ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

" Ανάπτυξη πλατφόρμας απεικόνισης και πρόβλεψης τιμής ηλεκτρικής ενέργειας με χρήση υβριδικού αλγόριθμου μηχανικής μάθησης "

ΠΟΥΡΙΚΗΣ ΑΘΑΝΑΣΙΟΣ

ΕΠΙΒΛΕΠΩΝ: Παναγιώτης Αλεφραγκής

ΠΑΤΡΑ 2021

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή

Πάτρα, Ημερομηνία

ΕΠΙΤΡΟΠΗ ΑΞΙΟΛΟΓΗΣΗΣ

1. Ονοματεπώνυμο, Υπογραφή

2. Ονοματεπώνυμο, Υπογραφή

3. Ονοματεπώνυμο, Υπογραφή

Υπεύθυνη Δήλωση Φοιτητή

Βεβαιώνω ότι είμαι συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τη συγκεκριμένη εργασία. Η έγκριση της διπλωματικής εργασίας από το Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών του Πανεπιστημίου Πελοποννήσου δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος. Η παρούσα εργασία αποτελεί πνευματική ιδιοκτησία του φοιτητή Πουρίκη Αθανάσιου που την εκπόνησε. Στο πλαίσιο της πολιτικής ανοικτής πρόσβασης ο συγγραφέας/δημιουργός εκχωρεί στο Πανεπιστήμιο Πελοποννήσου, 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' οποιονδήποτε 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.

Περιεχόμενα

ΠΙΝΑΚΑΣ ΣΥΝΤΟΜΟΓΡΑΦΙΟΝ

ΔΕΠ	Διαδικασία Ενοποιημένου Προγραμματισμού
SMP	System Marginal Price
KNN	*k*-Nearest Neighbors
LSTM	Long short-term memory
RNN	Recurrent Neural Network
RMSE	Root mean square errors
MAE	Mean Absolute Error
MAPE	Mean Percentage Absolute Error
ΑΔΜΗΕ	Ανεξάρτητος Διαχειριστής Μεταφοράς Ηλεκτρικής Ενέργειας
OLS	Ordinary Least Squares
ΜΑΠΕ	Μη Ανανεώσιμες Πηγές Ενέργειας
ΑΠΕ	Ανανεώσιμες Πηγές Ενέργειας
MWh	Megawatt Hour
SCADA	Supervisory Control and Data Acquisition
XGBoost	Extreme Gradient Boosting
HEnEx	Hellenic Energy Exchange
ML	Machine Learning

ΠΕΡΙΛΗΨΗ

Με τις νέες επένδυσης που έχει φέρει η απορρύθμιση του τομέα της ηλεκτρικής ενέργειας στην Ελλάδα τα τελευταία χρονιά, επιδιώκεται κάθε ανταγωνιστικό πλεονέκτημα με την πρόβλεψη τιμής είναι ένα από τα μεγαλύτερα. Στα πλαίσια της πτυχιακής ερευνήθηκαν διάφορα μη ιδιόκτητα δεδομένα και εκπαιδεύτηκαν ένας αλγόριθμος πολλαπλής γραμμικής παλινδρόμησης, ένας αλγόριθμος k-πλησιέστερων γειτόνων, ένα μοντέλο XGBoost, ένα LSTM και τέλος βρέθηκε ο μέσος ορός τον αποτελεσμάτων τους ως την τελική πρόβλεψη. Επίσης αναπτύχθηκε πλατφόρμα απεικόνισης δεδομένων και API.

With the new investments that the deregulation of the energy sector has yielded in recent year in Greece, every competitive advantage is pursued and price prediction is the most sought after. For this dissertation nonproprietary data were researched and trained a multiple linear regression algorithm, an algorithm K-Nearest neighbors, a XGBoost model, an LSTM and their mean value from their results was found as the final prediction. A data visualization platform and API were also developed.

Κεφάλαιο 1: Εισαγωγή

Τα τελευταία χρόνια η Ελληνική αγορά ενέργειας υπέστη εκτεταμένες μεταρρυθμίσεις εισάγοντας ευκαιρίες για επενδύσεις σε μια ανταγωνιστική αγορά με βάση τις ανανεώσιμες πηγές ενέργειας. Η πρόβλεψη της ωριαίας τιμής για την αγορά της επομένης ημέρας (DAM) είναι ένα σημαντικό ανταγωνιστικό πλεονέκτημα για την πλήρη εκμετάλλευση της παραγωγής αλλά και της αποφυγής περιττών εξόδων.

1.1. Τι είναι η μηχανική μάθηση

Ο όρος μηχανική μάθηση πρωτοεμφανίστηκε το 1959 από τον Arthur L. Samuel για το παιχνίδι της ντάμας [1]. Η μηχανική μάθηση είναι μια υποκατηγορία του τομέα της τεχνητής νοημοσύνης και χρησιμοποιεί δεδομένα για να εκπαίδευση ένα μοντέλο για πρόβλεψη, ταξινόμηση, ή ομαδοποίηση χωρίς να έχει προγραμματιστεί ρητά για κάθε συγκεκριμένο πρόβλημα.

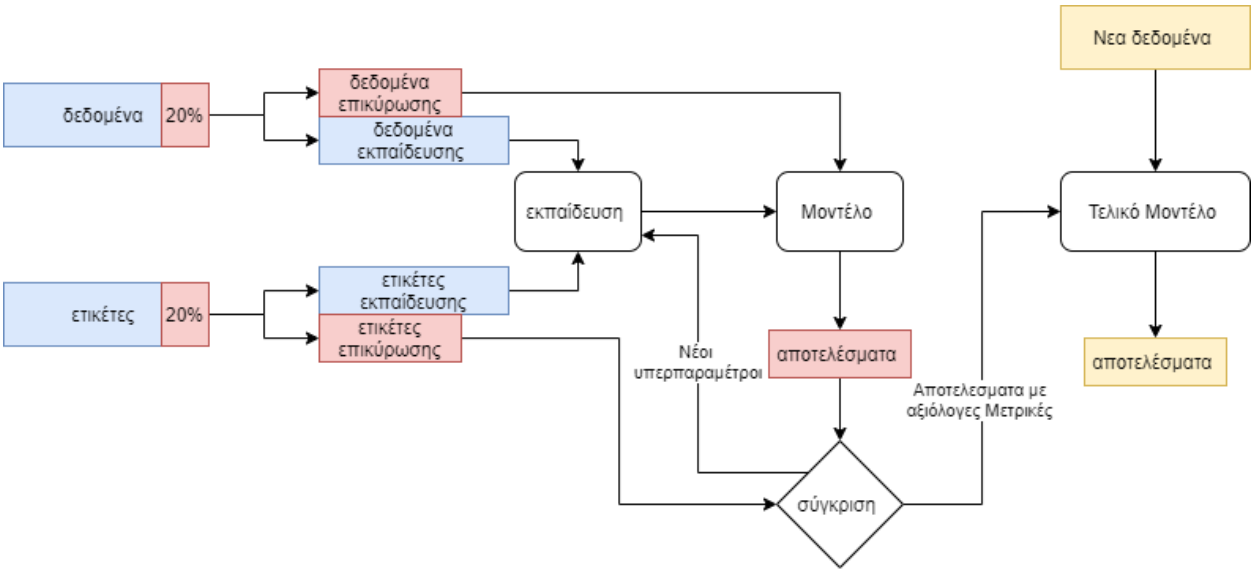
1.2. Βασικοί τομείς μηχανικής μάθησης

Η μηχανική μάθηση χωρίζετε σε τομείς και οι τέσσερις κυριότεροι είναι:

1.2.1. Επιβλεπόμενη Μάθηση

Η επιβλεπόμενη μάθηση πρόκειται για προσέγγιση μιας συνάρτησης με μεταβλητές εισόδου x_n μεταβλητή εξόδου y. Ο όρος επιβλεπόμενη προκύπτει από το γεγονός ότι γνωρίζουμε τη μεταβλητή εξόδου. Χρησιμοποιεί ιστορικά δεδομένα για να εκπαιδεύσει το μοντέλο και

προβλέπει τις μελλοντικές τιμές βάση αυτών.



1.1 Περιγραφή εκπαίδευσης επιβλεπομένου μοντέλου

1.2.2. Μη-Επιβλεπόμενη Μάθηση

Αντιθέτως στη Μη- Επιβλεπόμενη Μάθηση δεν γνωρίζεται η μεταβλητή εξόδου αλλά οι αλγόριθμοι δημιουργούν μια συνάρτηση που βρίσκει συσχετίσεις ανάμεσα στα δεδομένα και περιλαμβάνει ομαδοποίηση, ανίχνευση ανωμαλιών κ.α.. Ένα βασικό μειονέκτημα είναι ο μεγάλος όγκος δεδομένων που χρειάζεται για την εκπαίδευση ώστε να βρεθούν αξιοπρεπή αποτελέσματα.

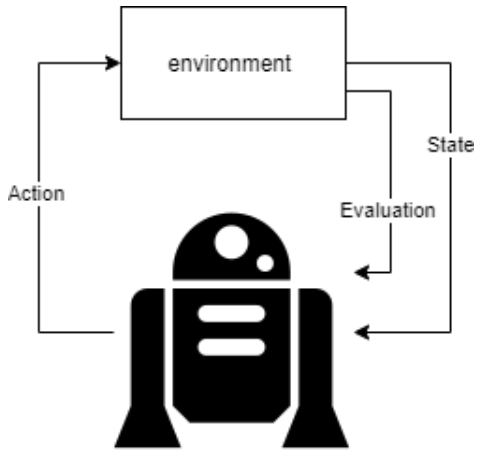
1.2.3. Ημι-Επιβλεπόμενη Μάθηση

Η ημι-επιβλεπόμενη μάθηση είναι συνδυασμός της επιβλεπόμενης και μη-επιβλεπόμενης μάθησης καθώς χρησιμοποιεί δεδομένα που γνωρίζουμε την μεταβλητή εξόδου και μη, με την πλειοψηφία των δεδομένων εκπαίδευσης να έχουν άγνωστη μεταβλητή εξόδου. Χρησιμοποιείτε

όταν εργαζόμαστε με δεδομένα όπου τα παραδείγματα είναι δύσκολο ή δαπανηρό να επισημανθούν. Έχει χρησιμοποιηθεί για τον εντοπισμό ανεπιθύμητων μηνυμάτων και απάτης.

1.2.4. Ενισχυτική Μάθηση

Είναι μια μέθοδος που αναλόγως του αποτελέσματος ανταμείβεται ή τιμωρείτε ο πράκτορας. Αυτό επιτρέπει στον πράκτορα να αναζητά καταστάσεις που θα είναι επικερδείς και να αποφύγει της μη επιθυμητές. Η πρακτική εφαρμογή των αλγορίθμων μηχανικής μάθησης είναι περιορισμένη λόγω απαγορευτικής ανάγκης όγκου δεδομένων και περιορισμένων περιπτώσεων χρήσης. Χρησιμοποιείται κυρίως στην ρομποτική.

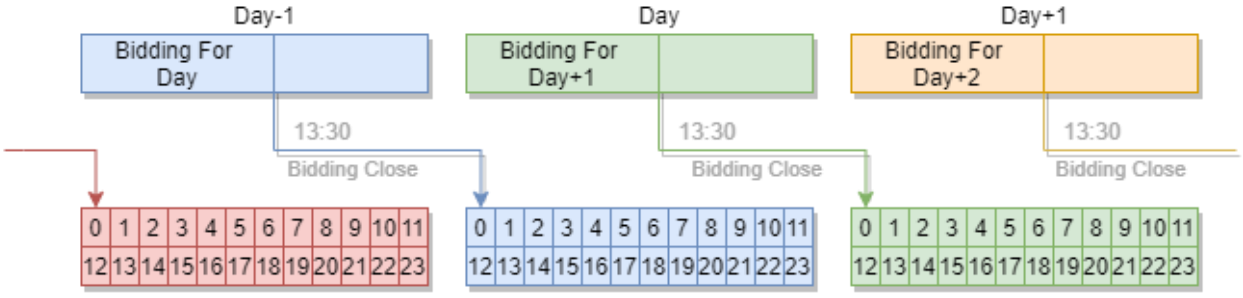


1.2 Περιγραφή μοντέλου ενισχυτικής μάθησης

Κεφάλαιο 2 : Σκοπός

2.1. Περιγραφή του προβλήματος

Σε αντίθεση με άλλες χρηματοπιστωτικές αγορές η αγορά άμεσης παράδοσης ηλεκτρικής ενέργειας λειτουργεί σε χρονοδιάγραμμα επόμενης ημέρας(day-ahead) όπου οι τιμές για τις επόμενες 24 ώρες υποβάλλονται την προηγούμενη μέρα.

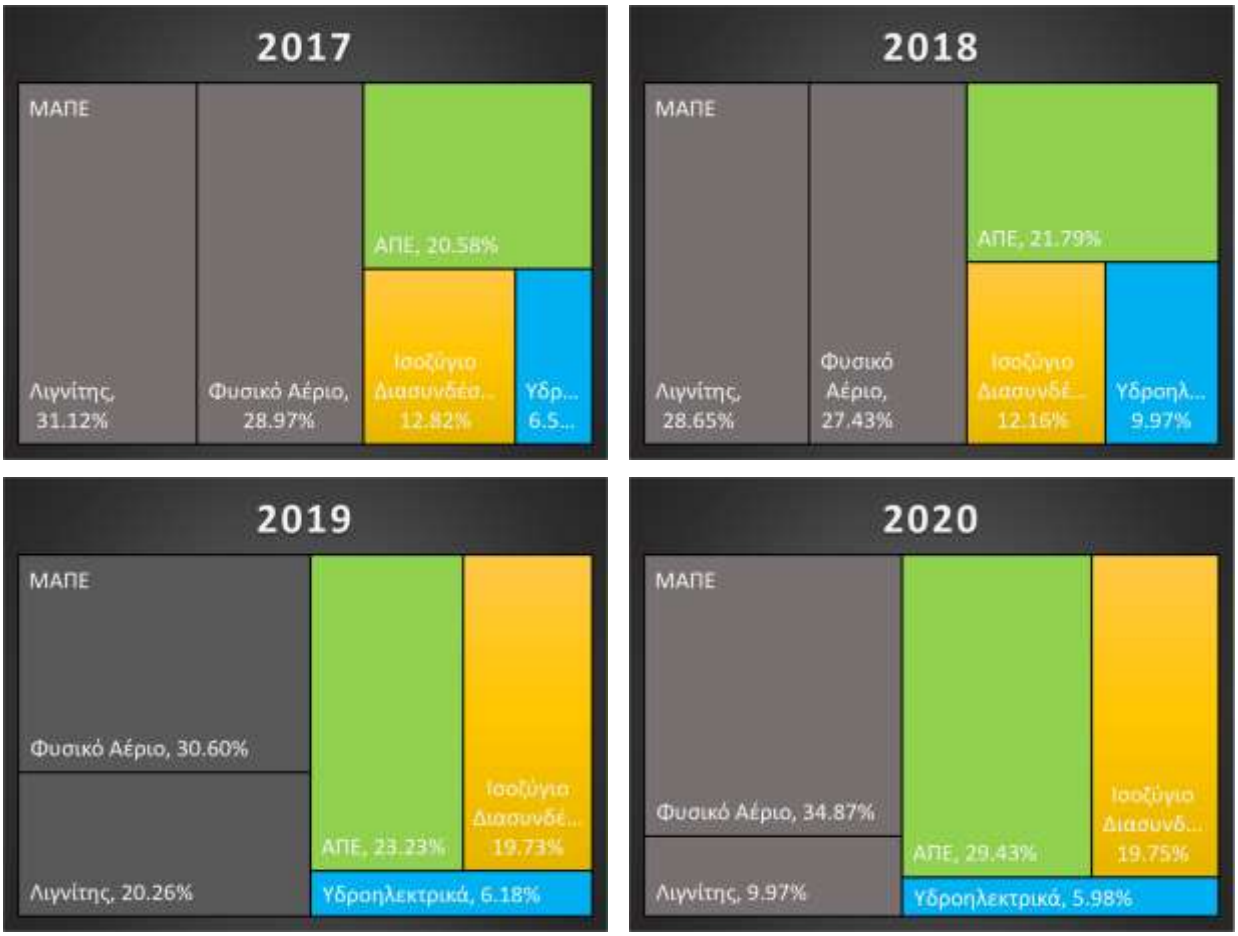


2.1 Απεικόνιση χρονοδιαγράμματος HEnEx

Ο σκοπός αυτής της πτυχιακής είναι η έρευνα κατάλληλων δεδομένων και αλγορίθμων για την πρόβλεψη της μελλοντικής τιμής ενέργειας, η απεικόνιση και η δημιουργία μιας διεπαφής προγραμματισμού εφαρμογών (API) για τη απόκτηση αυτών των δεδομένων με έναν εύκολο τρόπο.

Καθώς η ελληνική αγορά ενέργειας κινείτε προς τις ανανεώσιμες πήγες με καινούργιους επενδυτές, καθιστά αναγκαία την γνώση της τιμής για την δημιουργία στρατηγικών, ειδικότερα σε μικρότερες μονάδες παραγωγής που δεν έχουν κάποιο μεγάλο ποσοστό της ολικής παραγωγής για τον καθορισμό της τιμής.

Η Ελλάδα κυριαρχείτε ακόμα από μη ανανεώσιμες πηγές ενέργειας (ΜΑΠΕ) . Όμως τα τελευταία χρόνια βλέπουμε την απεξάρτηση από ρυπογόνες παραγωγικές τεχνικές όπως ο λιγνίτης και αύξηση των ΑΠΕ και φυσικού αερίου [3].



2.2 Treemap παραγωγής ηλεκτρικής ενέργειας ανά χρόνο

Καθώς η πρόβλεψη είναι συνεχής μεταβλητή, το πρόβλημα παίρνει την μορφή παλινδρόμησης [2]. Η παλινδρόμηση είναι μια στατιστική μέθοδος για την εκτίμηση μιας εξαρτώμενης μεταβλητής (label) με μια ή περισσότερες ανεξάρτητες (features).

2.2. Εξέλιξη τεχνολογίας

Οι προβλέψεις με σχέση την ηλεκτρική ενέργεια απασχολούν τον τομέα της μηχανικής μάθησης εδώ και χρόνια. Το 2021 ο Lago, ο Rafał Weron [3] κ.ά. πραγματοποίησαν μια έρευνα στην οποία παρουσίασαν την τελευταία τεχνολογία, τις εξελίξεις για πρόβλεψη της τιμής ηλεκτρικής ενέργειας και αναγνώρισαν μερικά προβλήματα που μαστίζουν τον τομέα όπως

ερευνά που αναπτύσσουν προηγμένες στατιστικές τεχνικές και τις συγκρίνουν μόνο με απλές μεθόδους και δείχνουν ότι οι στατιστικές μέθοδοι είναι σαφώς καλύτερες, σε άλλες υπάρχουσες μελέτες η περίοδος δοκιμής είναι πολύ σύντομη για να δώσει σαφή αποτελέσματα π.χ. μια εβδομάδα και κυριότερα ορισμένες δεν παρέχουν αρκετές πληροφορίες για την αναπαραγωγή της έρευνας.

