκε για την δημιουργία γραφικών παραστάσεων και διαγραμμάτων.
- SCI-Kit Learn: Χρησιμοποιήθηκε για την προεπεξεργασία δεδομένων, επιλογή χαρακτηριστικών, εκπαίδευση μοντέλων και αξιολόγηση αυτών.
- Tensorflow: πρόκειται για πλήθος βιβλιοθηκών για την ανάπτυξη εφαρμογών μηχανικής μάθησης.

4.3.1 Pandas

Η βιβλιοθήκη Pandas είναι μια από τις κυριότερες βιβλιοθήκες της Python. Πρόκειται για μια γρήγορη και ευέλικτη βιβλιοθήκη που βοηθά στην ανάλυση και χειρισμό δομημένων δεδομένων. Επιτρέπει τον καθαρισμό, μετασχηματισμό, την ομαδοποίηση, συγχώνευση και ανάλυση των δεδομένων με τρόπους παρόμοιους με του Excel ή της SQL. Υποστηρίζει την φόρτωση δεδομένων από πλήθος αρχείων και μορφών όπως για παράδειγμα CSV, Excel, SQL κ.α. Περιλαμβάνει εργαλεία για την διαχείριση ελλিপών τιμών, μετατροπές τύπων, pivot tables, resampling χρονοσειρών και περιέχει ενσωματωμένο plotting. Αποτελεί θεμέλιο κομμάτι της

Python για data science και machine learning καθώς μπορεί να ενσωματωθεί με άλλες βιβλιοθήκες όπως οι NumPy, Matplotlib και Sci-Kit learn.

4.3.2 NumPy

Η βιβλιοθήκη αυτή είναι η βασικότερη βιβλιοθήκη της Python για αριθμητικούς υπολογισμούς. Παρέχει γρήγορο και αποδοτικό τρόπο εργασίας με πολυδιάστατους πίνακες (ndarrays), στους οποίους βασίζονται σχεδόν όλες οι υπόλοιπες επιστημονικές βιβλιοθήκες. Προσφέρει πλήθος εργαλείων για γραμμική άλγεβρα, στατιστικές πράξεις, broadcasting, διανυσματικοποιημένους πίνακες καθώς και γεννήτριες τυχαίων αριθμών. Προκειμένου να επιτευχθεί η μέγιστη ταχύτητα, η βιβλιοθήκη έχει υλοποιηθεί σε κώδικα βελτιστοποιημένο C.

4.3.3 Matplotlib

Η Matplotlib είναι η κυριότερη βιβλιοθήκη της Python για οπτικοποίηση και χρησιμοποιείται στη δημιουργία στατικών, διαδραστικών και animated γραφημάτων. Προσφέρει ένα μεγάλο φάσμα διαγραμμάτων όπως γραμμικά, ράβδων, scatter, heatmaps, 3D plots κ.α. Δίνει τον πλήρη έλεγχο κάθε στοιχείου του διαγράμματος, όπως άξονες, χρώματα και ετικέτες (labels). Η βιβλιοθήκη αυτή οπτικοποιεί τα δεδομένα από πίνακες και data frames τα οποία προηγούμενος είχαν επεξεργαστεί από άλλες βιβλιοθήκες όπως Pandas και NumPy.

4.3.4 Sci-Kit Learn

Πρόκειται για μια από τις σημαντικές και διαδεδομένες βιβλιοθήκες ανοιχτού κώδικα σχετικά με την μηχανική μάθηση στην γλώσσα Python. Χρησιμοποιείται για την κανονικοποίηση και προεπεξεργασία των δεδομένων και χρονοσειρών, την εκπαίδευση μοντέλων κ.α. Είναι γραμμένη σε γλώσσες Python, C και C++. Η ανάπτυξη της στηρίχθηκε στις βιβλιοθήκες NumPy, SciPy και Matplotlib. Περιέχει μοντέλα προεπεξεργασίας δεδομένων (scaling, encoding, feature selection, feature extraction), μοντέλα μηχανικής μάθησης Classification (SVM, Random Forest, Logistic Regression), Regression (Ridge, Random Forest, Gradient Boosting και Clustering (k-Means, hierarchical clustering) Pipelines για ολοκληρωμένες ροές

προεπεξεργασίας και μοντέλου κ.α. Στην εργασία χρησιμοποιήθηκαν για την προεπεξεργασία των χρονοσειρών τα StandarScaler, MinMaxScaler και RobustScaler.

4.3.5 Tensorflow

Η Tensorflow είναι ανοιχτού κώδικα (open source) βιβλιοθήκη η οποία αναπτύχθηκε από την ομάδα Google Brain. Είναι η σημαντικότερη βιβλιοθήκη για machine και deep learning, επιτρέποντας τον σχεδιασμό, την εκπαίδευση και την ανάπτυξη νευρωνικών δικτύων. Το όνομα TensorFlow προέρχεται από τις λειτουργίες που εκτελούν νευρωνικά δίκτυα σε πολυδιάστατους πίνακες δεδομένων, οι οποίοι αναφέρονται ως τανυστές (tensors). Μπορεί να χρησιμοποιηθεί σε περισσότερες γλώσσες προγραμματισμού εκτός της Python, όπως για παράδειγμα οι γλώσσες C++ και Java. Επιπρόσθετα υποστηρίζονται εργαλεία για την οπτικοποίηση και παρακολούθηση της εκπαίδευσης των μοντέλων, όπως είναι το TensorBoard, καθώς και μηχανισμούς ανάπτυξης μοντέλων σε παραγωγικά συστήματα μέσω του TensorFlow Serving.

Προσφέρει προηγμένες δυνατότητες για την αυτοματοποιημένη παραγωγή του διανύσματος κλίσης (gradient vector), επιτρέποντας τον ακριβή υπολογισμό των μεταβολών σε κάθε παράμετρο του μοντέλου κατά τη διαδικασία της εκπαίδευσης. Παράλληλα, διαθέτει ένα ευρύ φάσμα ενσωματωμένων συναρτήσεων για την αξιολόγηση της απόδοσης, διευκολύνοντας τον προσδιορισμό κρίσιμων μεγεθών, όπως οι μετρικές ακρίβειας και οι τιμές των συναρτήσεων απώλειας (loss functions).

Ιδιαίτερης σημασίας είναι η παροχή εξειδικευμένων αλγορίθμων βελτιστοποίησης (optimizers), όπως οι ADAM, ADAGRAD και ο Gradient Descent (SGD). Οι συγκεκριμένοι μηχανισμοί βελτιστοποίησης προσφέρουν διαφορετικές μεθοδολογίες για τη ρύθμιση των εσωτερικών παραμέτρων του δικτύου, επηρεάζοντας άμεσα την ταχύτητα σύγκλισης και τη συνολική αποτελεσματικότητα του μοντέλου.

5. Σύστημα Καταγραφής Δεδομένων

5.1 Εισαγωγή Κεφαλαίου

Τα πρωτογενή δεδομένα της παρούσας μελέτης λήφθηκαν από τα αρχεία του Εθνικού Αστεροσκοπείου Αθηνών (ΕΑΑ). Σε αυτό το κεφάλαιο παρατίθεται ο εξοπλισμός που χρησιμοποιεί η υπηρεσία για την καταγραφή των παραμέτρων. Το 2006 ξεκίνησε η δημιουργία δικτύου παρατήρησης καιρού της ελληνικής επικράτειας από το Ινστιτούτο Περιβαλλοντικών Ερευνών και Βιώσιμης Ανάπτυξης του Εθνικού Αστεροσκοπείου Αθηνών (ΙΕΠΒΑ/ΕΑΑ), υπό την ονομασία NOAAN (National Observatory of Athens Automative Network).

Ο τύπος μετεωρολογικού σταθμού που χρησιμοποιείται στο δίκτυο είναι ο Davis Vantage Pro 2 και οι παράμετροι καταγραφής είναι θερμοκρασίας αέρας περιβάλλοντος, σχετική υγρασία, ταχύτητα και διεύθυνση ανέμου, βροχόπτωση, ατμοσφαιρική πίεση, ηλιακή και υπεριώδη ακτινοβολία. Με γνώμονα τη διασφάλιση της αξιοπιστίας των προγνωστικών μοντέλων και την ακρίβεια των μετρήσεων, εφαρμόζεται πρωτόκολλο ετήσιας συντήρησης. Αυτή περιλαμβάνει τεχνικό έλεγχο, βαθμονόμηση αισθητήρων και προληπτική τους αντικατάσταση διασφαλίζοντας ότι οι καταγραφές παραμένουν εντός του επιτρεπτού εύρους σφάλματος.

Όλοι οι αισθητήρες μπορούν να καταγράψουν σε διάφορες μονάδες όπως για παράδειγμα τα Volt, Ampere, Ohm κ.α. Δεν είναι εφικτό το κάθε αισθητήριο να καταγράψει ακριβώς την τιμή για το δεδομένο που χρειαζόμαστε. Για παράδειγμα ένας αισθητήρας καταγραφής της ταχύτητας ανέμου, με την χρήση οπών και ενός οπτικού πομποδέκτη, καταγράφει σε μονάδες Hz. Η μέτρηση φτάνει σε έναν επεξεργαστή στον οποίο έχει ρυθμιστεί μια αντιστοιχία των Hz μέτρησης σε αναλογία με μονάδες μέτρησης της ταχύτητας ανέμου, για παράδειγμα m/s. Συνεπώς είναι κομβικής σημασίας η αντιστοιχία μεταξύ των μονάδων αυτών να είναι η σωστή και να μην υπάρχουν αποκλίσεις. Πρέπει να σημειωθεί στο σημείο αυτό, πως οι αισθητήρες φθείρονται με την πάροδο του χρόνου από την διαρκή χρήση τους, είτε αλλιώνονται καθώς εκτίθενται συνεχώς σε εξωτερικούς χώρους. Αυτές οι φθορές μπορούν να προκαλέσουν αστοχία στην μέτρηση και να διαταράξουν την ομαλή λειτουργία του. Για τον λόγο αυτό στου αισθητήρες εκτελείται διαδικασία βαθμονόμησης, ανά χρονικά διαστήματα τα οποία ορίζονται από την εκάστοτε κατασκευάστρια εταιρία. Κατά την διαδικασία βαθμονόμησης, ελέγχεται η

μέτρηση του αισθητηρίου και συγκρίνεται με την μέτρηση πρότυπου αισθητηρίου προκειμένου να εντοπιστεί τυχόν απόκλιση στην τιμή καταγραφής. Με τον τρόπο αυτό πιστοποιείται η ορθή καταγραφή και ομαλή λειτουργία του αισθητήρα. Οι τιμές στις καταγραφές που γίνονται από αισθητήρες στους οποίους έχει περάσει η ημερομηνία ελέγχου, συντήρησης και βαθμονόμησης τους, είναι αμφιβόλου ποιότητας και πολύ πιθανόν να οδηγήσουν σε άστοχα συμπεράσματα.

5.2 Αισθητήρες Σταθμού

Ανεμόμετρο

Χρησιμοποιείται το μοντέλο Vantage Pro2 (6410), πρόκειται για συνδυασμό μηχανικών αισθητήρων για την καταγραφή της κατεύθυνσης και ταχύτητας του ανέμου. Πρόκειται για μηχανικό αισθητήρα και η καταγραφή της κατεύθυνσης ανέμου γίνεται με έναν ανεμοδείκτη με ποτενσιόμετρο, πτερύγιο που ευθυγραμμίζεται πάντα με τη φορά του ανέμου με δυνατότητα περιστροφής 0-360° και ακρίβεια $\pm 3^\circ$. Ο ανεμοδείκτης είναι τοποθετημένος σε έναν ρότορα ο οποίος με την κίνηση του τροποποιεί την τιμή του ποτενσιόμετρου. Το αισθητήριο στέλνει μια τιμή μέτρησης σε μονάδες Ohm και ο ελεγκτής αντιστοιχεί την τιμή αυτή σε μοίρες ώστε να προσδιοριστεί σωστά η κατεύθυνση του ανέμου. Η ορθή τοποθέτηση του αισθητηρίου είναι κομβικής σημασίας στην μέτρηση του δείκτη ανέμου, καθώς ο βορράς φέρει την μέτρηση της μηδενικής μοίρας. Έτσι μια, έστω και μικρή, απόκλιση στην ευθυγραμμισμένη ως προς τον βορρά τοποθέτηση του αισθητήρα, μπορεί να επιφέρει λανθασμένες μετρήσεις. Εν αντιθέση με τον αισθητήρα καταγραφής της ταχύτητας του αέρα, ο οποίος δεν επηρεάζεται από την προσανατολισμένη τοποθέτηση. Ο αέρας κινεί τα ανεμοκύπελα του αισθητηρίου οι οποίες είναι τοποθετημένες σε έναν ρότορα στον οποίο υπάρχουν οπές και με την χρήση ενός οπτικού πομποδέκτη, γίνεται καταγραφή μιας μέτρησης σε μονάδες Hz. Η τιμή αυτή στέλνεται στον ελεγκτή ο οποίος την μετατρέπει σε μονάδες μέτρησης είτε m/s είτε km/h.



Εικόνα 7 Ανεμόμετρο Vantage Pro2 (6410)

Το εύρος μέτρησης της ταχύτητας ανέμου είναι 1–67 m/s, 3–241 km/h με ακρίβεια $\pm$ μέγιστο (5%, 3 km/h/1 m/s) για μεγάλα ανεμοκύπελλα είτε 1,5–79 m/s, 5–282 km/h με ακρίβεια $\pm$ μέγιστο (5%, 5 km/h/1,5 m/s) για μικρά ανεμοκύπελλα.

Θερμοκρασία - Σχετική υγρασία

Όπως και στην προηγούμενη περίπτωση έτσι και σε αυτόν τον αισθητήρα, έχουμε διπλή καταγραφή της θερμοκρασίας και της σχετικής υγρασίας.



Εικόνα 8 Αισθητήρας Θερμοκρασίας - Σχετικής Υγρασίας

Για την καλύτερη λειτουργία και προστασία τους, οι αισθητήρες είναι τοποθετημένοι μέσα σε μια συσκευή ασπίδας ακτινοβολίας με αναρρόφηση ανεμιστήρα.

Ο αισθητήρας αποτελείται από μια δίοδο πυριτίου σε σύνδεση PN. Η δίοδος διαρρέεται από σταθερό ρεύμα, καθώς η θερμοκρασία του περιβάλλοντος μεταβάλλεται και αλλάζει η αντίσταση του πυριτίου, που είναι και το υλικό κατασκευής της διόδου. Έτσι με την διαφορά στην τιμή του ρεύματος, πραγματοποιείται και η καταγραφή της μέτρησης. Το εύρος μέτρησης είναι από -40 έως $+65$ °C με ακρίβεια $\pm 0,3$ °C, $+15,6$ – $+37,8$ °C. Η καταγραφή της σχετικής υγρασίας γίνεται με την χρήση ενός στοιχείου πυκνωτή φιλμ (Film Capacitor). Αποτελείται ουσιαστικά από ένα λεπτό στρώμα (φιλμ) πολυμερούς υλικού που βρίσκεται ανάμεσα σε δύο μεταλλικά ηλεκτρόδια. Το φιλμ λειτουργεί ως “διηλεκτρικό”, απορροφώντας μόρια νερού τροποποιείται και η χωρητικότητα του πυκνωτή. Το εύρος μέτρησης είναι 0–100% με ακρίβεια $\pm 3\%$, 0–90%.

Βροχόμετρο

Πρόκειται για μηχανικό αισθητήρα, στην βάση του οποίου είναι τοποθετημένο ένα “κουτάλι”. Το “κουτάλι” είναι σχεδιασμένο να παραμένει σε οριζόντια θέση όταν είναι άδειο, ενώ όταν μαζευτεί μια συγκεκριμένη ποσότητα νερού σε αυτό, μετατοπίζεται.



Εικόνα 9 Βροχόμετρο

Με τη χρήση ενός μαγνήτη, παράγεται ένας παλμός σε κάθε μετατόπιση του “κουταλιού” σε μονάδα μέτρησης Hz. Όπως και στους προηγούμενους αισθητήρες η τιμή στέλνεται στον επεξεργαστή ο οποίος κάτι την αντιστοίχιση σε μονάδες καταγραφής της βροχόπτωσης. Βάση της ευρωπαϊκής σταθεράς το “κουτάλι” δέχεται βάρος που αντιστοιχεί σε 0,2mm βροχής. Με τον τρόπο αυτό καταγράφεται η συνολική βροχή (Ac_R). Ταυτόχρονα γίνεται και καταγραφή της έντασης της βροχής (Rain Rate) με την χρήση ενός εσωτερικού χρονομέτρου στον σταθμό, όπου υπολογίζεται η συχνότητα λήψης των παλμών του αισθητήρα. Το εύρος καταγραφής του αισθητήρα είναι 0–100 mm/ώρα με ακρίβεια $\pm$ μέγιστο (3%, 0,2 mm), για ρυθμούς βροχής έως 50 mm/h.

Ατμοσφαιρική Πίεση

Ο αισθητήρας που χρησιμοποιείται είναι πιεζοαντιστατικό αισθητήρας πίεσης πυριτίου και είναι τοποθετημένος στο εσωτερικό της κονσόλας. Στο εσωτερικό του αισθητήρα υπάρχει ένα λεπτό διάφραγμα από πυρίτιο, καθώς τροποποιείται η ατμοσφαιρική πίεση κάμπτεται ελαφρώς το διάφραγμα τροποποιώντας έτσι την τιμή της καταγραφής του. Με τον τρόπο αυτό μεταβάλλεται η τιμή της αντίστασης και γίνεται η καταγραφή της πίεσης. Έχει εύρος μέτρησης 540–1100 hPa με ακρίβεια $\pm$ 1,0 hPa.

5.3 Εμφάνιση και Αποθήκευση Δεδομένων

Οι σταθμοί που είναι τοποθετημένοι σε χώρους ιδιοκτησία του ελληνικού δημοσίου, όπως τα σχολεία, νοσοκομεία ή δημόσια κτίρια, αποστέλλουν τα δεδομένα διαδικτυακά μέσω της δημόσιας σύνδεσης “Syzefxis”. Ταυτόχρονα υπάρχει και ένας μικρός όγκος δεδομένων ο οποίος λαμβάνεται από ιδιώτες, οι οποίοι οφείλουν να παρέχουν τις κατάλληλες συνθήκες για την συνεχή ροή δεδομένων. Η μεταφορά όλων των δεδομένων γίνεται σε έναν διακομιστή στις εγκαταστάσεις του ΕΑΑ, ο οποίος είναι αφιερωμένος στην αρχειοθέτηση δεδομένων και τον ποιοτικό έλεγχο αυτών. Οι παραπάνω αισθητήρες έχουν αρκετά γρήγορη ενημέρωση των τιμών μέτρησης που κυμαίνεται από μερικά δευτερόλεπτα μέχρι ένα λεπτό. Η καταγραφή όλων αυτών των τιμών σε τόσο σύντομους χρόνους θα δημιουργούσε έναν μη λειτουργικό όγκο δεδομένων. Για τον λόγο αυτό το ΕΑΑ πραγματοποιεί καταγραφές του μέσου όρου των τιμών

ανά δέκα λεπτά. Επιπρόσθετα προτού αποθηκευτούν στην βάση τα δεδομένα, πραγματοποιείται έλεγχος και διαγραφή τυχόν εσφαλμένων μετρήσεων.

