ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΔΡΟΛΟΓΙΑΣ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗΣ ΥΔΑΤΙΚΩΝ ΠΟΡΩΝ

Πρόβλεψη καταναλώσεων Νερού – Ενέργειας με χρήση εξελιγμένων μοντέλων μηχανικής μάθησης

ΜΙΧΑΛΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ

Επιβλέποντες καθηγητές:

Μακρόπουλος Χρήστος, Καθηγητής Ε.Μ.Π.

Ευστρατιάδης Ανδρέας, Επίκουρος Καθηγητής Ε.Μ.Π.

Αθήνα, Φεβρουάριος 2022

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΠΟΛΙΤΙΚΩΝ ΜΗΧΑΝΙΚΩΝ

ΕΡΓΑΣΤΗΡΙΟ ΥΔΡΟΛΟΓΙΑΣ ΚΑΙ ΑΞΙΟΠΟΙΗΣΗΣ ΥΔΑΤΙΚΩΝ ΠΟΡΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Πρόβλεψη καταναλώσεων Νερού – Ενέργειας με χρήση εξελιγμένων μοντέλων μηχανικής μάθησης

DIPLOMA THESIS

Prediction of Water - Energy consumption using advanced machine learning models

ΜΙΧΑΛΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ

Επιβλέποντες: Μακρόπουλος Χρήστος, Καθηγητής Ε.Μ.Π.

Ευστρατιάδης Ανδρέας, Επίκουρος Καθηγητής Ε.Μ.Π.

“Καμία σπουδαία ανακάλυψη δεν έγινε

ποτέ χωρίς μια τολμηρή εικασία.”

Ισαάκ Νεύτων

ΕΥΧΑΡΙΣΤΙΕΣ

Η παρούσα Διπλωματική Εργασία πραγματοποιήθηκε στο Εργαστήριο Υδρολογίας και Αξιοποίησης Υδατικών Πόρων του Εθνικού Μετσόβιου Πολυτεχνείου στη σχολή Πολιτικών Μηχανικών. Με την ολοκλήρωση της εργασίας αυτής ολοκληρώνεται και ένας πενταετής κύκλος σπουδών. Ένας δύσκολος κύκλος σπουδών με αρκετές απογοητεύσεις αλλά ακόμα περισσότερες επιβραβεύσεις κατά την διάρκειά του. Οι εμπειρίες, οι γνώσεις και οι χαρές που έχω λάβει τα τελευταία πέντε χρόνια είναι υπεύθυνες για την εξέλιξη της προσωπικότητας μου αλλά και για την αλλαγή στον τρόπο που προσεγγίζω τα καθημερινά προβλήματα με τα οποία έρχομαι αντιμέτωπος.

Για τον λόγο αυτό θα ήθελα αρχικά να ευχαριστήσω τους επιβλέποντες της εργασίας μου και πιο συγκεκριμένα τον κ. Μακρόπουλο Χρήστο, Καθηγητή Ε.Μ.Π. και τον κ. Ανδρέα Ευστρατιάδη Επίκ. Καθηγητή Ε.Μ.Π., οι οποίοι δέχθηκαν να με αναλάβουν και να με βοηθήσουν στην επιλογή ενός θέματος που θα ταίριαζε στα ενδιαφέροντα μου αλλά και στις δεξιότητές μου. Χωρίς αυτούς το τελικό αποτέλεσμα της εργασίας δεν θα ήταν ποτέ σε αυτό το επίπεδο.