Το 2019 έρευνα από Díaz κ.α. έδειξε ότι χρησιμοποιώντας Gradient boosting regression trees και μη ιδιόκτητα δεδομένα από τον Ισπανικό Διαχειριστή Συστήματος Μεταφοράς έχει βελτιωμένα αποτελέσματα από μοντέλα πολλαπλής γραμμικής παλινδρόμησης [4]. Αυτό δίνει την δυνατότητα σε μικρές μονάδες με ανεπαρκή δεδομένα να γίνουν πιο ανταγωνιστικές.

Το 2018 ερευνητές από τη Σχολή Επιστήμης και Μηχανικής Πληροφορικής Πανεπιστήμιου Lanzhou στην Κίνα, έδειξαν ότι ένα LSTM βελτιστοποιημένο με Adam και δεδομένα από την Νέα Νότια Ουαλία της Αυστραλίας έδειξε καλύτερα αποτελέσματα από κλασικά μοντέλα όπως τα ARIMA, ANN, hybrid ARIMAANN [5].

2.3. Δεδομένα

Η χρονική περίοδος των δεδομένων περιορίζεται από τις 01/11/2020 λόγω της αλλαγής υπολογισμού της τιμής εφαρμόζοντας το μοντέλο στόχος (target model) που αλλάζει το ελληνικό μοντέλο στα πρότυπα του Ευρωπαϊκού και καθιστά προηγούμενα της ημερομηνίας ιστορικά δεδομένα άκυρα.

2.4. Ανάλυση δεδομένων

Ένα από τα βασικότερα κομμάτια της επιστήμης δεδομένων είναι η ανάλυση αυτών. Τα δεδομένα πριν χρησιμοποιηθούν για την εκπαίδευση πρέπει να υποβληθούν σε καθαρισμό και μετατροπές.

2.4.1.1. Συσχέτιση Pearson

Ο Συντελεστής συσχέτισης Pearson δείχνει την γραμμική συσχέτιση ανάμεσα σε 2 μεταβλητές. Αν έχουμε y η τιμή ηλεκτρικής ενέργειας ανά MWh και x μια ανεξάρτητη μεταβλητή έχουμε :

$$r_{xy} = \frac{n\sum x_t y_t - \sum x_t \sum y_t}{\sqrt{n\sum x_t^2 - (\sum x_t)^2}\sqrt{n\sum y_t^2 - (\sum y_t)^2}}$$

Όπου n αριθμός τον στοιχείων και $x_t y_t$ στοιχεία την στιγμή t

2.4.2. Οριακή τιμή συστήματος

Οι προηγούμενες τιμές ενέργειας εκδίδονται από το Ελληνικό Χρηματιστήριο Ενέργειας και θα αντιστοιχηθούν με τα παρακάτω δεδομένα. Η συλλογή των δεδομένων θα γίνει με την χρήση ανιχνευτή ιστού (Web Crawler) καθώς δεν προσφέρεται API για λήψη δεδομένων. Το σύνολο των δεδομένων αποτελείτε από χρόνο σειρές της μιας ώρας και θα πρέπει να γίνουν αλλαγές στα υπόλοιπα υποσύνολα για να συμβαδίζουν με αυτό.

2.4.3. Απαιτήσεις ΑΔΜΗΕ

Καθημερινά ο Ανεξάρτητος Διαχειριστής Μεταφοράς Ηλεκτρικής Ενέργειας (ΑΔΜΗΕ) εκδίδει απαιτήσεις που χρησιμοποιούνται στην Διαδικασία Ενοποιημένου Προγραμματισμού (ΔΕΠ) στις 09:35 ώρα Ελλάδος [6]. Η ΔΕΠ είναι μια διαδικασία η οποία εκτελείται από τους Διαχειριστές Συστημάτων Μεταφοράς που χρησιμοποιούν σύστημα κεντρικού προγραμματισμού και κατανομής (Central Dispatch Systems). Η διαδικασία αποσκοπεί στην κάλυψη των προβλεπόμενων αποκλίσεων παραγωγής/ζήτησης και τη διασφάλιση των απαιτούμενων εφεδρειών. Οι μεταβλητές αυτού του υποσυνόλου είναι :

$Renewables_t$: Η πρόβλεψη παραγωγής των ΑΠΕ (συμπεριλαμβάνει φωτοβολταϊκά, ανεμογεννήτριες, Βιομάζα κ.λπ.)

$Non - Dispatcheble_t$: Η ΑΠΕ που δεν μπορούν να κατανεμηθούν, δηλαδή μονάδες ΑΠΕ οι οποίες παράγουν στοχαστικά, ανάλογα με τις καιρικές συνθήκες (φωτοβολταϊκά, ανεμογεννήτριες) και των οποίων η παραγωγή δεν μπορεί να προγραμματιστεί διότι δεν έχουν την δυνατότητα αποθήκευσης ώστε να μπορούν αυξομειώσουν την παραγωγή τους λαμβάνοντας συγκεκριμένες εντολές.

Man_Hydro_t: Τα υδροηλεκτρικά μπορούν να προγραμματίσουν την παραγωγή τους, δηλαδή να συλλέγουν νερό στον ταμιευτήρα τους και να το απελευθερώσουν όποτε κρίνεται απαραίτητο (πχ υψηλή τιμή αγοράς) ώστε να παραχθεί ενέργεια. Επειδή τα ποτάμια πρέπει να έχουν συνεχόμενη ροή νερού, σε πολλές περιπτώσεις πρέπει ένα ποσό νερού να φεύγει από τον ταμιευτήρα ώστε να μην υπάρχει κίνδυνος υπερχείλισης αλλά και η συνέχιση της ύδρευσης. Η υποχρεωτική αυτή ποσότητα νερού εκφράζεται ως υποχρεωτική παραγωγή ενέργειας του κάθε Υδροηλεκτρικού.

$Commissioning_t$, Αφορά νέες μονάδες, οι οποίες βρίσκονται σε δοκιμαστική και όχι κανονική λειτουργία. Κατά το διάστημα αυτό είναι πιο ελαστικά τα όρια λειτουργίας τους και δεν λαμβάνουν κυρώσεις, λόγω μη προβλέψιμης συμπεριφοράς λειτουργίας.

$Reserve\_Up_t$, $Reserve\_Down_t$ Για να διασφαλιστεί η ευστάθεια του δικτύου, πρέπει πάντα να υπάρχει τόση διαθέσιμη παραγωγή, όση και η ζήτηση (σε επίπεδα δευτερολέπτου). Αυτό είναι δύσκολο να διασφαλιστεί, γιατί η λειτουργία και ο προγραμματισμός της αγοράς στηρίζεται σε προβλέψεις (παραγωγής ΑΠΕ, φορτίου). Επίσης πρέπει να ληφθούν υπόψη και τυχόν απρόβλεπτες καταστάσεις, όπως πχ το να βγει εκτός μια μεγάλη μονάδα παραγωγής λόγω βλάβης κτλ. Έτσι λοιπόν, ο ΑΔΜΗΕ έχει φροντίσει να υπάρχουν κάθε στιγμή οι απαραίτητες εφεδρείες (up/down) δηλαδή μονάδες σε ετοιμότητα, ικανές να ανεβάσουν ή να κατεβάσουν την ισχύ παραγωγής τους ακριβώς όταν ζητηθεί, ώστε να φροντίσουν την τήρηση του ισοζυγίου.

Ο ΑΔΜΗΕ προσφέρει δεδομένα σε χρόνο σειρά ανά μισή ώρα και πρέπει να γίνει η μετατροπή τους σε ώρες για να συμβαδίζουν με τις τιμές που εκδίδονται από το Ελληνικό Χρηματιστήριο Ενέργειας.

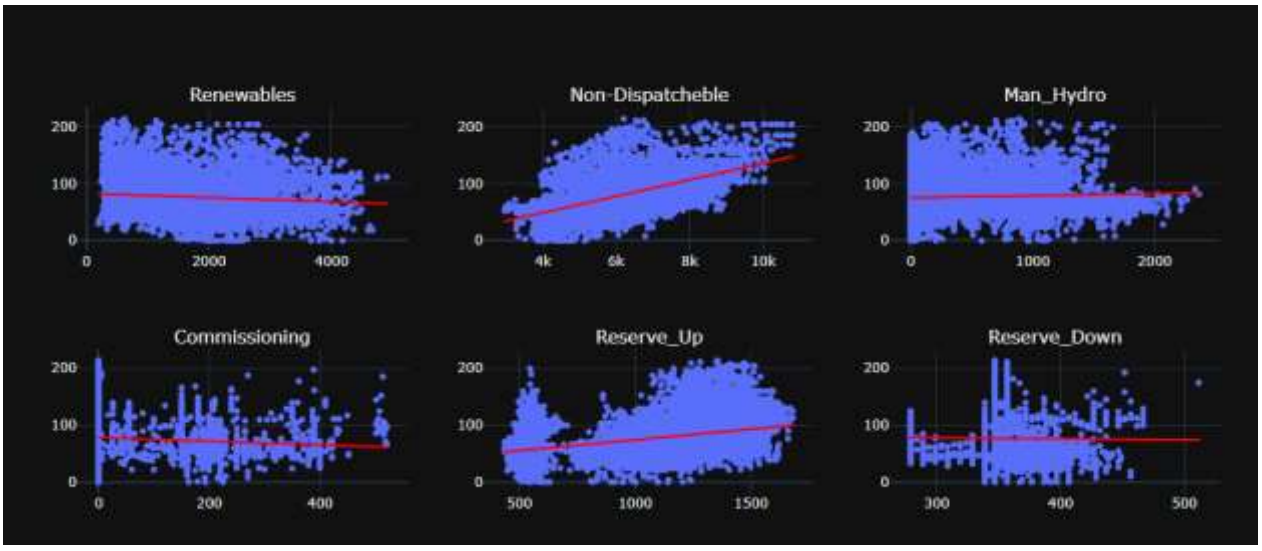
2.4.3.1. Ανάλυση Pearson

Η ανάλυσή Pearson ανάμεσα στις μεταβλητές από τις απαιτήσεις ΑΔΜΗΕ και της πραγματικής τιμής (SMP) έχει τα εξής αποτελέσματα.

Renewables	*Non-Dispatcheble*
-0.11	*0.67*
Man_Hydro	*Commissioning*
0.11	*-0.07*
Reserve_Up	*Reserve_Down*
0.37	*-0.08*

2.3 Πίνακας Ανάλυσης Pearson ανάμεσα στις μεταβλητές από την ΔΕΠ και της και της (SMP)

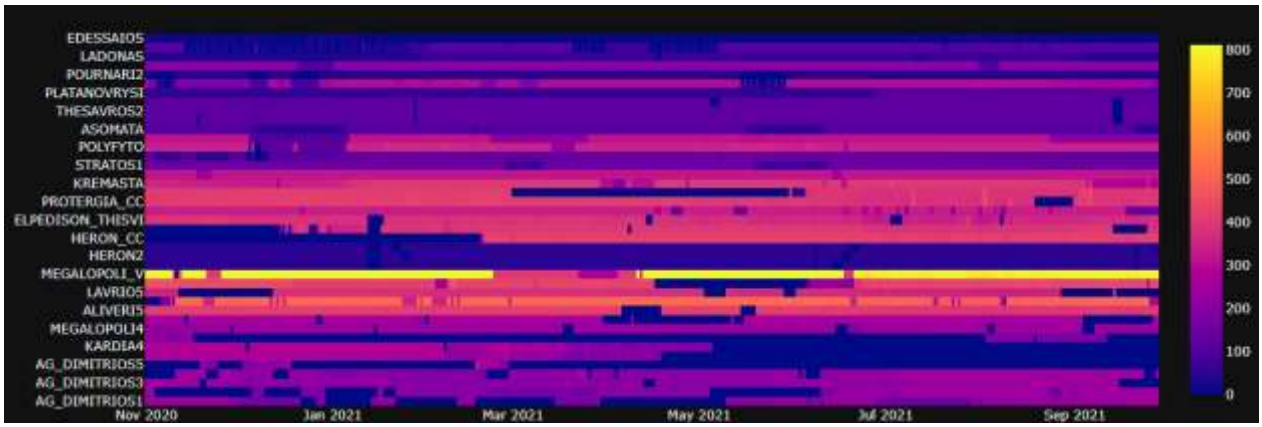
Όπως βλέπουμε υπάρχει φανερή γραμμική συσχέτιση ανάμεσα σε κάποιες μεταβλητές, ειδικότερα στην μεταβλητή *Non-Dispatcheble*, δυστυχώς η προμήθεια δεδομένων τιμών πρώτων υλών φυσικού αερίου και λιγνίτη αποδείχθηκε αδύνατη. Επίσης βλέπουμε μια μικρή αρνητική συσχέτιση ανάμεσα στις ανανεώσιμες πήγες ενέργειας.



2.4 Γραμμική συσχέτιση (OLS) ανάμεσα στις απαιτήσεις από ΑΔΜΗΕ με την τιμή ηλεκτρικής ενέργειας

2.4.4. Διαθεσιμότητα Μονάδων

Σε αυτό το υποσύνολο δεδομένων παρατηρούμε τις μονάδες παραγωγής ηλεκτρικής ενέργειας που θα λειτουργήσουν την ημέρα του δείγματος καθώς και την προβλεπόμενη διαθεσιμότητα τους σε MW. Το υποσύνολο αυτό αποτελείτε από 42 μονάδες παραγωγής ηλεκτρικής ενέργειας από μη ανανεώσιμες πηγές. Ορισμένες έχουν παύσει την λειτουργία τους και άλλες είναι προγραμματισμένες να σταματήσουν να παράγουν ηλεκτρική ενέργεια καθώς η Ελλάδα απομακρύνεται από αυτές σε ανανεώσιμες πηγές.



2.5 Χάρτης θερμότητας παραγωγής ηλεκτρικής ενέργειας

2.4.4.1. Ανάλυση Pearson

Με Pearson βλέπουμε αρκετά μεγάλες συσχετίσεις ανάμεσα στην οριακή τιμή συστήματος ανά μονάδα παραγωγής.

AG_DIMITRIOS1	AG_DIMITRIOS2	AG_DIMITRIOS3	AG_DIMITRIOS4	AG_DIMITRIOS5	KARDIA3	KARDIA4
0.33	0.49	0.29	0.36	-0.25	-0.75	-0.77
MEGALOPOLI3	MEGALOPOLI4	MELITI	ALIVERI5	LAVRIO4	LAVRIO5	KOMOTINI
0.36	-0.18	0.02	-0.07	0.08	0.01	-0.01
MEGALOPOLI_V	HERON1	HERON2	HERON3	HERON_CC	ELPEDISON_THESS	ELPEDISON_THISVI
0.16	-0.02	0.03	-0.01	0.63	0.39	-0.21
ALOUMINIO	PROTERGIA_CC	KORINTHOS_POWER	KREMASTA	KASTRAKI	STRATOS1	ILARIONAS
-0.17	-0.37	0.06	0.31	0.09	0.03	0.29
POLYFYTO	SFIKIA	ASOMATA	THESAVROS1	THESAVROS2	THESAVROS3	PLATANOVRYSI
0.26	0.24	0.22	0.22	0.27	0.13	0.77
POURNARI1	POURNARI2	P_AOOU	LADONAS	PLASTIRAS	EDESSAIOS	AGRAS
0.28	0.23	-0.13	0.45	0.44	0.02	-0.34

2.6 Ανάλυση Pearson ανάμεσα στης μονάδες και την μέση τιμή μέρας

2.4.5. Δεδομένα καιρού

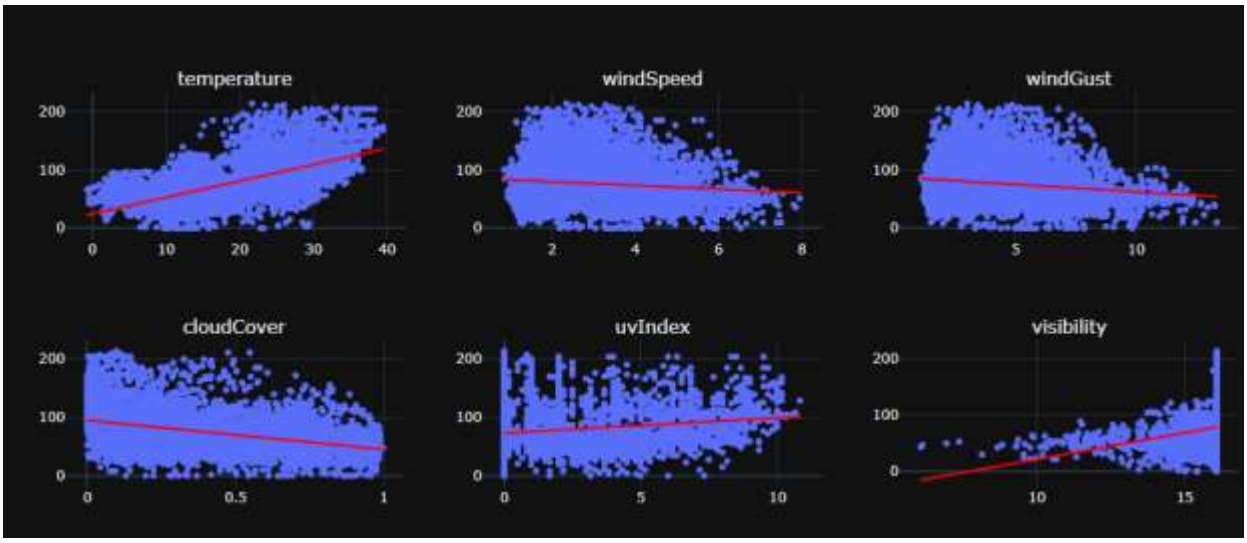
Η συσχέτιση ανάμεσα στον καιρό και στην ηλεκτρική ενέργεια είναι εμφανείς καθώς καθορίζει την παραγωγή αλλά και την κατανάλωση αρά και την τιμή. Για την συλλογή θα χρησιμοποιηθεί υπηρεσία DarkSkyApi API της εταιρείας Apple. Ο άξονας των καιρικών δεδομένων έχει ερευνηθεί αρκετά με ικανοποιητικά αποτελέσματα [7] [8]. Θα βρεθεί η μέση από μεγαλύτερα αστικά κέντρα τις Ελλάδας των μεταβλητών θερμοκρασίας, ταχύτητας ανέμου, ρίπων ανέμου, νεφοκάλυψη, Δείκτης UV, ορατότητα (*temperature*, *windSpeed*, *windGust*, *cloudCover*, *uvIndex*, *visibility*).

2.4.5.1. Ανάλυση Pearson

Η ανάλυση κατά Pearson επιβεβαιώνει την υπόθεση συσχέτισης καιρού και τιμή ηλεκτρικής ενέργειας.

temperature	windSpeed
0.68	-0.10
windGust	cloudCover
-0.14	-0.44
uvIndex	visibility
0.22	0.18

2.7 Πίνακας Ανάλυσης Pearson ανάμεσα στα Δεδομένα καιρού και SMP)



2.8 Γραμμική συσχέτιση (OLS) ανάμεσα με δεδομένα καιρού με την τιμή ηλεκτρικής ενέργειας

2.4.6. Σύνολα Δεδομένων

Με τα προηγούμενα δεδομένα θα δημιουργήσουμε τα εξής Σύνολα δεδομένων(datasets)

- requirements: μόνο οι απαιτήσεις από ΑΔΜΗΕ
- requirements_units: απαιτήσεις από ΑΔΜΗΕ με διαθεσιμότητα μονάδων
- requirements_weather: απαιτήσεις από ΑΔΜΗΕ με πρόβλεψη καιρού
- requirements_units_ weather: Όλα τα υποσύνολα

Σε κάθε σύνολο προστίθεται και η τιμή ηλεκτρικής ενέργειας με τιμή μετατόπισης 24 (24 ώρες) δηλαδή 1 ημέρας.