Ο έλεγχος ποιότητας των μετεωρολογικών δεδομένων υψηλής συχνότητας του ΕΑΑ εκτελείται σε δύο βήματα. Αρχικά εφαρμόζεται δύο φορές την ημέρα ένας πρωτογενής έλεγχος ποιότητας προκειμένου να εντοπιστούν και να επισημανθούν δεδομένα αμφιβόλου ποιότητας. Οι αλγόριθμοι αυτοί εκτός από τον έλεγχο της ποιότητας των δεδομένων, αποστέλλουν αυτόματα μηνύματα σφαλμάτων στους διαχειριστές του δικτύου του ΕΑΑ. Οι αλγόριθμοι περιλαμβάνουν ελέγχους για συμμόρφωση με το εύρος λειτουργίας κάθε αισθητήρα, υπέρβαση προκαθορισμένων ακραίων τιμών για κάθε μεταβλητή, ύποπτες επίμονες τιμές. Με τον τρόπο αυτό, μπορούν να εντοπιστούν ελαττωματικά όργανα μέτρησης, απώλεια επικοινωνίας είτε μεμονωμένων οργάνων είτε ολόκληρου του σταθμού και άλλα ζητήματα που μπορεί να χρειαστεί έλεγχος ή αντικατάσταση του εξοπλισμού. Στο δεύτερο βήμα, ελέγχονται όλα τα δεδομένα, πριν συμπεριληφθούν στην βάση δεδομένων, με εκείνα των υπολοίπων γειτονικών σταθμών. Τυχόν δεδομένα με χωρικές ή και χρονικές ασυνέπειες αποκλείονται λόγω μη αποδεκτής ποιότητας. Αυτός ο έλεγχος εκτελείται πριν από την ενσωμάτωση οποιωνδήποτε νέων δεδομένων στη βάση δεδομένων. Μόλις ολοκληρωθεί και το δεύτερο βήμα, όλα τα καλής ποιότητας δεδομένα εισάγονται και αρχειοθετούνται την βάση δεδομένων. και διατίθενται στο ευρύ κοινό.

6. Υλοποίηση Κώδικα Πρόγνωσης

6.1 Εισαγωγή Κεφαλαίου

Στο παρόν κεφάλαιο αναπτύσσεται και αναλύεται βήμα βήμα η υλοποίηση του θέματος της διπλωματικής, αρχής γενομένου από την εύρεση και λήψη των μετεωρολογικών δεδομένων μέχρι και την τελική πρόγνωση. Αρχικά περιγράφεται η διαδικασία εύρεσης, λήψης και διαχείρισης των μετεωρολογικών χρονοσειρών από την επίσημη βάση δεδομένων του Εθνικού Αστεροσκοπείου Αθηνών (ΕΑΑ). Έπειτα αναλύονται και παρουσιάζονται οι συγκρίσεις διαφόρων μεθόδων κανονικοποίησης και αλγορίθμων βελτιστοποίησης προκειμένου να βρεθεί και να χρησιμοποιηθούν οι βέλτιστες λύσεις στο τελικό στάδιο της πρόγνωσης. Τέλος, παρουσιάζονται και αναλύονται τα αποτελέσματα της τελικής πρόγνωσης.

6.2 Διαχείριση Χρονοσειρών

Ως πρώτο βήμα για την υλοποίηση της διπλωματικής εργασίας, ήταν η έρευνα στο διαδίκτυο για τον εντοπισμό και την λήψη ιστορικών δεδομένων από το ΕΑΑ. Τα δεδομένα εντοπίστηκαν και λήφθηκαν από τον πάρα κάτω σύνδεσμο του ΕΑΑ.

<https://data.climpact.gr/dataset/2f5bbe2a-7e27-40e7-9ff6-1dcc08c507fa>

Το ΕΑΑ διαθέτει τα ιστορικά μετεωρολογικά δεδομένα ανά σταθμό και μετρήσεις χρονικής περιόδου είτε 10 λεπτών είτε μιας ημέρας. Επιλέχθηκαν τα δεδομένα από τον ιστορικό σταθμό του Θησείου, ο οποίος είναι ο πρώτος επίγειος μετεωρολογικός σταθμός στον ελλαδικό χώρο με δεδομένα από τον έτος 1901.

Καθώς ο σκοπός της εργασίας δεν είναι η σωστή και ρεαλιστική πρόγνωση καιρού, ούτε η εύρεση του βέλτιστου μοντέλου αυτής, αλλά η κατανόηση των τεχνολογιών που χρησιμοποιούνται και η διαδικασία της. Επιλέχθηκε η χρονική περίοδος των δεδομένων να είναι μια μέτρηση ανά ημέρα, αφενός για τον λόγο που αναφέρθηκε προηγουμένως αλλά και για τον όγκο της χρονοσειράς. Αν γινόταν η επιλογή των δεδομένων με χρονική περίοδο των δέκα λεπτών, η διαχείριση τους και η εκτέλεση του κώδικα θα ήταν αρκετά χρονοβόρα και

πολύπλοκη. Επίσης, για τους ίδιους λόγους, τα δεδομένα που χρησιμοποιήθηκαν αφορούν την χρονική περίοδο 1-1-2010 μέχρι 31-12-2023.

Η τιμή των δεδομένων ανά ημέρα έχει προκύψει λαμβάνοντας τον μέσο όρο των τιμών των μετρήσεων ανά δέκα λεπτά για την κάθε μέρα. Αντίστοιχα οι τιμές των δεκάλεπτων μετρήσεων έχουν προκύψει από τον μέσο όρο των τιμών που στέλνει στο σύστημα καταγραφής ο κάθε αισθητήρας, καθώς οι αισθητήρες είναι ικανοί να αποστέλλουν μια τιμή ανά λίγα δευτερόλεπτα έως μερικά λεπτά.

Όπως αναφέρθηκε στο κεφάλαιο 5, το ΕΑΑ έχει ήδη προβεί σε ενέργειες αποσφαλμάτωσης και αφαίρεσης μη επιτρεπτών και λογικών τιμών στα δεδομένα αυτά. Τα δεδομένα αυτά θεωρούνται αξιόπιστα και θα μπορούσαν να χρησιμοποιηθούν και σε άλλα μοντέλα πρόγνωσης τα οποία θα στοχεύουν σε ουσιαστική και επιτηδευμένη καιρική πρόγνωση.

Το αρχείο που λήφθηκε είναι τύπου csv στου οποίου η κάθε γραμμή αφορά μια μέτρηση, ενώ οι στήλες του αφορούν έναν τύπο μέτρησης. Στο αρχείο εμφανίζονται ανά στήλη, με την σειρά δεδομένα για τις μετρήσεις της Μέσης Θερμοκρασίας (T_{mean}), Μέγιστη Θερμοκρασία (T_{max}), Ελάχιστη Θερμοκρασία (T_{min}), Μέση Σχετική Υγρασία (RH_{mean}), Μέγιστη Σχετική Υγρασία (RH_{max}), Ελάχιστη Σχετική Υγρασία (RH_{min}), Μέση Ατμοσφαιρική Πίεση (Prs_{mean}), Μέγιστη Ατμοσφαιρική Πίεση (Prs_{max}), Ελάχιστη Ατμοσφαιρική Πίεση (Prs_{min}), Συνολική Ποσότητα Βροχής (Ac_R), Μέση Ταχύτητα Ανέμου (WS_{mean}), Κατεύθυνση Ανέμου (DWD), Ριπή Ανέμου (WG). Το αρχείο φέρει συνολικά δεκατέσσερις στήλες και πέντε χιλιάδες εκατόν δεκατέσσερις (5114) γραμμές. Γίνεται εύκολα κατανοητό πως ακόμα και με τις ημερήσιες καταγραφές, το αρχείο δεν μπορεί να θεωρηθεί μικρό σε έκταση. Μέρος του συνολικού αρχείου παρατίθεται παρακάτω, υπό την μορφή που έχει και στο αρχείο .csv που χρησιμοποιήθηκε.

Τα LSTM χρησιμοποιούν τρία dataset, ένα για την εκπαίδευση (training) του μοντέλου, ένα για τον έλεγχο (test) και ρύθμιση των παραμέτρων και ένα δοκιμαστικό (validation) για την τελική πρόγνωση. Έτσι, κατά την εκτέλεση του αλγορίθμου, το αρχικό dataset πρόκειται να χωριστεί σε τρία ξεχωριστά dataset τα: Training, Test και Validation.

Date	T_ mean	T_ max	T_ min	RH_ mean	RH_ max	RH_ min	Prs_ mean	Prs_ max	Prs_ min	Ac_ R	WS_ mean	DWD	WG
2010-01-01	17.9	23.9	12.8	61.4	91	33	1003.6	1006.3	1002	0.2	4	WSW	35.4
2010-01-02	15.6	18.9	12.1	57.4	70	45	1005.2	1008.7	1001.5	0	6.8	WSW	51.5
2010-01-03	13.5	16.2	10	56	76	39	1011.7	1016.7	1008.6	0	5	WSW	40.2
2010-01-04	9.5	11.8	7.9	50.7	60	38	1011.7	1023.1	1016.6	0	4.3	NNE	22.5
2010-01-05	13.4	16.7	9.2	70.5	82	54	1021.3	1022.1	1015.5	0	7.9	S	41.8
2010-01-06	16.8	18.2	15.8	79.3	83	72	1018.7	1015.6	1011.1	0	15.2	S	51.5
2010-01-07	16.6	18.4	13.2	80.3	88	75	1012.9	1019.9	1012.9	0	4.5	SSE	29
2014-04-27	16.8	18.8	13.5	76.1	83	65	1008.7	1010.2	1007.5	1.2	4.1	NE	22.5
2014-04-28	17	21.1	13	68.6	86	51	1008	1009.9	1006.4	1.6	5.3	W	43.5
2014-04-29	18.4	22.5	15.3	60.3	76	45	1009.7	1011	1008.4	0	4.5	W	24.1
2014-04-30	18.2	22.2	14.9	55.8	69	41	1008.7	1010.1	1007.5	0	4.4	W	33.8
2014-05-01	18.4	22.8	13.3	61.4	78	43	1010.3	1011.1	1009.1	0	3.4	SSW	30.6
2014-05-02	18.8	23.1	14.8	67.7	83	46	1010.5	1011.9	1009.5	0	2.8	SSW	24.1
2014-05-03	18.9	22.2	16.6	74.6	84	59	1009.5	1011.5	1005.8	0	4	SSW	29
2014-04-27	16.8	18.8	13.5	76.1	83	65	1008.7	1010.2	1007.5	1.2	4.1	NE	22.5
2023-01-23	14.1	17.3	11.4	74.8	91	56	1022	1024.4	1020.3	7.4	2.8	S	22.5
2023-01-24	13.4	17.2	9.9	77.8	93	60	1024.1	1025.1	1023	2.8	3.6	NE	27.4
2023-01-25	10.1	11.6	9.4	75	78	71	1022.7	1025.1	1018.2	0	13	NE	45.1
2023-01-26	9.5	12.3	7.1	85.8	92	74	1010.9	1018.4	1006.7	36.4	5	NE	49.9
2023-01-27	10.3	14.1	6.9	80.9	92	58	1005.5	1007.1	1003.7	2.6	1.3	MME	16.1
2023-01-28	10.8	14.5	8.7	75.1	88	55	1011.8	1015.6	1007.2	0.8	1.9	NNE	16.1
2023-01-29	9.8	12.3	7.3	70.9	89	55	1016.9	1018.4	1015.4	1	2.4	NE	19.3

Πίνακας 1 Μορφή Αρχείου

6.3 Εύρεση Βέλτιστων Μεθόδων Κανονικοποίησης και Αλγορίθμου Βελτιστοποίησης

Η χρήση των κατάλληλων μεθόδων κανονικοποίησης είναι κομβικής σημασίας καθώς το μοντέλο εξαρτάται σε απόλυτο βαθμό από τις εισόδους του. Οι ίδιες χρονοσειρές με την χρήση διαφορετικών scaler μπορεί να μας οδηγήσουν σε τελείως διαφορετικά αποτελέσματα καθώς επηρεάζεται άμεσα η στατιστική συμπεριφορά των χρονοσειρών, η σταθερότητα της εκπαίδευσης του μοντέλου και η ικανότητα του να μαθαίνει μη γραμμικές σχέσεις.

Προκειμένου να πραγματοποιηθεί μια ορθή υλοποίηση καιρική πρόγνωσης, κρίθηκε απαραίτητο να διερευνηθεί ποιά μέθοδος κανονικοποίησης (scaler) είναι η βέλτιστη ανά παράμετρο και για ποιον αλγόριθμο βελτιστοποίησης (optimizer) της μηχανικής μάθησης. Έτσι δημιουργήθηκαν τρία ξεχωριστά configurations με scalers τα οποία διαχειρίζονται με ξεχωριστούς τρόπους τις χρονοσειρές. Για την καλύτερη κατανόηση, γίνεται οπτικοποίηση της μετατροπής των χρονοσειρών από τους scalers, με γραφικές παραστάσεις οι οποίες κατηγοριοποιούνται σε KDE Comparison, Time Series Overlay, Original-Scaled και Boxplots. Η κάθε μια από τις γραφικές μας βοηθάει να κατανοήσουμε τις μετατροπές που πραγματοποιεί το κάθε scaler.

Αρχικά έχουμε το Configuration 1 που περιέχει γραμμικούς scalers οι οποίοι δεν παραμορφώνουν τις κατανομές των τιμών. Θα μπορούσε να χαρακτηριστεί ως συντηρητική προσέγγιση, καθώς πραγματοποιείται επί της ουσίας ένα τυπικό scaling χωρίς αλλοιώσεις στο pattern των χρονοσειρών με αποτέλεσμα η εκπαίδευση του LSTM είναι σταθερή και προβλέψιμη.

Οι scalers που χρησιμοποιούνται ανά παράμετρο είναι:

Για τις θερμοκρασίες και τις πιέσεις χρησιμοποιείται ο StandardScaler, ο οποίος τροποποιεί τις τιμές έτσι ώστε ο μέσος όρος (μ) να γίνει 0 και η τυπική απόκλιση (σ) να γίνει 1. Δεν αλλάζει το σχήματα της κατανομής και για τον λόγο αυτό θεωρείται χρήσιμο στα LSTM. Για την σχετική υγρασία, χρησιμοποιείται ο MinMaxScaler, ο οποίος μετατρέπει την μικρότερη τιμή του dataset σε 0, την μεγαλύτερη σε 1 και όλες τις υπόλοιπες σε δεκαδικές

τιμές μέσα στο εύρος $[0,1]$. Διατηρείται το σχήματα της αρχικής κατανομής αλλάζοντας απλά την κλίμακα.

Για τις τιμές της βροχής, χρησιμοποιείται ο $\log1p$ και ο MinMaxScaler . Η χρήση του Log1p scaler είναι απαραίτητη καθώς μειώνει τα έντονα και απότομα spikes και σε συνδυασμό με τον MinMaxScaler οριοθετείται στο εύρος $[0,1]$.

Τέλος στη παράμετρος του αέρα, χρησιμοποιείται ο Robust Scaler . Καθώς άνεμος έχει ριπές, οι τιμές του εμφανίζουν αρκετά spikes. Έτσι ο Robust Scaler βρίσκει την διάμεσο τιμή βάζοντας σε σειρά τις τιμές από την μικρότερη προς την μεγαλύτερη. Στην διάμεσο δίνεται η τιμή 0 και οι υπόλοιπες τιμές τοποθετούνται γύρω από αυτή.

Στο Configuration 2 οι scalers που χρησιμοποιήθηκαν στοχεύουν να μετατρέψουν τα δεδομένα σε Gaussian και στην ανθεκτικότητα των ακραίων τιμών ή heavy tails. Το configuration αυτό, αφενός βοηθάει το LSTM να “μάθει” αν και μπορεί να υπάρξει δυσκολία στην ερμηνεία των τιμών καθώς δεν χρησιμοποιούνται γραμμικοί scalers.

Για τις τιμές θερμοκρασίας, πίεσης και ανέμου χρησιμοποιήθηκε ο $\text{PowerTransformer Scaler}$. Ο scaler σταθεροποιεί την διακύμανση των τιμών και μειώνει την ασυμμετρία (skewness) τους. Καθώς οι χρονοσειρές των θερμοκρασιών μπορούν να έχουν ακραίες τιμές, με την χρήση του scaler αυτού, έχουμε μια καλύτερη χρήση στο LSTM. Επίσης, οι τιμές των πιέσεων αν και έχουν μικρές διακυμάνσεις μπορεί να εμφανίσουν ασυμμετρία η οποία θα αντιμετωπιστεί από τον scaler. Οι τιμές του ανέμου μπορεί να εμφανίζουν heavy tails, δηλαδή να υπάρχουν πλήθος τιμών που να εμφανίζονται για μεγάλο χρονικό διάστημα. Το φαινόμενο αυτό μπορεί να μας αλλιώσει αρκετά το LSTM, αντιμετωπίζεται όμως με τον $\text{Power Transformer Scaler}$.

Για τις καταγραφές της σχετική υγρασίας χρησιμοποιήθηκε ο Robust Scaler ο οποίος περιγράφηκε και στο Configuration 1.

Για τις καταγραφές της βροχής χρησιμοποιήθηκαν οι Log1p και Robust Scalers . Όπως και στην προηγούμενη περίπτωση, γίνεται χρήση του Log1p Scaler για να μειωθούν αρχικά τυχόν απότομες κορυφές που υπάρχουν στις τιμές και στην συνέχεια ο Robust Scaler .

Τέλος, στο Configuration 3 έχουν επιλεγεί $\text{QuantileTransformers scalers}$ οι οποίοι μετατρέπουν την κατανομή σε Gaussian και Uniform. Πρόκειται για το επιθετικό scaling καθώς τροποποιείται πλήρως η δομή των δεδομένων. Αν και υπάρχει μεγάλη πιθανότητα να

παραμορφωθούν χρήσιμα patterns για το LSTM, εξαλείφονται πλήρως τυχόν ασυμμετρίες και ακραίες τιμές.

