ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ & ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόβλεψη μέσης ωριαίας κατανάλωσης ηλεκτρικής ισχύος με χρήση μεθόδων μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αθανάσιου-Μάριου Δ. Γεωργακόπουλου

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Αθήνα, Ιούλιος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΕΠΙΚΟΙΝΩΝΙΩΝ, ΗΛΕΚΤΡΟΝΙΚΗΣ
& ΣΥΣΤΗΜΑΤΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πρόβλεψη μέσης ωριαίας κατανάλωσης ηλεκτρικής ισχύος με χρήση μεθόδων μηχανικής μάθησης

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

του

Αθανάσιου-Μάριου Δ.
Γεωργακόπουλου

Επιβλέπουσα: Θεοδώρα Βαρβαρίγου
Καθηγήτρια ΕΜΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 22η Ιουλίου 2022.

.....	.....	.....
Θεοδώρα Βαρβαρίγου Καθηγήτρια ΕΜΠ	Εμμανουήλ Βαρβαρίγος Καθηγητής ΕΜΠ	Συμεών Παπαβασιλείου Καθηγητής ΕΜΠ

Αθήνα, Ιούλιος 2022.

.....

Αθανάσιος-Μάριος Γεωργακόπουλος
Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Ο σκοπός της παρούσας εργασίας είναι να προσαρμόσει διάφορους αλγορίθμους μηχανικής μάθησης και αρχιτεκτονικές βαθιάς μάθησης στο πρόβλημα της πρόβλεψης χρονοσειρών. Με άλλα λόγια μετατρέπουμε το πρόβλημα της παλινδρόμησης στην πρόβλεψη χρονοσειρών. Για να το πετύχουμε αυτό εξετάζουμε διάφορες τεχνικές με τις οποίες τα μοντέλα εκπαιδεύονται και παράγουν τις ζητούμενες προβλέψεις. Τα δεδομένα μας προέρχονται από ένα σύνολο μετρήσεων της κατανάλωσης ισχύος σε μία τριφασική γραμμή, οι οποίες μετά από κατάλληλη επεξεργασία μας δίνουν τις τελικές χρονοσειρές.
Αρχικά, παρουσιάστηκαν οι βασικές έννοιες που αφορούν τις χρονοσειρές. Αναφερθήκαμε στην ανάλυση χρονοσειρών μέσω διαφόρων δεικτών και στην διαδικασία πρόβλεψης με στατιστικές μεθόδους. Οι στατιστικές μέθοδοι που αναλύθηκαν είναι η απλοϊκή, οι μέθοδοι εκθετικής εξομάλυνσης, τα μοντέλα ARIMA, η αποσύνθεση και ο απλός μέσος όρος. Για τις τρεις πρώτες παρουσιάσαμε και τις εποχιακές εκδοχές τους. Επίσης, κάναμε μία σύντομη αναφορά στην μέθοδο Θ, στην γραμμική παλινδρόμηση και στον αλγόριθμο Facebook Prophet. Είναι λογικό να θέλουμε να αξιολογήσουμε τις προβλέψεις μας. Για να γίνει αυτό είναι απαραίτητο να έχουμε τα πραγματικά δεδομένα ώστε να τα αντιπαραβάλλουμε με την πρόβλεψή μας. Αν τα δεδομένα αυτά είναι μελλοντικές τιμές πρέπει να περιμένουμε μέχρι να γίνουν διαθέσιμα. Στην εργασία μας προβλέπουμε τις τελευταίες τιμές των χρονοσειρών, τις οποίες θεωρούμε άγνωστες. Στην παρούσα εργασία θα δουλέψουμε με το MAPE, που ανήκει στην κατηγορία των ποσοστιαίων σφαλμάτων.
Οι μέθοδοι μηχανικής μάθησης που χρησιμοποιήσαμε για να κάνουμε προβλέψεις είναι η γραμμική παλινδρόμηση, οι μηχανές διανυσμάτων υποστήριξης, οι κ-κοντινότεροι γείτονες, το δένδρο απόφασης, το τυχαίο δάσος και η ενίσχυση κλίσης. Από το κομμάτι της βαθιάς μάθησης χρησιμοποιούμε τα τεχνητά νευρωνικά δίκτυα και τον σύγχρονο αλγόριθμο N-Beats. Σύντομη αναφορά γίνεται στα συνελικτικά νευρωνικά δίκτυα και στα αναδρομικά νευρωνικά δίκτυα. Η εκπαίδευση των μοντέλων σε κάθε περίπτωση γίνεται μετατρέποντας την χρονοσειρά σε δείγματα, όπου ως εισόδους θα έχουν παρελθοντικές τιμές.
Για την διευκόλυνση της διαδικασίας δημιουργίας προβλέψεων δημιουργήσαμε μία εφαρμογή. Στην εφαρμογή αυτή ο χρήστης μπορεί να κάνει πρόβλεψη με όσους από τους έξι αλγορίθμους επιθυμεί ή να δημιουργήσει ένα νευρωνικό δίκτυο για τον σκοπό αυτό. Πέραν αυτού, η εφαρμογή δίνει την δυνατότητα στον χρήστη να δει τα δεδομένα καθώς και τις προβλέψεις που έκανε στο παρελθόν. Τέλος, πραγματοποιήσαμε διάφορα πειράματα πρόβλεψης και συγκρίναμε τους αλγορίθμους και τις τεχνικές βάσει των αποτελεσμάτων τους.

Λέξεις Κλειδιά

Χρονοσειρές, Πρόβλεψη, Μηχανική Μάθηση, Βαθιά Μάθηση, Δένδρο Απόφασης, Ενίσχυση Κλίσης, Μηχανές Διανυσμάτων Υποστήριξης, Κ-Κοντινότεροι Γείτονες, Τυχαίο Δάσος, Γραμμική Παλινδρόμηση, Εφαρμογή μίας σελίδας, Dask

Abstract

The cause of this work is to adapt various machine learning algorithms and deep learning architectures to the problem of time-series forecasting. In other words, we convert the regression problem into time-series forecasting. In order to do this we examine various techniques, which are used to train models and produce the asked predictions. The data we used comes from a set of measurements of power consumption from a three-phase electrical line. With proper processing we make the final time-series.
Initially, basic concepts of forecasting have been presented. We referred to time-series analysis via various indicators and to forecasting process via statistical methods. The statistical methods that we referred to, are Naive, exponential smoothing, ARIMA models, decomposition and simple average. For the first three methods, we included the seasonal versions of them. Furthermore, we made a short reference to theta method, to linear regression, and to the Facebook Prophet algorithm. It's logical that we want to evaluate the predictions. In order to do this, it's necessary to have the actual data to contrast them with the predictions. If these data are future values we need to await them. In this work we forecast the last values of the time-series, which we consider unknown. In this work we will use MAPE, which belongs to absolute percentage errors. The machine learning methods that we used to make predictions are linear regression, support vector machines, k-nearest neighbors, decision tree, random forest and gradient boosting. From the part of deep learning we used custom neural networks and the modern algorithm N-Beats. We made a short reference to convolutional neural networks and recurrent neural networks. Training of models in all cases is made by using converted time-series to samples, which have as input some past values.
In order to facilitate the process of producing predictions, we created a web application. In this application, user can make a forecast using as many algorithms as he wants from the six presented algorithms or can create a neural network for this purpose. Besides this, application enables user to see the data and the past predictions. Finally, we run several experimental forecasts and compared algorithms and techniques based on their results.

Keywords

Time-Series, Forecast, Machine Learning, Deep Learning, Decision Tree, Gradient Boosting, Support Vector Machines, K-Nearest Neighbors, Random Forest, Linear Regression, Single-Page Application, Dask

Ευχαριστίες

Καταρχάς θα ήθελα να ευχαριστήσω την κα. Βαρβαρίγου για την ευκαιρία που μου έδωσε να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα και να έρθω λόγω αυτού σε επαφή με μία σειρά από σύγχρονες τεχνολογίες και εργαλεία. Επίσης, θα ήθελα να ευχαριστήσω τους υποψήφιους διδάκτορες Αχιλλέα Μαρινάκη και Αναστάσιο Νικολακόπουλο για την καθοδήγηση και την καλή συνεργασία κατά την εκπόνηση της παρούσας εργασίας. Τέλος, ευχαριστώ τους γονείς μου για την στήριξη που μου παρείχαν όλα αυτά τα χρόνια.

Αθανάσιος-Μάριος Γεωργακόπουλος
Αθήνα, Ιούλιος 2022

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1

Εισαγωγή

Η εποχή την οποία διανύουμε χαρακτηρίζεται από την ύπαρξη μεγάλου όγκου δεδομένων. Έξυπνοι μετρητές και αισθητήρες έχουν την ικανότητα να πραγματοποιούν διαφόρων ειδών μετρήσεις. Η αξιοποίηση των δεδομένων του παρόντος και του παρελθόντος μπορεί να μας δώσει μία εικόνα για το μέλλον. Με βάση τα δεδομένα αυτά, είναι σύνηθες να επιχειρούμε να κάνουμε προβλέψεις για το μέλλον και βάσει αυτών των προβλέψεων να προσαρμόζουμε τις δράσεις μας. Ο τομέας των προβλέψεων έχει εφαρμογή σε πολλούς τομείς της ανθρώπινης ζωής. Στις επιχειρήσεις γίνονται προβλέψεις για το ύψος των πωλήσεων των προϊόντων και αναλόγως σχεδιάζονται πολιτικές προώθησης, διαφημιστικές δράσεις και γίνεται προμήθεια υλικών. Διάφοροι οργανισμοί πραγματοποιούν προβλέψεις για διάφορα θέματα, όπως για παράδειγμα οι προβλέψεις του Ο.Η.Ε. για τον παγκόσμιο πληθυσμό. Μία άλλη γνωστή εφαρμογή των προβλέψεων αφορά το χρηματηστήριο, όπου επιχειρείται να προβλεφθεί η μελλοντική τιμή των μετοχών. Οι προβλέψεις χρησιμοποιούνται επίσης μεταξύ άλλων στον επαγγελματικό αθλητισμό, στην πολιτική, στην μετεορολογία και στις τηλεπικοινωνίες για τον σχεδιασμό αποδοτικών δικτύων. Οι προβλέψεις μπορούν να αφορούν και για την κατανάλωση ηλεκτρικής ενέργειας.

Η βασική έννοια γύρω από την οποία κινούμαστε είναι η χρονοσειρά, η οποία αποτελείται από μία χρονολογική ένδειξη και τις τιμές του μεγέθους για την κάθε χρονική περίοδο. Για να προβλέψουμε τις μελλοντικές τιμές της χρονοσειράς, σύμφωνα με τις καθιερωμένες μεθόδους, βασιζόμαστε στις παρελθοντικές παρατηρήσεις. Υποθέτουμε, δηλαδή, πως το παρελθόν αποτελεί μια καλή ένδειξη για το μέλλον. Στις μεθόδους αυτές, που λέγονται μέθοδοι χρονοσειρών, εντάσσονται οι ιδιαιτέρως διαδεδομένες στατιστικές προβλέψεις. Σε αυτήν την περίπτωση ουσιαστικά εφαρμόζουμε μαθηματικές σχέσεις επί των παρελθοντικών παρατηρήσεων και παράγουμε ως έξοδο την πρόβλεψή μας.

Για την πρόβλεψη των χρονοσειρών μπορούμε να χρησιμοποιήσουμε και μεθόδους μηχανικής μάθησης. Με τις μεθόδους αυτές ξεφεύγουμε από την λογική του κλασσικού προγραμματισμού. Στην περίπτωση του κλασσικού προγραμματισμού, έχουμε ένα σύνολο σαφώς διατυπωμένων κανόνων, υπό την μορφή ενός προγράμματος, βάσει των οποίων μία είσοδος μας δίνει κάποιο αποτέλεσμα, την έξοδο. Ωστόσο, για μία σειρά προβλημάτων δεν είναι δυνατόν να διατυπώσουμε τους κανόνες που απαιτούνται για την γενική επίλυσή τους. Σε αυτές τις περιπτώσεις κινούμαστε με μηχανική μάθηση, όπου δίνουμε στο σύστημά μας ένα σύνολο εισόδων, μαζί με τις εξόδους που επιθυμούμε να παράγει για κάθε μία απ'αυτές και αφήνουμε το σύστημα να μάθει μόνο του τους κανόνες βάσει των οποίων αντιστοιχίζονται οι είσοδοι στις εξόδους. Αφότου το σύστημα μάθει τους κανόνες, δηλαδή όταν ολοκληρωθεί η εκπαίδευσή του, δίνοντας του καινούργιες άγνωστες είσοδους μπορεί παράξει εξόδους βάσει των κανόνων που έμαθε. Η μηχανική μάθηση, επομένως, εισάγει ουσιαστικά ένα νέο προγραμματιστικό υπόδειγμα. Τα τελευταία χρόνια έχει αυξήθει σημαντικά η δημοφιλία της μηχανικής μάθησης, λόγω της υψηλής διαθεσιμότητας σε εκτενή σύνολα δεδομένων και της βελτίωσης στο κομμάτι του υλικού των υπολογιστών. Εφαρμογές της μηχανικής μάθησης συναντώνται μεταξύ άλλων στην αναγνώριση ήχου και εικόνας, στην ιατρική για τις διαγνώσεις, στα συστήματα συστάσεων για προϊόντα, στις τράπεζες για την αναγνώριση απάτης και στις προβλέψεις.

Σκοπός της παρούσας εργασίας είναι η μελέτη των τεχνικών πρόβλεψης είτε είναι καθαρά στατιστικές είτε κάνουν χρήση μεθόδων μηχανικής μάθησης και η εφαρμογή τους στον τομέα της κατανάλωσης ισχύος. Σημειώνεται πως τα δεδομένα μας προέρχονται από

τις μετρήσεις κατανάλωσης ενεργού ισχύος μίας τριφασικής γραμμής, στις οποίες έχουμε εφαρμόσει κάποιες αλλαγές για να λάβουμε τις τρεις χρονοσειρές που θα μελετήσουμε και θα προβλέψουμε.
Η παρούσα εργασία αποτελείται από οκτώ κεφάλαια. Το πρώτο κεφάλαιο είναι η εισαγωγή. Στο δεύτερο κεφάλαιο ασχολούμαστε με την παρουσίαση των βασικών ζητημάτων γύρω από την ανάλυση των χρονοσειρών. Ειδικότερα, δίνουμε τον ορισμό τους, αναλύουμε τα βασικά χαρακτηριστικά τους καθώς και μία σειρά από δείκτες που μπορούν να χρησιμοποιηθούν για να βγάλουμε χρήσιμα συμπεράσματα επί αυτών. Ακολούθως, αναλύουμε το ζήτημα της πρόβλεψης μέσω στατιστικών μεθόδων, τους οποίους περιγράφουμε και μέσω εξισώσεων, Το κεφάλαιο ολοκληρώνεται με την αναφορά στα μέτρα απόδοσης των προβλέψεων. Στο επόμενο κεφάλαιο, ασχοληθήκαμε με το ζήτημα της αποθήκευσης των χρονοσειρών. Εκεί παρουσιάσαμε σε θεωρητικό επίπεδο μία σειρά εργαλείων που αξιοποιήσαμε για την δημιουργία και αποθήκευση των δεδομένων μας. Στο κεφάλαιο 4, αφού κάνουμε μια εκτενή εισαγωγή στις έννοιες της τεχνητής νοημοσύνης, της μηχανικής μάθησης και της βαθιάς μάθησης, παρουσιάσαμε έξι αλγορίθμους μηχανικής μάθησης. Στη συνέχεια, εξετάζουμε τον τρόπο με τον οποίο από την αρχική μας χρονοσειρά δημιουργούμε τα δείγματα με τα οποία εκπαιδεύουμε τους αλγορίθμους. Κατόπιν, παρουσιάζουμε τις βασικές στρατηγικές που ακολουθούμε για να παράξουμε τις ζητούμενες προβλέψεις. Το κεφάλαιο ολοκληρώνεται με το πρόβλημα της βελτιστοποίησης των υπερπαραμέτρων προσαρμοσμένο στην περίπτωση που έχουμε δεδομένα με χρονολογική εξάρτηση. Το πέμπτο κεφάλαιο περιλαμβάνει θέματα που σχετίζονται με τα νευρωνικά δίκτυα, ενώ παρουσιάζουμε και έναν σύγχρονο αλγόριθμο που ανήκει στην κατηγορία της βαθιάς μάθησης, τον αλγόριθμο N-Beats. Στο αμέσως επόμενο κεφάλαιο έχουμε μία εκτενή παρουσίαση της εφαρμογής που δημιουργήσαμε για την παραγωγή προβλέψεων. Στο κεφάλαιο 7 έχουμε συμπεριλάβει μία σύντομη ανάλυση των δεδομένων μας και το πειραματικό κομμάτι της εργασίας μας. Η εργασία ολοκληρώνεται με ένα κεφάλαιο που περιέχει μία σειρά από σκέψεις για μελλοντικές επεκτάσεις της.

Κεφάλαιο 2

Βασική Θεωρία Χρονοσειρών

2.1 Ανάλυση και Πρόβλεψη Χρονοσειρών

Ως χρονοσειρά ή χρονολογική σειρά ορίζουμε μία ακολουθία παρατηρήσεων που παίρνονται σε ισαπέχουσες χρονικές στιγμές. Μία χρονοσειρά εκφράζει την εξέλιξη ενός στοχαστικού συστήματος, δηλαδή ενός συστήματος με τυχαία κατά μάλλον ή ήττον συμπεριφορά και μπορεί να περιγραφεί μέσω μίας ακολουθίας τυχαίων μεταβλητών.[55] Η συστηματική μελέτη μίας χρονοσειράς ξεκινάει με την επισκόπηση του γραφήματός της στο πεδίο του χρόνου. Τα βασικά χαρακτηριστικά που βλέπουμε μέσα από αυτην είναι η τάση, η εποχιακότητα, η κυκλικότητα και οι ασυνέχειες. Ως τάση ορίζουμε την μακροπρόθεσμη μεταβολή του μέσου επιπέδου των τιμών της χρονοσειράς. Για να μπορούμε να θεωρήσουμε την μεταβολή αυτή μακροπρόθεσμη είναι σαφές ότι απαιτείται να διαθέτουμε ικανό κάθε φορά αριθμό παρατηρήσεων. Η κυκλικότητα αντιπροσωπεύει μια κυματοειδή μεταβολή που οφείλεται σε εξωγενείς παράγοντες και εμφανίζεται σε περιόδους όχι απαραίτητα σταθερές. Οι μεταβολές αυτές έχουν συνήθως διάρκεια τουλάχιστον δύο έτη[25]. Η εποχιακότητα ορίζεται σαν μια περιοδική διακύμανση με μήκος περιόδου σταθερό και μικρότερο του έτους. Οι αλλαγές που οφείλονται σ'αυτήν θεωρούνται εξηγήσιμες καθότι επαναλαμβάνονται με τον ίδιο τρόπο ανά τον χρόνο. Ως ασυνέχειες θεωρούνται οι απότομες αλλαγές στο πρότυπο της χρονοσειράς που δεν μπορούν να εξηγηθούν βάσει του παρελθόντος. Αναλόγως της διάρκειάς τους μπορούμε να τις κατατάξουμε σε outliers και level shifts. Οι πρώτες έχουν παροδικό χαρακτήρα, ενώ οι δεύτερες μόνιμο. Οι ασυνέχειες μπορούν να εντοπιστούν αυτόματα με ειδικούς στατιστικούς ελέγχους. Αυτή την τακτική ακολουθούμε όταν έχουμε μεγάλο πλήθος χρονοσειρών προς ανάλυση και είναι δύσκολο να έχουμε οπτική επαφή με κάθε μια ξεχωριστά. Γενικά, θεωρούμε ως μη κανονικές διακυμάνσεις εκείνες που απομένουν όταν έχουν απομονωθεί τα υπόλοιπα χαρακτήριστικά της χρονοσειράς (τάση, κύκλος και εποχιακότητα) και περιέχουν το στοιχείο της τυχαιότητας. Η γραφική απεικόνιση των δεδομένων μπορεί να αποκαλύψει και εσφαλμένες τιμές. Σε περίπτωση εντοπισμού μηδενικών ή ελλειπουσών τιμών θα πρέπει να παρέμβουμε και να τις διορθώσουμε καθώς δημιουργούν προβλήματα στην εφαρμογή των μεθόδων πρόβλεψης.[56] Οι ελλείπουσες τιμές οφείλονται σε αστοχία μέτρησης, ενώ οι μηδενικές τιμές μπορεί να αφορούν πέραν από αστοχίες μέτρησης και πραγματικές καταγραφές. Στην δεύτερη περίπτωση αναφερόμαστε σε χρονοσειρές διακοπτόμενης ζήτησης και δεν απαιτείται καμία ενέργεια μεταβολής τους. Για την εξάλειψη των ελλειπουσών τιμών συνήθως χρησιμοποιούμε είτε τον μέσο όρο των παρατηρήσεων που προηγούνται και έπονται αυτής είτε των παρατηρήσεων των αντίστοιχων περιόδων αν η χρονοσειρά παρουσιάζει έντονη εποχιακότητα. Για τα δε outliers αν η χρονοσειρά παρουσιάζει έντονη τάση τότε τα διορθώνουμε με γραμμική παρεμβολή, αλλιώς χρησιμοποιούμε μία σταθερή τιμή (π.χ. την αμέσως προηγούμενη ή ένα άλλο καθορισμένο όριο).

Πέραν της γραφικής απεικόνισης της χρονοσειράς και της εξαγωγής των πρώτων συμπερασμάτων επί αυτής είναι χρήσιμο να κάνουμε και μία βασική στατιστική ανάλυση της, χρησιμοποιώντας μια ποικιλία στατιστικών δεικτών. Οι βασικοί δείκτες είναι:

Η *μέση τιμή*, δηλαδή ο απλός γραμμικός μέσος όρος των τιμών των παρατηρήσεων που δείχνει το επίπεδο γύρω απ'το οποίο κινείται η χρονοσειρά μας και υπολογίζεται ως εξής:

$$\overline{Y} = \frac{1}{n}\sum_{i=1}^{n} Y_i$$

Η *μέγιστη και ελάχιστη τιμή*, οι οποίες αποτελούν μια πρώτη εκτίμηση της διακύμανσης των δεδομένων, αλλά και της τυχαιότητας που περιέχουν.
Η *τυπική απόκλιση* (διασπορά) που εκφράζει τον βαθμό κατά τον οποίο οι παρατηρήσεις της χρονοσειράς είναι διεσπαρμένες γύρω από τη μέση τιμή και υπολογίζεται ως εξής:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}{n}}$$

Η *διακύμανση*, που είναι το τετράγωνο της τυπικής απόκλισης.
Η *συνδιακύμανση*, η οποία αποτελεί ένα μέτρο σχέσης μεταξύ δύο τυχαίων μεταβλητών. Με αυτήν βλέπουμε κατά πόσο οι μεταβλητές μεταβάλλονται αναλόγως (θετική συνδιακύμανση), αντιστρόφως αναλόγως (αρνητική συνδιακύμανση) ή είναι ασυσχέτιστες (μηδενική συνδιακύμανση). Στις χρονοσειρές συνηθίζεται να υπολογίζουμε την συνδιακύμανση των δεδομένων σε σχέση με τον αύξοντα αριθμό της χρονικής περιόδου, έτσι ώστε να δούμε αν υπάρχει γραμμική συσχέτιση (το γράφημα προσεγγίζει μία ευθεία). Υπολογίζουμε την συνδιακύμανση για τις τυχαίες μεταβλητές X,Y ως ακολούθως:

$$COV(X,Y) = \frac{1}{n}\sum_{i=1}^{n}[(X_i - \overline{X})(Y_i - \overline{Y})]$$

Ο *συντελεστής Pearson* ή αλλιώς συντελεστής γραμμικής συσχέτισης εκφράζει τη συγκέντρωση των σημείων ενός διαγράμματος διασποράς γύρω από την ευθεία παλινδρόμησης, δίνοντας ένα μέτρο της γραμμικής συσχέτισης μεταξύ των δύο μεταβλητών.[56] Ο συντελεστής παίρνει τιμές μεταξύ -1 και 1 και αναλόγως πόσο κοντά είναι το μέτρο του στην μονάδα τόσο πιο ισχυρή είναι η γραμμική συσχέτιση. Εντελώς παρομοίως με την συνδιακύμανση η συσχέτιση γίνεται συνήθως θεωρώντας ως μια μεταβλητή το χρόνο. Ο υπολογισμός του συντελεστή Pearson γίνεται από τον ακόλουθο τύπο:

$$r_{XY} = \frac{\sum_{i=1}^{n}[(X_i - \overline{X})(Y_i - \overline{Y})]}{\sum_{i=1}^{n}(X_i - \overline{X})^2 \sum_{i=1}^{n}(X_i - \overline{X})^2}$$

Ο *συντελεστής αυτοσυσχέτισης* εκφράζει τη συσχέτιση μεταξύ παρατηρήσεων της ίδιας μεταβλητής με χρονική υστέρηση k περιόδους και λαμβάνει τιμές μεταξύ 0 και 1. Για παράδειγμα, ο συντελεστής ACF_2 μετράει την συσχέτιση μεταξύ της τιμής της χρονοσειράς για μια χρονική στιγμή t και της τιμής της χρονοσειράς για την χρονική στιγμή t-2. Όσο μεγαλύτερος είναι τόσο μεγαλύτερη είναι και η συσχέτιση. Πρόκειται για πολύ χρήσιμο δείκτη στον εντοπισμό της εποχιακότητας. Για τις χρονοσειρές με τάση, οι συντελεστές αυτοσυσχέτισης για μικρή υστέρηση έχουν μεγάλη θετική τιμή, γιατί οι κοντινές χρονικά παρατηρήσεις δεν διαφέρουν σημαντικά μεταξύ τους.[25] Στις σειρές αυτές, βάσει της άνω παρατήρησης, οι συντελεστές μειώνονται αργά όσο αυξάνεται η τιμή της υστέρησης. Ο συντελεστής αυτοσυσχέτισης υπολογίζεται βάσει του ακόλουθου τύπου:

$$ACF_k = \frac{\sum_{i=1+k}^{n}[(Y_i - \overline{Y})(Y_{i-k} - \overline{Y})]}{\sum_{i=1+k}^{n}(Y_i - \overline{Y})^2}$$

Ο *συντελεστής μεταβλητότητας* που αποτελεί ένα κανονικοποιημένο μέτρο της διασποράς των παρατηρήσεων. Δεν εξαρτάται από το επιπέδο των παρατηρήσεων, ωστόσο δεν μπορεί να

υπολογιστεί όταν η μέση τιμή τους ισούται με μηδέν. Για τον υπολογισμό του εφαρμόζουμε τον παρακάτω τύπο:

$$c_v = \frac{\sigma}{\overline{Y}} 100\%$$

Όπως αναφέρθηκε και παραπάνω μπορούμε να αποκτήσουμε αίσθηση για την εποχιακότητα των δεδομένων μας μέσω του συντελεστή αυτοσυσχέτισης. Είναι σημαντικό να ελέγξουμε αριθμητικά αν η συνιστώσα της εποχιακότητας είναι σημαντική ή όχι στην χρονοσειρά μας. Ο έλεγχος της σημαντικότητας της εποχιακότητας μίας χρονοσειράς εφαρμόζεται μέσω του ελέγχου αυτοσυσχέτισης δεδομένων με περίοδο καθυστέρησης (k) ίσης με τον αριθμό των περιόδων ενός κύκλου εποχιακότητας (pos) σε σύγκριση με τις αυτοσυσχετίσεις περιόδου καθυστέρησης εώς και μίας μονάδας μικρότερης από τον αριθμό των περιόδων ενός κύκλου εποχιακότητας.[56] Μια χρονοσειρά θεωρείται εποχιακή αν και μόνο αν ισχύει:

$$|ACF_{pos}| > Limit$$

$$limit = t_{critical}\sqrt{\frac{1 + 2(ACF_1 + \sum_{i=2}^{pos-1} ACF_i^2)}{n}}$$

Η τιμή του συντελεστή $t_{critical}$ καθορίζεται από το επίπεδο εμπιστοσύνης που επιθυμούμε για τον έλεγχο σημαντικότητας εποχιακής συμπεριφοράς. Έχουμε τα ακόλουθα ζεύγη:

Επίπεδο Εμπιστοσύνης	$t_{critical}$
80%	1.28
90%	1.645
95%	1.96
98%	2.33
99%	2.58

Πίνακας 2.1: Ζεύγη $t_{critical}$ και επιπέδων εμπιστοσύνης

Ένα μοντέλο πρόβλεψης αντιπροσωπεύει την διαδικασία που ακολουθείται για να παραχθούν προβλέψεις. Για την παραγωγή ποσοτικών προβλέψεων έχουμε τα *μοντέλα χρονοσειρών* και τα *αιτιοκρατικά μοντέλα*. Στα πρώτα κάνουμε την υπόθεση πως η μεταβολή της τιμής της χρονοσειράς ακολουθεί ένα λανθάνον πρότυπο που επαναλαμβάνεται στο χρόνο και διατηρείται σταθερό. Οι προβλέψεις γίνονται με την προέκταση του προτύπου αυτού. Ως είσοδο του μοντέλου έχουμε τις ιστορικές παρατηρήσεις. Οι μέθοδοι του μοντέλου αυτού λέγονται μέθοδοι χρονοσειρών. Οι κυριότερες είναι η αποσύνθεση, οι μέθοδοι εξομάλυνσης και οι αυτοπαλινδρούμενες μέθοδοι κινητού μέσου όρου. Στα αιτιοκρατικά μοντέλα υποθέτουμε την ύπαρξη μίας σταθερής σχέσης μεταξύ του υπό πρόβλεψη μεγέθους και ορισμένων μεταβλητών. Το πρώτο το αναφέρουμε ως εξαρτημένη μεταβλητή και τις δεύτερες ως ανεξάρτητες. Σε αυτήν την περίπτωση χρειάζεται να αναλύσουμε την κάθε μία μεταβλητή είτε εξαρτημένη είτε ανεξάρτητη. Εμείς θα ασχοληθούμε στην παρούσα εργασία κυρίως με μεθόδους χρονοσειρών.
Η παραγωγή προβλέψεων μπορεί να γίνει με εφαρμογή στατιστικών μεθόδων (*στατιστική πρόβλεψη*), με συμβολή ειδικών και εμπειρογνωμώνων (*κριτική πρόβλεψη*), ενώ μπορεί να αφορά και κάποιον επιθυμητό στόχο (*πρόβλεψη στόχου ή προυπολογισμού*). Στην κριτική πρόβλεψη σημαντικό ρόλο παίζουν οι γνώσεις και η διαίσθηση ενός ατόμου ή μίας επιτροπής

ατόμων. Οι κριτικές προβλέψεις δεν απαιτούν μεγάλο αριθμό ιστορικών παρατήρησεων και λαμβάνουν υπόψιν ειδικά γεγονότα και ενέργειες. Ωστόσο, είναι επιρρεπείς σε προκαταλήψεις. Η πρόβλεψη στόχου αναφέρεται σε μία επιθυμητή αναπτυξιακή κατάσταση και χρησιμοποιεί την έννοια του ρυθμού ανάπτυξης. Ο δείκτης του ρυθμού ανάπτυξης αποτελεί ένα μέτρο της αυξητικής ή φθίνουσας πορείας μίας χρονοσειράς για ένα χρονικό διάστημα. Εκφράζεται σε ποσοστιαία μορφή και αναφέρεται στη σύγκριση του ύψους των δεδομένων του τελευταίου έτους σε σχέση με τα υπόλοιπα διαθέσιμα δεδομένα. Βάσει της επιθυμητής μελλοντικής κατάστασης τροποποιούμε τον ρυθμό ανάπτυξης που υπολογίζεται από τις στατιστικές μεθόδους για τον ορίζοντα πρόβλεψης. Η τροποποίηση μπορεί να εφαρμοστεί είτε εξίσου σε κάθε περίοδο ή να δοθεί μεγαλύτερο βάρος στις τελευταίες περιόδους του ορίζοντα πρόβλεψης.[56] Η πρόβλεψη αυτή περιέχει μεροληψία και σφάλματα. Η τελική πρόβλεψη μπορεί να είναι συνδυασμός των τριών μεθόδων ή και όχι.
Συνήθως, επιζητούμε να κάνουμε πρόβλεψη σε βάθος χρόνου. Ο αριθμός των περιόδων που καλούμαστε να προβλέψουμε λέγεται ορίζοντας πρόβλεψης. Αναλόγως την τιμή του χωρίζουμε τις προβλέψεις σε βραχυπρόθεσμες, όπου η τιμή του ορίζοντα είναι μικρή, συνήθως μικρότερη των τριών περιόδων (π.χ. σχεδιασμός αποθήκης), μεσοπρόθεσμες όπου ο ορίζοντας είναι συνήθως λίγο μεγαλύτερος του οικονομικού έτους, 12-15 περίοδοι αν αναφερόμαστε σε μηνιαία χρονοσειρά και την μακροπρόθεσμη όπου είναι μεγαλύτερος των τριών ετών. Είναι σημαντικό να παρατηρήσουμε ότι οι διάφορες μέθοδοι λειτουργούν καλύτερα ή χειρότερα ανάλογα με τον ορίζοντα πρόβλεψης. Γενικά όμως ισχύει πως το επίπεδο αβεβαιότητας μεγαλώνει καθώς καλούμαστε να προβλέψουμε για μεγαλύτερο ορίζοντα. Στην δική μας περίπτωση θα ασχοληθούμε με βραχυπρόθεσμες και μεσοπρόθεσμες προβλέψεις.

2.2 Στατιστικές Προβλέψεις

Οι στατιστικές προβλέψεις αναφέρονται στην εφαρμογή στατιστικών μοντέλων χρονοσειρών ή αιτιοκρατικών μοντέλων επί μίας σειράς δεδομένων με σκοπό την αυτοματοποιημένη και συστηματική παραγωγή προβλέψεων.[56] Οι προβλέψεις αυτές μπορούν να συνοδεύονται από κάποιο διάστημα εμπιστοσύνης, όπου δίνουμε ένα εύρος τιμών που μπορεί να πάρει στο μέλλον η χρονοσειρά με μεγάλη πιθανότητα, ή και όχι. Στην δεύτερη περίπτωση αναφερόμαστε σε σημειακές προβλέψεις. Στο παρών κεφάλαιο θα ασχοληθούμε με σημειακές προβλέψεις. Οι μέθοδοι πρόβλεψης αυτού του τύπου μπορούν να εφαρμοστούν άμεσα σε μεμονωμένες χρονοσειρές ή μαζικά σε δέσμη χρονοσειρών, καθώς δεν απαιτούν πολλούς υπολογιστικούς πόρους. Οι μέθοδοι αυτές προϋποθέτουν ότι το πρότυπο της χρονοσειράς θα συνεχιστεί στο μέλλον ενώ δεν λαμβάνει υπόψιν ειδικά γεγονότα και ενέργειες που ενδέχεται να πραγματοποιηθούν στο μέλλον, κάτι που συμβαίνει στις κριτικές προβλέψεις. Τέλος, οι στατιστικές μέθοδοι απαιτούν ικανό πλήθος ιστορικών παρατηρήσεων η συλλογή των οποίων κάποιες φορές δεν είναι εύκολη. Στην συνέχεια θα αναφερθούμε στις σημαντικότερες στατιστικές μεθόδους προβλέψεων. Σε κάθε περίπτωση θεωρούμε πως έχουμε T ιστορικές παρατηρήσεις, Y_i, με $i \in \{1,2,...,T\}$ και προβλέψεις F_j με $j \in \{1,2,...,T,T+1,...,T+H\}$, όπου H ο ορίζοντας πρόβλεψης.
Η μέθοδος *Naive* αποτελεί την απλούστερη στατιστική μέθοδο. Η πρόβλεψη για μία χρονική στιγμή είναι ίση με την πραγματική παρατήρησή της ακριβώς προηγουμένης χρονικής περιόδου (t-1). Δηλαδή, έχουμε $F_t = Y_{t-1}$. Έτσι, για κάθε μελλοντική περίοδο έχουμε ως πρόβλεψη την τελευταία διαθέσιμη παρατήρησή. Συνεπώς, ισχύει ότι $F_{t+T} = Y_T$, με $t \in \{1,2,...,H\}$ Η μέθοδος αυτή συνήθως δεν παράγει ακριβείς προβλέψεις και χρησιμοποιείται ως σημείο αναφοράς για πιο πολύπλοκες μεθόδους. Σε κάποιες περιπτώσεις όμως έχει αξιοσημείωτα καλή απόδοση, όπως σε κάποιες οικονομικού τύπου χρονοσειρές. Με την Naive υποθέτουμε πως η πιο πρόσφατη παρατήρησή είναι η πιο σημαντική και οι προηγούμενες δεν παρέχουν ουσιαστικά καμία πληροφορία για το μέλλον.

Η μέθοδος του *απλού μέσου όρου* δίνει ως πρόβλεψη για όλες τις χρονικές περιόδους τον μέσο όρο των παρατηρήσεών μας. Δηλαδή, έχουμε $\mathrm{F_{T+t}} = \frac{1}{T}\sum_{i=1}^{T} Y_i$, με $t \in \{1,2,...,\mathrm{H}\}$. Συνεπώς, η μέθοδος αυτή θεωρεί πως όλες οι παρατηρήσεις έχουν ίση σημασία για την πρόβλεψη του μέλλοντος.
Μια παραλλαγή της Naive είναι η *εποχιακή Naive* που χρησιμοποιείται για δεδομένα με έντονο το στοιχείο της εποχιακότητας. Σε αυτήν την περίπτωση θέτουμε κάθε πρόβλεψη ίση με την τελευταία παρατήρηση της αντίστοιχης όμως περιόδου (π.χ. αν θέλουμε να προβλέψουμε την τιμή για κάποιον Οκτώβριο θα πάρουμε την τιμή του τελευταίου διαθέσιμου Οκτωβρίου). Για εποχιακή περίοδο n, ισχύει ότι:

$$F_{T+t} = Y_{T+t-n(\lfloor \frac{t-1}{n} \rfloor+1)},\ \mu\varepsilon\ t \in \{1,2,...,H\}$$

Μια άλλη παραλλαγή της Naive είναι η μέθοδος της *τάσης*. Εδώ οι προβλέψεις αυξάνονται ή μειώνονται με τον χρόνο κατά μία τιμή που ισούται με την μέση αλλαγή που βλέπουμε στις παρατηρήσεις μας . Ουσιαστικά ενώνουμε με μία γραμμή την πρώτη και την τελευταία παρατήρηση και στη συνέχεια επεκτείνουμε την γραμμή στο μέλλον για όσες περιόδους ορίζει ο ορίζοντας πρόβλεψης μας.
Στην ανάλυση και πρόβλεψη χρονοσειρών αναφερθήκαμε στις τέσσερις συνιστώσες κάθε χρονοσειράς. Μπορούμε να εξάγουμε κάθε μια από αυτές με την διαδικασία της αποσύνθεσης, όπου εφαρμόζουμε μια σειρά από απλές μαθηματικές σχέσεις επί της χρονοσειράς μας. Η αποσύνθεση χρησιμοποιείται συνήθως για την καλύτερη κατανόηση της χρονοσειράς, μπορεί να χρησιμοποιηθεί για την βελτίωση των προβλέψεων μας αλλά και η ίδια ως μέθοδος πρόβλεψης.Η μαθηματική διατύπωση της αποσύνθεσης είναι η εξής:

$$Y_t = f(S_t, T_t, C_t, R_t)$$

Όπου $\mathrm{S_t}$ είναι η συνιστώσα της εποχιακότητας, $\mathrm{T_t}$ είναι η συνιστώσα της τάσης, $\mathrm{C_t}$ είναι η συνιστώσα της κυκλικότητας και $\mathrm{R_t}$ είναι η συνιστώσα της τυχαιότητας, όλες για την χρονική στιγμή t. Συνήθως, υποθέτουμε ότι η άνω συναρτησιακή σχέση είναι η προσθετική ή η πολλαπλασιαστική, δηλαδή είτε έχουμε το προσθετικό μοντέλο:

$$Y_t = S_t + T_t + C_t + R_t$$

είτε το πολλαπλασιαστικό:

$$Y_t = S_t * T_t * C_t * R_t$$

Σημειώνεται ότι πολλές φορές συγχωνεύουμε τις συνιστώσες της τάσης και της κυκλικότητας. Στο παρών κεφάλαιο θα ασχοληθούμε με το πολλαπλασιαστικό μοντέλο. Θα αναφερθούμε σε δύο μεθόδους αποσύνθεσης, τους κινητούς μέσους όρους και την κλασσική μέθοδο αποσύνθεσης.
Η μέθοδος των κινητών μέσων όρων αποτελεί την απλούστερη μέθοδο αποσύνθεσης. Ως στόχο έχει την εκτίμηση της σειράς τάσης-κύκλου για κάθε παρατήρηση. Για να πετύχουμε αυτήν την εκτίμηση χρησιμοποιούμε γειτονικές παρατηρήσεις. Αναλόγως του τύπου του κινητού μέσου όρου που χρησιμοποιούμε οι γειτονικές αυτές παρατηρήσεις συνεισφέρουν με διαφορετικό βάρος στο κάθε αποτέλεσμα. Για τον απλό κινητό μέσο όρο, σε κάθε παρατήρηση που θέλουμε να υπολογίσουμε την σειρά τάσης-κύκλου απλώς βρίσκουμε τον μέσο όρο n τιμών γύρω από την παρατήρηση μας. Ο αριθμός n πρέπει να είναι περιττός. Με άλλα λόγια προσθέτουμε n mod 2 παρατηρήσεις πριν και n mod 2 παρατηρήσεις μετά την παρατήρηση μας, καθώς και την παρατήρησή μας και ύστερα διαιρούμε με το πλήθος n. Εδώ κάθε παρατήρηση συμμετέχει εξίσου στον υπολογισμό της τελικής τιμής. Η άνω διαδικασία εφαρμόζεται σ'όλο το μήκος της χρονοσειράς μας εκτός των n mod 2 πρώτων και n mod

2 τελευταίων παρατηρήσεων λόγω έλλειψης δεδομένων. Οι τιμές αυτές υπολογίζονται με διάφορες τεχνικές, όπως το backcasting. Η επιλογή του n είναι σημαντική καθότι πέραν ότι όσο μεγαλύτερο είναι τόσο πιο πολλές κενές θέσεις μένουν στην αρχή και το τέλος, καθορίζει το βαθμό εξομάλυνσης των δεδομένων. Όσο μεγαλύτερο είναι τόσο περισσότερο εξομαλύνονται τα δεδομένα. Πέραν όμως του απλού κινητού μέσου όρου μπορούμε να χρησιμοποιήσουμε σταθμισμένο κινητό μέσο όρο, όπου οι γειτονικές παρατηρήσεις έχουν άνισα βάρη στον υπολογισμό, διπλό κινητό μέσο όρο, όπου έχουμε εφαρμογή σε δύο στάδια ίσων ή άνισων μηκών του απλού κινητού μέσου όρου (και εδώ έχουμε συνολικά άνισα βάρη) και πραγματοποιείται διπλή εξομάλυνση, αλλά και κεντρικό μέσο όρο, ο οποίος συνδυάζει απλό και διπλό κεντρικό μέσο όρο για να παρακάμψει το ζήτημα του περιττού n. Και στον κεντρικό μέσο όρο οι παρατηρήσεις έχουν άνισα βάρη στον υπολογισμό. Μπορούμε, τέλος, να χρησιμοποιήσουμε συνδυασμούς από κεντρικούς μέσους όρους αυτούσιους για να κάνουμε προβλέψεις.
Η κλασσική μέθοδος αποσύνθεσης αποτελείται από μία σειρά βημάτων. Αρχικά, χρησιμοποιώντας έναν κινητό μέσο όρο μήκους ανάλογο της εποχιακότητας της χρονοσειράς αφαιρούμε την εποχιακότητα και την τυχαιότητα από τα δεδομένα μας. Ακολούθως, διαιρούμε τα δεδομένα με το αποτέλεσμα του προηγούμενου βήματος και παίρνουμε τους λόγους εποχιακότητας απ'τους οποίους στην συνέχεια βρίσκουμε τους δείκτες εποχιακότητας με μέσο όρο των λόγων στις αντίστοιχες περιόδους και κανονικοποίηση των αποτελεσμάτων. Στο επόμενο βήμα απλώς διαιρούμε τα δεδομένα με τους κατάλληλους δείκτες εποχιακότητας ώστε να πάρουμε την αποεποχικοποιημένη χρονοσειρά. Τα βήματα αυτά μπορούν να παραληφθούν αν η χρονοσειρά μας δεν είναι εποχιακή. Εν συνεχεία, χρησιμοποιώντας έναν κινητό μέσο όρο μήκους τριών ή έξι παρατηρήσεων επί της αποεποχικοποιημένης χρονοσειράς βρίσκουμε την σειρά τάσης-κύκλου. Η τυχαιότητα μπορεί να υπολογιστεί με διαίρεση των τιμών της αποεποχικοποιημένης χρονοσειράς με την σειρά τάσης-κύκλου. Από την σειρά τάσης-κύκλου με διάφορες τεχνικές (π.χ. απλή γραμμική παλινδόμηση για γραμμική τάση) μπορούμε να εξάγουμε την τάση, ενώ με διαίρεση της σειράς-τάσης κύκλου με την υπολογισθείσα τάση παίρνουμε την συνιστώσα της κυκλικότητας. Έχοντας αναλύσει την αρχική χρονοσειρά στις συνιστώσες της μπορούμε να τις συνθέσουμε πολλαπλασιαστικά ώστε να εκτιμήσουμε μελλοντικές τιμές της χρονοσειράς. Στην πρόβλεψη δεν χρήσιμοποιούμε την συνιστώσα της τυχαιότητας. Έτσι έχουμε:

$$F_{T+i} = S_{T+i} * T_{T+i} * C_{T+i}$$

Στο κομμάτι του S_{T+i} χρησιμοποιούμε τον αντίστοιχο δείκτη εποχιακότητας, ενώ στις υπόλοιπες συνιστώσες προεκτείνουμε τα αποτελέσματά μας με κάποια από τις διαθέσιμες μεθόδους.
Μία σημαντική κατηγορία στατιστικών μεθόδων αποτελούν οι *μέθοδοι εκθετικής εξομάλυνσης*. Οι μέθοδοι αυτές λειτουργούν στην λογική πως όσο πιο πρόσφατα είναι τα δεδομένα τόσο περισσότερη πληροφορία περιέχουν. Έτσι, η βαρύτητα που δίνουν στις παρατηρήσεις φθίνει όσο προχωράμε πίσω στον χρόνο. Αυτές οι μέθοδοι χρησιμοποιούνται ιδιαιτέρως σε περιπτώσεις βραχυπρόθεσμου σχεδιασμού και παρουσιάζουν πολυ καλά αποτελέσματα ακόμη και συγκριτικά με πιο πολύπλοκες μεθόδους καθώς δεν επηρεάζονται από τις ιδιομορφίες των προτύπων των δεδομένων ή από σποραδικά εμφανιζόμενες ακραίες τιμές που παρατηρούνται στα δεδομένα.
Το πρώτο μοντέλο της κατηγορίας λέγεται *απλη εκθετική εξομάλυνση* (Simple Exponential Smoothing-SES) και επιλέγεται όταν τα δεδομένα μας χαρακτηρίζονται από απουσία τάσης. Σε αυτήν την περίπτωση το μοντέλο ορίζεται από τρεις εξισώσεις:

$$e_t = Y_t - F_t$$

$$S_t = S_{t-1} + \alpha e_t$$

$$F_{t+1} = S_t$$

Ως e δηλώνουμε το σφάλμα, δηλαδή την απόκλιση μεταξύ πραγματικής τιμής και πρόβλεψης, ως S ορίζουμε το επίπεδο και F την πρόβλεψη για την χρονική περίοδο t. Η παράμετρος α λέγεται συντελεστής εξομάλυνσης και λαμβάνει τιμές στο διάστημα [0,1]. Στην περίπτωση που θέλουμε να υπολογίσουμε περισσότερες της μίας σημειακές προβλέψεις (ο παραπάνω τύπος αναφέρεται σε πρόβλεψη της επόμενης χρονικής περιόδου μόνο) τότε θέτουμε απλώς όλες τις επόμενες προβλέψεις ίσες με την τελευταία υπολογισμένη πρόβλεψη (επίπεδες προβλέψεις), καθώς το μοντέλο θεωρεί την ύπαρξη ενός σταθερού επιπέδου. Αν ξεδιπλώσουμε τις παραπάνω εξισώσεις μπορούμε να γράψουμε την πρόβλεψη και ως ακολούθως:

$$F_{t+1} = \alpha Y_t + (1-\alpha)F_t$$

Βάσει αυτού βλέπουμε πως η πρόβλεψη είναι ένας γραμμικός συνδυασμός της προηγούμενης παρατήρησης και της προηγούμενης πρόβλεψης με βάρη βάσει του συντελεστή εξομάλυνσης. Ομοίως, συμβαίνει και στον υπολογισμό της αμέσως προηγούμενης πρόβλεψης όπου συμμετέχει η παραπροηγούμενη πρόβλεψη και η παραπροηγούμενη πραγματική τιμή. Συνεπώς, αν κινηθούμε με τον τρόπο αυτό πίσω στον χρόνο βλέπουμε πως η πρόβλεψη για την επόμενη περίοδο εξαρτάται από όλες τις ιστορικές παρατηρήσεις με βάρη που φθίνουν εκθετικά. Δηλαδή, καταλήγουμε στην ακόλουθη σχέση:

$$F_{t+1} = \alpha Y_t + \alpha(1-\alpha)Y_{t-1} + \alpha(1-\alpha)^2\Upsilon_{t-2} + ... + \alpha(1-\alpha)^{t-1}\Upsilon_1 + (1-\alpha)^t F_1$$

Για την μέθοδο αυτή καλούμαστε να επιλέξουμε δύο παραμέτρους, τον συντελεστή εξομάλυνσης και το αρχικό επίπεδο. Ως αρχικό επίπεδο μπορούμε να επιλέξουμε τον μέσο όρο των παρατηρήσεων, τον μέσο όρο των πρώτων παρατηρήσεων, την πρώτη παρατήρηση ή κάποια άλλη τιμή που να αντανακλά το επίπεδο των δεδομένων μας. Για την επιλογή του συντελεστή εξομάλυνσης συνήθως χρησιμοποιούμε γραμμική αναζήτηση για αυτήν που ελαχιστοποιεί το μέσο τετραγωνικό σφάλμα στα ιστορικά δεδομένα μας. Η λογική είναι πως όταν έχουμε μικρότερο μέσο τετραγωνικό σφάλμα το μοντέλο προσεγγίζει καλύτερα τα δεδομένα μας και συνεπώς είναι πιθανότερο να δώσει καλύτερες προβλέψεις. Για τιμές κοντά στην μονάδα η μέθοδος δίνει πολύ μεγάλη αξία στην τελευταία παρατήρηση και σημαντικά λιγότερη στις παλαιότερες ενώ για τιμή ίση με το ένα η μέθοδος ταυτίζεται με την Naive. Αν η τιμή του συντελεστή είναι μηδέν τότε ως πρόβλεψη δίνουμε το αρχικό επίπεδο που έχουμε ορίσει.