Κεφάλαιο 3 : Αλγόριθμοι

Για την πρόβλεψη θα εξετάσουμε διάφορους αλγόριθμους με διαφορετικές αρχιτεκτονικές.

3.1. Πολλαπλή γραμμική παλινδρόμηση

Ο αλγόριθμος πολλαπλής γραμμικής παλινδρόμησης μοντελοποιεί την σχέση ανάμεσα σε ανεξάρτητες μεταβλητές με μια εξαρτώμενη [9]. Στα πλαίσια της πτυχιακής εργασίας θα χρησιμοποιηθεί, η εκδοχή των ελάχιστων τετραγώνων, ως ένα βασικό μοντέλο (baseline model) αφού πρόκειται για ένα από τα πιο απλά και αξιόπιστα μοντέλα. Η τεχνική ελάχιστων τετραγώνων ελαχιστοποιεί το άθροισμα των τετραγωνικών διαφορών μεταξύ των ανεξάρτητων δεδομένων και της γραμμής παλινδρόμησης. Ο αλγόριθμος βρίσκει συντελεστές W_i όπου αντικαθιστούμε την έκφραση $mx + b$ με

$$\hat{y} = \left(\sum_{i=1}^{totalFeatures} W_i feature_i \right) + W_0$$

οπού W_i τα βάρη και $feature_i$ τα δεδομένα. Τα βάρη υπολογίζονται κατά

$$W_i = \frac{\sum_{t=0}^{timestamps} \left(x_{t,i} - \right)\left(y_{t,i} - \bar{y}_i\right)}{\sum_{t=0}^{timestamps} \left(x_{t,i} - \bar{x}_i\right)^2}$$

Οπού $\bar{x}_i$ η μέση τιμή του χαρακτηριστικού i και

$$W_0 = \bar{y} \sum - W_i \bar{x}_i$$

3.2. Παλινδρόμηση πλησιέστερων γειτόνων

Πρόκειται για ένα ακόμα απλό αλλά αξιόπιστο μοντέλο που έχει χρησιμοποιηθεί στον τομέα των οικονομικών αλλά και στην πρόβλεψη παραγωγής ηλεκτρικής ενέργειας [10] [11]. Δεν δημιουργεί κάποιο μοντέλο αλλά αποθηκεύει τα δεδομένα, έπειτα υπολογίζει την απόσταση Minkowski

$$Dis(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=0}^{totalFeatures} |x_i \ - \ y_i|^p\right)^{\frac{1}{p}}$$

οπού αν $p = 1$ πρόκειται για την απόσταση Manhattan και αν $p = 2$ πρόκειται για την Ευκλείδεια απόσταση, οπού $\boldsymbol{x}$ πίνακας με χαρακτηριστικά και $\boldsymbol{y}$ πίνακας με τα καινούργια δεδομένα. Δημιουργούμε πίνακα δεικτών

$$\bigcup_{k=0}^{K} arg\ min_k(\{Dis(x_m, y)\}_{m=0}^{M})$$

όπου $arg\ min_k$ επιστρέφει τον κ μικρότερο δείκτη π.χ. $arg\ min_2$ ο δείκτης με την δεύτερη μικρότερη απόσταση του συνόλου δεδομένων από τις μικρότερες αποστάσεις μεγέθους K . Η πρόβλεψη υπολογίζεται με $\frac{\sum_{k=0}^{K} label_k}{K}$ οπού $label_k$ τιμή ηλεκτρικής ενέργειας με δείκτη k από το σύνολο δεικτών K .

3.3. Παλινδρόμηση XGBoost

Ο αλγόριθμος Extreme Gradient Boosting είναι ένα σύνολο(ensemble) αλγόριθμων μηχανικής μάθησης και χρησιμοποιείτε είτε για κατηγοριοποίηση είτε για παλινδρόμηση. Το σύνολο αποτελείτε από δέντρα αποφάσεων, είναι βασισμένο στην τεχνική gradient boosting ειδικότερα από τον Friedman κ.ά. [12], και βελτιώνονται από μια συνάρτηση διαφοροποιημένου λάθους καθόδου κλίσης. Όταν δεν μπορεί να βελτιωθεί κάποιο από τα προηγούμενα δέντρα η γενικότερη βελτίωση του αλγόριθμου επιτυγχάνετε με την εισαγωγή ενός καινούργιου δέντρου. Η πρόβλεψη $\widehat{y_t}$ επιτυγχάνετε από τον υπολογισμό $\widehat{y}_t = \sum_{k=1}^{K} f_k(x_t)$, $f_k \in F$ οπού F το σύνολο τον δέντρων και x_t δείγμα της χρονικής στιγμής t με $x_t = \{x_0, x_1, \dots, x_{totalFeatures}\}$ [13]. Το σύνολο $F = \{f(x) = w_{q(x)}\}$ Οπού q η δομή του δέντρου που αντιστοιχεί ένα label στον αντίστοιχο δείκτη φύλλου. Για να γίνει γνωστή η κάθε συνάρτηση του μοντέλου ελαχιστοποιούμε το ακόλουθο

$$L = \sum_{t} l(\widehat{y_t}, y_t) + \sum_{k} \Omega(f_k)$$

Οπού $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$ όπου l είναι μια διαφοροποιήσιμη κυρτή συνάρτηση απώλειας που μετρά τη διαφορά μεταξύ της πρόβλεψης $\widehat{y_t}$ και του στόχου y_t και το T ο αριθμός φύλλων αναλόγως με τον αριθμό τον χαρακτηριστικών του συνόλου δεδομένων.

3.3.1. Gradient Tree Boosting

Για την βελτιστοποίηση τον συναρτήσεων που περιλαμβάνονται στο μοντέλο σαν παράμετροι που δεν μπορούν να υπολογιστούν με συμβατικούς μεθόδους βελτιστοποίησης σε Ευκλείδειο χώρο θα εκπαιδευτούν με πρόσθετο τρόπο. Όπου $\hat{y}_t^{(i)}$ η πρόβλεψη την χρονική στιγμή t στην i-th επανάληψη

$$L^{(i)} = \sum_{t=1}^{n} l\left(y_t, \hat{y}_t^{(i-1)} + f_i(x_t)\right) + \Omega(f_i)$$

Με n το πλήθος των ετικετών. Προσέγγιση δεύτερης τάξης μπορεί να χρησιμοποιηθεί για τη γρήγορη βελτιστοποίηση του στόχου σε γενικό πεδίο.

$$L^{(i)} \cong \sum_{t=1}^{n} \left[l\left(y_t, \hat{y}_t^{(i-1)}\right) + g_t f_i(x_t) + \frac{1}{2} h_t f_i^2(x_t) \right] + \Omega(f_i)$$

Οπού $g_t = \partial_{\hat{y}_t^{(i-1)}} l\left(y_t, \hat{y}_t^{(i-1)}\right)$ και $h_t = \partial^2_{\hat{y}_t^{(i-1)}} l\left(y_t, \hat{y}_t^{(i-1)}\right)$

και αφαιρώντας τις σταθερές έχουμε:

$$\hat{L}^{(i)} = \sum_{t=1}^{n} \left[g_t f_i(x_t) + \frac{1}{2} h_t f_i^2(x_t) \right] + \Omega(f_i)$$

Ορίζουμε j ως σύνολο φύλλων $I_j = \{t | q(x_t) = j\}$ και επεκτείνοντας Ω έχουμε

$$L^{(i)} \cong \sum_{t=1}^{n} \left[l\left(y_t, \hat{y}_t^{(i-1)}\right) + g_t f_i(x_t) + \frac{1}{2} h_t f_i^2(x_t) \right] + \gamma T + \frac{1}{2} \lambda \sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T} \left[\left(\sum_{t \in I_j} g_t \right) w_j + \frac{1}{2} \left(\sum_{t \in I_j} h_t + \lambda \right) w_j^2 \right] + \gamma T$$

Για μια στατική δομή $q(x)$ μπορούμε να υπολογίζουμε τα βέλτιστα βάρη w_j^* του φύλλου j κατά

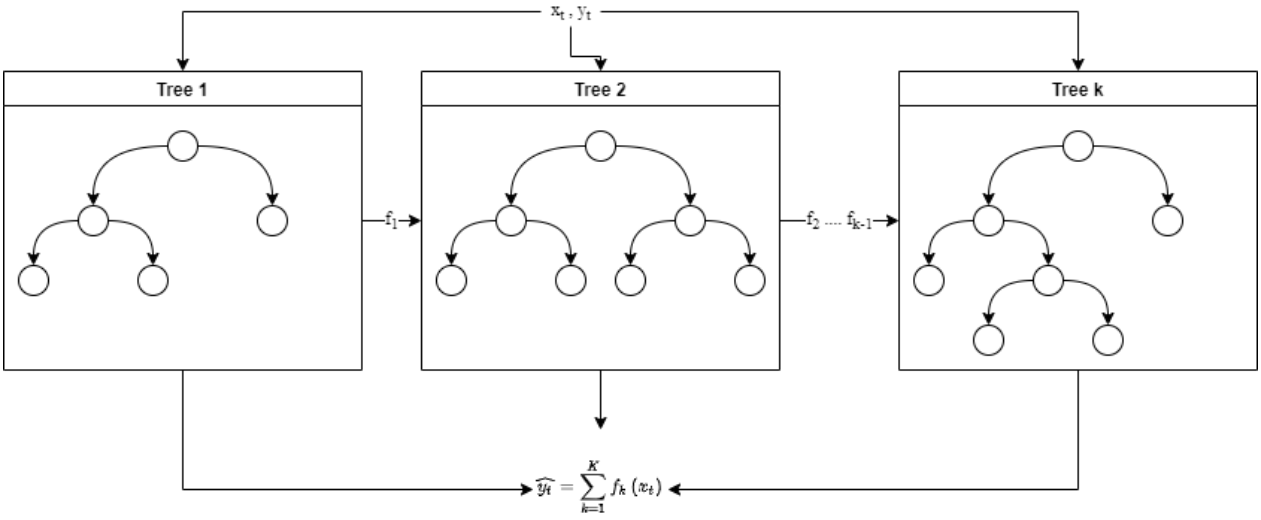
$$w_j^* = -\frac{\sum_{t \in I_j} g_t}{\sum_{t \in I_j} h_t + \lambda}$$

Και υπολογίζουμε την αντίστοιχη βέλτιστη τιμή κατά

$$\hat{L}^{(i)}(q) = -\frac{1}{2}\sum_{j=1}^{T} \frac{\sum_{t \in I_j} g_t}{\sum_{t \in I_j} h_t + \lambda} \gamma T$$

Η παραπάνω συνάρτηση μπορεί να χρησιμοποιηθεί ως βαθμολογητής για τη μέτρηση της ποιότητας δομής δέντρου q.

Το μοντέλο παίρνει την τελική μορφή.



3.1 Αρχιτεκτονική XGB

3.4. Μακράς βραχυπρόθεσμης μνήμης

Η μακράς βραχυπρόθεσμης μνήμης (Long short-term memory) είναι μια αρχιτεκτονική επαναλαμβανομένου νευρωνικού δικτύου(Recurrent neural network) με συνδέσεις ανατροφοδότησης που του επιτρέπουν να μάθουν όχι μόνο σε μεμονωμένα σημεία δεδομένων όπως τα νευρωνικά δίκτυα πρόσω τροφοδότησης(feedforward networks) αλλά και σε ακολουθίες δεδομένων.

Δημιουργήθηκαν για την αντιμετώπιση του εξαφανιζόμενου προβλήματος κλίσης που το εμποδίζουν να μάθει εξαρτήσεις μακράς εμβέλειας [14] και έχουν χρησιμοποιηθεί επιτυχώς για την πρόβλεψη χρονοσειρών [15] [16]. Σε μερικές περιπτώσεις βλέπουμε βελτίωση από κλασικές μορφές πρόβλεψης όπως είναι το ARIMA κατά 85% [17]. Για την υλοποίηση θα χρησιμοποιηθεί η βιβλιοθήκη pytorch [18].

Το LSTM παίρνει την μορφή:

$$i_t = \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi})$$

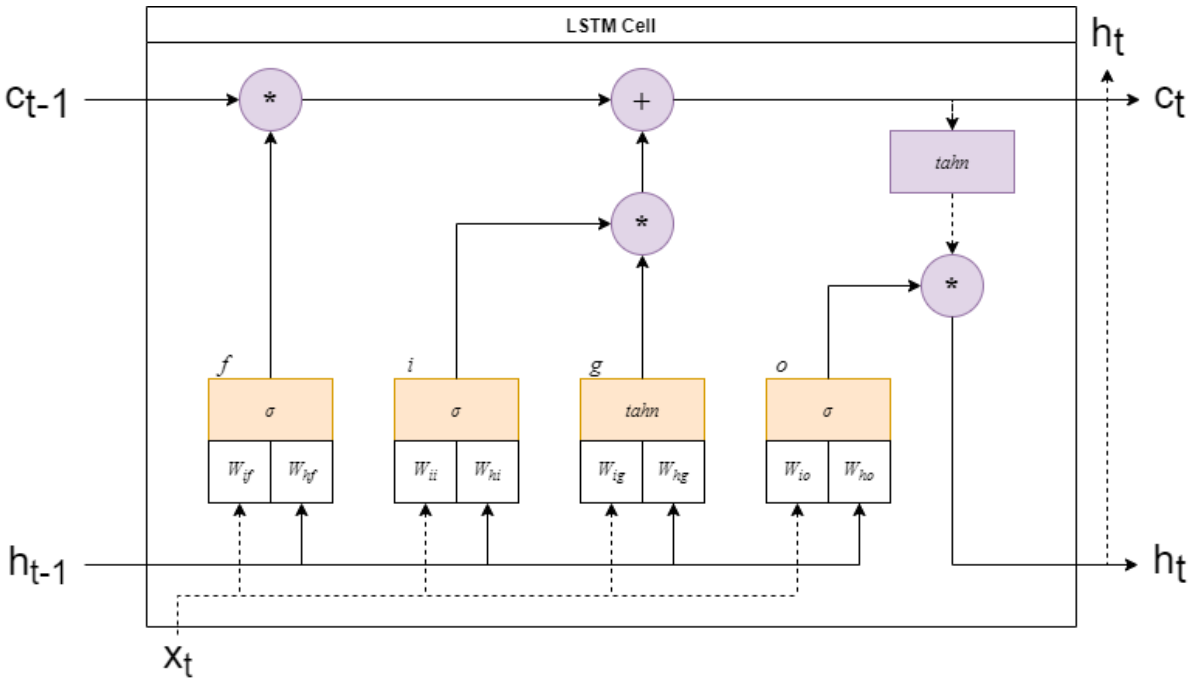
$$f_t = \sigma\big(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\big)$$

$$g_t = tanh\big(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\big)$$

$$o_t = \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho})$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

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



Εικόνα 3.2 Αρχιτεκτονική LSTM

Οπού h_t είναι η κρυφή κατάσταση την στιγμή t,

W είναι ο πίνακας βαρών (πχ W_{ii} πίνακας βαρών πύλης εισόδου στην είσοδο),

ομοίως το bias,

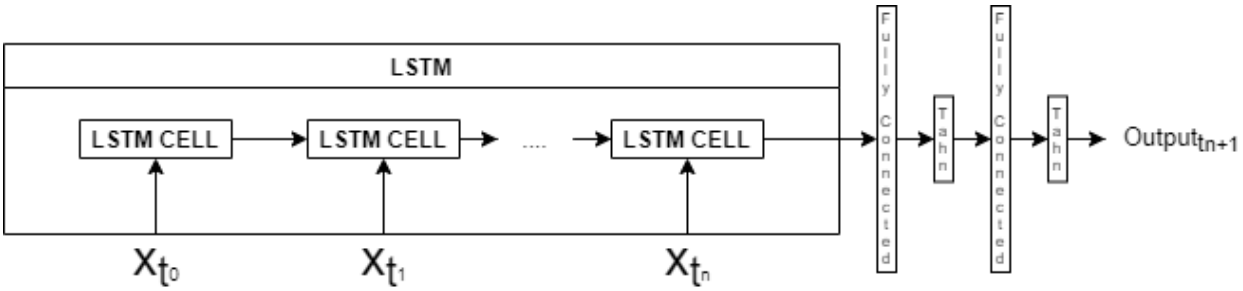
c_t είναι η κατάσταση στοιχείου την στιγμή t,x_t είναι η είσοδος την στιγμή t,

i_t είναι η πύλη εισόδου (input gate),f_t είναι η πύλη λήθης (forget gate),

g_t είναι η πύλη στοιχείου (cell gate),o_t είναι η πύλη εξόδου (output gate)

σ είναι η σιγμοειδής συνάρτηση, $\odot$ είναι η πράξη Hadamard,

Η τελική μορφή είναι :



3.3 Τελική μορφή μοντέλου LSTM

Με μέγεθος παρτίδας 32 ακολουθίες, μήκος ακολουθίας 1 μέρας δηλαδή 24 δείγματα, βάθος 1 και μέγεθος κρυφού επιπέδου 128.

3.5. Εκπαίδευση

Για την εκπαίδευση των μοντέλων θα χρησιμοποιηθεί η τεχνική διασταυρωμένη επικύρωση με αναζήτηση πλέγματος (GridSearchCV) για την επιλογή των υπερπαραμέτρων ώστε να αποφευχθεί η υπερπροσαρμογή (overfitting) ή υποπροσαρμογή (underfitting) του μοντέλου στο κάθε σύνολο εκπαίδευσης. Ο αλγόριθμος GridSearchCV λειτουργεί χωρίζοντας το σύνολο των παρεχόμενων δεδομένων σε ν υποσύνολα και κάνει εξαντλητική αναζήτηση σε καθορισμένες τιμές παραμέτρων για το μοντέλο.

Μετά την επιλογή των υπερπαραμέτρων για τα τελικά αποτελέσματα τα σύνολα εκπαίδευσης θα χωριστούν ως εξής, για τη δοκιμή θα χρησιμοποιηθεί η τελευταία εβδομάδα $\{x_T, x_{T-1}, \ldots, x_{T-i}\}$ όπου T ο αριθμός δειγμάτων και $i = T - 7 * 24$, ο διαχωρισμός του συνόλου εκπαίδευσης και η επικύρωση θα γίνει με τυχαίο τρόπο οπού θα χρησιμοποιηθεί το 80% του $\{x_{T-i}\}$ για την εκπαίδευση και 20% για την επικύρωση.