Θα ήθελα να ευχαριστήσω θερμά όλη την ερευνητική ομάδα του τομέα και πιο συγκεκριμένα τον Παναγιώτη Δήμα (Πολιτικό Μηχανικό Ε.Μ.Π., MSc, Υπ. Δρ.), τον Παναγιώτη Κοσσιέρη (Πολιτικό Μηχανικό Ε.Μ.Π., MSc, Δρ. Μηχανικό), τον Γιάννη Τσουκαλά (Πολιτικό Μηχανικό Ε.Μ.Π., MSc, Δρ. Μηχανικό), τον Διονύση Νικολόπουλο (Πολιτικό Μηχανικό Ε.Μ.Π., MSc, Υπ. Δρ.), τον Νίκο Πελεκάνο (Πολιτικό Μηχανικό Ε.Μ.Π.), τον Σπύρο Τσαταλιό (Πολιτικό Μηχανικό ΕΜΠ,), τον Γιώργο Μωραΐτη (Πολιτικό Μηχανικό ΕΜΠ, MSc, Υπ. Δρ.) καθώς και την Τζωρτζίνα Σακκή (Πολιτικό Μηχανικό ΕΜΠ, Υπ. Δρ.) οι οποίοι με το έντονο ερευνητικό τους ενδιαφέρον ήταν πάντα θετικοί στο άκουσμα των προβληματισμών μου και πλήρως επεξηγηματικοί στις απορίες που προέκυψαν μέχρι την υλοποίηση της εργασίας αυτής.

Ακόμα δεν θα έπρεπε να αμεληθεί η προσφορά των μελών της ΕΥΔΑΠ και ειδικότερα η βοήθεια της κ. Κωνσταντίνας Σταυρούλια ως Διευθύντρια Δικτύου Ύδρευσης. Η βοήθεια της ήταν μοναδική αφού με την καθοδήγηση της εντοπίστηκε αμέσως η ουσία του προβλήματος και με την πολυετή εμπειρία της δημιούργησε μια ορμή προς την σωστή κατεύθυνση για την επίλυσή του.

Σε προσωπικό επίπεδο θα ήθελα να ευχαριστήσω όλους τους φίλους μου και τους συμφοιτητές μου για την αμέριστη συμπαράσταση τους αλλά και για τις όμορφες στιγμές που ζήσαμε σε όλο το ταξίδι της πενταετούς φοίτησης.

Τέλος θέλω να ευχαριστήσω πολύ την οικογένειά μου οι οποίοι υπήρξαν πάντα ένα ανεκτίμητο στήριγμα για εμένα και στους οποίους οφείλω όλη την διαδρομή των σπουδών μου, μέχρι και σήμερα.