Μία επέκταση της απλής εκθετικής εξομάλυνσης για δεδομένα με γραμμική τάση είναι το *μοντέλο γραμμικής τάσης* (Holt Exponential Smoothing). Σ'αυτήν την περίπτωση το μοντέλο περιγράφεται από τις ακόλουθες εξισώσεις:

$$e_t = Y_t - F_t$$
$$S_t = S_{t-1} + T_{t-1} + \alpha e_t$$
$$T_t = T_{t-1} + \alpha\beta e_t$$
$$F_{t+m} = S_t + mT_t$$

Πέραν των μεταβλητών που είδαμε στην SES, έχουμε επιπλέον την τάση T, τον χρονικό ορίζοντα πρόβλεψης m και τον συντελεστή εξομάλυνσης της τάσης β που όπως και ο συντελεστής εξομάλυνσης για το επίπεδο παίρνει τιμές στο διάστημα [0,1]. Εκτός του αρχικού επιπέδου και του συντελεστή α πρέπει να ορίσουμε την αρχική τάση T_0 καθώς και τον συντελεστή β. Για την αρχική τάση μπορούμε μεταξύ άλλων να χρησιμοποιήσουμε την διαφορά δεύτερης και πρώτης παρατήρησης, την διαφορά n-οστής και πρώτης δια n. Για το αρχικό επίπεδο δεν αλλάζει κάτι σε σχέση με όσα αναφέρθηκαν στην SES. Όσον αφορά τους δύο συντελεστές κάνουμε και πάλι γραμμική αναζήτηση για κάθε συνδυασμό τους με στόχο την ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος. Συνήθως, η βέλτιστη τιμή του β είναι μικρότερη από αυτήν του α.[56]

Μία προσαργμογή της μεθόδου γραμμικής τάσης για χρονοσειρές με μη γραμμικής τάσεις είναι τα μοντέλα *μη γραμμικής τάσης*. Πιο συγκεκριμένα, εισάγουμε την παράμετρο διόρθωσης της τάσης φ, η οποία ελέγχει τον ρυθμό αύξησης των τιμών που παράγει το μοντέλο. Εδώ το μοντέλο περιγράφεται από τις παρακάτω εξισώσεις:

$$e_t = Y_t - F_t$$
$$S_t = S_{t-1} + \varphi T_{t-1} + \alpha e_t$$
$$T_t = \varphi T_{t-1} + \alpha\beta e_t$$
$$F_{t+m} = S_t + \sum_{i=1}^{m} \varphi^i T_t$$

Στην τελευταία εξίσωση που περιγράφει την πρόβλεψη για την χρονική περίοδο t+m βλέπουμε πως εκτός του τελευταίου υπολογισμένου επιπέδου στο πέρας της χρονοσειράς μας έχουμε την συμβολή της τάσης. Όμως εδώ δεν γίνεται ένας γραμμικός υπολογισμός της όπως στην προηγούμενη μέθοδο λόγω της παραμέτρου εξομάλυνσης φ. Η τιμή που παίρνει αυτή η παράμετρος καθορίζει και το ακριβές μοντέλο που προκύπτει. Ειδικότερα, για φ=0 έχουμε την SES, για $0 < \varphi < 1$ τότε έχουμε το μοντέλο φθίνουσας τάσης (Damped Exponential Smoothing), για φ=1 έχουμε το μοντέλο γραμμικής τάσης και για $\varphi > 1$ έχουμε το μοντέλο εκθετικής τάσης. Το τελευταίο χαρακτηρίζεται από προκατάληψη και συνήθως αποφεύγεται. Συνεπώς, για κάθε ειδός τάσης που μπορούμε να παρατηρήσουμε στα δεδομένα μας, δημιουργούμε το κατάλληλο μοντέλο προσαρμόζοντας την παράμετρο φ. Τα παραπάνω μοντέλα υποθέτουν πως η χρονοσειρά μας δεν είναι εποχιακή. Ωστόσο, μπορούμε να τα επεκτείνουμε ώστε να καλύπτουν και αυτήν την περίπτωση. Πιο συγκεκριμένα, θα εισάγουμε έναν εποχιακό παράγοντα με στόχο την διόρθωση των προβλέψεων σύμφωνα με την αναμενόμενη εποχιακή διακύμανση. Διακρίνουμε την εποχιακή συμπεριφορά των χρονοσειρών σε δύο τύπους, την προσθετική και την πολλαπλασιαστική. Στην πρώτη, το πλάτος του εποχιακού προτύπου, δηλαδή το εύρος της εντός του χρονικού διαστήματος που αντιστοιχεί στο έτος, παραμένει σταθερό στον χρόνο και ανεξάρτητο του μέσου επιπέδου εντός του έτους, συνεπώς είναι σταθερό παρά την όποια τάση. Στην δεύτερη περίπτωση, το πλάτος του εποχιακού πρότυπου είναι ανάλογο του μέσου επιπέδου της χρονοσειράς.[30] Άρα, αν έχουμε χρονοσειρά με τάση μεταβάλλεται και η ένταση της εποχιακότητας. Αν έχουμε προσθετική εποχιακότητα ο εποχιακός παράγοντας είναι η διαφορά της κάθε παρατήρησης με τον μέσο όρο των τιμών της χρονοσειράς για όλο το έτος, ενώ για πολλαπλασιαστική είναι ο λόγος της κάθε παρατήρησης με τον προαναφερθέντα μέσο όρο. Χρησιμοποιώντας τον παράγοντα αυτό υπολογίζουμε τις αποεποχικοποιημένες τιμές των δεδομένων, ειτέ αφαιρώντας τον απ'τα δεδομένα (προσθετική εποχιακότητα) είτε διαιρώντας τα δεδομένα μ'αυτόν (πολλαπλασιαστική εποχιακότητα). Αν θέλουμε να πάμε από τα αποεποχικοποιημένα δεδομένα στα αρχικά ακολουθούμε την αντίστροφη διαδικασία. Η εποχιακότητα μπορεί να ενσωματωθεί τόσο για εκθετική εξομάλυνση με σταθερό επίπεδο όσο και με τάση (γραμμική ή μη γραμμική).
Όταν έχουμε πολλαπλασιαστικό μοντέλο εποχιακότητας και σταθερό επίπεδο, αναφερόμαστε στο μοντέλο *Winters*. Οι εξισώσεις που το ορίζουν είναι οι εξής:

$$e_t = Y_t - F_t$$
$$S_t = S_{t-1} + \frac{\alpha e_t}{I_{t-p}}$$
$$I_t = I_{t-p} + \frac{\gamma e_t}{S_{t-1}}$$
$$F_{t+m} = S_t I_{t-kp+m}$$

Ο εποχιακός παράγοντας συμβολίζεται με I, ενώ ως p θεωρούμε το πλήθος των περιόδων ανά έτος. Για παράδειγμα, αν έχουμε ωριαίες παρατηρήσεις p=24. Τονίζεται ότι το επίπεδο S_t

είναι αποεποχικοποιημένο και μεταβάλλεται κατά ένα ποσοστό α του αποεποχικοποιημένου επίσης σφάλματος πρόβλεψης e_t. Στον υπολογισμό της πρόβλεψης χρησιμοποιούμε τον εποχιακό παράγοντα του τελευταίου έτους. Γι'αυτό, χρησιμοποιούμε τον μετρητή k για το έτος που βρίσκεται η χρονική περίοδος πρόβλεψης t+m. Η σημαντικότερη διαφοροποίηση με τα προηγούμενα μοντέλα είναι η τρίτη εξίσωση που καθορίζει την αλλαγή του εποχιακού παράγοντα στον χρόνο βάσει του σφάλματος, του επιπέδου της προηγούμενης περιόδου και ενός συντελεστή εξομάλυνσης γ που λαμβάνει τιμές στο [0,1]. Και εδώ χρειάζεται να κάνουμε τις απαραίτητες αρχικοποιήσεις για να τρέξει η μέθοδός μας. Για τους αρχικούς εποχιακούς παράγοντες χρειαζόμαστε p παρατηρήσεις, άρα οι αρχικοποιήσεις των επιπέδων θα γίνουν για την περίοδο p και όχι μηδέν όπως προηγουμένως. Αυτό σημαίνει ότι το μοντέλο παράγει προβλέψεις ξεκινώντας από την περίοδο p+1. Για το αρχικό επίπεδο παίρνουμε τον μέσο όρο των πρώτων p παρατηρήσεων ενώ για το αρχικό επίπεδο τάσης (στα μοντέλα που έχουν) χρησιμοποιούμε 2p παρατηρήσεις παίρνοντας τις διαφορές των αντίστοιχων περιόδων με ίσα βάρη δια το πλήθος των περιόδων. Ακολουθούν οι εξισώσεις για το προσθετικό μοντέλο σε περίπτωση που έχουμε τάση, που λέγεται *Holt-Winters*:

$$e_t = Y_t - F_t$$

$$S_t = S_{t-1} + T_{t-1} + \alpha e_t$$

$$T_t = T_{t-1} + \alpha\beta e_t$$

$$I_t = I_{t-p} + \gamma e_t$$

$$F_{t+m} = S_t + mT_t + I_{t-kp+m}$$

Σημειώνεται ότι οι εξισώσεις για τους υπόλοιπους συνδυασμούς τύπου εποχιακότητας και επιπέδου, π.χ. πολλαπλασιαστική εποχιακότητα με μη γραμμική τάση, ακολουθούν την ίδια λογική με τις προαναφερθείσες και για τον λόγο αυτό θα τις παραλείψουμε.
Μία σημαντική οικογένεια μοντέλων πρόβλεψης είναι τα μοντέλα *ARIMA*. Τα μοντέλα αυτά στοχεύουν στην περιγραφή των αυτοσυσχετίσεων μεταξύ των δεδομένων μας. Στην γενική τους μορφή αποτελούν γραμμικό συνδυασμό παρελθοντικών τιμών, του τυχαίου παράγοντα (σφάλμα πρόβλεψης) και σχετικών στοχαστικών παραγόντων. Κάθε φορά προσπαθούμε να ανακαλύψουμε τον βέλτιστο τέτοιο συνδυασμό για να κάνουμε την πρόβλεψή μας. Από την στιγμή που στηριζόμαστε στο παρελθόν όσο βαθύτερα προβλέπουμε για το μέλλον τόσο περισσότερο βασιζόμαστε στις προβλέψεις μας έναντι των παλαιότερων πραγματικών παρατηρήσεων. Συνεπώς, αφού οι προβλέψεις περιέχουν σφάλμα το οποίο μεταδίδεται, η μέθοδος αυτή ενδείκνυται για βραχυπρόθεσμες προβλεψεις.
Πριν προχωρήσουμε στην ανάλυσή των ολοκληρωμένων αυτοπαλινδρομικών μοντέλων κινητών μέσων όρων θα αναφερθούμε σε κάποιες απαραίτητες έννοιες. Μια χρονοσειρά όνομάζεται *στάσιμη* όταν όλα τα πιθανοθεωρητικά χαρακτηριστικά της, δηλαδή η μέση τιμή, η διακύμανση και η αυτοσυνδιακύμανση, παραμένουν αναλλοίωτα στο χρόνο. Πρόκειται για μία ιδιότητα που δεν εξαρτάται από την χρονική περίοδο που παρατηρούμε την χρονοσειρά. Μια χρονοσειρά με σαφή τάση και έντονη εποχιακότητα δεν είναι στάσιμη. Αν μία χρονοσειρά δεν είναι στάσιμη μπορούμε να την μετατρέψουμε σε τέτοια αρχικά λογαριθμίζοντας τις τιμές της και ύστερα αν είναι απαραίτητο με διαφόριση πρώτου ή παραπάνω βαθμού. Η διαφόριση γίνεται παίρνοντας διαδοχικές πρώτες διαφορές μεταξύ των παρατηρήσεων. Αν η χρονοσειρά είναι έντονα εποχιακή τότε μπορούμε να κάνουμε εποχιακή διαφόριση δηλαδή κάθε φορά να παίρνουμε την διαφορά από την παρατήρηση της αντίστοιχης περιόδου του προηγούμενου έτους. Αν μετά την πρώτη διαφόριση η χρονοσειρά δεν γίνει στάσιμη μπορούμε να εφαρμόσουμε ξανά διαφόριση επί του αποτελέσματος της πρώτης διαφόρισης, δηλαδή να κάνουμε διαφόριση δευτέρου βαθμού κ.ο.κ. Ο *συντελεστής μερικής αυτοσυσχέτισης* (PACF) δείχνει κατά πόσο σχετίζεται η τιμή της χρονοσειράς από μια παρελθοντική της τιμή χωρίς να λαμβάνεται υπόψιν η επίδραση των ενδιάμεσων τιμών. Η τιμή του υπολογίζεται

από τις παρακάτω σχέσεις (τον συμβολίζουμε με φ και συμβολίζουμε τους συντελεστές αυτοσυσχέτισης ρ):

$$\varphi_{11} = \rho_1$$

$$\varphi_{kk} = \rho_k - \sum_{j=1}^{k-1} \frac{\varphi_{\kappa-1,j}\rho_{k-j}}{1 - \sum_{j=1}^{k-1} \varphi_{k-1,j}\rho_j},\ k = 2, 3, 4, ...$$

$$\varphi_{kj} = \varphi_{k-1,j} - \varphi_{kk}\varphi_{k-1,k-j}, k = 3, 4, ...\ j = 1, 2, ...$$

Ορίζουμε τον *τελεστή ολίσθησης* ως εξής:

$$By_t = y_{t-1}$$

$$B(By_t) = B^2 y_t = y_{t-2}$$

Ορίζουμε την *προσδοκώμενη πιθανοφάνεια* για ένα μοντέλο ως έναν δείκτη για το πόσο πιθανό είναι οι τιμές του μοντέλου να προσεγγίζουν τις πραγματικές τιμές της χρονοσειράς. Έχουμε:

$$L = \prod_{t=1}^{T} (\frac{1}{2\pi\sigma_t^2})^{1/2} e^{-\sum_{t=1}^{T} (\frac{(X_t - F(X_t))^2}{2\sigma t^2})}$$

ή έπειτα από λογαρίθμηση:

$$-2logL = n[log(2\pi) + 1 + log(\frac{RSS}{n})]$$

,όπου L η προσδοκόμενη πιθανοφάνεια ταύτισης του μοντέλου με τα αρχικά δεδομένα, $F(X_t)$ η πρόβλεψη του μοντέλου για την περίοδο t, n ο αριθμός των παρατηρήσεων, e_t το σφαλμα πρόβλεψης, σ^2 η διακύμανση των σφαλμάτων του μοντέλου και RSS το άθροισμα των τετραγωνικών σφαλμάτων του μοντέλου. Βασικό προαπαιτούμενο για να εφαρμοστεί το μοντέλο ARIMA είναι η στασιμότητα της χρονοσειράς μας. Υπάρχουν διάφορα τεστ για τον έλεγχο της στασιμότητας, όπως το Augmented Dickey-Fuller (ADF) τεστ και το Kwiatkowski–Phillips–Schmidt–Shin (KPSS) τεστ.
Τα μοντέλα ARIMA αποτελούνται από ένα μοντέλο αυτοπαλινδρόμησης (Autoregressive Model) τάξης p, το οποίο συμβολίζεται AR(p), από ένα μοντέλο κινητού μέσου όρου (Moving Average Model) τάξης q, το οποίο συμβολίζεται με MA(q) και ένα μοντέλο διαφόρισης τάξης d. Θα λέμε πως έχουμε μοντέλο ARIMA(p,d,q).
Το μοντέλο αυτοπαλινδρόμησης θεωρεί γραμμική σχέση μεταξύ της τιμής της χρονοσειράς για κάποια χρονική στιγμή t και σε ένα πλήθος παρελθοντικών τιμών της. Για μοντέλο τάξης p, το παραπάνω αποτυπώνεται αλγεβρικά ως εξής:

$$y_t = c + \varphi_1 y_{t-1} + \varphi_2 y_{t-2} + ... + \varphi_p y_{t-p} + e_t$$

Όπου φ οι συντελεστές αυτοσυσχέτισης του μοντέλου AR για την αντίστοιχη χρονική υστέρηση και c μια σταθερά. Για την σταθερά μπορούμε να γράψουμε ότι:

$$c = \mu(1 - \varphi_1 - \varphi_2 - ... - \varphi_p)$$

Όπου μ η μέση τιμή της χρονοσειράς. Αν θέσουμε $\overline{y_t} = y_t - \mu$ και χρησιμοποιήσουμε τον τελεστή ολίσθησης μπορούμε να γράψουμε την παραπάνω σχέση ως εξής:

$$(1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p)\overline{y_t} = e_t$$

Το μοντέλο κινητού μέσου όρου θεωρεί γραμμική σχέση μεταξύ της τιμής της χρονοσειράς για κάποια χρονική στιγμή t και σε ένα πλήθος σφαλμάτων του μοντέλου για το παρελθόν.[57] Για μοντέλο τάξης q, το παραπάνω αποτυπώνεται αλγεβρικά ως εξής:

$$y_t = c - \theta_1 e_{t-1} - \theta_2 e_{t-2} - ... - \theta_q e_{t-q} + e_t$$

Όπου ϑ οι συντελεστές μερικής αυτοσυσχέτισης του μοντέλου MA για την αντίστοιχη χρονική υστέρηση και c μια σταθερά. Για την σταθερά μπορούμε να γράψουμε ότι:

$$c = \mu$$

Όπου μ η μέση τιμή της χρονοσειράς. Αν ϑέσουμε $\overline{y_t} = y_t - \mu$ και χρησιμοποιήσουμε τον τελεστή ολίσθησης μπορούμε να γράψουμε την παραπάνω σχέση ως εξής:

$$\overline{y_t} = (1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q)e_t$$

Σε κάθε περίπτωση τονίζεται πως στα μοντέλα MA και AR δεν χρησιμοποιούμε αυτούσιους τους συντελεστές ACF και PACF της χρονοσειράς αλλά προσεγγίσεις αυτών.
Συνδυάζοντας τα παραπάνω δύο μοντέλα μαζί με ένα μοντέλο διαφόρισης προκύπτουν τα μοντέλα ARIMA για την πρόβλεψη και ανάλυση χρονοσειρών. Το συνολικό μοντέλο ARIMA(p,d,q) κάνοντας χρήση του τελεστή ολισθησης περιγράφεται από την παρακάτω εξίσωση[57]:

$$(1 - \varphi_1 B - \varphi_2 B^2 - ... - \varphi_p B^p)(1 - B)^n (1 - B^m)^N y_t = c + (1 - \theta_1 B - \theta_2 B^2 - ... - \theta_q B^q)e_t$$

Όπως μπορεί κανείς εύκολα να παρατηρήσει, η εξίσωση περιέχει έναν όρο για κάθεμια συνιστώσα του μοντέλου ARIMA. Στο κομμάτι της διαφόρισης, δηλαδή ο δεύτερος όρος του πρώτου μέρους, το B^m αντιπροσωπεύει την εποχιακή διαφόριση, ενώ οι εκθέτες ν,N δείχνουν τον βαθμό απλής και εποχιακής διαφόρισης. Για την σταθερά στο μοντέλο ARIMA έχουμε:

$$c = \mu(1 - \varphi_1 - \varphi_2 - ... - \varphi_p, \ \gamma\iota\alpha \ N = n = 0$$

$$c = 0, \ \gamma\iota\alpha \ N, n \neq 0$$

Η σημαντικότερη επιλογή που έχουμε να κάνουμε για τα μοντέλα ARIMA είναι αυτή των παραμέτρων p,d και q. Για αυτήν υπάρχουν τεχνικές που αφορούν την μορφή των διαγραμμάτων των συντελεστών ACF και PACF, αλλά και κάποια στατιστικού τύπου κριτήρια. Τα πιό γνωστά είναι το *Akaike's Information Criterion* (AIC) και το *Bayesian Information Criterion* (BIC). Τα κριτήρια αυτά αξιολογούν κατά πόσο ταιριάζει το εξεταζόμενο μοντέλο στην χρονοσειρά συναρτήσει της πολυπλοκότητάς του. Γενικά, ισχύει ότι με την αύξηση της πολυπλοκότητας μειώνεται η προκατάληψη των μοντέλων, με τον κίνδυνο όμως της υπερπροσαρμογής και συνεπώς της χαμηλής απόδοσης. Τα δύο κριτήρια υπολογίζονται μέσω της μέγιστης πιθανοφάνειας, δηλαδή βλέπουν πόσο κοντά στα πραγματικά δεδομένα κινούνται τα μοντέλα μας. Ως βέλτιστο μοντέλο παίρνουμε αυτό που ελαχιστοποιεί την τιμή του εκάστοτε κριτηρίου. Πέραν αυτών των δύο χρησιμοποιούμε και το AIC_c, που είναι παραλλαγή του AIC, αν ϑέλουμε να δώσουμε μεγαλύτερο βάρος στην πολυπλοκότητα του μοντέλου. Οι τιμές των κριτηρίων για τις τρεις περιπτώσεις είναι οι εξής:

$$AIC = -2logL + 2(p + q + k + 1)$$

, με k=0 για c=0 και k=1 διαφορετικά.

$$AIC_c = AIC + \frac{2(p + q + k + 1)(p + q + k + 2)}{n - p - q - k - 2}$$

και

$$BIC = AIC + log(n)(p + q + k + 1)$$

Η πρόβλεψη γίνεται αντικαθιστώντας τον δείκτη t στην εξίσωση του μοντέλου με T+h, με το T το πλήθος των γνωστών δεδομένων και το h να παίρνει τιμή ανάλογως της χρονικής περίοδου προς πρόβλεψη. Τα σφάλματα που αφορούν μελλοντικές τιμές ϑεωρούνται μηδενικά. Για να είναι κατάλληλο ένα μοντέλο πρέπει να είναι στατιστικά σημαντικό. Έτσι

εφαρμόζουμε διαγνωστικά στατιστικά test, τα τεστ τύπου t,ώστε να δούμε κατά πόσο συσχετίζονται τα σφάλματα του μοντέλου.
Υπάρχει μία παραλλαγή των μεθόδων ARIMA για την μοντελοποίση δεδομένων με έντονη εποχιακότητα, τα μοντέλα SARIMA (Seasonal ARIMA). Η διαφορά τους με τα απλά μοντέλα ARIMA έγκειται στη ύπαρξη επιπλέον εποχιακών όρων που προστίθενται στην εξίσωση που περιγράφει το κάθε μοντέλο. Πιο συγκεκριμένα, στον κάθε όρο της εξίσωσης πολλαπλασιάζουμε με τον αντίστοιχο εποχιακό. Γράφουμε τα μοντέλα μας με την μορφή ARIMA(p,d,q) $(P,D,Q)_m$, όπου P,D,Q οι εποχιακές τάξεις και m το πλήθος των περιόδων αν έτος (π.χ. 12 για ωριαία δεδομένα)[25]. Για παράδειγμα, αν έχουμε μοντέλο ARIMA(1,1,1) $(1,1,1)_4$ έχουμε την εξής εξίσωση:

$$(1-\varphi_1 B)(1-\Phi_1 B^4)(1-B)(1-B^4)y_t = (1-\theta_1 B)(1-\Theta_1 B^4)e_t$$

Όσον αφορά τον καθορισμό των παραμέτρων ισχύουν όσα περιγράφηκαν παραπάνω, με την διαφορά πως για τους εποχιακούς παράγοντες πάντοτε αναφερόμαστε στις τιμές με καθυστέρηση ίση με το πλήθος των περιόδων του έτους. Σημειώνεται ότι το D αναφέρεται σε εποχιακή παραγώγιση ενώ το d σε απλή.
Τέλος, είναι σημαντικό να σημειώσουμε ότι στην παρούσα παράγραφο αναφέραμε κάποιες από τις σημαντικότερες στατιστικές μεθόδους, αλλά όχι όλες. Υπάρχουν αρκετές ακόμα ενδιαφέρουσες μέθοδοι, όπως η *παλινδρόμηση*, κατά την οποία εξετάζουμε την μεταβολή της τιμής της χρονοσειράς (εξαρτημένη μεταβλητή) βάσει της μεταβολής μίας ή περισσότερων ανεξάρτητων μεταβλητών (απλή ή πολλαπλή παλινδρόμηση). Η πιό γνωστή μέθοδος παλινδρόμησης είναι η *απλή γραμμική παλινδρόμηση*, όπου υποθέτουμε γραμμική σχέση μεταξύ (συνήθως) του χρόνου και της χρονοσειράς μας και αναζητούμε την εξίσωση της ευθείας που περιγράφει την σχέση αυτή βέλτιστα, κινούμενοι στην λογική ελαχιστοποίησης του αθροίσματος των τετραγώνων των διαφορών των πραγματικών τιμών της χρονοσειράς από τις τιμές που προκύπτουν από την εξίσωση παλινδρόμησης (που έχει την μορφή $Y_i = a + b\ X_i$). Μία άλλη γνωστή μέθοδος είναι *μέθοδος Θ* (Theta), όπου μέσω της παραμέτρου ϑ αποσυνθέτουμε την χρονοσειρά σε δύο ή περισσότερες γραμμές Theta τις οποίες προεκτείνουμε με κάποια μέθοδο πρόβλεψης ή αυτούσιες. Ως πρόβλεψη παίρνουμε τον συνδυασμό των προβλέψεων από τις γραμμές. Οι γραμμές, βάσει της τιμής της παραμέτρου ϑ μπορούν να προσεγγίζουν την μακροπρόθεσμη τάση της χρονοσειράς ή τα βραχυπρόθεσμα χαρακτηριστικά της. Τέλος, θα κάνουμε μία σύντομη αναφορά στην μέθοδο *Facebook Prophet*. Η μέθοδος αυτή είναι μια μέθοδος αποσύνθεσης που ακολουθεί το προσθετικό μοντέλο. Πιο συγκεκριμένα, διακρίνει τα χαρακτηριστικά της τάσης, της εποχιακότητας και των ημερολογιακών προσαρμογών, δηλαδή του πως επηρεάζεται η χρονοσειρά από ειδικά ημερολογικά γεγονότα (π.χ. εορτές). Έχει την δυνατότητα να μοντελοποιεί πολλαπλές εποχιακότητες (π.χ. ημερήσια και εβδομαδιαία), ενώ χαρακτηρίζεται από ταχύτητα και προσαρμοστικότητα.[46]

2.3 Μέτρα απόδοσης προβλέψεων

Ιδιαίτερα σημαντικό είναι να ορίσουμε κάποια μέτρα με τα οποία θα αξιολογούμε τις προβλέψεις μας. Πριν προχωρήσουμε στην παρουσίαση των σημαντικότερων στατιστικών δεικτών που υπάρχουν για τον σκοπό αυτό, θα κάνουμε έναν διαχωρισμό στις τιμές που προβλέψαμε με το όποιο μοντέλο μας. Οι προβλέψεις μας χωρίζονται στην προσαρμογή του μοντέλου, για τις οποίες υπάρχουν διαθέσιμες και οι αντίστοιχες παρατηρήσεις μας και τις μελλοντικές προβλέψεις για τις οποίες δεν έχουμε ακόμα τα πραγματικά δεδομένα. Τους δείκτες αυτούς τους ονομάζουμε σφάλματα και στόχος μας είναι να ελαχιστοποιήσουμε την απόλυτη τιμή τους.
Ορίζουμε ως σφάλμα την διαφορά μεταξύ πραγματικής τιμής και πρόβλεψης για την ίδια χρονική περίοδο, δηλαδή έχουμε:

$$e_i = Y_i - F_i$$

Στην λογική αυτή έχουμε το σφάλμα του μοντέλου πρόβλεψης (in-sample error) που αφορά την προσαρμογή του μοντέλου μας και το πραγματικό σφάλμα (out-of-sample error), όταν γίνουν διαθέσιμα τα δεδομένα που προβλέψαμε. Μπορούμε να χωρίσουμε τα σφάλματα σε τρεις κατηγορίες, αυτά που εξαρτώνται από την κλίμακα των δεδομένων (Μέσο Σφάλμα, Μέσο Απόλυτο Σφάλμα, Μέσο Τετραγωνικό Σφάλμα και Ρίζα Μέσου Τετραγωνικού Σφάλματος), τα ποσοστιαία σφάλματα, που είναι ανεξάρτητα της κλίμακας (Μέσο Απόλυτο Ποσοστιαίο Σφάλμα και Μέσο Απόλυτο Ποσοστιαίο Σφάλμα) και τα σχετικά σφάλματα (Theil's U Statistic, Relative Measures και Μέσο Απόλυτο Κανονικοποιημένο Σφάλμα), όπου συγκρίνουμε το σφάλμα της μεθόδου μας με αυτό μίας άλλης μεθόδου. Στην συνέχεια θα παρουσιάσουμε κάθε ένα από τα παραπάνω σφάλματα.
Ο πρώτος δείκτης που θα παρουσιάσουμε είναι το μέσο σφάλμα (Mean Error-ME) γνωστό και ως bias (προκατάληψη). Αποτελεί τον απλό προσημασμένο μέσο όρο των σφαλμάτων. Θετικές τιμές του δηλώνουν απαισιοδοξία του μοντέλου και αρνητικές αισιοδοξία. Υπολογίζεται με τον εξής τύπο:

$$ME = \frac{1}{n}\sum_{i=1}^{n}(Y_i - F_i)$$

Το μέσο απόλυτο σφάλμα (Mean Absolute Error-MAE) δηλώνει ένα μέσο μέτρο της αστοχίας της πρόβλεψης χωρίς να δίνεται έμφαση στην κατεύθυνση της πρόβλεψης.[56] Επομένως, λαμβάνουμε μια μη αρνητική τιμή είτε το μοντέλο μας προβλέπει συστηματικά χαμηλότερη τιμή της πραγματικής είτε υψηλότερη. Για τον υπολογισμό του έχουμε τον κάτωθι τύπο:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - F_i|$$

Το μέσο τετραγωνικό σφάλμα (Mean Squared Error-MSE) είναι ακόμη ένα σφάλμα εξαρτώμενο από τις μονάδες των δεδομένων. Σε αντίθεση με το απλό σφάλμα, το ME και το MAE που είναι εκφρασμένα στις μονάδες των δεδομένων το MSE εκφράζεται στο τετράγωνο της μονάδας τους. Το MSE δίνει, λόγω της ύψωσης στο τετράγωνο, μεγαλύτερο βάρος στα μεγάλα σφάλματα και μικρότερο βάρος στα μικρά σφάλματα. Προκύπτει ως εξής:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - F_i)^2$$

Η ρίζα μέσου τετραγωνικού σφάλματος (Root Mean Squared Error-RMSE) προκύπτει από το MSE αν πάρουμε την τετραγωνική του ρίζα. Οι ιδιότητες του ταυτίζονται με την διαφορά ότι αυτό είναι εκφρασμένο στις μονάδες της χρονοσειράς. Υπολογίζεται σύμφωνα με τον ακόλουθο τύπο:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(Y_i - F_i)^2}$$

Εισερχόμαστε στην κατηγορία των ποσοστιαίων σφαλμάτων. Τα σφάλματα αυτά είναι σε ποσοστιαία μορφή και ως εκ τούτου ανεξάρτητα των μονάδων των χρονοσειρών. Έτσι μπορούμε να συγκρίνουμε την ακρίβεια μοντέλων επί διαφορετικών ειδών χρονοσειρών. Το πρώτο μέτρο της κατηγορίας είναι το μέσο απόλυτο ποσοστιαίο σφάλμα (Mean Absolute Percentage Error-MAPE). Το σφάλμα αυτό δεν μπορεί να υπολογιστεί σε χρονοσειρές με μηδενικές παρατηρήσεις γιατί περιέχει τις τιμές τους στον παρονομαστή. Επίσης, η μετρική αυτή τείνει να δίνει μεγαλύτερη ποινή στα αρνητικά σφάλματα λόγω της μη ύπαρξης άνω

ορίου σε αντίθεση με τα θετικά σφάλματα που περιορίζονται στο 100%. Για τον υπολογισμό του, έχουμε:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{Y_i - F_i}{Y_i}|100\%$$

Μία παραλλαγή του MAPE, είναι το συμμετρικό μέσο απόλυτο ποσοστιαίο σφάλμα (Symmetric Mean Absolute Percentage Error-sMAPE). Η διαφορά τους βρίσκεται στον παρονομαστή, όπου αντί για την ίδια την παρατήρηση έχουμε το ημιάθροισμα της με την πρόβλεψη. Έτσι, ο δείκτης αποκτά και πάνω όριο και πλέον μπορεί να πάρει τιμές στο διάστημα [0%, 200%]. Ωστόσο, ο δείκτης sMAPE δεν μεταχειρίζεται με τον ίδιο τρόπο τις αισιόδοξες και απαισιόδοξες προβλέψεις (π.χ. για πραγματική τιμή 1000 και προβλέψεις 900 και 1100 δίνει διαφορετικά αποτελέσματα). Το sMAPE υπολογίζεται βάσει του παρακάτω τύπου:

$$sMAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{Y_i - F_i}{(\frac{Y_i+F_i}{2})}|100\% = \frac{1}{n}\sum_{i=1}^{n}|\frac{2(Y_i - F_i)}{Y_i + F_i}|100\%$$

Όταν έχουμε περισσότερα του ενός μοντέλα πρόβλεψης, μπορούμε να συγκρίνουμε την απόδοση τους αφού πρώτα υπολογίσουμε κάποιον κοινό δείκτη σφάλματος για το κάθε ένα από αυτά. Διαλέγουμε μία μέθοδο αναφοράς (benchmark) και υπολογίζουμε το σχετικό σφάλμα διαιρώντας το σφάλμα της μεθόδου μας με το σφάλμα της μεθόδου αναφοράς. Πιό συγκεκριμένα, έχουμε:

$$RelError = \frac{Error_i}{Error_{benchmark}}$$

Με τον τρόπο αυτό μπορούμε να δούμε κατά πόσο είναι καλύτερη ή χειρότερη η μέθοδος μας από την μέθοδο αναφοράς. Συνήθως, χρησιμοποιούμε ως μέθοδο αναφοράς μια απλή μέθοδο. Τονίζεται ότι ως μετρική error μπορεί να χρησιμοποιηθεί οποιαδήποτε από τις γνωστές (π.χ. MAE ή MAPE).
Ένας άλλος δείκτης ακρίβειας είναι το μέσο απόλυτο κανονικοποιημένο σφάλμα (Mean Absolute Scaled Error-MAsE). Με το μέτρο αυτό επιλύεται το ζήτημα της απροσδιοριστίας και της ανισότητας της σημασίας των μικρών και μεγάλων λαθών. Μοναδική περίπτωση απροσδιοριστίας είναι όταν έχουμε μια χρονοσειρά με σταθερή τιμή. Σε περίπτωση που υπολογιστεί τιμή μικρότερη της μονάδας τότε η μέθοδος μας έχει κατά μέσο όρο καλύτερη απόδοση απ'την Naive, ενώ το αντίθετο συμβαίνει για τιμή μεγαλύτερη της μονάδας. Για τον υπολογισμό του έχουμε:

$$MAsE = \frac{\frac{1}{n}\sum_{i=1}^{n}|Y_i - F_i|}{\frac{1}{n-1}\sum_{i=2}^{n}|Y_i - Y_{i-1}|}$$

Τελευταίος δείκτης σφάλματος είναι ο Theil U Statistic. Σε αυτήν την περίπτωση έχουμε και πάλι ένα σχετικό σφάλμα όπως στο MAsE, καθώς γίνεται άμεση σύγκριση με μία μέθοδο αναφοράς, την Naive, με την διαφορά πως δίνεται μεγαλύτερη βαρύτητα στις μεγαλύτερες αποκλίσεις, καθώς τα σφάλματα τετραγωνίζονται. Για τιμές μεγαλύτερες της μονάδας η μέθοδός μας έχει χειρότερη απόδοση απ'την Naive, ενώ για τιμές μικρότερες της μονάδας το αντίθετο. Όσο απομακρυνόμαστε από την μονάδα τόσο χειρότερη και καλύτερη αντίστοιχα είναι η απόδοση της μεθόδου σε σχέση με την μέθοδο Naive. Υπολογίζουμε τον δείκτη αυτόν ως εξής:

$$U = \sqrt{\frac{\sum_{i=2}^{n}(\frac{Y_i - F_i}{Y_{i-1}})^2}{\sum_{i=2}^{n}(\frac{Y_i - Y_{i-1}}{Y_{i-1}})^2}}$$

Αξίζει να σημειωθεί πως χρησιμοποιώντας διαφορετικά σφάλματα μπορούμε να καταλήξουμε σε διαφορετική κατάταξη των μεθόδων ως προς την απόδοση τους. Το καθένα από τα παραπάνω μέτρα απόδοσης έχει τα θετικά του και τα αρνητικά του. Εμείς στην παρούσα εργασία θα χρησιμοποιήσουμε το μέσο απόλυτο ποσοστιαίο σφάλμα για την μέτρηση της

απόδοσης των μεθόδων πρόβλεψης των χρονοσειρών μας καθώς είναι σχετικά απλό και κατανοητό, μπορεί να γίνει εύκολα αντιληπτή η σημασία του από τον χρήστη (στο κομμάτι της εφαρμογής), ενώ δεν υπάρχει κίνδυνος για απροσδιοριστία καθώς όλες οι τιμές στις χρονοσειρές μας έχουν τιμές μεγαλύτερες του μηδενός.

Κεφάλαιο 3

Αποθήκευση Χρονοσειρών και Λοιπών Δεδομένων

3.1 Αποθήκευση χρονοσειρών σε Βάσεις Δεδομένων

Με τις βάσεις δεδομένων συνηθίζεται να μοντελοποιούμε και να αποθηκεύουμε δεδομένα που αφορούν την τωρινή κατάσταση του κόσμου (π.χ. το προσωπικό μίας εταιρείας). Ωστόσο, πολλές φορές είναι χρήσιμη η αποθήκευση και ανάκληση πληροφοριών και για καταστάσεις του παρελθόντος. Τα τελευταία χρόνια υπάρχει έντονη η ανάγκη για αποθήκευση και χειρισμό χρονολογικών δεδομένων που μπορεί να προέρχονται από διάφορες πηγές όπως αισθητήρες ή ιατρικό ιστορικό ασθενών. Οι βάσεις δεδομένων που αποθηκεύουν πληροφορίες για καταστάσεις της πραγματικότητας στην διάρκεια του χρόνου ονομάζονται *χρονολογικές βάσεις δεδομένων*. Στις βάσεις δεδομένων υπάρχει διάκριση μεταξύ του χρόνου του συστήματος όπου το γεγονός είναι τρέχον στην βάση δεδομένων και του πραγματικού χρόνου παρατήρησης του γεγονότος. Ο πρώτος ονομάζεται χρόνος συναλλαγής , δημιουργείται αυτόματα και βασίζεται στην σειριοποιησιμότητα της συναλλαγής. Ο δεύτερος ονομάζεται έγκυρος χρόνος και πρέπει να τον παρέχουμε εμείς στο σύστημα (άρα και να τον μετρήσουμε με ακρίβεια).
Το πρότυπο SQL ορίζει τους τύπους date (YYYY-MM-DD - π.χ. 2011-09-22), time (hh:mm:ss[.nnnnnnn] πχ 21:39:21.9999945) και timestamp με πεδία date και time για την αποθήκευση δεδομένων χρόνου. Δεδομένη είναι η ανάγκη καθορισμού της ζώνης ώρας μαζί με τον χρόνο. Το UTC (Universal Coordinated Time) είναι ένα τυπικό σημείο αναφοράς για τον καθορισμό του χρόνου με τους τοπικούς χρόνους να ορίζονται ως μετατοπίσεις από το αυτό. Η SQL περιλαμβάνει τους τύπους time with time zone και timestamp with time zone, που καθορίζουν την ώρα ως τοπικό χρόνο μαζί με την μετατόπιση της τοπικής ώρας απ'την ώρα UTC[39] (π.χ. -6 για την ανατολική ώρα στις ΗΠΑ).
Στην βάση δεδομένων MongoDB μπορούμε να αποθηκεύσουμε χρονολογικά δεδομένα με τον BSON τύπο Date, που πρόκειται για έναν 64bit integer που αναπαριστά τον αριθμό των milliseconds απ'την Unix εποχή. Αν έχουμε ημερομηνίες πριν από 00:00:00 UTC της 1ης Ιανουαρίου 1970 ο αριθμός είναι αρνητικός, αλλιώς είναι θετικός. Η MongoDB αποθηκεύει τις ημερομηνίες σε ώρα UTC χωρίς μετατόπιση ώρας. Υπάρχει επίσης ο τύπος timestamp που γενικά χρησιμοποιείται εσωτερικά στην βάση.[7] Πέραν όλων αυτών μπορούμε να αποθηκεύσουμε την πληροφορία του χρόνου και ως string κάτι όμως που δεν προκρίνεται για λόγους απόδοσης.
Ως βάση δεδομένων για χρονοσειρές θεωρούμε τον τύπο των βάσεων δεδομένων που είναι σχεδιασμένες και βελτιστοποιημένες για χειρισμό δεδομένων που συνοδεύονται από πεδίο χρόνου (χρονοσφραγίδα). Τέτοιες βάσεις επιτρέπουν στους χρήστες να δημιουργούν, ανανεώνουν, καταστρέφουν και να οργανώνουν διάφορες χρονοσειρές με έναν πιο αποδοτικό τρόπο σε σχέση με τις συνηθισμένες βάσεις δεδομένων. Η βασική διαφορά των δεδομένων που περιέχουν με τα συνηθισμένα δεδομένα είναι πως τα ερωτήματα που θέτουμε επί αυτών σχετίζονται κυρίως με τον χρόνο. Η έκρηξη στην παραγωγή χρονικών μετρήσεων των τελευταίων δεκαετιών καθιστά τις βάσεις αυτού του τύπου αν όχι απαραίτητες σίγουρα ιδιαίτερα σημαντικές. Οι βάσεις αυτές βρίσκουν εφαρμογή σε δίκτυα συσκευών που

συλλέγουν και ανταλλάσουν πληροφορία αλλά και στην παρακολούθηση λειτουργιών και στη συλλογή δεδομένων πραγματικού χρόνου με σκοπό την ανάλυση τους και την λήψη αποφάσεων.
Οι βάσεις δεδομένων αυτού του τύπου πρέπει να τοποθετούν μετρήσεις κοντινού χρόνου σε κατά το δυνατόν πιο κοντινό κομμάτι υλικού ώστε να προσφέρουν γρήγορη πρόσβαση σε αυτές και συνεπώς πιο αποδοτική ανάλυση τους. Με τον τρόπο αυτό οι βάσεις εγγυώνται πως τα ερωτήματα χρονικού διαστήματος εκτελούνται γρήγορα. Λόγω της φύσης των ερωτημάτων πρέπει να εξασφαλίζεται μια συμβατή και εύχρηστη γλώσσα ερωτημάτων. Ιδιαίτερα σημαντικό είναι λόγω του μεγάλου φόρτου εργασίας να γίνονται αποδοτικά οι εργασίες εγγραφής και ανάγνωσης της βάσης (η συχνότητα εγγραφής μετρήσεων είναι συνήθως επιπέδου δευτερολέπτου και μεγαλύτερη). Ένα άλλο ζήτημα που πρέπει να λαμβάνεται υπόψιν είναι η κλιμακωσιμότητα καθότι η ποσότητα των δεδομένων αυξάνεται με πολύ μεγάλους ρυθμούς (μεγάλη συχνότητα εγγραφής μετρήσεων). Τέλος, πρέπει να υπάρχει πρόβλεψη για συμπίεση δεδομένων, λειτουργίες σχετικές με ανάλυση χρονοσειρών και την διατήρηση των δεδομένων.[31]
Τα τελευταία χρόνια η εκρηκτική ανάπτυξη του Internet of Things αλλά και της real time data analysis σε συνδυασμό με τα πλεονεκτήματα απόδοσης και καλής κλιμακωσιμότητας που προσφέρουν οι βάσεις χρονοσειρών έχει οδηγήσει σε αύξηση της δημοφιλίας τους. Υπάρχουν πολλές βάσεις αυτού του τύπου. Οι πέντε πιο δημοφιλείς με φθίνουσα σειρά κατάταξης, σύμφωνα με την ιστοσελίδα db-engines.com, η οποία κατατάσσει τις βάσεις σύμφωνα με τις αναφορές στα κοινωνικά δίκτυα, τις αναζήτησεις στο διαδίκτυο, τη συχνότητα των τεχνικών συζητήσεων στο διαδίκτυο και τις σχετικές προσφορές εργασίας, είναι οι InfluxDB, Kdb+, Prometheus, Graphite και TimescaleDB. Σημειώνεται πως η κατάταξη αυτή ισχύει για τον Μάιο του 2022.
Η *InfluxDB* είναι μια μη σχεσιακή βάση δεδομένων χρονοσειρών ανοιχτού κώδικα γραμμένη στη γλώσσα προγραμματισμού Go. Η InfluxDB περιλαμβάνει την InfluxQL, που αποτελεί μία γλώσσα ερωτημάτων παρόμοια με την SQL. Η InfluxDB αποτελεί κομμάτι της λεγόμενης TICK στοίβας (Telegraf – InfluxDB – Chronograf – Kapacitor). Πριν προχωρήσουμε στην ανάλυση της, θα κάνουμε μια αναφορά στα υπόλοιπα συστατικά μέρη της στοίβας TICK.
Το *Telegraf* είναι ένας server agent που βασίζεται σε plugins, για την συλλογή, αναφορά και καταγραφή μετρήσεων. Με τον όρο plugin (πρόσθετο πρόγραμμα) εννοούμε ένα κομμάτι λογισμικού που προσθέτει νέες λειτουργίες σε ένα host πρόγραμμα χωρίς όμως να το αλλοιώνει, δηλαδή απλώς το επεκτείνει. Λέγοντας agent (πράκτορας) εννοούμε μία διεργασία που τρέχει στο παρασκήνιο και ξεκινάει από τον χρήστη (άμεσα ή έμμεσα). Ο πράκτορας τυπικά απαιτεί είσοδο από τον χρήστη. Το Telegraf λαμβάνει από τον χρήστη ως είσοδο ένα αρχείο διαμόρφωσης (configuration file) ώστε να συλλέξει μετρήσεις από συγκεκριμένα plugins εισόδου και να τις στείλει σε συγκεκριμένα plugins εξόδου. Το ποια θα είναι τα plugins εισόδου και εξόδου καθορίζεται από το αρχείο διαμόρφωσης. Τα plugins εισόδου του Telegraf, με την σειρά τους μπορεί να λαμβάνουν μετρήσεις απευθείας από το σύστημα στο οποίο τρέχουν, από API τρίτων μερών ή και μέσω διαφόρων υπηρεσιών πελάτη (πχ Apache Kafka). Τα plugins εξόδου στέλνουν τις μετρήσεις σε μία ποικιλία χώρων αποθήκευσης, υπηρεσιών και ουρών μηνυμάτων όπως η InfluxDB, το Graphite, το MQTT κ.ά.
Το *Chronograf* αποτελεί το γραφικό περιβάλλον της στοίβας για την εκτέλεση λειτουργιών διαχείρησης και την προβολή των δεδομένων της βάσης. Ειδικότερα, μέσω αυτού ρυθμίζουμε την παρακολούθηση και τις ειδοποιήσεις σχετικά με το σύστημά μας. Παράλληλα, μας παρέχει μια οπτική επαφή με τα δεδομένων μέσω διαφόρων γραφημάτων.
Το *Kapacitor* είναι μία μηχανή επεξεργασίας δεδομένων για την InfluxDB, είτε πρόκειται για δεδομένα ροής είτε για δέσμες δεδομένων. Το Kapacitor μας επιτρέπει να τρέξουμε την δική μας συγκεκριμένη λογική ή και ορισμένες από τον χρήστη συναρτήσεις για την πραγματοποίηση διάφορων λειτουργιών.[33]
Στην InfluxDB το βασικό δομικό στοιχείο είναι τα measurements που αντιστοιχούν στους πίνακες της SQL και περιέχουν τις παρατηρήσεις μας. Μια βάση δεδομένων (database)

μπορεί να περιέχει μια ή περισσότερες measurements. Οι παρατηρήσεις ονομάζονται points και αντιστοιχούν στις εγγραφές της SQL. Κάθε παρατήρηση περιλαμβάνει το measurement που ανήκει, το σύνολο ετικετών του (tag set), το σύνολο πεδίων του (field set) και μια χρονοσφραγίδα (timestamp). Ως σύνολο ετικετών ορίζουμε το σύνολο ζευγών κλειδιού-τιμής ετικέτας (tag key - tag value). Κάθε τέτοιο ζευγάρι ονομάζεται ετικέτα (tag). Οι παρατηρήσεις μπορούν να έχουν καμμία, μία ή περισσότερες ετικέτες. Ως σύνολο πεδίων ορίζουμε το σύνολο ζευγών κλειδιού-τιμής πεδίου (field key - field value). Κάθε τέτοιο ζευγάρι ονομάζεται πεδίο. Οι παρατηρήσεις πρέπει να έχουν τουλάχιστον ένα πεδίο. Οι τιμές πεδίου είναι η πραγματική πληροφορία που καταγράφουμε και μπορούν να παίρνουν οποιοδήποτε υποστηριζόμενο τύπο δεδομένων. Στις τιμές ετικετών συνήθως τοποθετούμε χρήσιμα μεταδεδομένα με τύπο δεδομένων αυστηρά string. Μια σημαντική σημείωση είναι πως επί των ετικετών υπάρχουν ευρετήρια, άρα τα ερωτήματα επί αυτών είναι πιο αποδοτικά σε σχέση με τα αντίστοιχα επί των πεδίων. Ως σειρά (series) αναφερόμαστε στο σύνολο δεδομένων με κοινά measurement, σύνολα ετικετών και κλειδιά πεδίων. Κάθε point προσδιορίζεται μοναδικά από το timestamp του και την σειρά του.

Στην InfluxDB μπορούμε να ορίσουμε την πολιτική διατήρησης των δεδομένων (retention policy). Η πολιτική αυτή περιγράφει το χρονικό διάστημα (duration) που η βάση διατηρεί τα δεδομένα, πόσα αντίγραφά τους αποθηκεύει (replication factor) και το χρονικό εύρος που καλύπτεται από τις ομάδες τεμαχίων (shard groups duration). Η πολιτική αυτή είναι μοναδική ανά βάση δεδομένων. Με την δημιουργία μίας βάσης δεδομένων η InfluxDB αυτομάτως επιλέγει ως πολιτική την λεγόμενη autogen σύμφωνα με την οποία το διάστημα διατήρησης των δεδομένων είναι άπειρο, το πλήθος των αντιγράφων ίσο με ένα και το χρονικό εύρος των ομάδων τεμαχίων επτά μέρες. Φυσικά μπορούμε να δημιουργήσουμε την δική μας πολιτική.[26]

Η InfluxDB πραγματοποιεί μια οριζόντια διαίρεση των δεδομένων σε τεμάχια (shards) τα οποία αποθηκεύει εσωτερικά σε ομάδες τεμαχίων (shard groups).[31] Κάθε τεμάχιο ανήκει σε μια μόνο ομάδα τεμαχίων. Οι παρατηρήσεις εντός της ίδιας ομάδας τεμαχίων αποθηκεύονται στην ίδια τοποθεσία στον δίσκο. Σε κάθε ομάδα τεμαχίων αποθηκεύονται δεδομένα των οποίων η χρονοσφραγίδα βρίσκεται εντός ενός χρονικού εύρους που ορίζεται από την πολιτική διατήρησης. Για την επιλογή του κατάλληλου χρονικού εύρους για τις ομάδες τεμαχίων απαιτείται να ληφθεί υπόψιν το γεγονός ότι όσο μεγαλύτερο εύρος τόσο καλύτερη απόδοση έχουμε στην συμπίεση των δεδομένων και σε κάποια ερωτήματα λόγω του μεγαλύτερου πλήθους δεδομένων στην ίδια τοποθεσία, ενώ όσο μικρότερο είναι τόσο πιο αποδοτικά πραγματοποιείται η διαγραφή των δεδομένων. Η InfluxDB διαγράφει τις παρατηρήσεις ανά ομάδα τεμαχίων και όχι το καθένα ξεχωριστά, όταν ο χρόνος λήξης του διαστήματος της ομάδας ξεπεραστεί από τον χρόνο που ορίζει η πολιτική διατήρησης.

Η εισαγωγή δεδομένων σε μία βάση και ειδικότερα σε ένα measurement γίνεται βάσει του line protocol, που ορίζει μια μορφή κειμένου την οποία ακολουθούμε για κάθε εισαγόμενη παρατήρηση. Πιο συγκεκριμένα, για κάθε νέα παρατήρηση εισάγουμε:

```
<measurement>[,<tag set>] <field set> [<timestamp>]
```

Ως measurement αναφέρουμε την συλλογή εντός της οποίας θα τοποθετηθεί η προς εισαγωγή παρατήρηση. Είναι υποχρεωτικό να προσδιορίζεται και πρέπει να βρίσκεται στην αρχή. Ύστερα, αν θέλουμε μπορούμε να προσθέσουμε το σύνολο ετικετών, διαχωρισμένο από τη measurement αυστηρά με ένα κόμμα. Το tag set θα έχει την μορφή:

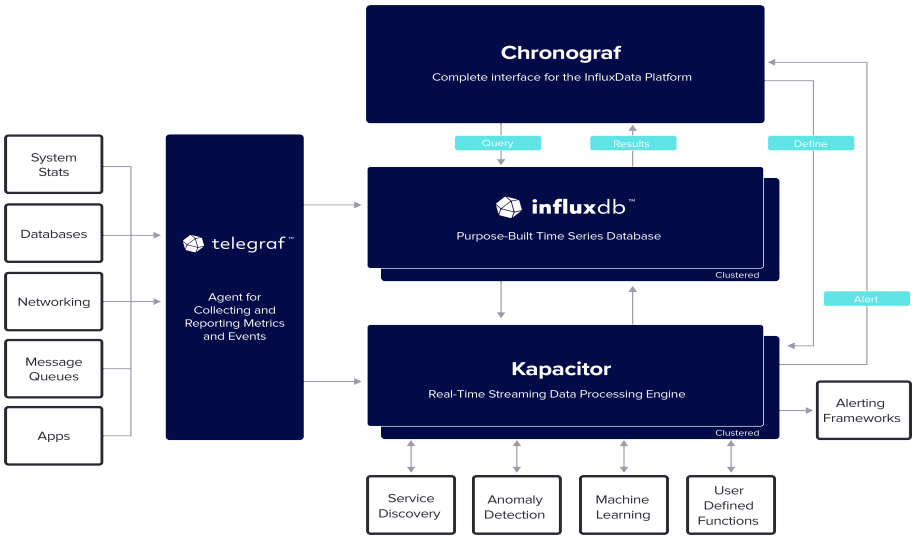
```
<tag key>=<tag value>[,<tag key>=<tag value>]
```

Δηλαδή, μπορούμε να βάλουμε ένα ή περισσότερα ζεύγη κλειδιού-τιμής διαχωρισμένα με ένα κόμμα μεταξύ τους. Οι τιμές ανατίθενται στα κλειδιά με τον τελεστή "=". Μετά το σύνολο ετικετών ή το measurement (αν δεν έχουμε καμμία ετικέτα) αφήνουμε αυστηρά ένα κενό και ύστερα τοποθετούμε το σύνολο πεδίου, που έχει την μορφή:

```
<field key>=<field value>[,<field key>=<field value>]
```

Είναι υποχρεωτικό να εισάγουμε τουλάχιστον ένα ζεύγος πεδίου. Και εδώ τα ζεύγη

χωρίζονται με ένα κόμμα και τα κλειδιά λαμβάνουν τιμή μέσω του τελεστή '='. Τέλος, προαιρετικά μετά από ένα κενό μπορούμε να εισάγουμε την χρονοσφραγίδα μας. Η μορφή της χρονοσφραγίδας είναι Unix nanosecond timestamp. Δηλαδή, πρόκειται για τον αριθμό των nanoseconds που έχουν περάσει από την εποχή UNIX, δηλαδή τις 00:00:00 UTC της 1ης Ιανουαρίου του 1970. Προσθέτοντας τα nanoseconds προκύπτει ο ακριβής χρόνος για την παρατήρησή μας. Αν δεν εισάγουμε χρονοσφραγίδα στην παρατήρηση προσδιορίζεται ως χρονοσφραγίδα ο τοπικός χρόνος του συστήματος σε UTC. Η InfluxDB χειρίζεται εξ'ορισμού τον χρόνο με ακρίβεια nanosecond. Αν θέλουμε μπορούμε να επιλέξουμε διαφορετικό επίπεδο ακρίβειας. Μπορούμε να εισάγουμε παρατηρήσεις μέσω του CLI (Command Line Interface), μέσω Client Libraries ή plugins που συνδέονται με την βάση στα πλαίσια ενός συστήματος.[26]
Έχουμε δημιουργήσει την βάση δεδομένων energy και εντός της το measurement power. Κάθε παρατήρηση έχει δύο tags. Ένα με tag key Building_Name και ένα με tag key Device_Serial. Ως τιμή ετικέτας έχουν το αντίστοιχο όνομα κτηρίου και αριθμό συσκευής που πραγματοποίησε την μέτρηση. Στην περίπτωση μας όλες οι παρατηρήσεις έχουν ως Building_Name το "PC-LAB" και ως Device_Serial το "102.XXX.XXXXX". Έχουμε τρία πεδία για κάθε παρατήρηση. Τα κλειδιά πεδίου είναι pwrA, pwrB, pwrC και αντιστοιχούν στις τρείς φάσεις της γραμμής μας. Ως τιμή πεδίου κάθε παρατήρηση παίρνει την αντίστοιχη μέτρηση της κατανάλωσης ενεργού ισχύος. Τέλος, κάθε παρατήρηση περιλαμβάνει ένα timestamp όπου σημειώνεται ο χρόνος εισαγωγής της στο measurement.
Αξίζει να σημειωθεί ότι υπάρχουν closed-source εκδόσεις της InfluxDB που περιλαμβάνουν επιπλέον λειτουργικότητες αφορορικά με θέματα διαθεσιμότητας, κλιμακωσιμότητας, αντιγράφων ασφαλείας και επαναφοράς των δεδομένων.