3.5.1. Μακράς βραχυπρόθεσμης μνήμης

Η εκπαίδευση ανάμεσα στα μοντέλα μηχανικής μάθησης και βαθιάς μάθησης έχει αρκετές ομοιότητες όμως με κάποιες μεγάλες διαφορές. Μαζί με τον παραπάνω διαχωρισμό τον δεδομένων λόγο του μικρού μεγέθους θα γίνει η τεχνητή διόγκωση του συνόλου δεδομένων με την τεχνική ολισθαίνων παραθύρων με ακολουθία

$$S_k = [x_k, x_{k+1}, \ldots, x_{k+23}] \text{ με } k \in [0, i-24).$$

Για την εκπαίδευση θα χρησιμοποιηθεί ο αλγόριθμος βελτιστοποίησης Adam [19] και η συνάρτηση λάθους μέσης απόλυτης τιμής. Ο αλγόριθμος Adam είναι μια μέθοδος που υπολογίζει προσαρμοστικά ποσοστά μάθησης για κάθε παράμετρο, πραγματοποιώντας μεγαλύτερες ενημερώσεις για σπάνιες και μικρότερες ενημερώσεις για συχνές παραμέτρους [20].

3.5.1.1. Αλγόριθμος Adam

Παρακάτω παρατίθεται ο ψευδοκώδικας του αλγόριθμου με Μέγεθος βήματος ή ρυθμός μάθησης α = 0.001, bias $\beta_1 = 0.9, \beta_2 = 0.999$ και ϵ =1e-8.

Έστω $f(\theta)$ μια στοχαστική κλιμακωτή συνάρτηση που μπορεί να διαφοροποιηθεί με αναφορά στις παραμέτρους θ. Ο σκοπός είναι η ελαχιστοποίηση της συνάντησης $\mathbb{E}[f(\theta)]$ με αναφορά στις παραμέτρους θ. Με $f_1(\theta), \ldots f_T(\theta)$ συμβολίζουμε τα επιτεύγματα της στοχαστικής συνάρτησης σε επόμενα χρονικά βήματα $1, \ldots, T$. Με $g_t \leftarrow \nabla_\theta f_t\left(\theta_{(t-1)}\right)$ δηλώνουμε την κλίση, δηλαδή το διάνυσμα μερικών παραγώγων του f_t με αναφορά στις παραμέτρους θ που αξιολογείται στο χρονικό βήμα t.

Ο αλγόριθμος ενημερώνει τους εκθετικούς κινούμενους μέσους όρους της κλίσης m_t και της τετραγωνικής κλίσης v_0 όπου οι υπερπαράμετροι β_1, β_2 ελέγχουν τους εκθετικούς ρυθμούς

διάσπασης αυτών των κινούμενων μέσων όρων. Οι ίδιοι οι κινητοί μέσοι όροι είναι εκτιμήσεις της 1ης στιγμής (η μέση τιμή) και της 2ης ακατέργαστης στιγμής (η μη κεντρική διακύμανση) της κλίσης.

Απαιτείται: α: Μέγεθος βήματος

Απαιτείται: $\beta_1, \beta_2 \in [0, 1)$: Εκθετικές φθίνουσες τιμές για εκτιμήσεις στιγμής

Απαιτείται: $f(\theta)$: Στοχαστική αντικειμενική συνάρτηση με παραμέτρους θ

Απαιτείται: θ_0: Διάνυσμα αρχικών παραμέτρων

$m_0 \leftarrow 0$ (Αρχικοποίηση 1ου διανύσματος στιγμής)

$v_0 \leftarrow 0$ (Αρχικοποίηση 2ου διανύσματος στιγμής)

$t \leftarrow 0$ (Αρχικοποίηση χρονικού βήματος)

Όσο θ_t δεν έχει συγκλίνει

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t\left(\theta_{(t-1)}\right)$ (Λάβετε κλίσεις με αναφορά στο στοχαστικό στόχο στο χρονικό βήμα t)

$m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Ενημέρωση εκτίμησης 1ου διανύσματος στιγμής)

$v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Ενημέρωση εκτίμησης 2ου ακατέργαστου διανύσματος στιγμής)

$\widehat{m}_t \leftarrow \frac{m_t}{(1-\beta_1^t)}$ (Υπολογισμός διόρθωσης εκτίμησης 1ου διανύσματος στιγμής)

$\hat{u}_t \leftarrow \frac{v_t}{(1-\beta_2^t)}$ (Υπολογισμός διόρθωσης εκτίμησης 2ου ακατέργαστου διανύσματος στιγμής)

$\theta_t \leftarrow \theta_{(t-1)} - \frac{a}{\sqrt{\hat{v}_t}+\epsilon}\widehat{m}_t$ (Ενημέρωση παραμέτρων)

Τέλος Όσο

Επιστροφή θ_t

3.4 Ψευδοκώδικας Adam

Για τον υπολογισμό της απόκλισης της πρόβλεψης από την πραγματική τιμή θα χρησιμοποιηθεί η συνάρτηση Μέσου απολύτου λάθους στο τμήμα 4.1.1.

Κεφάλαιο 4: Αποτελέσματα

Στον τομέα παλινδρόμηση μηχανικής μάθησης δεν είναι δυνατή αξιολόγηση ενός μοντέλου μόνο από μια μετρική, η αξιολόγηση γίνεται εφικτή από την σύγκριση τους. Για την σύγκριση τον μοντέλων θα χρησιμοποιηθούν οι εξής μετρικές.

4.1. Μετρικές

4.1.1. Μέσο Απόλυτο Λάθος

Το μέσο απόλυτο λάθος (MAE) μετρά το μέσο απόλυτο μέγεθος των σφαλμάτων σε ένα σύνολο προβλέψεων.

$$\frac{\sum_{i=0}^{n-1}|(y_i - \hat{y}_i)|}{n}$$

Όπου $\hat{y}_i$ η πρόβλεψη της τιμής του δείγματος i, το y_i είναι η πραγματική τιμή και το n είναι ο αριθμός τον δειγμάτων.

4.1.2. Ρίζα Μέσου Τετραγωνικού Λάθους

Η ρίζα τετραγωνικού μέσου λάθους (RMSE) ή ρίζα τετραγωνική μέση απόκλιση είναι ένας ποσοτικός κανόνας που επίσης υπολογίζει το μέσο λάθος [21]. Είναι η τυπική απόκλιση των σφάλματων της πρόβλεψης. Λόγο του τετραγωνισμού το RMSE τιμωρεί τα μοντέλα με τις μεγαλύτερες αποκλίσεις από τη γραμμή παλινδρόμησης.

$$\sqrt{\frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{n}}$$

Όπου $\hat{y}_i$ η πρόβλεψη της τιμής του δείγματος i, το y_i είναι η πραγματική τιμή και το n είναι ο αριθμός τον δειγμάτων.

4.1.3. Συντελεστής προσδιορισμού

Ο συντελεστής προσδιορισμού [22] μπορεί να ερμηνευτεί ως το ποσοστό της διακύμανσης της εξαρτημένης μεταβλητής με την ανεξάρτητη μεταβλητή. Σε προβλήματα παλινδρόμησης τα μοντέλα μηχανικής μάθησης αξιολογούνται συνήθως με τις μετρικές MAE, RMSE και MAPE, λόγο μη θετικών τιμών δεν μπορεί να χρησιμοποιηθεί η MAPE. Πρόσφατη έρευνα υποστηρίζει ότι ο συντελεστής προσδιορισμού είναι μια από της πιο αξιόπιστες [23]. Η μετρική αυτή δείχνει πως θα ανταπεξέλθει το μοντέλο σε δείγματα που δεν έχουν παρατηρηθεί και όπως καταλαβαίνουμε δεν μπορεί να χρησιμοποιηθεί για την σύγκριση διαφορετικών μοντέλων και σύνολο δεδομένων για τον λόγο της εξάρτησης του από αυτών.

Ο συντελεστής προσδιορισμού (R^2) δείχνει το ποσοστό της διακύμανσης ανάμεσα στην πρόβλεψη με την πραγματική τιμή και κυμαίνεται ανάμεσα 1.0 με $-\infty$ με το 1.0 ως το καλύτερο αποτελέσματα.

Αν $\hat{y}_i$ η πρόβλεψη την στιγμή i και y_i η πραγματική τιμή έχουμε

$$R^2 = 1 - \frac{\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2}{\sum_{i=0}^{n-1}(y_i - \bar{y}_i)^2}$$

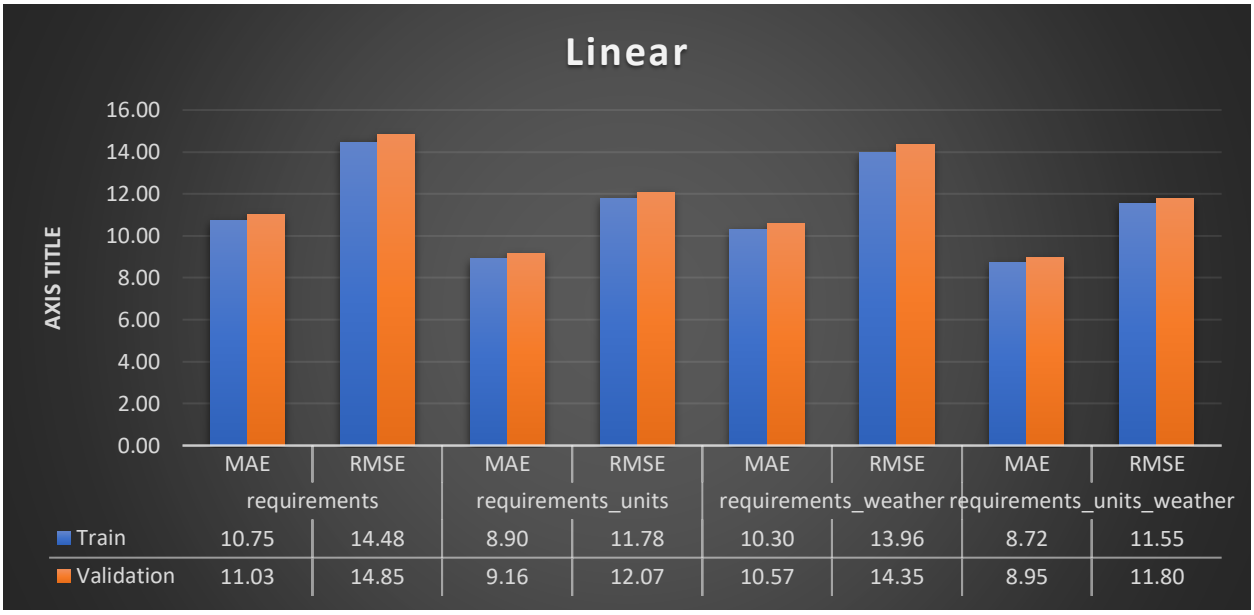
Οπού $\bar{y}_i = \frac{1}{n}\sum_{i=0}^{n-1} y_i$ *και* $\sum_{i=0}^{n-1}(y_i - \hat{y}_i)^2 = \sum_{i=0}^{n-1} \varepsilon_i^2$

4.2. Αποτελέσματα Μοντέλων

Για την υλοποίηση τον μοντέλων μηχανικής μάθησης θα χρησιμοποιηθεί η βιβλιοθήκη Sci-kit learn [24].

4.2.1. Πολλαπλή γραμμική παλινδρόμηση

Όπως μπορούμε να καταλάβουμε δεν υπάρχουν υπερπαράμετροι σε ένα κλασικά γραμμικό μοντέλο που μπορούμε να ψάξουμε.



4.1 Αποτελέσματα Γραμμικής Παλινδρόμησης MAE-RMSE

Dataset	**Train**	**Validation**
requirements	81.32%	78.56%
requirements_units	87.63%	85.85%
requirements_weather	82.62%	80.00%

4.2Αποτελέσματα Γραμμικής Παλινδρόμησης R2

Λαμβάνοντας υπόψιν την απλότητα του μοντέλου βλέπουμε αξιοσέβαστα αποτελέσματα και παρατηρούμε τη βελτίωση τον αποτελεσμάτων με την εισαγωγή νέων χαρακτηριστικών. Το υποσύνολο Διαθεσιμότητα Μονάδων προσφέρει εμφανή βελτίωση διαφορετικά με τα Δεδομένα καιρού, μια τάση που θα παρατηρηθεί και στα υπόλοιπα μοντέλα.

4.2.2. Παλινδρόμηση πλησιέστερων γειτόνων

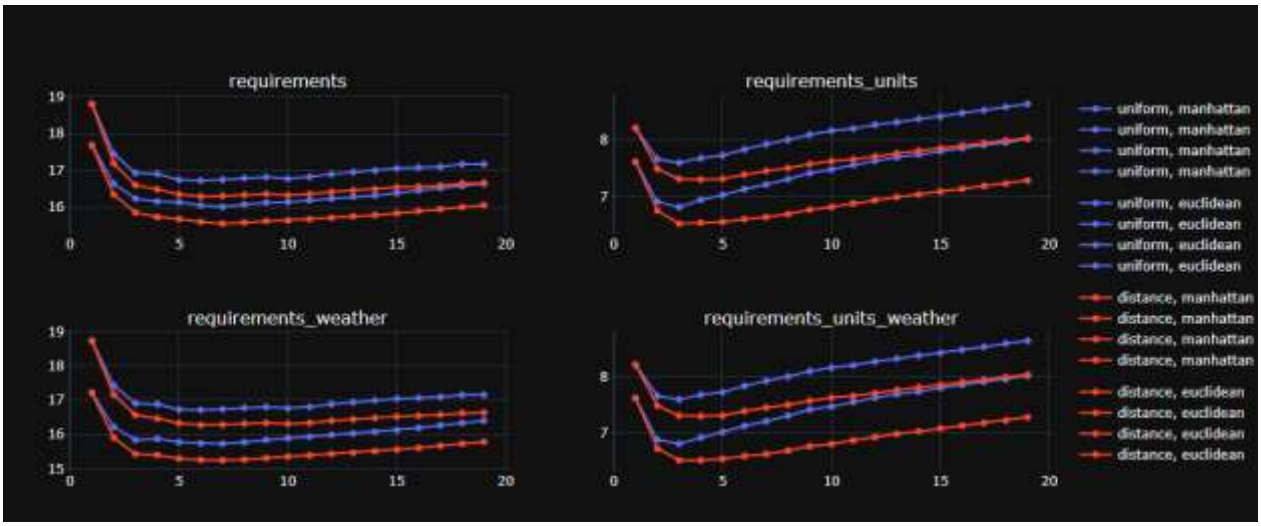
Εκτελώντας την αναζήτηση πλέγματος με υπερπαραμέτρους

- Γείτονες 'n_neighbors' : 1 μέχρι 20
- Βάρη 'weights' : ['uniform', 'distance']
- Απόστασή 'metric' : ['manhattan' , 'euclidean']

τα αποτελέσματα στην εικόνα 4.2.2.1 με $CV = 5$ υποσυνόλων

$mean\_test\_score = \frac{\sum_{i=1}^{CV} split\_test\_score_i}{CV}$.

Παρατηρούμε την προαναφερόμενη τάση για τα σύνολα δεδομένων αλλά και μια αισθητή βελτίωση στις προβλέψεις



4.3 Υπερπαράμετροι KNN

Παίρνουμε τις καλύτερες υπερπαραμέτρους για κάθε σύνολο δεδομένων

- requirements: {*'metric'*: 'manhattan', 'n_*neighbors*': 6, *'weights'*: 'distance'}

  mean_test_score = 13.5034

- requirements_units: {*'metric'*: 'manhattan', 'n_*neighbors*': 4, *'weights'*: 'distance'}

  mean_test_score = 6.3861

- requirements_weather: {*'metric'*: 'manhattan', 'n_*neighbors*': 7, *'weights'*: 'distance'} *mean_test_score* = 13.2203

- requirements_units_weather: {*'metric'*: 'manhattan', *'n_neighbors'*: 4, *'weights'*: 'distance'} *mean_test_score* = 6.3722

Και έχουμε τα εξής αποτελέσματα

4.4 Αποτελέσματα παλινδρόμηση πλησιέστερων γειτόνων MAE-RMSE

Dataset	**Train**	**Validation**
requirements	100.00%	68.85%
requirements_units	100.00%	93.46%
requirements_weather	100.00%	69.58%
requirements_units_weather	100.00%	93.46%

4.5Αποτελέσματα παλινδρόμηση πλησιέστερων γειτόνων R2

Λόγω του υπολογισμού των βαρών ανάλογα με το αντίστροφο της απόστασης από το σημείο ερωτήματος, εξηγεί το λάθος εκπαίδευσης = 0, καθώς τα δεδομένα εκπαίδευσης είναι ήδη γνωστά και κυριαρχούν τα αποτελέσματα.

Βλέπουμε ότι ο αλγόριθμος εξάγει την περισσότερη πληροφορία από την διαθεσιμότητα μονάδων, με τα δεδομένα καιρού να συμβάλουν ελάχιστα έως και καθόλου στην τελική πρόβλεψη.

4.2.3. Παλινδρόμηση XGBOOST

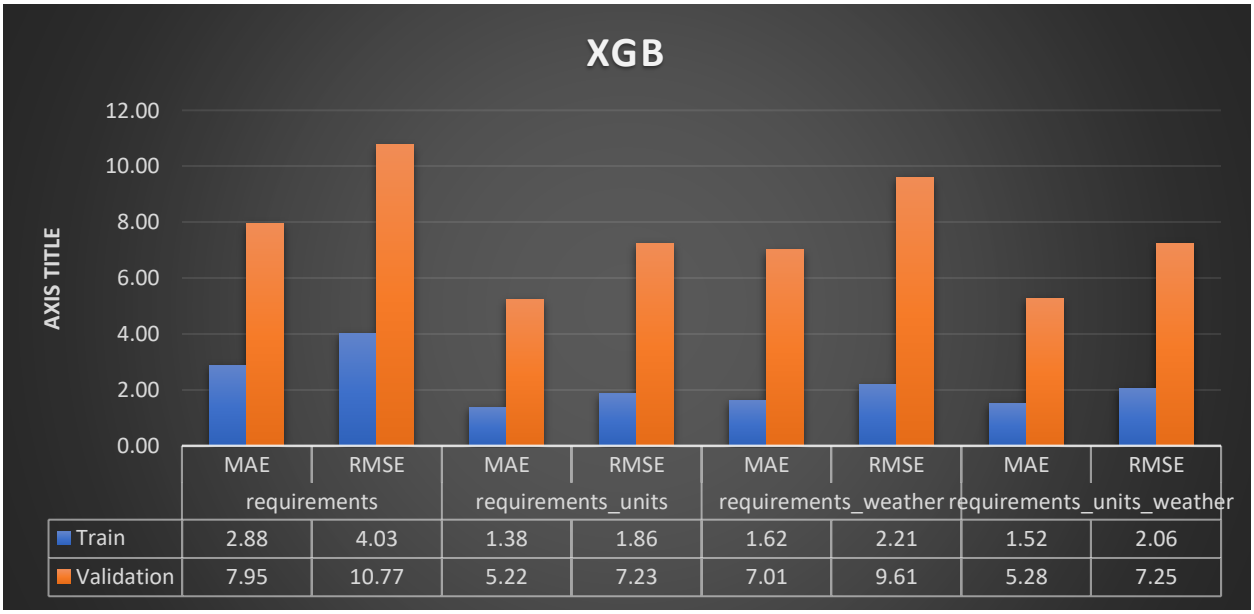
Η περιοχή αναζήτησης αποτελείται από τις υπερπαραμέτρους:

- βαθμός μάθησης (learning_rate): [0,09],
- μέγιστο βάθος (max_depth): 1 μέχρι 15,
- n εκτιμητές (n_estimators): [100,150,200],
- Υπό δείγμα (subsample): [0,8,1],
- δείγμα στήλης ανά δέντρο ("colsample_bytree): [0.8,1],
- δείγμα στήλης ανά επίπεδο (colsample_bylevel): [0,8,1],
- τυχαία κατάσταση (random_state): [96]

Λόγω του πλήθους τον υπερπαραμέτρων η αναζήτηση είναι αρκετά χρονοβόρα.