Προκειμένου να γίνει σύγκριση μεταξύ των scaler και optimizer, χρησιμοποιήθηκαν οι μετρήσεις των παραμέτρων MAE και MSE. Η παράμετρος MAE αναφέρεται στο Μέσο Απόλυτο Σφάλμα. Το MSE μετράει το Μέσο Τετραγωνικό Σφάλμα. Επί της ουσίας, είναι ο δείκτης που μετράει πόσο σταθερό είναι το μοντέλο και αν γίνονται σημαντικά λάθη. Δηλαδή στο πόσο κατά μέσο όρο αποκλίνει πρόβλεψη από την πραγματική τιμή. Μαθηματικά εκφράζονται ως εξής:

$$\text{MAE} = \frac{1}{N} \sum_{t=1}^N |X_{\text{true}} - X_{\text{pred}}|$$

$$\text{MSE} = \frac{1}{N} \sum_{t=1}^N (X_{\text{true}} - X_{\text{pred}})^2$$

όπου N = πλήθος τιμών, X_{true} = πραγματική τιμή, X_{pred} = προβλεπόμενη τιμή

Δημιουργήθηκαν τρεις ξεχωριστοί κώδικες στους οποίους εφαρμόζονται ξεχωριστά τα configuration των scalers. Η υλοποίηση του κώδικα με τη χρήση του Google Colab, το οποίο είναι άμεσα συνδεδεμένο με τον driver και το path στο οποίο βρίσκεται το datasheet σε μορφή csv. Οι κώδικες αυτοί βοήθησαν στην ομαδοποίηση και διαχωρισμό των αποτελεσμάτων και στην καλύτερη κατανόηση των αποτελεσμάτων. Παρατίθεται ολόκληρος μόνο ο κώδικας για το configuration 1, καθώς δεν υπάρχει κάποια ιδιαίτερη διαφορά στους υπόλοιπους δύο κώδικες πέρα του τμήματος της εφαρμογής των scalers.

Η αρχιτεκτονική που επιλέχθηκε για το LSTM είναι βαθμιαία μειούμενη $64 \rightarrow 32 \rightarrow 16$ η οποία βοηθά στην καλύτερη συμπίεση των χαρακτηριστικών, με το μικρότερο ρίσκο υπερφόρτωσης, ενώ παράλληλα διατηρείται η ικανότητα του δικτύου να μαθαίνει σύνθετες χρονικές εξαρτήσεις. Καθώς υπάρχουν λιγότερες μονάδες στα βαθύτερα στρώματα, μειώνεται η υπολογιστική επιβάρυνση και βελτιώνεται η σταθερότητα των gradients.

6.3.1 Κώδικας Σύγκρισης Scalers

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import pandas.api.types as ptypes

from sklearn.preprocessing import StandardScaler, MinMaxScaler,
RobustScaler

from tensorflow.keras import layers, models

from tensorflow.keras.optimizers import Adam, RMSprop, Nadam

from tensorflow.keras.callbacks import EarlyStopping


def load_and_preprocess(file_path):

    df = pd.read_csv(file_path, parse_dates=["Date"])

    # keep Date column for splitting

    df = df.replace('---', np.nan)

    df = df.dropna().reset_index(drop=True)

    wind_dir_map = {

        "N": 0, "NNE": 22.5, "NE": 45, "ENE": 67.5,

        "E": 90, "ESE": 112.5, "SE": 135, "SSE": 157.5,

        "S": 180, "SSW": 202.5, "SW": 225, "WSW": 247.5,

        "W": 270, "WNW": 292.5, "NW": 315, "NNW": 337.5

    }

    # If DWD exists, map to degrees and create sin/cos

    if "DWD" in df.columns:

        df["DWD_deg"] = df["DWD"].map(wind_dir_map)
```

```
theta = np.deg2rad(df["DWD_deg"])

df["DWD_sin"] = np.sin(theta)

df["DWD_cos"] = np.cos(theta)

df = df.drop(columns=["DWD", "DWD_deg"])

return df

# VARIABLE GROUPS

temp_cols = ["T_mean", "T_max", "T_min"]
rh_cols = ["RH_mean", "RH_max", "RH_min"]
prs_cols = ["Prs_mean", "Prs_max", "Prs_min"]
rain_col = ["Ac_R"]
wind_cols = ["WS_mean", "WG"]

# DWD_sin, DWD_cos remain as is and are not scaled

# SCALERS CONFIGURATION

def get_config1():
    return {
        "name": "Config1_original_like",
        "temp": StandardScaler(),
        "rh": MinMaxScaler(),
        "prs": StandardScaler(),
        "rain": ("log1p", MinMaxScaler()),
        "wind": RobustScaler()
    }
```

```
# SPLIT BY DATE

def split_by_date(df, date_col="Date"):
    """
    Splits df into train/test/validation by date ranges:
        - train: 2010-01-01 .. 2019-12-31
        - test:  2020-01-01 .. 2021-12-31
        - val:   2022-01-01 .. end
    """
    df = df.copy()
    df[date_col] = pd.to_datetime(df[date_col])

    train_mask = (df[date_col] >= "2010-01-01") & (df[date_col] <= "2019-
12-31")
    val_mask    = (df[date_col] >= "2020-01-01") & (df[date_col] <= "2021-
12-31")
    test_mask   = (df[date_col] >= "2022-01-01")

    train_df = df.loc[train_mask].reset_index(drop=True)
    val_df = df.loc[test_mask].reset_index(drop=True)
    test_df  = df.loc[val_mask].reset_index(drop=True)

    return train_df, val_df, test_df

# APPLY SCALERS

def apply_scalers_to_dfs(train_df, test_df, val_df, config):
    # Work on copies
    train = train_df.copy()
    test  = test_df.copy()
    val   = val_df.copy()
```

```
# Keep Date column aside

date_col = "Date" if "Date" in train.columns else None

if date_col:

    train_dates = train[date_col].copy()
    test_dates = test[date_col].copy()
    val_dates = val[date_col].copy()

    # drop Date for numeric transforms

    train = train.drop(columns=[date_col])
    test = test.drop(columns=[date_col])
    val = val.drop(columns=[date_col])

# Ensure numeric

train = train.apply(pd.to_numeric, errors='coerce')
test = test.apply(pd.to_numeric, errors='coerce')
val = val.apply(pd.to_numeric, errors='coerce')

# Drop rows with NaNs that may have appeared after conversion

train = train.dropna().reset_index(drop=True)
test = test.dropna().reset_index(drop=True)
val = val.dropna().reset_index(drop=True)

# TEMP

temp_scaler = config["temp"]

temp_scaler.fit(train[temp_cols])

train[temp_cols] = temp_scaler.transform(train[temp_cols])
test[temp_cols] = temp_scaler.transform(test[temp_cols])
val[temp_cols] = temp_scaler.transform(val[temp_cols])
```

```
# RH

rh_scaler = config["rh"]
rh_scaler.fit(train[rh_cols])
train[rh_cols] = rh_scaler.transform(train[rh_cols])
test[rh_cols] = rh_scaler.transform(test[rh_cols])
val[rh_cols] = rh_scaler.transform(val[rh_cols])

# PRS

prs_scaler = config["prs"]
prs_scaler.fit(train[prs_cols])
train[prs_cols] = prs_scaler.transform(train[prs_cols])
test[prs_cols] = prs_scaler.transform(test[prs_cols])
val[prs_cols] = prs_scaler.transform(val[prs_cols])

# RAIN

mode, rain_scaler = config["rain"]

if mode == "loglp":
    train_rain = np.loglp(train[rain_col])
    rain_scaler.fit(train_rain)
    train[rain_col] = rain_scaler.transform(train_rain)
    test[rain_col] = rain_scaler.transform(np.loglp(test[rain_col]))
    val[rain_col] = rain_scaler.transform(np.loglp(val[rain_col]))
else:
    rain_scaler.fit(train[rain_col])
    train[rain_col] = rain_scaler.transform(train[rain_col])
    test[rain_col] = rain_scaler.transform(test[rain_col])
    val[rain_col] = rain_scaler.transform(val[rain_col])
```

```
# WIND

wind_scaler = config["wind"]
wind_scaler.fit(train[wind_cols])
train[wind_cols] = wind_scaler.transform(train[wind_cols])
test[wind_cols] = wind_scaler.transform(test[wind_cols])
val[wind_cols] = wind_scaler.transform(val[wind_cols])

# Reattach Date column
if date_col:
    train[date_col] = train_dates.iloc[:len(train)].values
    test[date_col] = test_dates.iloc[:len(test)].values
    val[date_col] = val_dates.iloc[:len(val)].values

return train, test, val

# WINDOWING

def create_dataset_multi(dataset, look_back=24):
    """
    dataset: numpy array shape (n_samples, n_features)
    """
    X, y = [], []
    for i in range(len(dataset) - look_back):
        X.append(dataset[i:i + look_back])
        y.append(dataset[i + look_back])
    return np.array(X), np.array(y)
```

```
# LSTM MODEL

def build_model(input_shape, optimizer):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        layers.LSTM(64, return_sequences=True),
        layers.BatchNormalization(),
        layers.LSTM(32, return_sequences=False),
        layers.BatchNormalization(),
        layers.Dense(16, activation='relu'),
        layers.Dense(input_shape[1])
    ])
    model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
    return model

# SET OPTIMIZER

def get_optimizer(name):
    if name == "adam": return Adam()
    if name == "rmsprop": return RMSprop()
    if name == "nadam": return Nadam()
    raise ValueError("Unknown optimizer")
```

```
# RUN EXPERIMENT

def run_experiment(train_df, test_df, val_df, config, optimizer_name,
look_back=24, epochs=40):

    optimizer_obj = get_optimizer(optimizer_name)

    # Fit scalers on train and transform all sets
    train_scaled, test_scaled, val_scaled =
apply_scalers_to_dfs(train_df, test_df, val_df, config)

    # Drop Date column for modeling if present
    for df in (train_scaled, test_scaled, val_scaled):
        if "Date" in df.columns:
            df.drop(columns=["Date"], inplace=True)

    # Convert to numpy float32
    train_arr = train_scaled.values.astype(np.float32)
    test_arr = test_scaled.values.astype(np.float32)
    val_arr = val_scaled.values.astype(np.float32)

    # Create windows
    X_train, y_train = create_dataset_multi(train_arr,
look_back=look_back)
    X_test, y_test = create_dataset_multi(test_arr,
look_back=look_back)
    X_val, y_val = create_dataset_multi(val_arr, look_back=look_back)

    # If any set is too small for windowing, raise informative error
    if len(X_train) == 0:
        raise ValueError("Training set too small for the chosen
look_back.")
```

```
if len(X_test) == 0:
    raise ValueError("Test set too small for the chosen look_back.")
if len(X_val) == 0:
    print("Warning: Validation set too small for windowing;
predictions on validation will be skipped.")

model = build_model((look_back, X_train.shape[2]), optimizer_obj)

early_stop = EarlyStopping(monitor='val_loss', patience=8,
restore_best_weights=True)

history = model.fit(
    X_train, y_train,
    validation_data=(X_test, y_test),
    epochs=epochs,
    batch_size=32,
    callbacks=[early_stop],
    verbose=1)

# Predict on test and val
y_pred_test = model.predict(X_test, verbose=0)
mae_per_param_test = np.mean(np.abs(y_pred_test - y_test), axis=0)
mse_per_param_test = np.mean((y_pred_test - y_test)**2, axis=0)

results = {
    "optimizer": optimizer_name,
    "columns": train_scaled.columns.tolist(),
    "mae_per_param_test": mae_per_param_test,
    "mse_per_param_test": mse_per_param_test,
    "mae_total_test": np.mean(mae_per_param_test),
    "mse_total_test": np.mean(mse_per_param_test),
```

```
        "history": history
    }

    # Validation predictions
    if len(X_val) > 0:
        y_pred_val = model.predict(X_val, verbose=0)
        mae_per_param_val = np.mean(np.abs(y_pred_val - y_val), axis=0)
        mse_per_param_val = np.mean((y_pred_val - y_val)**2, axis=0)
        results.update({
            "mae_per_param_val": mae_per_param_val,
            "mse_per_param_val": mse_per_param_val,
            "mae_total_val": np.mean(mae_per_param_val),
            "mse_total_val": np.mean(mse_per_param_val),
            "y_pred_val": y_pred_val,
            "y_val": y_val
        })

    return results

# SCALERS PLOTS

def plot_scalers(train_df, test_df, val_df, config, sample_size=300):

    train_num = train_df.copy()
    test_num = test_df.copy()
    val_num = val_df.copy()

    for col in train_num.columns:
        train_num[col] = pd.to_numeric(train_num[col], errors='coerce')
```

```
for col in test_num.columns:
    test_num[col] = pd.to_numeric(test_num[col], errors='coerce')
for col in val_num.columns:
    val_num[col] = pd.to_numeric(val_num[col], errors='coerce')

# Drop rows with NaNs that
train_num = train_num.dropna().reset_index(drop=True)
test_num = test_num.dropna().reset_index(drop=True)
val_num = val_num.dropna().reset_index(drop=True)

train_scaled, test_scaled, val_scaled =
apply_scalers_to_dfs(train_num, test_num, val_num, config)

# Train KDE per numeric column
for col in train_num.columns:
    if col in ["DWD_sin", "DWD_cos", "Date"]:
        continue
    if not ptypes.is_numeric_dtype(train_num[col]):
        print(f"Skipping non-numeric column for KDE: {col}")
        continue

    if train_num[col].dropna().shape[0] < 10:
        print(f"Skipping {col} for KDE (too few values).")
        continue

    plt.figure(figsize=(10,4))
    sns.kdeplot(train_num[col].dropna(), label="Train Original",
linewidth=1.5)
    sns.kdeplot(train_scaled[col].dropna(), label="Train Scaled",
linewidth=1.5)
    plt.title(f"{col} - Train KDE (original vs scaled)")
```

```
plt.legend()

plt.grid(True)

plt.tight_layout()

plt.show()

# Time overlays

for df_orig, df_scaled, label in [(train_num, train_scaled, "Train"),
                                   (test_num, test_scaled, "Test"), (val_num, val_scaled, "Val")]:

    for col in df_orig.columns:

        if col in ["DWD_sin", "DWD_cos", "Date"]:

            continue

        if not ptypes.is_numeric_dtype(df_orig[col]):

            continue

        orig = df_orig[col].values[:sample_size]

        scaled = df_scaled[col].values[:sample_size]

        if len(orig) < 2 or len(scaled) < 2:

            continue

        plt.figure(figsize=(12,3))

        plt.plot(orig, label=f"{label} Original", linewidth=1.5)

        plt.plot(scaled, label=f"{label} Scaled", linewidth=1.5)

        plt.title(f"{col} - Time Overlay ({label})")

        plt.legend()

        plt.grid(True)

        plt.tight_layout()

        plt.show()

# MAIN

def main():

    file_path = "/content/drive/MyDrive/Διπλωματική ΕΑΠ Κιούσης/lab
files/athens.csv"

    df = load_and_preprocess(file_path)
```

```
# split into train/test/val by dates

train_df, test_df, val_df = split_by_date(df, date_col="Date")

print(f"Train rows: {len(train_df)}, Test rows: {len(test_df)}, Val
rows: {len(val_df)}")

config = get_config1()

optimizers = ["adam", "rmsprop", "nadam"]

# Visualize scalers effect on the three sets
plot_scalers(train_df, test_df, val_df, config)

for opt in optimizers:
    print(f"\n=== Running {config['name']} with {opt} ===")

    res = run_experiment(train_df, test_df, val_df, config, opt,
look_back=24, epochs=40)

    print("\nMAE per parameter (TEST):")

    for col, val in zip(res["columns"], res["mae_per_param_test"]):
        print(f" {col}: {val:.4f}")

    print("\nMSE per parameter (TEST):")

    for col, val in zip(res["columns"], res["mse_per_param_test"]):
        print(f" {col}: {val:.4f}")

    print(f"\nTotal MAE (TEST): {res['mae_total_test']:.4f}")
    print(f"Total MSE (TEST): {res['mse_total_test']:.4f}")

    if "mae_total_val" in res:
        print(f"\nTotal MAE (VAL predictions):
{res['mae_total_val']:.4f}")

        print(f"Total MSE (VAL predictions):
{res['mse_total_val']:.4f}")
```

```
# Plot training loss

plt.figure(figsize=(8,4))

plt.plot(res["history"].history['loss'], label='Train Loss')

plt.plot(res["history"].history['val_loss'], label='Val (Test) Loss')

plt.title(f"Training Loss - {config['name']} + {opt}")

plt.legend()

plt.grid(True)

plt.show()