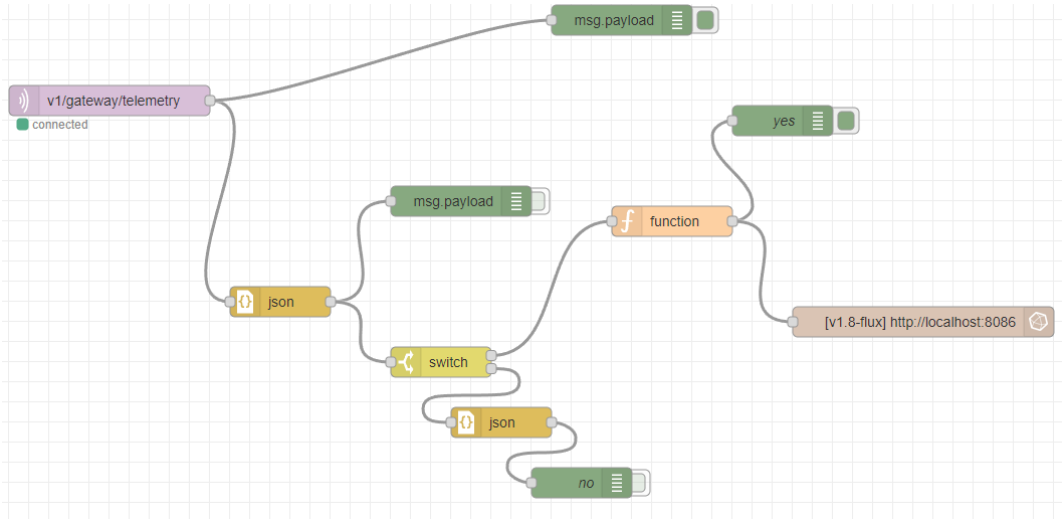


Σχήμα 3.1: Το TICK stack [33]

Τα δεδομένα μας προέρχονται από μία τριφασική γραμμή ρεύματος. Πιο συγκεκριμένα, συλλέγουμε τα δεδομένα μέσω μίας εφαρμογής τύπου IoT (Internet of Things). Λέγοντας Internet of Things αναφερόμαστε στην λογική της διασύνδεσης ενός συνόλου φυσικών αντικειμένων που ενσωματώνουν στοιχεία ηλεκτρονικής, λογισμικού, αισθητήρων και δικτύων, μέσω του διαδικτύου ή άλλου δικτύου επικοινωνιών, με άλλες συσκευές και συστήματα για την ανταλλαγή πληροφοριών. Τα βασικά εργαλεία που έχουμε χρησιμοποιήσει είναι ένας έξυπνος μετρητής, η βάση δεδομένων InfluxDB, το πρόγραμμα Node-RED και το πρωτόκολλο επικοινωνίας MQTT (Message Queuing Telemetry Transport).
Το Node-RED είναι ένα ανοιχτού κώδικα εργαλείο JavaScript με το οποίο συνδέουμε συσκευές υλικού, διεπεφές προγραμματισμού εφαρμογής και διαδικτυακές υπηρεσίες. Το

Node-RED δημιουργεί ροές δεδομένων μεταξύ των μερών ενός κυκλώματος τις οποίες ελέγχει μέσω κώδικα. Πιο συγκεκριμένα, δημιουργούμε κυκλώματα αποτελούμενα από κόμβους εισόδου, εξόδου και επεξεργασίας. Μέσω αυτών επιτελούμε διάφορες λειτουργίες όπως παρακολούθηση δεδομένων και αποστολή ειδοποιήσεων αλλά και ανάλυση και επεξεργασία δεδομένων. Οι ροές που δημιουργούμε αποθηκεύονται σε μορφή JSON. Με τους κόμβους εισόδου τροφοδοτούμε την ροή με δεδομένα, με τους κόμβους εξόδου ερχόμαστε σε επαφή με το αποτέλεσμα της επεξεργασίας και με τους κόμβους επεξεργασίας μπορούμε να εκτελέσουμε διάφορες λειτουργίες επί των δεδομένων βάσει κώδικα που έχουμε γράψει εμείς. Ιδιαίτερα σημαντικό είναι το γεγονός ότι υπάρχει μεγάλη ποικιλία βιβλιοθηκών που μπορούν να ενσωμάτωθουν στο εργαλείο αυτό δίνοντας την δυνατότητα να συμπεριλάβει πολλών τύπων συσκευές. Το Node-RED παρέχει συνεπώς στον χρήστη μεγάλο βαθμό αφαιρετικότητας καθότι μπορεί να συνδέει γρήγορα διάφορες συσκευές μεταξύ τους και να ελέγχει πλήρως το κύκλωμα μέσω ενός βολικού γραφικού περιβάλλοντος χρήστη.[27]
Το MQTT είναι ένα πρωτόκολλο μηνυμάτων που βασίζεται στο μοντέλο δημοσίευσης / συνδρομής (publish / subscribe) και χρησιμοποιείται για την επικοινωνία ανάμεσα σε μηχανημάτα. Στο MQTT έχουμε δύο μέρη που επικοινωνούν μεταξύ τους, τον μεσολαβητή (broker) και τον πελάτη (client). Ο πελάτης μπορεί είτε να δημοσιεύει επί ενός θέματος (topic) είτε να εγγράφεται ως συνδρομητής σε ένα θέμα είτε και τα δυο. Στην πρώτη περίπτωση (publish) στέλνει μηνύματα στον μεσολαβητή που τα φιλτράρει και τα στέλνει στους συνδρομητές. Ο πελάτης που στέλνει μηνύματα με πληροφορία (publisher) και πελάτης που λαμβάνει τα μηνύματα εν τέλει (subscriber) δεν γνωρίζονται μεταξύ τους αλλά τους γνωρίζει ο μεσολαβητής, κάτι που μας απαλλάσει από ζητήματα διαλειτουργικότητας και κάνει εύκολη την κλιμάκωση ενός δικτύου επικοινωνίας με αυτό το πρωτόκολλο. Η σύνδεση μεταξύ πελάτη και μεσολαβητή γίνεται με ανταλλαγή πακέτων όπως και ο τερματισμός της. Κάθε φορά που ο δημοσιεύων πελάτης στέλνει δεδομένα, επιλέγει το επίπεδο του QoS (Quality of Service) και αναλόγως λαμβάνει έναν αριθμό επιβεβαιώσεων για το μήνυμα που έστειλε. Στο Node-RED έχουμε την δυνατότητα να προσθέσουμε κόμβους MQTT, όπου ρυθμίζουμε τα απαραίτητα για να συνδεθούμε με τον μεσολαβητή. Οι κόμβοι μπορεί να είναι εισόδου που λειτουργούν ως subscribers και περνάνε τα ληφθέντα μηνύματα στην ροή μας, αλλά και εξόδου που λειτουργούν ως publishers και στέλνουν στον μεσολαβητή μηνύματα.[27]
Για το κύκλωμά μας χρησιμοποιήσαμε έναν μεσολαβητή τύπου Mosquitto, που αποτελεί έναν ανοιχτού κώδικα μεσολαβητή μηνυμάτων συμβατό με το πρωτόκολλο MQTT. Επίσης, χρησιμοποιήσαμε ως publisher πελάτη τον έξυπνο μετρητή. Προκειμένου να το πετύχουμε αυτό εγκαταστήσαμε όλες τις απαραίτητες βιβλιοθήκες. Από τον μεσολαβητή λαμβάνουμε μηνύματα με τις μετρήσεις μας στον πελάτη subscriber που παριστάνεται με έναν κόμβο εισόδου στο κύκλωμά μας. Την είσοδο αυτή ο subscriber την περνάμε από τον κόμβο JSON που μετατρέπει την πληροφορία από μορφή JSON σε JavaScript Object προκειμένου να μπορεί να την επεξεργαστεί ο κόμβος επεξεργασίας, που περιέχει JavaScript κώδικα, αλλά και από έναν κόμβο switch όπου ξεχωρίζουμε τα επιθυμητά μηνύματα. Εκεί σχηματοποιούμε τα δεδομένα μας στην επιθυμητή μορφή, επιλέγοντας πεδία και τοποθετόντας ετικέτες. Τέλος, το αποτέλεσμα της επεξεργασίας διοχετεύεται στην InfluxDB. Για να το κάνουμε αυτό εισάγουμε έναν κόμβο εξόδου με τον οποίο συνδεόμαστε με την υπηρεσία που φιλοξενεί την βάση μας και του διοχετεύουμε το αποτέλεσμα του κόμβου επεξεργασίας. Παρακάτω παρατίθεται το διάγραμμα της ροής. Οι πράσινοι κόμβοι εξόδου χρησιμοποιούνται για τον έλεγχο των ενδιάμεσων τιμών.

Για την InfluxDB υπάρχουν βιβλιοθήκες πελάτη, δηλαδή εξειδικευμένα πακέτα για διάφορες γλώσσες προγραμματισμού ώστε να είναι δυνατή η ομαλή ενσωμάτωση της βάσης στα διάφορα προγράμματα. Υπάρχει μεγάλη ποικιλία τέτοιων πακέτων. Ενδεικτικά, υπάρχουν βιβλιοθήκες για τις γλώσσες προγραμματισμού Python, JavaScript και PHP. Μέσω ενός Python script που χρησιμοποιεί την βιβλιοθήκη πελάτη influxdb θα λάβουμε δεδομένα από την βάση μας. Ο κώδικας είναι ο ακόλουθος:



Σχήμα 3.2: Η ροή μας στο Node-RED

```
1  #!/usr/bin/python3
2  from influxdb import InfluxDBClient
3
4  client = InfluxDBClient(host = 'XXX.XXX.XX.XX',
5                          port = 8086,
6                          username = 'XXXXX',
7                          password = 'XXXXXXXXXXXXXXX',
8                          database = 'energy')
9
10 result=client.query('SELECT * FROM "power" WHERE time >=now()-4w')
11 points = result.get_points(tags = {
12                 'Building_Name': 'PC-LAB',
13                 'Device_Serial': 'XXX.XXX.XXXXXX'})
14 f = open("datapoints.csv", "w")
15 f.write("Time,pwrA,pwrB,pwrC\n")
16 for point in points:
17     f.write(f'{point["time"]}, {point["pwrA"]},
18     {point["pwrB"]}, {point["pwrC"]}')
19     f.write('\n')
20 f.close()
21 client.close()
```

Σημειώνεται πως οι γραμμές 17 και 18 είναι μία γραμμή στον πραγματικό κώδικα που έχει γραφτεί έτσι για λόγους παρουσίασης.

Καταρχάς, για να συνδεθούμε στην βάση δεδομένων δημιουργούμε ένα στιγμιότυπο της κλάσης InfluxDBClient. Ως παραμέτρους περνάμε όλη την απαραίτητη πληροφορία προκειμένου να μπορούμε να στέλνουμε απευθείας αιτήματα στην υπηρεσία που φιλοξενεί την βάση διαμέσου του δημιουργηθέντα πελάτη (client). Ειδικότερα, στην παράμετρο host εισάγουμε το αναγνωριστικό της υπηρεσίας που φιλοξενεί την βάση (δηλαδή την δημόσια IP διεύθυνση), στην παράμετρο port την θύρα που χρησιμοποιεί η υπηρεσία, ακολούθως στις παραμέτρους username και password τοποθετούμε τον κωδικό και το όνομα χρήστη που έχουμε δημιουργήσει στην βάση και του έχουμε δώσει δικαίωμα να διαβάσει δεδομένα και τέλος δίνουμε το όνομα της βάσης με την οποία θα συνδεθούμε.

Αφότου λοιπόν καθορίσαμε την σύνδεσή μας στην βάση μπορούμε να υποβάλουμε ένα ερώτημα (query) και να λάβουμε τα δεδομένα μας. Για να το πετύχουμε αυτό θα χρησιμοποιήσουμε την μέθοδο query του αντικειμένου client που δημιουργήσαμε προηγουμένως. Ως μοναδικό όρισμα δίνουμε το ίδιο το ερώτημα το οποίο είναι συμβατό με την γλώσσα InfluxQL. Εσωτερικά, υπάρχει η παράμετρος method που λαμβάνει την

default τιμή GET και καθορίζει τον τύπο του HTTP αιτήματος (request) στην υπηρεσία που φιλοξενεί την βάση. Με την μέθοδο αυτή θα μπορούσαμε να στείλουμε άλλου τύπου HTTP request (πχ POST). Επίσης, σημαντική είναι η παράμετρος epoch με την οποία καθορίζεται η μορφή του πεδίου του χρόνου που λαμβάνουμε. Την epoch δεν την θέτουμε αλλά αφήνουμε με την default τιμή της που είναι None και μας επιστρέφει τις χρονοσφραγίδες στην μορφή RFC3339 UTC με ακρίβεια nanosecond. Η μορφή RFC3339 UTC είναι η YYYY-MM-DDThh:mm:ss.nnnnnnnnnZ π.χ. 2021-12-19T12:48:03.379000000Z. Με το Z στο τέλος εννοείται πως έχουμε μηδενική μετατόπιση ως προς την UTC ώρα. Λόγω ακρίβειας λάβαμε λιγότερα μηδενικά. Η μέθοδος αυτή επιστρέφει ως αποτέλεσμα τα ζητούμενα δεδομένα ως ένα αντικείμενο της κλάσης ResultSet. Εμείς με το query απλώς λάβαμε όλες τις παρατηρήσεις του measurement power των τελευταίων τεσσάρων εβδομάδων. Για αυτό χρησιμοποιήσαμε την συνάρτηση now() που επιστρέφει τον χρόνο του τοπικού server σε επίπεδο nanosecond.
Στην συνέχεια, θα χειριστούμε το αποτέλεσμα του άνω query, για να πάρουμε τις παρατηρήσεις και να τις γράψουμε σε ένα αρχείο τύπου CSV (Comma-Separated Calues). Για αυτόν τον σκοπό θα χρησιμοποιήσουμε την μέθοδο get_points. Μέσω του ορίσματος tags θα φιλτράρουμε τα δεδομένα για να λάβουμε μόνο τα πεδία του χρόνου και των ενεργών ισχύων. Το αποτέλεσμα είναι ένας generator με όλες τις παρατηρήσεις. Αφού δημιουργήσουμε ένα csv αρχείο για εγγραφή με όνομα 'datapoints.csv' και γράψουμε στην πρώτη γραμμή τα ονόματα των στηλών θα γράψουμε μία γραμμή για κάθε παρατήρηση με τα δεδομένα να διαχωρίζονται με κόμμα. Αφού γράψουμε όλες τις παρατηρήσεις κλείνουμε το αρχείο και τέλος με την μέθοδο close του InfluxDBClient αντικειμένου που έχουμε φτιάξει κλείνουμε την HTTP συνεδρία που ανοίξαμε στην αρχή.
Αξίζει να σημειωθεί πως με την βιβλιοθήκη αυτή μπορούμε να πραγματοποιήσουμε πολλές λειτουργίες επί της βάσης InfluxDB. Μπορούμε να εκτελέσουμε λειτουργίες επί ενός συγκεκριμένου measurement (δημιουργία, καταστροφή, εισαγωγή δεδομένων κ.λ.π.) αλλά και λειτουργίες διαχειριστικής φύσεως, όπως δημιουργία χρήστη, λήψη λίστας με όλες τις βάσεις δεδομένων, λήψη λίστας με όλες τις measurements, δημιουργία πολιτικής διατήρησης. Σε κάθε περίπτωση η βιβλιοθήκη αυτή προσφέρει μεγάλη γκάμα δυνατοτήτων και χαρακτηρίζεται από την απλότητα της Python.

3.2 Αποθήκευση δεδομένων στο Cloud

Οι βάσεις δεδομένων μπορούν να χωριστούν σε δύο μεγάλες κατηγορίες, τις σχεσιακές και τις μη σχεσιακές. Η πλειονότητα των βάσεων είναι σχεσιακές. Μία σχεσιακή βάση δεδομένων βασίζεται στο σχεσιακό μοντέλο δεδομένων. Το σχεσιακό μοντέλο χρησιμοποιεί ένα σύνολο από πίνακες που αντιπροσωπεύουν τα δεδομένα και τις σχέσεις μεταξύ αυτών των δεδομένων. Κάθε πίνακας έχει πολλές στήλες και κάθε στήλη έχει ένα μοναδικό όνομα. Οι πίνακες είναι γνωστοί ως σχέσεις. Το σχεσιακό μοντέλο βασίζεται στις εγγραφές. Η βάση δεδομένων είναι δομημένη σε εγγραφές σταθερής μορφής διαφόρων τύπων. Κάθε πίνακας περιέχει εγγραφές ενός συγκεκριμένου τύπου. Κάθε τύπος εγγραφής ορίζει έναν σταθερό αριθμό πεδίων ή ιδιοτήτων, Οι στήλες του πίνακα αντιστοιχούν στις ιδιότητες του τύπου της εγγραφής.[39] Κάθε εγγραφή περιλαμβάνει ένα πεδίο που λειτουργεί ως αναγνωριστικό της (primary key) ενώ αναλόγως των σχέσεων μεταξύ των δεδομένων το αναγνωριστικό αυτό μπορεί να χρησιμοποιηθεί για αναφορά στην εγγραφή αυτή εντός άλλης εγγραφής (foreign key). Στα θετικά των σχεσιακών βάσεων είναι η απλότητα και η σαφήνεια τους καθώς και η εξασφάλιση μη ύπαρξης διπλότυπων λόγω της ύπαρξης των κλειδιών. Στα αρνητικά μπορούμε να τοποθετήσουμε την κλιμακωσιμότητα και την έλλειψη ελαστικότητας λόγω του περιοριστικού σχήματος δεδομένων που ακολουθείται.
Η *MongoDB* είναι μια μη-σχεσιακή βάση δεδομένων. Ειδικότερα, πρόκειται για μια document database που αποθηκεύει τα δεδομένα σε δομές της μορφής BSON, που

αποτελεί επέκταση του σχήματος JSON για την υποστήριξη επιπλέον τύπων δεδομένων. Τα δεδομένα αποθηκεύονται σε ζεύγη κλειδιού τιμής.[29] Η μη ύπαρξη σχήματος στην MongoDB την καθιστά ιδιαίτερα ευέλικτη υπό την έννοια ότι μπορούμε να αποθηκεύσουμε οποιουδήποτε τύπου δεδομένα σε οποιοδήποτε document εντός μιας βάσης. Τόσο στην SQL όσο και στις no-SQL βάσεις δεδομένων αναφερόμαστε βάσεις δεδομένων (databases). Ωστόσο, στις δεύτερες έχουμε αντί για πίνακες collections, αντί για εγγραφές έχουμε documents και αντί για στήλες έχουμε fields. Η MongoDB προσφέρει μεγάλη ευελιξία και παρουσιάζει εξαιρετική συμπεριφορά ως προς τον τομέα της κλιμακωσιμότητας. Επιλέξαμε την MongoDB καθώς η μορφή των δεδομένων της είναι εύκολα διαχειρίσιμη. Ως *cloud computation* ορίζουμε το μοντέλο υπολογισμού κατά το οποίο έχουμε πρόσβαση σε απομακρυσμένους υπολογιστικούς πόρους συμπεριλαμβανομένων servers, δικτύων και πόρων αποθήκευσης δεδομένων. Η τεχνική αυτή προσφέρει πολλά πλεονεκτήματα, όπως η δυνατότητα για μεγάλη κλιμακωσιμότητα διάφορων λειτουργιών καθώς και εξειδικευμένες λύσεις υψηλής απόδοσης για κάθε περίπτωση.

Υπάρχουν χοντρικά τριών ειδών υπηρεσίες Cloud που προσφέρονται, το *Software as a Service* (SaaS), όπου έχουμε λογισμικό το οποίο το διαχειρίζεται και το προσφέρει ο πάροχος στον χρήση επί πληρωμής μέσω διαδικτύου, το *Platform as a Service* (PaaS), όπου ο παρέχεται στον πελάτη το περιβάλλον για δικά του προγράμματα (π.χ. φιλοξενία υπηρεσίας ιστού) χωρίς όμως να δίνεται ο έλεγχος του λειτουργικού συστήματος, του υλικού και της υποδομής δικτύου επί της οποίας τρέχουν και το *Infrastructure as a Service* (IaaS), όπου ο πελάτης χρησιμοποιεί εικονικές μηχανές, εξυπηρετητές και χώρο αποθήκευσης κατευθείαν (π.χ. απλή αποθήκευση δεδομένων χρήστη).

Η αρχιτεκτονική του Cloud Computing μπορεί να συνοψιστεί στο επίπεδο *υλικού* (Hardware Layer) , όπου περιλαμβάνονται τα πραγματικά μηχανήματα και το υλικό (εξυπηρετητές, διακόπτες, καλώδια, συστήματα ψύξης κ.λ.π.), στο επίπεδο *υποδομής* (Infrastructure Level) στο οποίο γίνεται ο διαχωρισμός των πόρων υπολογισμού και αποθήκευσης με εικονικές τεχνολογίες, στο επίπεδο *πλατφόρμας* (Platforms Layer) που αποτελείται από λειτουργικά συστήματα και Frameworks εφαρμογών, ώστε να γίνεται η εγκατάσταση των εφαρμογών στα εικονικά μηχανήματα και τέλος στο επίπεδο *εφαρμογής* (Application Layer) που αποτελείται από υπηρεσίες Cloud οι οποίες είναι θεμελιωμένες στα προηγούμενα επίπεδα.

Η εγκατάσταση και η χρήση του cloud computing γίνεται βασικά με τέσσερις τρόπους: Στο *δημόσιο* μοντέλο (Public Cloud) οι διάφορες υπηρεσίες είναι διαθέσιμες στο γενικό κοινό από ένα τρίτο μέρος, τον πάροχο, μέσω του διαδικτύου. Στο *ιδιωτικό* μοντέλο (Private Cloud) η διαχείριση των δεδομένων και των διαδικασιών γίνεται αποκλειστικά εντός του οργανισμού του παρόχου, ενώ ο πάροχος έχει πλήρη έλεγχο επί των υποδομών του cloud. Το μοντέλο αυτό χαρακτηρίζεται από μεγαλύτερη ασφάλεια καθώς η πρόσβαση των χρηστών και τα δικαιώματα τους είναι σαφώς καθορισμένα από τον οργανισμό πάροχο. Στο μοντέλο *κοινότητας* (Community Cloud) το σύνολο των πόρων ελέγχεται και χρησιμοποιείται από ένα σύνολο οργανισμών με κοινά ενδιαφέροντα. Τα μέλη της κοινότητας έχουν πρόσβαση σε δεδομένα και εφαρμογές.[43] Τέλος, υπάρχουν τα *υβριδικά* μοντέλα (Hybrid Cloud) που συνδυάζουν κάποια από τα παραπάνω, που παραμένουν όμως ξεχωριστές οντότητες εντός των μοντέλων.[22] Συχνά, οι χρήστες χρησιμοποιούν δημόσια Clouds για λειτουργίες επεξεργασίας ή άλλες υπηρεσίες λογισμικού και τα ιδιωτικά για αποθήκευση των ευαίσθητων δεδομένων τους.

Ωστόσο, με την χρήση του Cloud έχουν ανακύψει διάφορα ζητήματα. Υπάρχουν προκλήσεις ασφάλειας, διαθεσιμότητάς και ιδιωτικότητας των δεδομένων, ζητήματα νομικής φύσεως ως προς την αποθήκευση των δεδομένων, θέματα διαλειτουργικότητας μεταξύ υπηρεσιών και ζητήματα απόδοσης του δικτύου. Ένα άλλο ζήτημα είναι η απόλυτη εξάρτηση χρηστών από έναν πάροχο υπηρεσιών Cloud με ότι αυτό συνεπάγεται για τους χρήστες.

Για την επίλυση ορισμένων εκ των παραπάνω ζητημάτων αναπτύχθηκε το Multi-Cloud Computing κατά το οποίο τα συστήματα Cloud χρησιμοποιούν πολλά δίκτυα και υπηρεσίες

Cloud ταυτόχρονα. Εν γένει τα συστήματα αυτά μπορούν να χρησιμοποιούν περισσότερους από έναν παρόχους Cloud. Με άλλα λόγια, σ'αυτά τα συστήματα ο χρήστης μπορεί να χρησιμοποιήσει διαφορετικές υπηρεσίες για διαφορετικές εφαρμογές (π.χ. να αποθηκεύσει τα δεδομένα του σε ιδιωτικό Cloud και να πραγματοποιήσει ανάλυση δεδομένων σε άλλη υπηρεσία Cloud ή να αποθηκεύει δεδομένα σε διαφορετικές υπηρεσίες Cloud)[22]
Το MongoDB Atlas είναι μία υπηρεσία βάσεων δεδομένων τύπου Multi-Cloud που υποστηρίζει βάσεις δεδομένων MongoDB καθώς και διάφορες λειτουργικότητες επί αυτών. Πρόκειται, δηλαδή για μία *DBaaS* (DataBase as a Service) που αποτελεί μια εξειδικευμένη PaaS. Συγκεκριμένα χρησιμοποιεί έναν συνδυασμό πολλών δημοσίων, ιδιωτικών και υβριδικών Clouds από διαφορετικούς παρόχους για να προσφέρει διάφορες δυνατότητες στους χρήστες. Παράλληλα, προσφέρει την δυνατότητα αποθήκευσης δεδομένων σε Multi-Cloud Clusters ώστε τα δεδομένα να είναι κατανεμημένα σε πολλαπλά δημόσια Clouds επιτρέποντας ευκολία μεταφοράς δεδομένων και ανθεκτικότητα.[50] Πέραν αυτών, παρέχει και ένα δωρεάν πακέτο στο οποίο μπορούμε να χρησιμοποιήσουμε ένα Cluster μηχανημάτων σε ένα Cloud (απλό Cloud Computing) και να αποθηκεύσουμε τα δεδομένα μας δωρεάν (με όριο 512MB). Εμείς χρησιμοποιήσαμε το πακέτο αυτό. Διαλέξαμε να ανεβάσουμε τις βάσεις μας στο Cloud και όχι τοπικά για λόγους διαθεσιμότητας και ασφάλειας. Δημιουργήσαμε δυο collections, μία για τις χρονοσειρές μας και μια για την αποθήκευση των αποτελεσμάτων των πειραμάτων. Περισσότερες λεπτομέρειες για τα περιεχόμενά τους στο κομμάτι του Backend της εφαρμογής.

Κεφάλαιο 4

Πρόβλεψη με Αλγορίθμους Μηχανικής Μάθησης

4.1 Τεχνητή Νοημοσύνη

Ένα βασικό χαρακτηριστικό του ανθρώπινου είδους είναι η νοημοσύνη. Η μελέτη της νοημοσύνης είναι ένα από τα πιο παλιά θέματα που απασχόλησαν την ανθρωπότητα, που προσπάθησε να περιγράψει τον μηχανισμό με τον οποίο μαθαίνουμε, απομνημονεύουμε, βλέπουμε, αντιλαμβανόμαστε και συλλογιζόμαστε. Το πεδίο της τεχνητής νοημοσύνης επιχειρεί να κατανοήσει αλλά και να κατασκευάσει νοήμονες οντότητες. Υπάρχουν πολλοί ορισμοί του επιστημονικού πεδίου της τεχνητής νοημοσύνης. Άλλοι επικεντρώνονται στις διαδικασίες σκέψης και την συλλογιστική, ενώ άλλοι ασχολούνται περισσότερο με το κομμάτι της συμπεριφοράς. Κάποιοι μετρούν την επιτυχία βάσει της εγγύτητας προς τις ανθρώπινες επιδόσεις, σε αντίθεση με κάποιους άλλους που τη μετρούν σε σχέση με την έννοια της ορθολογικότητας. Ένα σύστημα είναι ορθολογικό όταν κάνει πάντα το σωστό δεδομένων των όσων γνωρίζει. Ένας γενικότερος ορισμός που καλύπτει τις παραπάνω κατηγορίες ορισμών είναι ο εξής:

> *Τεχνητή Νοημοσύνη* είναι ο τομέας της Επιστήμης των Υπολογιστών που ασχολείται με τη σχεδίαση και την υλοποίηση προγραμμάτων τα οποία είναι ικανά να μιμηθούν τις ανθρώπινες γνωστικές ικανότητες, εμφανίζοντας έτσι χαρακτηριστικά που αποδίδουμε συνήθως σε ανθρώπινη συμπεριφορά, όπως για παράδειγμα η επίλυση προβλημάτων, η αντίληψη μέσω όρασης, η μάθηση, η εξαγωγή συμπερασμάτων, η κατανόηση φυσικής γλώσσας κτλ.[54]

Στην τεχνητή νοημοσύνη έχουν συνεισφέρει ιδέες, τεχνικές και απόψεις διάφοροι κλάδοι, με τους κυριότερους να είναι η φιλοσοφία, τα μαθηματικά, τα οικονομικά, οι νευροεπιστήμες, η ψυχολογία, η τεχνολογία υπολογιστών, η θεωρία ελέγχου και η γλωσσολογία. Η τεχνητή νοημοσύνη συστηματοποιεί και αυτοματοποιεί τις διανοητικές εργασίες γι'αυτό μπορεί να εφαρμοστεί σε κάθε ανθρώπινη διανοητική δραστηριότητα.[37] Θα μπορούσαμε να πούμε πως η τεχνητή νοημοσύνη περικλείει ένα πλήθος ερευνητικών πεδίων, από γενικού σκοπού, όπως η αντίληψη και η συλλογιστική έως πιό συγκεκριμένων, όπως το σκάκι, η διάγνωση ασθενειών και άλλα.

Μπορούμε να διακρίνουμε δυο βασικές προσεγγίσεις για την τεχνητή νοημοσύνη. Η πρώτη είναι η κλασική ή συμβολική τεχνητή νοημοσύνη που βασίζεται στην κατανόηση των νοητικών διεργασιών και ασχολείται με την προσομοίωση της ανθρώπινης νοημοσύνης προσεγγίζοντας την με αλγορίθμους και συστήματα που βασίζονται στη γνώση, χρησιμοποιώντας ως δομικές μονάδες τα σύμβολα. Ένα σύμβολο μπορεί να αναπαριστά μια έννοια ή μια σχέση μεταξύ εννοιών. Τα σύμβολα αποκτούν νόημα όταν συνδέονται μεταξύ τους. Τα ονόματα των συμβόλων είναι συνήθως τέτοια ώστε να έχουν νόημα για τους ανθρώπους. Η δεύτερη είναι η υπολογιστική νοημοσύνη ή συνδετική ή μη συμβολική τεχνητή νοημοσύνη που βασίζεται στη μίμηση βιολογικών διεργασιών, όπως η διαδικασία της εξέλιξης των ειδών ή η λειτουργία του εγκεφάλου (π.χ. νευρωνικά δίκτυα και γενετικοί αλγόριθμοι).[54] Αν και η συμβολική τεχνητή νοημοσύνη είναι κατάλληλη για την επίλυση καλώς ορισμένων λογικών προβλημάτων, έχει αποδειχτεί πως είναι ακατάλληλη για την

επίλυση πιό σύνθετων ασαφών προβλημάτων, όπως η ταξινόμηση εικόνων και η αναγνώριση φωνής.[12] Τέτοιου είδους προβλήματα τα προσεγγίζουμε με μεθόδους που ανάγονται στην κατηγορία της μηχανικής μάθησης.

4.2 Μηχανική Μάθηση

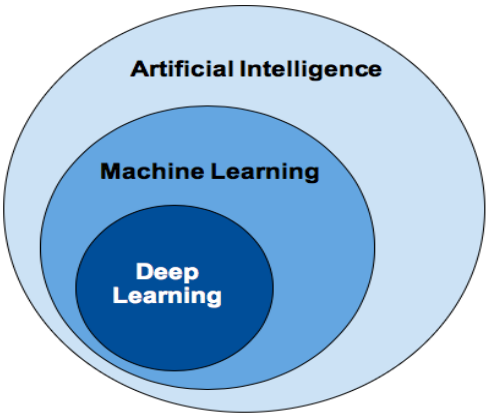
Ένα από τα σημαντικότερα πεδία της τεχνητής νοημοσύνης είναι η μηχανική μάθηση. Γενικότερα, η μάθηση για ένα σύστημα συνδέεται με την ικανότητα πρόσληψης γνώσης κατά την αλληλεπίδραση με το περιβάλλον στο οποίο βρίσκεται και την ικανότητα βελτίωσης του τρόπου εκτέλεσης των ενεργειών του μέσα από την επανάληψη, δηλαδή το σύστημα προσαρμόζεται αναφορικά με τις λειτουργίες που επιτελεί και έτσι βελτιώνει την απόδοσή του. Λέγοντας, λοιπόν, *μηχανική μάθηση* εννοούμε την δυνατότητα των συστημάτων τεχνητής νοημοσύνης να μαθαίνουν μέσω της εξαγωγής προτύπων από δεδομένα που λαμβάνουν από το περιβάλλον τους.[20] Σημαντικό ρόλο στο κατά πόσο μαθαίνουν καλά τα συστηματα παίζει η αναπαράσταση των δεδομένων με τα οποία τα τροφοδοτούμε. Στα συστήματα συμβολικής τεχνητής νοημοσύνης, η μάθηση πραγματοποιείται με την καταχώρηση της γνώσης αφού αυτή έχει δομηθεί και αναπαράσταθεί καταλλήλως. Άλλα συστήματα τεχνητής νοημοσύνης μαθαίνουν μετασχηματίζοντας την εσωτερική τους δομή. Υπάρχουν δύο κατηγορίες μηχανικής μάθησης, η *επιβλεπόμενη* (supervised learning) και η *μη επιβλεπόμενη* (unsupervised learning).
Στην μάθηση με επίβλεψη το σύστημα προσπαθεί να μάθει να προσομοιώνει την λειτουργία μίας συνάρτησης (συνάρτηση στόχος), που αντιστοιχίζει δεδομένα εισόδου σε τιμές εξόδου (στόχοι). Η τιμή που προβλέπει η συνάρτηση λέγεται εξαρτημένη μεταβλητή ή μεταβλητή εξόδου, ενώ η είσοδος που λαμβάνει είναι ένα σύνολο τιμών που λέγονται ανεξάρτητες μεταβλητές ή χαρακτηριστικά.[54] Δοθέντος ενός συνόλου δειγμάτων που περιέχουν χαρακτηριστικά και επιθυμητές εξόδους (ετικέτες), το σύστημα μας εξετάζει διάφορες εναλλακτικές συναρτήσεις προκειμένου να προσεγγίσει την συνάρτηση στόχου. Τα δεδομένα για τα οποία γνωρίζουμε τόσο την είσοδο όσο και την τιμή της εξόδου ονομάζονται δεδομένα εκπαίδευσης. Οι ετικέτες που αντιστοιχούν στα χαρακτηριστικά συχνά τοποθετούνται από ανθρώπους κάτι που καθιστά τα δεδομένα αυτά ακριβά. Η επιβλεπόμενη μάθηση βασίζεται στην επαγωγική υπόθεση βάσει της οποίας αν έχουμε προσεγγίσει καλά την συνάρτηση στόχο για μεγάλο αριθμό δειγμάτων, η προσέγγιση θα είναι ικανοποιητική και για περιπτώσεις δειγμάτων που δεν έχουν εξεταστεί. Η επιβλεπόμενη μάθηση ασχολείται κυρίως με δυο κατηγορίες προβλημάτων, την ταξινόμηση (classification) και την παλινδρόμηση (regression). Στα πρώτα παράγουμε σαν έξοδο με τα μοντέλα μας διακριτές τιμές, τις κλάσσεις ενώ στα δεύτερα αριθμητικές τιμές. Μία υποκατηγορία της επιβλεπόμενης μάθησης είναι η *αυτο-επιβλεπούμενη μάθηση* (self-supervised learning). Σε αυτήν την περίπτωση οι ετικέτες των δειγμάτων παράγονται από τα ίδια τα δεδομένα, συχνά μέσω ενός ευριστικού αλγορίθμου.[12] Εφαρμογή της αυτοεπιβλεπούμενης μάθησης είναι οι auto-encoders. Η δεύτερη μεγάλη κατηγορία της μηχανικής μάθησης είναι η μάθηση χωρίς επίβλεψη. Σ'αυτήν την περίπτωση παρέχονται δεδομένα τα οποία δεν περιέχουν κάποια επιθυμητή έξοδο. Στόχος του συστήματός είναι να ανακαλύψει συσχετίσεις και ομάδες βασισμένο μόνο στις ιδιότητες των δεδομένων.[54] Η διαδικασία αυτή μπορεί να χρησιμοποιηθεί και ως μέρος της ανάλυσης και κατανόησης των δεδομένων. Η μη επιβλεπόμενη μάθηση ασχολείται με ζητήματα όπως η συσταδοποίηση και η μείωση διαστατιτκότητας των δεδομένων. Τέλος, μια περίπτωση μη επιβλεπόμενης μάθησης αποτελεί η *ενισχυτική μάθηση* (reinforcement learning) κατά την οποία το σύστημα προσπαθεί να μάθει μέσα από την αλληλεπίδραση με το περιβάλλον. Βασικό στοιχείο της κατηγορίας αυτής είναι η ανταμοιβή (reward). Στόχος είναι να μεγιστοποιήσουμε την ανταμοιβή που λαμβάνουμε. Το σύστημα πρέπει μόνο του να ανακαλύψει ποιές ενέργειες αποφέρουν την μεγαλύτερη δυνατή ανταμοιβή. Είναι συχνό σε διαδραστικά προβλήματα να

μην μπορούμε να εξάγουμε την επιθυμητή συμπεριφορά για κάθε πιθανή κατάσταση και συνεπώς το σύστημα θα πρέπει να μπορεί να μαθαίνει όχι από πρότυπα με επιθυμητή έξοδο αλλά από την δική του εμπειρία.[54] Σε τέτοιες περιπτώσεις είναι κατάλληλη η ενισχυτική μάθηση.

4.3 Βαθιά Μάθηση

Όπως αναφέραμε παραπάνω, η απόδοση των αλγορίθμων μηχανικής μάθησης εξαρτάται από την αναπαράσταση των δεδομένων με τα οποία τους τροφοδοτούμε. Είναι σύνηθες στην επιστήμη των υπολογιστών διαφορετική οργάνωση των δεδομένων να μας επιτρέπει να επιτελούμε λειτουργίες σε σημαντικά διαφορετικό χρόνο. Ακριβώς με την ίδια λογική πολλά προβλήματα που βρίσκονται στον χώρο της τεχνητής νοημοσύνης μπορούν να επιλυθούν αποδοτικότερα αν επιλέξουμε σωστά το σύνολο των χαρακτηριστικών που έχουν τα δείγματα εισόδου μας. Για πολλά προβλήματα, λόγω της φύσης τους ή για άλλους λόγους, είναι δύσκολο να επιλέξουμε τα κατάλληλα χαρακτηριστικά. Μια λύση σε αυτό το ζήτημα είναι να χρησιμοποιήσουμε μηχανική μάθηση για να εξάγουμε τα ίδια τα χαρακτηριστικά (representation learning).[20] Με άλλα λόγια, αντί να αποτυπώνουμε αναπαραστάσεις δεδομένων σε μεταβλητές εξόδου, αποτυπώνουμε αναπαραστάσεις σε άλλες αναπαραστάσεις. Όταν αναζητούμε τα κατάλληλα χαρακτηριστικά, στοχεύουμε στον διαχωρισμό των παραγόντων διαφοροποίησης που περιγράφουν τα δεδομένα. Οι παράγοντες αυτοι είναι οι έννοιες μέσω των οποίων αντιλαμβανόμαστε την διαφοροποίηση στα δεδομένα (ένταση στον ήχο, χρώμα στην εικόνα κλπ). Συνήθως, απαιτείται να απλοποιήσουμε αυτούς τους παράγοντες καθώς και να απορρίψουμε αρκετούς από αυτούς. Ωστόσο, πολλές φορές είναι πολύ δύσκολο να εξάγουμε υψηλού επιπέδου αφηρημένα χαρακτηριστικά από δεδομένα, γιατί κάποιοι από τους παράγοντες διαφοροποίησής τους αναγνωρίζονται μόνο με χρήση της λογικής του ανθρώπου. Η *βαθιά μάθηση* (Deep Learning), που αποτελεί μια υποκατηγορία της μηχανικής μάθησης, επιλύει το παραπάνω ζήτημα, εισάγωντας αναπαραστάσεις που εκφράζονται βάσει απλούστερων αναπαραστάσεων. Όπως δηλώνει και το όνομα της κατηγορίας, τα συστήματα βαθιάς μάθησης αποτελούνται από πολλαπλά επίπεδα. Με την μέθοδο αυτή οικοδομούμε σύνθετες έννοιες χρησιμοποιώντας απλούστερες. Τα επίπεδα ενός συστήματος βαθιάς μάθησης είναι πως αυτά περιγράφουν το τρόπο με τον οποίο σχετίζονται οι έννοιες μεταξύ τους. Σε όρους συναρτήσεων, έχουμε μία συνάρτηση που δομείται βάσει πολλών μικρότερων, οι οποίες παρέχουν μια αναπάράσταση των δεδομένων. Μια άλλη προσέγγιση για την βαθιά μάθηση είναι πως επιτρέπουν στο υπολογιστικό σύστημα να μάθει ένα πρόγραμμα που απαρτίζεται από πολλά βήματα. Κάθε επίπεδο του συστήματός μας μπορεί να θεωρηθεί ως μία κατάσταση όπου αποθηκεύονται τα αποτελέσματα της παράλληλης εκτέλεσης ενός συνόλου εντολών, με τα αποτελέσματα να μπορούν να χρησιμοποιηθούν στα επόμενα επίπεδα. Όσο πιο βαθιά είναι η αρχιτεκτονική του συστήματος τόσο περισσότερα σύνολα μπορούν να εκτελεστούν στην σειρά. Αυτή η εμφωλευμένη ιεραρχία των εννοιών προσφέρει μεγάλη ισχύ και προσαρμοστικότητα στα μοντέλα αυτού του τύπου.[20]. Η βαθιά μάθηση έχει σημειώσει σημαντικά αποτελέσματα σε ζητήματα που αφορούν όραση και ακοή, όπως αναγνώριση φωνής σε σχεδόν ανθρώπινο επίπεδο και μεγάλη βελτίωση στην μετατροπή από κείμενο σε ομιλία.[12]

Στην παρούσα εργασία θα μετατρέψουμε την πρόβλεψη χρονοσειρών σε πρόβλημα αυτο-επιβλεπόμενης μάθησης (παλινδρόμηση), το οποίο θα το προσεγγίσουμε με μοντέλα τόσο βαθιάς μάθησης, όπως τα νευρωνικά δίκτυα, τα συνελικτικά δίκτυα, τα ανατροφορδοτούμενα νευρωνικά δίκτυα και το μοντέλο NBEATS, αλλά και πιο "κλασσικές" μεθόδους μηχανικής μάθησης, το δένδρο απόφασης, το τυχαίο δάσος, τον αλγόριθμο ενίσχυσης κλίσης, την γραμμική παλινδρόμηση, τον αλγόριθμο των κ-κοντινότερων γειτόνων και τις μηχανές διανυσμάτων υποστήριξης.



Σχήμα 4.1: Η τεχνητή νοημοσύνη ως σύνολο [51]

4.4 Μέθοδοι Μηχανικής Μάθησης

Όπως αναφέραμε προηγουμένως τα δεδομένα μας χωρίζονται στα σύνολα εκπαίδευσης και ελέγχου. Χρησιμοποιώντας τα δεδομένα εκπαίδευσης προσαρμόζουμε κατάλληλα τα μοντέλα μας ώστε να προσομοιώνουν την λειτουργία μίας συνάρτησης που αντιστοιχεί τις εισόδους σε εξόδους. Η προσαρμογή γίνεται χρησιμοποιώντας μία συνάρτηση κόστους η οποία μας δηλώνει κατά πόσο είμαστε κοντά στην επιθυμητή αποτύπωση των δεδομένων. Τα μοντέλα μας αναλόγως του πόσο σύνθετα είναι μπορούν να πετύχουν καλύτερη ή χειρότερη προσαρμογή επί των δεδομένων εκπαίδευσης. Ωστόσο, καλή απόδοση στα δεδομένα εκπαίδευσης δεν σημαίνει απαραιτήτως εξίσου καλή απόδοση και στα δεδομένα ελέγχου, κάτι που είναι και ο στόχος μας. Είναι συχνό το φαινόμενο όπου το μοντέλο απομνημονεύει τα δεδομένα εκπαίδευσης και λειτουργεί μόνο βάσει αυτών. Αυτό μπορεί να συμβεί εξαιτίας της ύπαρξης κάποιου χαρακτηριστικού των δεδομένων εκπαίδευσης που δεν υπάρχει στην υποκείμενη προς μοντελοποίηση συνάρτηση.[21] Το φαινόμενο αυτό λέγεται υπερπροσαρμογή (overfitting) και προσπαθούμε να το αποφύγουμε. Σκοπός μας είναι γνώση που αποκτάει το μοντέλο μας από την εκπαίδευση, να γενικεύται καλά σε δεδομένα που δεν έχει ξαναδεί. Πριν ξεκινήσουμε με την ανάλυση των αλγορίθμων μηχανικής μάθησης, θα αναφερθούμε στις δύο κατηγορίες στις οποίες χωρίζονται οι μεθόδοι μηχανικής μάθησης, τις παραμετρικές και τις μη παραμετρικές. Στις πρώτες κάνουμε μία υπόθεση για την μορφή της υπό μοντελοποίησης συνάρτησης και προσπαθούμε απλώς να υπολογίσουμε τις απαραίτητες μεταβλητές, μέσω των δεδομένων εκπαίδευσης. Οι μέθοδοι αυτές έχουν το πλεονέκτημα της απλότητας, ωστόσο πολλές φορές η υπόθεση στην οποία βασίζονται απέχει πολύ από την πραγματικότητα. Στις μη παραμετρικές μεθόδους ψάχνουμε μία εκτίμηση της συνάρτησης βάσει των δεδομένων εκπαίδευσης, χωρίς να κάνουμε κάποια υπόθεση για την μορφή της. Με τις μεθόδους αυτού του τύπου μπορούμε να μοντελοποιήσουμε μεγαλύτερη ποικιλία σχημάτων συναρτήσεων, ωστόσο απαιτούν μεγάλο πλήθος παρατηρήσεων για να παράγουν ακριβείς εκτιμήσεις.[48]. Στα πλαίσια της παρούσας εργασίας θα αναλύσουμε έξι αλγορίθμους. Θα ξεκινήσουμε με την γραμμική παλινδρόμηση, μετά θα αναφερθούμε στους κ-κοντινότερους γείτονες, ύστερα στις μηχανές διανυσμάτων υποστήριξης και αμέσως μετά σε τρεις δενδροειδείς αλγορίθμους. Πιο συγκεκριμένα, θα γράψουμε για το δένδρο αποφάσης, το τυχαίο δάσος και την ενίσχυση κλίσης.

Γραμμική Παλινδρόμηση

Η γραμμική παλινδρόμηση αποτελεί την πιό απλή μέθοδο μηχανικής μάθησης. Παρά την απλότητά που την διακρίνει είναι ιδιαίτερα χρήσιμη και εξακολουθεί να χρησιμοποιείται ευρέως. Πολλές πιο σύνθετες μέθοδοι αποτελούν γενικεύσεις και επεκτάσεις της γραμμικής

παλινδρόμησης. Θα αναλύσουμε, αρχικά, την απλή γραμμική παλινδρόμηση και ύστερα θα αναφερθούμε στην πολλαπλή γραμμική παλινδρόμηση.
Για την *απλή γραμμική παλινδρόμηση* θεωρούμε πως τα δείγματά μας έχουν μία είσοδο μίας διάστασης και έξοδο επίσης μίας. Για την περίπτωση αυτή υιοθετούμε την βασική παραδοχη πως τα δεδομένα εξόδου έχουν γραμμική σχέση με τα δεδομένα εξόδου, δηλαδή ισχύει πως:

$$\Upsilon = \beta_0 + \beta_1 X + \varepsilon$$

Στην γραμμική σχέση έχουμε προσθέσει τον όρο ε, που αντιπροσωπεύει την τυχαιότητα, υπό την έννοια πως αποδεχόμαστε πως η πραγματική σχέση εισόδου-εξόδου δεν μπορεί να μοντελοποιηθεί γραμμικά με απόλυτη ακρίβεια. Ο όρος αυτός είναι ανεξάρτητος της εισόδου. Στόχος μας είναι να βρούμε μια προσέγγιση των συντελεστών β_0 και β_1,τα $\hat{\beta_0}$ και $\hat{\beta_1}$, έτσι ώστε να μπορούμε να περιγράψουμε τα δεδομένα μας με την κάτωθι σχέση:

$$y_i = \hat{\beta_0} + \hat{\beta_1} x_i$$

Με άλλα λόγια προσπαθούμε να βρούμε μία ευθεία η οποία να είναι κατά το δυνατόν πιο κοντά στα δεδομένα μας. Η εγγύτητα της ευθείας επιτυγχάνεται με διάφορους τρόπους. Αυτός που χρησιμοποιείται γενικά, είναι η ελαχιστοποίση των τετραγώνων των σφαλμάτων. Ειδικότερα, ψάχνουμε τους συντελεστές για τους οποίους ελαχιστοποιείται η ποσότητα:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = \sum_{i=1}^{n} (y_i - \hat{\beta_0} - \hat{\beta_1} x_i)^2$$

Όπου $\hat{y_i}$ είναι η τιμή που δίνει η αναπαράσταση που εξετάζουμε, ενώ y_i η πραγματική έξοδος. Τονίζεται ότι η διαδικασία αυτή αφορά μόνο τα δεδομένα του συνόλου εκπαίδευσης. Εν συνεχεία, με τους συντελεστές που υπολογίσαμε και την προηγούμενη γραμμική σχέση, θα παράξουμε τις προβλέψεις μας για τις εισόδους των προτύπων του συνόλου αξιολόγησης (test set). Αποδεικνύεται ότι για να ελαχιστοποιείται η ποσότητα RSS πρέπει οι συντελεστές να είναι:

$$\hat{\beta_1} = \frac{\sum_{i=1}^{n} (x_i - \overline{x})(y_i - \overline{y})}{\sum_{i=1}^{n} (x_i - \overline{x})^2}$$

$$\hat{\beta_0} = \overline{y} - \hat{\beta_1} \overline{x}$$

Όπου $\overline{x}$ και $\overline{y}$ οι μέσοι όροι των εισόδων και των εξόδων των δειγμάτων εκπαίδευσης.
Τις περισσότερες φορές τα δείγματα έχουν περισσότερες της μίας εισόδους. Για αυτές τις περιπτώσεις υπάρχει η *πολλαπλή γραμμική παλινδρόμηση*. Η λογική της πολλαπλής γραμμικής παλινδρόμησης είναι εντελώς παρόμοια με αυτή της απλής, με την διαφορά πως τώρα έχουμε περισσότερους όρους στην γραμμική εξίσωση. Εδώ υποθέτουμε για p μεταβλητές εισόδου πως η σχέση της εξόδου με τις εισόδους είναι η παρακάτω:

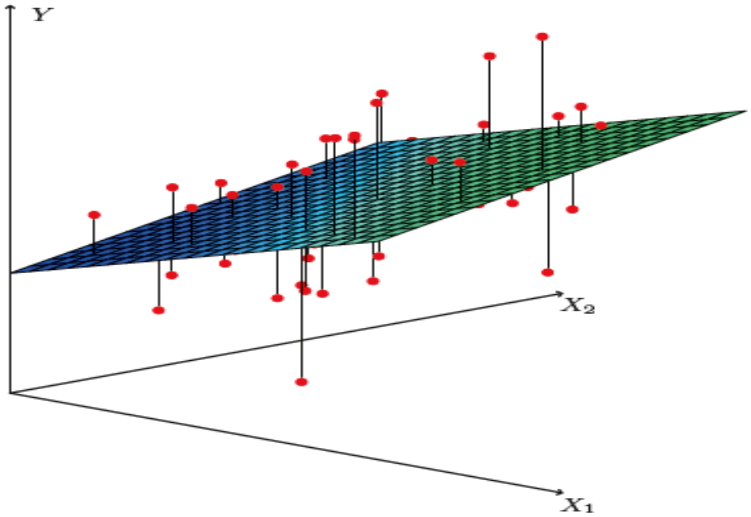
$$\Upsilon = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + ... + \beta_p X_p + \varepsilon$$

Εδώ, αντίστοιχα με πριν, προσπαθούμε να προσεγγίσουμε τους συντελεστές της εξίσωσης, ούτως ώστε να προσεγγίσουμε τα δεδομένα μας με μία σχέση της μορφής:

$$y_i = \hat{\beta_0} + \hat{\beta_1} x_{i1} + \hat{\beta_2} x_{i2} + ... + \hat{\beta_p} x_{ip}$$

Η παραπάνω σχέση αντιστοιχεί σε ένα υπερεπίπεδο (επίπεδο-για εισόδους δύο διαστάσεων). Ζητούμε αυτό να είναι κατά το δυνατόν πιο κοντά στα δεδομένα εκπαίδευσης. Και πάλι αυτό το πετυχαίνουμε μέσω της ελαχιστοποίησης του αθροίσματος των τετραγώνων των σφαλμάτων:

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y_i})^2 = \sum_{i=1}^{n} (y_i - \hat{\beta_0} - \hat{\beta_1} x_{i1} - \hat{\beta_2} x_{i2} - ... - \hat{\beta_p} x_{ip})^2$$



Σχήμα 4.2: Το επίπεδο παλινδρόμησης και τα σφάλματα απ'τις πραγματικές τιμές[48]

Ο υπολογισμός των συντελεστών της εξίσωσης του μοντέλου μας στην περίπτωση της πολλαπλής παλινδρόμησης είναι πιο πολύπλοκος από πριν. Αρχικά, ορίζουμε τους πίνακες X,Y από τα δεδομένα μας:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \text{ και } X = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1p} \\ 1 & x_{21} & x_{22} & \dots & x_{2p} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{np} \end{bmatrix}$$

Ορίζουμε επίσης τον πίνακα $\hat{\beta}$ που περιέχει τους ζητούμενους συντελεστές:

$$\hat{\beta} = \begin{bmatrix} \hat{\beta}_0 & \hat{\beta}_1 & \dots & \hat{\beta}_p \end{bmatrix}^T$$

Ο ζητούμενος πίνακας $\hat{\beta}$ μπορεί να υπολογιστεί ως εξής:

$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

Απαραίτητο είναι ο πίνακας $(X^T X)$ να είναι αντιστρέψιμος. Ωστόσο, επειδή ο υπολογισμός του προαναφερθέντα τύπου είναι επιρρεπής σε σφάλματα στρογγυλοποίσης προτείνεται ο κάτωθι τύπος για τον υπολογισμό του πίνακά μας[49] :

$$\hat{\beta}^* = (\mathcal{X}^T \mathcal{X})^{-1} \mathcal{X}^T \mathcal{Y} \text{ και } \hat{\beta}_0 = \overline{y} - \hat{\beta}^{*T} \overline{x}$$

Επεξήγηση πινάκων:

$$\hat{\beta}^* = \begin{bmatrix} \hat{\beta}_1 & \dots & \hat{\beta}_p \end{bmatrix}^T$$

$$\mathcal{Y} = \begin{bmatrix} y_1 - \overline{y} \\ y_2 - \overline{y} \\ \vdots \\ y_n - \overline{y} \end{bmatrix} \text{ και } \mathcal{X} = \begin{bmatrix} (x_{11} - \overline{x_1}) & \dots & (x_{1p} - \overline{x_p}) \\ (x_{21} - \overline{x_1}) & \dots & (x_{2p} - \overline{x_p}) \\ \vdots & \vdots & \vdots \\ (x_{11} - \overline{x_1}) & \dots & (x_{np} - \overline{x_p}) \end{bmatrix}$$

Σημειώνεται επίσης πως $\overline{y}$ είναι η μέση τιμή των εξόδων των δειγμάτων και $\overline{x}_i$ η μέση τιμη για τις αντίστοιχες μεταβλητές εισόδου των δειγμάτων, δηλαδή η μέση τιμή για την πρώτη είσοδο των δειγμάτων, για την δεύτερη κ.λ.π.
Μια εναλλακτική της πολλαπλής γραμμικής παλινδρόμησης θα ήταν να φτιάξουμε ένα μοντέλο απλής γραμμικής παλινδρόμησης για κάθε μια από τις μεταβλητές εισόδου και να τα

συνδυάσουμε για την παραγωγή της πρόβλεψης. Ωστόσο, αυτή η προσέγγιση δεν προτιμάται γιατί έχει δυο αρνητικά, πρώτον ότι δεν είναι ξεκάθαρος ο συνδυασμός του αποτελέσματος του κάθε ενός ξεχωριστού μοντέλου και δεύτερον η κατασκευή ξεχωριστών μοντέλων δεν λαμβάνει υπόψιν πιθανές συσχετίσεις μεταξύ των εισόδων. Τέλος, τα μοντέλα γραμμικής παλινδρόμησης μπορούν να επεκταθούν για να καλύπτουν περιπτώσεις μη γραμμικής σχέσης εισόδου-εξόδου. Μια τέτοια προσέγγιση είναι η *πολυωνυμική παλινδρόμηση*. Σ'αυτήν την περίπτωση περιλαμβάνουμε όρους υψωμένους στο τετράγωνο ή κάποια άλλη δύναμη στο μοντέλο μας. Μερικοί περιορισμοί για την γραμμική παλινδρόμηση πέραν της γραμμικής σχέσης μεταξύ της εισόδου και της εξόδου είναι πως οι ακραίες τιμές μπορούν να οδηγήσουν σε προβλήματα, ενώ η ύπαρξη ισχυρής συσχέτισης μεταξύ των μεταβλητών εισόδου ενδέχεται να δημιουργήσει δυσκολίες.[48]

Κ-Κοντινότεροι Γείτονες

Η μέθοδος των *κ-κοντινότερων γειτόνων* (K-Nearest Neighbors) είναι ένας απλός και διαισθητικός αλγόριθμος που εφαρμόζεται τόσο σε προβλήματα ταξινόμησης όσο και σε προβλήματα παλινδρόμησης. Όπως σε κάθε περίπτωση, έχουμε τα δεδομένα εκπαίδευσης για κάθε ένα από τα οποία γνωρίζουμε τις εισόδους και την έξοδο, καθώς και τα δεδομένα ελέγχου για τα οποία έχουμε μόνο τις εισόδους. Ο αλγόριθμος μπορεί να περιγραφεί από τρία βασικά βήματα:

1. Για κάθε δείγμα για το οποίο θέλουμε να υπολογίσουμε την τιμή εξόδου, υπολογίζουμε την απόσταση του από τα σημεία που αντιστοιχούν στα δείγματα του συνόλου εκπαίδευσης.

2. Επιλέγουμε τα κ πλησιέστερα σημεία για το δείγμα μας.

3. Για να παράξουμε την ζητούμενη πρόβλεψη συνδυάζουμε τις τιμές εξόδου των κ δειγμάτων-γειτόνων που επιλέχθηκαν στο προηγούμενο βήμα.