Μετά την εκτέλεση της αναζήτησης έχουμε τις εξής υπερπαραμέτρους για κάθε σύνολο δεδομένων.

- Requirements: {**'colsample_bylevel'**: 0.8, **'colsample_bytree'**: 1, **'learning_rate'**: 0.09, 'max_depth': 9, **'n_estimators'**: 150**, 'random_state'**: 96, **'subsample'**: 1} ***mean_test_score***: = 11.07986888
- requirements_units: {**'colsample_bylevel'**: 1, **'colsample_bytree'**: 1, **'learning_rate'**: 0.09, **'max_depth'**: 9, **'n_estimators'**: 150, **'random_state'**: 96, **'subsample'**: 0.8} ***mean_test_score*** = 5.218708281
- **requirements_weather**: {**'colsample_bylevel'**: 0.8**, 'colsample_bytree'**: 1, **'learning_rate'**: 0.09, **'max_depth'**: 9**, 'n_estimators'**: 150, **'random_state'**: 96, **'subsample'**: 0.8} ***mean_test_score*** = 8.308658536
- requirements_units_weather: {**'colsample_bylevel'**: 0.8, **'colsample_bytree'**: 1, **'learning_rate'**: 0.09**, 'max_depth'**: 9, **'n_estimators'**: 100, **'random_state'**: 96, **'subsample'**: 1} ***mean_test_score*** *= 5.079820192*



4.6 Αποτελέσματα Παλινδρόμησης XGBOOST MAE-RMSE

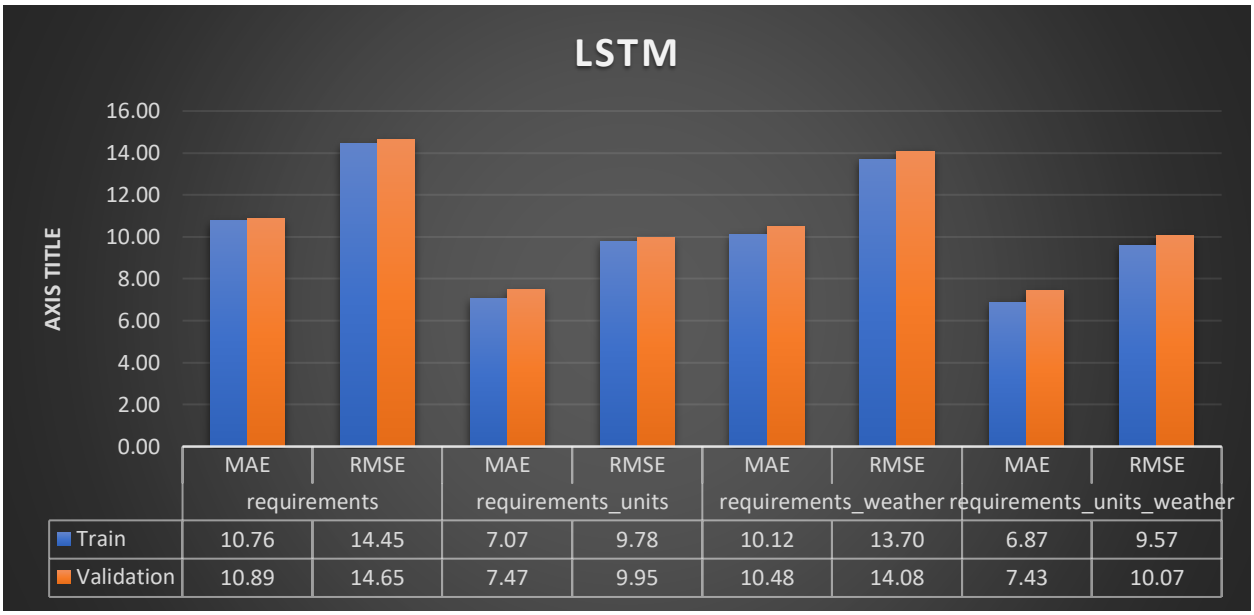
Dataset	**Train**	**Validation**
requirements	98.55%	88.74%
requirements_units	99.69%	94.92%
requirements_weather	99.57%	91.03%
requirements_units_weather	99.62%	94.89%

4.7Αποτελέσματα Παλινδρόμησης XGBOOST R2

Βλέπουμε τα έως τώρα καλύτερα αποτελέσματα αν και υπάρχει η πιθανότητα υπερβολικής προσαρμογή στα δεδομένα εκπαίδευσης.

4.2.4. Μακράς βραχυπρόθεσμης μνήμης

Χρησιμοποιώντας την τεχνική της πρόωρης διακοπής, οπού αποθηκεύουμε το τρέχων καλύτερο μοντέλο και συνεχίζουμε την εκπαίδευση για την καλύτερη εποχή + 20 ανανεώνοντας την καλύτερη εποχή και το μοντέλο όταν βρεθεί καινούργιο ελάχιστο έχουμε τα εξής αποτελέσματα.



4.8 Αποτελέσματα Μακράς βραχυπρόθεσμης μνήμης MAE-RMSE

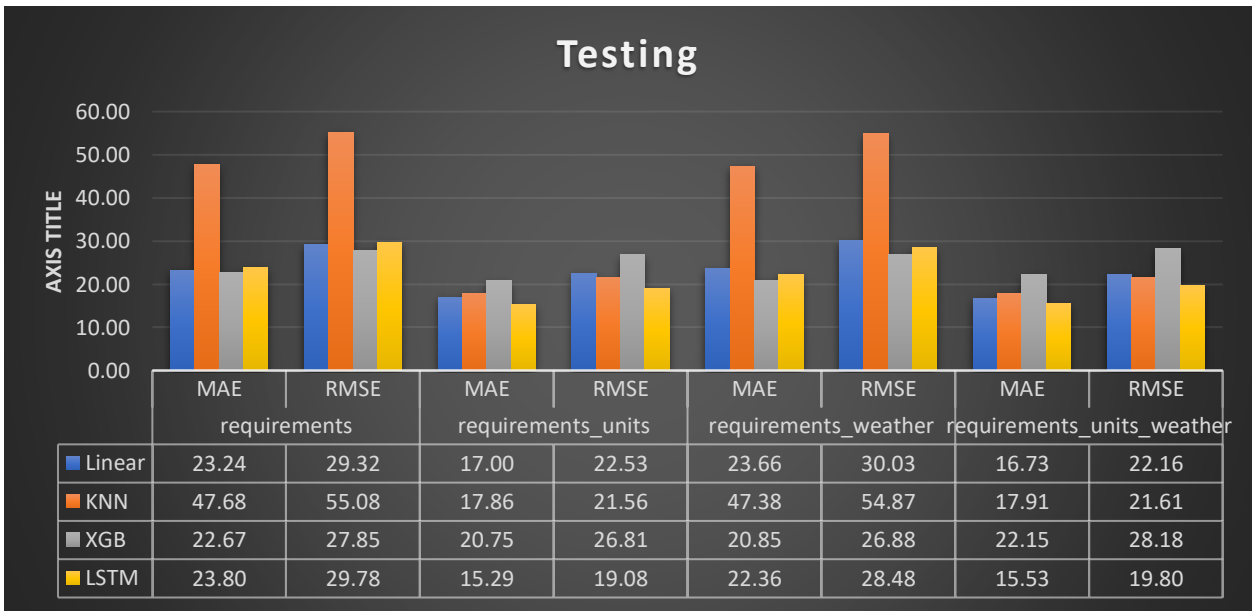
Dataset	**Train**	**Validation**
requirements	81.39%	79.14%
requirements_units	91.47%	90.38%
requirements_weather	83.28%	80.73%
requirements_units_weather	91.83%	90.14%

4.9 Αποτελέσματα Μακράς βραχυπρόθεσμης μνήμης R2

Δεδομένου του σχετικού μικρού όγκου δεδομένων, για ένα LSTM το οποίο μπορεί να εξάγει και εποχιακές πληροφορίες, ένα από τα μεγαλύτερα μειονεκτήματα είναι ότι δεν υπάρχει παραπάνω από 1 χρόνια δεδομένων. Παρόλα αυτά έχει αξιοπρεπέστατα αποτελέσματα που θα βελτιωθούν με την πάροδο του χρόνου.

4.2.5. Πειραματικά Αποτελέσματα

Έχοντας εκπαιδεύσει τα μοντέλα με την πλειοψηφία των δεδομένων θα χρησιμοποιηθεί η τελευταία εβδομάδα για να διαπιστώσουμε πως θα ανταπεξέλθει το καθένα σε πραγματική λειτουργία. Συγκρίνοντας τα αποτελέσματα, στα διαγράμματα 4.10 και 4.11, παρατηρούμε μεγάλες αποκλίσεις από τα δεδομένα της εκπαίδευσης με τα αποτελέσματα της δοκιμής. Η πανδημία SARS-CoV-2 που έχει παραλύσει τον πλανήτη τα τελευταία 2 χρόνια έχει επιφέρει μεγάλες προσαυξήσεις τιμών πρώτον υλών την εξεταζόμενη περίοδο. Όμως λόγο αυτής της αστάθειας τιμών είναι μια καλή περίοδος εξέτασης για να διαπιστωθεί πως θα ανταπεξέλθουν τα μοντέλα σε μη κανονικές συνθήκες αγοράς.



	MAE requirements	RMSE requirements	MAE requirements_units	RMSE requirements_units	MAE requirements_weather	RMSE requirements_weather	MAE requirements_units_weather	RMSE requirements_units_weather
Linear	23.24	29.32	17.00	22.53	23.66	30.03	16.73	22.16
KNN	47.68	55.08	17.86	21.56	47.38	54.87	17.91	21.61
XGB	22.67	27.85	20.75	26.81	20.85	26.88	22.15	28.18
LSTM	23.80	29.78	15.29	19.08	22.36	28.48	15.53	19.80

4.10 Σύγκριση μετρικων MAE-RMSE τον αποτελεσμάτων της τελευταίας εβδομάδας όλων των μοντέλων

Dataset	Linear	KNN	XGB	LSTM
requirements	13.44%	-205.59%	21.90%	10.70%
requirements_units	48.88%	53.20%	27.64%	63.35%
requirements_weather	9.18%	-203.21%	27.22%	18.31%
requirements_units_weather	50.56%	52.95%	20.00%	60.51%

4.11 Σύγκριση R2 τον αποτελεσμάτων της τελευταίας εβδομάδες όλων των μοντέλων

Κατά την πειραματική λειτουργία παρατηρούμε ότι τα μοντέλα βγάζουν παρόμοια δεδομένα με καλύτερο το LSTM πράγμα που οδηγεί στο συμπέρασμα ότι ανταπεξέρχεται καλύτερα από τα κλασικά μοντέλα σε περιόδους αστάθειας.

Θα επιλεχθεί το δεύτερο σύνολο δεδομένων (requirements_units) καθώς προσφέρει αδιαμφησβήτητα τα καλύτερα αποτελέσματα και για να μειωθεί η πολυδιαστατικότητα δεν θα προστεθούν τα δεδομένα καιρού καθώς δεν προσφέρουν χρήσιμες πληροφορίες. Ο βασικός λόγος είναι η μεταβλητή renewables που προσφέρει ο ΑΔΜΗΕ καθώς πρόκειται για πρόβλεψη της παραγωγής βάση του καιρού.

4.2.6. Προτεινόμενο μοντέλο

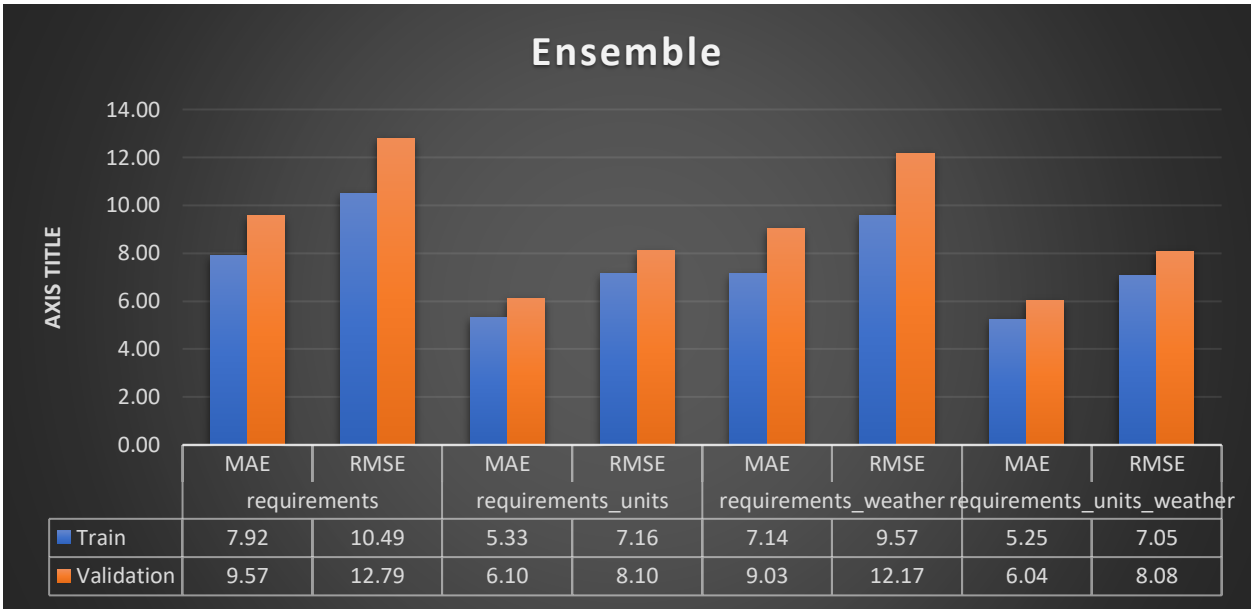
Μια λύση είναι η συλλογή όλων των αποτελεσμάτων και την δημιουργία ενός ensemble οπού θα υπολογίζουμε τον μέσο όρο τον 4 μοντέλων και θα παράγουμε μια καινούργια τιμή.

$$\hat{y}_t(\hat{\boldsymbol{y}}_t) = \frac{1}{numModels} \sum_{i=0}^{model} \hat{y}_{ti}$$

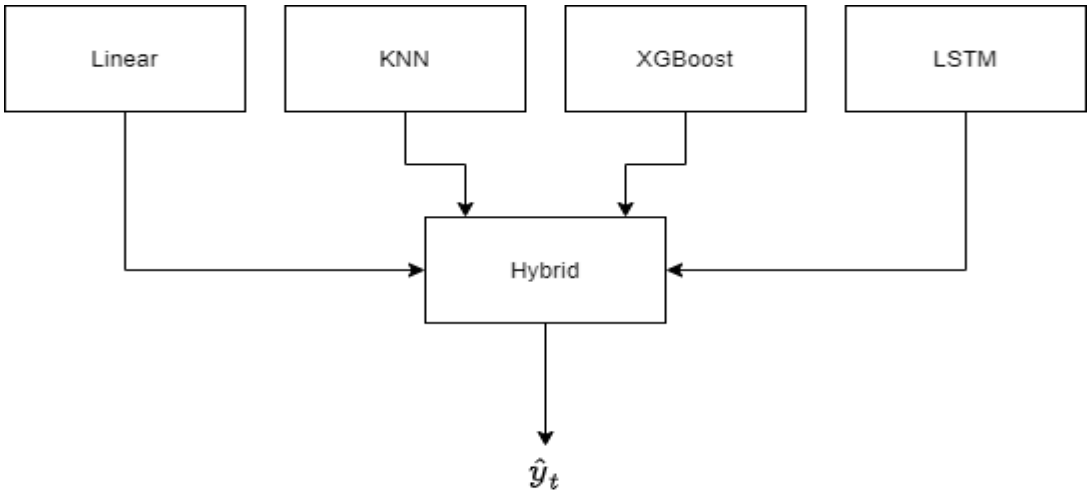
Οπού $\hat{\boldsymbol{y}}_t$ είναι ο πίνακας με τις προβλέψεις την στιγμή t τον μοντέλων, $\hat{y}_{ti}$ η πρόβλεψη την στιγμή t για το μοντέλο i και num_models το πλήθος τω ν μοντέλων.

Dataset	Train	Validation
requirements	90.18%	84.09%
requirements_units	95.43%	93.62%
requirements_weather	91.83%	85.60%
requirements_units_weather	95.56%	93.66%

4.12 Αποτελέσματα Υβριδικού μοντέλου R2

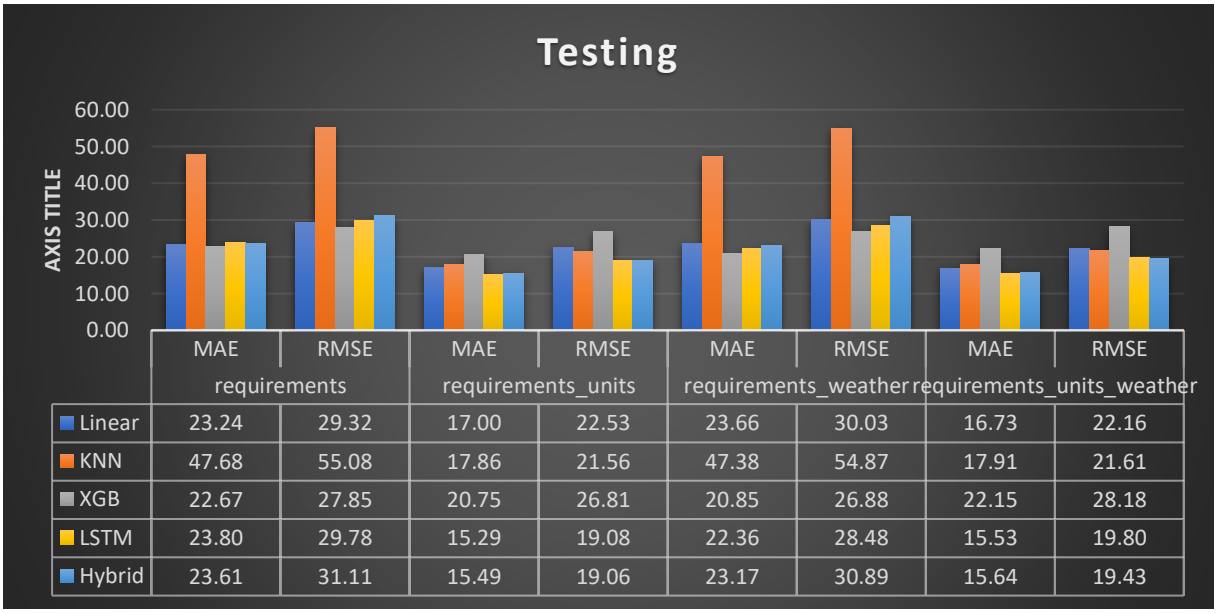


4.13 Αποτελέσματα Υβριδικού μοντέλου MAE-RMSE



4.14 Αρχιτεκτονική υβριδικού

Συγκρίνοντας όλα το μοντέλα στην περίοδο δοκιμής παρατηρούμε ελάχιστα καλύτερα αποτελέσματα.