Χρήστος Μιχαλόπουλος

Αθήνα, 2022

ΠΕΡΙΛΗΨΗ

Η Εταιρεία Υδρεύσεως και Αποχετεύσεως Πρωτευούσης Α.Ε. (ΕΥΔΑΠ), είναι η μεγαλύτερη εταιρεία στην Ελλάδα που δραστηριοποιείται στην αγορά και διάθεση νερού. Το πελατολόγιο της ΕΥΔΑΠ στον τομέα της ύδρευσης περιλαμβάνει περίπου 4.400.000 πελάτες (2.160.000 συνδέσεις), ενώ το μήκος των αγωγών ανέρχεται σε 14.000 χλμ. Με αφορμή την κρίση λειψυδρίας όπου έπληξε το λεκανοπέδιο της Αττικής την δεκαετία του 1990, άρχισε σταδιακά να αναδεικνύεται η σημαντικότητα της διαχείρισης του πόσιμου νερού. Ένα μεγάλο κεφάλαιο της διαχείρισης του πόσιμου νερού είναι ο περιορισμός των απωλειών στην διαδικασία μεταφοράς αλλά και ο τυπολογικός προσδιορισμός της διαρροής. Η πιο βασική μεθοδολογία για τον υπολογισμό των απωλειών σε ένα κλειστό δίκτυο είναι η μέθοδος του ισοζυγίου. Η μέθοδος εκμεταλλεύεται της αρχή της συνέχειας των ρευστών για την εύρεση του χαμένου νερού, δηλαδή ό,τι εισέρχεται στο δίκτυο θα πρέπει να εξέρχεται από αυτό. Η διαφορά του όγκου νερού που εισέρχεται στο δίκτυο με τον όγκο που εξέρχεται από το δίκτυο είναι οι συνολικές απώλειες. Ως γνωστόν όμως η πραγματικότητα απέχει πολύ από την θεωρία, ένα πρόβλημα που στην μικρή κλίμακα μπορεί να είναι πολύ εύκολο να επιλυθεί δεν σημαίνει ότι θα είναι εύκολη η επίλυση του στην μεγαλύτερη κλίμακα. Η αξιοπιστία του ισοζύγιού εξαρτάται άμεσα από τις μετρήσεις των επιμέρους καταναλώσεων. Παρόλα αυτά, λόγω της μεγάλης διάστασης του δικτύου η ΕΥΔΑΠ δεν έχει τη δυνατότητα να μετρήσει όλα τα παροχόμετρα ανά τρίμηνο. Από το σύνολο των περίπου 2.200.000 μετρητών κάθε τρίμηνο η μέτρηση περίπου 200.000 από αυτούς δεν γίνεται έγκαιρα, δηλαδή στο πλαίσιο μίας χρονιάς δεν πραγματοποιούνται κατά μέσο όρο 800.000 μετρήσεις. Αυτές οι ατελείς καταγραφές στο σύνολο των καταναλωτών προσδίδουν σφάλμα κατά την συνολική εκτίμηση των απωλειών στο δίκτυο. Ο στόχος επίσης είναι η όσο το δυνατόν καλύτερη εκτίμηση του όγκου νερού που έχει καταναλωθεί προκειμένου να έχουμε ένα ισοζύγιο όσο γίνεται πιο κοντά στην πραγματικότητα και έτσι να εντοπίζονται οι απώλειες του δικτύου με πιο γρήγορο και αξιόπιστο τρόπο. Επίσης, τα αποτελέσματα αυτά μπορούν να έχουν και άλλη βοηθητική χρήση για την εταιρία καθώς μπορούν να ενισχύσουν την πιο ακριβή τιμολόγηση για τους πελάτες που δεν έχει πραγματοποιηθεί μέτρηση (τεκμαρτές τιμολογήσεις). Για τις ανάγκες του προβλήματος τα δεδομένα και οι υπολογισμοί πραγματοποιούνται σε υπολογιστικό περιβάλλον Python. Λόγω της πολυπλοκότητας του προβλήματος επιλέγεται η χρήση στατιστικών μοντέλων καθώς και μηχανικής μάθησης για παλινδρόμηση όπως ARIMA, Seasonal ARIMA, Neural Networks, Random Forest. Ακόμα συντάχθηκαν νέες μεθοδολογίες όπου εκμεταλλεύονται την μορφή των δεδομένων. Δηλαδή πραγματοποιούνται προγνώσεις βάσει των καταναλωτών στους οποίους έχουν γίνει μετρήσεις. Οι μεθοδολογίες αυτές στηρίζονται σε αλγορίθμους ομαδοποίησης όπως kNN (k πλησιέστεροι γείτονες) και μοντέλα μίξης γκαουσιανών κατανομών. Τα μοντέλα δοκιμάστηκαν σε συνθετικά δεδομένα λόγω απουσίας πραγματικών. Για λόγους πληρότητας κρίθηκε απαραίτητο τα μοντέλα να δοκιμαστούν και σε πραγματικά δεδομένα. Τα τελευταία χρόνια η σχετική τεχνολογία έχει επεκταθεί πολύ στον τομέα της διαχείρισης της ενέργειας, με αποτέλεσμα τέτοιου είδους δεδομένα να είναι άμεσα προσβάσιμα στην επιστημονική κοινότητα για αυτό τον λόγο επιλέχθηκαν δεδομένα ηλεκτρικής κατανάλωσης ενέργειας από δέκα πολιτείες της Αμερικής, τα οποία και ανακτήθηκαν από ελεύθερα διαθέσιμες πηγές.