Από τα κατά τα άλλα απλά παραπάνω βήματα βλέπουμε ότι πρέπει να αποσαφηνίσουμε κάποια πράγματα. Καταρχάς, τι ορίζουμε ως απόσταση μεταξύ των δειγμάτων. Ο πιο συνηθισμένος τρόπος υπολογισμού της απόστασης μεταξύ των δειγμάτων, τα οποία αντιστοιχούν σε σημεία στον χώρο με διαστάσεις ίσες με το πλήθος των χαρακτηριστικών τους, είναι μέσω της ευκλείδειας απόστασης. Ωστόσο, υπάρχουν και άλλες μετρικές για τον υπολογισμό της απόστασης. Έστω πως έχουμε δείγματα με είσοδο n διαστάσεων. Θεωρούμε δύο σημεία στον χώρο, το x και το z. Στα πλάισια της διπλωματικής θα εξετάσουμε τις παρακάτω μετρικές:

Σημειώνεται πως η μετρική minkowski για p = 1 ταυτίζεται με την μετρική manhattan, ενώ για p = 2 ταυτίζεται με την ευκλείδεια.[40] Επίσης, στην μετρική αυτή μπορούμε να συνοδεύσουμε με βάρος την κάθε απόλυτη διαφορά, κάτι που επιλέξαμε να μην κάνουμε εδώ. Μια τροποποίηση του αλγορίθμου θα ήταν αντί του προσδιορισμού των γειτόνων με

Μετρική	Τύπος
euclidean	$\sqrt{(\sum_{i=1}^{n}(x_i - z_i)^2)}$
manhattan	$\sum_{i=1}^{n} \lvert x_i - z_i \rvert$
chebyshev	$max(\lvert x_i - z_i \rvert)$
minkowski	$(\sum_{i=1}^{n} \lvert x_i - z_i \rvert^p)^{\frac{1}{p}}$

Πίνακας 4.1: Μετρικές απόστασης

την παράμετρο κ μέσω απόστασης, να χρησιμοποιηθεί μία ακτίνα ε. Πιο συγκεκριμένα, θα επιλέγονται τα σημεία η απόσταση των οποίων από το σημείο μας είναι μικρότερη ή ίση με την ακτίνα ε. Η τεχνική αυτή έχει όμως τον κίνδυνο για τιμές του ε να μην υπάρχει κανένα σημείο που να πληρεί τις προϋποθέσεις για να συμπεριληφθεί στους γείτονες (σ'αυτήν την περίπτωση αυξάνουμε το ε). Η κατάσταση αυτή δεν είναι επιθυμητή και περιπλέκει την διαδικασία, συνεπώς προτιμούμε την αρχική μας προσέγγιση.[35]
Αφότου επιλέξουμε τους ζητούμενους κ γείτονες βάσει της απόστασης από το σημείο μας, σύμφωνα με το τρίτο βήμα χρησιμοποιούμε τις τιμές εξόδου τους για να παράξουμε την πρόβλεψή μας. Οι τιμές εξόδου των γειτόνων μπορούν να συνδυαστούν με δυο τρόπους, παίρνοντας τον μέσο όρο τους ή παίρνοντας τον σταθμισμένο μέσο όρο τους σύμφωνα με την απόσταση του καθενός. Στην πρώτη περίπτωση κάθε έξοδος συμμετέχει στο άθροισμα με το ίδιο βάρος, ενώ στην δεύτερη περίπτωση συμμετέχει με βάρος αντιστρόφως ανάλογο της απόστασης του από το σημείο για το οποίο κάνουμε πρόβλεψη, δηλαδή δίνουμε μεγαλύτερη βαρύτητα στους γείτονες που βρίσκονται πιο κοντά.
Υπάρχει μία ακόμα λεπτομέρεια σχετικά με την υλοποίηση του πρώτου βήματος. Υπάρχει η επιλογή να μην υπολογιστούν όλες οι αποστάσεις μεταξύ των σημείων, αλλά ένα μέρος αυτών έτσι ώστε να γίνει οικονομία πράξεων και ο αλγόριθμος μας να τρέξει γρηγορότερα. Πιο συγκεκριμένα, έχουμε τρεις επιλογές για τον αλγόριθμο που θα υπολογίσει ουσιαστικά τους κοντινότερους γείτονες. Η πρώτη επιλογή είναι ο *Brute Force* κατά τον οποίο βρίσκουμε εξαντλητικά όλες τις αποστάσεις. Η επιλογή αυτή είναι η πιο απλή, δουλεύει καλά για μικρό αριθμό δειγμάτων, αλλά δημιουργείται ζήτημα απόδοσης για περιπτώσεις που ο αριθμός των δειγμάτων είναι πολύ μεγάλος. Για τον λόγο αυτό έχουμε άλλες δύο επιλογές αλγορίθμων που περικλείουν την λογική των δενδρικών δομών. Ο πρώτος λέγεται *K-D Tree* και ο δεύτερος *Ball Tree*. Η λογική τους είναι πως αν ένα σημείο Α απέχει πολύ από ένα σημείο Β,το οποίο είναι πολύ κοντά σε ένα τρίτο σημείο Γ, τότε τα σημεία Α και Γ είναι βέβαιο πως απέχουν πολύ, χωρίς να χρειάζεται να υπολογιστεί η μεταξύ τους απόσταση. Ο αλγόριθμος K-D Trees βασίζεται στα δυαδικά δένδρα, διαχωρίζει το δείγμα και είναι πολύ γρήγορος για δεδομένα χαμηλών διαστάσεων (δηλαδή με χαμηλό πλήθος μεταβλητών εισόδου). Ο αλγόριθμος Ball Tree, αντίθετα, εμφανίζει πολύ καλύτερη απόδοση σε δεδομένα με υψηλές διαστάσεις, αν και η κατασκευή της δενδροειδούς δομής είναι πιο κοστοβόρα σ'αυτήν την περίπτωση.[4]
Από την περιγραφή του αλγορίθμου είναι εύκολο να αντιληφθούμε ότι είναι κομβικής σημασίας η επιλογή της παραμέτρου κ, δηλαδή του πλήθους των προς επιλογή γειτόνων. Μια μεγάλη τιμή της παραμέτρου κ μειώνει την επίδραση του θορύβου και οδηγεί σε καλύτερες προβλέψεις. Ωστόσο, μικρές τιμές του κ έχουν θετική επίδραση στην απλότητα υλοποίησης του αλγορίθμου.[16]

Μηχανές Διανυσμάτων Υποστήριξης

Οι *μηχανές διανυσμάτων υποστήριξης* (Support Vector Machines) είναι μία τεχνική μηχανικής μάθησης που είναι ιδιαίτερα δημοφιλής τόσο για προβλήματα ταξινόμησης όσο και για προβλήματα παλινδρόμησης με πολύ καλά αποτελέσματα και στα δύο. Προφανώς, στην παρούσα εργασία θα αναφερθούμε στην χρήση τους για τα δεύτερα προβλήματα. Η βασική ιδέα του αλγορίθμου είναι να βρούμε μία συνάρτηση που να δίνει ως έξοδο τιμή που να απέχει το πολύ κατά ε από την τιμή της πραγματικής εξόδου για κάθε ένα από τα δείγματα εκπαίδευσης, ενώ την ίδια στιγμή είναι επίπεδη[42]. Ως επίπεδη αποκαλούμε μια συνάρτηση, η οποία είναι διαφορίσιμη σε όλο το πεδίο ορισμού της και έχει μηδενική παράγωγο σε κάποιο σημείο. Με άλλα λόγια ανεχόμαστε σφάλματα στο διάστημα [-ε, +ε] αλλά όχι έξω από αυτό. Στην απλούστερη περίπτωση η συνάρτηση αυτή μπορεί να έχει γραμμική μορφή, δηλαδή έχουμε:

$$f(x) = < w, x > + \ b$$

Ως w,x παριστάνουμε δύο διανύσματα με διαστάσεις αυτές των χαρακτηριστικών των δειγμάτων εισόδου. Το b είναι ένας αριθμός, ενώ γράφοντας $< w, x >$ εννοούμε το εσωτερικό γινόμενο των δύο διανυσμάτων. Επιδιώκουμε, δηλαδή, η ευθειά μας να περιέχει εντός του περιθωρίου, που σχηματίζεται με δύο ευθείες σε απόσταση ε γύρω της, όλα τα σημεία από το σύνολο εκπαίδευσής. Όταν βρισκόμαστε σε τρεις διαστάσεις αντί για ευθεία έχουμε επίπεδο, ενώ για περισσότερες διαστάσεις, που είναι και το πιο σύνηθες, έχουμε υπέρεπίπεδο. Επιστρέφοντας στην εξίσωση της συνάρτησης, για να πετύχουμε το κομμάτι της επιπεδότητας της συνάρτησης, ζητούμε το διάνυσμα w να έχει κατά το δυνατόν το μικρότερο μέτρο. Συνεπώς, μαζί με τους περιορισμούς για τα σημεία μας, σχηματίζουμε το κάτωθι πρόβλημα βελτιστοποίησης:

$$\text{ελαχιστοποίηση} \quad \frac{1}{2}\|w\|^2$$

$$\text{υπό περιορισμούς} \quad \begin{cases} y_i - < w, x_i > - b \ \leq \varepsilon \\ < w, x_i > + b - \ y_i \ \leq \varepsilon \end{cases}$$

Ωστόσο, είναι πρακτικά αδύνατον για φυσιολογικές τιμές του ε να ισχύουν οι παραπάνω περιορισμοί για όλα τα πρότυπα εκπαίδευσης. Έτσι εισάγουμε τις μεταβλητές περιθωρίου (slack variables), τις οποίες συμβολίζουμε ξ_i και ξ_i^*, που δείχνουν την απόσταση των σημείων από την γραμμή ορίου, επιτρέποντας την ύπαρξη σημείων εκτός του περιθωρίου. Πλέον το πρόβλημά μας διαμορφώνεται ως εξής:

$$\text{ελαχιστοποίηση} \quad \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$$

$$\text{υπό περιορισμούς} \quad \begin{cases} y_i - < w, x_i > - b \ \leq \varepsilon + \xi_i \\ < w, x_i > + b - \ y_i \ \leq \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* \geq 0 \end{cases}$$

Η σταθερά C καθορίζει τον βαθμό ευαισθησίας μας στα σημεία εκτός περιθωρίου, σαν βάρος στην συμμετοχή τους στο κόστος το οποίο ελαχιστοποιούμε. Για την τιμή των μεταβλητών, που προστίθεται στο κόστος, έχουμε τον παρακάτω τύπο:

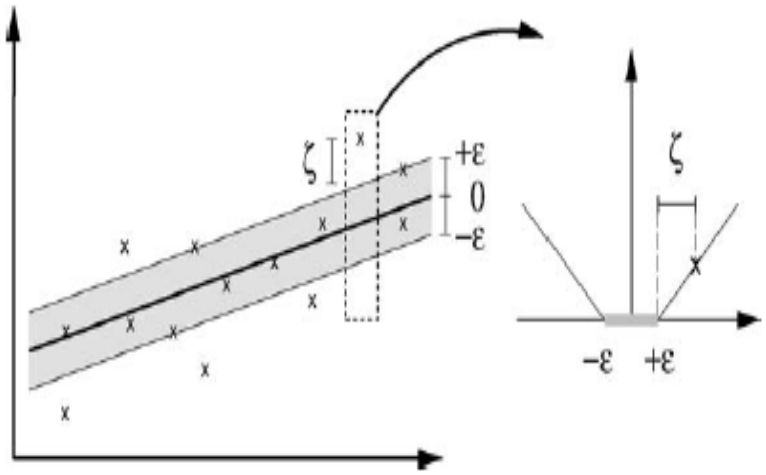
$$\xi = \begin{cases} 0, \text{αν το πρότυπο είναι εντός περιθωρίου} \\ \zeta = |y - f(x)| - \varepsilon, \text{αλλιώς} \end{cases}$$

Ερμηνεύοντας την τιμή που παίρνει η μεταβλητή χαλάρωσης, βλέπουμε ότι σε περίπτωση που το σημείο βρίσκεται εντός του περιθωρίου, δηλαδή δεν απέχουν από την ευθεία μας περισσότερο από ε, τότε δεν συμβάλουν καθόλου στο κόστος. Ωστόσο, ο αλγόριθμος αυτός μπορεί να αντιμετωπίσει και περιπτώσεις μη γραμμικότητας, μέσω του λεγόμενου "τεχνάσματος πυρήνα" (kernel trick). Καταρχάς, χρησιμοποιούμε μία συνάρτηση Φ, με την οποία αποτυπώνουμε τα χαρακτηριστικά των προτύπων σε έναν χώρο χαρακτηριστικών (feature space) F. Η συνάρτηση αυτή μπορεί να εισάγει μη γραμμικούς όρους και να αλλάξει τον αριθμό των διαστάσεων. Θεωρούμε ως πυρήνα μία συνάρτηση που λαμβάνει δύο πρότυπα (που έχουν περάσει από την Φ) και επιστρέφει έναν αριθμό, εδώ το εσωτερικό τους γινόμενο. Υπάρχουν διαφορετικοί πυρήνες στους οποίους θα αναφερθούμε αμέσως μετά. Το παραπάνω μαθηματικά γράφεται ως εξής:

$$k(x, x^{'}) =< \Phi(x), \Phi(x^{'}) >$$

Αν διατυπώσουμε το παραπάνω πρόβλημα βελτιστοποίησης μέσω πολλαπλασιαστή Lagrange και χρησιμοποιώντας το τέχνασμα πυρήνα, έχουμε:

$$\text{μεγιστοποίηση} \quad -\frac{1}{2}\sum_{i,j=1}^{l}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)k(x_i, x_j) - \varepsilon\sum_{i=1}^{l}(\alpha_i + \alpha_i^*) + \sum_{i=1}^{l} y_i(\alpha_i - \alpha_i^*)$$



Σχήμα 4.3: Η ευθεία παλινδρόμησης και η μεταβλητή ξ [42]

$$\text{υπό περιορισμούς} \quad \begin{cases} \sum_{i=1}^{l}(\alpha_i - \alpha_i^*) = 0 \\ \alpha_i, \alpha_i^* \in [0, C] \end{cases}$$

Επίσης, τώρα γράφουμε την συνάρτησή μας και το w ως εξής:

$$\begin{cases} w = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)\Phi(x_i) \\ f(x) = \sum_{i=1}^{l}(\alpha_i - \alpha_i^*)k(x_i, x) + b \end{cases}$$

Σημειώνεται ότι τα α_i, α_i^* είναι οι πολλαπλασιαστές Lagrange με τιμή μεγαλύτερη ή ίση του μηδενός, ενώ στο μη γραμμικό πρόβλημα η βελτιστοποίση γίνεται στον χώρο γνωρισμάτων F και όχι στον χώρο της αρχικής εισόδου (δηλαδή στο χώρο με διάσταση το πλήθος των γνωρισμάτων των προτύπων εισόδου). Για τον υπολογισμό της σταθεράς b που βλέπουμε στον τύπο της συνάρτησής μας, χρησιμοποιούμε τις συνθήκες Karush-Kuhn-Tucker. Επομένως, για την παραγωγή πρόβλεψης το πρότυπο εισόδου παιρνάει από την συνάρτηση Φ και ύστερα με την χρήση του πυρήνα υπολογίζεται μία πράξη με κάθε ένα από τα πρότυπα εκπαίδευσης. Τα αποτελέσματα των πράξεων συνοδευόμενα από ένα βάρος αθροίζονται μεταξύ τους και ύστερα με την υπολογισμένη σταθερά b, για να παράξουν την τελική μας πρόβλεψη.[42]
Τέλος, θα αναφέρουμε τους πυρήνες που θα δοκιμάσουμε στα πλαίσια της εργασίας. Έχουμε τους τέσσερις βασικούς πυρήνες:

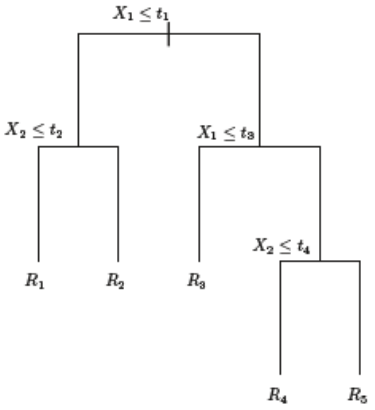
Σημειώνεται ότι $\|x\| =< x, x >$, ενώ τα C_0,d και γ είναι παράμετροι των μοντέλων[24], τιμές των οποίων δοκιμάζουμε για να βρούμε το καλύτερο δυνατόν. Για κάθε περίπτωση ισχύει $\gamma > 0$.

Όνομα Πυρήνα	Τύπος
Linear	$< x, x' >$
Polynomial	$(\gamma < x, x' > + C_0)^d$
RBF	$exp(-\gamma\|x - x'\|^2)$
Sigmoid	$\tanh(\gamma < x, x' > + C_0)$

Πίνακας 4.2: Διάφοροι πυρήνες

Δένδρο Απόφασης

Τα *δένδρα απόφασης* δεν αποτελούν απλώς έναν αλγόριθμο μηχανικής μάθησης, αλλά κατά μία έννοια ορίζουν μια κατηγορία τέτοιων αλγορίθμων. Πρόκειται για μια μέθοδο που εφαρμόζεται τόσο σε προβλήματα κατηγοριοποίησης όσο και παλινδρόμησης. Ένα δένδρο απόφασης αποτελείται από την ρίζα, τους ενδιάμεσους κόμβους ή κόμβους απόφασης και τα φύλλα ή τερματικούς κόμβους. Η ρίζα είναι ο κόμβος που περιλαμβάνει όλες τις παρατηρήσεις του συνόλου εκπαίδευσης και βρίσκεται στην κορυφή του δένδρου. Σε κάθε ενδιάμεσο κόμβο και στην ρίζα εφαρμόζεται μια συνθήκη διαχωρισμού, βάσει της οποίας χωρίζουμε τα δεδομένα σε δύο μέρη. Στο αριστερό μέρος έχουμε τα δεδομένα για τα οποία η συνθήκη διαχωρισμού είναι αληθής, ενώ στο δεξί μέρος τα δεδομένα για τα οποία είναι ψευδής. Η συνθήκη διαχωρισμού είναι μια ανισότητα για μία μεταβλητή εισόδου. Έτσι, τα δεδομένα των ενδιάμεσων κόμβων και της ρίζας μεταφέρονται σε παρακάτω επίπεδα, μέχρις ότου να φτάσουν στα φύλλα όπου δεν πραγματοποιείται κάποιος έλεγχος συνθήκης, αλλά σταματάει η διαδικασία διαχωρισμού. Το δένδρο απόφασης διαχωρίζει τον χώρο των μεταβλητών εισόδου σε τμήματα, τα οποία ορίζονται σύμφωνα με τις συνθήκες διαχωρισμού και είναι αυστηρά μη αλληλοεπικαλυπτόμενα. Κάθε φύλλο αντιστοιχεί σε ένα τμήμα. Από την στιγμή που έχουμε ένα δένδρο απόφασης, η διαδικασία της πρόβλεψης είναι μια πολύ απλή διαδικασία. Όπως είπαμε πριν κάθε φύλλο περιέχει ένα σύνολο παρατηρήσεων που ανήκει στο τμήμα του χώρου των μεταβλητών εισόδου που ορίζεται από τις συνθήκες που οδηγούν στο συγκεκριμένο φύλλο. Ως κλαδιά του δένδρου ορίζουμε τις υποδιαιρέσεις του, δηλαδή τα υποδένδρα. Όταν θέλουμε να κάνουμε πρόβλεψη, εκκινώντας από την ρίζα, ελέγχουμε τις συνθήκες διαχωρισμού και καταλήγουμε σε ένα φύλλο. Ως πρόβλεψη παίρνουμε τον μέσο όρο των εξόδων των παρατηρήσεων που περιλαμβάνονται στο φύλλο. Από την στιγμή που το υποσύνολο των δεδομένων σε κάθε τερματικό κόμβο είναι σταθερό, η παραγόμενη πρόβλεψη είναι σταθερή. Συνεπώς, το δένδρο απόφασης μπορεί να θεωρηθεί σαν μία εκτίμηση ιστογράμματος για το υπερεπίπεδο παλινδρόμησης.[11]



Σχήμα 4.4: Παράδειγμα δένδρου απόφασης [48]

Η κατασκευή του δένδρου απόφασης γίνεται ακολουθώντας την λογική των άπληστων αλγορίθμων. Ένας άπληστος αλγόριθμος (greedy algorithm) κάνει πάντοτε την επιλογή που φαίνεται καλύτερη τη δεδομένη χρονική στιγμή. Με άλλα λόγια, κάνει μία τοπικά βέλτιστη επιλογή με την ελπίδα πως η επιλογή αυτή θα οδηγήσει σε μία καθολικά βέλτιστη λύση.[44] Στα δένδρα αποφάσης, σε κάθε βήμα η επιλογή έχει να κάνει με την συνθήκη διαχωρισμού των δεδομένων. Ειδικότερα, καλούμαστε να επιλέξουμε ποιά μεταβλητή εισόδου των δεδομένων θα χρησιμοποιήσουμε και με ποιά τιμή θα την συγκρίνουμε. Για να επιλέξουμε τα παραπάνω δυο, χρειαζόμαστε να αξιολογήσουμε τις πιθανές συνθήκες. Για τον σκοπό αυτό υπάρχουν διάφορα κριτήρια. Παρακάτω θα εξετάσουμε τα δύο βασικότερα, τα οποία θα δοκιμάσουμε στην παρούσα εργασία. Θεωρούμε τα σύνολα δεδομένων που προκύπτουν από μία συνθήκη διαχωρισμού:

$$R_1 = \{X | X_j < s\} \text{ και } R_2 = \{X | X_j \geq s\}$$

Το πρώτο και το πιο δημοφιλές κριτήριο είναι η ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος, δηλαδή της ποσότητας:

$$\frac{1}{|R_1|} \sum_{i:x_i \in R_1} (y_i - \hat{y}_{R_1})^2 + \frac{1}{|R_2|} \sum_{i:x_i \in R_2} (y_i - \hat{y}_{R_2})^2$$

Το δεύτερο κριτήριο είναι η ελαχιστοποίηση του μέσου απολύτου σφάλματος, δηλαδή της ποσότητας:

$$\frac{1}{|R_1|}\sum_{i:x_i\in R_1}|y_i-\hat{y}_{R_1}|+\frac{1}{|R_2|}\sum_{i:x_i\in R_2}|y_i-\hat{y}_{R_2}|$$

Γράφοντας $\hat{y}_{R_1}$ εννούμε την πρόβλεψη από το σύνολο που δημιουργείται, δηλαδή ο μέσος όρος των σημείων. Άλλα κριτήρια είναι η εκδοχή του Friedman για το μέσο τετραγωνικό σφάλμα και η ελαχιστοποίηση της απόκλισης Poisson[41]. Επαναλαμβάνουμε την διαδικασία μέχρι να φτάσουμε στα φύλλα και έτσι δημιουργούμε ένα δένδρο απόφασης.
Η παραπάνω διαδικασία ως έχειν θα οδηγήσει σε ένα ιδιαιτέρως πολύπλοκο δένδρο απόφασης, το οποίο θα μοντελοποιεί άριστα τα δεδομένα εκπαίδευσης και θα έχει εξαιρετική απόδοση σε αυτά. Ωστόσο, δεν μπορούμε να πούμε το ίδιο για την απόδοσή του στα δεδομένα ελέγχου. Με άλλα λόγια αν δεν αλλάξουμε κάτι, στο δένδρο απόφασης εμφανίζεται το φαινόμενο της υπερπροσαρμογής. Για να το αποφύγουμε και να έχουμε καλή απόδοση και στα δεδομένα ελέγχου, είτε εφαρμόζουμε κάποιους κανόνες περιορισμού του δένδρου είτε πραγματοποιούμε κλάδεμα (pruning).
Οι κανόνες περιορισμού του δένδρου αποτελούν παραμέτρους του μοντέλου που δημιουργούμε. Πιο συγκεκριμένα, μπορούμε να περιορίσουμε το βάθος του δένδρου (max depth), δηλαδή την μέγιστη απόσταση ρίζας και φύλλων, να ορίσουμε τον ελάχιστο αριθμό παρατηρήσεων που πρέπει να έχει ένας κόμβος για να μπορεί να σπάσει (min samples split), δηλαδή να γίνει ενδιάμεσος αντί για φύλλο καθώς και να προσδιορίσουμε τον ελάχιστο αριθμό παρατηρήσεων που πρέπει να περιέχουν τα φύλλα (min samples leaf) για να αποφύγουμε διασπάσεις. Με αυτούς τους κανόνες το δένδρο περιορίζεται κατά την κατασκευή του.
Η προσέγγιση του κλαδέματος αφορά αλλαγές σε ένα ήδη δημιουργηθέν πολύπλοκο δένδρο (T_0). Για το κλάδεμα ακολουθούμε τον αλγόριθμο *Minimal Cost-Complexity Pruning*. Ο αλγόριθμος αυτός παραμετροποιείται από την παράμετρο πολυπλοκότητας a. Για ένα υποδένδρο T, ορίζουμε το μέτρο κόστους-πολυπλοκότητας:

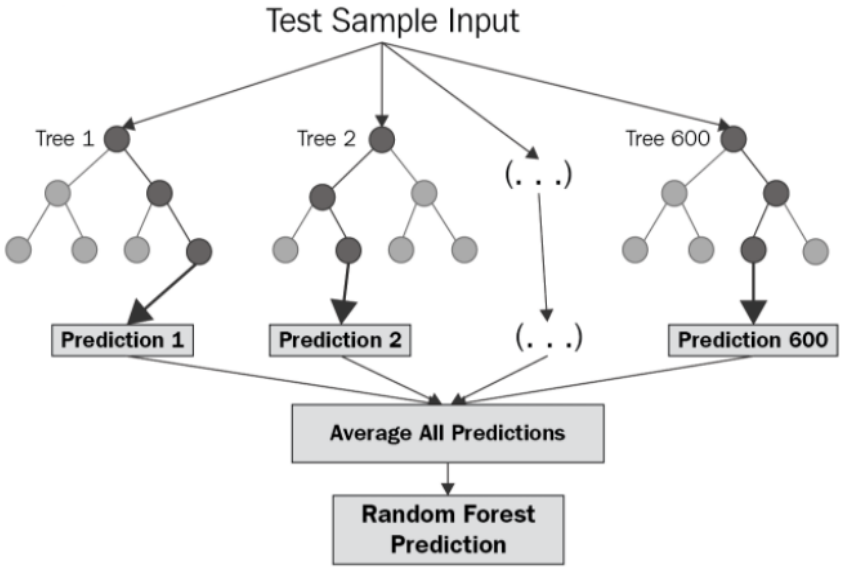
$$R_a(T)=R(T)+a|T|$$

$$\text{με } R(T)=\sum_{m=1}^{|T|}\sum_{i:x_i\in R_m}(y_i-\hat{y}_{R_m})^2$$

Σημειώνεται ότι ως $|T|$ θεωρούμε το πλήθος των φύλλων του δένδρου T, R_m ένα σύνολο δεδομένων ενός φύλλου εντός του δένδρου και $\hat{y}_{R_m}$ η πρόβλεψη για το φύλλο. Ο αλγόριθμος εντοπίζει το υποδένδρο το οποίο ελαχιστοποιεί το μέτρο κόστους-πολυπλοκότητας. Για a=0 λαμβάνουμε το αρχικό υποδένδρο. Αν αυξήσουμε το a εμφανίζεται ποινή για το πλήθος των φύλλων, άρα τιμωρείται η πολυπλοκότητα και συνεπώς οδηγούμαστε σε ένα απλούστερο υποδένδρο ως απάντηση. Έστω ενδιάμεσος κόμβος t, που ορίζει το υποδένδρο T_t. Ορίζουμε ως a_{eff} την τιμή για την οποία $R_a(T_t)=R_a(t)$, δηλαδή ταυτίζονται το κόστος-πολυπλοκότητας του κόμβου σαν μονάδα και του αθροίσματος των τερματικών που ανήκων στο υποδένδρο που ορίζει. Για να παράξει το τελικό αποτέλεσμα ο αλγόριθμος υπολογίζει και αφαιρεί κάθε φορά τον ενδιάμεσο κόμβο που έχει το μικρότερο a_{eff} πάνω από ένα όριο το οποίο αποτελεί και το κριτήριο τερματισμού του αλγορίθμου.[3]
Τα δένδρα απόφασης χαρακτηρίζονται από την απλότητά τους και την ευκολία ανάλυσής τους. Ο χρήστης δεν χρειάζεται να έχει στατιστικές γνώσεις και κάποιο ιδιαίτερο μαθηματικό υπόβαθρο για να μπορέσει να διαβάσει και να κατανοήσει ένα δένδρο απόφασης. Ωστόσο, τα δένδρα απόφασης εμφανίζουν συχνά το φαινόμενο της υπερπροσαρμογής, ενώ γενικά από μόνα τους δεν θεωρούνται τόσο αποδοτικά όσο άλλες πιο σύνθετες μεθόδοι μηχανικής μάθησης.

Τυχαίο Δάσος

Για να ξεπεραστεί το ζήτημα της υστέρησης στο κομμάτι της απόδοσης, χρησιμοποιούμε έναν συνδυασμό πολλών δένδρων απόφασης. Η μέθοδος του *τυχαίου δάσους* (Random Forest) ακολουθεί αυτήν την λογική. Πριν αναλύσουμε τον αλγόριθμο αυτόν, θα αναφερθούμε στην διαδικασία της ενσάκισης (bagging). Σε αυτήν την περίπτωση, έχουμε πολλά σύνολα εκπαίδευσης με κάθε ένα από τα οποία εκπαιδεύουμε ένα ξεχωριστό μοντέλο (δένδρο απόφασης). Ως πρόβλεψη παίρνουμε τον μέσο όρο των προβλέψεων των μοντέλων. Τα σύνολα εκπαίδευσης προέρχονται από το αρχικό μας σύνολο με την μέθοδο της δειγματοληψίας με αντικατάσταση. Η πρόβλεψη με δένδρα απόφασης παρουσιάζει υψηλή μεταβλητότητα. Αν εκπαιδεύσουμε ένα δένδρο απόφασης με το ήμισυ των δεδομένων εκπαίδευσης και εκπαιδεύσουμε ένα δεύτερο με τα υπόλοιπα δεδομένα, τότε η πρόβλεψη τους για την ίδια είσοδο πιθανότατα θα διαφέρει σημαντικά. Με την ενσάκιση αντιμετωπίζουμε αυτό το ζήτημα. Η διαδικασία αυτή γενικά οδηγεί σε προβλέψεις μεγαλύτερης ακρίβειας.
Η μέθοδος του τυχαίου δάσους χρησιμοποιεί την διαδικασία της ενσάκισης, διαφοροποιώντας την όμως στο κομμάτι της κατασκευής των δένδρων απόφασης. Πιο συγκεκριμένα, σε κάθε βήμα κατά την δημιουργία της κάθε μίας συνθήκης διαχωρισμού επιλέγουμε μεταξύ ενός υποσυνόλου των μεταβλητών εισόδου για την ανισότητα που θα φτιάξουμε. Το υποσύνολο αυτό επιλέγεται τυχαία από το σύνολο των μεταβλητών εισόδου. Σε κάθε βήμα φτιάχνεται ένα διαφορετικό τέτοιο υποσύνολο. Το πλήθος των μεταβλητών εισόδου εντός του κάθε υποσυνόλου επιλέγεται συνήθως να είναι η τετραγωνική ρίζα του πλήθους των εισόδων των δειγμάτων μας ή ο λογάριθμος του πλήθους τους ή σε κάθε περίπτωση ένα ποσοστό του. Με τον τρόπο αυτό εξασφαλίζουμε την ύπαρξη ποικιλίας δενδρών απόφασης. Η διαδικασία της πρόβλεψης στα τυχαία δάση ταυτίζεται με την διαδικασία πρόβλεψης που περιγράφηκε παραπάνω στην ενσάκιση.



Σχήμα 4.5: Ένα τυχαίο δάσος με 600 δένδρα απόφασης [9]

Ενίσχυση Κλίσης

Μια άλλη προσέγγιση για την βελτίωση της απόδοσης των δένδρων απόφάσεων είναι η ενίσχυση (boosting). Σε αντίθεση με την ενσάκιση, όπου κάθε δένδρο απόφασης είναι ανεξάρτητο των υπολοίπων, σε αυτή την περίπτωση τα δένδρα αναπτύσσονται με ακολουθιακή λογική βασιζόμενα στα προηγούμενα δένδρα που έχουν δημιουργηθεί. Κάθε

δένδρο φτιάχνεται επί μίας εκδοχής του συνόλου δεδομένων εκπαίδευσης.[48] Η λογική της ενίσχυσης είναι πως αφού φτιάξουμε ένα δένδρο απόφασης, χρησιμοποιούμε τα σφάλματά του, προκειμένου να φτιάξουμε το επόμενο δένδρο, ώστε να κάνουμε πρόβλεψη και γι'αυτά, ενισχύοντας έτσι το πρώτο δένδρο. Η συμμετοχή του καινούργιου δένδρου καθορίζεται από μια παράμετρο που λέγεται ρυθμός μάθησης (learning rate). Το κάθε καινούργιο δένδρο επιλέγεται να έχει συγκεκριμένο μέγεθος. Η διαδικασία αυτή συνεχίζεται για πολλά δένδρα. Υπάρχουν διάφοροι αλγόριθμοι που χρησιμοποιούν αυτήν την προσέγγιση, όπως ο AdaBoost, ο XGBoost και ο Gradient Boosting. Εμείς θα ασχοληθούμε με τον τελευταίο. Ο αλγόριθμος *ενίσχυσης κλίσης* (Gradient Boosting) χρησιμοποιεί όλες τις ιδέες που περιγράφηκαν παραπάνω. Καταρχάς, επιλέγουμε την συνάρτηση σφάλματος L την οποία επιδιώκουμε να ελαχιστοποιήσουμε. Η συνάρτηση αυτή μετράει πόσο απέχει η πρόβλεψη από την πραγματική τιμή. Η πιο σύνηθης επιλογή είναι το τετράγωνο του υπολοίπου. Άλλη επιλογή είναι η απόλυτη διαφορά. Ο αλγόριθμος αποτελείται από βήματα, στο κάθε ένα από τα οποία φτιάχνουμε ένα δένδρο απόφασης και ανανεώνουμε την συνάρτηση πρόβλεψής μας, $f_m(x)$. Ο αλγόριθμος μπορεί να περιγραφεί ως εξής:

1. Αρχικοποιούμε την συνάρτηση πρόβλεψής μας. Σε αυτήν την περίπτωση επιλέγουμε ως πρόβλεψη για όλα τα N δείγματα εκπαίδευσης μία σταθερα. Η σταθερά προκύπτει ως κάτωθι:

$$f_0(x) = argmin_\rho \sum_{i=1}^{N} L(y_i, \rho)$$

   Για την περίπτωση του τετραγώνου του υπολοίπου, η σταθερά ισούται με τον μέσο όρο των εξόδων των δειγμάτων. Άλλη επιλογή είναι να θέσουμε την αρχική πρόβλεψη ίση με μηδέν. Μια άλλη ιδέα είναι να τρέξουμε έναν άλλο αλγόριθμο πρόβλεψης, π.χ. Random Forest, και να θέσουμε το αποτέλεσμα ως αρχική πρόβλεψη.[28]

2. Για κάθε έναν από τα προς δημιουργία δένδρο αποφάσης υπολογίζουμε την αρνητική παράγωγο της συνάρτησης σφάλματος, βάσει αυτής φτιάχνουμε ένα δένδρο απόφασης και ύστερα σύμφωνα με το δένδρο ανανεώνουμε την συνάρτηση πρόβλεψης. Ειδικότερα, αρχικά υπολογίζουμε:

$$z_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f_(x_i)}\right]_{f=f_{m-1}}$$

   Για την περίπτωση του τετραγώνου του υπολοίπου, η τιμή αυτή ταυτίζεται με το απλό σφάλμα. Το δένδρο που δημιουργούμε βάσει των δειγμάτων εκπαίδευσης με εξόδο την z_{im} αντί της y_i, υπακούει στους κανόνες που θέτουμε για το μέγεθός του (βάθος κλπ).

   Η συνάρτηση προβλέψεων ανανεώνεται σύμφωνα με τα παρακάτω[53]:

$$f_m(x) = f_{m-1}(x) + \lambda \sum_{j=1}^{J} \rho_{jm} I(x \in R_{jm})$$

$$\rho_{jm} = argmin_\rho \sum_{x_i \in R_{jm}} L\left(y_i, f_{m-1}(x_i) + \sum_{j=1}^{J} \rho I(x \in R_{jm})\right)$$

$$I(x \in R_{jm}) = \begin{cases} 1, \text{ αν } x \in R_{jm} \\ 0, \text{ αλλιώς} \end{cases}$$

   Στους παραπάνω τύπους, όπου R_{jm} τα σύνολα από τις συνθήκες διαχωρισμού στο δένδρο απόφασης m, ρ_{jm} η έξοδος του αντίστοιχου συνόλου (μέσος όρος των εξόδων των παρατηρήσεων αν μιλάμε για την περίπτωση του τετραγώνου του υπολοίπου ως συνάρτηση κόστους) και λ ο ρυθμός εκπαίδευσης.

3. Για να παράξουμε την ζητούμενη πρόβλεψη για ένα δείγμα του συνόλου ελέγχου, χρησιμοποιούμε την τελική μορφή που έλαβε η συνάρτηση πρόβλεψης.

Σχετικά με τις παραμέτρους του αλγορίθμου σημειώνεται ότι μικρότεροι ρυθμοί εκπαίδευσης τείνουν να οδηγήσουν σε καλύτερα αποτελέσματα, αλλά μέσω περισσότερων επαναλήψεων. Εάν το πλήθος των επαναλήψεων είναι υπερβολικά μεγάλο ο αλγόριθμος εμφανίζει θέματα υπερπροσαρμογής.[28]

4.5 Η πρόβλεψη χρονοσειρών ως πρόβλημα μηχανικής μάθησης

Παραπάνω είδαμε έξι αλγορίθμους μηχανικής μάθησης που ανήκουν στην κατηγορία της επιβλεπόμενης μάθησης. Αυτό σημαίνει πως η εκπαίδευσή τους απαιτεί ένα σύνολο εκπαίδευσης που αποτελείται από δείγματα με μία σειρά εισόδου και μία έξοδο. Τα δεδομένα μας είναι σε μορφή χρονοσειράς. Προκειμένου λοιπόν να χρησιμοποιήσουμε τους αλγορίθμους μηχανικής μάθησης και τις δομές της βαθιάς μάθησης, θα κάνουμε μία μετατροπή των χρονοσειρών ώστε να δημιουργηθούν τα δείγματα του συνόλου εκπαίδευσης. Πιο συγκεκριμένα, ως εισόδους θα έχουμε παρελθοντικές παρατηρήσεις και ως έξοδο την αμέσως επόμενη. Έτσι, θα σπάσουμε την χρονοσειρά σε ένα σύνολο τέτοιων δειγμάτων. Για να παράξουμε πρόβλεψη, αφού έχουμε εκπαιδεύσει το μοντέλο μας, δίνουμε τις τελευταίες χρονικά τιμές ως είσοδο πριν την στιγμή για την οποία ζητούμε πρόβλεψη. Το πλήθος των μεταβλητών εισόδου, δηλαδή των παρελθοντικών παρατηρήσεων είναι δική μας επιλογή. Σχηματικά, έχουμε αρχικά τα δεδομένα μας στην μορφή χρονοσειράς:

y_1	y_2	y_3	...	y_{N-1}	y_N

Τα δεδομένα μας σε μορφή δειγμάτων (θεωρούμε 4 μεταβλητές εισόδου):

y_1	y_2	y_3	y_4	y_5
y_2	y_3	y_4	y_5	y_6
y_3	y_4	y_5	y_6	y_7
...	...	...	...	...
y_{N-5}	y_{N-4}	y_{N-3}	y_{N-2}	y_{N-1}
y_{N-4}	y_{N-3}	y_{N-2}	y_{N-1}	y_N

Μετά την διπλή κατακόρυφη γραμμή σε κάθε γραμμή (δείγμα) τοποθετείται η αντίστοιχη έξοδος. Για παράδειγμα, στο πρώτο δείγμα ως μεταβλητές εισόδου έχουμε τις παρατηρήσεις y_1, y_2, y_3 και y_4, ενώ ως μεταβλητή εξόδου την y_5. Πριν προχωρήσουμε στο κομμάτι της πρόβλεψης, θα δημιουργήσουμε δύο παραλλαγές των άνω δειγμάτων. Η πρώτη θα έχει ως έξοδο κάθε δείγματος όχι την αμέσως επόμενη χρονικά παρατήρηση, αλλά την παρατήρηση που βρίσκεται τρεις χρονικές περιόδους ύστερα. Φυσικά, μπορούμε να γενικεύσουμε για οποιαδήποτε χρονική απόσταση εισόδων και εξόδου. Η δεύτερη θα έχει πολλαπλές εξόδους, τις επόμενες δυο χρονικά, από τις εισόδους, παρατηρήσεις. Είναι σαφές ότι αυτό μπορεί να γενικευτεί και για περισσότερες των δύο εξόδους. Οι δύο αυτές παραλλαγές θα χρησιμοποιηθούν παρακάτω σε κάποιες στρατηγικές που θα ακολουθήσουμε για την παραγωγή προβλέψεων.

Η πρώτη παραλλαγή:

y_1	y_2	y_3	y_4	y_7
y_2	y_3	y_4	y_5	y_8
y_3	y_4	y_5	y_6	y_9
...	...	...	...	...
y_{N-7}	y_{N-6}	y_{N-5}	y_{N-4}	y_{N-1}
y_{N-6}	y_{N-5}	y_{N-4}	y_{N-3}	y_N

Βλέπουμε πως σε αυτήν την περίπτωση έχουμε λιγότερα δείγματα σε σχέση με πριν.

Η δεύτερη παραλλαγή:

y_1	y_2	y_3	y_4	y_5	y_6
y_2	y_3	y_4	y_5	y_6	y_7
y_3	y_4	y_5	y_6	y_7	y_8
...	...	...	...	...	...
y_{N-6}	y_{N-5}	y_{N-4}	y_{N-3}	y_{N-2}	y_{N-1}
y_{N-5}	y_{N-4}	y_{N-3}	y_{N-2}	y_{N-1}	y_N

Και εδώ τα δείγματα είναι λιγότερα σε σχέση με την αρχική μας προσέγγιση.

Θεωρούμε πως θέλουμε να κάνουμε πρόβλεψη με ορίζοντα H. Για $H = 1$ προβλέπουμε μόνο την επόμενη χρονική στιγμή, ενώ για $H > 1$ προβλέπουμε για περισσότερες χρονικές στιγμές. Σε κάθε περίπτωση θεωρούμε πως έχουμε d παρελθοντικές τιμές ως είσοδο. Το κάθε μοντέλο μπορεί να αναπαρασταθεί ως μία συνάρτηση που λαμβάνει την είσοδο και παράγει ως έξοδο μία πρόβλεψη. Το μοντέλο κατά την εκπαίδευση θα το συμβολίσουμε ως f, ενώ μόλις έχει εκπαιδευτεί και έχει διαμορφωθεί πλήρως ως $\hat{f}$. Ως y_i θα συμβολίζουμε τις παρατηρήσεις, ενώ ως $\hat{y}_i$ τις προβλέψεις. Με N περιγράφουμε το πλήθος των τιμών της χρονοσειράς μας. Για την παραγωγή των προβλέψεων με αλγορίθμους μηχανικής μάθησης υπάρχουν τέσσερις βασικές στρατηγικές:

1. Η *αναδρομική* (recursive), όπου εκπαιδεύουμε ένα μοντέλο για να κάνει προβλέψεις με ορίζοντα ένα, δηλαδή για να προβλέπει την τιμή της χρονοσειράς μόνο για την επόμενη χρονική στιγμή. Η εκπαίδευση γίνεται με τα πρώτα δείγματα που εξετάσαμε. Στην στρατηγική αυτή, κάθε πρόβλεψη που κάνουμε τροφοδοτείται στο μοντέλο ως είσοδος για την επόμενη πρόβλεψη. Πιο συγκεκριμένα, αφού έχουμε ολοκληρώσει την εκπαίδευση του μοντέλου μας, για να προβλέψουμε τις H ζητούμενες τιμές, αρχικά προβλέπουμε για ορίζοντα ένα χρησιμοποιώντας ως είσοδο τις d τελευταίες τιμές της χρονοσειράς. Ύστερα, για την επόμενη χρονική στιγμή παίρνουμε ως είσοδο τις d-1 τελευταίες τιμές της χρονοσειράς και την πρώτη πρόβλεψη. Για την τρίτη πρόβλεψη, τις d-2 τιμές της χρονοσειράς και τις δύο προβλέψεις. Η διαδικασία αυτή συνεχίζεται μέχρι να κάνουμε και τις H ζητούμενες προβλέψεις. Τα μοντέλα αυτής της στρατηγικής είναι εκτεθιμένα στα σφάλματα εκτίμησης καθότι όσο πιο βαθιά πηγαίνουμε χρονικά στις προβλέψεις μας τόσο περισσότερο χρησιμοποιούμε προβλέψεις αντί για πραγματικές τιμές. Μαθηματικά, η εκπαίδευση του μοντέλου εκφράζεται ως εξής[45]:

$$y_{t+1} = f(y_t, y_{t-1}, ..., y_{t-d+1}), \text{ για t} \in \{d, ..., N-1\}$$

Η διαδικασία πρόβλεψης εκφράζεται ως εξής:

$$\hat{y}_{N+h} = \begin{cases} \hat{f}(y_N, ..., y_{N-d+1}), \text{ αν } h = 1 \\ \hat{f}(\hat{y}_{N+h-1}, ..., \hat{y}_{N+1}, y_N, ..., y_{N-d+h}), \text{ αν } h \in \{2, ..., d\} \\ \hat{f}(\hat{y}_{N+h-1}, ..., \hat{y}_{N+h-d}), \text{ αν } h \in \{d+1, ..., H\} \end{cases}$$

Στον πρώτο κλάδο περιγράφεται η πρώτη πρόβλεψη. Στον δεύτερο έχουμε προβλέψεις με είσοδο τόσο από πραγματικές παρελθοντικές τιμές της χρονοσειράς όσο και από προηγούμενες προβλέψεις. Στον τρίτο κλάδο έχουμε προβλέψεις με χρήση μόνο παλαιών προβλέψεων ως εισόδους.

2. Η *απευθείας* (direct), όπου εκπαιδεύουμε Η ανεξάρτητα μοντέλα προκειμένου το καθένα να μάθει να προβλέπει για μία ξεχωριστή περίοδο του ορίζοντα. Η εκπαίδευση γίνεται με δείγματα που ακολουθούν την λογική της μορφής της πρώτης παραλλαγής που δείξαμε παραπάνω. Εν συνεχεία, για να κάνουμε πρόβλεψη με ορίζοντα Η, κάνουμε μία πρόβλεψη με το κάθε μοντέλο για κάθε περίοδο του ορίζοντα και συνενώνουμε τις προβλέψεις. Η στρατηγική αυτή δεν συσσωρεύει τα σφάλματα από τις προηγούμενες προβλέψεις καθώς σε όλα τα μοντέλα δίνονται ως δεδομένα εισόδου για την πρόβλεψη οι τελευταίες d παρατηρήσεις της χρονοσειράς, ωστόσο τα μοντέλα δεν λαμβάνουν υπόψιν πιθανές συσχετίσεις μεταξύ των προβλέψεων. Ένα άλλο μειονέκτημα της στρατηγικής αυτής είναι πως απαιτεί πολλούς υπολογιστικούς πόρους καθώς έχουμε Η μοντέλα.[10] Για κάθε ένα μοντέλο (f_h) έκφράζουμε την εκπαίδευσή του ως εξής:

$$y_{t+h} = f_h(y_t, y_{t-1}, ..., y_{t-d+1}), \text{ για t} \in \{d, ..., N-H\} \text{ και για h} \in \{1, ..., H\}$$

Οι προβλέψεις του καθενός μοντέλου μπορούν να αποτυπωθούν ως εξής:

$$\hat{y}_{N+h} = \hat{f}_h(y_N, y_{t-1}, ..., y_{N-d+1}), \text{ για h} \in \{1, ..., H\}$$

3. Η απευθείας-αναδρομική (DirRec), η οποία συνδυάζει τις προηγούμενες δύο. Όπως η απευθείας στρατηγική, υπολογίζει τις προβλέψεις με διαφορετικό μοντέλο για κάθε τιμή του ορίζοντα, αλλά σε κάθε βήμα μεγενθύνει το σύνολο των εισόδων προσθέτωντας ως μεταβλητή την προηγούμενη προβλέψη, ακολουθώντας την λογική της αναδρομικής στρατηγικής. Κάθε μοντέλο παράγει και πάλι μία πρόβλεψη-έξοδο αλλά έχει διαφορετικό αριθμό εισόδων. Ειδικότερα, κάθε επόμενο μοντέλο έχει μία είσοδο παραπάνω από το προηγούμενο. Για την διαδικασία πρόβλεψης, κάθε μοντέλο μεταδίδει στο επόμενο την πρόβλεψη που έκανε για να την χρησιμοποιήσει ως είσοδο μαζί με τις υπόλοιπες εισόδους. Όπως και στην απευθείας στρατηγική η τελική πρόβλεψη προκύπτει με συνένωση των προβλέψεων των Η μοντέλων.

   Για την εκπαίδευση των μοντέλων έχουμε:

$$y_{t+h} = f_h(y_{t+h-1}, ..., y_{t-d+1}), \text{ για t} \in \{d, ..., N-H\} \text{ και για h} \in \{1, ..., H\}$$

   Για την πρόβλεψη των τιμών από κάθε μοντέλο μπορούμε να γράψουμε:

$$\hat{y}_{N+h} = \begin{cases} \hat{f}_h(y_N, y_{t-1}, ..., y_{N-d+1}), \text{ για } h = 1 \\ \hat{f}_h(\hat{y}_{N+h-1}, ..., \hat{y}_{N+1}, y_N, y_{t-1}, ..., y_{N-d+1}), \text{ για h} \in \{2, ..., H\} \end{cases}$$

   Όπως βλέπουμε και από τους παραπάνω τύπους, η πρώτη πρόβλεψη γίνεται με χρήση των d τελευταίων τιμών της χρονοσειράς. Η επόμενη με τις τιμές αυτές και την πρώτη πρόβλεψη. Η τρίτη με την προηγούμενη είσοδο και την δεύτερη πρόβλεψη και ούτω καθεξής. Η εκπαίδευση των Η μοντέλων, προφανώς γίνεται με αντίστοιχο αριθμό εισόδων, με την διαφορά πως οι είσοδοι είναι μόνο παρελθοντικές τιμές της χρονοσειράς.

4. Η στρατηγική των *πολλαπλών εισόδων-πολλαπλών εξόδων* (Multiple-Input Multiple-Output). Οι προηγούμενες στρατηγικές μπορούν να χαρακτηριστούν ως στρατηγικές μίας εξόδου καθώς τα μοντέλα που δημιουργούμε σ'αυτές έχουν πάντοτε μία έξοδο. Η λογική της αντιστοίχησης των εισόδων σε μία έξοδο δεν δίνει

έμφαση στην ύπαρξη στοχαστικών εξαρτήσεων μεταξύ των μελλοντικών τιμών, κάτι που επηρεάζει την ακρίβεια πρόβλεψης[45]. Η λογική της στρατηγικής πολλαπλών εισόδων-πολλαπλών εξόδων είναι να αποτυπώσει αυτές τις στοχαστικές εξάρτησεις που χαρακτηρίζουν την χρονοσειρά και στις προβλεπόμενες τιμές. Στην στρατηγική αυτή εκπαιδεύουμε ένα μοντέλο που μαθαίνει να παράγει εξόδο διαστάσης Η. Τα δείγματα εκπαίδευσης έχουν την μορφή της δεύτερης παραλλαγής.

Η εκπαίδευση του μοντέλου μας γίνεται σύμφωνα με τον ακόλουθο τύπο:

$$[y_{t+H}, ..., y_{t+1}] = f(y_t, y_{t-1}, ..., y_{t-d+1}), \text{ για t} \in \{d, ..., N - H\}$$

Το μοντέλο πραγματοποιεί όλες τις προβλέψεις σε ένα βήμα με το παρακάτω τύπο:

$$[\hat{y}_{t+H}, ..., \hat{y}_{t+1}] = \hat{f}(y_N, y_{N-1}, ..., y_{N-d+1})$$

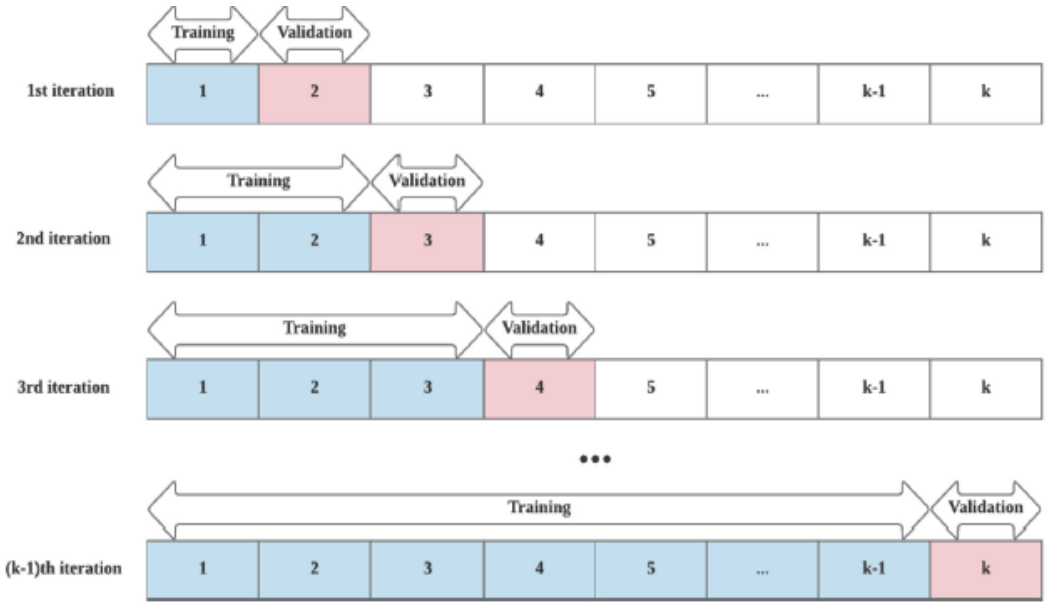
Αξίζει να σημειώσουμε πως μπορούμε να συνδυάσουμε την απευθείας στρατηγική με την στρατηγική πολλαπλών εισόδων-πολλαπλών εξόδων. Μία στρατηγική που προβλέψεων που προκύπτει από τον συνδυασμό των DirRec και των πολλαπλών εισόδων-πολλαπλών εξόδων είναι η DIRMO. Σ'αυτήν την περίπτωση, χωρίζουμε τον ορίζοντα σε blocks και για κάθε τέτοιο προβλέπουμε με την χρήση ενός μοντέλου της λογικής DirRec. Συνεπώς, για μέγεθος block s, θα έχουμε H/s τέτοια μοντέλα. Όπως μπορεί κανείς να αντιληφθεί, για διαφορετικές τιμές της παραμέτρου s ($s \in \{1, 2, ..., H\}$) μπορούμε να φτιάξουμε διαφορετικές στρατηγικές. Αν $s = 1$, τότε είμαστε στην περίπτωση της DirRec ενώ για $s = H$ έχουμε στρατηγική πολλαπλών εισόδων-πολλαπλών εξόδων.

Όπως είδαμε στην σχετική παράγραφο, οι έξι αλγόριθμοι μηχανικής μάθησης έχουν μία ή περισσότερες εισόδους και μία έξοδο. Ως εκ τούτου, για να παράξουμε προβλέψεις με ορίζοντα μεγαλύτερο της μονάδας με αυτούς, θα ακολουθήσουμε μία από τις τρείς πρώτες στρατηγικές. Για τα μοντέλα βαθιάς μάθησης μπορούμε να ακολουθήσουμε οποιαδήποτε στρατηγική θέλουμε καθότι ο αριθμός των εξόδων είναι κάτι που μπορούμε να επιλέξουμε ελεύθερα. Στα πλαίσια της παρούσας εργασίας, θα χρησιμοποιήσουμε τις τρεις πρώτες στρατηγικές για τους αλγορίθμους μηχανικής μάθησης και όλες τις στρατηγικές πλην της DirRec για τα νευρωνικά δίκτυα. Κάναμε αυτήν την επιλογή με την λογική πως η στρατηγική πολλαπλών εισόδων-πολλαπλών εξόδων εκφράζει την συσχέτιση μεταξύ των προβλέψεων, όντας όμως ταυτόχρονα πιο αποδοτικός καθώς έχουμε ένα μοντέλο αντί για Η διαφορετικά μοντέλα. Τέλος, για τα υπόλοιπα μοντέλα βαθιάς μάθησης θα χρησιμοποιήσουμε την στρατηγική πολλαπλών εισόδων-πολλαπλών εξόδων.