	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE
	requirements		requirements_units		requirements_weather		requirements_units_weather	
Linear	23.24	29.32	17.00	22.53	23.66	30.03	16.73	22.16
KNN	47.68	55.08	17.86	21.56	47.38	54.87	17.91	21.61
XGB	22.67	27.85	20.75	26.81	20.85	26.88	22.15	28.18
LSTM	23.80	29.78	15.29	19.08	22.36	28.48	15.53	19.80
Hybrid	23.61	31.11	15.49	19.06	23.17	30.89	15.64	19.43

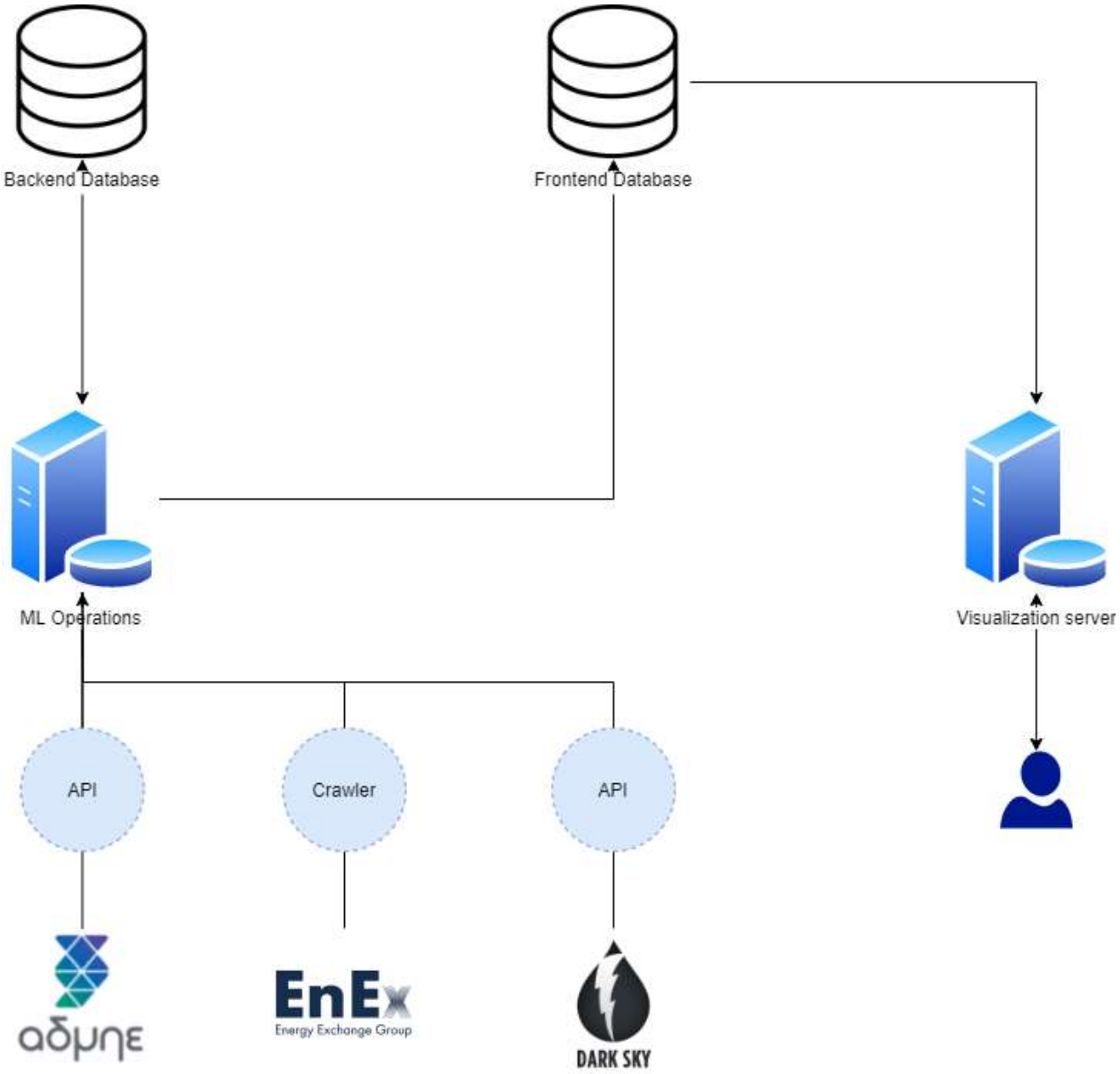
4.15Σύγκριση μετρικών MAE-RMSE τον αποτελεσμάτων της τελευταίας εβδομάδες όλων των μοντέλων

Dataset	**Linear**	**KNN**	**XGB**	**LSTM**	**Hybrid**
requirements	13.44%	-205.59%	21.90%	10.70%	2.50%
requirements_units	48.88%	53.20%	27.64%	63.35%	63.40%
requirements_weather	9.18%	-203.21%	27.22%	18.31%	3.90%
requirements_units_weather	50.56%	52.95%	20.00%	60.51%	61.97%

4.16 Σύγκριση R2 τον αποτελεσμάτων της τελευταίας εβδομάδες όλων των μοντέλων

Κεφάλαιο 5: Πλατφόρμα

Η πλατφόρμα αποτελείτε από 2 βασικά κομμάτια, στον διακομιστή που γίνεται η προεπεξεργασία δεδομένων και η εκπαίδευση, για την καλύτερη και ποιο ασφαλή λειτουργία, και στον διακομιστή που εξυπηρετεί τον χρήστη για την απεικόνιση δεδομένων.



5.1 Αρχιτεκτονική της πλατφόρμας

5.1. Ιστοσελίδα απεικόνισης δεδομένων

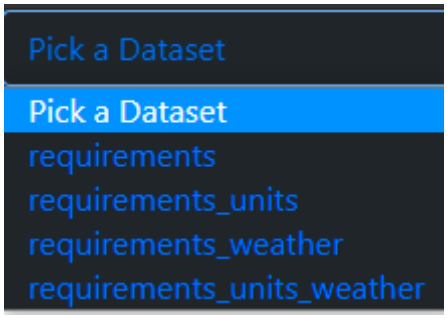
Η πλατφόρμα αναπτύχτηκε με την βιβλιοθήκη απεικόνισης plotly και το διαδικτυακό πλαίσιο(web framework) flask ένα μικροπλαίσιο που δεν έχει αφαιρεμένη διεπαφή βάσης δεδομένων και δεν απαιτεί άλλες βιβλιοθήκες όπως παρόμοια εργαλεία τρίτων [25] [26]. Στην ιστοσελίδα μοντελοποιούνται οι 7 τελευταίες μέρες για λόγους ταχύτητας και ευχρηστίας. Χρησιμοποιείτε το ολοκληρωμένο διαδικτυακό περιβάλλον ανάπτυξης python anywhere όπως προτείνεται και το βιβλίο Python Machine Learning του Sebastian Raschka [27].

5.1.1. Διεπαφή χρήστη

Η πλοήγηση της ιστοσελίδας γίνεται μέσω της μπάρας πλοήγησης οπού μπορεί να επιλεχθεί το μοντέλο ή απεικόνιση δεδομένων και η περίοδος.

5.2Μπάρα πλοήγησης

Η επιλογή συνόλου δεδομένων μπορεί να γίνει με το αναπτυσσόμενο μενού και φορτώνει δυναμικά την ήδη επιλεγμένη σελίδα με τα καινούργια δεδομένα.



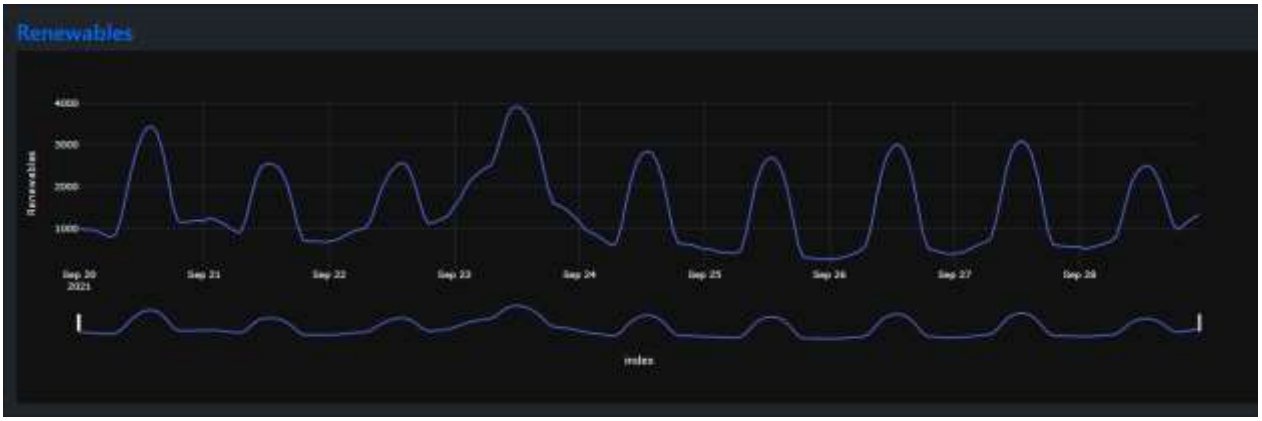
5.3 Αναπτυσσόμενο μενού

5.1.2. Αρχική σελίδα

Η αρχική σελίδα αποτελείτε από τα γραφήματα με την απεικόνιση δεδομένων του τρέχοντος συνόλου δεδομένων.

5.4 Απεικόνιση 'Κηροπήγιων'

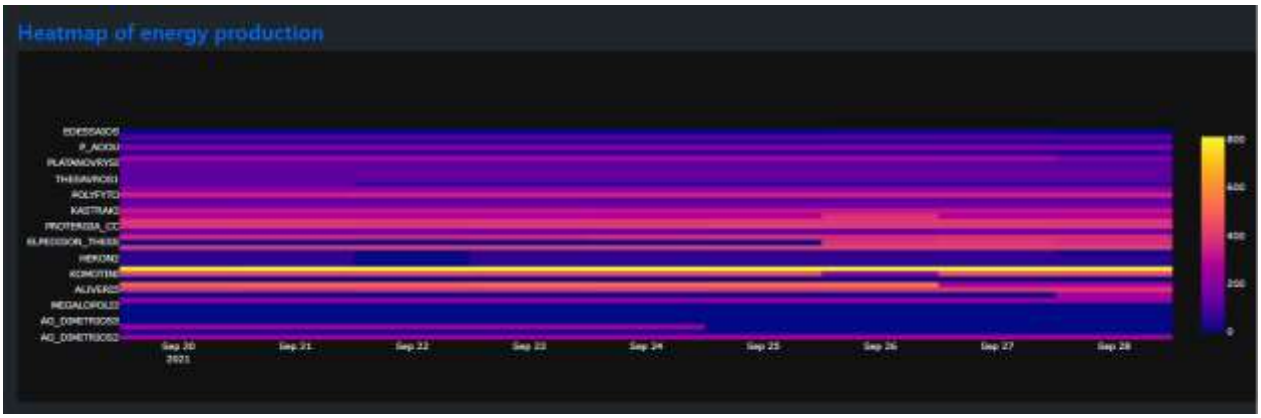
Ο χρήστης μπορεί να μεταβείτε στα υπόλοιπα γραφήματα με την χρήση της γραμμής κύλισης.



5.5 Γράφημα ΑΠΕ

Στα σύνολα δεδομένων με την διαθεσιμότητα μονάδων επίσης παρατίθεται ένας θερμικός χάρτης με την παραγωγή ηλεκτρικής ενέργειας, λόγω του μεγέθους του συνόλου δεδομένων, μαζί με το διάγραμμα κηροπηγίων που δείχνει την υψηλότερη, την χαμηλότερη, την τιμή στην

αρχή της αγοράς και στο τέλος.



5.6 Χάρτης θερμότητας Παραγωγής Ηλεκτρικής ενέργειας

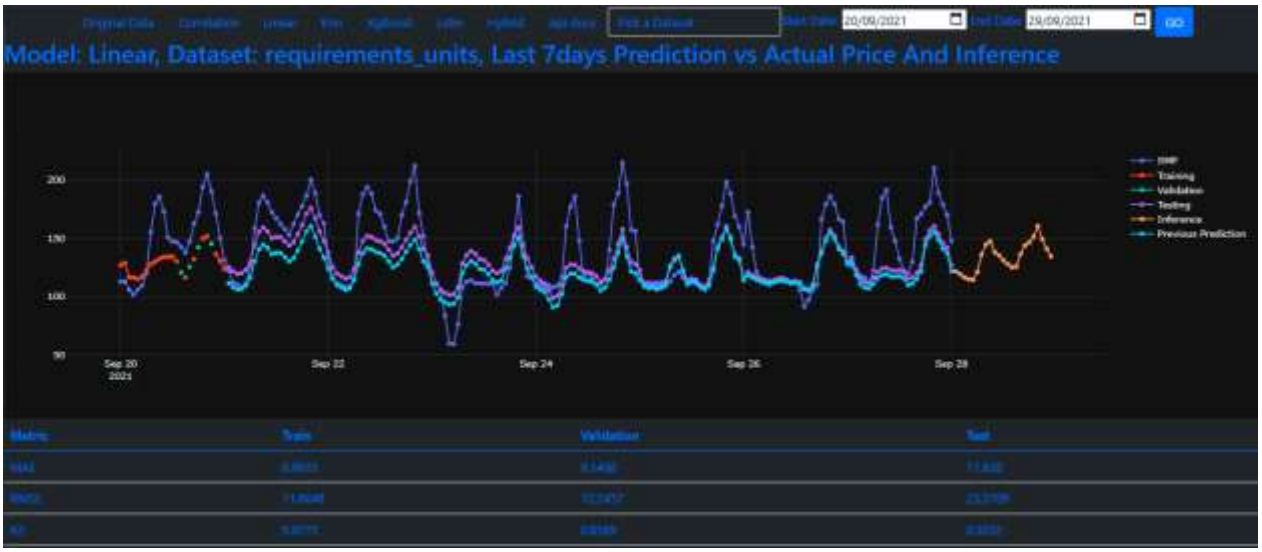
5.1.3. Σελίδα Συσχετίσεων

Επίσης μαζί με την απεικόνιση τον δεδομένων δίνετε και η συσχέτιση του χαρακτηριστικού με την τιμή ηλεκτρονικής ενέργειάς

5.7 Απεικόνιση συσχέτισης τιμής ηλεκτρικής ενέργειας με ΑΠΕ

5.1.4. Σελίδα μοντέλων μηχανικής μάθησης

Για κάθε μοντέλο μηχανικής μάθησης απεικονίζεται η πραγματική τιμή, η πρόβλεψη με τα καινούργια δεδομένα με το όνομα Prediction, το συμπέρασμα με την πρόβλεψη για την αγορά επομένης ημέρας με το όνομα Inference και τα αποθηκευμένα συμπεράσματα με το όνομα Previous Prediction.



5.8 Σελίδα Γραμμικού Μοντέλου

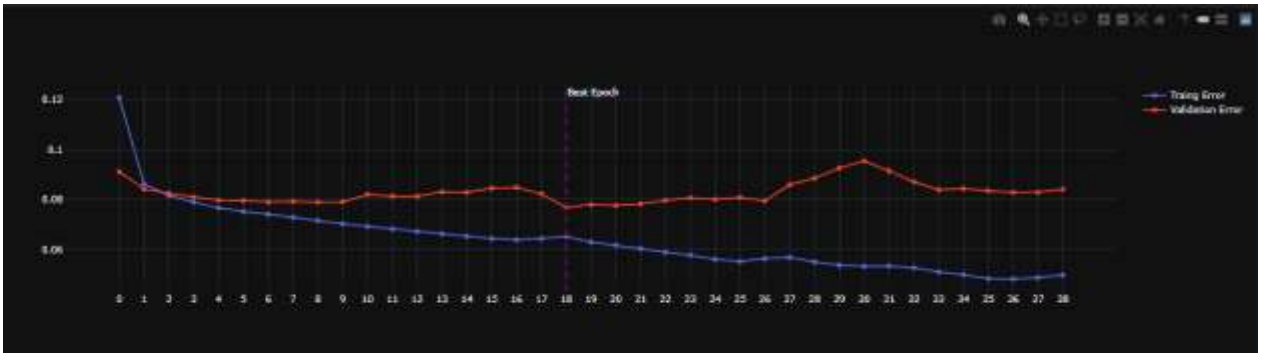
Μαζί με κάθε μοντέλο δίνονται και οι μετρικές της τρέχουσας εκπαίδευσης όπως ορίζονται στην ενότητα 2.3.

Metric	Train	Validation	Test
MAE	5.3731	6.1233	15.8061
RMSE	7.201	8.4513	20.4364
R2	0.9549	0.9328	0.6201

5.9 Πίνακας Μετρικών

5.1.4.1. Σελίδα μοντέλων βαθιάς μάθησης

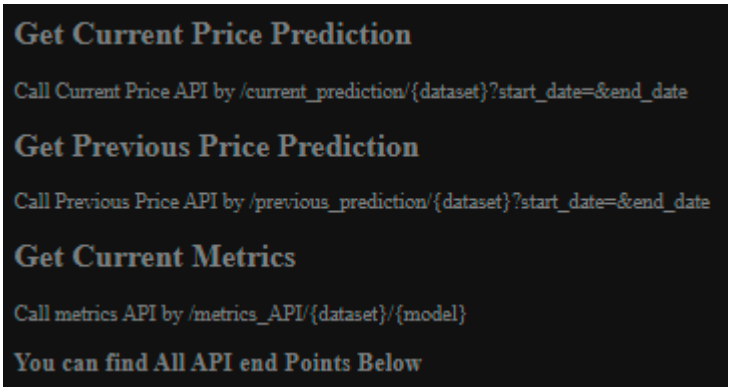
Μαζί με τις προηγούμενες πληροφορίες για τα επαναλαμβανόμενα νευρωνικά δίκτια δίνεται και το διάγραμμά λάθους εκπαίδευσης και επικύρωσης μαζί με τη καλύτερη εποχή πού αποθηκεύουμε το καλύτερο στιγμιότυπο του μοντέλου.

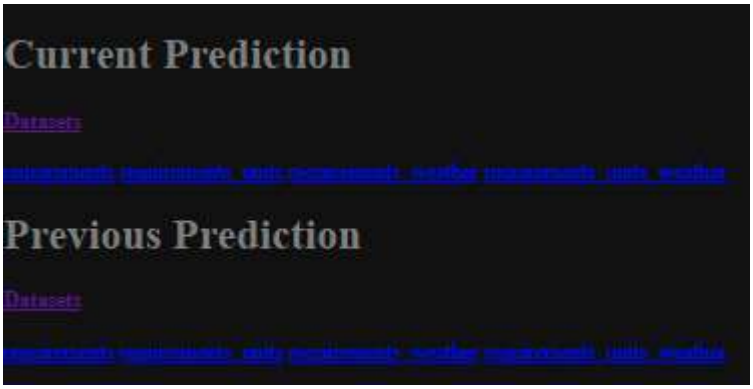


5.10 Lstm Σύνολο δεδομένων requirements_units λάθος εκπαίδευσης, λάθος επικύρωσης και καλύτερη εποχή εκπαίδευσης

5.1.5. Διεπαφή εφαρμογών προγραμματισμού(API)

Μαζί με την απεικόνιση δίνεται η δυνατότητα λήψης των δεδομένων μέσο API.