ABSTRACT

The Athens Water Supply and Sewerage Company (EYDAP), is the largest active company in Greece in the water market. EYDAP's clientele in the field of water supply includes about 4,400,000 customers (2,160,000 connections), while the length of the pipelines counts to 14,000 km. The big water crisis that struck Athens in the nineties highlighted the importance of drinking water management. A big chapter of drinking water management is the reduction of losses in the transport process, but also the typological determination of the leak. The most basic methodology for calculating losses in a closed network is the water balance method. The method exploits the principle of continuity of fluids to find the amount of lost water, that is, what enters the network should come out (in terms of total water volume). The difference between the volume of water entering the network and the volume leaving the network equals the total losses. However, a problem that can be easily solved at small scale, does not mean that will be that easy on a larger scale. Balancing the solution requires measurements from consumer consumption. Nevertheless, due to the large size of the problem, EYDAP is not able to measure all the installed flow meters at the quarter-year scale. From the grand total of about 2,200,000 flowmeters that must be recorded, EYDAP is not able to measure about 200,000 of them. Hence, within a year approximately 800,000 measurements are not performed. These incomplete recordings contribute to the overall estimation of network losses. The aim of this work is to better estimate the volume of water the users have consumed to better estimate the water balance for calculating and locating losses. These results can of course have another use for the company, namely a more accurate pricing for customers whose consumption has not been recorded (imputed invoice). For the needs of the problem the data and calculations will be performed in Python computing environment. Due to the complexity of the problem, the use of statistical models is chosen, in combination with Machine Learning for regression models, such as ARIMA, Seasonal ARIMA, Neural Networks, Random Forest. New methodologies have also been developed which take advantage of the format of the data format. That is, forecasts based on the consumers that have been measured are made utilizing such methodologies. The latter are based on clustering algorithms such as kNN (k Nearest Neighbors) and Gaussian mixture models. The models are tested on synthetic data access to the real data was not available till the thesis completion. For reasons of completeness, it was deemed necessary to test the models on real data. In recent years, technology has expanded greatly in the field of energy management, with the result that such data are easily accessible to the scientific community. For this reason, electricity consumption data were selected from ten American states, retrieved from publicly available online sources