4.6 Βελτιστοποίηση υπερπαραμέτρων

Από την περιγραφή των αλγορίθμων μηχανικής μάθησης βλέπουμε πως κάθε αλγόριθμος περιλαμβάνει μία σειρά από ρυθμίσεις που ελέγχουν την συμπεριφορά του. Για παράδειγμα, στον αλγόριθμο των κ-κοντινότερων γειτόνων ρυθμίζοντας το πλήθος των κ εξεταζόμενων γειτόνων καθορίζουμε την λειτουργία του. Οι ρυθμίσεις αυτές ονομάζονται *υπερπαράμετροι* (hyperparameters). Είναι σαφές πως διαφορετικές υπερπαραμέτροι οδηγούν τους αλγορίθμους σε καλύτερη ή χειρότερη απόδοση. Συνεπώς, πρέπει να ακολουθήσουμε μια διαδικασία ώστε να επιλέξουμε τις κατάλληλες τιμές για αυτές.
Πριν όμως εξετάσουμε τον τρόπο εύρεσης των καλύτερων τιμών για τις υπερπαραμέτρους των μοντέλων μας, πρέπει να ορίσουμε τον τρόπο με τον οποίο αποτιμάμε την απόδοση των μοντέλων για κάθε συνδυασμό των υπερπαραμέτρων. Ειδικότερα, θέλουμε να έχουμε μία εκτίμηση για το σφάλμα των μοντέλων στο σύνολο ελέγχου και βάσει αυτού να κατατάξουμε τα μοντέλα μας. Για τον σκοπό αυτό θα αναφερθούμε σε κάποιες μεθόδους

επαναδειγματοληψίας. Η βασική λογική των μεθόδων αυτών είναι να κρατούν ένα μέρος του συνόλου εκπαίδευσης εκτός της διαδικασίας προσαρμογής του μοντέλου και να δοκιμάζουν το μοντέλο επί αυτού.
Η πρώτη μέθοδος είναι αυτή του συνόλου επικύρωσης (validation set ή hold out set). Σε αυτήν την περίπτωση, διαχωρίζουμε τυχαία το σύνολο εκπαίδευσης σε δύο μέρη, το σύνολο που θα χρησιμοποιηθεί για την εκπαίδευση του μοντέλου και το σύνολο για την αξιολόγηση του. Τα δύο σύνολα μπορούν να είναι ισομεγέθη ή όχι, δηλαδή να κρατήσουμε λιγότερα δείγματα για την αξιολόγηση του μοντέλου. Με τον τυχαίο διαχωρισμό εννοούμε πως παίρνουμε τυχαία δείγματα απ'το σύνολο εκπαίδευσης για το κάθε υποσύνολο. Το σφάλμα του μοντέλου στο σύνολο επικύρωσης αποτελεί την εκτίμηση για το σφάλμα στο σύνολο ελέγχου. Η μέθοδος αυτή είναι ιδιαιτέρως απλή, αλλά το αποτέλεσμά της εξαρτάται σε πολύ μεγάλο βαθμό από τον διαχωρισμό των δειγμάτων, συνεπώς διακατέχεται από τυχαιότητα. Για να ξεπεράσουμε αυτό το ζήτημα εισάγουμε τις μεθόδους διασταυρούμενης επικύρωσης (cross-validation). Η μέθοδος Leave-One-Out Cross-Validation (LOOCV), κρατάει σε κάθε επανάληψη ένα δείγμα για αξιολόγηση και χρησιμοποιεί τα υπόλοιπα για εκπαίδευση. Στην πρώτη επανάληψη κρατάει το πρώτο, στην δεύτερη το δεύτερο κ.ο.κ. Το συνολικό σφάλμα επικύρωσης είναι ο μέσος όρος των σφαλμάτων που υπολογίστηκαν σε όλες τις επαναλήψεις. Η μέθοδος αυτή είναι υπολογιστικά κοστοβόρα, διότι εκπαιδεύει ένα μοντέλο τόσες φορές όσο το μέγεθος του συνόλου εκπαίδευσης. Εναλλακτικά της LOOCV, υπάρχει η μέθοδος k-Fold Cross-Validation. Στην μέθοδο αυτή, χωρίζουμε τυχαία (ίδια λογική με το hold out set) το σύνολο εκπαίδευσης σε k κομμάτια. Στην πρώτη επανάληψη χρησιμοποιούμε το πρώτο κομμάτι για την αξιολόγηση του μοντέλου και τα υπόλοιπα k-1 για την εκπαίδευση. Στην δεύτερη τον ρόλο του συνόλου επικύρωσης παίρνει το δεύτερο κομμάτι και για την εκπαίδευση χρησιμοποιούμε τα υπόλοιπα συμπεριλαμβανομένου του πρώτου κομματιού. Η λογική αυτή ακολουθείται για k επαναλήψεις. Ως σφάλμα επικύρωσης επιστρέφεται ο μέσος όρος των σφαλμάτων κάθε επανάληψης. Παρατηρούμε ότι για k ίσο με το πλήθος δειγμάτων του συνόλου εκπαίδευσης η k-Fold Cross-Validation συμπίπτει με την LOOCV[48]
Οι παραπάνω τεχνικές για την αξιολόγηση των μοντέλων χρειάζονται κάποιες αλλαγές για να χρησιμοποιηθούν στα δεδομένα μας. Επειδή τα δεδομένα μας έχουν συγκεκριμένη χρονολογική σειρά δεν μπορούμε να δημιουργήσουμε τα υποσύνολα με τυχαίο τρόπο, όπως στις παραπάνω περιπτώσεις. Αυτό που κάνουμε είναι απλώς να δημιουργούμε τα υποσύνολα επί των χρονολογικά ταξινομημένων δεδομένων. Άρα, στα δεδομένα του πρώτου κομματιού του k-Fold Cross-Validation έχουμε δεδομένα που είναι παλαιότερα από αυτά του δεύτερου κ.ο.κ. Με την ίδια λογική, στο κομμάτι του συνόλου επικύρωσης έχουμε μεταγενέστερα δεδομένα από το κομμάτι του συνόλου εκπαίδευσης. Πέραν όμως από αυτό, στην ίδια λογική, η μέθοδος k-Fold Cross-Validation απαιτεί μία ακόμη τροποποίηση. Πιο συγκεκριμένα, μπορούμε να χρησιμοποιήσουμε σε κάθε επανάληψη για εκπαίδευση μόνο τα κομμάτια που βρίσκονται πριν από αυτά που χρησιμοποιούνται για αξιολόγηση. Η τροποποίηση αυτή λέγεται k-Fold Forward Cross-Validation. Σε αυτήν την περίπτωση χρειάζεται προσοχή, καθώς για μεγάλες τιμές του k, στις πρώτες επαναλήψεις τα σύνολα εκπαίδευσης θα είναι ιδιαιτέρως μικρά.[52].
Σε κάποιες περιπτώσεις η επιλογή των βέλτιστων υπερπαραμέτρων μπορεί να γίνει από τον ίδιο τον χρήστη. Για να γίνει αυτό είναι απαραίτητο να υπάρχει ένα πολύ καλό σημείο εκκίνησης. Με άλλα λόγια, πρέπει ο χρήστης να γνωρίζει πάνω-κάτω κάποιες τιμές γύρω από τις οποίες θα κίνηθει αναζητώντας την βέλτιστη λύση. Το αρχικό αυτό σημείο προκύπτει εμπειρικά με την χρήση παρόμοιων μοντέλων σε παρεμφερή προβλήματα για μεγάλο χρονικό διάστημα. Επειδή όμως το αρχικό αυτό σημείο συνήθως δεν υπάρχει, προσανατολιζόμαστε σε αυτόματες λύσεις.
Η λογική βάσει της οποίας κινούμαστε είναι να βρούμε τον συνδυασμό υπερπαραμέτρων για τις οποίες βελτιστοποιείται μια αντικειμένική συνάρτηση, κάποιες φορές υπό περιορισμούς (π.χ. χρόνος εκτέλεσης). Πιο συγκεκριμένα, δοκιμάζουμε συνδυασμούς για να βρούμε



Σχήμα 4.6: k-Fold Forward Cross-Validation [52]

αυτόν που ελαχιστοποιεί το σφάλμα επικύρωσης. Υπάρχουν δύο βασικές τεχνικές για να επιλέξουμε τον ζητούμενο συνδυασμό υπερπαραμέτρων, η *αναζήτηση πλέγματος* (Grid Search) και η *τυχαία αναζήτηση* (Random Search).

Στην αναζήτηση πλέγματος ο χρήστης επιλέγει ένα πεπερασμένο σύνολο για κάθε υπερπαράμετρο. Ο αλγόριθμος εκπαιδεύει ένα μοντέλο για κάθε έναν δυνατό συνδυασμό τιμών των υπερπαραμέτρων. Ο συνδυασμός που παράγει το χαμηλότερο σφάλμα επικύρωσης επιλέγεται ως ο καλύτερος. Το μεγαλύτερο μειονέκτημα της αναζήτησης πλέγματος είναι πως απαιτεί πολλούς υπολογιστικούς πόρους. Αν έχουμε m υπερπαραμέτρους και n πιθανές τιμές για κάθε μία από αυτές, θα εκπαιδευτούν μοντέλα (και εν συνεχεία αυτά θα αξιολογηθούν) για n^m περιπτώσεις.[20] Συνεπώς, το πλήθος των απαιτούμενων υπολογισμών ακόμη και για μοντέλα με λίγες υπερπαραμέτρους είναι μεγάλο.

Η τυχαία αναζήτηση αποτελεί μία εναλλακτική της αναζήτησης πλέγματος με σημαντικά λιγότερες υπολογιστικές απαιτήσεις. Σε αυτήν την περίπτωση, ορίζουμε μία κατανομή για κάθε υπερπαράμετρο βάσει της οποίας ο αλγόριθμος επιλέγει τυχαία τιμές. Ο αλγόριθμος τυχαίας αναζήτησης επιλέγει ένα συγκεκριμένο πλήθος συνδυασμών υπερπαραμέτρων, που ορίζει ο χρήστης, εκπαιδεύει ένα μοντέλο με κάθε έναν από αυτούς και αξιολογεί την απόδοσή του βάσει του σφάλματος επικύρωσης. Στο τέλος επιστρέφει τον συνδυασμό με την καλύτερη επίδοση. Σε αυτήν την περίπτωση ο χρήστης μπορεί να ορίσει μεγάλα σύνολα πιθανών τιμών των μεταβλητών χωρίς επιβάρυνση του κόστους υπολογισμού καθώς οι επαναλήψεις του αλγορίθμου είναι καθορισμένες.

Κεφάλαιο 5

Πρόβλεψη με Αρχιτεκτονικές Βαθιάς Μάθησης

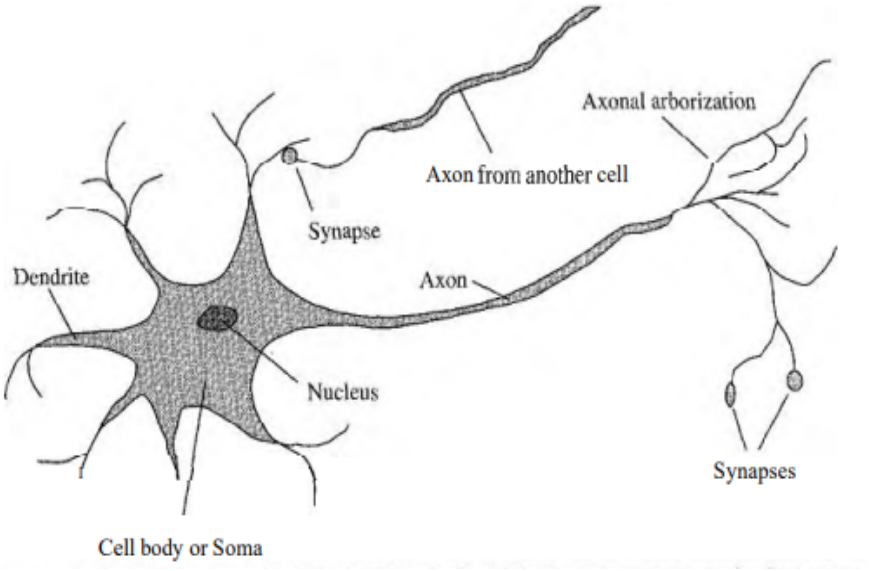
5.1 Νευρωνικά Δίκτυα

Βιολογικός Νευρώνας και Ανθρώπινος Εγκέφαλος

Όπως αναφέρθηκε στο προηγούμενο κεφάλαιο, συνεισφορά στην τεχνητή νοημοσύνη έχουν, μεταξύ άλλων κλάδων, οι νευροεπιστήμες, οι οποίες ασχολούνται με την μελέτη του νευρικού συστήματος και ιδιαίτερα του εγκεφάλου. Η δομική μονάδα του εγκεφάλου είναι το νευρικό κύτταρο ή νευρώνας (neuron). Ένας νευρώνας αποτελείται από έναν κυτταρικό κορμό ή σώμα (soma), που περιέχει έναν κυτταρικό πυρήνα (nucleus). Από τον κυτταρικό κορμό διακλαδίζονται ίνες, που ονομάζονται δενδρίτες (dendrites) και ο άξονας (axon) που αποτελεί μια μεγάλη ίνα. Οι άξονες έχουν μεγάλο μήκος. Οι δενδρίτες αποτελούν τα σημεία εισόδου, ενώ ο άξονας το σημείο εξόδου για τον νευρώνα. Ένας νευρώνας συνδέεται με 10 έως 10.000 νευρώνες, διαμέσου σημείων σύνδεσης που λέγονται συνάψεις (synapses). Τα σήματα διαδίδονται από νευρώνα σε νευρώνα με ηλεκτροχημική αντίδραση. Τα σήματα αυτά έλεγχουν την εγκεφαλική δραστηριότητα βραχυπρόθεσμα, και επιτρέπουν επίσης μακρόχρονες αλλαγές στην θέση και την συνδετικότητα των νευρώνων.[37] Η ικανότητα μάθησης και μνήμης που παρουσιάζει ο εγκέφαλος οφείλεται στην ικανότητα των συνάψεων να μεταβάλουν την αγωγιμότητά τους. Τα ηλεκτρικά σήματα που λαμβάνει ως είσοδο ο νευρώνας συνδυάζονται και αναλόγως αν το αποτελέσμα ξεπερνάει ένα κατώφλι το σήμα διαδίδεται προς τους άλλους νευρώνες. Ο βιολογικός νεωρώνας αποκρίνεται σε χρόνο της τάξης των χιλιοστών του δευτερολέπτου, ωστόσο ο ανθρώπινος εγκέφαλος μπορεί να λειτουργεί εξαιρετικά γρήγορα, καθότι όλοι οι νευρώνες με τις συνάψεις τους είναι ταυτόχρονα ενεργοί. Με άλλα λόγια η υπολογιστική ικανότητα του εγκεφάλου είναι μοιρασμένη σε όλο του τον όγκο, δηλαδή πρόκειται για ένα παράλληλο και κατανεμημένο υπολογιστικό σύστημα.[54]Ο ανθρώπινος εγκέφαλος δομείται από εκατοντάδες δισεκατομμύρια νευρώνες συνεπώς είναι αδύνατον να αντιγράψουμε με ακρίβεια την δομή και την λειτουργία του. Αυτό που κάνουμε στην πράξη είναι να φτιάχνουμε απλούστερα μοντέλα που βασίζονται όμως στις δομές και τις λειτουργίες του εγκεφάλου.

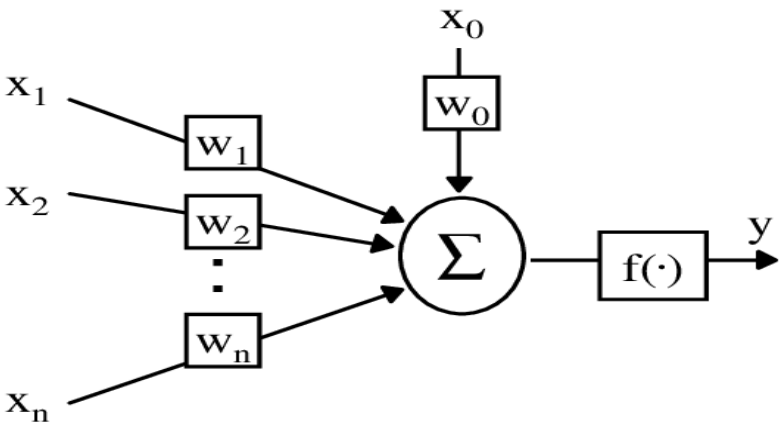
Τεχνητά Νευρωνικά Δίκτυα

Ο τεχνητός νευρώνας αποτελεί το βασικό συστατικό δομικό στοιχείο των τεχνητών νευρωνικών δικτύων με τα οποία θα ασχοληθούμε σε αυτό το κεφάλαιο. Η δομή του βασίζεται στη αυτή του βιολογικού νευρώνα. Πιό συγκεκριμένα, ένας τεχνητός νευρώνας αποτελείται από ένα σύνολο διασυνδέσεων, έναν αθροιστή και μία συνάρτηση ενεργοποίησης. Κάθε διασύνδεση συνοδεύεται από ένα βάρος, το οποίο μπορεί να είναι θετικό ή αρνητικό. Μέσω των διασυνδέσεων ο νευρώνας λαμβάνει σήματα εισόδου τα οποία πολλαπλασιάζονται με τα αντίστοιχα βάρη. Εν συνεχεία, οι σταθμισμένες είσοδοι προστίθονται στον αθροιστή και το αποτέλεσμα δίνεται ως είσοδος στην συνάρτηση ενεργοποίησης του νευρώνα, όπου παράγεται η έξοδος του νευρώνα. Στο μοντέλο του τεχνητού νευρώνα, το οποίο ονομάζεται μοντέλο McCulloch-Pitts, εκτός από τις εισόδους από τους άλλους νευρώνες, θεωρούμε



Σχήμα 5.1: Ο βιολογικός νευρώνας [37]

μία επιπλέον σταθερή είσοδο $x_0 = 1$, που ακολουθείται από το βάρος της w_0, το οποίο λέγεται βάρος πόλωσης (bias weight). Μπορούμε να θεωρήσουμε πως ένας νευρώνας ορίζει ένα υπερεπίπεδο στον χώρο των εισόδων του.



Σχήμα 5.2: Ο τεχνητός νευρώνας [14]

Μαθηματικά, η έξοδος του νευρώνα εκφράζεται ως εξής:

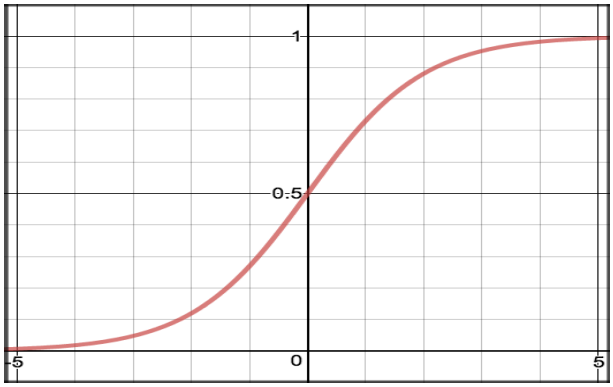
$$y = f(\sum_{i=0}^{n} w_i x_i)$$

Οι βασικότερες συναρτήσεις ενεργοποίησης είναι οι ακόλουθες:

- Η συνάρτηση κατωφλίου (Threshold Function), η οποία δίνει ένα, αν η είσοδος είναι μεγαλύτερη ή ίση του μηδενός και μηδέν διαφορετικά.
- Η σιγμοειδής συνάρτηση (Sigmoid Function), η οποία έχει γραφική παράσταση με σχήμα S. Ορίζεται ως αυστηρά αύξουσα συνάρτηση που επιδεικνύει κομψή ισορροπία μεταξύ γραμμικής και μη γραμμικής συμπεριφοράς[21] Παράδειγμα σιγμοειδούς συνάρτησης αποτελεί η λογιστική με τύπο:

$$f(x) = \frac{1}{1 + \exp(-ax)}$$

Το a είναι ένας συντελεστής που ρυθμίζει την ταχύτητα μετάβασης μεταξύ των δύο ασυμπτωτικών τιμών. Άλλες συναρτήσεις που ανήκουν στην οικογένεια της σιγμοειδούς συνάρτησης είναι η αντίστροφη εφαπτομένη και η υπερβολική εφαπτομένη. Η σιγμοειδής συνάρτηση έχει το πλεονέκτημα ότι είναι διαφορίσιμη.



Σχήμα 5.3: Η λογιστική συνάρτηση για a=1 [5]

- Η γραμμική συνάρτηση (Linear Function) η οποία αποτελεί μια απλή γραμμική αποτύπωση της εισόδου στην έξοδο, δηλαδή ισχύει:

$$f(x) = ax$$

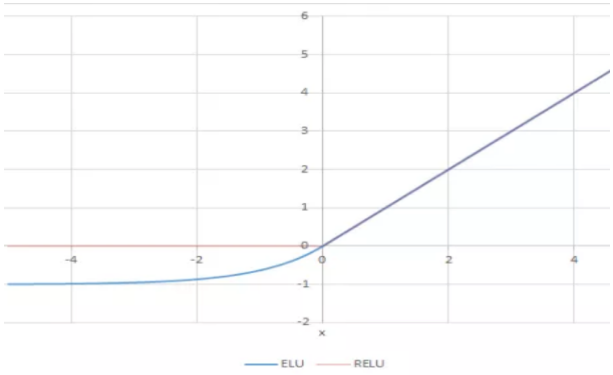
Για a=1, η έξοδος ταυτίζεται με την είσοδο.

- Η συνάρτηση ReLU (Rectified Linear Unit) που περιγράφεται από τον τύπο:

$$f(x) = max\{0, x\}$$

- Η συνάρτηση ELU (Exponential Linear Unit) η οποία ορίζεται ως εξής:

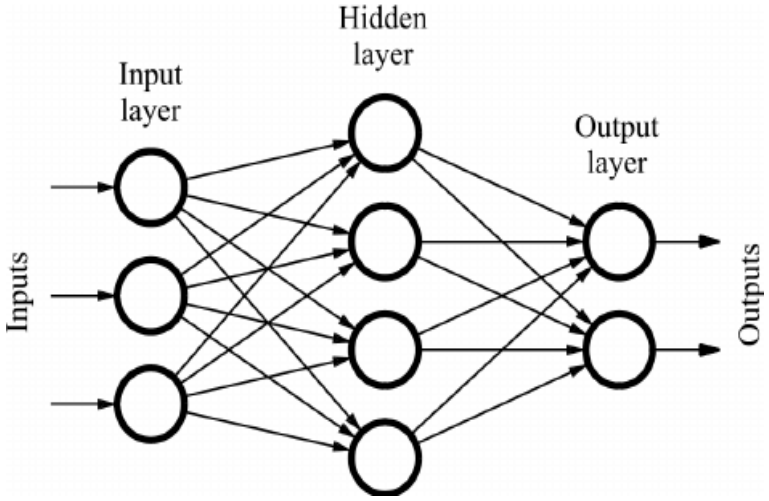
$$f(x) = \begin{cases} x, x > 0 \\ a(\exp(x) - 1), x \leq 0 \end{cases}$$



Σχήμα 5.4: Οι συναρτήσεις ReLU και ELU [5]

Τα τεχνητά νευρωνικά δίκτυα αποτελούνται από ένα πλήθος νευρώνων οργανωμένων σε επίπεδα. Το πρώτο επίπεδο ονομάζεται επίπεδο εισόδου (input layer) και χρησιμοποιείται για την εισαγωγή των δεδομένων. Οι νευρώνες στο επίπεδο αυτό απλώς μεταφέρουν την είσοδο στα επόμενα επίπεδα. Στην συνέχεια, έχουμε κανένα, ένα ή περισσότερα ενδιάμεσα ή κρυφά επίπεδα (hidden layers) ακολουθούμενα από το επίπεδο εξόδου (output layer). Τα

τεχνητά νευρωνικά δίκτυα χωρίζονται σε δίκτυα ενός επιπέδου και πολυεπίπεδα. Τα πρώτα περιλαμβάνουν μόνο το επίπεδο εισόδου, όπου δεν πραγματοποιείται κάποιος υπολογισμός και το επίπεδο εξόδου. Τα δεύτερα περιλαμβάνουν και κρυφά επίπεδα. Πέραν αυτού τα νευρωνικά δίκτυα χωρίζονται σε δίκτυα πρόσθιας τροφοδότησης (feedforward) και με ανατροφοδότηση (recurrent). Στα πρώτα οι νευρώνες συνδέονται μόνο με νευρώνες του αμέσως επόμενου επιπέδου, ενώ μπορούμε να έχουμε και άλλου τύπου συνδέσεις. Στα δίκτυα πρόσθιας τροφοδότησης η έξοδος κάθε επιπέδου τροφοδοτείται ως είσοδος στο αμέσως επόμενο επίπεδο. Τέλος, αποκαλούμε ένα νευρωνικό δίκτυο πλήρως συνδεδεμένο (fully connected) αν περιέχει όλες τις δυνατές συνδέσεις μεταξύ των νευρώνων (βάσει της κατηγορίας που ανήκει) και μερικώς συνδεδεμένο (partially connected) αν λείπουν κάποιες συνδέσεις. Στην παρούσα εργασία θα ασχοληθούμε με fully connected δίκτυα εμπρόσθιας τροφοδότησης.



Σχήμα 5.5: Fully connected Feed Forward νευρωνικό δίκτυο [15]

Ένα τεχνητό νευρωνικό δίκτυο εκτελεί δύο λειτουργίες, την μάθηση (learning) ή εκπαίδευση (training) και την ανάκληση (recall). Λέγοντας μάθηση, εννοούμε την προσαρμογή των βαρών του δικτύου (επί των συνδέσεων των νευρώνων) ώστε το δίκτυο να αποτυπώνει σωστά τις εισόδους που λαμβάνει στις αντίστοιχες επιθυμητές εξόδους. Οι είσοδοι αυτές και οι επιθυμητές έξοδοι προέρχονται από τα δείγματα εκπαίδευσης. Είναι σαφές πως χρειαζόμαστε μία μετρική για να ελέγχουμε κατά πόσο το δίκτυο μας κάνει αυτήν την αποτύπωση σωστά. Γι'αυτό χρησιμοποιούμε την συνάρτηση σφάλματος (loss function), που υπολογίζει πόσο απέχει η έξοδος του δικτύου από την επιθυμητή. Η υπολογισμένη απόσταση της εξόδου απ'το επιθυμητό αποτέλεσμα χρησιμοποιείται ώστε να αλλάξουμε τα βάρη του δικτύου στην κατεύθυνση που θα μειώσει την τιμή της συνάρτησης κόστους, όσον αφορά το δείγμα που εξετάζουμε. Η προσαρμογή των βαρών γίνεται από τον βελτιστοποιητή (optimizer), ο οποίος εφαρμόζει τον αλγόριθμο της οπισθοδιάδοσης του σφάλματος [12], τον οποίο θα εξετάσουμε στην συνέχεια. Σημειώνεται πως τα νευρωνικά δίκτυα μπορούν να μαθαίνουν και στα πλαίσια της μη επιβλεπόμενης μάθησης, κάτι που δεν θα μας απασχολήσει στην παρούσα εργασία. Στην ανάκληση για συγκεκριμένες εισόδους και βάρη υπολογίζουμε την έξοδο.

Τα νευρωνικά δίκτυα μπορούν να χρησιμοποιηθούν τόσο για προβλήματα ταξινόμησης όσο και για προβλήματα παλινδρόμησης. Στην παρούσα εργασία θα ασχοληθούμε με τα δεύτερα. Η εκπαίδευση των μοντέλων γίνεται σε κύκλους που ονομάζονται εποχές. Τα μοντέλα νευρωνικών δικτών δεν χειρίζονται όλα τα δεδομένα μονομιάς, αλλά τα σπάνε σε κομμάτια, τις δέσμες (batch). Πιο συγκεκριμένα, σε κάθε βήμα το δίκτυο δέχεται ως είσοδο, ένα-ένα, όλα τα διανύσματα εισόδου των δείγματα εκπαίδευσης που ανήκουν σε μία δέσμη δειγμάτων, υπολογίζει τις εξόδους, και βάσει της απόστασης από τις επιθυμητές εξόδους ανανεώνει τα

βάρη του δικτύου, μία φορά αθροιστικά για ολόκληρη την δέσμη. Αυτό επαναλαμβάνεται για όλες τις δέσμες και έτσι ολοκληρώνεται μία εποχή. Η εκπαίδευση του νευρωνικού δικτύου περιλαμβάνει πολλές εποχές. Μια άλλη προσέγγιση είναι η επαυξητική μάθηση κατά την οποία η αναπροσαρμογή των βαρών γίνεται μτά την χρήση ενός από τα διανύσματα εκπαίδευσης. Η προσέγγιση αυτή μπορεί να είναι χρήσιμη όταν τα δεδομένα εκπαίδευσης γίνονται σταδιακά διαθέσιμα.

Τα πολυεπίπεδα Feed Forward τεχνητά νευρωνικά δίκτυα εκπαιδεύονται με τον αλγόριθμο ανάστροφης διάδοσης του σφάλματος (backpropagation), ο οποίος αποτελεί μια διαδικασία βελτιστοποίησης επικλινούς καθόδου (gradient descent optimization procedure)[54]. Ειδικότερα, ελαχιστοποιούμε την συνάρτηση σφάλματος που ορίζεται ως το μέσο τετραγωνικό σφάλμα μεταξύ της εξόδου του δικτύου και της επιθυμητής εξόδου για p δείγματα εκπαίδευσης, με m εξόδους. Ως f θα ονομάζουμε γενικά την συνάρτηση ενεργοποίησης και input το άθροισμα των σταθμισμένων εισόδων από το προηγούμενο επίπεδο. Έτσι, έχουμε:

$$E = \frac{1}{p} \sum_{p} \sum_{k=1}^{m} (y_{kp} - f(input_{kp}))$$

Πριν αναλύσουμε τον αλγόριθμο Backpropagation, θα αναφερθούμε στην λογική της βελτιστοποίησης επικλινούς καθόδου. Όπως αναφέραμε παραπάνω σε κάθε βήμα αλλάζουμε τα βάρη για να μειώσουμε την συνάρτηση κόστους. Αρχικά, θα εξετάσουμε ένα βάρος. Το θέμα που προκύπτει είναι κατά ποιόν τρόπο θα αλλάξουμε το βάρος αυτό για να μειωθεί η συνάρτηση κόστους, αν δηλαδή πρέπει να μειώσουμε ή να αυξήσουμε το βάρος. Μια απλοϊκή ιδέα θα ήταν να δοκιμάσουμε, για σταθερές τις τιμές των υπολοίπων βαρών, διαφορετικές τιμές του βάρους και να δούμε πως ανταποκρίνεται η συνάρτηση κόστους. Για να το κάνουμε αυτό, δίνουμε τις ίδιες εισόδους στο νευρωνικό, αλλά με διαφορετικό το συγκεκριμένο βάρος, σε κάθε επίπεδο οι νεωρώνες αθρόιζουν τις σταθμισμένες εισόδους και μέσω της συνάρτησης ενεργοποίησης δίνουν έξοδο στο επόμενο επίπεδο, μέχρι και την παραγωγή εξόδου απ'τους νευρώνες στο επίπεδο εξόδου. Η διαδικασία αυτή λέγεται προς τα εμπρός πέρασμα (forward pass). Χρησιμοποιώντας τις εξόδους βλέπουμε την μεταβολή στην συνάρτηση κόστους. Συνεπώς, για ένα βάρος θα μπορούσαμε να τρέξουμε δύο προς τα εμπρός περάσματα, ένα για μικρότερη τιμή της τρέχουσας και ένα για μεγαλύτερη. Ωστόσο, το νευρωνικό δίκτυο έχει χιλιάδες έως εκατομμύρια τέτοια βάρη. Συνεπώς, η προσέγγιση αυτή δεν είναι αποδεκτή λόγω υπολογιστικού κόστους.

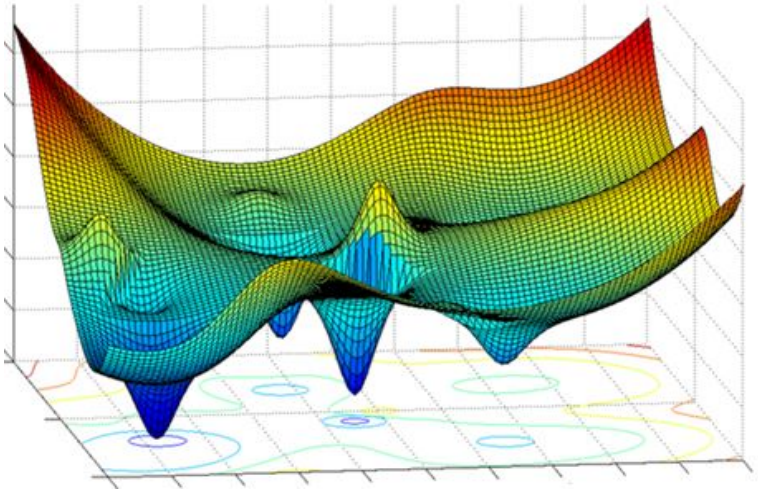
Αντί αυτής της προσέγγισης θα χρησιμοποιήσουμε την παράγωγο της συνάρτησης σφάλματος ως προς τα βάρη του νευρωνικού δικτύου. Ως γνωστόν, η παράγωγος μίας συνάρτησης ως προς μία μεταβλητή δείχνει την συμπεριφορά της ως προς την μεταβλητή αυτή, δηλαδή αν είναι αύξουσα ή φθίνουσα. Άρα, αυτό που πρέπει να κάνουμε για να μειώσουμε την τιμή της συνάρτησης κόστους είναι να αυξήσουμε το βάρος αν η συνάρτηση κόστους είναι φθίνουσα ή να το μειώσουμε αν είναι αύξουσα. Θα γενικεύσουμε αυτήν την λογική για όλα τα βάρη του δικτύου μας. Υπολογίζοντας την μερική παράγωγο της συνάρτησης για κάθε βάρος, θα έχουμε μία εικόνα για την κατεύθυνση και το μέγεθος της αλλαγής της συνάρτησης κόστους αν αλλάξουμε κάθε μία από τα βάρη μας. Έτσι, θα αλλάξουμε όλα τα βάρη βάσει αυτής της παραγώγου ακολουθώντας την προηγούμενη λογική.

Η αναζήτηση του συνδυασμού των βαρών που ελαχιστοποιούν την συνάρτηση κόστους θα μπορούσε να γίνει μηδενίζοντας την παράγωγό της. Ωστόσο, η αλγεβρική επίλυση του προβλήματος αυτού δεν επιλέγεται λόγω πολυπλοκότητας καθότι έχουμε πάρα πολύ μεγάλο αριθμό μεταβλητών (βάρη). Αντί αυτού σε κάθε βήμα θα αλλάζουμε τα βάρη και θα κατευθυνόμαστε προς τον βέλτιστο συνδυασμό τους, ως εξής:

$$W_1 = W_0 - \lambda grad(f)(W_0)$$

Όπου W_0 τα βάρη πριν την τροποποίησή τους και W_1 μετά από αυτήν. Όπως βλέπουμε τα βάρη αλλάζουν βάσει της παραγώγου της συνάρτησης κόστους, αλλά και με πολλαπλασιασμό

με μία σταθερά που λέγεται ρυθμός εκπαίδευσης (learning rate). Η σταθερά αυτή καθορίζει την ταχύτητα σύγκλισης προς τον βέλτιστο συνδυασμό. Η συνάρτηση κόστους είναι μία πολύπλοκη συνάρτηση. Χαρακτηρίζεται από τοπικά μέγιστα και ελάχιστα. Στόχος μας είναι να βρεθούμε σε ολικό ελάχιστο. Με πολύ μικρή τιμή του ρυθμού εκπαίδευσης η κατάβαση επί της καμπύλης της συνάρτησης κόστους θα χρειαστεί πολλά μικρά βήματα με κίνδυνο εγκλωβισμού σε τοπικό ελάχιστο. Ισχύει πως ο απαιτούμενος χρόνος εκπαίδευσης σε περιπτώσεις μικρού ρυθμού εκπαίδευσης είναι αυξημένος σε σχέση με περιπτώσεις μεγαλύτερης τιμής αυτού. Αν όμως ο ρυθμός εκπαίδευσης είναι πολύ μεγάλος, οι ανανεώσεις των βαρών μπορεί να οδηγούν σε τυχαίες θέσεις στην καμπύλη. Με άλλα λόγια είναι πιθανόν να προσπεραστεί το σημείο ελαχίστου κόστους και να βρεθούμε σε κατάσταση παλινδρόμησης γύρω από το σημείο που αντιστοιχεί στις βέλτιστες τιμές των βαρών. Μια προσέγγιση για την αποφυγή του εγκλωβισμού στα τοπικά ελάχιστα είναι η χρήση της ορμής (momentum) από τους αλγορίθμους βελτιστοποίησης, κατά την οποία για την ανανέωση των βαρών βασιζόμαστε πέραν της τιμής της παραγώγου της συνάρτησης κόστους και στην προηγούμενη ανανέωση.[12]



Σχήμα 5.6: Μια συνάρτηση σφάλματος για δύο βάρη [19]

Με τον αλγόριθμο Backpropagation καθορίζουμε την συμμετοχή των βαρών κάθε νευρώνα κάθε επιπέδου στο συνολιό σφάλμα του δικτύου το οποίο υπολογίζεται μετά από ένα προς τα εμπρός πέρασμα. Ύστερα ξεκινώντας από το επίπεδο εξόδου και πηγαίνοντας προς το επίπεδο εισόδου διορθώνουμε τα βάρη των συνδέσεων των νευρώνων, κάτι που ονομάζεται προς τα πίσω πέρασμα. Έστω, λοιπόν, y_k η έξοδος ένος νευρώνα του επιπέδου εξόδου μεγέθους m, z_j η έξοδος ενός τυχαίου νευρώνα του προηγουμένου κρυφού επιπέδου μεγέθους q,που συνδέεται με τον νευρώνα του επιπέδου εξόδου, w_{jk} το βάρος της μεταξύ τους σύνδεσης. Θεωρούμε πως θα'ναι:

$$y_k = f(input_k) = f(\sum_{j=1}^{q} w_{jk} z_j)$$

Αντίστοιχα, αν θεωρήσουμε ως u_i τις εξόδους του κρυφού επιπέδου πριν από το τελευταίο κρυφό επίπεδο μεγέθους n, καθώς και w_{ij} τα βάρη των συνδέσεων μεταξύ των αντίστοιχων νευρώνων των κρυφών επιπέδων, έχουμε:

$$z_j = f(input_j) = f(\sum_{i=1}^{n} w_{ij} u_i)$$

Για τον αλγόριθμο υποθέτουμε πως οι συναρτήσεις ενεργοποίησης είναι μονότονα αύξουσες και παραγωγίσιμες. Αρχικά, τα βάρη έχουν τυχαίες τιμές. Ξεκινώντας το πίσω πέρασμα,

τροποποιούμε πρώτα τις τιμές των βαρών στις συνδέσεις με τους νευρώνες εξόδου κατά την κάτωθι ποσότητα:

$$\Delta w_{jk} = \lambda \, \delta_k \, z_j$$

Όπου δ_k ο ρυθμός μεταβολής του σφάλματος ως προς την είσοδο στο νευρώνα k. Για τους νευρώνες επιπέδου εξόδου έχουμε (με Y_k συμβολίζουμε την επιθυμητή έξοδο):

$$\delta_k = (Y_k - y_k) f^{'}(\sum_{j=1}^{q} w_{jk} z_j)$$

Στην συνέχεια, για τα βάρη που αφορούν τα υπόλοιπα επίπεδα, έφαρμόζουμε την παρακάτω τροποποίηση:

$$\Delta w_{ij} = \lambda \, \delta_j \, u_i$$

Όπου δ_k ο ρυθμός μεταβολής του σφάλματος ως προς την είσοδο στο νευρώνα k. Για τους νευρώνες επιπέδου εξόδου έχουμε (με Y_k συμβολίζουμε την επιθυμητή έξοδο):

$$\delta_j = f^{'}(\sum_{i=1}^{n} w_{ij} u_j) \sum_{k=1}^{m} \delta_k w_{jk}$$

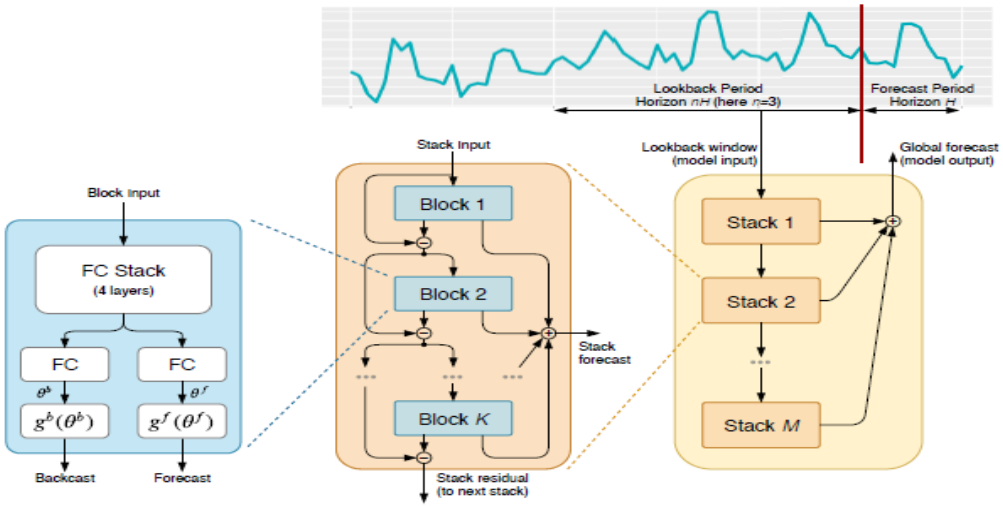
Ο αλγόριθμος ανανεώνει τα βάρη σε κάθε βήμα σύμφωνα με τα δείγματα των δεσμών που χρησιμοποιεί, βάσει των παραπάνω σχέσεων. Ο αλγόριθμος τερματίζει μόλις ικανοποιηθεί το κριτήριο τερματισμού του. Ειδικότερα, συνήθως τερματίζει μόλις η συνάρτηση σφάλματος πέσει κάτω από μία συγκεκριμένη τιμή είτε αν ολοκληρωθούν όλες οι εποχές εκπαίδευσης.Όπως στους αλγορίθμους εκπαίδευσης, συνιθίζεται να κρατάμε δεδομένα για την αξιολόγηση του μοντέλου. Συνηθίζεται, αν δεν υπάρξει αξιοσημείωτη μείωση στην τιμή της συνάρτησης σφάλματος στα δεδομένα αξιολόγησης, μετά από κάποιον αριθμό εποχών, να διακόπτουμε την εκπαίδευση του νευρωνικού δικτύου. Η τεχνική αυτή λέγεται Early Stopping.

Στην προηγούμενη ενότητα αναφερθήκαμε στις στρατηγικές με τις οποίες αντιμετωπίζουμε το πρόβλημα της παραγωγής προβλέψεων με μεθόδους μηχανικής μάθησης. Όσον αφορά τα νευρωνικά δίκτυα είχε αναφερθεί ότι θα χρησιμοποιήσουμε τις στρατηγικές πολλαπλών εισόδων-πολλαπλών εξόδων, την αναδρομική στρατηγική και την ευθεία στρατηγική. Κάθε μία από αυτές τις στρατηγικές μεταφράζεται σε διαφορετική τοπολογία στο νευρωνικό δίκτυο. Ειδικότερα, σε κάθε περίπτωση το μέγεθος του επιπέδου εισόδου ταυτίζεται με το πλήθος των εισόδων που χρησιμοποιεί η κάθε στρατηγική. Στις τρεις προαναφερθήσες στρατηγικές το μέγεθος της εισόδου είναι σταθερό και ίσο με το διάνυσμα εισόδου των δειγμάτων εκπαίδευσης (ίσο δηλαδή με το πλήθος των χαρακτηριστικών εισόδου). Το πλήθος των κρυφών επιπέδων καθώς και το μέγεθος κάθε ενός από αυτά είναι κάτι ανεξάρτητο της στρατηγικής. Θα πρέπει όμως να έχουμε πάντοτε κατά νου ότι ένα νευρωνικό δίκτυο που δεν είναι αρκετά πολύπλοκο είναι πιθανόν να μην πετύχει να μοντελοποιήσει επιτυχώς τα δεδομένα εκπαίδευσης, δηλαδή έχουμε ατελή μάθηση ή υποπροσαρμογή, ενώ ένα ιδιαίτερα πολύπλοκο μοντέλο ενδέχεται να μοντελοποιήσει υπερβολικά τα δεδομένα εκπαίδευσης[54], δηλαδή να έχουμε υπερπροσαρμογή. Όσον αφορά το επίπεδο εξόδου είτε έχουμε έναν νευρώνα, αν η στρατηγική είναι μίας εξόδου, είτε έχουμε πολλούς νευρώνες και συγκεκριμένα τόσους όσοι είναι το πλήθος των εξόδων στα δεδομένα μας, αν η στρατηγική είναι πολλαπλών εξόδων. Στην περίπτωση της ευθείας στρατηγικής δημιουργούμε πολλά νευρωνικά δίκτυα, το καθένα από τα οποία προβλέπει για έναν ορίζοντα έχοντας έναν νευρώνα στο επίπεδο εξόδου.

5.2 Ο αλγόριθμος N-Beats

Ο αλγόριθμος N-Beats αποτελεί μια σύγχρονη μέθοδο βαθιάς μάθησης που προτάθηκε από τους Oreshkin, Chapados, Carpov και Bengio[34] για την πρόβλεψη χρονοσειρών. Η

αρχιτεκτονική του μοντέλου αποτελείται από μία σειρά από στοίβες (Stack), καθεμία από τις οποίες έχει μία σειρά από Blocks. Κάθε στοίβα παράγει δύο αποτελέσματα, ένα που δίνεται ως είσοδος στην επόμενη και λέγεται υπόλοιπο στοίβας (Stack Residual) καθώς και ένα που συμμετέχει σε ένα ολικό άθροισμα, το οποίο άθροισμα αποτελεί την έξοδο του μοντέλου. Η έξοδος αυτή λέγεται πρόβλεψη στοίβας (Stack Forecast). Η είσοδος του μοντέλου (και της πρώτης στοίβας) είναι ένα παράθυρο παρελθοντικών παρατηρήσεων πριν την περίοδο για την οποία ζητούμε να κάνουμε πρόβλεψη ορίζοντα H. Το πλήθος των παρατηρήσεων εισόδου είναι ακέραιο πολλαπλάσιο του ορίζοντα πρόβλεψης, δηλαδή nH (π.χ. 3H).
Μία στοίβα αποτελείται από Blocks τα οποία συνδέονται μεταξύ τους. Ειδικότερα, παρομοίως με τις στοίβες, τα Blocks έχουν δύο εξόδους και μία είσοδο. Η μία έξοδος συμμετέχει σε ένα ολικό άθροισμα για να προκύψει η πρόβλεψη της στοίβας. Η έξοδος αυτή πρόκειται για μια πρόβλεψη ορίζοντα H βάσει της εισόδου (Forecast). Την άλλη έξοδο, που αποτελεί πρόβλεψη της εισόδου του Block (άρα έχει nH τιμές), την αφαιρούμε από την είσοδο του Block και το αποτέλεσμα δίνεται ως είσοδος στο επόμενο Block της στοίβας, με εξαίρεση το τελευταίο Block όπου δίνεται στο πρώτο Block της επόμενης στοίβας (και πρόκειται για το υπόλοιπο της παρούσας στοίβας). Η αφαίρεση γίνεται ανά στοιχείο της ίδιας θέσης.



Σχήμα 5.7: Η αρχιτεκτονική του N-Beats [34]

Το Block περιλαμβάνει ένα πλήρως συνδεδεμένο νευρωνικό δίκτυο τεσσάρων στρωμάτων με συναρτήσεις ενεργοποίησεις ReLU. Η είσοδος του Block και του δικτύου έχει μέγεθος nH. Η έξοδος του δικτύου διακλαδίδεται σε δύο δίκτυα του ενός στρώματος τα οποία έχουν γραμμικές συνάρτησεις ενεργοποίησης. Ως έξοδο τα δίκτυα αυτά παράγουν τους προς τα εμπρός και προς τα πίσω συντελεστές επέκτασης θ_l^f και θ_l^b (backward and forward expansion coefficients). Το l στον δείκτη αναφέρεται στον αριθμό του Block. Εν συνεχεία, από τους συντελεστές αυτούς παράγουμε την πρόβλεψη του Block για την είσοδο (Backcast) και την πρόβλεψη του Block για ορίζοντα H (Forecast). Αξίζει να σημειώσουμε πως η είσοδος σε κάθε Block είναι το σφάλμα πρόβλεψης της εισόδου του προηγούμενου Block. Με άλλα λόγια, κάθε Block λαμβάνει ως είσοδο το κομμάτι που δεν κατάφερε να μοντελοποιήσει το προηγούμενο Block.
Υπάρχουν δύο διαμορφώσεις της παραπάνω αρχιτεκτονικής, η γενική αρχιτεκτονική (generic architecture) και η ερμηνεύσιμη αρχιτεκτονική (interpretable architecture). Η διαφορά τους έχει να κάνει με την παραγωγή των προβλέψεων στα Blocks. Η γενική αρχιτεκτονική δεν εξαρτάται από ειδική γνώση σχετικά με την χρονοσειρά. Έστω, $\hat{y}_l$ η πρόβλεψη του Block l και $\hat{x}_l$ η εκτίμηση του ίδιου Block για την είσοδό του. Τότε στην γενική αρχιτεκτονική θα έχουμε:

$$\hat{y}_l = V_l^f \theta_l^f + b_l^f$$

$$\hat{x}_l = V_l^b \theta_l^b + b_l^b$$

Οι συντελεστές στις παραπάνω σχέσεις έχουν μορφή πίνακα. Οι συντελεστές που χρησιμοποιούμε μαθαίνονται από το δίκτυο, χωρίς να έχουμε επιβάλλει κανέναν περιορισμό. Ως εκτούτου δεν έχουν εξ'ορισμού κάποια συγκεκριμένη δομή συνεπώς δεν μπορούμε να ερμηνεύσουμε το αποτέλεσμα. Στην ερμηνεύσιμη αρχιτεκτονική ακολουθούμε την λογική της αποσύνθεσης της χρονοσειράς στις συνιστώσες τάσης και εποχιακότητας. Έτσι, αρχικά οι πρώτες στόιβες αφαιρούν την τάση από την είσοδο πριν την παραδόσουν στις επόμενες στοίβες που μοντελοποιούν την εποχιακότητα. Οι μερικές προβλέψεις των στοιβών σ'αυτήν την περίπτωση είναι πλήρως ερμηνεύσιμες. Η τάση μοντελοποιείται μέσω ενός πολυωνύμου βαθμού p, ενώ η εποχιακότητα χρησιμοποιώντας σειρές Fourier. Αναλυτικότερα, για την παραγωγή των προβλέψεων (αντίστοιχα και για την πρόβλεψη της εισόδου) έχουμε τους παρακάτω τύπους (όπου l ο αριθμός του Block και s ο αριθμός της στοίβας)[34]:

$$\hat{y}_{s,l} = \sum_{i=0}^{p} \theta_{s,l,i}^f t^i \ \text{(τάση)}$$

$$\hat{y}_{s,l} = \sum_{i=0}^{H/2-1} \theta_{s,l,i}^f cos(2\pi it) + \theta_{s,l,i}^f sin(2\pi it) \ \text{(εποχιακότητα)}$$

$$t \in [0, 1, 2, ..., H-2, H-1]^T / H$$

Αξίζει να σημειώσουμε πως το μοντέλο N-Beats είναι ένα σχετικά απλό και γενικό μοντέλο. Είναι εύκολο στην χρήση του καθώς δεν απαιτούνται αλλαγές στα δεδομένα για την εφαρμογή της μεθόδου, ενώ η εκπαίδευσή του γίνεται γρήγορα. Τέλος, η εφαρμογή του στην πράξη έχει δείξει ότι με αυτό μπορούμε να κάνουμε προβλέψεις πολύ καλής ποιότητας.

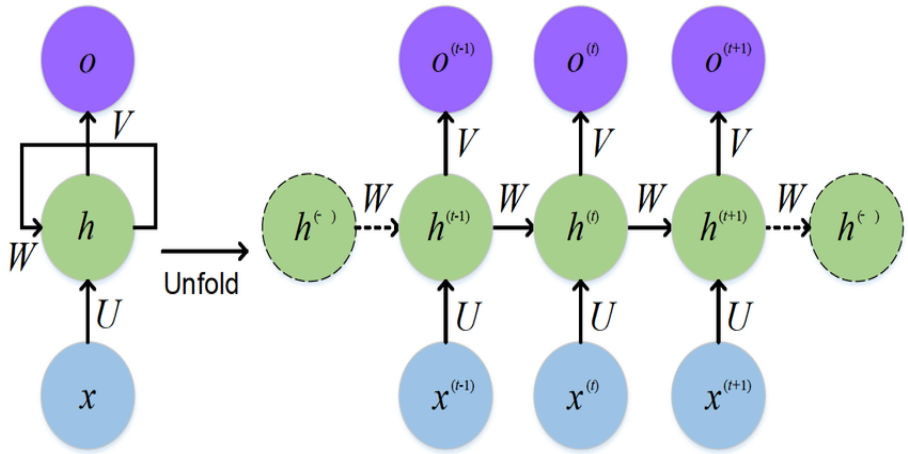
5.3 Άλλες Αρχιτεκτονικές Νευρωνικών Δικτύων

Νευρωνικά Δίκτυα με Ανατροφοδότηση

Τα νευρωνικά δίκτυα με ανατροφοδότηση (Recurrent Neural Networks-RNN) αποτελούν μία ομάδα νευρωνικών δικτύων κατάλληλα για επεξεργασία ακολουθιακών δεδομένων. Τα δίκτυα αυτά μπορούν να χειριστούν ακολουθίες πολύ μεγάλου μεγέθους, ενώ δεν αντιμετωπίζουν πρόβλημα ακόμη και αν οι ακολουθίες εισόδου έχουν μεταβλητό μέγεθος.[20] Σε αντίθεση με τα δίκτυα που έχουμε δει ως τώρα, τα ανατροφοδοτούμενα νευρωνικά δίκτυα κρατούν πληροφορία σχετικά με τα δεδομένα που επεξεργάστηκαν στο παρελθόν, διαθέτουν δηλαδή μνήμη. Με άλλα λόγια διατηρούν μια μεταβλητή κατάστασης που περιέχει πληροφορία για όσα έχουν δει μέχρι την εκάστοτε χρονική στιγμή. Η μεταβλητή κατάστασης ενημερώνεται χρησιμοποιώντας την προηγούμενη τιμή της και την παρούσα είσοδο. Η αρχική τιμή της ορίζεται από εμάς. Σε κάθε βήμα, η έξοδος υπολογίζεται από την τρέχουσα κατάσταση, συνεπώς εξαρτάται από την παρούσα είσοδο και την κατάσταση του προηγούμενου βήματος. Όμως η κατάσταση του προηγούμενου βήματος εξαρτάται από την είσοδο του προηγούμενου βήματος και την αμέσως προηγούμνη κατάστασή της. Έτσι, ακολουθώντας την λογική αυτή στο βάθος των βημάτων, η κάθε έξοδος εξαρτάται από την παρούσα είσοδο και τις παλαιότερες είσοδους.

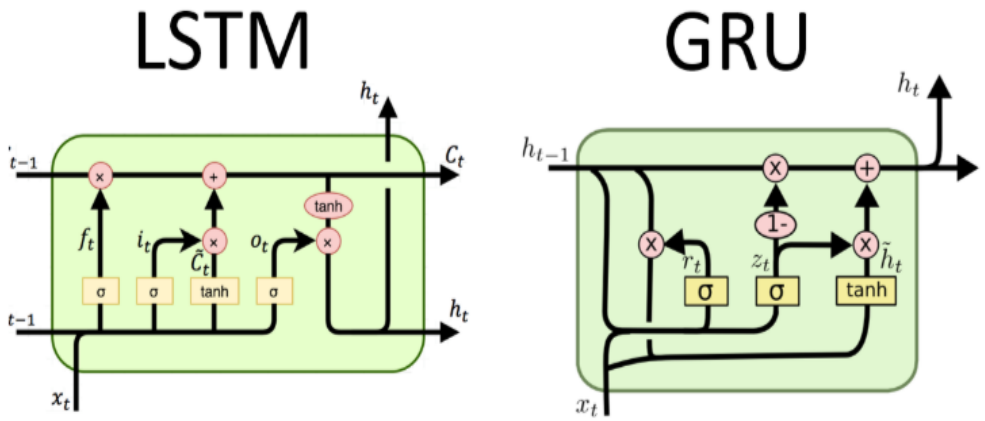
Τα ανατροφοδοτούμενα νευρωνικά δίκτυα έχουν βάρη για τον υπολογισμό της κατάστασης από την είσοδο, της κατάστασης από την προηγούμενη κατάσταση και από την τρέχουσα κατάσταση στην είσοδο. Τα βάρη συνοδεύονται από κατώφλια. Η προσαρμογή των βαρών γίνεται κατά την εκπαίδευση του δικτύου. Τα δίκτυα αυτής της κατηγορίας ακολουθούν την ιδιότητα του διαμοιρασμού παραμέτρων μεταξύ διαφορετικών μερών του μοντέλου.

Λόγω αυτής της ιδιότητας, μπορούμε να εφαρμόσουμε μοντέλα αυτής της αρχιτεκτονικής σε εισόδους διαφορετικής μορφής (μεγέθους). Αφότου έχουμε εκπαιδεύσει το δίκτυο μας, για κάθε βήμα που προκύπτει βάσει του μήκους της κάθε εισόδου, τα βάρη που χρησιμοποιούμε για τον υπολογισμό της εξόδου μένουν σταθερά.