if __name__ == "__main__":
    main()
```

Ο κώδικας έχει τμηματοποιηθεί σε ξεχωριστές συναρτήσεις ώστε να ελέγχετε η ροή του σε ένα σημείο του κώδικα, συγκεκριμένα την συνάρτηση `main()` και να είναι ευκολότερα κατανοητός και ευανάγνωστος. Η συνάρτηση `main()` ελέγχει την ροή του προγράμματος και μέσα σε αυτή καλούνται οι υπόλοιπες συναρτήσεις, καλείται και “διαβάζεται” το `datasheet`, εμφανίζονται αναλυτικά για κάθε scaler οι τιμές των MAE και MSE καθώς και τα plot των training losses.

Η μέθοδος `load_and_preprocess()` φορτώνει και επεξεργάζεται το αρχικό σύνολο δεδομένων. Λαμβάνει σαν όρισμα το `dataset` και αρχικά αφαιρεί την πρώτη στήλη των ημερομηνιών η οποία δεν μπορεί να χρησιμοποιηθεί απευθείας στην μάθηση του μοντέλου. Εν συνεχεία, αντικαθιστά τυχόν κενά κελιά ώστε το τελικό σύνολο να είναι πλήρως αριθμητικό και να μην υπάρχουν ασυνέχειες. Επίσης, γίνεται μετατροπή της παραμέτρου της διεύθυνσης ανέμου σε γωνιακή (μοίρες) αναπαράσταση και στη συνέχεια σε δύο συνιστώσες μέσω ημιτόνου και συνημιτόνου, έτσι ώστε να αποφευχθούν ασυνέχειες σε περίπτωση μετατροπής σε απλή αριθμητική κωδικοποίηση. Η μέθοδος επιστρέφει ένα καθαρό και πλήρως αριθμητικό `Data Frame (df)` έτοιμο για χρήση σε νευρωνικό δίκτυο.

Να αναφερθεί πως εκτό συνάρτησης υπάρχουν τα ορίσματα των μεταβλητών που ομαδοποιούνται σε κατηγορίες. Η κατηγοριοποίηση αυτή πραγματοποιήθηκε προκειμένου να δημιουργηθούν λίστες και να διαχειρίζονται αυτές αντί για κάθε μια παράμετρο ξεχωριστά.

Επόμενη συνάρτηση είναι η `get_config1()` στην οποία ορίζεται ποιος scaler θα εφαρμοστεί σε κάθε λίστα μεταβλητών. Όπως ήδη αναφέρθηκε, στο configuration 1 για την θερμοκρασία και την πίεση χρησιμοποιείται ο `StandardScaler`, για την υγρασία ο `MinMaxScaler`, για την βροχόπτωση ο `log1p` και στην συνέχεια ο `MinMaxScaler` και τέλος για την παράμετρο του ανέμου χρησιμοποιείται ο `RobustScaler`.

Η συνάρτηση `split_by_date()` δέχεται δύο ορίσματα, το `Data Frame (df)` και το `date_col` το οποίο είναι η παράμετρος της συνάρτησης που καθορίζεται η ημερομηνία διαχωρισμού του `Data Frame`. Έτσι στην συνάρτηση αυτή το επεξεργασμένο `data frame` διαχωρίζεται σε `train_df`, `test_df` και `val_df` βάση της ημερομηνίας. Πιο συγκεκριμένα στο `train_df` αποθηκεύονται όλες οι μετρήσεις από 01-01-2010 μέχρι 31-12-2019. Στο `test_df` αποθηκεύονται όλες οι μετρήσεις από 01-01-2020 μέχρι 31-12-2021. Τέλος στο `val_df` αποθηκεύονται οι υπόλοιπες μετρήσεις οι οποίες είναι από 01-01-2022 μέχρι 31-12-2023. Η μέθοδος επιστρέφει τα τρία αυτά `data frames`.

Ο κώδικας συνεχίζει στην μέθοδο `apply_scalers_to_df()` η οποία δέχεται τα τρία νέα `data frames` και το configuration των scalers που θέσαμε προηγουμένως. Όλες οι τροποποιήσεις των `data frames` γίνονται σε αντίγραφα για να μην αλλοιωθούν τα αρχικά. Τα δεδομένα της στήλης `Date` ανά `data frame` αποθηκεύεται προσωρινά στα `train_dates`, `test_date` και `val_dates`. Εφαρμόζονται στην πράξη οι scalers που ορίστηκαν προηγουμένως για κάθε παράμετρο και τέλος, εισάγεται η στήλη `Date`. Επιστρέφονται τα τρία νέα `data frames` (`train`, `test`, `val`) στα οποία έχουν εφαρμοστεί οι scalers.

Εν συνεχεία, η μέθοδος `create_dataset_multi()` μετατρέπει την χρονοσειρά σε εποπτευόμενο σύνολο δεδομένων κατάλληλο για μοντέλο LSTM. Λαμβάνει ως είσοδο το `dataset` και έναν ακέραιο. Για κάθε χρονική στιγμή, η μέθοδος δημιουργεί ένα παράθυρο εισόδου με σταθερό μήκος στο οποίο περιλαμβάνονται οι προηγούμενες 24 μέρες και ορίζεται ως στόχος η τιμή της επόμενης χρονικής στιγμής. Με τον τρόπο αυτό δημιουργούνται τα τρισδιάστατα tensors που απαιτεί το LSTM. Η πρώτη διάσταση είναι το πλήθος των δειγμάτων, η δεύτερη το μήκος του παραθύρου και η τρίτη ο αριθμός των χαρακτηριστικών.

Επόμενη συνάρτηση είναι η `build_model()` η οποία κατασκευάζει το νευρωνικό δίκτυο που χρησιμοποιείται για την πρόγνωση. Στο μοντέλο έχουν υλοποιηθεί δύο διαδοχικά LSTM επίπεδα, τα οποία επιτρέπουν την εκμάθηση βραχυπρόθεσμων και μακροπρόθεσμων εξαρτήσεων στις χρονοσειρές. Μεταξύ των επιπέδων αυτών, έχουν υλοποιηθεί επίπεδα `Batch`

Normalization προκειμένου να σταθεροποιήσουν την διαδικασία εκπαίδευσης και να επιταχύνουν την σύγκλιση. Η δομή του μοντέλου είναι $64 \rightarrow 32 \rightarrow 16$ ώστε να έχουμε καλύτερη συμπίεση των χαρακτηριστικών με όσο το δυνατόν μικρότερο ρίσκο υπερπροσαρμογής. Στο μοντέλο προστίθεται και ο optimizer και υπολογίζονται τα κόστη συνάρτησης MAE και MSE για κάθε παράμετρο.

Η συνάρτηση `get_optimizer()` δέχεται ως είσοδο το όνομα ενός optimizer και επιστρέφει το αντίστοιχο αντικείμενο Keras. Η μέθοδος αυτή επιτρέπει την εύκολη εναλλαγή μεταξύ των τριών επιλεγμένων optimizers οι οποίοι είναι οι: Adam, RMSprop και Nadam. Επίσης, έχει προστεθεί ένα `ValueError` στην περίπτωση που ζητηθεί ξεχωριστός optimizer. Κατ' αυτόν τον τρόπο ο κώδικας σταματά την ροή εμφανίζοντας το απαραίτητο μήνυμα.

Μια από τις κομβικές συναρτήσεις του κώδικα είναι η `run_experiment()` στην οποία υλοποιείται ένα πλήρες πείραμα για έναν συγκεκριμένο scaler configuration και optimizer βάση των ορισμάτων που θα δοθούν στην κλήση της που γίνεται στην `main()`. Αρχικά δέχεται ως εισόδους τα τρία dataset, εφαρμόζει σε αυτά τους scalers καλώντας την μέθοδο `apply_scalers_to_df()` και στη συνέχεια αφαιρεί την στήλη της ημερομηνίας. Καλεί την συνάρτηση `create_dataset_multi()` για να δημιουργήσει το παράθυρο σημάτων (windowing) Εν συνεχεία δημιουργείται το LSTM μοντέλο χρησιμοποιώντας το σχήμα των `x_train`, ορίζει το early stop βάση του `val_loss` και εκπαιδεύει το μοντέλο με τις παραμέτρους `x_train` και `y_train`. Χρησιμοποιεί τις παραμέτρους `x_test` και `y_test` για να ελέγξει την απόδοση του μοντέλου και τέλος υπολογίζει τις τιμές των μετρικών MAE και MSE χρησιμοποιώντας στο μοντέλο τα `x_val` και `y_val`. Η μέθοδος επιστρέφει το όνομα του optimizer που χρησιμοποιήθηκε, τις τιμές MAE και MSE καθώς και το ιστορικό της εκπαίδευσης του μοντέλου.

Τέλος, η μέθοδος `plot_scaler()` οπτικοποιεί το αποτέλεσμα της εφαρμογής των scaler σε κάθε παράμετρο. Αρχικά, μετατρέπονται όλες οι στήλες σε αριθμητική μορφή και διαγράφονται τυχόν NaN τιμές. Στη συνέχεια δημιουργούνται για κάθε μεταβλητή τρία διαγράμματα. Το διάγραμμα κατανομής πυκνότητας πριν και μετά την κανονικοποίηση (Histogram + KDE) και την επικάλυψη των χρονοσειρών (Time-series overlay).

Τα δύο αυτά διαγράμματα επιλέχθηκαν προκειμένου να υπάρχει μια λεπτομερής καταγραφή του αποτελέσματος των scalers. Το KDE (Kernel Density Estimation) plot εμφανίζει την κατανομή των τιμών πριν και μετά την εφαρμογή του scaler. Η καμπύλη του διαγράμματος, μας δείχνει σε ποιές περιοχές του άξονα των τιμών γίνεται η συγκέντρωση επιτρέποντας

ταυτόχρονα την άμεση και εύκολη αναγνώριση ουρών, λοξότητας και πολυτροπικότητας. Το διάγραμμα Time-series overlay μας δείχνει πως τροποποιείται η δυναμική της χρονοσειράς πριν και μετά την εφαρμογή του scaler. Από το διάγραμμα αυτό μπορούμε να δούμε τα patterns που δημιουργήθηκαν και χρησιμοποιήθηκαν στην εκπαίδευση του LSTM.

Όπως αναφέρθηκε, η ίδια δομή κώδικα χρησιμοποιήθηκε και για τα άλλα δύο scaler configuration. Η μόνη διαφορά βρίσκεται στην μέθοδο `get_configx()` στην οποία δηλώνονται οι scalers ανά παράμετρο.

Για το Configuration 2 η μέθοδος είναι η εξής:

```
def get_config2():  
    return {  
        "name": "Config2_robust_power",  
        "temp": PowerTransformer(method="yeo-johnson"),  
        "rh": RobustScaler(),  
        "prs": PowerTransformer(method="yeo-johnson"),  
        "rain": ("log1p", RobustScaler()),  
        "wind": PowerTransformer(method="yeo-johnson")  
    }
```

Για το Configuration 3 η μέθοδος είναι η εξής:

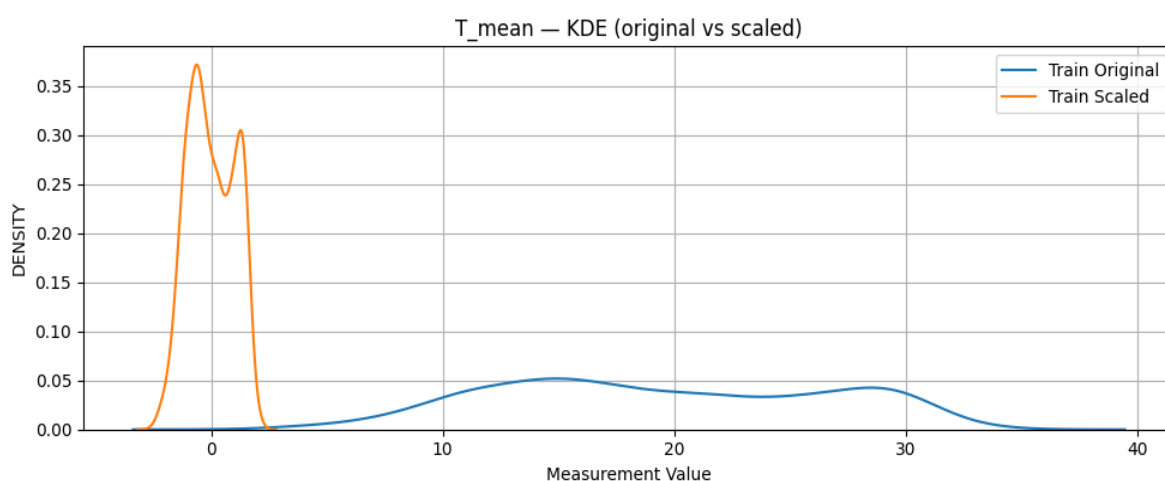
```
def get_config3():  
    return { "name": "Config3_quantile",  
        "temp": QuantileTransformer(output_distribution="uniform",  
random_state=42),  
        "rh": QuantileTransformer(output_distribution="uniform",  
random_state=42),  
        "prs": QuantileTransformer(output_distribution="normal",  
random_state=42),  
        "rain": ("log1p", QuantileTransformer(output_distribution="normal",  
random_state=42)),  
        "wind": QuantileTransformer(output_distribution="uniform",  
random_state=42)  
    }
```

Οι κώδικες έχουν χρόνο εκτέλεσης περίπου τα 8 λεπτά έκαστος. Ο χρόνος αυτός προκύπτει από τον όγκο δεδομένων εισόδων καθώς και από τις πολύπλοκες διαδικασίες οι οποίες εκτελούνται για την κανονικοποίηση, δημιουργία και εκπαίδευση του μοντέλου. Για τον λόγο αυτό προτιμήθηκε να μην χρησιμοποιηθεί ένας ενιαίος κώδικας καθώς και για την καλύτερη κατανόηση αυτού.

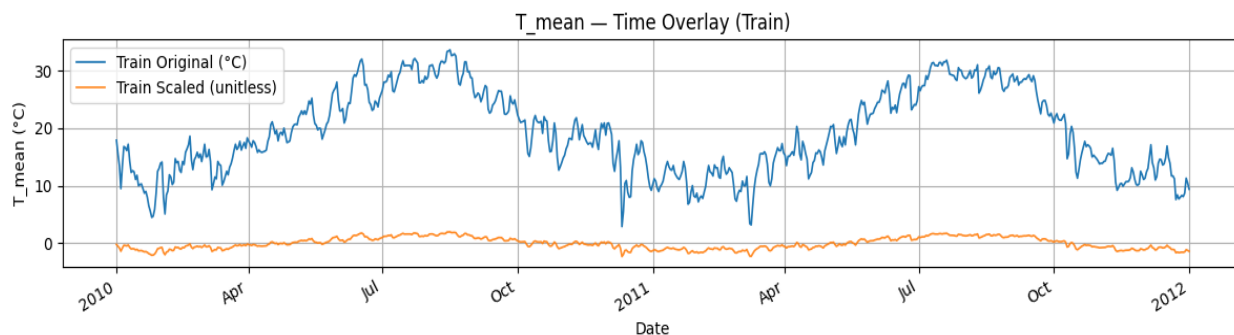
6.3.2 Αποτελέσματα Συγκρίσεων

Προκειμένου να αποφευχθεί η μεγάλη έκταση της διπλωματικής, παρατίθεται σύγκριση των διαγραμμάτων KDE και Time Series Overlay για την παράμετρο της μέσης τιμής της θερμοκρασίας (T_mean) μετά την χρήση των scalers. Στο KDE διάγραμμα, εμφανίζεται η κατανομή των τιμών πριν και μετά την χρήση του εκάστοτε scaler.

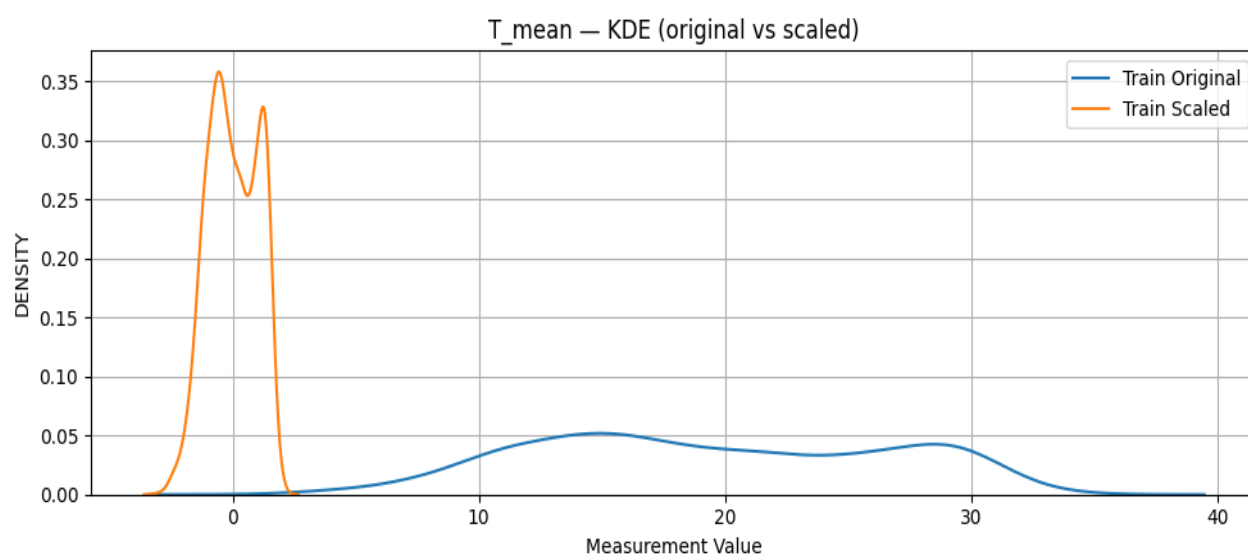
Τα διαγράμματα KDE και Time Series Overlay για την μέση τιμή της θερμοκρασία (T_mean) είναι τα εξής:



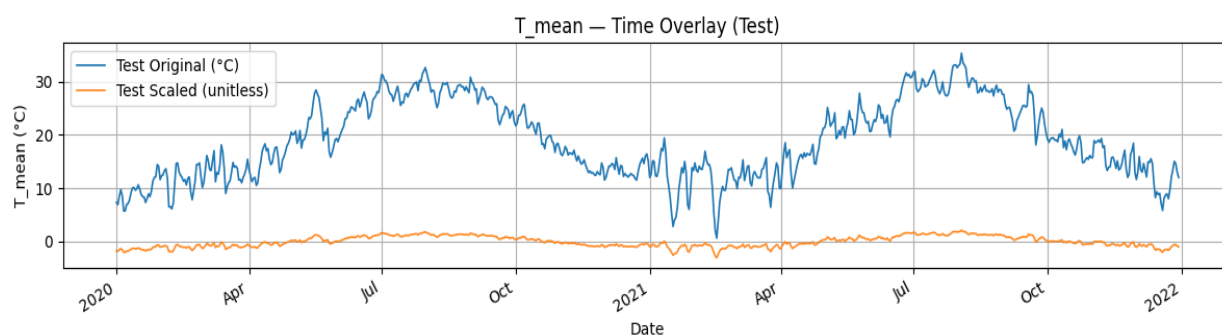
Εικόνα 10 Configuration 1 KDE T_mean με StandarScaler



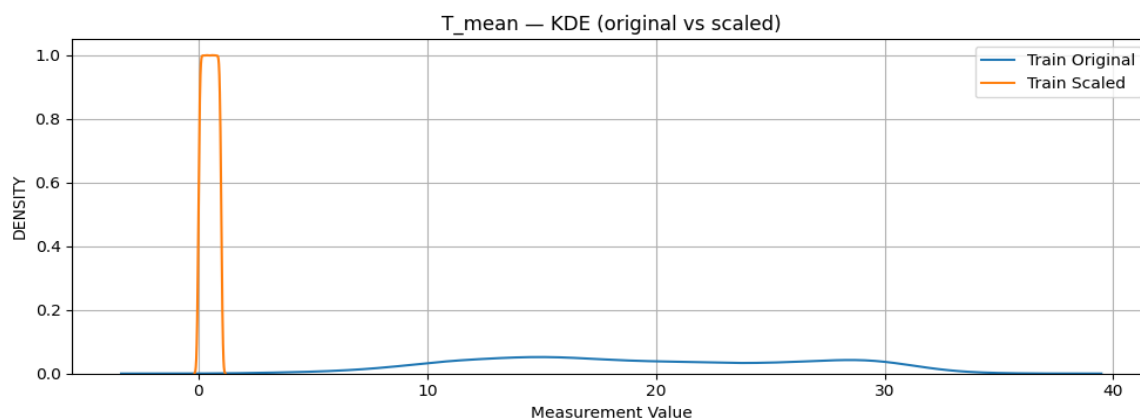
Εικόνα 11 Configuration 1 Time Series Overlay T_mean με StandarScaler



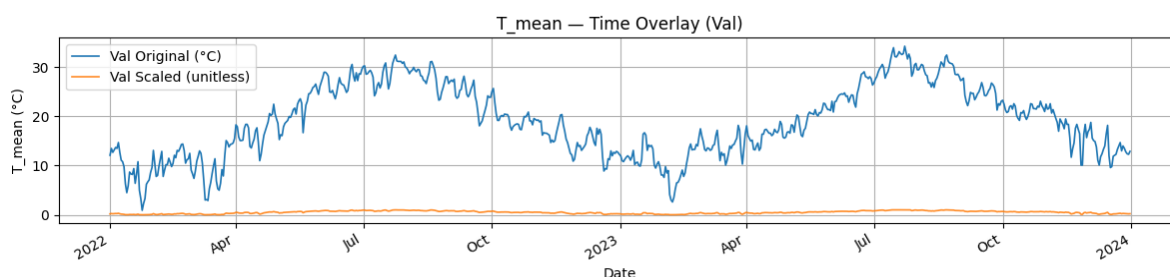
Εικόνα 12 Configuration 2 KDE T_mean με PowerTransformer(Yeo-Johnson)



Εικόνα 13 Configuration 2 Time Series Overlay T_mean με PowerTransformer(Yeo-Johnson)



Εικόνα 14 Configuration 3 KDE T_mean με QuantileTransformer (normal)



Εικόνα 15 Configuration 3 Time Series Overlay T_mean με QuantileTransformer (normal)

Στα διαγράμματα του Configuration 1 με χρήση του StandarScaler παρατηρούμε στο KDE δύο κορυφές στο διάγραμμα μετά την χρήση του scaler. Η μέση τιμή της νέας χρονοσειράς βρίσκεται κοντά στο μηδέν 0 και είναι η μέση τιμή που προκύπτει από την αρχική χρονοσειρά, έτσι δημιουργούνται δύο κορυφές γύρω από την μέση τιμή. Η μια κορυφή βρίσκεται στην τιμή 0,37 και η δεύτερη στην τιμή 0,31. Οι δύο αυτές κορυφές μας δηλώνουν δύο συγκεντρώσεις τιμών. Στο Time Series Overlay παρατηρούμε πως οι κυμάνσεις της χρονοσειράς μετά την εφαρμογή του scaler, ακολουθεί αρκετά την αρχική. Η χρονοσειρά μετατοπίζεται σε νέα κλίμακα με μηδενικό μέσο και μοναδιαία διασπορά. Γίνεται κατανοητό πως ο StandarScaler είναι γραμμικός και δεν αλλοιώνεται η χρονική πληροφορία.

Στα διαγράμματα του Configuration 2 με χρήση του PowerTransformer(Yeo-Johnson) παρατηρούμε στο KDE, μετατόπιση της μέσης τιμής της νέας χρονοσειράς. Έτσι η κατανομή γίνεται πιο συγκεντρωμένη γύρω από την μέση τιμή. Πλέον η μια κορυφή βρίσκεται στην τιμή 0,36 ενώ η δεύτερη στην τιμή 0,34. Στο διαγραμμα TimeSeriesOverlay παρατηρείται πως η γενική μορφή της νέας κυματομορφής είναι αναγνωρίσιμη, αλλά υπάρχει εξομάλυνση στις διακυμάνσεις των τιμών καθώς περιορίζονται οι ακραίες τιμές. Η χρονική δομή παραμένει,

ωστόσο η ένταση των μεταβολών έχει μειωθεί. Αυτό συνέβει καθώς ο Yeo-Johnson μειώνει την λοξότητα και εξισορροπεί τις διακυμάνσεις, διατηρώντας τα χρονικά μοτίβα.

Εκ πρώτης όψης, οι δύο scalers έχουν αρκετά κοινά και η κύρια διαφορά τους είναι πως ο PowerTransformer δίνει μια πιο ομαλή και συμμετρική κατανομή εξισορροπώντας τις διακυμάνσεις.

Στα διαγράμματα του Configuration 3 με χρήση του QuantileTransformer (normal) παρατηρούνται αρκετά διαφορετικές κυματομορφές σε σύγκριση με τους δύο προηγούμενους scalers. Αυτό συμβαίνει καθώς ο scaler μετατρέπει την κατανομή σε Gaussian μορφή. Στο KDE παρατηρείται μια υψηλή και στενή κορυφή, χαρακτηριστικό της επιβολής μιας ιδανικής κανονικής κατανομής με την χρήση quantile mapping. Στο διάγραμμα TimeSeriesOverlay παρατηρείται πως δεν υπάρχει αντιστοιχία της νέας με την αρχική κυματομορφή. Ο scaler έχει εξαλείψει την πολυτροπικότητα και τις ουρές και συγκεντρώνει την πυκνότητα των τιμών γύρω από το μηδέν. Αν και αρκετά ομογενοποιημένη και συμμετρική η κατανομή η οποία βοηθά στην εκπαίδευση του νευρωνικού δικτύου, ωστόσο δεν ανταποκρίνεται στην αρχική κυματομορφή και στην ρεαλιστική απεικόνιση της θερμοκρασίας.

Προκειμένου να διαπιστώσουμε ποιος συνδυασμός scaler και optimizer είναι ο κατάλληλος βάση των μετρικών MAE/MSE τοποθετήθηκαν οι τιμές των μετρικών σε τρεις πίνακες ανά optimizer.

Παράμετροι	Configuration 1		Configuration 2		Configuration 3	
	MAE	MSE	MAE	MSE	MAE	MSE
T_mean	0.1584	0.0449	0.1579	0.0433	0.1699	0.047
T_max	0.2177	0.0766	0.2153	0.0756	0.2181	0.0784
T_min	0.2007	0.0654	0.1762	0.0507	0.1771	0.0513
RH_mean	0.1154	0.0146	0.0952	0.0136	0.0928	0.0142
RH_max	0.1232	0.0149	0.095	0.0139	0.0913	0.015
RH_min	0.1098	0.0174	0.1027	0.0164	0.1021	0.0179
Prs_mean	0.3351	0.1894	0.3246	0.1772	0.3181	0.1728
Prs_max	0.3256	0.1948	0.3345	0.2018	0.3226	0.1915
Prs_min	0.3313	0.1829	0.3262	0.1824	0.3234	0.1814
Ac_R	0.0818	0.0169	0.988	0.0182	0.0901	0.0174
WS_mean	0.5076	0.4041	0.5256	0.4282	0.5085	0.4393
WG	0.4737	0.3245	0.487	0.3824	0.4745	0.368
DWD_sin	0.4945	0.3181	0.5021	0.3368	0.5033	0.3294
DWD_cos	0.5034	0.4186	0.503	0.4033	0.5187	0.4084

Πίνακας 2 Optimizer Adam MAE/MSE

Παράμετροι	Configuration 1		Configuration 2		Configuration 3	
	MAE	MSE	MAE	MSE	MAE	MSE
T_mean	0.1584	0.0449	0.1579	0.0433	0.1699	0.047
T_max	0.2177	0.0766	0.2153	0.0756	0.2181	0.0784
T_min	0.2007	0.0654	0.1762	0.0507	0.1771	0.0513
RH_mean	0.1154	0.0146	0.0952	0.0136	0.0928	0.0142
RH_max	0.1232	0.0149	0.095	0.0139	0.0913	0.015
RH_min	0.1098	0.0174	0.1027	0.0164	0.1021	0.0179
Prs_mean	0.3351	0.1894	0.3246	0.1772	0.3181	0.1728
Prs_max	0.3256	0.1948	0.3345	0.2018	0.3226	0.1915
Prs_min	0.3313	0.1829	0.3262	0.1824	0.3234	0.1814
Ac_R	0.0818	0.0169	0.988	0.0182	0.0901	0.0174
WS_mean	0.5076	0.4041	0.5256	0.4282	0.5085	0.4393
WG	0.4737	0.3245	0.487	0.3824	0.4745	0.368
DWD_sin	0.4945	0.3181	0.5021	0.3368	0.5033	0.3294
DWD_cos	0.5034	0.4186	0.503	0.4033	0.5187	0.4084

Πίνακας 3 Optimizer RMSprop MAE/MSE

Παράμετροι	Configuration 1		Configuration 2		Configuration 3	
	MAE	MSE	MAE	MSE	MAE	MSE
T_mean	0.1594	0.0436	0.1632	0.0467	0.1642	0.0464
T_max	0.2218	0.0712	0.2156	0.0758	0.2319	0.089
T_min	0.1834	0.0533	0.1849	0.0536	0.1789	0.0676
RH_mean	0.094	0.0142	0.0952	0.0144	0.0987	0.0157
RH_max	0.0984	0.0143	0.0933	0.0139	0.0937	0.0139
RH_min	0.104	0.0171	0.1142	0.0201	0.1025	0.0164
Prs_mean	0.3102	0.1679	0.3249	0.1782	0.3204	0.1761
Prs_max	0.3243	0.1906	0.333	0.1994	0.3402	0.2006
Prs_min	0.3199	0.1788	0.3219	0.1783	0.326	0.1869
Ac_R	0.091	0.018	0.862	0.0183	0.0819	0.0171
WS_mean	0.49	0.4143	0.5057	0.4294	0.4987	0.424
WG	0.4704	0.3629	0.4783	0.3724	0.4766	0.3669
DWD_sin	0.5201	0.4319	0.5231	0.4066	0.511	0.4452
DWD_cos	0.5187	0.4091	0.5266	0.4059	0.5176	0.4135

Πίνακας 4 Optimizer Nadam MAE/MSE

Από τα δεδομένα των παραπάνω πινάκων, προκύπτει ο επόμενος πίνακας με τις βέλτιστες μετρήσεις MAE και MSE. Σκοπός είναι να εντοπιστούν οι μικρότερες τιμές των μετρικών ανά παράμετρο και να καθοριστεί ο βέλτιστος scaler και optimizer για αυτές. Οι τιμές που επιλέχθηκαν δηλώνουν τις μικρότερες μέσες αποκλίσεις (MAE) από τις πραγματικές τιμές καθώς και η μικρότερη τετραγωνική απόκλιση (MSE) ώστε να αποφεύγονται τα μεγάλα λάθη. Με αυτόν τον τρόπο το μοντέλο θα γίνει πιο σταθερό και ακριβές.

Parameter	Scaler	Optimizer
Temperature	PowerTransformer (Yeo-Johnson)	Adam
Relative Humidity	PowerTransformer (Yeo-Johnson)	Adam
Pressure	QuantileTransformer(normal)	Adam
Rain	Log1p + MinMaxScaler	Adam
Wind	RobustScaler	Adam

Πίνακας 5 Τελικοί Scalers και Optimizer

6.4 Υλοποίηση Κώδικα

Στην προηγούμενη ενότητα διερευνήθηκαν και εντοπίστηκαν οι βέλτιστες επιλογές των scalers ανά παράμετρο και η καλύτερη εφαρμογή optimizer. Σε αυτή την ενότητα θα διερευνηθούν και θα αναλυθούν οι παράμετροι του LSTM μοντέλου, θα συγκριθούν τα αποτελέσματα από την εφαρμογή διαφόρων παραμέτρων και τέλος θα παρατεθούν τα αποτελέσματα του επιλεγμένου μοντέλου καιρικής πρόγνωσης.

6.4.1 Κώδικας Καιρικής Πρόγνωσης

Αρχικά θα αναλυθεί και θα παρουσιαστεί ο τελικός κώδικας της διπλωματικής, προτού παρουσιαστούν και αναλυθούν η αρχιτεκτονική και οι παράμετροι του LSTM. Ο τελικός κώδικας έχει την ίδια δομή με τον προηγούμενο, καθώς και στους δύο τα κύρια σημεία τους είναι κοινά. Και σε αυτή την περίπτωση οι διαδικασίες είναι χωρισμένες σε functions και η διαχείριση τους γίνεται μέσα από την μέθοδο main(). Οι κοινές μέθοδοι αφορούν την είσοδο της αρχικής χρονοσειράς, διαχωρισμός σε τρία dataset τα train, validation και test, ορίζονται και εφαρμόζονται οι scalers στα dataset, Ορίζεται το περιθώριο παραθύρου, πραγματοποιείται

αρχικοποίηση και εκπαίδευση του LSTM με τα νέα scaled dataset. Η μέθοδος `train_model()` έχει υλοποιηθεί πιο αποτελεσματικά και είναι η εξής:

```
def train_model(X_train, y_train, X_val, y_val, epochs, batch_size):  
    model = build_model((X_train.shape[1], X_train.shape[2]))  
    early_stop = EarlyStopping(monitor='val_loss', patience=8,  
restore_best_weights=True)  
    history = model.fit(  
        X_train, y_train,  
        validation_data=(X_val, y_val),  
        epochs=epochs,  
        batch_size=batch_size,  
        callbacks=[early_stop],  
        verbose=1  
    )  
    return model, history
```

Η μέθοδος αυτή καθορίζει την εκπαίδευση του μοντέλου, χρησιμοποιώντας τις παραμέτρους `epochs`, `batch_size` και `patience` να συμβάλλουν καθοριστικά στην διαδικασία. Η παράμετρος `epochs` είναι ο αριθμός των περασμάτων που θα κάνει το νευρωνικό δίκτυο πάνω σε όλο το set. Όσο μεγαλύτερη η τιμή του `epochs`, τόσο περισσότερα τα περάσματα (update) που θα γίνουν στο δίκτυο και τόσο περισσότερο θα μαθαίνει το μοντέλο. Χρειάζεται όμως μεγάλη προσοχή καθώς μπορεί να προκύψει ζήτημα *overfitting*, δηλαδή το μοντέλο να φτάσει και να ξεπεράσει το σημείο εκείνο που το μοντέλο μαθαίνει σε μεγάλο βαθμό το validation set εις βάρος του training set οδηγώντας το σε μεγάλες αποκλίσεις και αστοχία στα τελικά αποτελέσματα. Η επόμενη βασική παράμετρος είναι το `batch_size` το οποίο καθορίζει πόσα δείγματα από το dataset θα τρέξουν στο κάθε πέρασμα του μοντέλου. Είναι άρρηκτα συνδεδεμένο με την παράμετρο του `epochs` καθώς οι αριθμοί των updates ανά epoch και ο συνολικός αριθμός των update εξαρτάται από τις τιμές των δύο αυτών παραμέτρων. Πιο συγκεκριμένα ο αριθμός των updates ανά epoch υπολογίζεται ως εξής:

$$\text{Steps per epoch} = \frac{\text{number of samples}}{\text{batch size}}$$

όπου number of samples είναι η τιμή της παραμέτρου look_back η οποία ορίζεται στην μέθοδο create_dataset_multi_form_df(),

Και ο συνολικός αριθμός updates υπολογίζεται ως:

$$\text{Total updates} = \text{epochs} * \text{steps per epochs}$$

Όσο μεγαλύτερη η τιμή του batch_size τόσο λιγότερα updates ανά epoch θα γίνουν και η μάθηση θα είναι σταθερή και αργή. Εν αντιθέτως, όσο μικρότερη η τιμή του batch size, τόσο περισσότερα τα updates ανά epoch, άρα και πολλά μικρότερα βήματα με αποτέλεσμα η εκπαίδευση γίνεται πιο θορυβώδης αλλά υπάρχει καλύτερη γενίκευση.

Στην μέθοδο αυτή γίνεται και κλήση του EarlyStopping των βιβλιοθηκών Keras. Το early stop είναι ένας μηχανισμός call back ο οποίος παρακολουθεί την διαδικασία εκπαίδευσης του μοντέλου και κρίνει βάση της τιμής του val_loss/train_loss, πότε θα σταματήσει η εκπαίδευση ώστε να αποφευχθεί το overfitting. Η παράμετρος patience δηλώνει πόσα epochs θα προηγηθούν ώστε να σταματήσει αυτόματα η εκπαίδευση αν αυξάνεται η τιμή του val_loss/train_loss. Συνεπώς και αυτή η παράμετρος είναι κομβικής σημασίας για την εκπαίδευση του LSTM, καθώς μια μεγάλη τιμή αυτής μπορεί να οδηγήσει σε λανθασμένες τιμές λόγω overfitting. Εν συνεχεία, με την κλήση restore_best_weights επιστρέφει τα καλύτερα βάρη που εμφανίστηκαν κατά την διάρκεια της εκπαίδευσης του μοντέλου.

Συνεχίζοντας την επεξήγηση με την μέθοδο inverse_transform_predictions() η οποία μετατρέπει τα dataset που εξάγονται μετά το LSTM, σε μετρήσιμες χρονοσειρές όπως οι αρχικές. Επί της ουσίας, μετατρέπει από τα scaled τα dataset σε μετρήσιμες τιμές την κάθε παράμετρο.

```
def inverse_transform_predictions(y_scaled, df_columns, fitted_scalers,
                                temp_cols, rh_cols, prs_cols, rain_col,
                                wind_cols):
    df_pred_scaled = pd.DataFrame(y_scaled, columns=df_columns)
    df_pred = df_pred_scaled.copy()
    df_pred[temp_cols] = fitted_scalers["temp"].inverse_transform(df_pred[temp_cols])
    df_pred[rh_cols] = fitted_scalers["rh"].inverse_transform(df_pred[rh_cols])
    df_pred[prs_cols] = fitted_scalers["prs"].inverse_transform(df_pred[prs_cols])
    if fitted_scalers["rain_mode"] == "log1p":
        df_pred[rain_col] = np.expm1(fitted_scalers["rain"].inverse_transform(df_pred[rain_col]))
    else:
        df_pred[rain_col] = fitted_scalers["rain"].inverse_transform(df_pred[rain_col])
    df_pred[wind_cols] = fitted_scalers["wind"].inverse_transform(df_pred[wind_cols])
    return df_pred
```

Η επόμενη μέθοδος `autoregressive_forecast()` η οποία παίρνει το τελευταίο παράθυρο και προβλέπει το επόμενο βήμα κάνοντας μια πρόβλεψη την φορά. Αφαιρεί την παλαιότερη τιμή και προσθέτει στο `window` την ένα τιμή και ξανά προβλέπει. Μέσω αυτής γίνεται η κλήση τη `inverse_transform_prediction()` ώστε να επιστρέψει τις πραγματικές τιμές ανά παράμετρο μετά την διαδικασία πρόβλεψης.

Η μέθοδος `plot_true_vs_pred()` δημιουργεί όλα τα διαγράμματα απεικόνισης της πραγματικής τιμής των χρονοσειρών και τον προβλεπόμενων τιμών που προέκυψαν με την εκτέλεση του κώδικα.

```
def plot_true_vs_pred(df_true, df_pred, dates=None,
                      title_prefix="Validation", figsize=(12, 3)):
    cols = df_true.columns.tolist()
```

```
n = len(cols)

plt.figure(figsize=(14, 3 * n))

for i, col in enumerate(cols):

    plt.subplot(n, 1, i+1)

    if dates is not None:

        plt.plot(dates, df_true[col].values, label=f"{col} (True)",
color='blue')

        plt.plot(dates, df_pred[col].values, label=f"{col} (Pred)",
color='red', alpha=0.8)

        plt.gca().xaxis.set_major_locator(mdates.AutoDateLocator())

plt.gca().xaxis.set_major_formatter(mdates.ConciseDateFormatter(mdates.AutoDateLocator()))

plt.gcf().autofmt_xdate()

    else:

        plt.plot(df_true[col].values, label=f"{col} (True)",
color='blue')

        plt.plot(df_pred[col].values, label=f"{col} (Pred)",
color='red', alpha=0.8)

        plt.title(f"{title_prefix} - {col}")

        plt.xlabel("Time")

        plt.ylabel("Parameter Value")

        plt.legend()

        plt.grid(True)

plt.tight_layout()

plt.show()
```

Τέλος παρατίθεται η μέθοδος main η οποία διαχειρίζεται όλη την ροή του κώδικα.