ΠΕΡΙΕΧΟΜΕΝΑ

1. Εισαγωγή

1.1 Σκοπός της εργασίας

Η συγκεκριμένη εργασία αποτελείται από ένα κύκλο βοηθητικών εργασιών που θα πρέπει να ολοκληρωθούν ώστε να μπορέσουν με όσο το δυνατόν μεγαλύτερη ακρίβεια να εντοπιστούν οι απώλειες σε ένα κλειστό δίκτυο διανομής πόσιμου νερού . Κατά την προσπάθεια υπολογισμού του ισοζυγίου αντιμετωπίζονται προβλήματα όπως ανακρίβειες στις μετρήσεις καθώς και νερό χρήσης το οποίο δεν τιμολογείται με αποτέλεσμα να μην καταχωρείται στο ισοζύγιο (π.χ. Νερό από χρήση πυροσβεστικού κρουνού, έκτακτες ανάγκες, εν γένει μη τιμολογούμενο νερό) . Η ΕΥΔΑΠ τα τελευταία χρόνια έχει κάνει μια αξιόλογη προσπάθεια για τον περιορισμό των μη καταγραφόμενων ποσοτήτων του δικτύου, τοποθετώντας σταδιακά μετρητές σε κρουνούς και σε δήμους οι οποίοι πραγματοποιούσαν καταναλώσεις χωρίς να κοστολογούνται. Ακόμα ένα πολύ σημαντικό βήμα για τον πιο ολοκληρωμένο τρόπο υπολογισμού των απωλειών και του εντοπισμού της πιθανής θέσης της βλάβης υδροδότησης είναι η τοποθέτηση ενδιάμεσων μετρητών σε κρίσιμους κόμβους του δικτύου. Η νέα αυτή σημαντική εργασία έχει ήδη ξεκινήσει και υπολογίζεται ότι θα ολοκληρωθεί στην επόμενη πενταετία. Ο στόχος της εργασίας έρχεται να επιλύσει ένα βασικό πρόβλημα που αντιμετωπίζει η ΕΥΔΑΠ κατά την κατάρτιση του ισοζυγίου που πραγματοποιεί στο τέλος κάθε έτους και έγκειται στη μη έγκαιρη καταμέτρηση των μετρητών των καταναλωτών. Πιο συγκεκριμένα από τούς 2.200.000 ενεργούς καταναλωτές δεν πραγματοποιείται μέτρηση για 200.000 μετρητές ανά τρίμηνο κατά μέσο όρο. Στην περίπτωση της ΕΥΔΑΠ όπου το ισοζύγιο ανάγεται σε ετήσια βάση, οι άγνωστες μετρήσεις γίνονται περίπου 800.000 κατά μέσο όρο. Για τον λόγο αυτό κρίνεται αναγκαία η προσπάθεια εξάλειψης του σφάλματος στο συνολικό ισοζύγιο μέσω μοντέλων πρόβλεψης των καταναλώσεων για τους καταναλωτές όπου δεν διατίθεται η μέτρηση. Μέσα από την εργασία αυτή εξετάζεται το ζήτημα της πρόβλεψης των μετρήσεων, αξιοποιώντας για την επίλυση του τρία είδη μοντέλων, τα στατιστικά (statistical), εποπτευόμενης μάθησης (supervised learning), ομαδοποίησης (clustering) καθώς και συνδυασμό αυτών. Κατά την υπάρχουσα ανάλυση επιδιώκεται, πέρα από το αμιγώς υπολογιστικό κομμάτι, και η βαθύτερη κατανόηση της φύσης, των ιδιαιτεροτήτων και των χαρακτηριστικών του κάθε μοντέλου μέσω της προσπάθειας ερμηνείας των αποτελεσμάτων τους.

Τα στατιστικά μοντέλα που χρησιμοποιήθηκαν στα πλαίσια της εργασίας είναι: ARIMA , Seasonal ARIMA καθώς και μοντέλο μίξης Γκαουσιανών κατανομών. Τα μοντέλα εποπτευομένης μάθησης που εξετάστηκαν είναι: Νευρώνικα Δίκτυα (Neural Networks), Δίκτυα Μακράς Βραχύχρονης Μνήμης (LSTM), Τυχαίο Δάσος (Random Forest) και τέλος τα μοντέλα ομαδοποίησης, όπου έγινε χρήση μοντέλων κοντινών γειτόνων (k- nearest neighbors) καθώς και ομαδοποίηση με μοντέλα μίξης (Mixture Models).

Στα πλαίσια της μελέτης έπρεπε να γίνει και η διαδικασία της μορφοποίησης των δεδομένων. Γενικά στην επιστήμη της ανάλυσης δεδομένων αυτό μπορεί να είναι και το πιο χρονοβόρο κομμάτι της έρευνας καθώς τα δεδομένα των μετρήσεων σπάνια έως ποτέ δεν είναι συστηματικά (έλλειψη μετρήσεων, σφάλματα και άλλα) και ιδανικά οργανωμένα. Ακόμα θα πρέπει να σημειωθεί πως όλος ο σχεδιασμός γίνεται σε προγραμματιστικό περιβάλλον Python με την βοήθεια έτοιμων πακέτων του συστήματος πακέτων Anaconda.

1.2 Διάρθρωση της εργασίας

Η παρούσα έκθεση πέρα από το κεφάλαιο εισαγωγής (1ο κεφάλαιο) , περιλαμβάνει άλλα έξι (6) κεφάλαια και ένα παράρτημα.

Στο **2ο Κεφάλαιο** γίνεται μια εισαγωγή στην διαδικασία εύρεσης βέλτιστων παραμέτρων από μαθηματική σκοπιά καθώς γίνεται επεξήγηση του υποβάθρου που απαιτείται για την κατανόηση του μηχανισμού και του τρόπου λειτουργίας των μοντέλων. Επίσης αναλύονται οι συναρτήσεις σφάλματος που έχουν χρησιμοποιηθεί.