5.11 Σελίδα τεκμηρίωσης API

Υπάρχουν 4 καταληκτικά σημεία:

Το /Api κλήσεις σε docs, models και datasets οπού docs η ιστοσελίδα με την τεκμηρίωση και με models, datasets που επιστέφονται, σε μορφή Json, τα ονόματα τον μοντέλων και τον συνόλων δεδομένων αντίστοιχα.

{"0":"Linear","1":"KnnModel","2":"XgbModel","3":"Lstm","4":"Hybrid_Lstm"}

5.12 Κλήση models

{"0":"requirements","1":"requirements_units","2":"requirements_weather","3":"requirements_units_weather"}

5.13 Κλήση dataset

Το /current_prediction οπού διαλέγοντας το σύνολο δεδομένων επιστέφεται για κάθε μοντέλο η πρόβλεψη για την επόμενη μέρα σε μορφή Json και μπορεί να φορτωθεί εύκολα σε

pandas DataFrame π.χ. /current_prediction/requirements.

5.14 Κλήση /current_prediction/requirements_units_weather

Το /previous_prediction οπού διαλέγοντας το σύνολο δεδομένων και την ημερομηνία με παραμέτρους start_date και end_date π.χ.

5.15 Κλήση /previous_prediction/requirements?start_date=2021-09-20&end_date=2021-09-27

Το /metrics_api που με την επιλογή συνόλων δεδομένων και μοντέλου η all για την επιστροφή όλων των μετρικών ανά συνόλων δεδομένων π.χ.

5.16 Κλήση metrics_api Για ολα τα μοντέλα

5.17 Κλήση metrics_api για γραμμική παλινδρόμηση

5.2. Διακομιστής εκπαίδευσης

Καθημερινά στις 07:35 UTC στον διακομιστή εκτελείτε η ενημέρωση των υποσυνόλων και των συνόλων εκπαίδευσης, η εκπαίδευση των μοντέλων και η ενημέρωση της βάσης δεδομένων του διακομιστή απεικόνισης.

5.3. Εγκατάσταση

Η εγκατάσταση μπορεί ευκολά να γινεί σε ένα σύστημα με εγκατεστημένη διανομή Linux Ubuntu ανοίγοντας μια διεπαφή γραμμής εντολών και είτε κατεβάζοντας το project η εκτελώντας την εντολή `$git clone https://github.com/ThanosPourikis/Thesis.git`

Η εγκατάστασης του διακομιστή απεικόνισής γίνεται εκτελώντας το bash script `$bash build_app` και `$bash build_server` για τον διακομιστή εκπαίδευσης.

Κεφάλαιο 6 : Συμπεράσματα

6.1. Συμπεράσματα

Σε αυτή την πτυχιακή αναπτύχθηκε μια πλατφόρμα απεικόνισες δεδομένων με συμπληρωματικό API για τη λήψη των δεδομένων. Επίσης δημιουργήθηκαν 4 μοντέλα linear, KNN, XGBoost και LSTM για προβλέψει τιμών με ένα Ensemble με τον μέσο ορό των προβλέψεων. Πλέον με τον εκσυγχρονισμό τον μονάδων παραγωγής ηλεκτρικής ενέργειας οι διάστασης των δεδομένων αυξάνονται εκθετικά καθιστώντας τις παραδοσιακές μεθόδους ανάλυσης δεδομένων απαρχαιωμένες. Η εισαγωγή καινούργιων τεχνολογιών μηχανικής μάθησης είναι πλέον αναγκαία. Όπως προανέφερε τα προβλήματα χρηματιστηρίου έχουν άγνωστες μεταβλητές που είναι δύσκολο να βρεθούν και λόγω της αστάθειας του εύρους τιμών ηλεκτρικής ενέργειας, το σφάλμα της πρόβλεψης τιμής ηλεκτρικής ενέργειας είναι αναπόφευκτο σε κάποιο βαθμό. Για αυτόν τον λόγω προτείνετε το υβριδικό μοντέλο για να ομαλυνθούν η τιμές και να αποφευχθούν οι κακές προβλέψεις καθώς το κάθε μοντέλο λειτουργεί καλύτερα σε διαφορετικές περιόδους.

6.2. Περεταίρω ανάπτυξη

Η έρευνα μπορεί να επεκταθεί σε περιορισμό των διαστάσεων τον δεδομένων με επιλογή χαρακτηριστικών από τα ήδη υπάρχοντα δεδομένα ή την εύρεση καινούργιων μεταβλητών [28]. Επίσης η πρόληψη της ενδοημερήσιας αγοράς αλλά και η κατηγοριοποίηση της σε σχέση με την αγορά επόμενης μέρας για την εύρεση τιμών οπού η ενδοημερήσια αγορά θα έχει μεγαλύτερη τιμή δίνοντας την δυνατότητα δέσμευσης ποσότητας ενέργειας και πώληση της σε αυτήν.

Ένα ακόμα κομμάτι που δεν έχει ερευνηθεί αρκετά είναι στον τομέα είναι η ανάλυση συναισθημάτων (sentiment analysis) όπως έχει εφαρμοστεί για πρόβλεψη τιμής κλεισίματος μετοχών [29].

Κεφάλαιο 7 : ΠΑΡΑΡΤΗΜΑ

7.1. Απεικόνιση

Ανακατεύθυνση στις σελίδες.

```
#Redirections
@app.route('/')
def home():
    return redirect('Dataset/requirements')

@app.route('/Api')
def api_redict():
    return redirect('Api/docs')
```

Σελίδα Απεικόνισης δεδομένων

```
#Original data page
@app.route('/Dataset/<dataset>',methods = ['GET'])
def index(dataset):
    db = DB(datasets_dict[dataset])
    start_date,end_date = get_dates(request.args)
    df = db.get_data('*', dataset,f'"index" < "{end_date}"and "index" > "{start_date}"')

    if 'units' in dataset:
        heatmap = get_heatmap(df.iloc[:,7:-7 if 'cloudCover' in df.columns else -1])
        df = df.drop(axis = 1,columns = df.iloc[:,6:-7 if 'cloudCover' in df.columns else -1])
    else:
        heatmap = None
    return render_template('home.jinja',title = f'Train Data For {dataset} Dataset For The Pas
t 7 Days',
    df=df,get_json = get_json_for_line_fig,candlestick = get_candlesticks(df.SMP),
    dataset = dataset,heatmap = heatmap,start_date = start_date,end_date = end_date)
```

Σελίδα Συσχέτισης

```
#Correlation page
@app.route('/Correlation/<dataset>',methods = ['GET'])
def corrolations(dataset):
    db = DB(datasets_dict[dataset])
    start_date,end_date = get_dates(request.args)
    df = db.get_data('*', dataset,f'"index" <= "{end_date}"and "index" >= "{start_date}"')
    if 'units' in dataset:
        df = df.drop(axis = 1,columns =df.iloc[:,6:-7 if 'cloudCover' in df.columns else -1])
    df = df.set_index('SMP').dropna()
    return render_template('correlation.jinja',title = f'Correlation For {dataset} Dataset For
 The Past 7 Days',
    df=df,get_json = get_json_for_fig_scatter,dataset = dataset,start_date = start_date,end_da
te = end_date)
```

Σελίδα Μοντέλου

```
#Model page
@app.route('/<name>/<dataset>',methods = ['GET'])
def page_for_ml_model(dataset,name):

    start_date,end_date = get_dates(request.args)


    db = DB(datasets_dict[dataset])
    df = db.get_data('*',name,f'"index" <= "{end_date}" and "index" >= "{start_date}"')
    df['Previous Prediction'] = db.get_data(f'"index","{name}"','infernce',f'"index" <= "{end_
date}" and "index" >= "{start_date}"')
    metrics = db.get_metrics(name)

    if 'Lstm' in name:
        hist = db.get_data('*',f'hist_{name}')
        return render_template('lstm.jinja', title = f'Model: {name}, Dataset: {dataset},  Las
t 7days Prediction vs Actual Price And Inference',
                            chart_json = get_json_for_line_scatter(df,df.columns),
                            table = get_table(metrics),
                            hist_json = get_json_for_line_scatter(hist,hist.columns,metrics.il
oc[0]['best_epoch']),
                            dataset = dataset,start_date = start_date,end_date = end_date)
    else:
        return render_template('model.jinja', title = f'Model: {name}, Dataset: {dataset},  La
st 7days Prediction vs Actual Price And Inference',
                            chart_json = get_json_for_line_scatter(df,df.columns),
                            table = get_table(metrics),dataset = dataset,start_date = start_da
te,end_date = end_date)
```

API:

```
#Api endpoints
@app.route('/Api/<route>')
def api(route):

    if route =='datasets':
        return jsonify(pd.DataFrame(datasets)[0].to_dict())
    elif route == 'models':
        return jsonify(pd.DataFrame(models)[0].to_dict())
    elif route == 'docs':
        return render_template('api.jinja',datasets= datasets,models = models)

@app.route('/current_prediction/<dataset>')
def current_prediction(dataset):
    try:
        db = DB(datasets_dict[dataset])
        df = pd.DataFrame()
        for model in models:
            df[model] = db.get_data('"index","Inference"',model).dropna()
        df.index = df.index.astype(str)
        return jsonify(df.to_dict())
    except:
        return 'No Prediction Possible'

@app.route('/previous_prediction/<dataset>',methods = ['GET'])
def previous_prediction(dataset):
    try:
        db = DB(datasets_dict[dataset])
        df = pd.DataFrame()
        start_date,end_date = get_dates(request.args)

        for model in models:
            df[model] = db.get_data(f'"index","{model}"','infernce',f'"index" <= "{end_date}" and "inde
x" >= "{start_date}"')
        df.index = df.index.astype(str)
        return jsonify(df.dropna().to_dict())
    except:
        return 'No Prediction Possible'

@app.route('/metrics_api/<dataset>/<model>')
def metrics_api(dataset,model):
    db = DB(datasets_dict[dataset])
    try :
        if model == 'all':
            dict = {}
            for model in models:
                dict[model] = db.get_metrics(model).loc[:,['Train','Validation','Test']].to_dict()
            return jsonify(dict)
        else:
            return jsonify(db.get_metrics(model).to_dict())
```

Όπως προανέφερα δεν υπάρχει διεπαφή βάσης δεδομένων στο flask, για αυτό δημιουργήθηκε βάση δεδομένων τεχνολογίας SQLite και διεπαφή:

Αρχικοποίηση αντικείμενου

```
class DB:
    def __init__(self,database):
        self.engine = sq.create_engine(sqlite.format(database),)
        self.connection = self.engine.connect()
```

Αποθήκευση Δεδομένων:

```
    def save_df_to_db(self, dataframe, df_name):
        try:
            dataframe.index = [str(i) for i in dataframe.index]
        except:
            pass
        dataframe.to_sql(df_name, self.connection, if_exists='replace')

    def save_metrics(self,metrics,model):
        try:
            self.save_df_to_db(metrics,f'metrics_{model}')
        except:
            metrics = metrics.reset_index().drop_duplicates(subset ='index',ke
ep='first').set_index('index')
            self.save_df_to_db(metrics,f'metrics_{model}')
```

γίνετε η μετατροπή του κλειδιού από τύπο ημερομηνίας σε αλφαριθμητικό γιατί δεν υποστηρίζεται η αποθήκευση ζωνών ώρας.

Ανάκτηση δεδομένων:

Δεδομένα για απεικόνιση

```
    def get_data(self, columns, table, condition = None):
        if condition == None:
            query = f'SELECT {columns} FROM {table}'
        else:
            query = f'SELECT {columns} FROM {table} WHERE {condition}'

        try:
            df = pd.read_sql(query, self.connection,index_col='index')
        except :
            df = pd.read_sql(query, self.connection)

        try:
            df.index = [datetime.fromisoformat(i) for i in df.index ]
        finally:
            return df

    def get_metrics(self,model):
        metrics = self.get_data('*',f'metrics_{model}')
        # metrics.index = pd.to_datetime(metrics.index)
        return metrics
```

Δεδομένα για εκπαίδευση:

```
    def get_dataset(self, columns, tables):
        tables_list = tables.split('_')
        tables = tables_list.pop(0)
        for i in tables_list:
            tables += f' INNER JOIN {i} USING("index")'

        query = f'SELECT {columns} FROM {tables}'

        df = pd.read_sql(query, self.connection,index_col='index')
        df.index = [datetime.fromisoformat(i) for i in df.index ]
        df = df.join(self.get_data('*','smp'))

        return df
```

Κώδικας για απεικονίσεις:

Απλό διάγραμμα γραμμής:

```
def get_json_for_line_fig(df,x,y):
    fig = px.line(df,x=x,y=y,template=template)
    fig = fig.update_xaxes(rangeslider_visible=True)
    return json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

Διάγραμμα διασποράς με ενωμένα σημεία:

```
def get_json_for_line_scatter(df,y,line = None):
    fig = go.Figure()
    fig.update_layout(template=template)
    for i in y:
        fig.add_trace(go.Scatter(x=df.index, y=df[i],
                    mode='lines+markers',
                    name=df[i].name))
        if line != None:
            fig.add_vline(x=line,line_dash="dash", line_color="purple",annotat
ion_text = 'Best Epoch')
    return json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

Διάγραμμα διασποράς με γραμμή παλινδρόμησης:

```
def get_json_for_fig_scatter(df,y,x):
    fig = px.scatter(df,x=x,y=y,trendline="ols",trendline_color_override='red'
,template=template)
    fig = fig.update_xaxes(rangeslider_visible=True)
    return json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

Διάγραμμα κηροπηγίων:

```
def get_candlesticks(df):
    export = pd.DataFrame()
    # export['High'] = [df[i:i+24].max() for i in range(0,len(df),24)]
    for i in range(0,len(df),24):
        temp = pd.DataFrame()
        temp['Date'] = [df[i:i+24].index[0]]
        temp['high'] = df[i:i+24].max()
        temp['low'] = df[i:i+24].min()
        temp['open'] = df[i:i+24].iloc[0]
        temp['close'] = df[i:i+24].iloc[-1]
        temp = temp.set_index('Date')
        export = pd.concat([export,temp])
        candlestick = go.Candlestick(x=export.index,
                open=export['open'], high=export['high'],
                low=export['low'], close=export['close'])

    fig = go.Figure(data=[candlestick])
    fig.update_layout(template = template)
    return json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

Θερμικός πίνακας:

```
def get_heatmap(df):
    units_24 = df.copy()
    units_24 = pd.DataFrame([units_24.iloc[i] for i in range(0,units_24.shape[
0],24)])
    units_24.index = [units_24.index[i].date() for i in range(units_24.shape[0
])]
    units_24.head()
    units_24_trans = units_24.iloc[:,:-1].transpose()

    heatmap = go.Heatmap(
        z = units_24_trans.values,
        x= units_24_trans.columns,
        y = units_24_trans.index
    )
    fig = go.Figure(data=[heatmap])
    fig.update_layout(template = template)
    return json.dumps(fig, cls=plotly.utils.PlotlyJSONEncoder)
```

Μετονομασία στήλης και στρογγυλοποίηση για τον πίνακα μετρικών:

```
def get_table(df):
    df = df.round(4).reset_index()
    df.columns.values[0] = 'Metric'
    return df
```

Ημερομηνίες εβδομάδας ή ημερομηνίες που δόθηκαν:

```
def get_dates(form=''):

    if 'end_date' in form :
        end_date = form['end_date']
    else :
        end_date = str(pd.to_datetime(date.today() + timedelta(days=2)))[:10]

    if 'start_date' in form :
        start_date = form['start_date']
    else:
        start_date = str(localTz.localize(pd.to_datetime(date.today()) - timed
elta(weeks=1)))[:10]

    return start_date,end_date
```

7.2. Εκπαίδευση

Αρχικοποίηση:

```
logging.basicConfig(filename='log.log',level=logging.DEBUG)

datasets = ['requirements','requirements_units','requirements_weather','requir
ements_units_weather']
database_in = 'dataset'
models = [LinearRegression,KNeighborsRegressor,XGBRegressor,LSTM]
model_names = ['Linear','KnnModel','XgbModel']
params = ['linear_params','knn_params','xgb_params']
```

Υπερπαράμετροι από αρχείο yaml:

```
try:
    with open ('yaml.yaml', 'r') as file:
        params_list = yaml.safe_load(file)
except Exception as e:
    print(e)
```

Ενημέρωση δεδομένων, αρχικοποίηση βάσης δεδομένων, δημιουργία πίνακα νημάτων:

```
update()
db_in =DB(database_in)
threads = []
```

Συνάρτηση ενημέρωσης δεδομένων:

```
def update():
    db = DB('dataset')
    try:
        start_date = db.get_data('MAX("index")','requirements').values[0,0]
        requirements = get_isp_data(get_excel_data(folder_path=con-
fig.ISP1['folder_path'],filetype=config.ISP1['file-
type'],start_date = start_date))
        requirements = pd.concat([db.get_data('*','requirements'),require-
ments])
    except:
        start_date = '2020-11-01'
        requirements = get_isp_data(get_excel_data(folder_path=con-
fig.ISP1['folder_path'],filetype=config.ISP1['file-
type'],start_date = start_date))

    requirements.to_csv('datasets/requirements.csv')
    db.save_df_to_db(dataframe=requirements.copy(),df_name='requirements')

    try:
        start_date = db.get_data('MAX("index")','units').values[0,0]
        units = get_unit_data(get_excel_data(folder_path=con-
fig.UNITS['folder_path'],filetype=config.UNITS['file-
type'],start_date = start_date))
        if not units.empty:
            units = pd.concat([db.get_data('*','units')[units.col-
umns],units]).fillna(0)
        else:
            units = db.get_data('*','units')
    except:
        start_date = '2020-11-01'
        units = get_unit_data(get_excel_data(folder_path=con-
fig.UNITS['folder_path'],filetype=config.UNITS['file-
type'],start_date = start_date))

    units.to_csv('datasets/units.csv')
    db.save_df_to_db(dataframe=units.copy(),df_name='units')

    download_weather_data()
    weather =get_weather_mean()
    db.save_df_to_db(dataframe=weather.copy(),df_name='weather')

    try:
        start_date = db.get_data('MAX("index")','smp').values[0,0]
        Smp = get_SMP_data(start_date)
        Smp = pd.concat([db.get_data('*','smp'),Smp])
    except:
        Smp = get_SMP_data()
```

```
    Smp.to_csv('datasets/SMP.csv')
    db.save_df_to_db(dataframe=Smp.copy(),df_name='smp')

    db = DB('requirements')
    db.save_df_to_db(dataframe=requirements.join(Smp),df_name='requirements')

    db = DB('requirements_units')
    db.save_df_to_db(dataframe=requirements.join(units).join(Smp),df_name='req
uirements_units')

    db = DB('requirements_weather')
    db.save_df_to_db(dataframe=requirements.join(weather).join(Smp),df_name='r
equirements_weather')

    db = DB('requirements_units_weather')
    db.save_df_to_db(dataframe=requirements.join(units).join(weather).join(Smp
),df_name='requirements_units_weather')
```