```
def main(file_path):

    temp_cols = ["T_mean", "T_max", "T_min"]

    rh_cols = ["RH_mean", "RH_max", "RH_min"]
```

```
prs_cols = ["Prs_mean", "Prs_max", "Prs_min"]

rain_col = ["Ac_R"]

wind_cols = ["WS_mean", "WG"]

df = load_data(file_path, date_col="Date")

df = add_wind_sincos(df, wind_col="DWD")

train_mask, val_mask, test_mask = split_by_date_masks(df,
date_col="Date")

print(f"Total rows: {len(df)}, Train: {train_mask.sum()}, Test:
{test_mask.sum()}, Val: {val_mask.sum()}")

config = get_default_scalers()

train_scaled, test_scaled, val_scaled, fitted_scalers =
fit_and_transform_scalers(
    df, train_mask, val_mask, test_mask, config,
    temp_cols=temp_cols, rh_cols=rh_cols, prs_cols=prs_cols,
    rain_col=rain_col, wind_cols=wind_cols, date_col="Date"
)

look_back = 21

X_train, y_train, _, _ =
create_dataset_multi_from_df(train_scaled, look_back=look_back,
date_col="Date")

X_val, y_val, _, val_target_dates =
create_dataset_multi_from_df(val_scaled, look_back=look_back,
date_col="Date")

X_test, y_test, _, test_target_dates =
create_dataset_multi_from_df(test_scaled, look_back=look_back,
date_col="Date")

if len(X_train) == 0:
    raise ValueError("Training set too small for the chosen
look_back.")
```

```
if len(X_test) == 0:
    raise ValueError("Test set too small for the chosen look_back.")
if len(X_val) == 0:
    print("Warning: Validation set too small for windowing;
validation will be skipped in training.")

# Train using test as validation

model, history = train_model(X_train, y_train, X_val, y_val,
epochs=40, batch_size=32)

# Plot training vs validation loss
plt.figure(figsize=(10,5))
plt.plot(history.history['loss'], label='Train Loss')
if 'val_loss' in history.history:
    plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("LSTM Training Loss (Adam)")
plt.xlabel("Epoch")
plt.ylabel("MSE Loss")
plt.legend()
plt.grid(True)
plt.show()

# Evaluate test set
y_pred_test = model.predict(X_test, verbose=0)
mae_per_param_test = np.mean(np.abs(y_pred_test - y_test), axis=0)
mse_per_param_test = np.mean((y_pred_test - y_test)**2, axis=0)

results = {
    "columns": columns,
    "mae_per_param_test": mae_per_param_test,
```

```
    "mse_per_param_test": mse_per_param_test,
    "mae_total_test": np.mean(mae_per_param_test),
    "mse_total_test": np.mean(mse_per_param_test),
    "history": history
}

print("\nTest MAE per variable:")
for col, val in zip(columns, mae_per_param_test):
    print(f"    {col}: {val:.4f}")
print(f"Total MAE (test): {results['mae_total_test']:.4f}")

# If validation windows exist do inverse-transform
if len(X_test) > 0:
    y_pred_test = model.predict(X_test, verbose=0)

    df_pred_test = inverse_transform_predictions(
        y_pred_test, columns, fitted_scalers,
        temp_cols, rh_cols, prs_cols, rain_col, wind_cols
    )

    df_true_test = inverse_transform_predictions(
        y_test, columns, fitted_scalers,
        temp_cols, rh_cols, prs_cols, rain_col, wind_cols
    )

    plot_true_vs_pred(df_true_test, df_pred_test,
dates=test_target_dates, title_prefix="Prediction")

    results.update({
        "df_pred_test": df_pred_test,
        "df_true_test": df_true_test
    })
```

```
else:

    print("No test windows to plot.")

return results
```

Η κεντρική μέθοδος `main()` φορτώνει τα δεδομένα και καλεί όλες τις απαραίτητες μεθόδους για διαχωρισμό του αρχικού dataset (train, validation και test) και κανονικοποίησης αυτών. Δημιουργεί το window του LSTM καθώς και ορίζει την εκπαίδευση του καλώντας την `train_model` ορίζοντας ταυτόχρονα τις τιμές των `epoch` και `batch_size`. Αξιολογεί το μοντέλο LSTM και καλεί την `inverse_transform_prediction()` ώστε να επιστρέψουν οι προβλέψεις των παραμέτρων σε πραγματικές τιμές. Καλεί την `plot_true_vs_pred()` ώστε να εμφανιστούν όλα τα διαγράμματα και εκτελεί πρόγνωση 24 ωρών.

Η κλήση της `main()` γίνεται στις γραμμές:

```
if __name__ == "__main__":
    file_path = "/content/drive/MyDrive/Διπλωματική ΕΑΠ Κιούσης/lab
files/athens.csv"
    results = main(file_path)
```

Με την χρήση των γραμμών αυτών, εκτελείται η `main()` και κατ' επέκταση ολόκληρος ο κώδικας. Επίσης διασφαλίζει πως αν εισαχθεί ο κώδικας σε άλλο script, δεν θα τρέξει αυτόματα ο κώδικας.

6.4.2 Σύγκριση Αρχιτεκτονικών LSTM

Αρχικά υλοποιήθηκε σε μια μεμονωμένη function το μοντέλο LSTM που χρησιμοποιήθηκε στον κώδικα.

```
def build_model(input_shape):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        layers.LSTM(64, return_sequences=True),
```