Στο **3ο Κεφάλαιο** αναφέρονται εκτενώς όλα τα υπολογιστικά πακέτα και το περιβάλλον που χρησιμοποιήθηκε για την εφαρμογή των μοντέλων στον ηλεκτρονικό υπολογιστή καθώς και τα πλεονεκτήματα και μειονεκτήματα τους.

Στο **4ο Κεφάλαιο** γίνεται ανάλυση των δεδομένων που διαθέτουμε για την πραγματοποίηση των προβλέψεων καθώς και για της διαδικασίας επεξεργασίας που πρέπει να πραγματοποιηθεί ώστε να τρέξουν τα υπολογιστικά πακέτα σωστά.

Στο **5ο Κεφάλαιο** παρουσιάζονται αναλυτικά οι μεθοδολογίες που ακολουθούνται για κάθε μοντέλο καθώς γίνεται εμβάθυνση σε αυτές από μαθηματικής προσέγγισης. Η κάθε μεθοδολογία έχει χωριστεί σε κατηγορίες ανάλογα με τον τρόπο λειτουργίας.

Στο **6ο Κεφάλαιο** γίνεται σύγκριση των αποδόσεων των μοντέλων για τις διάφορες συναρτήσεις σφάλματος που έχουν επιλεγεί καθώς γίνεται και σχολιασμός των αποτελεσμάτων. Τέλος υπογραμμίζονται πιθανές ιδέες για περισσότερη εμβάθυνση και επέκταση με υβριδικά μοντέλα.

Στο **7ο Κεφάλαιο** παρατίθεται η βιβλιογραφία που βασίστηκε η παρούσα εργασία.

Τέλος, στο Παράρτημα παρατίθεται ο κώδικας που συντάχθηκε για τις ανάγκες της εργασίας.

2. Θεωρητικό υπόβαθρο

2.1. Θεωρητικό υπόβαθρο για την κατανόηση της πρόβλεψης

Από τα αρχαιότατα χρόνια ο άνθρωπος επιδιώκει να κάνει προβλέψεις για τα θέματα που επηρεάζουν την ζωή του (καιρός, συμπεριφορά των συνανθρώπων του κ.ο.κ). Η έννοια της πρόβλεψης έχει απασχολήσει ολόκληρο το φάσμα της επιστήμης για την επίλυση πολλών και διαφορετικών προβλημάτων. Η μη εγκυρότητα της πρόβλεψης μπορεί να οδηγήσει σε μη επάρκεια ή και μερικές φορές σε πλήρη αστοχία των έργων του πολιτικού μηχανικού. Τέτοιες απαιτήσεις για προβλέψεις συναντά ο Υδραυλικός Μηχανικός κατά τον σχεδιασμό δικτύων ύδρευσης, αποχέτευσης, μηχανισμούς υπερχείλισης.

Για να επιτευχθεί η πιο ρεαλιστική πρόβλεψη, είναι αναγκαίο να εξεταστούν αποτελέσματα από διαφορετικά μοντέλα (με διαφορετική φιλοσοφία λειτουργίας) και να γίνει αξιολόγηση των δυνατών και αδύνατων σημείων τους.

2.1.1 Η μαθηματική έννοια του ακροτάτου

Κατά την εκπαίδευση (*training*) στόχος είναι η εξάλειψη του σφάλματος μεταξύ των τιμών που προβλέπουμε εν συγκρίσει των πραγματικών τιμών. Αυτή η ιδέα μπορεί να γενικευτεί σε ένα μαθηματικό πρόβλημα μείωσης του σφάλματος. Στόχος μας είναι η ελαχιστοποίηση της συνάρτησης κόστους (*loss function*).