Σχήμα 5.8: Οι υπολογισμοί του RNN ξεδιπλωμένοι στο χρόνο [17]

Το μοντέλο που αναλύσαμε παραπάνω είναι ιδιαιτέρως απλό, ενώ παρουσιάζει το πρόβλημα της εξαφανιζόμενης κλίσης κατά την εκπαίδευση, που γίνεται με οπισθοδιάδοση του σφάλματος. Αντ'αυτού χρησιμοποιούμε δύο άλλες αρχιτεκτονικές, τα Long Short-Term Memory (LSTM) και τα Gated Recurrent Units (GRU), ως δομικές μονάδες στα νευρωνικά δίκτυά μας. Τα LSTM αποθηκεύει πληροφορίες, αποτρέποντας έτσι την εξαφάνιση παλαιότερων σημάτων. Πιο συγκεκριμένα, χρησιμοποιώντας μία μεταβλητή υπολοίπου (curry) μπορεί να επαναχρησιμοποιεί παλαιότερη πληροφορία. Στην ανανέωση του υπολοίπου συμμετέχουν η παρούσα κατάσταση, είσοδος και το παρόν υπόλοιπο. Τα GRU μπορούν να θεωρηθούν ως μία απλούστευση των LSTM. Στην παρούσα εργασία δεν θα επεκταθούμε περισσότερο επί αυτών.



Σχήμα 5.9: Τα block σε ένα RNN [6]

Συνελικτικά Νευρωνικά Δίκτυα

Τα συνήθη νευρωνικά δίκτυα σε κάθε επίπεδο εκτελούν έναν πολλαπλασιασμό πινάκων, μεταξύ της εισόδου και των βαρών, με αποτέλεσμα την παραγόμενη έξοδο. Ως συνελικτικά νευρωνικά δίκτυα (Convolutional Neural Networks-CNN) ορίζουμε τα νευρωνικά δίκτυα

που αντί για πολλαπλασιασμό πινάκων χρησιμοποιούν συνέλιξη σε τουλάχιστον ένα από τα επίπεδα που τα απαρτίζει.[20] Μια συνήθης χρήση των δικτύων αυτών είναι για την επεξεργασία δεδομένων που έχουν τοπολογία πλέγματος, για παράδειγμα χρονοσειρές (πλέγμα μίας διάστασης) και δεδομένα εικόνων (πλέγμα pixels δύο διαστάσεων). Τα συνελικτικά δίκτυα έχουν την δυνατότητα να ανταποκρίνονται σε προβλήματα, όπου τα συνήθη νευρωνικά δεν μπορούν λόγω της μεγάλης υπολογιστικής τους πολυπλοκότητας, καθότι ο αριθμός των απαιτούμενων παραμέτρων στην περίπτωσή τους είναι μειωμένος.
Γνωρίζουμε ότι η συνέλιξη είναι μία πράξη που ορίζεται για δύο συνεχείς συναρτήσεις, ενώ μπορούμε να την ορίσουμε και για συναρτήσεις διακριτού χρόνου. Ο τελευταίος ορισμός είναι πιό χρήσιμος όταν ασχολούμαστε με δεδομένα σε υπολογιστές, όπως εδώ. Οι ορισμοί αυτοί της συνέλιξης είναι γνωστοί και ως εκ τούτου δεν θα τους επαναλάβουμε. Στα συνελικτικά δίκτυα ορίζουμε ως είσοδο (input) τον πρώτο τελεστή της συνέλιξης, ως πυρήνα (kernel) τον δεύτερο και ως χάρτη χαρακτηριστικών (feature map) την έξοδο. Σαν κανόνας ισχύει πως ο πυρήνας είναι σημαντικά μικρότερων διαστάσεων από την είσοδο. Τα δεδομένα μας πολύ συχνά είναι περισσότερων της μίας διάστασης. Η συνέλιξη ορίζεται αναλόγως. Για παράδειγμα, για δεδομένα δύο διαστάσεων έχουμε:

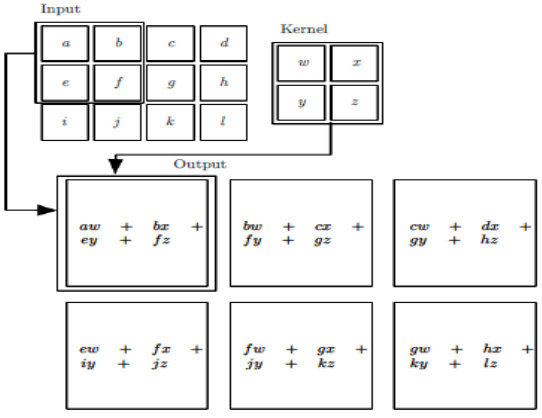
$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n)$$

Η συνέλιξη είναι πράξη αντιμεταθετική:

$$S(i,j) = (K * J)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n)$$

Συχνά, ως συνέλιξη χρησιμοποιούμε τον ορισμό της διασυσχέτισης (cross correlation), οπου έχουμε:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n)$$



Σχήμα 5.10: Συνέλιξη σε δύο διαστάσεις [20]

Τα συνελικτικά δίκτυα αποτελούνται από τρεις τύπους στρωμάτων, τα συνελικτικά, τα στρώματα ομαδοποίησης (pooling layers) και τα πλήρως συνδεδεμένα στρώματα[32]. Στα συνελικτικά στρώματα έχουμε πυρήνες, συνήθως μικρών διαστάσεων και πάντοτε βάθους ίσου με το βάθος της εισόδου, τους οποίους ολισθαίνουμε επί της εισόδου για να παράξουμε την έξοδο μέσω εσωτερικού γινομένου. Η ολίσθηση μπορεί να γίνεται με βήμα ένα ή μεγαλύτερο κάτι που καθορίζει το μέγεθος της εξόδου. Πολλές φορές για να ελέγξουμε το μέγεθος της εξόδου της συνέλιξης, καθότι αυτό είναι καταρχήν μικρότερο της εισόδου, χρησιμοποιούμε την τεχνική zero-padding. Σ'αυτήν την περίπτωση γεμίζουμε με μηδενικές

τιμές το όριο της εισόδου κατακόρυφα και οριζόντια αυξάνοντας έτσι τις διαστάσεις της. Οι τιμές των πυρήνων μαθαίνονται από το δίκτυο κατά την εκπαίδευσή του, ώστε να δίνουν τέτοια τιμή που να δηλώνει την ύπαρξη ή μη χαρακτηριστικού στην είσοδο. Με το στρώμα ομαδοποίησης στοχεύουμε στην μείωση της διάστασης της αναπαράστασης, άρα και της πολυπλοκότητας του μοντέλου. Ειδικότερα, χρησιμοποιώντας μια συνάρτηση, συνδυάζει στοιχεία της εισόδου με γειτονικά του και δίνει ως έξοδο μόνο ένα στοιχείο με την τιμή του αποτελέσματος. Η ευρύτερα χρησιμοποιούμενη συνάρτηση είναι αυτή του μεγίστου, που επιστρέφει απλώς την μεγαλύτερη τιμή. Τότε μιλάμε για max-pooling, το οποίο στα περισσότερα συνελικτικά δίκτυα εφαρμόζεται ως πυρήνας 2x2 με βήμα 2 και μειώνει κατά τα 3/4 το μέγεθος της εισόδου. Τέλος, έχουμε το πλήρως συνδεδεμένο στρώμα που περιέχει νευρώνες στην λογική των πλήρως συνδεδεμένων δικτών ευθείας τροφοδότησης, τα οποία αναλύσαμε στην αρχή του παρόντος κεφαλαίου.

Κεφάλαιο 6

Εφαρμογή για Προβλέψεις

6.1 Χρησιμοποιούμενα Εργαλεία

Στα πλαίσια της παρούσας εργασίας υλοποιήσαμε μία εφαρμογή για την παραγωγή προβλέψεων από τις τρείς χρονοσειρές μας μέσω των μεθόδων που αναλύσαμε στα προηγούμενα κεφάλαια. Η εφαρμογή δομείται σε δύο μέρη, στο κομμάτι που αφορά την διεπαφή χρήστη και γενικότερα την παρουσίαση της εφαρμογής (Front-End) και το κομμάτι που πραγματοποιεί τις προβλέψεις που ζητάει ο χρήστης, καθώς και τις δοσοληψίες με την βάση δεδομένων (Back-End). Πιο συγκεκριμένα, έχουμε δημιουργήσει μία εφαρμογή τύπου μίας σελίδας (Single Page Application), όπου φορτώνουμε στον φυλλομετρητή μόνο ένα αρχείο, το οποίο ανανεώνεται βάσει των επιλογών του χρήστη της εφαρμογής. Για το Back-End μέρος φτιάξαμε μία υπηρεσία που λειτουργεί ως εξυπηρετητής για τον πελάτη, που αντιπροσωπεύει το Front-End. Η χρήση της υπηρεσίας γίνεται μέσω διεπαφής προγραμματισμού εφαρμογής (Application Programming Interface-API). Ως API ορίζουμε την διεπαφή προγραμματιστικών διαδικασιών που παρέχεται από ένα λειτουργικό σύστημα, μία βιβλιοθήκη ή μία εφαρμογή, προκειμένου να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα ή/και ανταλλαγή δεδομένων [8]. Στην περίπτωσή μας το API είναι τύπου REST (RESTful API). Το REST (Representational State Transfer) είναι ένα αρχιτεκτονικό στυλ, που έχει τα ακόλουθα χαρακτηριστικά[18]:

- Διαχωρισμός πελάτη-εξυπηρετητή. Στην λογική της αρχής σχεδίασης του διαχωρισμού των ανησυχιών (separation of concerns) ακολουθούμε το μοντέλο πελάτης-εξυπηρετητή. Ο διαχωρισμός αυτός επιτρέπει στα μέρη του συστήματος να αναπτύσσονται ανεξάρτητα το ένα από το άλλο.

- Έλλειψη κατάστασης. Η επικοινωνία μεταξύ εξυπηρετητή και πελάτη πρέπει να μην χαρακτηρίζεται από καταστάσεις, υπό την έννοια ότι κάθε αίτημα από τον πελάτη προς τον εξυπηρετητή πρέπει να περιέχει όλη την απαραίτητη πληροφορία για να γίνει κατανοητό από τον δεύτερο. Με τον τρόπο αυτό έχουμε απλούστερα συστήματα.

- Αξιοποίηση κρυφής μνήμης. Τα δεδομένα εντός μίας απόκρισης μπορούν να αποθηκεύονται στην κρυφή μνήμη ή όχι. Στην πρώτη περίπτωση, ο πελάτης μπορεί να τα επαναχρησιμοποιήσει για ισοδύναμες αιτήσεις. Με τον τρόπο αυτό κάνουμε οικονομία στα αιτήματα, βελτιώνοντας έτσι την επίδοση του δικτύου.

- Ομοιόμορφη διεπαφή. Πρόκειται για βασικό χαρακτηριστικό της αρχιτεκτονικής αυτής, που προσφέρει καλύτερη εποπτεία των αλληλεπιδράσεων και οδηγεί σε απλούστερα συστήματα.

- Πολυεπίπεδη οργάνωση. Κάθε ένα στοιχείο δεν μπορεί να έχει πρόσβαση σε στοιχεία πέραν του αμέσως επόμενου επιπέδου. Με τα επίπεδα μπορούμε να ενθυλακώσουμε πεπαλαιωμένες υπηρεσίες. Τα ενδιάμεσα μέρη μπορούν να μεταβάλλουν το περιεχόμενο των μηνυμάτων.

- Κώδικας κατά παραγγελία. Είναι εφικτό για τον πελάτη να επεκτείνει την λειτουργικότητά του κατεβάζοντας και εκτελώντας κομμάτια κώδικα. Αυτό απλοποιεί τους πελάτες, καθότι αφαιρεί χαρακτηριστικά που πρέπει να έχουν προ-υλοποιηθεί. Με τον τρόπο αυτό το σύστημα γίνεται επεκτάσιμο.

Βασικό στοιχείο της αρχιτεκτονικής REST είναι ο πόρος (resource), δηλαδή οποιαδήποτε πληροφορία μπορεί να ονοματοδοτηθεί. Παραδείγματα πόρων αποτελούν ένα έγγραφο, μία εικόνα και ένα μη εικονικό αντικείμενο, όπως ο άνθρωπος. Οι πόροι μπορούν να είναι στατικοί ή να μεταβάλλονται ανά τον χρόνο. Για τον προσδιορισμό ενός πόρου στη REST χρησιμοποιούμε ένα αναγνωριστικό πόρου (resource identifier). Τα συστατικά μέρη της αρχιτεκτονικής εκτελούν πράξεις επί των πόρων, χρησιμοποιώντας μία αναπαράσταση τους (representation), η οποία αποτελεί μια ακολουθία bytes μαζί με τα μεταδεδομένα της αναπαράστασης (representation metadata), που περιγράφουν αυτά τα bytes. Ο πελάτης και ο εξυπηρετητής επικοινωνούν μέσω του πρωτοκόλλου HTTP.[36] Ειδικότερα, οι πελάτες αλληλεπιδρούν με το API μέσω διαφόρων τύπων μηνύματων HTTP, χρησιμοποιώντας κάποια από τις υποστηριζόμενες μεθόδους. Οι υποστηριζόμενες μέθοδοι που χρησιμοποιούνται πιο συχνά είναι οι GET, POST, PUT και DELETE. Μέσω GET αιτημάτων (GET request) προς το API, ο πελάτης ζητάει μία αναπαράσταση κάποιου πόρου. Με τα POST αιτήματα δημιουργούμε νέο πόρο βάσει κάποιας αναπαράστασης που παρέχουμε, ενώ με την PUT μέθοδο αλλάζουμε την κατάσταση ενός ήδη υπάρχοντα πόρου. Η μέθοδος DELETE, όπως δηλώνει και το όνομά της, χρησιμεύει για την διαγραφή ενός πόρου. Στην συνέχεια, θα αναλύσουμε τα εργαλεία που χρησιμοποιήσαμε για την δημιουργία των δύο μερών και αμέσως μετά θα παρουσιάσουμε την εφαρμογή μας.

FrontEnd Client

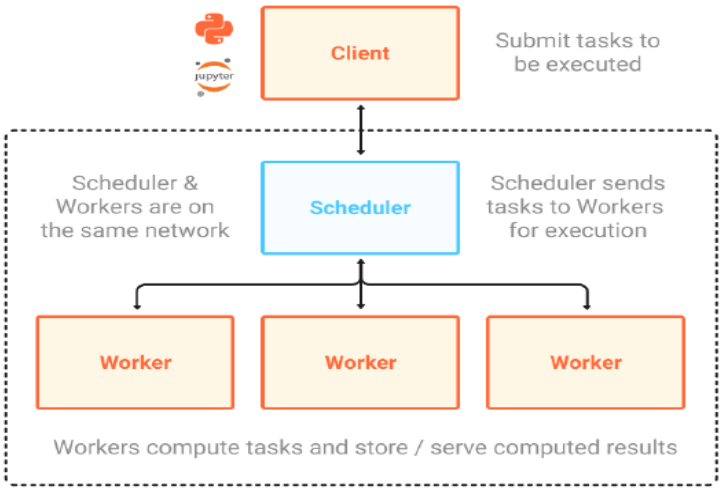
Για την δημιουργία του Front-End μέρους της εφαρμογής χρησιμοποιήσαμε ReactJS, η οποία είναι μια βιβλιοθήκη JavaScript για την δημιουργία διαδραστικών διεπαφών χρήστη. Η ReactJS επιλέχθηκε έναντι διαφόρων εναλλακτικών, όπως η AngularJS και η VueJS, διότι προσφέρει απλότητα στη χρήση σε συνδυασμό με εξαιρετική απόδοση. Η ReactJS μας δίνει την δυνατότητα να χρησιμοποιήσουμε JSX (JavaScript XML), που αποτελεί μια συντακτική επέκταση της JavaScript. Έτσι, μπορούμε να παρεμβάλουμε σε κώδικα JavaScript κομμάτια HTML (HyperText Markup Language). Στα πλαίσια της ReactJS, χρησιμοποιήσαμε JavaScript, HTML και CSS (Cascading Style Sheets) ούτως ώστε να δομήσουμε τις σελίδες μας και να τους δώσουμε κατάλληλη μορφή και λειτουργικότητα. Σημειώνεται πως είναι δυνατόν να μην γράψουμε τον κώδικά μας σε μορφή JSX, αλλά να γράψουμε μόνο JavaScript. Ωστόσο, πέραν από το θέμα της απλότητας, με την χρήση JSX η ReactJS παράγει καλύτερα μηνύματα λάθους και προειδοπήσης, κάτι ιδιαίτερα σημαντικό στην διαδικασία ανάπτυξης εφαρμογών. Η JSX μετατρέπεται σε κλήσεις της συνάρτησης React.createElement( ), μέσω της οποίας δημιουργούνται τα στοιχεία React (React Elements), από τα οποία η React διαμορφώνει την σελίδα μας.
Βασικό χαρακτηριστικό της ReactJS είναι η λογική χρησιμοποιήσης των συστατικών στοιχείων (components). Τα components είναι επαναχρησιμοποιούμενα κομμάτια κώδικα που επιστρέφουν ένα React element για να συμπεριληφθεί στην σελίδα μας.[13] Συνδυάζοντας πολλά components μπορούμε να δημιουργήσουμε σύνθετες σελίδες. Ως κώδικας τα components μπορούν να γραφούν υπό την μορφή κλάσσεων ή συναρτήσεων. Στην παρούσα εργασία θα χρησιμοποιήσουμε την συναρτησιακή επιλογή, που είναι η πιο σύγχρονη. Τα components μπορούν να περιλαμβάνουν άλλα components στο εσωτερικό τους. Με την λογική αυτή μπορούμε να δούμε το τελικό αποτέλεσμα σαν ένα δένδρο με την ιεραρχία των components. Τα components λαμβάνουν είσοδο (props) από τα components που τα χρησιμοποιούν, μέσω της οποίας διαμορφώνουν την λειτουργία τους. Σε αντίθεση με τα props που μπορούμε να τα διαβάσουμε μόνο, τα components μπορούν να περιέχουν καταστάσεις (states), οι οποίες μπορούν να αλλάζουν με τον χρόνο. Η διαχείρηση των states γίνεται εσωτερικά του component με λογική που γράφουμε για να εξυπηρετούμε τους σκοπούς της εφαρμογής. Όταν αλλάζει η τιμή του state, τότε το component επαναποδίδεται από την React στην σελίδα μας μαζί με τα components που περιέχει.
Ένας από τους λόγους που η ανάπτυξη εφαρμογών είναι σχετικά απλή με την React, είναι

τα Hooks. Ως Hooks ονομάζουμε συναρτήσεις με τις οποίες μπορούμε να χειριζόμαστε τα states των components καθώς και να εκτελούμε διάφορες άλλες λειτουργίες με έναν εύκολο τρόπο. Με τα Hooks μπορούμε να γράφουμε τα components σε μορφή συνάρτησης κάτι που απλοποιεί συντακτικά σε μεγάλο βαθμό τον κώδικα των εφαρμογών. Τα βασικότερα Hooks είναι τα useState, useEffect και useContext. Με την useState προσθέτουμε ένα state στο component εντός του οποίου την καλούμε, θέτοντας αν θέλουμε και την αρχική τιμή. Η συνάρτηση επιστρέφει την τρέχουσα τιμή του state καθώς και μία συνάρτηση με την οποία μπορούμε να αλλάξουμε την τιμή του.[23] Το useEffect Hook δίνει την δυνατότητα πραγματοποίησης των λεγόμενων παρενεργειών από ένα component που έχει συναρτησιακή μορφή. Μία χαρακτηριστική παρενέργεια είναι η λήψη δεδομένων μέσω αιτήματος σε API. Η useEffect λαμβάνει ως όρισμα μια συνάρτηση και έναν πίνακα εξαρτήσεων. Η συνάρτηση εκτελείται αν αλλάξει η τιμή μίας από τις εξαρτήσεις του πίνακα αυτού. Αν δώσουμε κενό πίνακα εξαρτήσεων τότε η συνάρτηση θα τρέξει μόνο μία φορά μετά την αρχική απόδοση του component. Μπορούμε επίσης να παραλήψουμε τον πίνακα εξαρτήσεων. Σε αυτήν την περίπτωση η συνάρτηση θα τρέχει σε κάθε επανεκτίμηση του component. Μια άλλη έννοια στη ReactJS είναι το πλαίσιο (context), το οποίο δίνει την δυνατότητα να χρησιμοποιούμε απευθείας δεδομένα και συναρτήσεις σε διαφορετικά components, χωρίς όμως να τα παρέχουμε ως props. Το useContext λαμβάνει ως όρισμα ένα δημιουργηθέν context object και επιστρέφει την τρέχουσα τιμή για το context αυτό. Σημειώνεται πως η React μας επιτρέπει να δημιουργήσουμε και τα δικά μας Hooks.
Αξίζει να αναφέρουμε πως υπάρχει μεγάλη ποικιλία βιβλιοθηκών διαθέσιμων για την React, οι οποίες δίνουν την δυνατότητα να υλοποιήσουμε διάφορες λειτουργίες στις εφαρμογές μας. Στα πλαίσια της παρούσας εργασίας χρησιμοποιήσαμε τη plotly.js για να εισάγουμε γραφικές παραστάσεις στην εφαρμογή και την react-paginate για την σελιδοποίηση των ολοκληρωμένων προβλέψεων κατά την παρουσίασή τους. Σε κάθε περίπτωση χρησιμοποιούμε τον διαχειριστή πακέτων npm (Node Package Manager) για να εγκαταστήσουμε τις βιβλιοθήκες μας.

BackEnd Service

Για το Back-End κομμάτι της εφαρμογής χρησιμοποιήσαμε αποκλειστικά την γλώσσα προγραμματισμού Python. Επιλέξαμε την γλώσσα αυτή λόγω της απλότητας που την χαρακτηρίζει, αλλά και την μεγάλη ποικιλία βιβλιοθηκών που την συνοδεύουν και αφορούν την ανάλυση δεδομένων και την μηχανική μάθηση. Ειδικότερα, δημιουργήσαμε ένα RESTful API με το framework FastAPI. Το FastAPI χαρακτηρίζεται από υψηλή απόδοση, ενώ η εκμάθησή του είναι πολύ εύκολη[1]. Η υψηλή απόδοση μεταφράζεται σε δυνατότητα εξυπηρέτησης μεγάλου αριθμού αιτημάτων ανά δευτερόλεπτο. Το FastAPI υπερέχει σ'αυτόν τον τομέα σε σχέση με άλλα δημοφιλή frameworks της Python, όπως το Django και Flask[47]. Το FastAPI υποστηρίζει την εκτέλεση ασύγχρονης Python. Η λογική του ασύγχρονου κώδικα είναι πως σε περίπτωση που κάποιο σημείο απαιτεί σημαντικό χρόνο για να εκτελεστεί, η Python δεν θα μείνει αδρανής περιμένοντας, αλλά θα εκτελέσει κάποιο άλλο έργο (π.χ. θα εξυπηρετήσει άλλο αίτημα). Παραδείγματα περιπτώσεων που απαιτούν χρόνο είναι η εκτέλεση ερωτημάτων σε βάση δεδομένων και η εγγραφή δεδομένων στον δίσκο. Λεπτομέρειες για την υλοποίηση του API θα δούμε παρακάτω. Η σύνδεση και η διάδραση με την βάση δεδομένων MongoDB πραγματοποιήθηκε μέσω της βιβλιοθήκης Motor. Όσον αφορά τις λειτουργίες μηχανικής μάθησης και ανάλυσης δεδομένων χρησιμοποιήθηκαν οι βιβλιοθήκες Tensorflow, Pandas, NumPy και Scikit-Learn.
Από την φύση της η διαδικασία διαμόρφωσης των αλγορίθμων μηχανικής μάθησης είναι μια διαδικασία που απαιτεί σημαντικό χρόνο. Ωστόσο, σε μία εφαρμογή ο χρόνος αυτός σε πολλές περιπτώσεις θεωρείται απαγορευτικός. Συνεπώς, για να βελτιώσουμε την ταχύτητα με την οποία παράγει αποτέλεσμα η εφαρμογή μας θα εκτελέσουμε μέρος των υπολογισμών μέσω μίας κατανεμημένης αρχιτεκτονικής. Για τον σκοπό αυτό θα χρησιμοποιήσουμε την

βιβλιοθήκη ανοιχτού κώδικα Dask.
Το Dask μας δίνει την δυνατότητα με ελάχιστες αλλαγές στον Python κώδικά μας, να τρέξουμε σημαντικές λειτουργίες κατανεμημένα σε ένα σύνολο υπολογιστικών κόμβων. Με άλλα λόγια, μας παρέχει ένα πλαίσιο για κατανεμημένη υπολογιστική (Distributed Computing).[38] Το Dask χρησιμοποιείται από πολλές επιχειρήσεις και οργανισμούς, όπως η Walmart και η NASA, για διάφορες εργασίες, με κυριότερες την ανάλυση μεγάλου όγκου δεδομένων και την μηχανική μάθηση. Στην περίπτωση αυτή, οι κόμβοι έχουν δική τους μνήμη και συνδέονται μέσω ενός δικτύου. Η επικοινωνία και ο συντονισμός γίνεται με ανταλλαγή μηνυμάτων διαμέσου του δικτύου, μέσω πρωτοκόλλου TCP. Το σύνολο των κόμβων αυτών αποτελεί μία συστάδα (Cluster). Υπάρχουν πολλές επιλογές για το πως θα σχηματίσουμε τεχνικά το Cluster. Μία επιλογή είναι να χρησιμοποιήσουμε υπολογιστές σε ένα τοπικό δίκτυο ή υπολογιστές σε απομακρυσμένες τοποθεσίες (SSH Cluster)[2]. Επίσης, μπορούμε να δημιουργήσουμε Cluster μέσω Cloud υπηρεσιών. Τέλος, μπορούμε να φτιάξουμε ένα Cluster σε ένα μηχάνημα, θεωρώντας ως υπολογιστικούς κόμβους διαφορετικές διεργασίες επί ενός πολυπύρηνου επεξεργαστή. Στα πλαίσια της διπλωματικής εργασίας ακολουθήσαμε την πρώτη επιλογή.
Κάθε Cluster αποτελείται από έναν πελάτη (Client), έναν προγραμματιστή (Scheduler) και έναν ή περισσότερους εργάτες (Workers). Ο πελάτης αποτελεί το σημείο επαφής του χρήστη με το Cluster και φιλοξενεί τον κώδικά του. Ο κώδικας αυτός σε μορφή γράφου εργασιών (Task Graph) δίνεται στον προγραμματιστή, ο οποίος διαχειρίζεται την ροή των εργασιών και στέλνει τις εργασίες στους εργάτες. Οι εργάτες εκτελούν τις εργασίες που τους ανατίθονται, αποθηκεύουν και μοιράζονται τα αποτελέσματα και ενημερώνουν τον προγραμματιστή, ο οποίος γνωρίζει τι έχει ολοκληρωθεί και τι εκκρεμεί. Τόσο οι εργάτες, όσο και ο προγραμματιστής είναι διεργασίες. Για τους εργάτες ορίζουμε κατά την δημιουργία τους το πλήθος των νημάτων που διαθέτουν προς υλοποίηση των εργασιών. Μεγαλύτερο πλήθος νημάτων αυξάνει τις δυνατότητες του εργάτη.



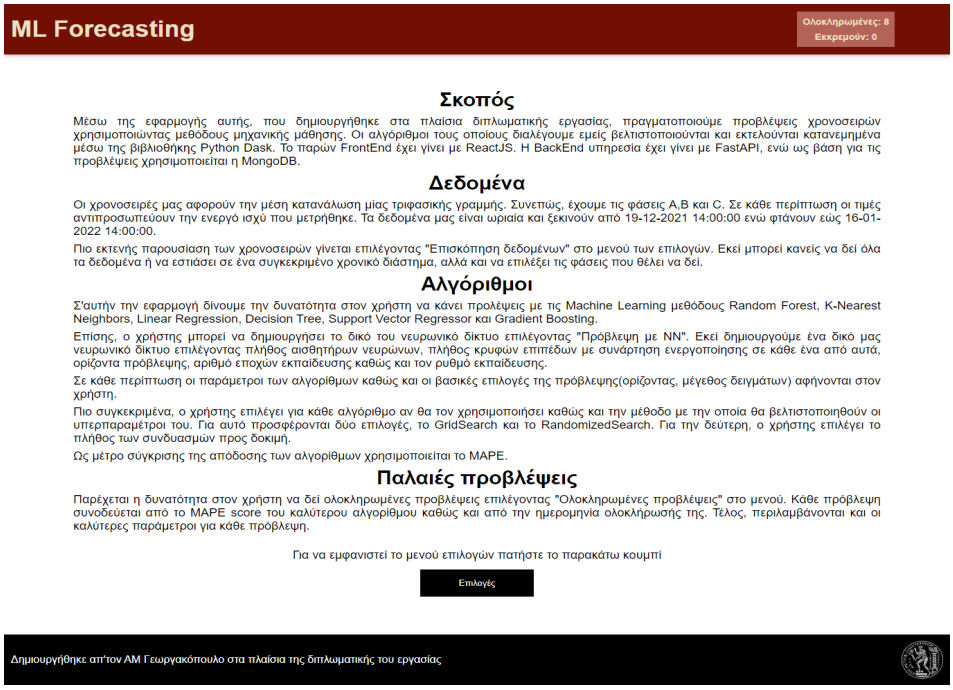
Σχήμα 6.1: Η αρχιτεκτονική του Dask Cluster [38]

Αξίζει να σημειώσουμε πως εναλλακτική του Dask αποτελεί το ευρέως χρησιμοποιούμενο Apache Spark. Επιλέξαμε το Dask διότι η ενσωμάτωσή του στον κώδικά μας ήταν ιδιαίτερα απλή, ενώ η δημιουργία και η ρύθμιση του Cluster ήταν σημαντικά απλούστερη απ'την αντίστοιχη διαδικασία στο Apache Spark. Τέλος, προτιμήσαμε το Dask και επειδή μας προσφέρει μία σειρά από γραφικά εργαλεία ελέγχου της διαδικασίας υλοποίησης των εργασιών (Dask Dashboard).

6.2 Παρουσίαση Εφαρμογής

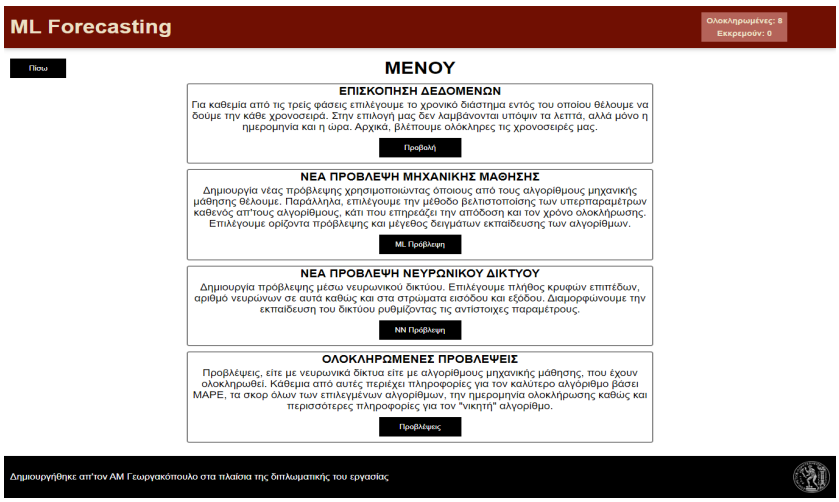
FrontEnd Client

Όπως αναφέρθηκε η εφαρμογή μας είναι τύπου μίας σελίδας (Single Page Application). Η σελίδα μας αποτελείται από τρία μέρη, την κεφαλίδα, το κυρίως περιεχόμενο και το υποσέλιδο. Το υποσέλιδο (Footer) περιέχει μόνο στοιχεία για τον δημιουργό της εφαρμογής, ενώ η κεφαλίδα (Header) περιέχει πέραν του ονόματος της εφαρμογής και δύο μετρητές. Ο ένας δείχνει το πλήθος των ολοκληρωμένων προβλέψεων και ο δεύτερος το πλήθος των εκκρεμουσών προβλέψεων. Για την κεφαλίδα και το υποσέλιδο έχουμε τα αντίστοιχα components ενώ το κυρίως περιεχόμενο σχηματοποιείται βάσει της κατάστασης της εφαρμογής, η οποία δηλώνεται μέσω της μεταβλητής κατάστασης globalState. Αναλόγως της τιμής της μεταβλητής αυτής το κυρίως περιεχόμενο παίρνει την μορφή του βάσει ενός ξεχωριστού component. Η εφαρμογή μας μπορεί να βρίσκεται σε 7 καταστάσεις. Για κάθε μία κατάσταση έχουμε το αντίστοιχο component, το οποίο βρίσκεται εντός ενός ομόνυμου αρχείου JavaScript και συνοδεύεται από ένα αρχείο με κώδικα CSS για την διαμόρφωση της εμφάνισης του HTML μέρους του.
Για globalState = 'Welcome', εμφανίζεται η σελίδα καλωσορίσματος στην οποία γράφουμε πληροφορίες για την ταυτότητα της εφαρμογής και τις δυνατότητες που προσφέρει στον χρήστη. Από την σελίδα αυτή οδηγούμαστε στο μενού πατώντας στο κουμπί με την ένδειξη "Επιλογές".



Σχήμα 6.2: Η σελίδα καλωσορίσματος

Για globalState = 'Menu', εμφανίζεται η σελίδα επιλογών (μενού). Κάθε επιλογή συνοδεύεται από ένα μικρό επεξηγηματικό κείμενο καθώς και ένα κουμπί πατώντας το οποίο οδηγούμαστε στην σελίδα της εκάστοτε επιλογής. Απ'το μενού μπορούμε επίσης να επιστρέψουμε στην αρχική σελίδα μέσω του κουμπιού με ένδειξη "Πίσω".
Για globalState = 'Phases Plots', παρουσιάζεται η πρώτη επιλογή, δηλαδή η επισκόπηση των τριών χρονοσειρών. Αρχικά, εμφανίζονται ολόκληρες και οι τρείς χρονοσειρές μας. Για κάθε χρονοσειρά μπορούμε να επιλέξουμε το χρονικό κομμάτι που μας ενδιαφέρει και πατώντας 'Προβολή' να εμφανίστει. Σε περίπτωση που δώσουμε μη έγκυρο διάστημα εμφανίζεται με κόκκινα γράμματα ένα μήνυμα λάθους. Σημειώνεται πως πρέπει να επιλέξουμε ημερομηνίες

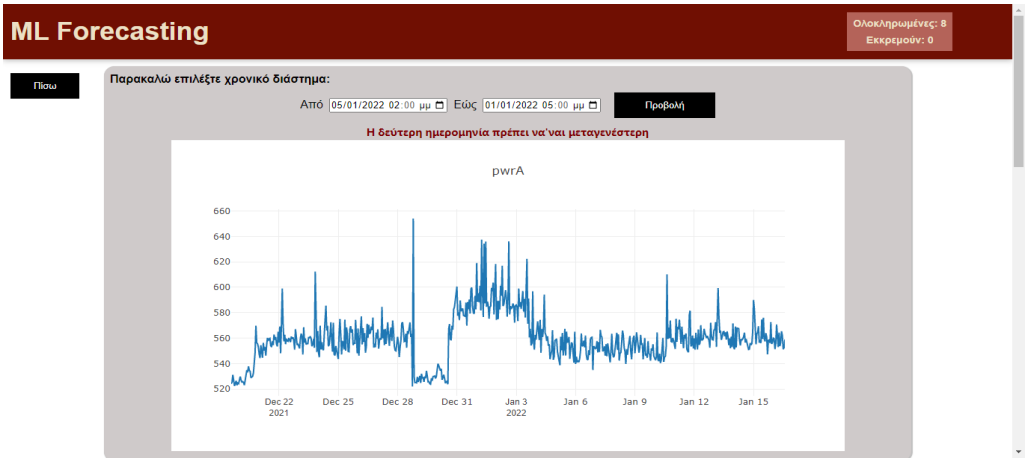


Σχήμα 6.3: Το μενού επιλογών

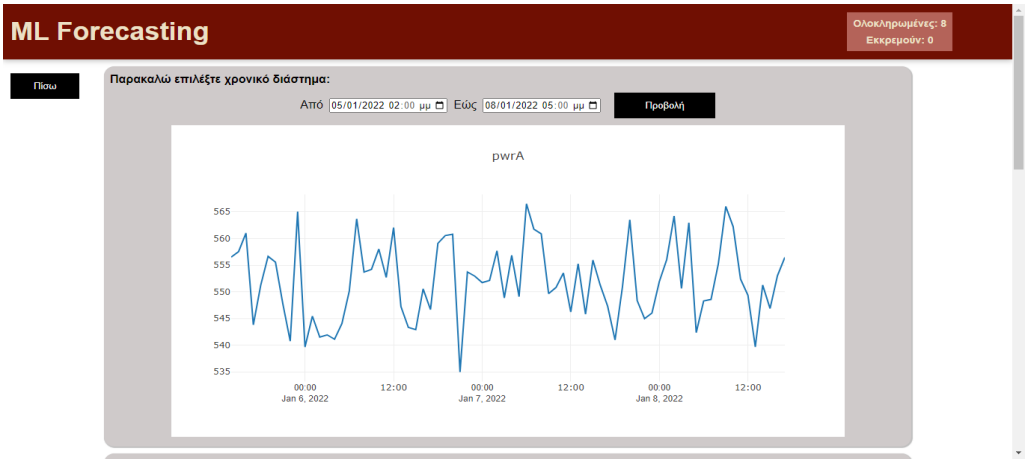
Σχήμα 6.4: Η σελίδα επισκόπησης των δεδομένων

σε επίπεδο ώρας, δηλαδή με μηδενικά λεπτά, καθότι τα δεδομένα μας είναι οργανωμένα σε αυτό το επίπεδο. Μη έγκυρο διάστημα μπορεί να έχουμε αν η ημερομηνία έναρξης είναι μεταγενέστερη της ημερομηνίας λήξης, εάν μία από τις δύο ημερομηνίες ξεπερνάει χρονικά τα όρια των σειρών μας, αν κάποια ημερομηνία δεν έχει την κατάλληλη μορφή ή δεν έχει δοθεί.

Για globalState = 'New Forecast', έχουμε την δεύτερη επιλογή που εμφανίζει την φόρμα για την επιλογή των χαρακτηριστικών της πρόβλεψης με αλγορίθμους μηχανικής μάθησης. Εδώ καλούμαστε, αφότου γράψουμε το όνομα του project πρόβλεψης που θα δημιουργηθεί, να επιλέξουμε ποιές φάσεις θα προβλέψουμε με τους αλγορίθμους, τον ορίζοντα για κάθε πρόβλεψη και το μέγεθος του παραθύρου για αυτές. Ως μέγεθος παραθύρου ορίζουμε τον αριθμό των παρελθοντικών παρατηρήσεων που θα χρησιμοποιούνται ως είσοδοι στο μοντέλο. Στην συνέχεια, επιλέγουμε όσους από τους έξι διαθέσιμους αλγορίθμους θέλουμε για να κάνουμε τις ζητούμενες προβλέψεις. Οι διαθέσιμοι αλγόριθμοι, οι οποίοι έχουν αναλυθεί στο κεφάλαιο 3 της παρούσας εργασίας, είναι ο αλγόριθμος των κ-κοντινότερων γειτόνων, η γραμμική παλινδρόμηση, το τυχαίο δάσος, η ενίσχυση κλίσης, το δένδρο αποφάσης και οι μηχανές διανυσμάτων υποστήριξης. Για όλους τους αλγορίθμους διαλέγουμε την μέθοδο

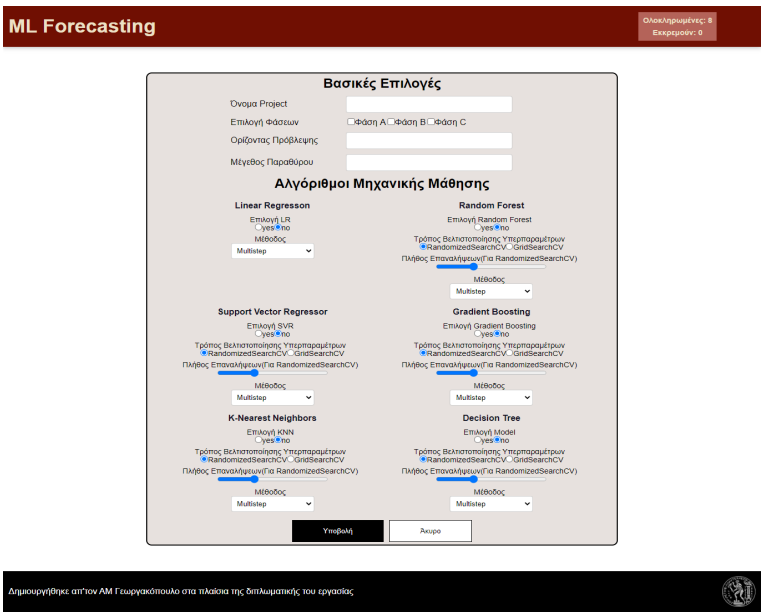


Σχήμα 6.5: Προβολή μηνύματος λάθους

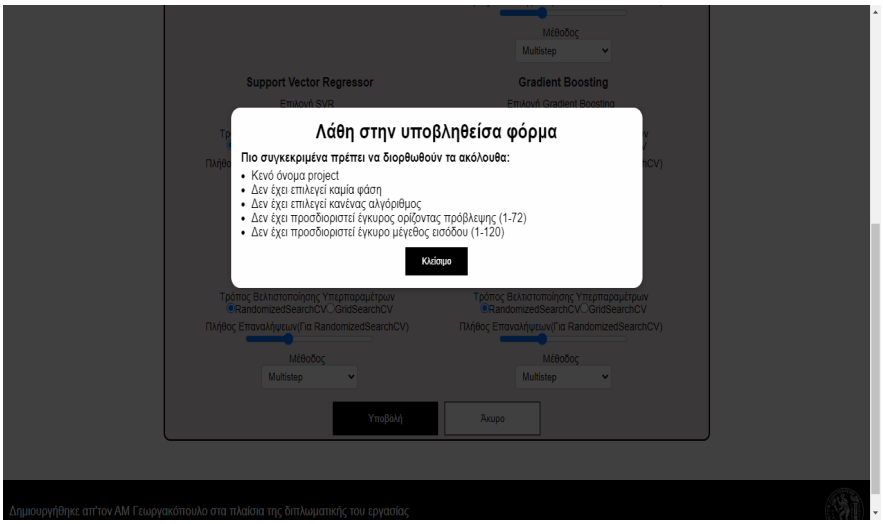


Σχήμα 6.6: Προβολή διαστήματος δεδομένων

που θα χρησιμοποιηθεί για την παραγωγή της πρόβλεψης. Διαθέσιμες επιλογές είναι η multistep (αναδρομική), η multimodel (απευθείας) και η chained-multimodel (DirRec). Επιπλέον, για όλους τους αλγορίθμους, πλην της γραμμικής παλινδρόμησης για την οποία λόγω του μικρού αριθμού παραμέτρων τρέχουμε πάντοτε αναζήτηση πλέγματος, διαλέγουμε την μέθοδο για την βελτιστοποίηση των υπερπαραμέτρων και το πλήθος των επαναλήψεων για την περίπτωση που ως μέθοδος επιλεγεί η τυχαία αναζήτηση (RandomizedSearchCV). Ως δεύτερη επιλογή προσφέρεται η αναζήτηση πλέγματος (GridSearchCV).
Μετα την υποβολή της φόρμας με τις προτιμήσεις μας για τις ζητούμενες προβλέψεις πραγματοποιείται έλεγχος και σε περίπτωση που υπάρχουν λάθη ή παραλήψεις παρουσιάζεται σχετικό μήνυμα. Για τον τρόπο που εμφανίζεται το μήνυμα θα αναφερθούμε στην συνέχεια. Για την εξαφάνιση του μηνύματος είτε πατάμε στο κουμπί με την ένδειξη "Κλείσιμο" είτε οπουδήποτε στο παρασκήνιο. Ως λάθη θεωρούμε την υποβολή φόρμας με κενό όνομα project ή όνομα project που έχει ήδη δοθεί στο παρελθόν, την μη επιλογή τουλάχιστον μίας φάσης για πρόβλεψη, την μη επιλογή τουλάχιστον ενός αλγορίθμου και την μη έγκυρη επιλογή ορίζοντα (1-72) ή μεγέθους παραθύρου (1-120).
Για globalState = 'New Neural', βλέπουμε την τρίτη επιλογή κατά την οποία διαμορφώνουμε προβλέψεις με νευρωνικό δίκτυο. Πιο συγκεκριμένα, επιλέγουμε και πάλι όνομα project, ποιές φάσεις θέλουμε να προβλέψουμε και τον ορίζοντα πρόβλεψης καθώς και τις απαραίτητες παραμέτρους για το νευρωνικό δίκτυο. Επιλέγουμε τις παραμέτρους που αφορούν την αρχιτεκτονική του νευρωνικού δικτύου, δηλαδή πλήθος νευρώνων επιπέδου



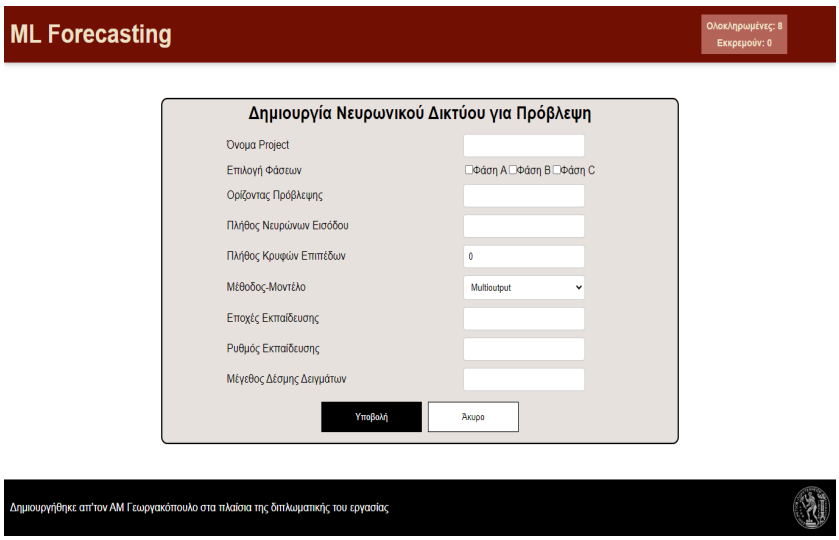
Σχήμα 6.7: Φόρμα για νέα πρόβλεψη



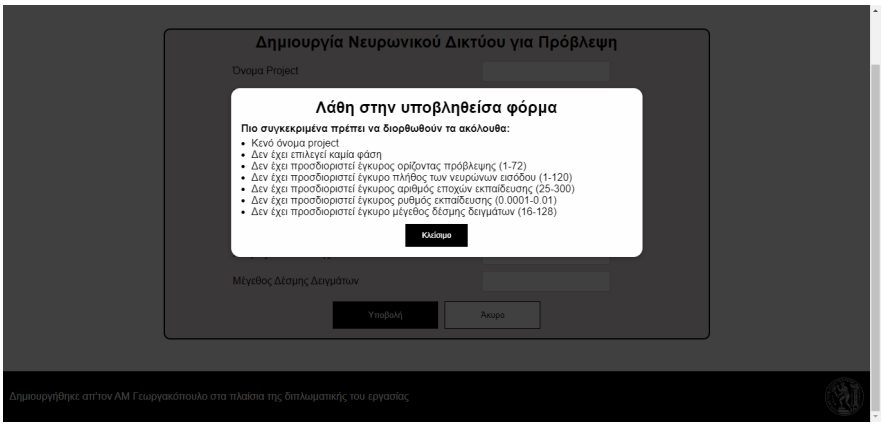
Σχήμα 6.8: Κάρτα με λάθη φόρμας (νέα πρόβλεψη)

εισόδου, πλήθος κρυφών επιπέδων, αριθμό νευρώνων και συνάρτηση ενεργοποίησης για κάθε ένα από αυτά (ReLU ή ELU), και αυτές που σχετίζονται με την εκπαίδευση του, δηλαδή τον αριθμό των εποχών εκπαίδευσης, το μέγεθος κάθε δέσμης δειγμάτων και τον ρυθμό εκπαίδευσης. Σημειώνεται πως αφού διαλέξουμε το πλήθος κρυφών επιπέδων, η φόρμα επεκτείνεται με επιλογές για κάθε ένα από τα επίπεδα αυτά. Όπως και για κάθε αλγόριθμο, έτσι και εδώ διαλέγουμε την τεχνική (στρατηγική) που θα ακολουθήσουμε για την παραγωγή προβλέψεων. Προσφέρουμε τρεις εναλλακτικές, την multioutput, την multistep (αναδρομικό μοντέλο) και το multimodel (απευθείας πρόβλεψη). Τέλος, μπορούμε υποβάλουμε την φόρμα με τις επιλογές μας ή να τις ακυρώσουμε και να επιστρέψουμε στο μενού επιλογών.

Όπως και στην προηγούμενη περίπτωση έτσι και εδώ μετα την υποβολή της φόρμας με τις προτιμήσεις μας για τις ζητούμενες προβλέψεις πραγματοποιείται έλεγχος και σε περίπτωση που υπάρχουν λάθη ή παραλήψεις παρουσιάζεται σχετικό μήνυμα. Για την εξαφάνιση του μηνύματος είτε πατάμε στο κουμπί με την ένδειξη "Κλείσιμο" είτε οπουδήποτε στο παρασκήνιο. Ως λάθη θεωρούμε την υποβολή φόρμας με κενό όνομα project ή όνομα project



Σχήμα 6.9: Φόρμα για πρόβλεψη με ΤΝΔ

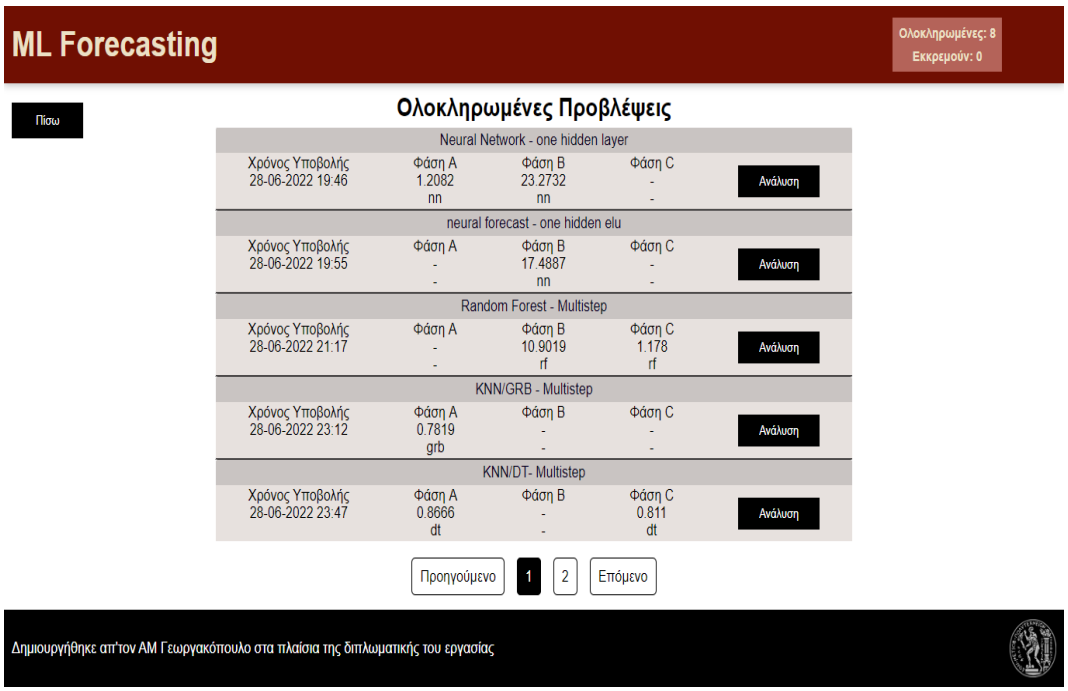


Σχήμα 6.10: Κάρτα με λάθη φόρμας (πρόβλεψη με ΤΝΔ)

που έχει ήδη δοθεί στο παρελθόν, την μη επιλογή τουλάχιστον μίας φάσης για πρόβλεψη και την μη έγκυρη επιλογή κάποιου ή κάποιων από τους ορίζοντα (1-72), μεγέθους παραθύρου (1-120), μέγεθος δέσμης (16-128), ρυθμού εκπαίδευσης (0.01-0.0001) και αριθμό εποχών (25-300). Λάθος θεωρούμε επίσης τον μη ολοκληρωμένο προσδιορίσμο του μεγέθους κάποιου ή κάποιων κρυφών επιπέδων.

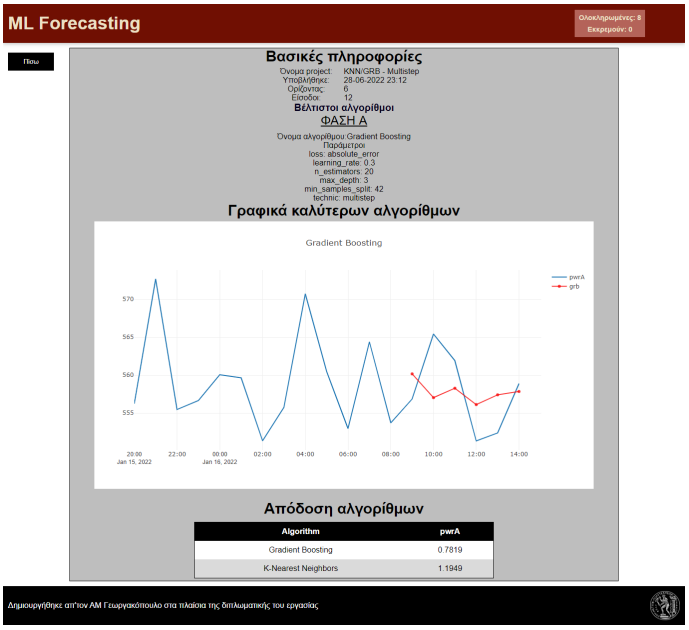
Για globalState = 'Completed Forecasts', έχουμε την τελευταία επιλογή του μενού, όπου βλέπουμε μία λίστα με ολοκληρωμένες προβλέψεις. Οι προβλέψεις εμφανίζονται ανά πέντε, δηλαδή έχουμε σελιδοποίηση στην παρουσίασή τους. Μπορούμε είτε να μεταβούμε στην επόμενη ή την προηγούμενη από την τρέχουσα σελίδα είτε να επιλέξουμε έναν συγκεκριμένο αριθμό σελίδας προβλέψεων που επιθυμούμε να εμφανιστεί. Για κάθε πρόβλεψη βλέπουμε πέραν του ονόματος της, τον χρόνο υποβολής της, τον αλγόριθμο με την καλύτερη απόδοση σε κάθε φάση που επιλέξαμε να προβλέψουμε, καθώς και το MAPE αυτού. Σε κάθε εγγραφή της λίστας υπάρχει ένα κουμπί, με ένδειξη "Ανάλυση", με το οποίο ο οδηγούμαστε σε μία αναφορά για την εκάστοτε πρόβλεψη.

Η αναφορά αυτή αντιστοιχεί σε globalState = 'View Forecast'. Η ανάλυση του project προβλέψεων χωρίζεται σε τρία μέρη. Στο πρώτο μέρος έχουμε τις βασικές πληροφορίες για τις προβλέψεις, δηλαδή βλέπουμε το όνομα του project, τον χρόνο υποβολής, τον ορίζοντα πρόβλεψης και το πλήθος των εισόδων για την πρόβλεψη. Ακολούθως, για κάθε φάση παρουσιάζεται ο βέλτιστος αλγόριθμος και οι παράμετροί του. Αν έχουμε μέθοδο πολλαπλών μοντέλων οι παράμετροι παραλείπονται. Στο δεύτερο μέρος βλέπουμε



Σχήμα 6.11: Λίστα με τις ολοκληρωμένες προβλέψεις

την γραφική παράσταση των προβλέψεων μαζί με αυτήν της προς πρόβλεψη χρονοσειράς. Στο τρίτο μέρος παρουσιάζεται ένας πίνακας με την απόδοση όλων των αλγορίθμων που επιλέχθηκαν σε όλες τις φάσεις για τις οποίες ζητήθηκε να γίνει πρόβλεψη.