```
layers.BatchNormalization(),
layers.LSTM(32, return_sequences=False),
layers.BatchNormalization(),
layers.Dense(16, activation='relu'),
layers.Dense(input_shape[1])
])
model.compile(optimizer='adam', loss='mse', metrics=['mae'])
return model
```

Η αρχιτεκτονική που επιλέχθηκε, περιλαμβάνει δύο επίπεδα LSTM και ένα Dense με χρήση Batch Normalization ανάμεσα σε αυτά.

Για την επιλογή των τελικών επιπέδων του LSTM μοντέλου που χρησιμοποιήθηκε στον κώδικα, προέκυψε από την σύγκριση των παραμέτρων Train και Validation Loss με μοντέλα που περιελάμβαναν τα εξής επίπεδα $128 \rightarrow 64 \rightarrow 32$, $32 \rightarrow 64 \rightarrow 32$, $64 \rightarrow 32 \rightarrow 64$, και το $64 \rightarrow 32 \rightarrow 16$.

Παρατίθενται παρακάτω τα μοντέλα LSTM με τα διαγράμματα τους.

Χρήση μοντέλου LSTM με επίπεδα: **$128 \rightarrow 64 \rightarrow 32$**



Εικόνα 16 LSTM $128 \rightarrow 64 \rightarrow 32$

Σε αυτή την περίπτωση το μοντέλο μαθαίνει και γενικεύει σε ικανοποιητικό επίπεδο. Καθώς όμως η αρχική του χωρητικότητα είναι 128 units, έχει ως αποτέλεσμα να υπάρχει μεγαλύτερο variance και να δημιουργούνται αρκετά spikes στο validation loss. Για τον λόγο αυτό το συγκεκριμένο μοντέλο θεωρείται ευαίσθητο.

Χρήση μοντέλου LSTM με επίπεδα: **32 → 64 → 32**



Εικόνα 17 LSTM 32 → 64 → 32

Από το διάγραμμα, παρατηρούμε ότι το μοντέλο έχει σταθερή εκπαίδευση με φθίνουσα πορεία (Train Loss) και σταθεροποίηση στο ~0.18, που σημαίνει ότι μαθαίνει χωρίς προβλήματα τα patterns της χρονοσειράς. Αντίστοιχα το Validation Loss έχει την ίδια συμπεριφορά συνεπώς υπάρχει σωστή γενίκευση, αν και υπάρχουν μικρές ασταθείς διακυμάνσεις στην περιοχή ~0.18-0.20.

Χρήση μοντέλου LSTM με επίπεδα: **64 → 32 → 64**



Εικόνα 18 LSTM 64 → 32 → 64

Το δίκτυο αυτό δείχνει μια πολύ ομαλή και σταθερή μάθηση με τα Train και Validation loss να σταθεροποιούνται στα 15 epochs. Δεν είναι όμως η ιδανική δομή για το LSTM καθώς θέλουμε να υπάρχει σύγκλιση στο τελευταίο επίπεδο.

Χρήση μοντέλου LSTM με επίπεδα: **64 → 32 → 16**



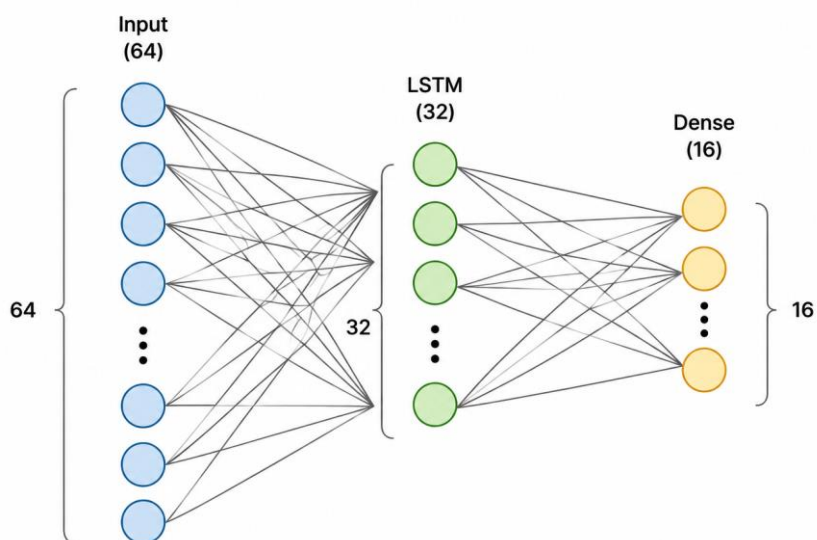
Εικόνα 19 LSTM 64 → 32 → 16

Στο μοντέλο αυτό, εμφανίζεται ομαλότητα και στο train loss και στο validation loss με σταθεροποίηση στις τιμές κοντά στο ~ 0.18 . Το μοντέλο είναι ισορροπημένο με σταθερή μάθηση και χωρίς αστάθειες, καθιστώντας το έτσι κατάλληλο για χρήση στην παρούσα εργασία.

Η αρχιτεκτονική του νευρωνικού δικτύου που επιλέχθηκε είναι η $64 \rightarrow 32 \rightarrow 16$ καθώς με την χρήση αυτής το δίκτυο είναι αρκετά σταθερό, δεν υπερ προσαρμόζεται και δεν αλλοιώνεται η μνήμη του.

Πιο αναλυτικά, το πρώτο επίπεδο LSTM με 64 νευρώνες, επεξεργάζεται ολόκληρη την ακολουθία και επιστρέφει μια νέα ακολουθία ίδιου μήκους. Με τον τρόπο αυτό το δίκτυο έχει την ικανότητα να μάθει χρονικά μοτίβα μεγάλης διάρκειας και να εξάγει ένα μεγάλο πλήθος αναπαραστάσεων από τα δεδομένα. Το επόμενο LSTM επίπεδο με 32 μονάδες, λαμβάνει την ακολουθία και την συμπυκνώνει σε ένα μόνο διάνυσμα, αυτό συμβαίνει γιατί το `return_sequence` είναι False. Έτσι, το δίκτυο δημιουργεί μια συνοπτική αλλά πλούσια αναπαράσταση του χρονικού παραθύρου. Τέλος το επίπεδο Dense με 16 νευρώνες και ενεργό ReLU, λειτουργεί ως μη γραμμικός μετασχηματιστής της αναπαράστασης που δημιουργήθηκε από το LSTM. Το επίπεδο αυτό επιτρέπει στο μοντέλο να μάθει αρκετά σύνθετες σχέσεις μεταξύ των χαρακτηριστικών και βελτιώνει την πρόβλεψη. Οι νευρώνες του Dense επιπέδου (16) έχουν το ίδιο πλήθος με τις μεταβλητές εξόδου. Ενδιάμεσα των επιπέδων, πραγματοποιείται Batch Normalization ώστε να υπάρξει σταθεροποίηση στην κατανομή των ενεργοποιήσεων και να επιταχυνθεί η εκπαίδευση. Η διαδικασία ολοκληρώνεται με την χρήση του Adam optimizer και των μετρικών MSE και MAE. Με τον τρόπο αυτό ο βελτιστοποιητής Adam ρυθμίζει την προσαρμογή των βαρών, το MSE διαμορφώνει τον στόχο της εκπαίδευσης και το MAE παρέχει πρακτική και σταθερή μέτρηση της πραγματικής απόδοσης του μοντέλου.

Καθώς δεν είναι εκτενές δίκτυο και στην προκειμένη περίπτωση έχουμε υπολογισμό αρκετών μεταβλητών, η μορφή αυτή του δικτύου είναι η κατάλληλη έτσι ώστε να μην εκπαιδευτεί με λάθος συσχετίσεις, να διατηρεί μικρό bias και variance ώστε να γίνεται καλύτερη γενίκευση ενώ ταυτόχρονα να μην υπερ προσαρμοστεί καμία μεταβλητή. Δεν χρησιμοποιήθηκαν Dropouts τα οποία μπορεί να αλλοιώσουν την μνήμη του LSTM και θα υπάρξει λανθασμένη εκπαίδευση αυτού. Συνεπώς το δίκτυο μας φέρει την παρακάτω μορφή.



Εικόνα 20 Απεικόνιση LSTM

6.4.3 Παραμετροποίηση LSTM

Επόμενο βήμα πριν την ολοκλήρωση του LSTM μοντέλου, είναι η ορθή παραμετροποίηση του. Στην υποενότητα αυτή θα καθοριστούν οι τελικές τιμές των παραμέτρων `look_back`, `epochs`, `batch_size` και `patience`, που όπως αναφέρθηκε και στην ενότητα 6.3.2 είναι παράμετροι κομβικής σημασίας για την ομαλή και σταθερή εκπαίδευση και λειτουργία του μοντέλου.

Σύνολο	1	2	3	4	5	6	7	8	9
look_back	7	21	21	21	42	21	21	14	7
epochs	100	100	40	40	40	40	40	40	40
batch_size	16	8	24	8	32	16	32	16	8
patience	10	10	8	8	8	8	8	8	8
run epochs	29	38	28	32	34	30	29	33	30
T_mean	0.1717	0.1783	0.1832	0.1723	0.1646	0.1707	0.1625	0.169	0.1713
T_max	0.208	0.2015	0.2075	0.2064	0.2011	0.2066	0.201	0.2017	0.2104
T_min	0.1898	0.1886	0.1903	0.1808	0.1682	0.176	0.1681	0.1704	0.1781
RH_mean	0.1334	0.1363	0.1288	0.1341	0.1306	0.1381	0.1222	0.1319	0.1335
RH_max	0.1423	0.1467	0.1413	0.1439	0.1355	0.1425	0.1293	0.1332	0.1465
RH_min	0.1464	0.1418	0.1381	0.1452	0.1502	0.1436	0.1628	0.1377	0.1525
Prs_mean	0.3463	0.3391	0.3458	0.3530	0.3527	0.3541	0.3387	0.3454	0.3558
Prs_max	0.357	0.3719	0.3652	0.3578	0.3636	0.3598	0.3409	0.3565	0.3679
Prs_min	0.3548	0.3676	0.3518	0.3649	0.3532	0.36	0.352	0.3582	0.354
Ac_R	0.0833	0.0936	0.0873	0.0836	0.0894	0.0891	0.0806	0.0815	0.0844
WS_mean	0.4917	0.5018	0.5059	0.4875	0.5191	0.4989	0.4872	0.4898	0.496
WG	0.4906	0.4956	0.4781	0.483	0.4815	0.4795	0.4757	0.4759	0.4773
DWD_sin	0.5317	0.541	0.5298	0.5455	0.523	0.5335	0.5078	0.5181	0.5168
DWD_cos	0.5507	0.5598	0.5561	0.5681	0.553	0.5615	0.5416	0.5477	0.5578

Πίνακας 6 Σύγκριση MAE ανά σύνολο παραμέτρων

Στον παρακάτω πίνακα αναγράφονται τα σύνολα που χρησιμοποιήθηκαν και οι τιμές των MAE. Για να καθοριστούν οι τιμές των παραμέτρων, εκτελέστηκε ο κώδικας με διαφορετικά σύνολα. Η σύγκριση για το βέλτιστο μοντέλο γίνεται με τις τιμές MAE ανά παράμετρο.

Από τον παραπάνω πίνακα γίνεται κατανοητό πως, με της χρήσης Early stopping το μοντέλο φτάνει τα 30 περίπου epoch ανά εκτέλεση. Αυτό σημαίνει πως επιτυγχάνεται η εκμάθηση του δικτύου μεταξύ 28 και 33 epochs. Από τα σημεία αυτά δεν υπάρχει βελτίωση στην εκπαίδευση, συνεπώς η τιμή της παραμέτρου epochs μπορεί με ασφάλεια να τεθεί στην τιμή 40. Η παράμετρος patience θα τεθεί στην τιμή 8, καθώς σταματά την εκπαίδευση του μοντέλου όταν για 8 συνεχόμενα epochs updates δεν υπάρξει βελτίωση του validation loss.

Παρατηρείται πως τα αποτελέσματα του μετρικού MAE ανά παράμετρο, επηρεάζονται περισσότερο από την τιμή του batch_size παρά από την τιμή του look_back. Συνεπώς η παράμετρος look_back ορίζεται με την τιμή 21, καθώς καθορίζει το windowing και προσμετρούνται οι 21 προηγούμενες μετρήσεις. Καθώς το dataset έχουμε μια μέτρηση ανά ημέρα, το windowing έχει περιθώριο 21 ημερών. Καθώς οι καταγραφές είναι σε βάθος δεκατριών ετών το παράθυρο αυτό δεν θεωρείται μεγάλο ώστε να αποκλίνει η εκπαίδευση του

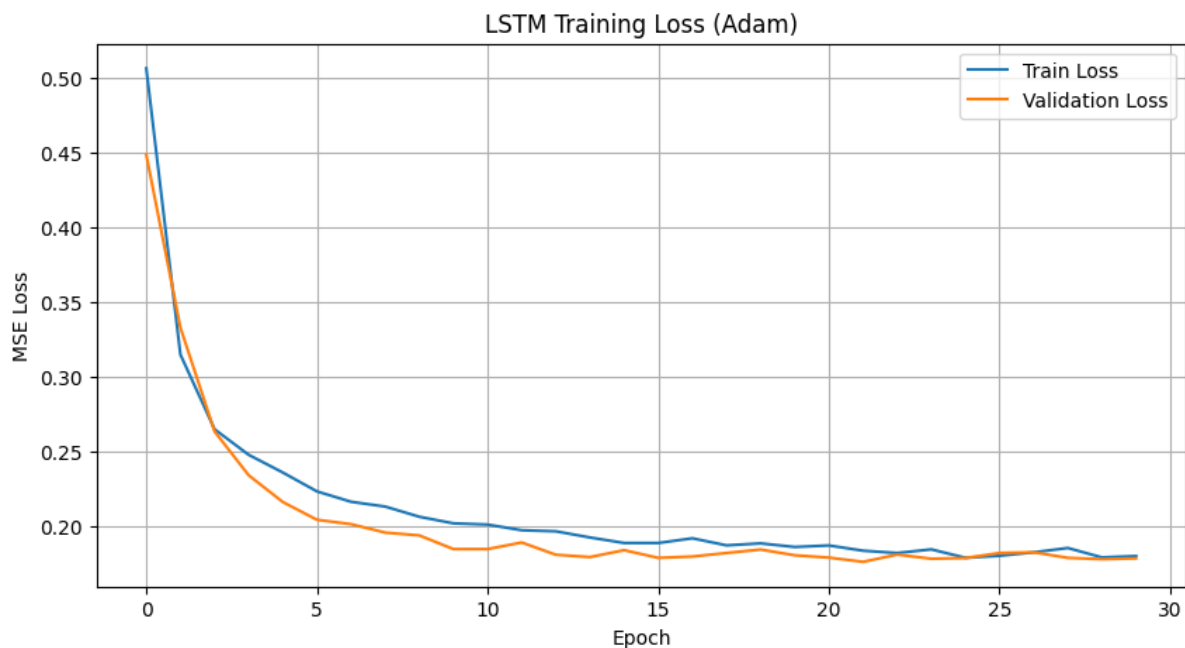
LSTM. Αυτό συμβαίνει καθώς η τιμή του `batch_size` καθορίζει πόσα δείγματα θα περάσουν στο κάθε epoch. Οι τιμές για την παράμετρο `batch_size` μεταξύ 16 και 32, έχουν τις χαμηλότερες τιμές MAE. Τα σύνολα με τα πιο ικανοποιητικά MAE είναι τα υπ' αριθμόν 5,6 και 7. Παρατίθενται τα διαγράμματα Train Loss για κάθε ένα από τα τρία σύνολα.



Εικόνα 21 Train Loss Σύνολο No. 5



Εικόνα 22 Train Loss Σύνολο No. 6



Εικόνα 23 Train Loss Σύνολο No. 7

Συγκρίνοντας τα τρία παραπάνω διαγράμματα, υπάρχει ομαλή εκπαίδευση και στα τρία σύνολα. Ωστόσο, το Validation Loss για τα σύνολα 5 και 6 παρουσιάζει αστάθεια με μικρές κορυφές. Η διαφορά στο πλήθος των epochs λόγω early stop, δεν επηρεάζει την εκπαίδευση και την συνολική απόδοση του μοντέλου, καθώς το σημείο που σταμάτησε η εκπαίδευση του, δηλώνει πως το μοντέλο δεν χρήζει περεταίρω μάθησης. Συνεπώς το ιδανικό σύνολο είναι το 7, αφενός για την ομαλή εκπαίδευση και σταθερή διακύμανση των train και validation loss και αφετέρου για τις χαμηλές τιμές στο μετρικό MAE, συγκριτικά με τα υπόλοιπα σύνολα.

Συνοψίζοντας οι παράμετροι παίρνουν τις εξής μετρήσεις

look_back = 21

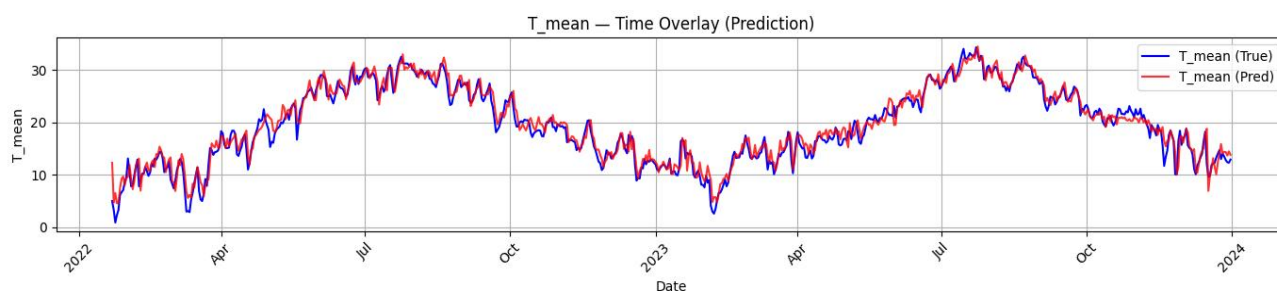
epochs = 40

batch_size = 32

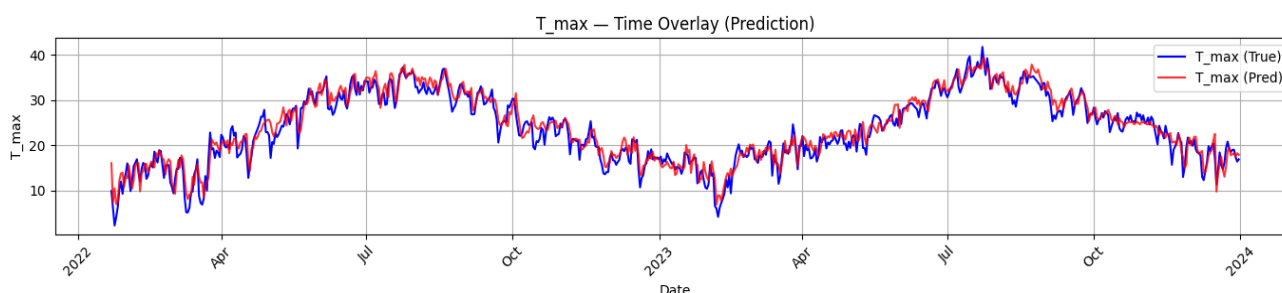
patience = 8

6.4.4 Πρόγνωση ανά παράμετρο

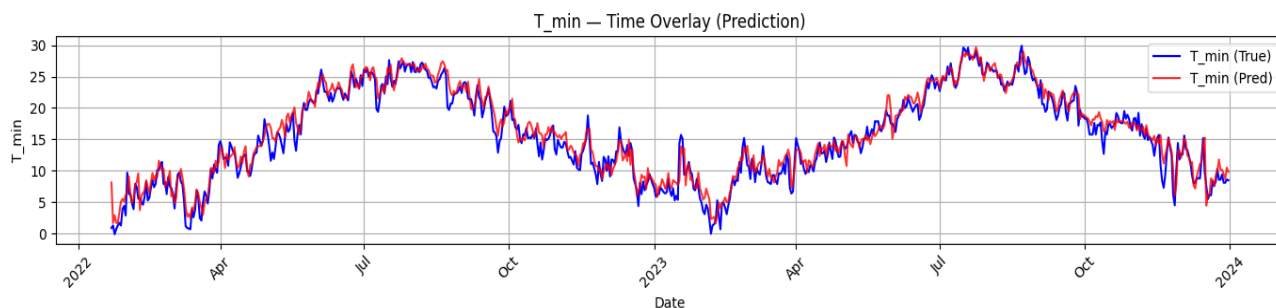
Έχοντας καθορίσει τα scalers και τους optimizer ανά παράμετρο, την αρχιτεκτονική και την παραμετροποίηση του μοντέλου LSTM που θα χρησιμοποιηθεί, πλέον παραμένει να ολοκληρωθεί η πρόβλεψη ανά παράμετρο και η σύγκριση των προβλεπόμενων τιμών με τις πραγματικές. Το μοντέλο έχει εκπαιδευτεί βάση του train dataset που περιέχει τις καταγραφές για τις ημερομηνίες από 1-1-2010 μέχρι 31-12-2019 και έχει ελεγχθεί με την χρήση του validation dataset περιέχει τις καταγραφές για τις ημερομηνίες από 1-1-2020 μέχρι 31-12-2021. Το test dataset με καταγραφές από 1-1-2022 έως 31-12-2023, χρησιμοποιήθηκε για να πραγματοποιηθεί η πρόγνωση ανά παράμετρο και με τις τιμές αυτού θα γίνει η σύγκριση. Η σύγκριση των αποτελεσμάτων του μοντέλου γίνεται με την χρήση διαγραμμάτων στα οποία απεικονίζονται η κυματομορφή των τιμών του test dataset και των τιμών που προέκυψαν από την πρόγνωση του LSTM.



Εικόνα 24 True vs Predicted Values Θερμοκρασίας (T_mean)

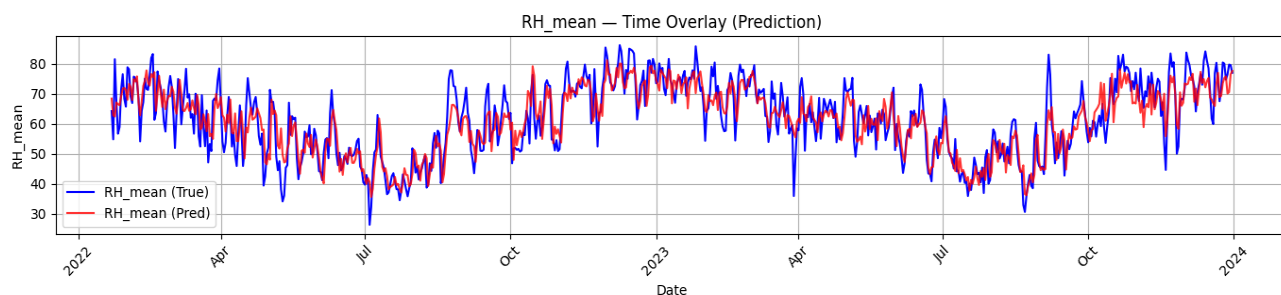


Εικόνα 25 True vs Predicted Values Θερμοκρασίας (T_max)

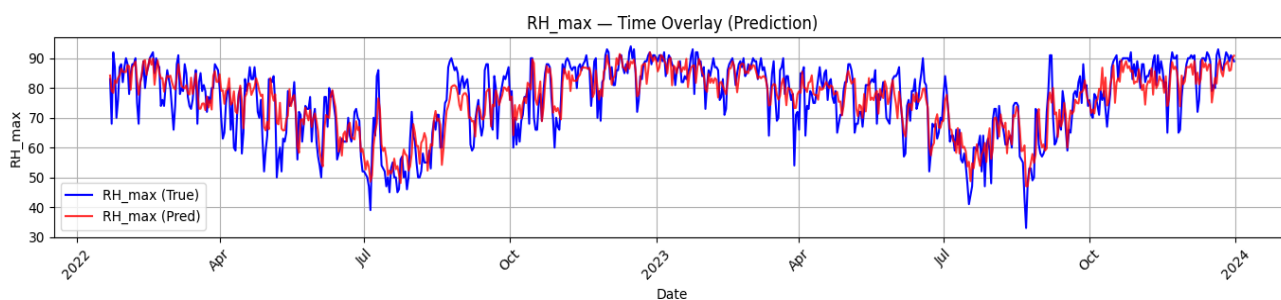


Εικόνα 26 True vs Predicted Values Θερμοκρασίας (T_{min})

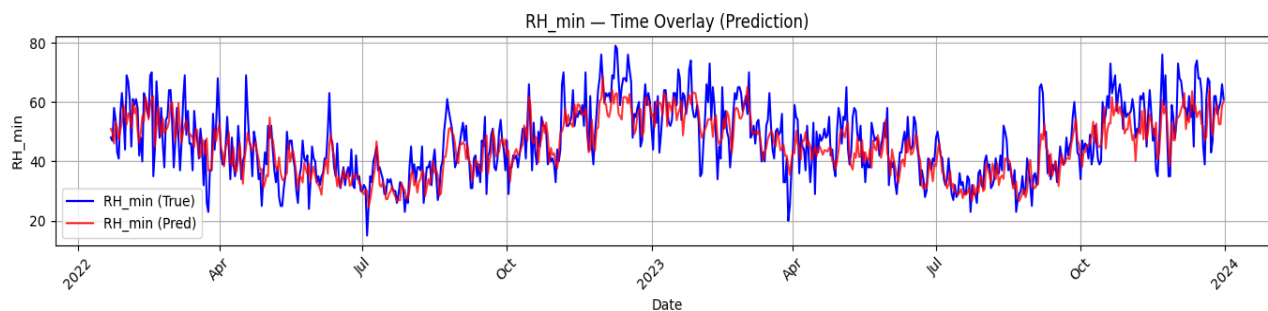
Τα παραπάνω γραφήματα δείχνουν τις πραγματικές τιμές και τις προβλεπόμενες, για τις τρεις παραμέτρους της θερμοκρασίας (Μέση, Μέγιστη, Ελάχιστη). Αποτυπώνεται πως το μοντέλο έχει συνεπή και σταθερή συμπεριφορά ως προς την συγκεκριμένη παράμετρο. Παρατηρείται πως η κυματομορφή των προβλεπόμενων τιμών ακολουθεί αυτή των πραγματικών. Συνεπώς, το μοντέλο έχει μάθει τις μακροχρόνιες τάσεις και περιοδικότητες των χρονοσειρών, με αποτέλεσμα μια αρκετά αξιόπιστη πρόγνωση.



Εικόνα 27 True vs Predicted Values Σχετικής Υγρασίας (RH_{mean})

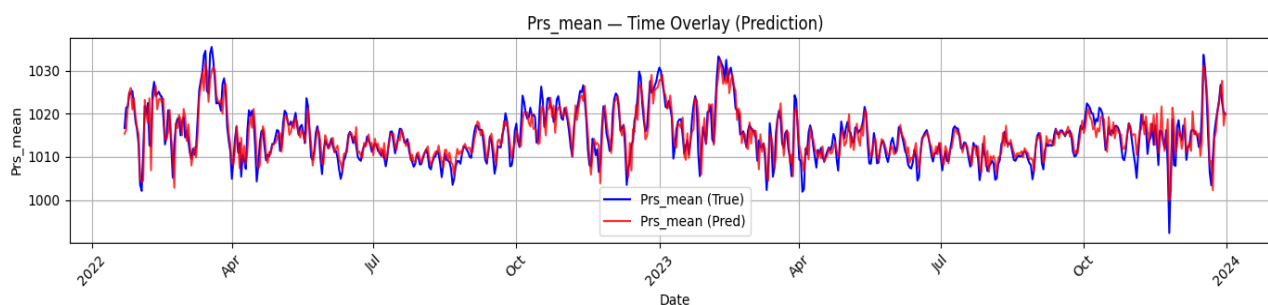


Εικόνα 28 True vs Predicted Values Σχετικής Υγρασίας (RH_{max})

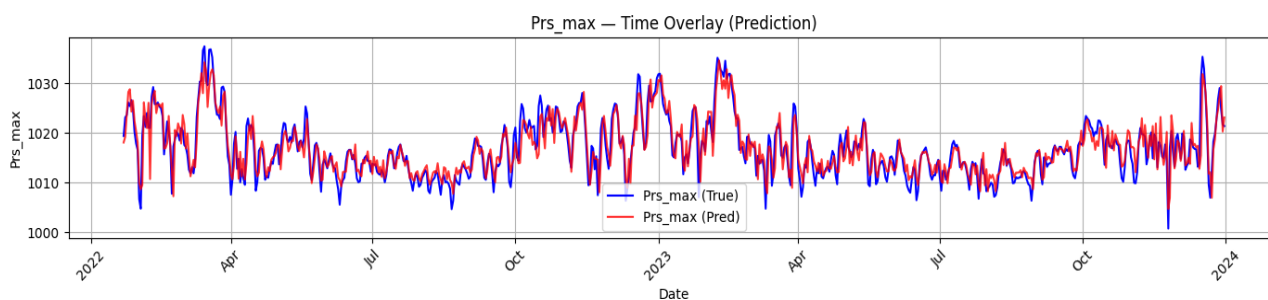


Εικόνα 29 True vs Predicted Values Σχετικής Υγρασίας (RH_min)