Εκπαίδευση κάθε μοντέλου σε δικό του νήμα:

```
for dataset_name in datasets:
    save_infernce(dataset_name)
    dataset = db_in.get_dataset('*',dataset_name)
    dataset.insert(dataset.shape[1]-1,'lag_24',dataset['SMP'].shift(24))
    dataset = dataset[dataset['lag_24'].notna()]
    for model,model_name,param in zip(models,model_names,params):
        threading.Thread(target=train_model,args = (model,model_name,dataset,d
ataset_name,params_list[param],)).start()
    threads.append(threading.Thread(target=Lstm,args = (LSTM,'Lstm',dataset,da
taset_name,params_list['Lstm_params'],)))

for i in threads:
    i.start()

for i in threads:
    i.join()
```

Συνάρτηση train_model:

```
def train_model(model,model_name,df,dataset_name,params):
    prediction,metrics = get_model_results(df,params[dataset_name],model_name,
model)
    db_out = DB(dataset_name)
    db_out.save_df_to_db(prediction,model_name)
    db_out.save_metrics(metrics,model_name)
```

Συνάρτηση get_model_results:

Διαχωρισμός δεδομένων και αρχικοποίηση μοντέλου:

```
def get_model_results(data,params,name,model, validation_size=0.2):
    # features = data.loc[:,data.columns!='SMP']
    # data = data.set_index('Date')
    export = pd.DataFrame(data['SMP'],data.index)
    if data.isnull().values.any():
        inference = data[-24:]
        test = data[-(8*24):-24]
        data = data[:-(8*24)]
    else:
        test = data[-(7*24):]
        data = data[:-(7*24)]
    features = data.loc[:,data.columns!='SMP']
    labels = (data.loc[:,'SMP'])
    model = model(**params)

    x_train, x_validate, y_train, y_validate = train_test_split(features, labe
ls, random_state=96,
                                                                test_size=vali
dation_size, shuffle=True)
```

Εκπαίδευση μοντέλου και επιστροφή δεδομένων:

```
    logging.info(f'Fitting {name} Model')

    model.fit(x_train,y_train)

    train = model.predict(x_train)
    val = model.predict(x_validate)
    pred = model.predict(test.loc[:,test.columns != 'SMP'])
    metrics = get_metrics_df(
        y_train,train,
        y_validate,val,
        test.loc[:,'SMP'],pred)

    export['Training'] = pd.DataFrame(train,index=x_train.index)
    export['Validation'] = pd.DataFrame(val,index=x_validate.index)
    export['Testing'] = pd.DataFrame(pred,index=test.index)
    try:
        export['Inference'] = pd.DataFrame(model.predict(inference.loc[:,infer
ence.columns != 'SMP']),index=inference.index)
    finally:
        return export,metrics
```

Εκπαίδευση LSTM:

```
def Lstm(LSTM,name,df,dataset_name,params):
    lstm = LstmMVInput(utils.MAE,df,name = f'{name} {dataset_name}',LSTM = LST
M,**params)
    lstm.train()
    prediction,metrics,hist,best_epoch = lstm.get_results()
    db_out = DB(dataset_name)

    db_out.save_df_to_db(hist,f'hist_{name}')
    db_out.save_df_to_db(prediction,name)
    metrics['best_epoch'] = best_epoch
    db_out.save_metrics(metrics,name)
```

Αντικείμενο εκπαίδευσης LSTM:

Αρχικοποίηση αντικειμένου:

```
class LstmMVInput:
    def __init__(self, loss_function, data,name,LSTM,
                 learning_rate,
                 validation_size,
                 sequence_length,
                 batch_size,
                 hidden_size,
                 num_layers,
                 output_dim,
                 num_epochs,
                 model = None):
        # data = data.set_index('Date')
        self.export = pd.DataFrame(data['SMP'],index=data.index)
        if data.isnull().values.any():
            self.inference = data[-24:]
            self.test = data[-(8*24):-24]
            data = data[:-(8*24)]
        else:
            self.test = data[-(7*24):]
            data = data[:-(7*24)]

        self.LSTM = LSTM
        self.features = data.loc[:,data.columns!='SMP']
        self.labels = data['SMP']
        self.input_size = self.features.columns.shape[0]
        self.validation_size = validation_size
        self.loss_function = loss_function
        self.sequence_length = sequence_length
        self.batch_size = batch_size
        self.hidden_size = hidden_size
        self.num_layers = num_layers
        self.output_dim = output_dim
        self.num_epochs = num_epochs
        self.learning_rate = learning_rate
        self.name = name
        self.model = model
```

Διαχωρισμός δεδομένων :

```
self.x_train, self.x_validate, self.y_train, self.y_validate = train_test_spli
t(self.features, self.labels,
        random_state=96,test_size=self.validation_size,shuffle=True)
```

Δημιουργία παραθύρων :

```
      x_train,y_train = sliding_windows(self.x_train,self.y_train,sequence_len
=self.sequence_length ,window_step=1)
       x_validate,y_validate = sliding_windows(self.x_validate,self.y_validat
e,sequence_len=self.sequence_length ,window_step=1)
```

Ομαλοποίηση σε εύρος -1,1:

```
        feature_t_s = MinMaxScaler(feature_range=(-1, 1))
        feature_v_s = MinMaxScaler(feature_range=(-1, 1))
        labels_t_s = MinMaxScaler(feature_range=(-1, 1))
        labels_v_s = MinMaxScaler(feature_range=(-1, 1))

        x_train = feature_t_s.fit_transform(x_train.reshape(-
1, x_train.shape[-1])).reshape(x_train.shape)
        x_validate = feature_v_s.fit_transform(x_validate.reshape(-
1, x_validate.shape[-1])).reshape(x_validate.shape)

        y_train = labels_t_s.fit_transform(y_train.squeeze())
        y_validate = labels_v_s.fit_transform(y_validate.squeeze())
```

Αρχικοποιήσεις μοντέλου συνάρτησης λάθους και αλγορίθμου βελτιστοποίησης πινάκων και ώρας αρχής:

```

      model = self.LSTM(input_size=self.input_size, hidden_size=self.hidden_si
ze,output_dim = self.output_dim,
                        num_layers=self.num_layers,batch_first=True)
        self.criterion = L1Loss()
        optimiser = Adam(model.parameters(), self.learning_rate)

        self.error_train = np.empty(0)
        self.error_val = np.empty(0)
        start_time = time.time()
```

Βρόγχος εκπαίδευσης:

```
train_data_loader = DataLoader(RequirementsSample(x_train,y_train),self.batch_
size, shuffle=False,drop_last=True)
        val_data_loader = DataLoader(RequirementsSample(x_validate,y_validate)
,self.batch_size, shuffle=False,drop_last=True)
        while(True): # Early Stopping If error hasnt decreased in N epochs STO
P
        # for i in range(self.num_epochs):
            model.train()
            err = []
            for j, k in train_data_loader:
                y_train_pred = model(j.float())
                loss = self.criterion(y_train_pred.squeeze(), k.squeeze().floa
t())
                err.append(loss.detach().item())
                optimiser.zero_grad()
                loss.backward()
                optimiser.step()
            self.error_train = np.append(self.error_train, (sum(err) / len(err
)))

            model.eval()
            err = []
            for j, k in val_data_loader:
                y_val_pred = model(j.float())
                loss = self.criterion(y_val_pred.squeeze(),k.squeeze().float()
)
                err.append(loss.detach().item())
            self.error_val = np.append(self.error_val,(sum(err)/len(err)))

            logging.info(f"{self.name}\t Time {time.time() - start_time:.4f}\t
 Epoch {self.error_val.shape[0]} {self.loss_function} \t Training\t{self.error
_train[-1]:.4f}\t Validation\t{self.error_val[-1]:.4f}")

            if self.error_val[-1] <= self.error_val.min() :
                self.model = copy.deepcopy(model)

            if (self.error_val.shape[0] - self.error_val.argmin()) > self.num_
epochs:
                break
```

Αποτελέσματα λάθους:

```
        self.best_epoch = self.error_val.argmin()
        logging.info(f'{self.name} Training Completed Best_epoch : {self.best_
epoch} Training Time {time.time() - start_time:.4f}')

        self.hist = pd.DataFrame()
        self.hist['Traing Error'] = self.error_train.tolist()
        self.hist['Validation Error'] = self.error_val.tolist()
```

Επιστροφή Αποτελεσμάτων:

```
    def get_results(self,test = None):
        scaler_f = MinMaxScaler((-1,1))
        scaler_l = MinMaxScaler((-1,1))
        scaler_l.fit_transform(np.array(self.y_train).reshape(-1,1))
        self.model.eval()

        x_train = scaler_f.fit_transform(np.array(self.x_train))
        x_train = Tensor(x_train.reshape(1,-1,self.x_train.shape[1]))
        train = self.model(x_train)
        train = train.detach().numpy().reshape(-1,1)
        train = scaler_l.inverse_transform(train)
        self.export = self.export.join(pd.DataFrame(train,index=self.y_train.i
ndex,columns=['Training']))

        x_validate = scaler_f.transform(np.array(self.x_validate))
        x_validate = Tensor(x_validate.reshape(1,-1,self.x_validate.shape[1]))
        val = self.model(x_validate)
        val = scaler_l.inverse_transform(val.detach().numpy().reshape(-1,1))
        self.export = self.export.join(pd.DataFrame(val,index=self.y_validate.
index,columns=['Validation']))

        x_test,y_test = self.test.loc[:,self.test.columns != 'SMP'],self.test.
loc[:,'SMP']
        x_test = scaler_f.transform(np.array(x_test))
        x_test = Tensor(x_test.reshape(1,-1,x_test.shape[1]))
        test = self.model(x_test)
        test = scaler_l.inverse_transform(test.detach().numpy().reshape(-1,1))
        self.export = self.export.join(pd.DataFrame(test,index=y_test.index,co
lumns=['Testing']))

        metrics = get_metrics_df(self.y_train,train,
                self.y_validate,val,
                y_test,test)

        try:
            inference = self.inference.loc[:,self.inference.columns != 'SMP']
            inference = scaler_f.transform(np.array(inference))
            inference = Tensor(inference.reshape(1,-1,inference.shape[1]))
            inference = self.model(inference)
            inference = scaler_l.inverse_transform(inference.detach().numpy().
reshape(-1,1))
            self.export['Inference'] =  pd.DataFrame(inference,index=self.infe
rence.index)
        finally:
            return self.export,metrics,self.hist,self.best_epoch
```

Παράθυρα:

```
def sliding_windows(features,labels,sequence_len = 24,window_step = 1):
    xX = []
    yY = []
    for i in range(0,features.shape[0]-sequence_len+1,window_step):

        xX.append(features[i: i + sequence_len])
        yY.append(labels[i: i + sequence_len])

    xX = np.array(xX)
    yY = np.array(yY)

    return xX,yY
```

Αντικείμενο Data Loader pytorch:

```
class RequirementsSample(torch.utils.data.Dataset):
    def __init__(self,features,labels):
        self.features = features
        self.labels = labels

    def __len__(self):
        return self.features.shape[0]

    def __getitem__(self,index):
        return np.array(self.features[index]), np.array(self.labels[index])
```

Αρχιτεκτονική LSTM:

```
### TODO Add functionality for bidirectionality
class LSTM(nn.Module):
    def __init__(self, input_size, hidden_size, num_layers, output_dim,drop=0.
1,batch_first=True):
        super(LSTM, self).__init__()
        # self.norm = nn.BatchNorm1d(24)
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first)
        self.drop = Dropout(drop)
        self.linear = nn.Linear(hidden_size,hidden_size)
        self.act = nn.Tanh()
        self.linear2 = nn.Linear(hidden_size,output_dim)
        self.act2 = nn.Tanh()

    def forward(self, x):
        # self.norm(x)
        output,_ = self.lstm(x)
        output = self.drop(output)
        output = self.linear(output)
        output = self.act(output)
        output = self.linear2(output)
        output = self.act2(output)

        return output
```

υπολογισμός μέσου ορού για την πρόβλεψη:

```
for dataset in datasets:
    db=DB(dataset)
    data = {}
    for i in model_names:
        data[i] = db.get_data('*',i)

    smp = data['Linear']['SMP']

    val = pd.concat([data[i]['Validation'].dropna() for i in data],axis=1).mea
n(axis=1)

    test = pd.concat([data[i]['Testing'].dropna() for i in data],axis=1).mean(
axis=1)

    inf =  pd.concat([data[i]['Inference'].dropna() for i in data],axis=1).mea
n(axis=1)

    data.pop('KnnModel',None)
    train = pd.concat([data[i]['Training'].dropna() for i in data],axis=1).mea
n(axis=1)

    prediction = pd.concat([smp,train,val,test,inf],axis=1)
    prediction.columns=['SMP','Training','Validation','Testing','Inference']

    metrics = get_metrics_df(prediction.loc[:,['SMP','Training']].dropna()['SM
P'],prediction['Training'].dropna(),
    prediction.loc[:,['SMP','Validation']].dropna()['SMP'],prediction['Validat
ion'].dropna(),
    prediction.loc[:,['SMP','Testing']].dropna()['SMP'],prediction['Testing'].
dropna())

    prediction,metrics
    db_out = DB(dataset)

    db_out.save_df_to_db(prediction,'Hybrid')
    db_out.save_metrics(metrics,'Hybrid')
```

Κεφάλαιο 8: Εικόνες

Κεφάλαιο 9: Βιβλιογραφία

[1] A. L. Samuel, «Some Studies in Machine Learning Using the Game of Checkers,» *IBM Journal of Research and Development,* τόμ. 3, p. 210–229, July 1959.

[2] M. P. Allen, «Understanding Regression Analysis,» Springer US, 2007, p. 91–95.

[3] J. Lago, G. Marcjasz, B. D. Schutter και R. Weron, «Forecasting day-ahead electricity prices: A review of state-of-the-art algorithms, best practices and an open-access benchmark,» *Applied Energy,* τόμ. 293, p. 116983, July 2021.

[4] G. Díaz, J. Coto και J. Gómez-Aleixandre, «Prediction and explanation of the formation of the Spanish day-ahead electricity price through machine learning regression,» *Applied Energy,* τόμ. 239, pp. 610-625, 2019.

[5] Z. Chang, Y. Zhang και W. Chen, «Effective Adam-Optimized LSTM Neural Network for Electricity Price Forecasting,» σε *2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS)*, 2018.

[6] ΑΔΜΗΕ, ΑΔΜΗΕ, [Ηλεκτρονικό]. Available: https://www.admie.gr/.

[7] R. F. Engle, C. W. J. Granger, J. Rice και A. Weiss, «Semiparametric Estimates of the Relation between Weather and Electricity Sales,» *Journal of the American Statistical Association,* τόμ. 81, p. 310–320, June 1986.

[8] C.-L. Hor, S. J. Watson και S. Majithia, «Analyzing the Impact of Weather Variables on Monthly Electricity Demand,» *IEEE Transactions on Power Systems,* τόμ. 20, p. 2078–2085, November 2005.

[9] J. D. Jobson, «Multiple Linear Regression,» σε *Springer Texts in Statistics*, Springer New York, 1991, p. 219–398.

[10] T. Ban, R. Zhang, S. Pang, A. Sarrafzadeh και D. Inoue, «Referential kNN Regression for Financial Time Series Forecasting,» σε *Neural Information Processing*, Springer Berlin Heidelberg, 2013, p. 601–608.

[11] F. H. Al-Qahtani και S. F. Crone, «Multivariate k-nearest neighbour regression for time series data 2014 A novel algorithm for forecasting UK electricity demand,» σε *The 2013 International Joint Conference on Neural Networks (IJCNN)*, 2013.

[12] J. H. Friedman, «Greedy function approximation: A gradient boosting machine.,» *The Annals of Statistics,* τόμ. 29, October 2001.

[13] T. Chen και C. Guestrin, «XGBoost,» σε *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016.

[14] S. Hochreiter και J. Schmidhuber, «Long Short-term Memory,» *Neural computation,* τόμ. 9, pp. 1735-80, December 1997.

[15] I. E. Livieris, E. Pintelas και P. Pintelas, «A CNN–LSTM model for gold price time-series forecasting,» *Neural Computing and Applications,* τόμ. 32, p. 17351–17360, April 2020.

[16] J. Cao, Z. Li και J. Li, «Financial time series forecasting model based on CEEMDAN and LSTM,» *Physica A: Statistical Mechanics and its Applications,* τόμ. 519, p. 127–139, April 2019.

[17] S. Siami-Namini, N. Tavakoli και A. S. Namin, «A Comparison of ARIMA and LSTM in Forecasting Time Series,» σε *2018 17th IEEE International Conference on Machine Learning and Applications ICMLA*, 2018.

[18] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai και S. Chintala, «PyTorch: An Imperative Style, High-Performance Deep Learning Library,» σε *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox και R. Garnett, Επιμ., Curran Associates, Inc., 2019, p. 8024–8035.

[19] D. P. Kingma και J. Ba, «Adam: A Method for Stochastic Optimization,» December 2014.

[20] S. Ruder, «An overview of gradient descent optimization algorithms,» September 2016.

[21] D. M. Allen, «Mean Square Error of Prediction as a Criterion for Selecting Variables,» *Technometrics,* τόμ. 13, p. 469–475, August 1971.

[22] S. Wright, "Correlation and Causation," pp. p. 557-585.-USDA, 1921.

[23] D. Chicco, M. J. Warrens και G. Jurman, «The coefficient of determination R-squared is more informative than SMAPE, MAE, MAPE, MSE and RMSE in regression analysis evaluation,» *PeerJ Computer Science,* τόμ. 7, p. e623, July 2021.

[24] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot και E. Duchesnay, «Scikit-learn: Machine Learning in Python,» *Journal of Machine Learning Research,* τόμ. 12, p. 2825–2830, 2011.

[25] M. Grinberg, Flask web development: developing web applications with python, " O'Reilly Media, Inc.", 2018.

[26] P. T. Inc., «Collaborative data science,» 2015. [Ηλεκτρονικό]. Available: https://plot.ly.

[27] S. Raschka, Python Machine Learning, Packt Publishing, 2015.

[28] F. Martínez, M. P. Frías, M. D. Pérez-Godoy και A. J. Rivera, «Dealing with seasonality by narrowing the training set in time series forecasting with k NN,» *Expert Systems with Applications,* τόμ. 103, p. 38–48, August 2018.

[29] Z. Jin, Y. Yang και Y. Liu, «Stock closing price prediction based on sentiment analysis and LSTM,» *Neural Computing and Applications,* τόμ. 32, p. 9713–9729, September 2019.

[30] T. Zhang, «Solving Large Scale Linear Prediction Problems Using Stochastic Gradient Descent Algorithms,» σε *Proceedings of the Twenty-First International Conference on Machine Learning*, New York, NY, USA, 2004.

[31] R. Weron, «Electricity price forecasting: A review of the state-of-the-art with a look into the future,» *International Journal of Forecasting,* τόμ. 30, p. 1030–1081, October 2014.

[32] A. D. Papalexopoulos και T. C. Hesterberg, «A regression-based approach to short-term system load forecasting,» *IEEE Transactions on Power Systems,* τόμ. 5, p. 1535–1547, 1990.

[33] J. Nowotarski και R. Weron, «Recent advances in electricity price forecasting: A review of probabilistic forecasting,» *Renewable and Sustainable Energy Reviews,* τόμ. 81, p. 1548–1568, January 2018.