Σχήμα 6.12: Αναλυτική παρουσίαση πρόβλεψης

Πριν αναλύσουμε το κομμάτι του BackEnd θα εξετάσουμε κάποια χαρακτηριστικά κομμάτια κώδικα από το FrontEnd. Σημειώνεται πως ολόκληρος ο κώδικας της εφαρμογής, τόσο για το FrontEnd όσο για το BackEnd, μαζί με τον υπόλοιπο κώδικα της εργασίας, βρίσκεται στο αποθετήριο github (`https://github.com/akisgeorgakopoulos/ML-Forecasting`). Θα ξεκινήσουμε με τον κώδικα ενός component που περιλαμβάνει πολλά από τα στοιχεία που αναφέρθηκαν προηγουμένως. Στο PhasesPlots.js έχουμε τον κώδικα για την περίπτωση που επιλέξουμε να δούμε τις χρονοσειρές μας (1η επιλογή μενού). Ο κώδικας είναι ο παρακάτω:

```
import React, {useEffect, useState} from 'react';
import PhasePlot from './PhasePlot';
import classes from './PhasesPlots.module.css';

const PhasesPlots = props => {
  const backHandler = () => {
        props.onBack('Menu')
  }
  const [isLoading,setIsLoading] = useState(true);
  const [loadedData,setLoadedData] = useState({"time":[],
                                               "pwrA":[],
                                               "pwrB":[],
                                               "pwrC":[]
                                              });
  useEffect(() => {
      window.scrollTo(0, 0);
      const fetchData = async () =>{
          const url = 'http://localhost:8000/data'
          const response = await fetch(url);
          if (!response.ok){
            throw new Error();
          }
          const responseData = await response.json();
          let timeseries = {"time":[],
                            "pwrA":[],
                            "pwrB":[],
                            "pwrC":[]};
          responseData.forEach(datapoint => {
              timeseries.time.push(datapoint.Time);
              timeseries.pwrA.push(datapoint.pwrA);
              timeseries.pwrB.push(datapoint.pwrB);
              timeseries.pwrC.push(datapoint.pwrC);
          }
          );
          setLoadedData(timeseries);
          setIsLoading(false);
      }
      fetchData().catch(error => {
          alert('Σφάλμα στην φόρτωση των χρονοσειρών...')
      });
    }, [])

  return(
          <>
              <button className={classes['back-button']}
              onClick={backHandler}>Πίσω
              </button>
              {isLoading &&
              <div className={classes["main-content"]}>
                  <h2 className={classes.title}>Loading...</h2>
              </div>
              }
              {!isLoading &&
              <>
                  <PhasePlot
                  phase='pwrA'
                  data={{"time":loadedData.time,
```

```
58                          "phase":loadedData.pwrA}}
59                  />
60                  <PhasePlot
61                  phase='pwrB'
62                  data={{"time":loadedData.time,
63                          "phase":loadedData.pwrB}}
64                  />
65                  <PhasePlot
66                  phase='pwrC'
67                  data={{"time":loadedData.time,
68                          "phase":loadedData.pwrC}}
69                  />
70              </>
71              }
72          </>
73      )
74  }
75
76  export default PhasesPlots;
```

Στις πρώτες τρεις γραμμές εισάγουμε τα απαραίτητα Hooks καθώς και το component PhasePlot που όπως θα δούμε χρησιμοποιείται για να δομήσουμε το συνολικό component PhasePlots. Στην τρίτη γραμμή συνδεόμαστε με τον κώδικα CSS για να διαμορφώσουμε την εμφάνιση των στοιχείων HTML που περιέχει το component μας. Ο κώδικας αυτός βρίσκεται στο αρχείο "PhasesPlots.module.css" εντός του ίδιου φακέλου. Όλοι οι φάκελοι που περιέχουν τα components με τα συνοδευτικά αρχεία CSS και ενδεχομένως και άλλα αρχεία με components (και τα αντίστοιχα αρχεία με CSS) βρίσκονται στον φάκελο components. Σε κάθε στοιχείο HTML ορίζουμε την ιδιότητα class (className στην JSX) ώστε μέσω αυτής να μπορούμε να το επιλέξουμε και να εφαρμόσουμε επί αυτού ένα σύνολο κανόνων που βρίσκονται εντός του κώδικα CSS. Στις γραμμές έξι έως οκτώ ορίζουμε την συνάρτηση backHandler με την οποία χειριζόμαστε το πάτημα του κουμπιού 'Πίσω'. Όπως φαίνεται από τον κώδικα, η συνάρτηση αυτή καλεί μια συνάρτηση που παρέχεται ως μέρος της εισόδου στο component, με όρισμα 'Menu'. Η συνάρτηση αυτή αλλάζει το globalState στην τιμή του ορίσματος με αποτέλεσμα να μετακινούμαστε στο μενού επιλογών. Για να χειριστούμε το πάτημα το κουμπιού, θέτουμε την event ιδιότητα onClick του στοιχείου που αντιπροσωπεύει το κουμπί, δηλαδή το στοιχείο button της γραμμής 45, που καθορίζει το τι γίνεται με το πάτημα του κουμπιού με την αναφορά στην προαναφερθήσα συνάρτηση. Έτσι όταν πατάμε το κουμπί "Πίσω" καλείται η συνάρτηση backHandler και οδηγούμαστε πίσω στο μενού. Ακολούθως, στις γραμμές 9-14 ορίζουμε μέσω του Hook useState δύο μεταβλητές κατάστασης, την isLoading που θα λαμβάνει Boolean τιμές και την loadedData που λαμβάνει ως τιμές αντικείμενα με ζεύγη κλειδιών-τιμών, όπου ως τιμές έχουμε τα δεδομένα των χρονοσειρών. Η loadedData θα περιέχει τα δεδομένα που θα λάβουμε μέσω request από το BackEnd (και κατ'επέκταση από την βάση δεδομένων μας). Όσον αφορά την isLoading, όπως φαίνεται και από το όνομά της, θα έχει τιμή True εάν η διαδικασία φόρτωσης των δεδομένων είναι εν εξελίξει και false εάν έχει ολοκληρωθεί και έχουμε τα δεδομένα μας.
Στις γραμμές 15-41 χρησιμοποιήσαμε το useEffect Hook. Η λίστα εξαρτήσεων είναι κενή, συνεπώς η συνάρτηση που βρίσκεται στο πρώτο όρισμα του Hook θα τρέξει μία φορά κατά την απόδοση του component. Στην γραμμή 16 έχουμε την μέθοδο scrollTo(0,0) προκειμένου μετά την φόρτωση να κατευθυνόμαστε αμέσως στην κορυφή της σελίδας. Η λογική μας είναι πως ο χρήστης πρέπει να πλοηγηθεί ξεκινώντας από την αρχή της σελίδας. Ακολούθως, στις γραμμές 17-37 ορίζουμε μία ασύγχρονη συνάρτηση, την fetchData. Εντός αυτής στέλνουμε ένα HTTP GET αίτημα στο API για την λήψη των δεδομένων μας. Αφού λάβουμε την απάντηση από το API θέτουμε ως false την μεταβλητή κατάστασης isLoading και ανανεώνουμε καταλλήλως την μεταβλητή loadedData. Αξίζει να αναφέρουμε και τον χειρισμό της περίπτωσης αποτυχίας του αιτήματος προς το API. Σε κάποιες περιπτώσεις,

για διάφορους λόγους, όπως σφάλματα δικτύου, ένα αίτημα μπορεί να μην επιστρέψει το επιθυμητό αποτέλεσμα. Σε αυτήν την περίπτωση λαμβάνουμε έναν κωδικό που σηματοδοτεί τον λόγο αποτυχίας (π.χ. 404). Στον κώδικα, αν υπάρξει τέτοια περίπτωση παράγουμε ένα Error. Στην συνέχεια κατά την κλήση της fetchData, στις γραμμές 38-40, αντιμετωπίζουμε την περίπτωση σφάλματος ενημερώνοντας τον χρήστη με κατάλληλο μήνυμα.
Στον παραπάνω κώδικα βλέπουμε πως το component έχει μορφή συνάρτησης (και ειδικότερα arrow function). Η συνάρτηση αυτή λαμβάνει είσοδους, στις οποίες έχουμε πρόσβαση στο σώμα της συνάρτησης μέσω της μεταβλητής props. Την συνάρτηση αυτή την εξάγουμε στην γραμμή 76 προκειμένου να την χρησιμοποιήσουμε σε κάποιο άλλο component αφού πρώτα την εισάγουμε, ακριβώς όπως εισάγαμε το component PhasePlot στην γραμμή 2. Όπως κάθε συνάρτηση έτσι και αυτή του component μας επιστρέφει κάποια τιμή. Στην προκειμένη περίπτωση επιστρέφεται JSX κώδικας. Στην React τα components επιστρέφουν αυστηρά ένα αντικείμενο. Για τον λόγο αυτό χρησιμοποιούμε τα Fragments, που αποτελούν περιτύλιγμα για την περίπτωση που θέλουμε να επιστρέψουμε ένα σύνολο αντικειμένων. Τα Fragments έχουν εδώ την μορφή κενής ετικέτας HTML (<> και </>) και βρίσκονται σε δύο ζεύγη στις γραμμές 44 και 72 και στην δεύτερη περίπτωση 54 και 70. Ο τρόπος αυτός αποτελεί την βραχεία σύνταξη (short syntax). Ένας άλλος τρόπος για να χρησιμοποιήσουμε Fragments είναι να γράψουμε $< React.Fragment >$ και ύστερα $< /React.Fragment >$. Τόσο στις γραμμές 48-52 όσο και στις 53-71 έχουμε περιπτώσεις απόδοσης υπό συνθήκη. Πιο συγκεκριμένα, μέσω μίας λογικής συνθήκης ελέγχουμε τι περιεχόμενο θα αποδοθεί ως τιμή του component. Αν η μεταβλητή isLoading είναι αληθής θα έχουμε ένα μήνυμα πως φορτώνει, ενώ σε αντίθετη περίπτωση θα έχουμε το περιεχόμενο μας, δηλαδή τις χρονοσειρές. Στην δεύτερη περίπτωση αποδίδονται στον χρήστη πέραν του κουμπιού (κάτι που συμβαίνει και στην πρώτη περίπτωση) ένα σύνολο με τρία components. Τα τρία αυτά components λαμβάνουν ως εισόδους την φάση που αναπαριστούν και τα απαιτούμενα δεδομένα.
Για την πρόσβαση και την ενημέρωση του αριθμού εκκρεμών και ολοκληρωμένων προβλέψεων χρησιμοποιήσαμε context και το useContext Hook. Για την χρήση των μεταβλητών και συναρτήσεων εντός του context απαιτούνται δύο βήματα, η παροχή του context και η κατανάλωση του. Σε ξεχωριστό φάκελο φτιάξαμε δύο αρχεία. Στο πρώτο "header-context.js" δημιουργήσαμε με την συνάρτηση createContext και εξάγαμε το context object. Εν συνεχεία, εισάγουμε το αντικείμενο αυτό και το μορφοποιούμε στο αρχείο "HeaderProvider.js". Στο αρχείο αυτό δημιουργούμε το ομόνυμο component μέσω του οποίου θα παρέχουμε το context στα components της εφαρμογής μας. Ο κώδικας για το component HeaderProvider είναι ο ακόλουθος:

```
1  import React, {useState} from 'react';
2  import HeaderContext from './header-context';
3
4  const HeaderProvider = props => {
5    const [completed,setCompleted] = useState(0);
6    const [pending,setPending] = useState(0);
7
8    const headerContext = {
9        completed:completed,
10       pending:pending,
11       addPending: () => {setPending((p)=> p+1)},
12       removePending: () => {setPending((p)=> p-1)},
13       updateCompleted: () => {
14           const fetchForecasts = async () =>{
15               const url = 'http://localhost:8000/completed-num';
16               const response = await fetch(url);
17               if (!response.ok){
18                 throw new Error();
19               }
```

```
20                const responseData = await response.json();
21                setCompleted(responseData.number);
22            }
23            fetchForecasts().catch(error =>{
24                alert('Σφάλμα κατά την εντοπισμό των προβλέψεων...');
25            })
26        }
27    }
28
29    return(
30        <HeaderContext.Provider value={headerContext} >
31            {props.children}
32        </HeaderContext.Provider>
33    )
34 }
35
36 export default HeaderProvider;
```

Στις γραμμές 29-34 επιστρέφουμε το Provider component, το οποίο εμπεριέχει τα components που έχουν πρόσβαση στο context, δηλαδή το καταναλώνουν. Στο χαρακτηριστικό value τοποθετούμε οτιδήποτε θέλουμε η αλλαγή του να αποτυπωθεί στα components καταναλωτές. Έτσι στον παραπάνω κώδικα δίνουμε πρόσβαση στους μετρητές ολοκληρωμένων και εκκρεμουσών προβλέψεων, καθώς και σε τρεις συναρτήσεις για τον άμεσο χειρισμό τους. Οι συναρτήσεις αυτές κάνουν χρήση των συναρτήσεων που επιστρέφει το useState Hook για την τροποποίηση της τιμής των μεταβλητών κατάστασης. Για να παρέχουμε λοιπόν το context, αφού εισάγουμε το HeaderProvider component, απλώς το χρησιμοποιούμε ως πατρικό component όσων θέλουμε να έχουν πρόσβαση στο context. Στην εφαρμογή μας, στο αρχείο App.js, που περιέχει το βασικό component που αναπαριστά την εφαρμογή, το HeaderProvider εσωκλείει όλα τα components μας.
Όπως αναφέρθηκε παραπάνω, τόσο στην πρόβλεψη με νευρωνικό δίκτυο όσο και σε αυτή με αλγορίθμους μηχανικής μάθησης πραγματοποιούμε έλεγχο των δεδομένων που υποβάλονται και σε περίπτωση σφάλματος ή παράλειψης εμφανίζουμε μία κάρτα με τα προς επίλυση ζητήματα. Για να το πετύχουμε αυτό βασιστήκαμε στην έννοια της πύλης (Portal). Πιο συγκεκριμένα, η React μας παρέχει την δυνατότητα να τοποθετήσουμε ένα component σε θέση που θα επιλέξουμε εμείς χωρίς να περιοριζόμαστε από το πατρικό component. Αρχικά, στο αρχείο index.html διαλέγουμε την θέση που θα πάρει το component μας. Τονίζεται πως στο div στοιχείο με id χαρακτηριστικό "root" τοποθετούμε την εφαρμογή μας, δηλαδή το App component. Αφού λοιπόν καθορίσουμε μέσω στοιχείου HTML μία θέση, μέσω της συνάρτησης ReactDOM.createPortal συνδέουμε το component (πρώτο όρισμα) με το στοιχείο που επιλέγεται βάσει της ιδιότητας id (δεύτερο όρισμα). Για παράδειγμα, στην εργασία μας γράφουμε:

```
1 const Modal = props => {
2   return(<>
3       {ReactDOM.createPortal(<Backdrop onClose={props.onClose}/>,
4                              document.getElementById('overlays'))}
5       {ReactDOM.createPortal(<ModalOverlay>{props.children}
6                               </ModalOverlay>,
7                              document.getElementById('overlays'))}
8   </>)
9 }
```

BackEnd Service

Το κομμάτι του BackEnd έχει υλοποιηθεί σε πέντε αρχεία. Τα πρώτα δύο (helpers.py και neural.py) περιέχουν τον κώδικα που υλοποιεί την απαραίτητη προεπεξεργασία των

δεδομένων και την διαδικασία των προβλέψεων. Τα υπόλοιπα τρία (main.py, model.py και database.py) χρησιμεύουν για την επικοινωνία με την βάση δεδομένων και τον ορισμό των σημείων μέσω των οποίων ο FrontEnd πελάτης έρχεται σε επαφή με την BackEnd υπηρεσία. Στην τελευταία περίπτωση ορίζουμε επακριβώς τι πρέπει να συμβεί μόλις η υπηρεσία μας λάβει ένα HTTP αίτημα.
Σημειώνεται πως προπαρασκευαστικά έχουμε τρέξει τον κώδικα του αρχείου prepare_utc.py, το οποίο διαβάζει τα δεδομένα όπως τα λαμβάνουμε από την βάση δεδομένων InfluxDB και αφού κάνει μία μετατροπή για την συμπερίληψη της μετατόπισης λόγω ζώνης ώρας υπολογίζει την μέση τιμή ανά ώρα. Έτσι τα δεδομένα μετατρέπονται σε ωριαία χρονοσειρά. Ακολούθως, αφαιρούμε τις ακραίες τιμές από τις χρονοσειρές μας. Στο αποτέλεσμα του προηγούμενου βήματος εφαρμόζουμε λογαρίθμηση και ύστερα παίρνουμε τις διαφορές πρώτου βαθμού του αποτελέσματος. Τέλος, αποθηκεύουμε το αποτέλεσμα σε ένα νέο αρχείο το οποίο χρησιμοποιούμε για να κάνουμε προβλέψεις στην συνέχεια. Περισσότερα για την προεπεξεργασία των δεδομένων αναφέρουμε στο αμέσως επόμενο κεφάλαιο.
Το αρχείο helpers.py περιέχει εκτός της *make_predictions*, μέσω της οποίας κάνουμε προβλέψεις με μεθόδους μηχανικής μάθησης, και τις βοηθητικές συναρτήσεις *get_samples*, *grid_optimization*, *rand_optimization* και *mape*. Η συνάρτηση *mape* επιστρέφει το MAPE μίας πρόβλεψης, η *get_samples* λαμβάνει ως είσοδο μία χρονοσειρά και βάσει του ορίζοντα, του μεγέθους εισόδου και της τεχνικής πρόβλεψης την μετατρέπει σε δείγματα εκπαίδευσης βάσει όσων αναφέραμε στο κεφάλαιο 3. Οι συναρτήσεις *grid_optimization* και *rand_optimization* χρησιμοποιούνται για να εκτελέσουμε grid search και randomized search επί συνόλων υπερπαραμέτρων που επίσης ορίζονται στο ίδιο αρχείο και καλύπτουν και τους έξι αλγορίθμους μας. Όσον αφορά την *make_predictions*, λαμβάνει ως παραμέτρους τον ορίζοντα πρόβλεψης, το μέγεθος της εισόδου, τις φάσεις προς πρόβλεψη, την υλοποίηση του αλγόριθμου, μία συμβολοσειρά που λειτουργεί ως κωδικός του αλγορίθμου για την επιλογή των υπερπαραμέτρων του, την τεχνική που θα ακολουθήσουμε για την πρόβλεψη, την μέθοδο βελτιστοποίησης των υπερπαραμέτρων του αλγορίθμου και το πλήθος των συνδυασμών που θα δοκιμάσουμε στην περίπτωση που επιλεχθεί η τυχαία αναζήτηση. Για την πρόβλεψη με αλγορίθμους μηχανικής μάθησης θα χρησιμοποιήσουμε τις πρώτες διαφορές των λογαριθμημένων δεδομένων. Τις τιμές αυτές τις περνάμε από την *get_samples* ώστε να φτιάξουμε τα δείγματα εκπαίδευσης και ελέγχου μας. Ακολούθως, μέσω της *grid_optimization* ή της *rand_optimization* βρίσκουμε τον βέλτιστο συνδυασμό υπερπαραμέτρων. Στην συνέχεια, αφού κάνουμε τις προβλέψεις μας με τις υπερπαραμέτρους αυτές, μέσω της τελευταίας τιμής του συνόλου εκπαίδευσης θα κατασκευάσουμε την λογαριθμημένη μορφή των προβλέψεων. Αμέσως μετά, μέσω της εκθετικής συνάρτησης θα λάβουμε τις τελικές τιμές των προβλέψεων. Αξίζει να σημειώσουμε πως χρησιμοποιούμε την συνάρτηση MultiOutputRegressor για την στρατηγική multimodel (direct) και RegressorChain για την στρατηγική chained-multimodel (DirRec), την TimeSeriesSplit για να πραγματοποιήσουμε την διασταυρούμενη-επικύρωση με την τροποποίηση για τις χρονοσειρές που αναλύσαμε στο κεφάλαιο 3. Και οι τρεις συναρτήσεις προέρχονται από την βιβλιοθήκη scikit learn.
Για την βελτιστοποίηση των υπερπαραμέτρων των αλγορίθμων μηχανικής μάθησης δοκιμάζουμε συνδυασμούς υπερπαραμέτρων είτε εξαντλητικά (grid search) είτε τυχαία (random search). Και για τις δύο περιπτώσεις έχουμε φτιάξει σύνολα με πιθανές τιμές. Σε κάθε περίπτωση τα σύνολα για την τυχαία αναζήτηση είναι πιο πλούσια καθότι θέλουμε να δώσουμε την δυνατότητα στο μοντέλο να δοκιμάσει μεγαλύτερη ποικιλία συνδυασμών, χωρίς να αυξάνεται το υπολογιστικό κόστος.
Για την περίπτωση της γραμμικής παλινδρόμησης (Linear Regression) εξετάζουμε μόνο την παράμετρο fit_intercept, η οποία παίρνει Boolean τιμές. Για τιμή True περιλαμβάνουμε στο μοντέλο μας εκτός των συντελεστών και την σταθερά (τιμή του μοντέλου για μηδενικές εισόδους), ενώ για False βρίσκουμε μόνο τους συντελεστές των μεταβλητών. Λόγω του μικρού πλήθους των παραμέτρων σε όλες τις περιπτώσεις τρέχουμε αναζήτηση πλέγματος

Παράμετρος	Τιμές (GS)
fit_intercept	True,False

Πίνακας 6.1: Υπερπαράμτροι γραμμικής παλινδρόμησης

για την εύρεση της βέλτιστης επιλογής.
Η βασική παράμετρος του αλγορίθμου των κ-κοντινότερων γειτόνων (K-Nearest Neighbors) είναι το πλήθος των γειτόνων που λαμβάνουμε υπόψιν στον υπολογισμό του αποτελέσματος. Η παράμετρος αυτή ονομάζεται n_neighbors. Πέραν τούτου, έχουμε επιλογές για τον ορισμό της απόστασης μεταξύ των δειγμάτων (metric), τον τρόπο συνδυασμού των τιμών των κ κοντινότερων γειτόνων (weights) καθώς και τον αλγόριθμο για την επιλογή των γειτόνων (algorithm). Τις ενδεχόμενες τιμές, που παρουσιάζουμε παρακάτω, τις έχουμε αναλύσει στο σχετικό κεφάλαιο. Σημειώνεται, πως η παράμετρος p αφορά την μετρική minkowski και την θεωρούμε σταθερή ίση με τρία, ενώ το μέγεθος φύλλου (leaf_size) αφορά τους αλγορίθμους KDTree και BallTree και επηρεάζει την ταχύτητα τους και τις απαιτήσεις τους για μνήμη αναφορικά με την δημιουργηθήσα δενδοειδή δομή.
Για τις μηχανές διανυσμάτων υποστήριξης (Support Vector Regressor) μπορούμε να διακρίνουμε δύο ειδών υπερπαραμέτρους, αυτές που αφορούν την συνάρτηση πυρήνα και όσες έχουν σχέση με την διαμόρφωση του γεωμετρικού αποτελέσματος του μοντέλου, δηλαδή της ευθείας και του περιθωρίου. Καταρχάς, έχουμε την παράμετρο epsilon που αποτελεί το μέγεθος του περιθωρίου του μοντέλου μας, δηλαδή η απόσταση της ευθείας της συνάρτησης του μοντέλου από κάθεμια από τις ευθείες που ορίζει το όριο του περιθωρίου εντός του οποίου επιδιώκουμε να βρίσκονται τα δείγματά μας. Συνεπώς, οι διάφορες τιμές του epsilon καθορίζουν το μέγεθος του περιθωρίου. Όσον αφορά το C, γνωρίζουμε πως πρόκειται για μία ποινή που δίνουμε για κάθε ένα δείγμα που έχει μείνει εκτός του περιθωρίου. Υψηλή τιμή της παραμέτρου αυτής μπορεί να οδηγήσει σε υπερπροσαρμογή του μοντέλου. Σχετικά με τις συναρτήσεις πυρήνα, έχουμε τέσσερις επιλογές, τις οποίες έχουμε παρουσιάσει στο κεφάλαιο 3. Η παράμετρος gamma χρησιμοποιείται στους πυρήνες RBF, Poly και Sigmoid. Η τιμή της gamma καθορίζει την καμπυλότητα της επιφάνειας του μοντέλου μας. Η παράμετρος coef0 συναντάται στους Poly και Sigmoid και αναφέρεται στην σταθερά που συμμετέχει στον τύπο των συναρτήσεων. Η παράμετρος degree αφορά μόνο την συνάρτηση Poly και αντιπροσωπεύει τον εκθέτη d του μαθηματικού τύπου της.
Για την μέθοδο του δένδρου αποφάσης (Decision Tree) έχουμε πέντε υπερπαραμέτρους. Καταρχάς, εξετάζουμε το κριτήριο με το οποίο αξιολογούμε την ποιότητα ενός διαχωρισμού

Παράμετρος	Τιμές (GS)	Τιμές (RS)
n_neighbors	2,7,...,97	2,3,...,99
algorithm	'auto', 'ball_tree', 'kd_tree', 'brute'	'auto', 'ball_tree', 'kd_tree', 'brute'
leaf_size	10,30,...,190	10,20,...,240
weights	'uniform', 'distance'	'uniform', 'distance'
metric	'minkowski','chebyshev', 'manhattan', 'euclidean'	'minkowski','chebyshev', 'manhattan', 'euclidean'
p	3	3

Πίνακας 6.2: Υπερπαράμτροι κ-κοντινότερων γειτόνων

Παράμετρος	Τιμές (GS)	Τιμές (RS)
kernel	'linear', 'poly', 'rbf', 'sigmoid'	'linear', 'poly', 'rbf', 'sigmoid'
C	1,21,41,61	1,11,21,...,91
degree	2,3,4,5	2,3,...,8
coef0	0.01, 0.10, 0.5, 0.8	0.01, 0.10, 0.3, 0.5, 0.8
gamma	'auto','scale'	'auto','scale'
epsilon	0.01, 0.09,..., 0.73	0.01,0.03,...,0.79

Πίνακας 6.3: Υπερπαράμετροι μηχανών διανυσμάτων υποστήριξης

(criterion). Έχουμε δύο επιλογές, το τετράγωνο του σφάλματος και το απόλυτο σφάλμα. Ακολούθως, ελέγχουμε διάφορες τιμές για το μέγιστο βάθος που επιτρέπουμε στο δένδρο να πάρει (max_depth). Αν η μεταβλητή αυτή λάβει την τιμή None, δεν επιβάλλουμε κανέναν περιορισμό για το βάθος του δένδρου, κάτι που μπορεί να έχει ως συνέπεια πολύπλοκα μοντέλα και πιθανώς φαινόμενα υπερπροσαρμογής. Πέραν του βάθους, έχουμε την παράμετρο min_samples_split που καθορίζει τον ελάχιστο αριθμό δειγμάτων που πρέπει να έχει ένας κόμβος για να επιτραπεί ο διαχωρισμός του. Αν δώσουμε μεγάλη τιμή δεν θα επιτρέψουμε πολλούς διαχωρισμούς και έτσι το τελικό δένδρο δεν θα είναι ιδιαίτερα πολύπλοκο, με κίνδυνο να μην μπορούμε να μοντελοποιήσουμε τα δεδομένα μας επαρκώς. Μία άλλη παράμετρος, με την οποία ασχολούμαστε είναι η min_samples_leaf, δηλαδή ο ελάχιστος αριθμός δειγμάτων που πρέπει να βρίσκεται σε ένα φύλλο. Για να εξετάσουμε έναν διαχωρισμό σε κάποιον κόμβο του δένδρου μας, βλέπουμε αν το πλήθος των δειγμάτων που αντιστοιχίζονται στο δεξί και αριστερό κλαδί είναι και τουλάχιστον ίσα με την τιμή της παραμέτρου αυτής. Μεγάλη τιμή της παραμέτρου οδηγεί σε μεγαλύτερα φύλλα και συνεπώς λιγότερο πολύπλοκα μοντέλα. Τέλος, έχουμε την παράμετρο ccp_alpha η οποία σχετίζεται με την διαδικασία κλαδέματος. Λαμβάνουμε υπόψιν μας δύο τιμές, την μηδενική κατά την οποία δεν συμβαίνει κλάδεμα στο δένδρο μας και την μη μηδενική (0.1), με την οποία επιτρέπουμε να γίνει κλάδεμα στο δένδρο μας.
Για τον αλγόριθμο του τυχαίου δάσους (Random Forest) "δανειζόμαστε" μία παραμέτρο από το δένδρο αποφάσης, το μέγιστο βάθος (max_depth), που αφορά την διαδικασία κατασκευής καθενός από τα δένδρα του μοντέλου. Εφαρμόζουμε την τεχνική bootstrap για

Παράμετρος	Τιμές (GS)	Τιμές (RS)
criterion	'squared_error', 'absolute_error'	'squared_error', 'absolute_error'
max_depth	2,5,10,40,None	2,3,5,8,10,15,20,40,100,None
min_samples_split	2,4,6,...,28	2,3,...,39
min_samples_leaf	1,3,5,...,29	1,2,...,39
ccp_alpha	0.0, 0.1	0.0, 0.1

Πίνακας 6.4: Υπερπαράμτροι δένδρων αποφάσης

την δημιουργία διαφορετικών συνόλων εκπαίδευσης με δειγματοληψία με αντικατάσταση από ένα ποσοστό των δειγμάτων εκπαίδευσης (max_samples), άρα θέτουμε True την παράμετρο bootstrap . Επίσης, διαλέγουμε τυχαία τις μεταβλητές εισόδου από ένα υποσύνολο τους με συγκεκριμένο πλήθος στοιχείων (max_features) για την επιλογή της συνθήκης διαχωρισμού σε κάθε βήμα κατασκευής του δένδρου. Το πλήθος αυτό μπορεί να είναι ένα συγκεκριμένο ποσοστό του συνόλου ή να προκύπτει από το πλήθος του συνόλου των εισόδων μέσω συνάρτησης (sqrt και log2), ενώ έχουμε συμπεριλάβει την περίπτωση όπου τα δύο σύνολα είναι ίδια (auto). Τέλος, ρυθμίζουμε το πλήθος των δένδρων αποφάσης (n_estimators) που συμμετέχουν στην διαμόρφωση του αποτελέσματος.

Παράμετρος	Τιμές (GS)	Τιμές (RS)
n_estimators	10,20,..,230,240	10,15,...,295
max_features	0.3, 0.4, 0.5, 0.6, 0.9,'sqrt','auto','log2'	0.3, 0.35, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9,'sqrt','auto','log2'
max_depth	2,5,10,40,None	2,5,10,20,40,100,None
max_samples	0.3, 0.5, 0.6, 0.8, 0.9	0.1,...,1.0
bootstrap	False	False

Πίνακας 6.5: Υπερπαράμετροι τυχαίου δάσους

Ο αλγόριθμος της ενίσχυσης κλίσης (Gradient Boosting) ανήκει στην κατηγορία των δενδροειδών αλγορίθμων. Ως εκ τούτου χρησιμοποιούμε και πάλι τις παραμέτρους max_depth και min_samples_split ως κατευθύνσεις για την δημιουργία των δένδρων αποφάσης του μοντέλου. Πέραν αυτών ρυθμίζουμε τον αριθμό των δένδρων που φτιάχνουμε (n_estimators), δηλαδή των σταδίων ενίσχυσης, καθώς και την συνεισφορά του κάθε εκτιμητή στην διαμόρφωση του συνολικού αποτελέσματος, μέσω του ρυθμού εκπαίδευσης (learning_rate). Τέλος, εξετάζουμε την συνάρτηση σφάλματος την οποία επιθυμούμε να βελτιστοποιήσουμε. Βάσει αυτής διαμορφώνονται οι εξισώσεις του μοντέλου μας. Στα πλαίσια της εργασίας μας, επιλέξαμε το απόλυτο σφάλμα και το τετραγωνικό σφάλμα.

Παράμετρος	Τιμές (GS)	Τιμές (RS)
loss	'squared_error', 'absolute_error'	'squared_error', 'absolute_error'
learning_rate	0.05,0.1,0.25,0.3,0.5	0.05, 0.10, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5, 0.75
n_estimators	10,20,...,240	10,20,...,290
max_depth	3,5,10,20	3,4,5,6,10,20
min_samples_split	1,12,22,32,42	2,7,12,.., 47

Πίνακας 6.6: Υπερπαράμετροι ενίσχυσης κλίσης

Το αρχείο neural.py περιέχει την συνάρτηση *nn_predictions*, η οποία λαμβάνει ως παραμέτρους τον ορίζοντα πρόβλεψης, το μέγεθος εισόδου, την φάση προς πρόβλεψη, την τεχνική πρόβλεψης, το πλήθος των κρυφών επιπέδων, το μέγεθος και την συνάρτηση

ενεργοποίσης καθενός από αυτά, το πλήθος των εποχών εκπαίδευσης, τον ρυθμό εκπαίδευσης και το μέγεθος της κάθε δέσμης δειγμάτων. Με την συνάρτηση αυτή αρχικά κατασκευάζουμε το τεχνητό νευρωνικό δίκτυό (ή τα δίκτυα αν έχουμε τεχνική multimodel) και το εκπαιδεύουμε σύμφωνα με τις προδιαγραφές που ορίζουν οι παράμετροι. Όσον αφορά την εκπαίδευση του μοντέλου αξίζει να σημειώσουμε πως χρησιμοποιούμε τον αλγόριθμο βελτιστοποίησης Adam, που αποτελεί μία μέθοδο στοχαστικής κατάβασης κλίσης (η στοχαστικότητα έχει να κάνει με τον τρόπο επιλογής των δεσμών κατά την εκπαίδευση), ενώ χρησιμοποιούμε την έννοια του πρόωρου σταματήματος (early stopping) σε περίπτωση που δεν υπάρχει σημαντική βελτίωση της απόδοσης του μοντέλου για τριάντα εποχές εκπαίδευσης. Για την μέτρηση της απόδοσης του μοντέλου κατά την εκπαίδευση κρατάμε το 10% των δεδομένων εκπαίδευσης χωρίς ανακάτεμμα λόγω της χρονολογικής φύσης τους. Με τον ίδιο τρόπο εκπαιδεύουμε και κάθενα από τα μοντέλα στην στρατηγική multimodel. Πριν όμως εκπαιδεύσουμε το μοντέλο μας ή τα μοντέλα μας, κανονικοποιούμε τα δεδομένα μας ώστε να βελτιώσουμε την εκπαίδευση του μοντέλου. Υπάρχουν διάφορες τεχνικές για αυτόν τον σκοπό. Επιλέξαμε την τεχνική της κλιμάκωσης μεγίστου-ελαχίστου. Πιο συγκεκριμένα, σε κάθε ένα από τα δεδομένα εισόδου ενός δείγματος εκπαίδευσης εφαρμόζουμε την ακόλουθη μετατροπή (X το διάνυσμα εισόδου, x ένα χαρακτηριστικό εντός του X):

$$x' = (x - X.min)/(X.max - X.min)$$

Ακολούθως, εφαρμόζουμε την ίδια μετατροπή στα δεδομένα εξόδου του δείγματος. Ο λόγος που δεν εφαρμόζουμε την μετατροπή σε όλα τα δεδομένα του δείγματος μαζί είναι πως θέλουμε να επιτρέψουμε τα δεδομένα εξόδου να παίρνουν και τιμές μεγαλύτερης της μονάδας, δηλαδή να ξεπερνούν το μέγιστο των δεδομένων εισόδου. Οι είσοδοι του δείγματος παίρνουν τιμές μεταξυ ενός και μηδέν (μηδέν η ελάχιστη είσοδος και ένα η μέγιστη). Για την πρόβλεψη παρέχουμε στο μοντέλο μας κανονικοποιημένες τιμές. Για την μετατροπή των τιμών χρησιμοποιήθηκε η συνάρτηση MinMaxScaler της βιβλιοθήκης scikit learn.
Για να επικοινωνήσει ο FrontEnd πελάτης με την υπηρεσία μας, στέλνει HTTP αιτήματα σε κάποιες συγκεκριμένες διευθύνσεις. Αναλόγως του πόρου που θέλει να αιτηθεί ή της λειτουργίας που θέλει να επιτελέσει στέλνει τον κατάλληλο τύπο αιτήματος (GET ή POST στην προκείμενη περίπτωση) στην κατάλληλη διεύθυνση. Οι διευθύνσεις αποτελούνται από δύο μέρη, την διεύθυνση βάσης (base URL) και το τελικό σημείο (endpoint). Με την ένωση των δύο (πρώτα η βάση) συμπληρώνεται η διεύθυνση. Στα πλαίσια της παρούσας εργασίας η διεύθυνση βάσης μας είναι η "http://localhost:8000", ενώ έχουμε δημιουργήσει τα παρακάτω endpoints:

- "/completed/", στο οποίο στέλνουμε HTTP GET και HTTP POST αιτήματα. Με τα πρώτα λαμβάνουμε όλες τις ολοκληρωμένες προβλέψεις. Με τα δεύτερα στέλνουμε στο σώμα του αιτήματος τα δεδομένα που περιγράφουν τις παραμέτρους για την πραγματοποίηση μίας ολοκληρωμένης πρόβλεψης, την οποία αφού υλοποιήσουμε μέσω της συνάρτησης make_predictions του αρχείου helpers.py αποθηκεύουμε στην βάση μας.

- "/completed-names/", όπου μέσω HTTP GET αιτήματων λαμβάνουμε όλα τα ονόματα που έχουμε δώσει για project προβλέψεων, προκειμένου να ελέγξουμε πως το project που πρόκειται να δημιουργήσουμε θα έχει μοναδικό όνομα.

- "/completed-num/", το οποίο δέχεται HTTP GET αιτήματα και επιστρέφει απλώς το πλήθος των ολοκληρωμένων προβλέψεων. Το παραπάνω χρησιμεύει στην ενημέρωση την κεφαλίδας της εφαρμογής μας η οποία περιέχει σχετικό μετρητή.

- "/data/". Μέσω αυτού λαμβάνουμε, ύστερα από HTTP GET αιτήμα, όσα από τα δεδομένα μας επιθυμούμε. Επειδή, η MongoDB αποθηκεύει τα δεδομένα χωρίς να κάνει τροποποιήση για την ζώνη ώρας, πραγματοποιούμε εμείς την απαραίτητη μετατροπή.

- ”/completed-nn/”. Αντίστοιχα με την πρώτη περίπτωση, δημιουργήσαμε το συγκεκριμένο endpoint για να λαμβάνουμε HTTP POST αιτήματα σχετικά με προβλέψεις με νευρωνικά δίκτυα. Η εφαρμογή πελάτης στέλνει ως σώμα των αιτημάτων τις παραμέτρους για την πρόβλεψη (π.χ. πλήθος εποχών εκπαίδευσης) και η υπηρεσία μας, αφού την πραγματοποιήσει μέσω του κώδικα της συνάρτησης nn_predictions από το αρχείο neural.py, επικοινωνεί με την βάση δεδομένων για την αποθήκευσή της.

Το RESTful API υλοποιήθηκε μέσω τριών αρχείων. Στο main.py έχουμε τον κώδικα για την δημιουργία των πέντε endpoints. Για τον χειρισμό των αιτημάτων χρησιμοποιούμε συναρτήσεις που εισάγουμε από το αρχείο database.py, στις οποίες γίνεται η απαραίτητη αλληλεπίδραση με την βάση δεδομένων μας. Σημειώνεται πως η σύνδεση, η λήψη και η αποθήκευση δεδομένων στις συλλογές της βάσης δεδομένων μας πραγματοποιείται μέσω της βιβλιοθήκης Motor η οποία είναι συμβατή με την ασύγχρονη μέθοδο της υπηρεσίας μας. Στο main.py, για να το δεύτερο endpoint γράφουμε:

```
@app.get('/completed-names/')
async def get_completed_names():
    response =  await fetch_completed_names()
    return response
```

Σε περίπτωση που σταλεί HTTP αίτημα εκτελείται ο κώδικας της συνάρτησης get_completed_names. Εντός της συνάρτησης αυτής, γίνεται μία κλήση στην συνάρτηση fetch_completed_names, την οποία εισάγουμε από το αρχείο database.py. Ο κώδικας της είναι ο ακόλουθος:

```
async def fetch_completed_names():
    rslt = []
    cursor =  CompletedForecasts.find({},{'project_name':1})
    async for document in cursor:
        rslt.append(Name(**document))
    return rslt
```

Στην γραμμή 2 αρχικοποιούμε τον πίνακα, όπου θα αποθηκεύσουμε το αποτέλεσμα του ερωτήματος προς την βάση. Στην αμέσως επόμενη γραμμή, κάνουμε το ερώτημα για να λάβουμε τα ονόματα των ολοκληρωμένων προβλέψεων. Για να το πετύχουμε αυτό, στο ερώτημα μας κάνουμε προβολή του συγκεκριμένου πεδίου (project_name). Στην συνέχεια, στι γραμμές 4-5, γεμίζουμε τον πίνακα rslt με τα αποτελέσματα του ερωτήματος. Όπου Name είναι το μοντέλο που αντιστοιχεί στο αποτέλεσμα. Τέλος, στην γραμμή 6 επιστρέφουμε τον πίνακα με το αποτέλεσμα του ερωτήματος.
Στο αρχείο model.py έχουμε γράψει μία σειρά από κλάσεις, οι οποίες ορίζουν την δομή των δεδομένων που αποθηκεύουμε και λαμβάνουμε από τις συλλογές της βάσης δεδομένων μας. Κάθε κλάση αποτελείται από ζεύγη κλειδιού τιμής, με το κλειδί να αντιπροσωπεύει το όνομα του πεδίου και την τιμή τον τύπο των δεδομένων που αποθηκεύουμε. Σημαντικό μέρος της δομής περιγράφεται από αναδρομικά μοντέλα, δηλαδή μοντέλα που ορίζονται χρησιμοποιώντας άλλα μοντέλα. Έχουμε δημιουργήσει μοντέλα για τα δεδομένα των δύο συλλογών καθώς και μοντέλα για τα δεδομένα που λαμβάνει το API από τον Front End πελάτη.
Παρακάτω παρουσιάζουμε τέσσερα από τα μοντέλα που ορίσαμε:

```
class DTParams(BaseModel):
    criterion: str
    max_depth: str
    min_samples_split: int
    min_samples_leaf: int
    ccp_alpha: float
    technique: str
```

Για κάθε έναν από τους αλγορίθμους, αλλά και για το νευρωνικό δίκτυο ορίζουμε ένα μοντέλο με τις παραμέτρους τους και την μέθοδο που ακολουθήθηκε για την παραγωγή των ζητούμενων προβλέψεων.

```
class PerformanceModel(BaseModel):
    pwrA: float
    pwrB: float
    pwrC: float
```

Έχουμε ορίσει ένα μοντέλο που περιέχει τρεις αριθμούς κινητής υποδιαστολής για την αποθήκευση της απόδοσης των αλγορίθμων στις τρεις φάσεις.

```
class DTInfo(BaseModel):
    name:str
    used:bool
    performance: PerformanceModel
    params_A: DTParams
    params_B: DTParams
    params_C: DTParams
```

Για κάθε αλγόριθμο (και για τα νευρωνικά δίκτυα) δημιουργούμε ένα μοντέλο με πληροφορίες για αυτό. Ειδικότερα, αποθηκεύουμε τις βέλτιστες παραμέτρους για κάθε φάση που θέλουμε να προβλέψουμε, μια Boolean τιμή που δείχνει εάν χρησιμοποιήθηκε ο αλγόριθμος ή όχι, την απόδοση του σε κάθε φάση και το πλήρες όνομα του αλγορίθμου. Παρατηρούμε ότι έχουμε αξιοποιήσει τα μοντέλα που ορίσαμε παραπάνω ως πεδία του μοντέλου αυτού.

```
class ScoresModel(BaseModel):
    nn: NNInfo
    lr: LRInfo
    rf: RFInfo
    svr: SVRInfo
    grb: GRBInfo
    knn: KNNInfo
    dt: DTInfo
```

Στην παραπάνω κλάση απλώς συγκεντρώνουμε τις πληροφορίες από όλους τους αλγορίθμους και το νευρωνικό δίκτυο.

```
class CompletedForecast(BaseModel):
    time: str
    project_name: str
    horizon: int
    window: int
    predictions_A: list
    predictions_B: list
    predictions_C: list
    algorithms: ScoresModel
```

Τέλος, δημιουργούμε το μοντέλο για τα δεδομένα που θα εισάγουμε στην συλλογή Completed_Forecasts της βάσης μας. Όπως βλέπουμε εκτός από τις πληροφορίες για τους αλγορίθμους και το νευρωνικό δίκτυο, έχουμε και δεδομένα που αφορούν την ίδια την πρόβλεψη, όπως το όνομα, ο ορίζοντας και οι λίστες με τις προβλέψεις που έκανε ο αλγόριθμος με την καλύτερη απόδοση σε κάθε φάση.

```
class Data(BaseModel):
    Time: datetime
    pwrA: float
    pwrB: float
    pwrC: float
```

Για την δεύτερη συλλογή της βάσης μας, την Timeseries, έχουμε το παραπάνω μοντέλο το οποίο χρησιμοποιούμε για να λάβουμε δεδομένα.

Οι λειτουργίες της εύρεσης του βέλτιστου συνδυασμού υπερπαραμέτρων είναι ιδιαιτέρως κοστοβόρες σε υπολογιστικό επίπεδο, με συνέπεια να απαιτούν πολύ χρόνο για την ολοκλήρωσή τους. Για την επιτάχυνση της αναζήτησης πλέγματος και της τυχαίας αναζήτησης δημιουργήσαμε ένα Cluster με τρεις υπολογιστές εντός ενός τοπικού δικτύου. Για την δημιουργία της διεργασίας που θα λειτουργήσει ως προγραμματιστής (Scheduler), τρέχουμε στη γραμμή εντολών την παρακάτω εντολή:

```
dask-scheduler
```

Για την δημιουργία της διεργασίας που θα λειτουργήσει ως εργάτης (Worker), πρέπει να παρέχουμε την διεύθυνση του προγραμματιστή και να ορίσουμε το πλήθος των νημάτων που θα του παρέχουμε μέσω της παραμέτρου nthreads. Ειδικότερα, τρέχουμε στη γραμμή εντολών την παρακάτω εντολή:

```
dask-worker tcp://<scheduler address> --nthreads t
```

Σημειώνεται, πως μέσω της παραμέτρου nprocs μπορούμε να εκκινήσουμε πολλαπλές διεργασίες εργάτη, με τόσα threads η κάθε μία όσα η τιμή της παραμέτρου nthreads. Η προκαθορισμένη τιμή της παραμέτρου είναι η μονάδα.
Για να δημιουργήσουμε στον κώδικά μας έναν πελάτη που να χρησιμοποιεί το Cluster, αφού εισάγουμε την συνάρτηση Client από την βιβλιοθήκη dask.distributed, γράφουμε:

```
client = Client("tcp://<scheduler address>")
```

Η ανάθεση της εκτέλεσης της αναζήτησης πλέγματος και της τυχαίας αναζήτησης από το Cluster γίνεται μέσω των αντίστοιχων συναρτήσεων της βιβλιοθήκης dask_ml.model_selection.

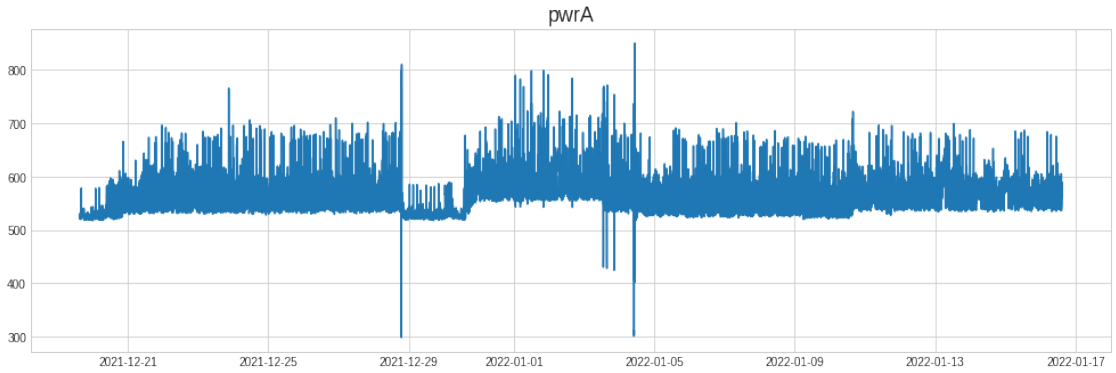
Σχήμα 6.13: Το ταμπλό ελέγχου του Dask

Κεφάλαιο 7

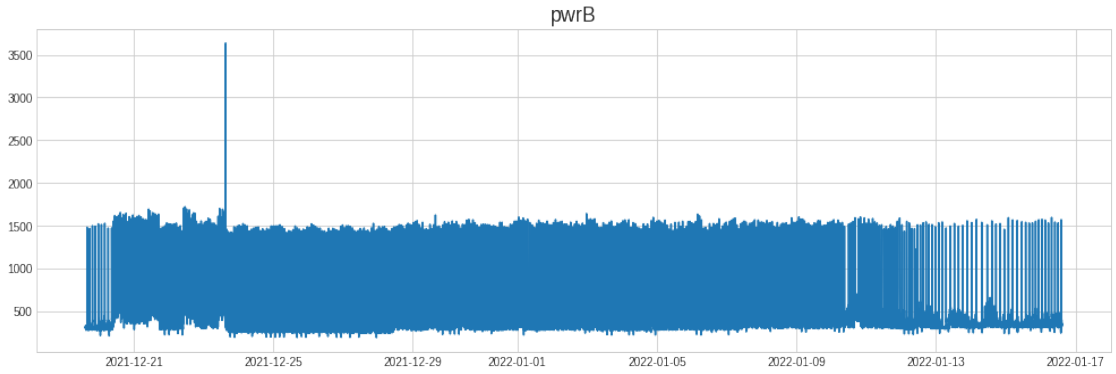
Πειραματικό κομμάτι - Ανάλυση και Πρόβλεψη Χρονοσειρών

7.1 Ανάλυση και Επεξεργασία Δεδομένων

Στο κεφάλαιο αυτό θα αναλύσουμε τις τρεις χρονοσειρές μας και ύστερα θα κάνουμε προβλέψεις επί αυτών. Για τις προβλέψεις θα χρησιμοποιήσουμε μεθόδους που αναλύσαμε στα προηγούμενα κεφάλαια. Πιο συγκεκριμένα, οι προβλέψεις θα γίνουν μέσω των έξι μεθόδων μηχανικής μάθησης, μέσω νευρωνικών δικτύων, του αλγόριθμου N-Beats, της απλοϊκής μεθόδου, της απλής μεθόδου εξομάλυνσης και των μοντέλων ARIMA.
Ξεκινώντας, θα δούμε τα δεδομένα μας όπως τα λαμβάνουμε μέσω του κώδικα στην σελίδα 33 του κεφαλαίου 3. Τονίζεται πως αρχικά τα δεδομένα μας είναι σε μορφή μηδενικής μετατόπισης ζώνης ώρας, άρα απαιτείται μετατροπή για να έρθουν στην ζώνη ώρας της Ελλάδας. Οι τιμές είναι σε Watt καθώς αφορούν ενεργό ισχύ.