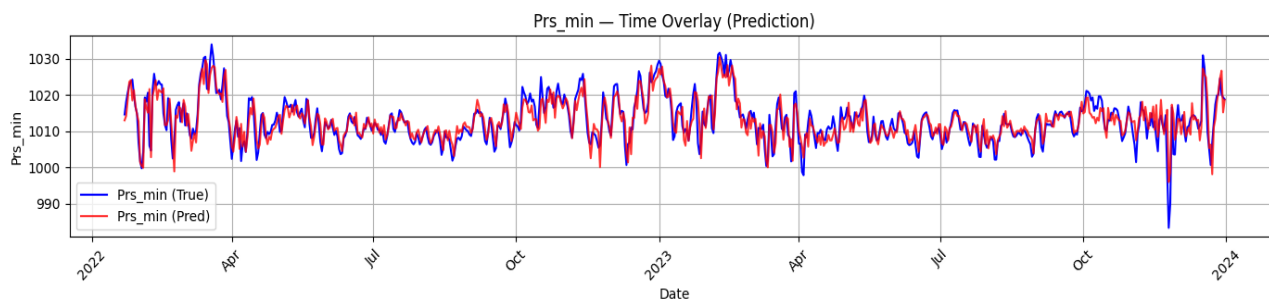
Η συμπεριφορά των τριών διαγραμμάτων της παραμέτρου, δηλώνει πως το μοντέλο έχει καταφέρει να αποδώσει την δυναμική της μεταβλητής σε όλες τις εκδοχές της. Η χρονοσειρά της σχετικής υγρασίας χαρακτηρίζεται από ομαλότητα στις διακυμάνσεις της και σαφείς εποχικές διακυμάνσεις. Για τον λόγο αυτό το μοντέλο μαθαίνει αποτελεσματικά τη δομή της και η πρόβλεψη γίνεται με μεγάλη ακρίβεια.



Εικόνα 30 True vs Predicted Values Πίεσης (Prs_mean)

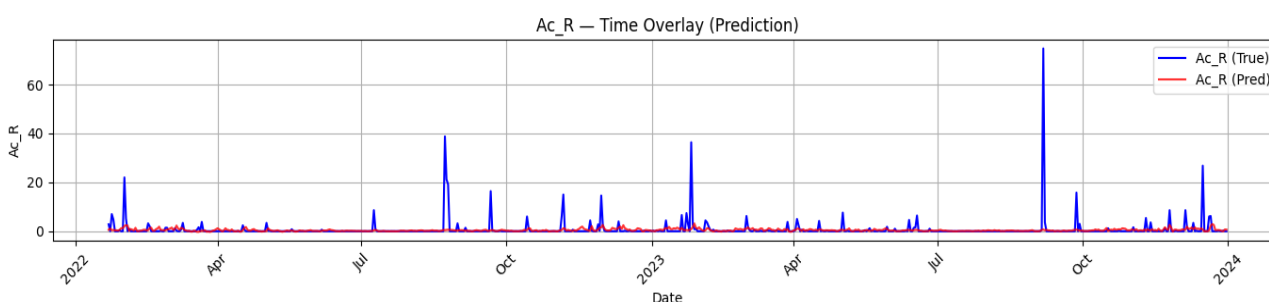


Εικόνα 31 True vs Predicted Values Πίεσης (Prs_max)



Εικόνα 32 True vs Predicted Values Πίεσης (Prs_min)

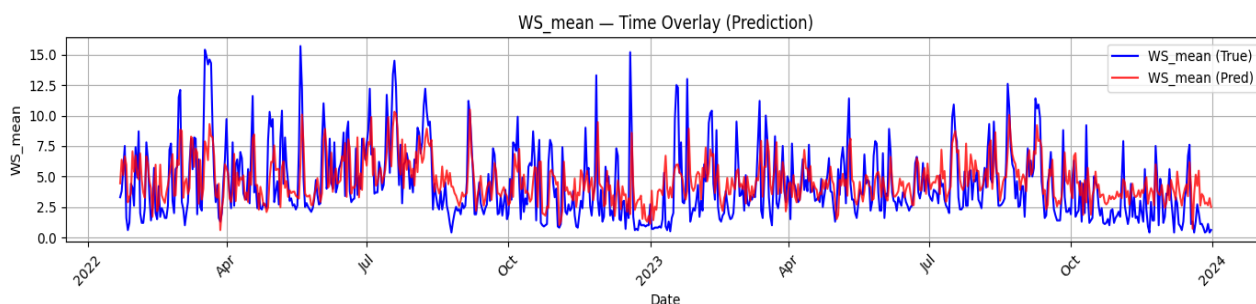
Όπως και στην περίπτωση των παραμέτρων της θερμοκρασίας, έτσι και στην παράμετρο της πίεσης, το μοντέλο έχει ανταπεξέλθει πλήρως να αποδώσει με συνέπεια τη δυναμική της μεταβλητής. Οι καταγραφές της πίεσης είναι από τις πιο σταθερές παραμέτρους με ελάχιστες και μικρές διακυμάνσεις. Για τον λόγο αυτό, καθίσταται σχετικά εύκολο από το μοντέλο να διαχειριστεί, να μάθει και να εκτελέσει μια ικανοποιητική πρόγνωση για την παράμετρο.



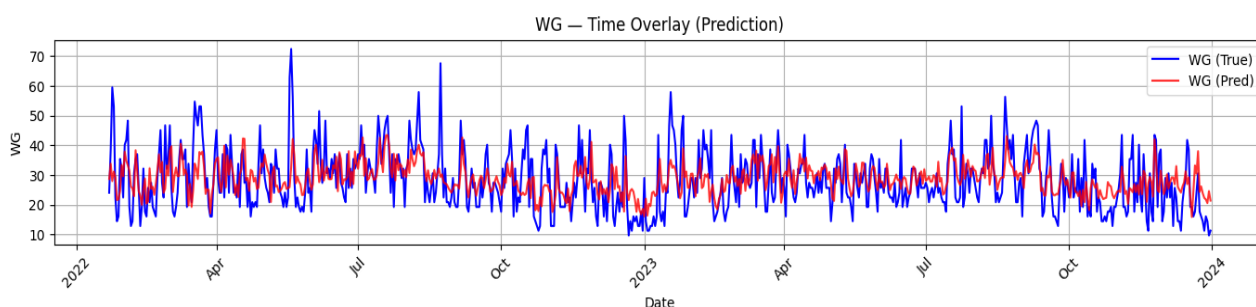
Εικόνα 33 True vs Predicted Values Βροχόπτωσης (Ac_R)

Το διάγραμμα της μεταβλητής βροχόπτωσης (Ac_R) αποτυπώνει την δυσκολία των συστημάτων πρόβλεψης με την χρήση νευρωνικών δικτύων τα οποία βασίζονται σε χρονοσειρές, για την συγκεκριμένη παράμετρο. Η αρχική χρονοσειρά παρουσιάζει ασυμμετρία με μεγάλα διαστήματα μηδενικών τιμών και σπάνιες απότομες αιχμές, όπως φαίνεται και στο διάγραμμα. Η απόκλιση ανάμεσα στις πραγματικές τιμές με τις υψηλές αιχμές και στις τιμές της πρόβλεψης, δεν υποδηλώνει αστοχία του μοντέλου. Αντιθέτως, το μοντέλο αφενός χρειάζεται πιο λεπτομερείς και μακροχρόνιες καταγραφές για να επιτευχθεί καλύτερη πρόγνωση. Επιπρόσθετα, η βροχόπτωση δεν εξαρτάται μόνο από τις τοπικές μετεωρολογικές μεταβλητές αλλά και από ευρύτερες συνθήκες. Με τα χρησιμοποιούμενα δεδομένα, η

συμπεριφορά και η πρόγνωση του μοντέλου είναι αναμενόμενη. Συνεπώς το γράφημα αναδεικνύει τη δομική αναντιστοιχία ανάμεσα στη φύση του φαινομένου και στο μοντέλο.

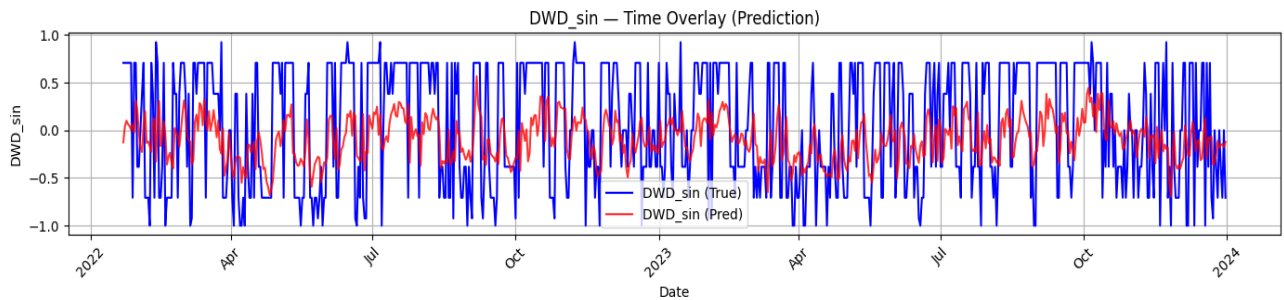


Εικόνα 34 True vs Predicted Values Ταχύτητας Ανέμου (WS_mean)

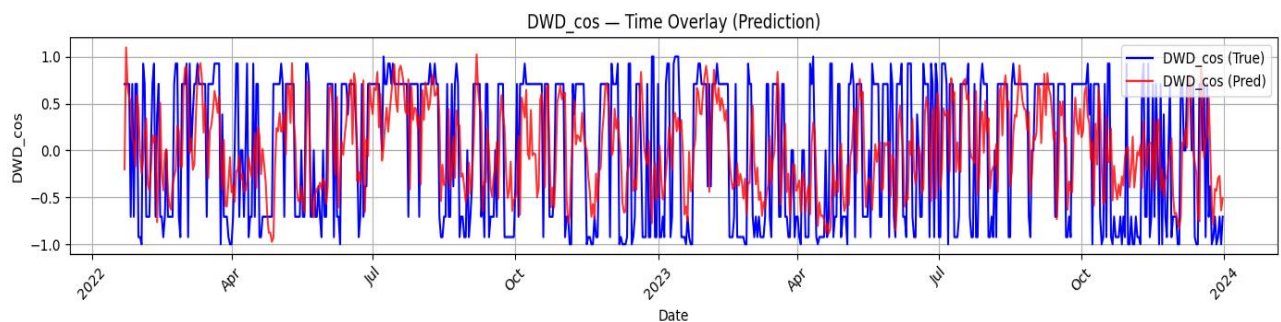


Εικόνα 35 True vs Predicted Values Ριπή Ανέμου (WG)

Οι παράμετροι της ταχύτητας και διεύθυνσης ανέμου, χαρακτηρίζονται από υψηλή και συχνή μεταβλητότητα, έντονο θόρυβο και περιορισμένη εποχικότητα, στοιχεία τα οποία καθιστούν την παράμετρο αρκετά απαιτητική και δύσκολη στην πρόβλεψη της. Η παράμετρος της ριπής ανέμου παρουσιάζει μεγαλύτερη μεταβλητότητα με απότομες κορυφές, καθιστώντας έτσι δυσκολότερη την διαδικασία πρόγνωσης από το μοντέλο. Ωστόσο, το μοντέλο δείχνει να έχει καταφέρει ως ένα βαθμό να ακολουθήσει την πορεία της κυματομορφής, αλλά να υπάρχει μια απόκλιση στις τιμές των κορυφών. Συνολικά τα διαγράμματα δείχνουν ότι το μοντέλο έχει επιτύχει μια ισορροπημένη απόδοση στις παραμέτρους, με την μέση ταχύτητα να αποδίδεται με μεγαλύτερη ακρίβεια απ' ό,τι η παράμετρος της ριπής ανέμου.



Εικόνα 36 True vs Predicted Values Διεύθυνση Ανέμου (DWD_sin)



Εικόνα 37 True vs Predicted Values Διεύθυνση Ανέμου (DWD_cos)

Οι κυματομορφές της διεύθυνσης ανέμου μέσω των συνιστωσών DWD_sin και DWD_cos , αποτελούν μια ιδιαίτερη περίπτωση καθώς μετατρέπουν την κυκλική μεταβλητή σε συνεχές και ομαλές χρονοσειρές. Τα διαγράμματα δείχνουν ότι το μοντέλο έχει καταφέρει να αποδώσει σε μεγάλο βαθμό τις δύο αυτές συνιστώσες ακολουθώντας τις πραγματικές τιμές χωρίς να παρουσιάζονται ιδιαίτερα σφάλματα. Η συμπεριφορά του DWD_sin δείχνει πως το μοντέλο έχει μάθει τις κυμάνεις της μεταβλητής μέσα στον χρόνο. Αντίστοιχα και το διάγραμμα του DWD_cos όπου στην περίπτωση αυτή η κυματομορφή της πρόγνωσης δείχνει να είναι καλύτερα προσαρμοσμένη. Συνολικά παρατηρείται πως προσέγγιση με ημιτονοειδείς και συνημιτονοειδείς συνιστώσες είναι αποτελεσματική για πρόβλεψη κυκλικών μεταβλητών.

7. Συμπεράσματα

Η παρούσα μεταπτυχιακή εργασία, διερεύνησε την ικανότητα ενός πολυμετάβλητου LSTM να προβλέπει βασικές μετεωρολογικές παραμέτρους αξιοποιώντας ιστορικά δεδομένα σε σύγχρονες τεχνικές κανονικοποίησης και ενσωμάτωσης κυκλικών μεταβλητών. Τα αποτελέσματα που προέκυψαν από την αξιολόγηση του μοντέλου στο κεφάλαιο 6, ανέδειξαν τις σημαντικές διαφοροποιήσεις ως προς την προβλεψιμότητα των επιμέρους παραμέτρων.

Το τελικό αποτέλεσμα της πρόγνωσης επηρεάζεται από πολλές και διαφορετικές παραμέτρους σε όλα τα στάδια της διαδικασίας. Τον κυριότερο αντίκτυπο στο μοντέλο και στα αποτελέσματα αυτού, έχουν οι αρχικές χρονοσειρές. Με πιο αναλυτικά πρωτογενή δεδομένα αλλά και με μεγαλύτερο όγκο αυτών, το μοντέλο έχει περισσότερα στοιχεία για να αναλύσει και να μάθει προκειμένου να επιτευχθεί όσο το δυνατόν πιο ακριβής πρόγνωση.

Εν συνεχεία των πρωτογενή δεδομένων, κρίσιμες καθίστανται και η επιλογή και χρήση των κατάλληλων scalars για την κάθε παράμετρο. Καθώς η τροποποίηση της αρχικής χρονοσειράς σε κανονικοποιημένη μορφή για χρήση στο μοντέλο, μπορεί να αλλοιώσει τα δεδομένα με αποτέλεσμα να υπάρχει διαφορετικό αποτέλεσμα στην απόδοση του μοντέλου ακόμα και αν χρησιμοποιούνται τα ίδια πρωτογενή δεδομένα.

Καθώς τα πρωτογενή δεδομένα δεν χαρακτηρίζονται από αναλυτικές καταγραφές και πληθώρα αυτών και η ύπαρξη ιδιαιτερωτήτων στην διαχείριση και την φύση της εκάστοτε παραμέτρου, το μοντέλο ανταπεξήλθε σε ικανοποιητικό επίπεδο στην τελική πρόγνωση αυτών. Οι παράμετροι των θερμοκρασιών και των πιέσεων, διαχειρίστηκαν με μεγαλύτερη ευκολία και ακρίβεια συγκριτικά με τις υπόλοιπες, καθώς αυτές οι παράμετροι χαρακτηρίζονται από εποχικότητα και απώλεια απότομων κορυφών.

Ανακεφαλαιώνοντας, τα αποτελέσματα της εργασίας δείχνουν πως το LSTM αποτελεί μια ιδιαίτερα αποτελεσματική πρακτική για την μοντελοποίηση των μετεωρολογικών χρονοσειρών. Η συνολική εικόνα επιβεβαιώνει ότι η βαθιά μάθηση μπορεί να αποτελέσει ισχυρό εργαλείο για την πρόγνωση και ανάλυση των μετεωρολογικών δεδομένων, λαμβάνοντας υπόψη τις ιδιαιτερότητες και δυσκολίες διαχείρισης κάθε μεταβλητής.

Βιβλιογραφία - Αναφορές

- [1] Ρ. –Ε. Κινγκ, Υπολογιστική Νοημοσύνη, Πανεπιστήμιο Πατρών, 1997
- [2] Ε. Κερανού, Τεχνητή Νοημοσύνη και Έμπειρα Συστήματα, ΕΑΠ, 2000.
- [3] Τσαντεκίδης Α., Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης. Αρχιτεκτονικές και Εκπαίδευση Βαθιών Νευρωνικών Δικτύων, 2016
- [4] Shumway, R. H., & Stoffer, D. S. Time series analysis and its applications : with R examples. Cham: Springer, 2017
- [5] Gurney Gevin. An introduction to neural networks. London: Ucl Press, 1997
- [6] Goodfellow, I., Bengio, Y., & Courville, A. Deep Learning. MIT Press, 2016
- [7] Han, J., Pei, J., & Kamber, M. Data Mining: Concepts and Techniques (3rd ed.). Morgan Kaufmann, 2011
- [8] Vanilla Recurrent Neural Network - Machine Learning Notebook, 2022
- [9] Schmidhuber, J.. Deep learning in neural networks: An overview. Neural Networks, 2015
- [10] Bishop C. M., Pattern Recognition and Machine Learning. Springer, 2006
- [11] Staudemeyer, R. C., & Morris, E. R. Understanding LSTM - a tutorial into Long Short-Term Memory Recurrent Neural Networks, 2019
- [12] C. Olah, Understanding LSTM Networks, Web Page, 2015.
- [13] F. Pedregosa, Scikit-learn: Machine learning in Python, J. Mach. Learn. Res., 2011
- [14] Blazakis, K., Katsigiannis, Stavrakakis, G. One-Day-Ahead Solar Irradiation and Windspeed Forecasting with Advanced Deep Learning Techniques, Energies, 2022
- [15] Chellappa R., Theodoridis S., Van Schaik A., Advances in Machine Learning and Deep Neural Networks, 2021
- [16] Lagouvardos K., Kotroni V., Bezes A., Koletsis I., Kopania T., Lykoudis S., Mazarakis N., Papagiannaki K., Vougioukas S., The automatic weather stations NOANN network of the National Observatory of Athens: operation and database, 2015
- [17] Salcedo Sanz S., Analysis, characterization, prediction, and attribution of extreme atmospheric events with machine learning and deep learning techniques: a review, 2023
Available at:
<https://scispace.com/papers/analysis-characterization-prediction-and-attribution-of-6g7jhnonye>

- [18] Justin Georgevici A., Terblanche M., Neural networks and deep learning: a brief introduction, 2019 Available at:
<https://drive.google.com/drive/folders/1j3kqzCTOImguL7R4mjHXtO8h4AzQMDn5>
- [19] MacGovern A., Developing trustworthy AI for weather and climate, 2024 Available at:
<https://scispace.com/papers/developing-trustworthy-ai-for-weather-and-climate-2ruwq5ytt0>
- [20] Nastos P.T., Paliatsos A.G., Koukouletsos K.V., Larissi I.K., Moustiris K.P., Artificial neural networks modeling for forecasting the maximum daily total precipitation at Athens, Greece, 2014 Available at:
<https://www.sciencedirect.com/science/article/abs/pii/S0169809513003360>
- [21] Zemnazi O., El Filali S., Ouahabi S., Weather Forecasting Using Artificial Neural Network (ANN): A Review Available at:
<https://www.sciencedirect.com/science/article/pii/S1877050924018015#bbib0009>
- [22] A study on Deep Neural Networks framework, IEEE Explore, Available at:
<https://ieeexplore.ieee.org/abstract/document/7867471>
- [23] Probabilistic weather forecasting with machine learning, 2024 Available at:
<https://www.nature.com/articles/s41586-024-08252-9>
- [24] A brief comparison of different learning methods for wind speed Forecasting, IEEE Explore, Available at:
<https://ieeexplore.ieee.org/document/9258733>.
- [25] Brownlee J., A Gentle Introduction to LSTM Autoencoders. Machine Learning Mastery. Available at:
<https://machinelearningmastery.com/lstm-autoencoders/>
- [26] Deep Learning based Wind Speed Forecasting-A Review IEEE Explore, Available at:
<https://ieeexplore.ieee.org/document/8776923>
- [27] Robust deep neural network for wind speed prediction IEEE Explore, Available at:
<https://ieeexplore.ieee.org/document/7391664>
- [28] Time Series Prediction Using LSTM Deep Neural Networks Available at:
<https://www.altumintelligence.com/articles/a/Time-Series-Prediction-Using-LSTM-Deep-Neural-Networks>.

Υπεύθυνη Δήλωση Συγγραφέα:

Δηλώνω ρητά ότι, σύμφωνα με το άρθρο 8 του Ν.1599/1986, η παρούσα εργασία αποτελεί αποκλειστικά προϊόν προσωπικής μου εργασίας, δεν προσβάλλει κάθε μορφής δικαιώματα διανοητικής ιδιοκτησίας, προσωπικότητας και προσωπικών δεδομένων τρίτων, δεν περιέχει έργα/εισφορές τρίτων για τα οποία απαιτείται άδεια των δημιουργών/δικαιούχων και δεν είναι προϊόν μερικής ή ολικής αντιγραφής, οι πηγές δε που χρησιμοποιήθηκαν περιορίζονται στις βιβλιογραφικές αναφορές και μόνον και πληρούν τους κανόνες της επιστημονικής παράθεσης.