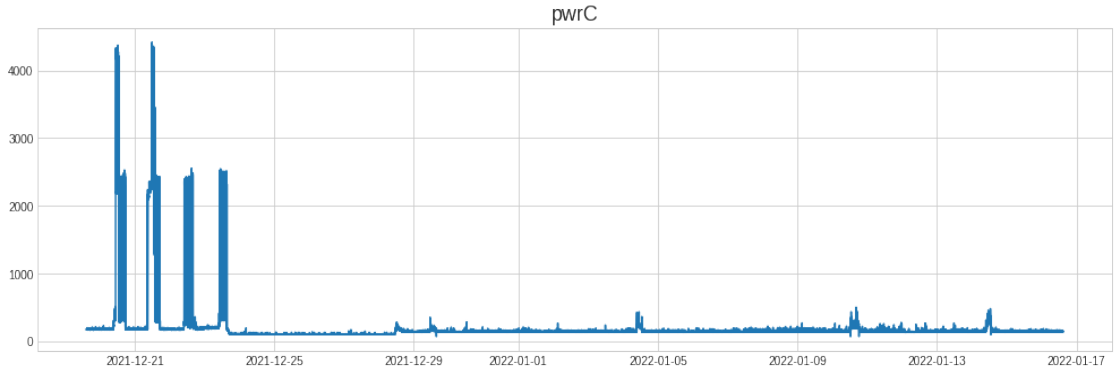


Σχήμα 7.1: Οι μετρήσεις για την φάση A



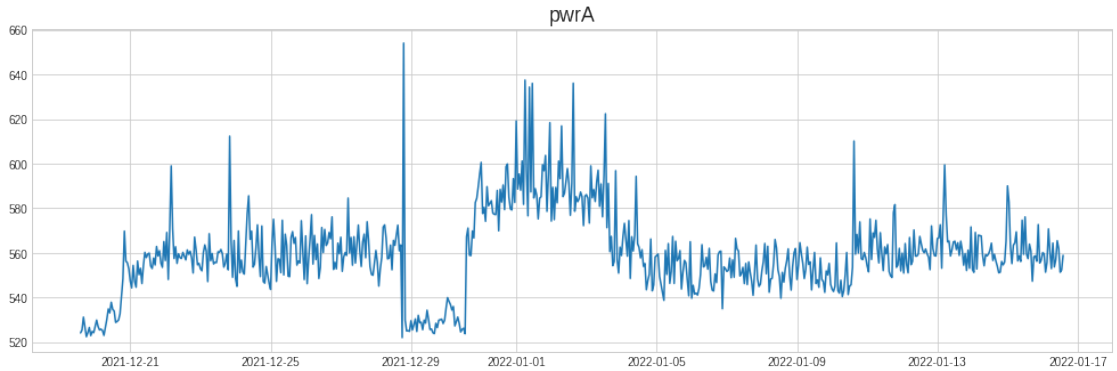
Σχήμα 7.2: Οι μετρήσεις για την φάση B

Όπως φαίνεται και από τα σχήματα έχουμε πολύ πυκνά γραφήματα με κάποιες ακραίες τιμές και έντονες διακυμάνσεις. Για ακρίβεια, έχουμε 225664 παρατηρήσεις για κάθε φάση. Ο

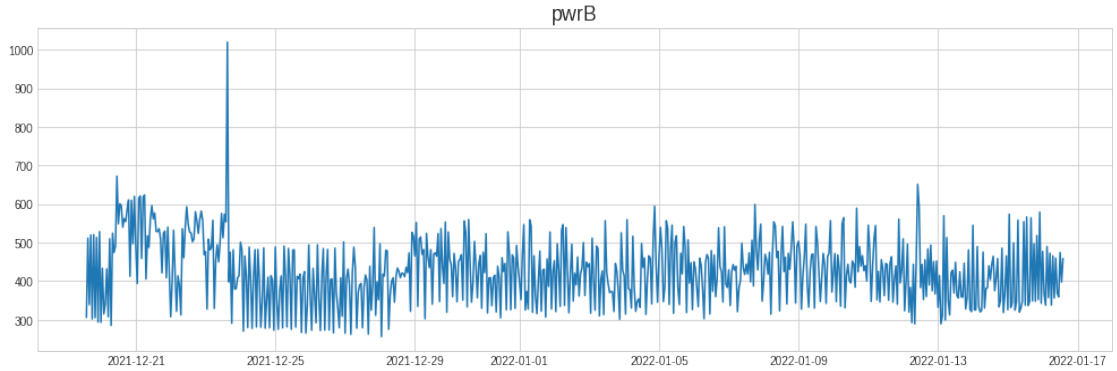


Σχήμα 7.3: Οι μετρήσεις για την φάση C

ακριβής χρόνος της πρώτης μέτρησης είναι 2021-12-19T12:48:03.379000Z (βλέπουμε και την μηδενική μετατόπιση για την ζώνη ώρας) και της τελευταίας 2022-01-16T12:46:50.303000Z. Τα δεδομένα μας έχουν πολύ υψηλή συχνότητα (χιλιοστά του δευτερολέπτου) ενώ δεν απέχουν ίσες χρονικές αποστάσεις μεταξύ τους. Το τελευταίο φαίνεται και αν μετρήσουμε απλώς τις παρατηρήσεις που αφορούν δύο μέρες. Για την 2021-12-22 έχουμε 16905 τιμές, ενώ για την 2021-12-21 19522.
Για να φτιάξουμε τις χρονοσειρές μας, θα λάβουμε τον μέσο όρο από τις μετρήσεις για κάθε ώρα. Με τον τρόπο αυτό προκύπτουν οι τρεις ωριαίες χρονοσειρές μας:

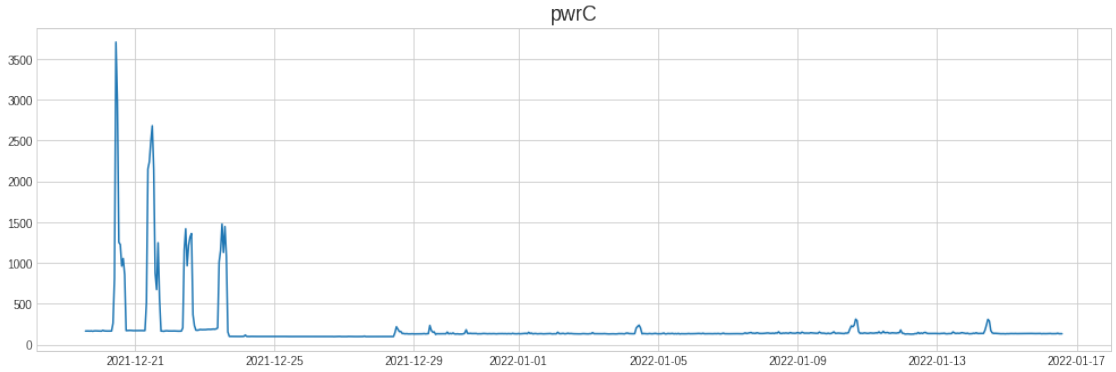


Σχήμα 7.4: Η χρονοσειρά της φάσης A



Σχήμα 7.5: Η χρονοσειρά της φάσης B

Πριν επεξεργαστούμε τα δεδομένα μας, θα παρουσιάσουμε κάποια βασικά στατιστικά στοιχεία για τις τρεις χρονοσειρές. Εμφανίζουμε το πλήθος των στοιχείων, την μέση



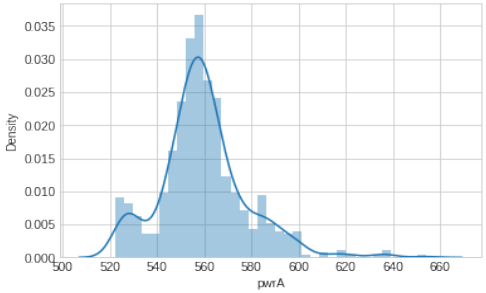
Σχήμα 7.6: Η χρονοσειρά της φάσης C

τιμή τους, την τυπική απόκλιση, το ελάχιστο, το μέγιστο και την τιμή που αντιστοιχεί στο 25%, 50% και 75% των δεδομένων. Για την παραγωγή αυτών στατιστικών στοιχείων έχουμε χρησιμοποιήσει την μέθοδο describe της βιβλιοθήκης pandas. Όπως βλέπουμε από τον παρακάτω πίνακα, πλέον έχουμε 673 παρατηρήσεις. Η χρονοσειρά A έχει πολύ μικρή διακύμανση ενώ η χρονοσειρά που αντιστοιχεί στην φάση C έχει ιδιαίτερα μεγάλη. Επίσης, βλέπουμε για τις χρονοσειρές B,C πως τα μέγιστα τους απέχουν πολύ μεγάλη απόσταση από την μέση τιμή των δεδομένων κάτι το οποίο φαίνεται και από τις γραφικές. Τέτοια σημεία επιδρούν αρνητικά στο κομμάτι των προβλέψεων και θα αφαιρεθούν.

	pwrA	pwrB	pwrC
count	673.000000	673.000000	673.000000
mean	559.573510	427.163527	196.502991
std	18.538336	86.619908	312.223985
min	522.013589	256.618158	99.257083
25%	550.241851	352.190909	133.269919
50%	558.172702	426.105533	136.343296
75%	566.998713	484.604741	143.561482
max	654.034702	1019.316855	3705.773844

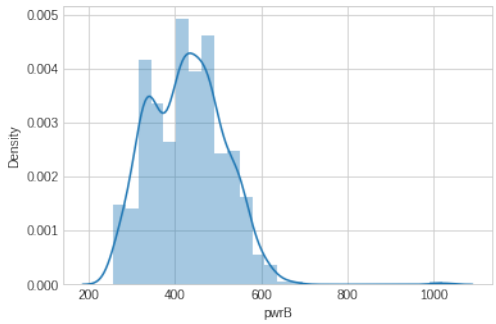
Σχήμα 7.7: Βασικά στατιστικά για τις ΧΣ

Ακολούθως, θα δούμε τα διαγράμματα κατανομής των τιμών για τις χρονοσειρές μας:

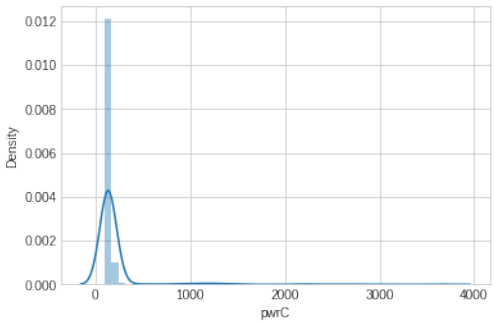


Σχήμα 7.8: Κατανομή τιμών για την φάση A

Παρατηρούμε πως τα δεδομένα των χρονοσειρών έχουν κατανομές που προσεγγίζουν την κανονική (κωνοειδείς μορφή με συμμετρία), με διαφορετικές μέσες τιμές και αποκλίσεις. Για



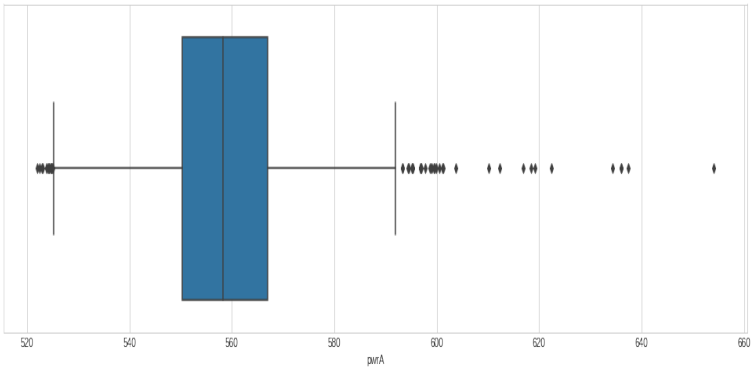
(a) Κατανομή τιμών για την φάση B



(b) Κατανομή τιμών για την φάση C

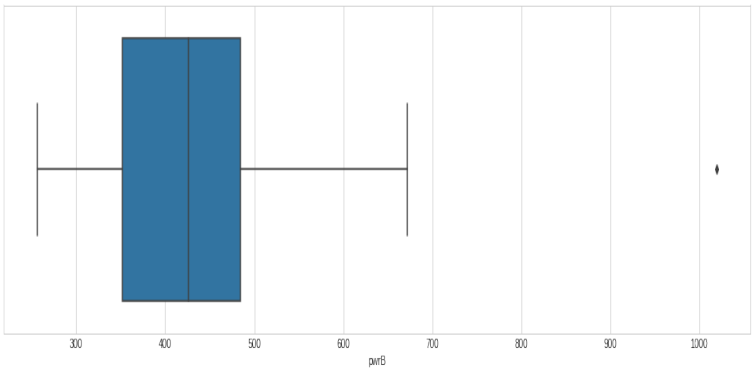
Σχήμα 7.9: Κατανομή τιμών για τις φάσεις B,C

την τρίτη χρονοσειρά βλέπουμε πως η συντριπτική πλειοψηφία των τιμών βρίσκεται κοντά στα 150 Watt, ενώ για τις άλλες δύο τα δεδομένα απλώνονται σε μεγαλύτερο εύρος τιμών. Για την δεύτερη χρονοσειρά βλέπουμε δύο κορυφές οι οποίες είναι κοντινές σε τιμές εν αντιθέσει με την πρώτη χρονοσειρά που οι κορυφές έχουν μεγαλύτερη διαφορά. Η πρώτη χρονοσειρά έχει "ουρές" που ελαττώνονται πιό ομαλά σε σχέση με τις αντίστοιχες των υπολοίπων, γεγονώς που εξηγείται από την μικρότερη απόκλιση που την χαρακτηρίζει.
Η χρονοσειρές μας δεν έχουν καμία μηδενική τιμή, αφού τα αρχικά μας δεδομένα περιείχαν μετρήσεις για όλες τις ώρες που φαίνονται στις χρονοσειρές. Για την εξάλειψη των ακραίων τιμών, θα χρησιμοποιήσουμε τα θηκόγράμματα (boxplots). Τα θηκογράμματα περιλαμβάνουν την διάμεσο (Q2), την τιμή που'ναι στο μέσο μεταξύ του ελαχίστου και της διαμέσου (Q1), την τιμή που'ναι στο μέσο μεταξύ του μεγίστου και της διαμέσου (Q3), μία τιμή που ορίζει το κάτω όριο του διαγράμματος και μία που ορίζει το άνω όριο. Για το κάτω όριο έχουμε την τιμή Q1 - 1.5(Q3-Q1) και για το άνω την τιμή Q3 + 1.5(Q3-Q1). Όσες τιμές είναι κάτω από το κάτω όριο και πάνω από το άνω όριο θεωρούνται ακραίες τιμές (outliers). Για την χρονοσειρά της φάσης A, έχουμε:



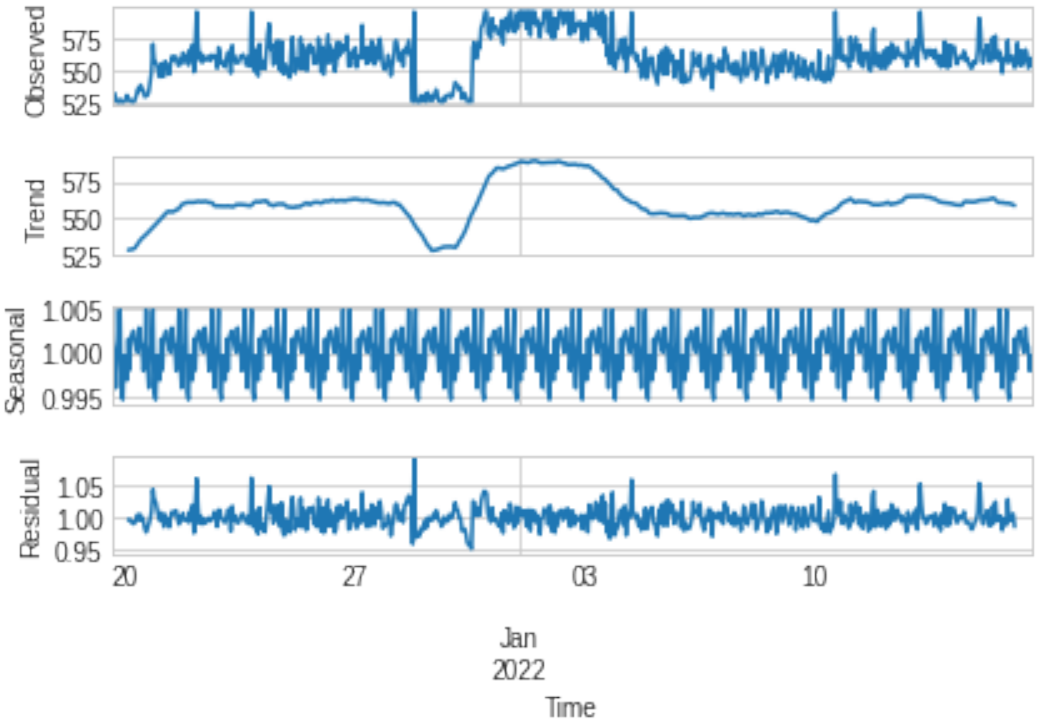
Σχήμα 7.10: Θηκόγραμμα για την φάση A

Βάσει του θηκογράμματος, θα αφαιρέσουμε τις τιμές που'ναι μεγαλύτερες του 595 και μικρότερες από 525, θέτοντάς τες ίσες με 595 και 525. Πρόκειται για το 6.3% των παρατηρήσεων. Τονίζεται πως οι αλλαγές που έγιναν και θα γίνουν στα δεδομένα εδώ εφαρμόστηκαν αυτούσιες στο κομμάτι της προεπεξεργασίας για την εφαρμογή του προηγούμενου κεφαλαίου. Για την χρονοσειρά της φάσης B έχουμε το παρακάτω θηκόγραμμα:
Είναι εύκολο να δει κανείς πως για την φάση B έχουμε σημαντικά λιγότερες ακραίες τιμές. Ακολουθώντας την ίδια λογική με πριν, αφαιρούμε τις τιμές που'ναι μεγαλύτερες του 680 και μικρότερες του 210, θέτωντας τες ίσες με 680 και 210.



Σχήμα 7.11: Θηκόγραμμα για την φάση B

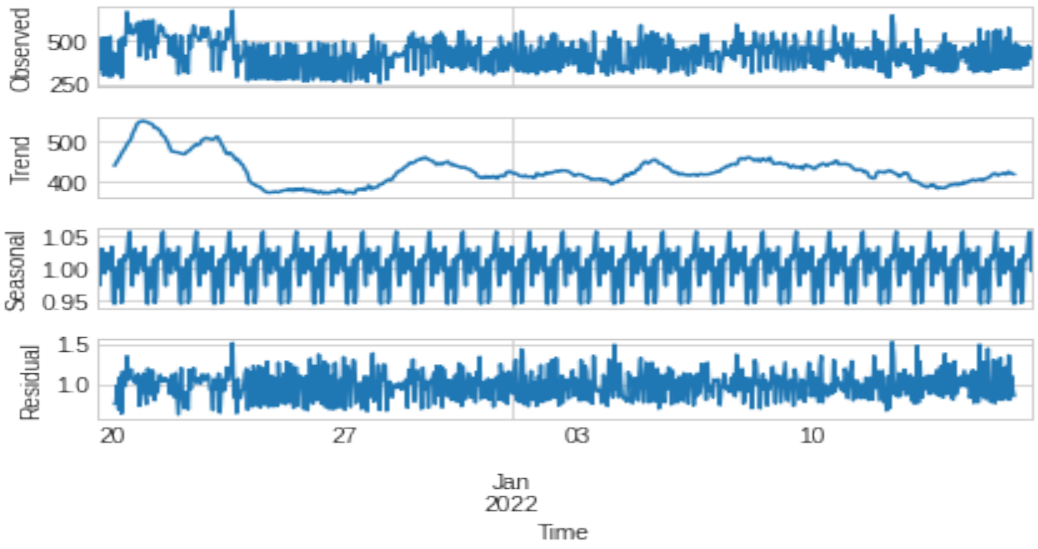
Για την φάση C, θα εφαρμόσουμε μία άλλη μέθοδο αφαίρεσης ακραίων τιμών καθότι η προηγούμενη οδηγούσε σε αφαίρεση υπερβολικά μεγάλου αριθμού σημείων κάτι που αλλοιώνει τον χαρακτήρα της χρονοσειράς και οδηγεί σε μειωμένη απόδοση των αλγορίθμων πρόβλεψης. Σ'αυτήν την περίπτωση, λοιπόν, θα αφαιρέσουμε απλώς το 8% των σημείων με την υψηλότερη τιμή και 0.001% αυτών με την χαμηλότερη τιμή.
Στην συνέχεια θα αποσυνθέσουμε τις χρονοσειρές μας για να δούμε τις συνιστώσες από τις οποίες αποτελούνται. Σε κάθε περίπτωση χρησιμοποιούμε το πολλαπλασιαστικό μοντέλο.



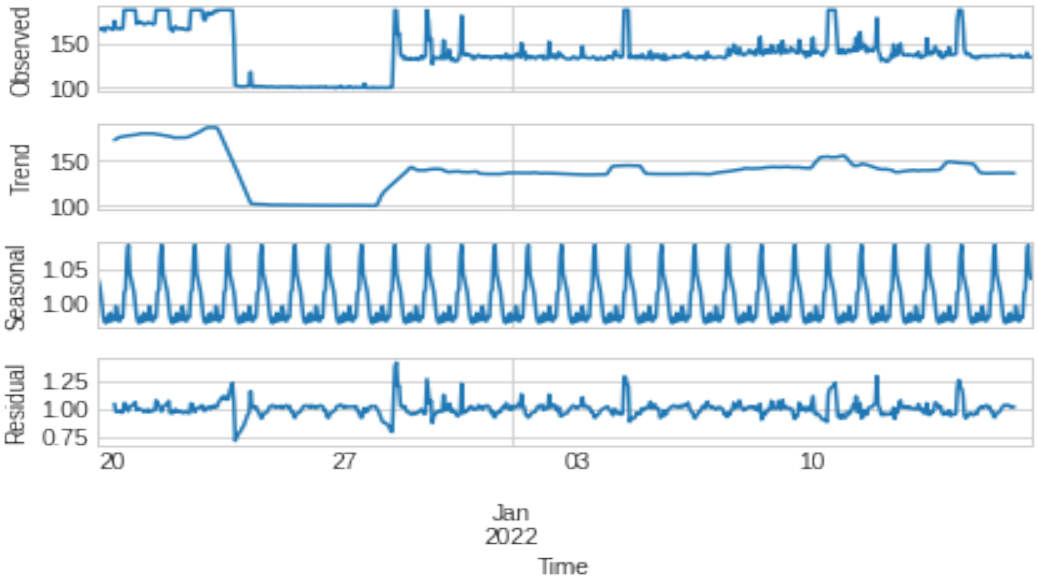
Σχήμα 7.12: Αποσύνθεση ΧΣ φάσης A

Το πρώτο γράφημα δείχνει την χρονοσειρά ύστερα από την αφαίρεση των ακραίων τιμών, το δεύτερο την συνιστώσα της τάσης, το τρίτο αυτήν της εποχιακότητας και το τέταρτο το κομμάτι της χρονοσειράς που δεν περιλαμβάνεται στις προηγούμενες συνιστώσες, δηλαδή το υπόλοιπο. Σημειώνεται, πως η αποσύνθεση έγινε μέσω της συνάρτησης seasonal_decompose απ'την βιβλιοθήκη statsmodels.tsa.seasonal.
Παρατηρούμε πως η χρονοσειρά δεν παρουσιάζει σταθερά ανοδική ή καθοδική τάση. Σε κάποια σημεία εμφανίζει αλλαγή επιπέδου, ωστόσο εν τέλει εμφανίζει μία σταθερότητα.

Σχετικά με την εποχιακή συνιστώσα, βλέπουμε πως δεν είναι αρκετά έντονη για να επηρεάσει σημαντικά την χρονοσειρά.
Η εφαρμογή της ίδιας μεθόδου αποσύνθεσης για την χρονοσειρά της φάσης B δίνει:



Σχήμα 7.13: Αποσύνθεση ΧΣ φάσης B



Σχήμα 7.14: Αποσύνθεση ΧΣ φάσης C

Σε αυτήν την περίπτωση, βλέπουμε από την συνιστώσα τάσης πως η χρονοσειρά έχει αρκετές αλλαγές επιπέδου, εμφανίζει εναλλαγές από θετική σε αρνητική τάση και αντίστροφα, χωρίς όμως να εμφανίζει κάποια έντονη συνολική τάση. Όσον αφορά την εποχιακότητα και εδώ βλέπουμε πιό έντονη εποχιακή συνιστώσα από πριν, ωστόσο όχι τόσο έντονη που να μπορούμε να χαρακτηρίσουμε την χρονοσειρά εποχιακή.
Αποσυνθέτοντας την τρίτη χρονοσειρά λαμβάνουμε τα αποτελέσματα της παρακάτω εικόνας. Και εδώ η σειρά χαρακτηρίζεται από έλλειψη τάσης αφού βλέπουμε σταθερότητα για το μεγαλύτερο μέρος της. Η εποχιακότητα εδώ είναι ακόμα πιό έντονη εδώ.
Σύνηθες είναι να εκτυπώνουμε τα διαγράμματα των συντελεστών αυτοσυσχέτισης και μερικής αυτοσυσχέτισης για να επιλέξουμε τους όρους για τα μοντέλα ARIMA σύμφωνα

με μία σειρά εμπειρικών κανόνων. Στα πλαίσια της παρούσας διπλωματικής, οι όροι των μοντέλων αυτών θα επιλεχθούν αυτόματα από την συνάρτηση που κατασκευάζει το μοντέλο, την συνάρτηση auto_arima, η οποία για τον σκοπό αυτό εξετάζει την τιμή του AIC για κάθε μοντέλο. Ως εκ τούτου θα παραλείψουμε τα προαναφερθέντα διαγράμματα. Τέλος, σημειώνεται πως τόσο το ADF τεστ όσο και το KPSS τεστ, έδειξαν πως και οι τρεις χρονοσειρές είναι στάσιμες.

7.2 Πρόβλεψεις και Αποτελεσμάτα

Στα πλαίσια της παρούσας εργασίας τρέξαμε μία σειρά από πειράματα προβλέψεων. Από τις χρονοσειρές κάθε φορά κρατήσαμε για προς πρόβλεψη τις τελευταίες τιμές των χρονοσειρών. Το πλήθος των παρατηρήσεων που κρατήσαμε είναι ίσο με τον ορίζοντα πρόβλεψης. Πραγματοποιήσαμε προβλέψεις τόσο με στατιστικές μεθόδους όσο και με μεθόδους μηχανικής μάθησης και νευρωνικά δίκτυα. Επίσης κάναμε προβλέψεις και με τον αλγόριθμο N-Beats, χρησιμοποιώντας την βιβλιοθήκη KerasBeats, που περιλαμβάνει μια υλοποίηση του αλγορίθμου. Οι προβλέψεις μας αφορούν τους ορίζοντες 6 και 24 ωρών. Για τις περιπτώσεις εκτός των στατιστικών μοντέλων λάβαμε διπλάσιες εισόδους από τον εκάστοτε ορίζοντα, ενώ όπου χρειάστηκε οι υπερπαράμετροι βελτιστοποιήθηκαν με τυχαία αναζήτηση 3000 επαναλήψεων επί των παραμέτρων που αναλύσαμε στο προηγούμενο κεφάλαιο. Η επιλογή των υπερπαραμέτρων και η εκπαίδευση των μοντέλων έγινε ακολουθώντας την ανάλυση των προηγούμενων κεφαλαίων. Σημειώνεται πως για την παραγωγή των προβλέψεων χρησιμοποιήσαμε την εφαρμογή που δημιουργήσαμε και κατ'επέκτασιν το Cluster υπολογιστών για την επιτάχυνσή τους.
Στους παρακάτω πίνακες παρουσιάζουμε την απόδοση των μοντέλων για όλες τις περιπτώσεις (Οι τιμές εδώ και σε κάθε άλλη περίπτωση αφορούν ποσοστά επι τοις εκατό):

ΦΑΣΗ	Οριζ.	LR	RF	GRB	SVR	DT	KNN
A	6	0.7586	0.8627	0.7819	0.7181	0.8666	1.1949
	24	1.1705	1.5179	1.7292	0.9724	1.7349	0.9077
B	6	**9.7399**	11.0312	11.5	10.2319	14.7747	14.1493
	24	14.9582	45.5031	31.9144	15.347	15.627	21.344
C	6	0.9129	1.3021	0.9053	0.8133	0.811	2.1657
	24	1.5991	5.0683	1.0735	0.9847	0.7547	8.0304

Πίνακας 7.1: Αποτελέσματα με στρατηγική multistep

ΦΑΣΗ	Οριζ.	LR	RF	GRB	SVR	DT	KNN
A	6	0.7182	1.0931	0.96	0.7283	0.8147	**0.6935**
	24	1.1845	0.877	2.1236	0.8666	**0.8162**	0.9
B	6	10.8996	11.7852	12.1362	11.6183	12.7828	13.7551
	24	15.9051	16.0885	16.5581	**13.9396**	14.4961	17.2771
C	6	0.9172	**0.7403**	0.7512	0.8016	0.8854	1.5339
	24	1.7948	1.7434	0.9611	0.79	1.1057	4.0797

Πίνακας 7.2: Αποτελέσματα με στρατηγική multimodel

ΦΑΣΗ	Οριζ.	LR	RF	GRB	SVR	DT	KNN
A	6	0.7123	0.9104	0.7962	0.7291	0.8609	1.1203
	24	1.1927	1.0632	0.8795	1.0066	0.9378	1.1936
B	6	10.911	12.1044	10.2913	11.1458	16.1386	14.9279
	24	15.852	41.389	41.0287	14.8045	29.9123	19.3884
C	6	0.9172	0.9636	0.9642	0.9577	1.1401	2.3151
	24	1.783	4.3025	2.3028	**0.7337**	1.7217	6.0485

Πίνακας 7.3: Αποτελέσματα με στρατηγική chained-multimodel

Για να δούμε πως απόδίδουν τα στατιστικά μοντέλα, συγκριτικά με τις μεθόδους μηχανικής μάθησης, επιλέξαμε δύο από τα δημοφιλέστερα, την απλή εκθετική εξομάλυνση (SES) και τα μοντέλα ARIMA. Όσον αφορά την SES τρέχουμε μια γραμμική αναζήτηση, σύμφωνα με όσα περιγράψαμε στο αντίστοιχο κομμάτι της θεωρίας, για να βρούμε την κατάλληλη τιμή της παραμέτρου a, ενώ ως αρχικό επίπεδο θεωρούμε τον μέσο όρο των τριών πρώτων παρατηρήσεων. Για τα μοντέλα ARIMA οι κατάλληλοι όροι προκύπτουν αυτόματα μέσω της συνάρτησης auto_arima της βιβλιοθήκης pmdarima, μέσω της οποίας κάνουμε και τις προβλέψεις, ενώ για την SES έγινε χρήση της συνάρτηση SimpleExpSmoothing της βιβλιοθήκης statsmodels. Για τον αλγόριθμο N-Beats επιλέξαμε την γενική διαμόρφωση της αρχιτεκτονικής του με τριάντα στοίβες των τεσσάρων στρωμάτων και για παραμέτρους εκπαίδευσης τις 100 εποχές, μέγεθος δέσμης δειγμάτων 128 και ρυθμό εκπαίδευσης 0.001. Τα αποτελέσματα για τις δύο μεθόδους και τον αλγόριθμο N-Beats είναι τα εξής:

ΦΑΣΗ	Οριζ.	SES	ARIMA	N-Beats
A	6	**0.7673**	0.8004	0.7943
	24	**1.0784**	1.1486	1.7850
B	6	10.6093	**8.2070**	10.6513
	24	15.9222	**15.5093**	16.588
C	6	**0.82**	0.8910	1.0004
	24	0.7547	**0.7284**	2.6784

Πίνακας 7.4: Αποτελέσματα για στατιστικά μοντέλα και N-Beats

Όσον αφορά το κομμάτι των νευρωνικών δικτύων, είναι σαφές πως επιλέγοντας τιμές για όλες τις παραμέτρους, που δίνουμε την δυνατότητα στον χρήστη να ορίσει, μπορούμε να φτιάξουμε χιλιάδες διαφορετικές αρχιτεκτονικές δικτύων και να τις εκπαιδεύσουμε επίσης με χιλιάδες διαφορετικούς τρόπους. Στην παρούσα εργασία θα εξετάσουμε την απόδοση δέκα αρχιτεκτονικών για κάθεμια από τις τρείς φάσεις. Αναλυτικότερα, έχουμε τις εξής αρχιτεκτονικές, μαζί με τις παραμέτρους εκπαίδευσης και τις ακολουθούμενες στρατηγικής για την παραγωγή των προβλέψεων:

a) Κανένα κρυφό επίπεδο, 100 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.001 και μέγεθος δέσμης δειγμάτων 32 και στρατηγική multioutput.

b) Κανένα κρυφό επίπεδο, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.0025 και μέγεθος δέσμης δειγμάτων 32 και στρατηγική multioutput.

c) Ένα κρυφό επίπεδο των 64 νευρώνων, με συνάρτηση ενεργοποίησης ReLU, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.001 και μέγεθος δέσμης δειγμάτων 64 και στρατηγική multioutput.

d) Ένα κρυφό επίπεδο των 64 νευρώνων, με συνάρτηση ενεργοποίησης ReLU, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.001, μέγεθος δέσμης δειγμάτων 64 και στρατηγική multistep.

e) Ένα κρυφό επίπεδο των 64 νευρώνων, με συνάρτηση ενεργοποίησης ELU, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.0015, μέγεθος δέσμης δειγμάτων 64 και στρατηγική multioutput.

f) Ένα κρυφό επίπεδο των 32 νευρώνων, με συνάρτηση ενεργοποίησης ELU, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.0015, μέγεθος δέσμης δειγμάτων 32 και στρατηγική multistep.

g) Δύο κρυφά επίπεδα των 32 νευρώνων, με συναρτήσεις ενεργοποίησης ReLU και στα δύο, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.002 και μέγεθος δέσμης δειγμάτων 16 και στρατηγική multioutput.

h) Δύο κρυφά επίπεδα των 32 νευρώνων, με συναρτήσεις ενεργοποίησης ReLU και ELU, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.005 και μέγεθος δέσμης δειγμάτων 16 και στρατηγική multioutput.

i) Κανένα κρυφό επίπεδο, 100 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.001 και μέγεθος δέσμης δειγμάτων 64 και στρατηγική multimodel.

j) Κανένα κρυφό επίπεδο, 150 εποχές εκπαίδευσης, ρυθμός εκπαίδευσης 0.005, μέγεθος δέσμης δειγμάτων 16 και στρατηγική multioutput.

Τα αποτελέσματα που λαμβάνουμε για τα νευρωνικά δίκτυα είναι τα παρακάτω:

	Ορίζοντας 6			Ορίζοντας 24		
Μοντέλο	A	B	C	A	B	C
a	1.6703	**14.9874**	1.1578	2.2148	22.0773	6.493
b	2.4255	18.271	1.0176	1.9139	18.3627	4.4669
c	1.238	19.6075	1.0761	1.3417	19.0965	**1.9962**
d	1.3148	16.828	0.9991	**1.3245**	19.562	3.501
e	1.3936	18.4141	1.1312	1.8122	19.6938	3.4285
f	1.2267	16.9361	1.0177	1.6437	37.869	5.2113
g	1.1149	18.263	1.166	1.3936	19.0471	2.2488
h	**1.1107**	16.2585	1.1385	1.3488	20.9868	2.1193
i	1.7452	17.1641	**0.7864**	2.4416	25.5302	6.3672
j	1.5723	18.533	1.1225	2.8993	**17.6956**	11.0014

Πίνακας 7.5: Αποτελέσματα μοντέλων νευρωνικών δικτύων

7.3 Συμπεράσματα

Θα ξεκινήσουμε σχολιάζοντας τον χρόνο ολοκλήρωσης των προβλέψεων. Είναι σαφές πως με την χρήση του Cluster των υπολογιστών επιταχύνθηκε σημαντικά η διαδικασία (βελτίωση περίπου 90% σε σχέση με εκτέλεση σε έναν υπολογιστή). Κάθε αλγόριθμος απαιτεί διαφορετικό χρόνο για την διαδικασία βελτιστοποίησης, εκπαίδευσης και πρόβλεψης. Επί της ουσίας μόνο η πρώτη διαδικασία είναι χρονοβόρα και ως εκ τούτου εκεί βλέπουμε διαφοροποίηση μεταξύ των αλγορίθμων. Μπορούμε να πούμε πως στην στρατηγική multistep η βελτιστοποίηση των υπερπαραμέτρων ολοκληρώνεται πολύ γρηγορότερα από τις δύο άλλες στρατηγικές, κάτι αναμενόμενο γιατί εκεί έχουμε να κάνουμε με ένα μοντέλο, ενώ στις άλλες έχουμε τόσα μοντέλα όσα και ο ορίζοντας πρόβλεψης. Από τους αλγόριθμους, λιγότερο χρόνο απαιτεί η γραμμική παλινδόμηση καθότι έχει μόλις δύο συνδυασμούς υπερπαραμέτρων προς εξέταση. Ιδιαίτερα γρήγορη είναι και η διαδικασία για τους αλγορίθμους κ-κοντινότεροι γείτονες, δένδρο απόφασης και μηχανές διανυσμάτων υποστήριξης χωρίς να έχουμε σημαντικές διαφορές μεταξύ τους. Από την άλλη, οι αλγόριθμοι του τυχαίου δάσους και ενίσχυσης κλίσης παρουσιάζουν μεγαλύτερους χρόνους. Αυτό οφείλεται στην φύση τους, καθώς χτίζονται μέσω πολλών μικρότερων μοντέλων (εκτιμητών) τα οποία συνδυάζονται για το τελικό αποτέλεσμα.
Σχετικά με την απόδοση των αλγορίθμων μηχανικής μάθησης (πίνακες 7.1-7.3) μπορούμε να σημειώσουμε τα παρακάτω:

- Σε όλες τις φάσεις, για κάθε στρατηγική και σε όλους τους αλγορίθμους η απόδοση που έχουμε για πρόβλεψη ορίζοντα 6 ωρών είναι καλύτερη (μικρότερο MAPE) από την περίπτωση του ορίζοντα 24 ωρών, με ελάχιστες εξαιρέσεις. Αυτό ήταν αναμενόμενο, καθώς όσο μεγαλώνει ο ορίζοντας σε μία πρόβλεψη τόσο αυξάνεται η αβεβαιότητα και συνεπώς είναι πιο δύσκολο να κάνουμε πρόβλεψη. Παρότι στις τέσσερις από τις έξι περιπτώσεις το μοντέλο κάνει την καλύτερη πρόβλεψη ακολουθώντας την στρατηγική multimodel, δεν μπορούμε να πούμε πως κάποια στρατηγική πρόβλεψης απ'τις τρεις σημειώνει γενικά καλύτερα αποτελέσματα.

- Εύκολα παρατηρεί κανείς πως τα μοντέλα γενικά κάνουν καλύτερες προβλέψεις για την χρονοσειρά της φάσης Α. Οι προβλέψεις για την φάση Β είναι χειρότερης ποιότητας, αλλά όχι κατά πάρα πολύ. Αυτές που είναι σημαντικά χειρότερες, είναι οι προβλέψεις για την χρονοσειρά της φάσης C. Το παραπάνω εξηγείται από την τυχαιότητα που χαρακτηρίζει την κάθε χρονοσειρά, η οποία καθιστά δύσκολη την πρόβλεψή της βάσει των παρελθοντικών τιμών.

- Για ορίζοντα έξι ώρων, την πρόβλεψη με την μεγαλύτερη ακρίβεια, δηλαδή με το μικρότερο MAPE, έκανε στην φάση Α ο αλγόριθμος των κ-κοντινότερων γειτόνων (0.6935), στην φάση Β η μέθοδος της γραμμικής παλινδρόμησης (9.7399) και στην φάση C ο αλγόριθμος του τυχαίου δάσους (0.7403). Για ορίζοντας ίσο με μία μέρα, η καλύτερη πρόβλεψη έγινε για την φάση Α από το δένδρο απόφασης (0.8162) και για τις φάσεις Β και C από τον αλγόριθμο των μηχανών διανυσμάτων υποστήριξης (13.9396 και 0.7337). Όλες οι περιπτώσεις πλην των προβλέψεων για την φάση Β στον ορίζοντα έξι ωρών και για την φάση C στον ορίζοντα μίας μέρας είναι βέλτιστες για όλα τα μοντέλα που δοκιμάσαμε.

- Παρά την απλότητά της, η γραμμική παλινδρόμηση πετυχαίνει καλές προβλέψεις. Μάλιστα, σε κάποιες περιπτώσεις όπως η πρόβλεψη για την χρονοσειρά της φάσης Β μέσω στρατηγικής multistep (αναδρομική), η γραμμική παλινδρόμηση παράγει τις καλύτερες προβλέψεις μεταξύ των έξι αλγορίθμων.

- Σε όλες τις στρατηγικές πρόβλεψης ο αλγόριθμος των κ-κοντινότερων γειτόνων παρουσιάζει μεγάλα MAPE στην φάση C, σημαντικά μεγαλύτερα από αυτά των

άλλων αλγορίθμων. Ωστόσο, σε αρκετές περιπτώσεις ο αλγόριθμος αυτός παράγει καλές προβλέψεις. Το γεγονός αυτό σε συνδυασμό με την ταχύτητα διαμόρφωσης και εκπαίδευσης του, μας οδηγούν στο συμπέρασμα πως αξίζει να επιχειρήθουν προβλέψεις με αυτόν.

- Συγκρίνοντας την απόδοση του δένδρου απόφασης, της ενίσχυσης κλίσης και του τυχαίου δάσους βλέπουμε πως παρότι το πρώτο είναι σημαντικά απλούστερο των άλλων δύο, που δομούνται από αυτό, δεν υπολείπεται συστηματικά στην απόδοσή των προβλέψεων από αυτά. Αυτό μας δείχνει πως η πολυπλοκότητα του μοντέλου δεν συνεπάγεται απαραίτητα και καλύτερο αποτέλεσμα. Επιπλέον, βλέπουμε πως το δένδρο απόφασης εμφανίζει την μεγαλύτερη πτώση στην απόδοσή του όταν προβλέπουμε για μεγαλύτερο ορίζοντα.

- Οι μηχανές διανυσμάτων υποστήριξης, αποδίδουν πολύ καλά στην φάση C και γενικά για την multistep στρατηγική.

Για τα υπόλοιπα μοντέλα που δοκιμάσαμε (πίνακες 7.4 και 7.5) σημειώνουμε τα εξής:

- Η μέθοδος απλής εκθετικής εξομάλυνσης παρουσιάζει πολύ καλά αποτελέσματα παρά την απλότητά της. Για την διαμόρφωση της μεθόδου απαιτείται ελάχιστος χρόνος, σε καμία περίπτωση συγκρίσιμος με αυτόν των μοντέλων μηχανικής μάθησης, ενώ η ποιότητα των προβλέψεών της δεν απέχει πολύ από την δική τους. Σημειώνεται πως η βέλτιστη τιμή του συντελεστή εξομάλυσης για την φάση C βρέθηκε ίσος με 1, συνεπώς η μέθοδος ταυτίζεται εκεί με την απλοϊκή (για την φάση A η αντίστοιχη τιμή ήταν 0.23 και για την B 0.09).

- Ο αλγόριθμος βαθιάς μάθησης N-Beats υπολείπεται της απλής εκθετικής εξομάλυνσης και των μοντέλων ARIMA. Αυτό πιθανώς εξηγείται από την ποσότητα των δεδομένων μας, δηλαδή ο αλγόριθμος ενδεχομένως χρειαζόταν περισσότερα δείγματα για να εκπαιδευτεί καταλλήλως για το πρόβλημα μας. Τροποποιήσεις για την βελτίωση της απόδοσής του, θα ήταν η επιλογή της ερμηνεύσιμης διαμόρφωσης για την αρχιτεκτονική του, αλλαγή του πλήθους των νευρώνων στα στρώματα και προσθήκη επιπλέον στρωμάτων στις στοίβες.

- Με τα μοντέλα ARIMA πετυχαίνουμε αρκετά καλές προβλέψεις, παρόμοιας ποιότητας με αυτές της απλής εκθετικής εξομάλυνσης. Σε δύο περιπτώσεις τα μοντέλα ARIMA πετυχαίνουν τις καλύτερες προβλέψεις μεταξύ όλων των μοντέλων που εξετάσαμε, στην φάση B για ορίζοντα έξι περιόδων (8.2070) και στην φάση C για ορίζοντα μίας ημέρας (0.7284). Στην πρώτη έχουμε μοντέλο ARIMA(2,1,6) και στην δεύτερη ARIMA(4,1,3).

- Παρά τις διαφορές στην δομή και την εκπαίδευσή τα μοντέλα νευρωνικών δικτύων δεν παρουσιάζουν πολύ μεγάλες διαφορές στην ποιότητα των προβλέψεών τους.

- Η αύξηση των εποχών εκπαίδευσης και του ρυθμού μάθησης δεν βελτιώνουν την επίδοση ενός μοντέλου αναγκαστικά. Αυτό φαίνεται συγκρίνοντας τα μοντέλα a και b. Το πιο εκπαιδευμένο μοντέλο b έχει καλύτερη απόδοση στον μεγαλύτερο ορίζοντα αλλά η απόδοσή του είναι χειρότερη στον μικρότερο ορίζοντα για τις φάσεις A και B.

- Δεν υπάρχει σημαντική διαφοροποίηση στην ποιότητα των προβλέψεων των νευρωνικών δικτύων από την εναλλαγή των στρατηγικών πρόβλεψης. Η καλύτερη πρόβλεψη για κάθε συνδυασμό ορίζοντα και φάσης, προέρχεται από ένα διαφορετικό μοντέλο νευρωνικού δικτύου. Ειδικότερα, για ορίζοντα ίσο με έξι ώρες, την φάση A προβλέπει καλύτερα το μοντέλο h (1.1107), την φάση B το μοντέλο a (14.9874) και την φάση C το μοντέλο i (0.7864). Για ορίζοντα ίσο με 24 ώρες, την φάση A προβλέπει καλύτερα το μοντέλο d (1.3245), την φάση B το μοντέλο j (17.6956) και την φάση C το μοντέλο c (1.9962).

Συνολικά, μπορούμε να σημειώσουμε πως οι μέθοδοι μηχανικής μάθησης, παρά τις υπολογιστικές απαιτήσεις για την επιλογή των βέλτιστων υπερπαραμέτρων, αξίζουν να χρησιμοποιηθούν καθώς από αυτές προκύπτουν τέσσερις από τις έξι καλύτερες προβλέψεις. Με τους αλγόριθμους μηχανικής μάθησης πετύχαμε τις καλύτερες προβλέψεις για την φάση A και στους δύο ορίζοντες, την φάση B για τον ορίζοντα μίας ημέρας και την φάση C για τον ορίζοντα έξι ωρών. Θα μπορούσαμε να πούμε πως σε γενικές γραμμές, με την χρήση μεθόδων μηχανικής μάθησης αντί της χρήσης απλών στατιστικών μοντέλων πρόβλεψης, είχαμε κόστος σε χρόνο κερδίσαμε όμως βελτίωση στην ακρίβεια των προβλέψεων μας. Τονίζεται πως με την επέκταση και βελτίωση του Cluster το χρονικό κόστος περιορίζεται, επιτρέποντάς μας να δοκιμάζουμε περισσότερους συνδυασμούς υπερπαραμέτρων και κατ'επέκτασιν να έχουμε καλύτερες πιθανότητες να διαμορφώσουμε μοντέλα που να κάνουν καλύτερες προβλέψεις. Η απόδοση των νευρωνικών δικτύων είναι μέτρια, ωστόσο αν ακολουθήσουμε και εκεί μία διαδικασία βελτιστοποίησης υπερπαραμέτρων λογικά θα βελτιωθεί. Καλές προβλέψεις έχουμε και με τις στατιστικές μεθόδους. Τέλος, σημειώνεται πως βελτίωση στην απόδοση των μεθόδων μηχανικής μάθησης μπορεί να φέρει και η αλλαγή του πλήθους των παρελθοντικών παρατηρήσεων που χρησιμοποιούμε ως εισόδους.

Κεφάλαιο 8

Μελλοντικές Επεκτάσεις

Η παρούσα εργασία μπορεί να επεκταθεί κυρίως σε δύο τομείς. Ο πρώτος είναι η εφαρμογή, όπου οι βελτιώσεις αφορούν την σχεδίασή της, την λειτουργικότητά της και την υλοποίησή της. Ο δεύτερος τομέας αφορά σε βελτιώσεις και επεκτάσεις στο θεωρητικό περιεχόμενο της εργασίας. Πέραν αυτών, μπορούμε να επεκτείνουμε το κομμάτι των πειραμάτων, κάνοντας περισσότερα πειράματα, για παράδειγμα εκτελώντας προβλέψεις για περισσότερους ορίζοντες, δοκιμάζοντας περισσότερα μεγέθη εισόδων για τους αλγορίθμους μηχανικής μάθησης και το νευρωνικό δίκτυο, αλλά και περισσότερες στατιστικές μεθόδους.
Αναφορικά με τις λειτουργικότητες της εφαρμογής μας, μία προφανής επέκταση είναι η συμπερίληψη βελτιστοποίησης υπερπαραμέτρων και για το νευρωνικό δίκτυο. Η βελτιστοποίηση αυτή θα μπορούσε κάλλιστα να γίνει με τον τρόπο που υλοποιείται για τους απλούς αλγορίθμους μηχανικής μάθησης. Επιπλέον, η εφαρμογή θα μπορούσε να γενικευτεί. Με άλλα λόγια, θα μπορούσαμε να μην περιοριζόμαστε στις τρεις συγκεκριμένες χρονοσειρές, αλλά να επιτρέπαμε στον χρήστη να εισάγει τις δικές του. Πέραν τούτου, μία άλλη επέκταση θα μπορούσε να είναι η out-of-sample προβλέψεις, όπου όμως δεν θα μπορούσαν να ελεγχθούν για την ποιότητά τους αμέσως, αλλά στο μέλλον, αν έχουμε τα κατάλληλα δεδομένα. Μια ακόμη επέκταση που εντάσσεται στο κομμάτι της λειτουργικότητας της εφαρμογής θα ήταν η παρουσίαση της πρόβλεψης όλων των επιλεγμένων αλγορίθμων και όχι μόνο του βελτίστου. Τέλος, θα μπορούσαμε να προσφέρουμε ως επιλογή την πρόβλεψη με στατιστικές μεθόδους, αλλά και την κατασκευή συνελικτικού δικτύου και αναδρομικού νευρωνικού δικτύου για την παραγωγή προβλέψεων. Σε κάθε περίπτωση θα πρέπει να δίνεται η δυνατότητα στον χρήστη να παραμετροποιεί πλήρως τις συγκεκριμένες αρχιτεκτονικές.
Όσον αφορά την σχεδίαση της εφαρμογής θα μπορούσαμε και εκεί να κάνουμε τροποποιήσεις στην εφαρμογή. Ειδικότερα, μία ιδέα θα ήταν να βλέπαμε το σχήμα του νευρωνικού δικτύου που δημιουργήσαμε. Πέραν της σχεδίασης, είναι σαφές πως η εφαρμογή μπορεί να επεκταθεί και στο κομμάτι του hardware. Όπως έχουμε γράψει μέρη της εφαρμογής εκτελούνται με την βοήθεια ενός Cluster υπολογιστών. Θα μπορούσαμε, λοιπόν, να επεκτείνουμε την εφαρμογή μας προσθέτοντας υπολογιστικούς κόμβους στο Cluster αυτό ή βελτιώνοντας τους παρόντες κόμβους αναβαθμίζοντάς τους. Η βιβλιοθήκη Dask μας επιτρέπει να πράξουμε το πρώτο με ιδιαίτερως μεγάλη ευκολία. Θα μπορούσαμε επίσης να εκτελούμε ακόμη περισσότερες λειτουργίες μέσω του Cluster μας.
Επεκτάσεις επιδέχεται και το θεωρητικό κομμάτι της εργασίας. Στο κεφάλαιο 2, στην παράγραφο 2.2, στην σελίδα 25, κάναμε μια σύντομη αναφορά σε μία σειρά από στατιστικές τεχνικές προβλέψεων (π.χ. μέθοδος Θ). Μία επέκταση της εργασίας είναι η πλήρης ανάλυσή τους. Επιπροσθέτως, υπάρχει πληθώρα μεθόδων μηχανικής μάθησης που θα μπορούσαν να προστεθούν στις έξι που έχουμε συμπεριλάβει στην παρούσα εργασία (π.χ. Lasso Regression και XGBoost). Πέραν των μεθόδων μηχανικής μάθησης υπάρχουν αρχιτεκτονικές βαθιάς μάθησης που μπορούμε να εμβαθύνουμε περισσότερο, όπως τα συνελικτικά δίκτυα και τα αναδρομικά δίκτυα, ενώ υπάρχουν και άλλοι σύγχρονοι αλγόριθμοι πέραν του N-Beats που περιέχουν αρχιτεκτονικές βαθιάς μάθησης που μπορούμε να αναλύσουμε (π.χ. το υβριδικό μοντέλο ESRNN). Γενικά, θα μπορούσαμε να πούμε πως τα πεδία της μηχανικής μάθησης και της βαθιά μάθησης περιλαμβάνουν τόσο μεγάλη ποικιλία θεμάτων προς ανάλυση, που είναι αδύνατον να καλυφθούν όλα επαρκώς σε μία εργασία.

Βιβλιογραφία

[1] URL: https://fastapi.tiangolo.com/.

[2] URL: https://docs.dask.org/en/stable/.

[3] *1.10. decision trees*. URL: https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning.

[4] *1.6. nearest neighbors*. URL: https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbor-algorithms.

[5] *Activation functions*. URL: https://ml-cheatsheet.readthedocs.io/en/latest/activation_functions.html.

[6] Admin. *LSTM vs GRU network: Which has better performance? - deep learning tutorial*. July 2020. URL: https://www.tutorialexample.com/lstm-vs-gru-network-which-has-better-performance-deep-learning-tutorial/.

[7] Ken W. Alger. *Quick start: BSON data types - date*. Mar. 2022. URL: https://www.mongodb.com/developer/quickstart/bson-data-types-date/.

[8] *API*. June 2022. URL: https://en.wikipedia.org/wiki/API.

[9] Chaya Bakshi. *Random Forest regression*. Apr. 2022. URL: https://levelup.gitconnected.com/random-forest-regression-209c0f354c84.

[10] Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne. "Machine learning strategies for time series forecasting". In: *European business intelligence summer school*. Springer. 2012, pp. 62–77.

[11] Leo Breiman et al. *Classification and regression trees*. Routledge, 2017.

[12] François Chollet. *Deep Learning with Python*. Manning, 2018. ISBN: 9781617294433.

[13] *Components and props*. URL: https://reactjs.org/docs/components-and-props.html.

[14] Sean Kevin Doherty. "Control of pH in chemical processes using artificial neural networks". PhD thesis. Liverpool John Moores University Liverpool, UK, 1999.

[15] Arnaud Nguembang Fadja, Evelina Lamma, Fabrizio Riguzzi, et al. "Vision Inspection with Neural Networks." In: *RiCeRcA@ AI* IA*. 2018.

[16] Fahimeh Farahnakian et al. "Energy aware consolidation algorithm based on k-nearest neighbor regression for cloud data centers". In: *2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*. IEEE. 2013, pp. 256–259.

[17] Weijiang Feng et al. "Audio visual speech recognition with multimodal recurrent neural networks". In: *2017 International Joint Conference on neural networks (IJCNN)*. IEEE. 2017, pp. 681–688.

[18] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures*. University of California, Irvine, 2000.

[19] Genesis. *Gradient descent- part 2*. Oct. 2018. URL: https://www.fromthegenesis.com/gradient-descent-part-2/.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[21] Simon Haykin. *Νευρωνικά Δίκτυα και Μηχανική Μάθηση*. 3η. Παπασωτηρίου, 2010. ISBN: 978-960-7182-64-7.

[22] Jiangshui Hong et al. "An overview of multi-cloud computing". In: *Workshops of the international conference on advanced information networking and applications*. Springer. 2019, pp. 1055–1068.

[23] *Hooks at a glance*. URL: https://reactjs.org/docs/hooks-overview.html.

[24] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. *A practical guide to support vector classification*. 2003.

[25] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018.

[26] *InfluxDB 1.8 documentation*. URL: https://docs.influxdata.com/influxdb/v1.8/.

[27] Ravi Kishore Kodali and Arshiya Anjum. "IoT based home automation using node-red". In: *2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT)*. IEEE. 2018, pp. 386–390.

[28] Ananth Mohan, Zheng Chen, and Kilian Weinberger. "Web-search ranking with initialized gradient boosted regression trees". In: *Proceedings of the learning to rank challenge*. PMLR. 2011, pp. 77–89.

[29] *Mongodb documentation*. URL: https://www.mongodb.com/docs/.

[30] Douglas C. Montgomery, Cheryl Jennings, and Murat Kulahci. *Introduction to Time Series Analysis and Forecasting*. Wiley, 2015.

[31] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. "Time series databases and influxdb". In: *Studienarbeit, Université Libre de Bruxelles* 12 (2017).

[32] Keiron O'Shea and Ryan Nash. "An introduction to convolutional neural networks". In: *arXiv preprint arXiv:1511.08458* (2015).

[33] *Open source time series platform - The tick stack*. Apr. 2022. URL: https://www.influxdata.com/time-series-platform/.

[34] Boris N Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *arXiv preprint arXiv:1905.10437* (2019).

[35] José Ortiz-Bejar et al. "K-nearest neighbor regressors optimized by using random search". In: *2018 IEEE international autumn meeting on power, electronics and computing (ROPEC)*. IEEE. 2018, pp. 1–5.

[36] Leonard Richardson, Mike Amundsen, and Sam Ruby. *RESTful Web APIs*. O'REILLY, 2013. ISBN: 9781449358068.

[37] Stuart Russell and Peter Norvig. *Τεχνητή Νοημοσύνη Μια σύγχρονη προσέγγιση*. Κλειδάριθμος, 2005. ISBN: 960-209-873-2.

[38] *Scaling Data Science in Python with Dask*. Coiled. URL: https://coiled.io/wp-content/uploads/2021/04/Coiled-Scaling-Data-Science-in-Python-with-Dask.pdf.

[39] Abraham Silberschatz, Henry F. Korth, and S. Sudarshan. *Συστήματα Βάσεων Δεδομένων*. M. Γκιούρδας, 2015. ISBN: 978-960-512-623-0.

[40] *Sklearn.metrics.DistanceMetric*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.DistanceMetric.html#sklearn.metrics.DistanceMetric.

[41] *Sklearn.tree.decisiontreeregressor*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html.

[42] Alex J Smola and Bernhard Schölkopf. "A tutorial on support vector regression". In: *Statistics and computing* 14.3 (2004), pp. 199–222.

[43] J Srinivas, K Venkata Subba Reddy, and A Moiz Qyser. "Cloud computing basics". In: *International journal of advanced research in computer and communication engineering* 1.5 (2012), pp. 343–347.

[44] Thomas Cormen; Charles Leiserson; Ronald Rivest; Clifford Stein. *Εισαγωγή στους αλγορίθμους*. 2nd ed. ΠΑΝΕΠΙΣΤΗΜΙΑΚΕΣ ΕΚΔΟΣΕΙΣ ΚΡΗΤΗΣ, 2016. ISBN: 978-960-524-473-6.

[45] Souhaib Ben Taieb et al. "A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition". In: *Expert systems with applications* 39.8 (2012), pp. 7067–7083.

[46] Sean J Taylor and Benjamin Letham. "Forecasting at scale". In: *The American Statistician* 72.1 (2018), pp. 37–45.

[47] *TechEmpower framework benchmarks*. URL: https://www.techempower.com/benchmarks/.

[48] Gareth James; Daniela Witten; Trevor Hastie; Robert Tibshirani. *An Introduction to Statistical Learning with Applications in R*. Springer, 2013. ISBN: 978-1-4614-7137-0.

[49] Sanford Weisberg. *Applied Linear Regression*. 4th ed. WILEY, 2014. ISBN: 978-1-118-38608-8.

[50] *What is mongodb atlas?* URL: https://www.mongodb.com/docs/atlas/.

[51] *What is the difference between 'Deep Learning','machine learning' and 'artificial intelligence'? is 'deep learning' related to 'data science'?* URL: https://www.quora.com/What-is-the-difference-between-Deep-Learning-Machine-learning-and-Artificial-Intelligence-Is-Deep-learning-related-to-data-science.

[52] Zheng Xiong et al. "Evaluating explorative prediction power of machine learning algorithms for materials discovery using k-fold forward cross-validation". In: *Computational Materials Science* 171 (2020), p. 109203.

[53] Yanru Zhang and Ali Haghani. "A gradient boosting method to improve travel time prediction". In: *Transportation Research Part C: Emerging Technologies* 58 (2015), pp. 308–324.

[54] Ιωάννης Βλαχάβας et al. *Τεχνητή Νοημοσύνη*. 3rd ed. Εκδόσεις Πανεπιστημίου Μακεδονίας, 2011. ISBN: 978-960-8396-64-7.

[55] Γ.Ε. Κοκκολάκης. *Σημειώσεις Ανάλυσης Χρονοσειρών*. Τομέας Μαθηματικών, Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών, Εθνικό Μετσόβιο Πολυτεχνείο. URL: http://www.math.ntua.gr/~kokolakis/SEMFE/TimeSeries_Ch_1.pdf.

[56] Φώτιος Πετρόπουλος and Βασίλειος Ασημακόπουλος. *Επιχειρησιακές Προβλέψεις*. Εκδόσεις Συμμετρία, 2013. ISBN: 978-960-266-333-2.

[57] Ευάγγελος Σπηλιώτης. *Ολοκληρωμένα Αυτοπαλινδρομικά Μοντέλα Κινητού Μέσου Όρου (ARIMA). Σημειώσεις για το μάθημα 'Τεχνικές Προβλέψεων'*. ΣΗΜΜΥ, ΕΜΠ. URL: https://www.fsu.gr/el/component/jdownloads/finish/6/1715.