Για την πιο απλή περίπτωση όπου η συνάρτηση έχει μόνο μια ανεξάρτητη μεταβλητή, μπορούμε απλά να συγκρίνουμε τα σημεία όπου η παράγωγος της συνάρτησης μηδενίζεται (τοπικά ελάχιστα) για την εύρεση του ολικού ελαχίστου στο πεδίο ορισμού της.

2.1.2 Υπολογισμός ακροτάτων συναρτήσεων πολλαπλών μεταβλητών

Σύμφωνα με τον λογισμό πολλαπλών μεταβλητών, αν η συνάρτηση κόστους που περιγράφει το πρόβλημα λαμβάνει πάνω από μία παράμετρο θα πρέπει να βρούμε τα σημεία που μηδενίζουν την κλίση της συνάρτησης (*gradient*).

$$\nabla := \left(\frac{\partial}{\partial x1}, \dots, \frac{\partial}{\partial xn}\right) \rightarrow gradf(x_n) = \nabla f(x_n) = \left(\frac{\partial f}{\partial x1}, \dots, \frac{\partial f}{\partial xn}\right) = \vec{0} \qquad (2.1)$$

όπου:

- $\frac{\partial}{\partial xn}$ η μερική παράγωγος ως προς x_n

Τα σημεία αυτά λέγονται κρίσιμα. Υπάρχουν μεθοδολογίες για τον χαρακτηρισμό αυτών των σημείων αλλά μπορούν να παρακαμφθούν με την μέθοδο της δοκιμής (σύγκριση ελαχίστου σημείου).

Για να μπορεί να γίνει η περιγραφή πολύπλοκων φαινομένων θα χρειαστούν πολύπλοκα μοντέλα με διανύσματα $\vec{x}$ ανώτερης διάστασης. Η προηγούμενη μεθοδολογία από τη φύση της καθίσταται ακατάλληλη για μεγάλο αριθμό μεταβλητών. Για την ανάγκη επίλυσης ενός μεγάλου συστήματος που τις περισσότερες φορές είναι μη γραμμικό, η προγραμματιστική πολυπλοκότητα (O) είναι πολύ μεγάλη για την υπολογιστική ισχύ που διαθέτουμε σήμερα.

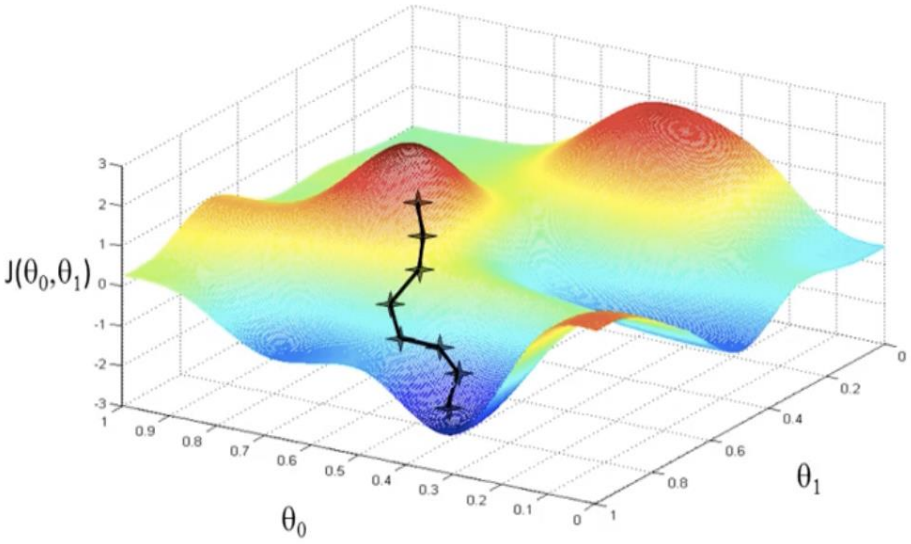
2.1.3 Μέθοδος κλίσης κατάβασης (Gradient descent)

Η μέθοδος κλίσης κατάβασης (Gradient descent) είναι ένας επαναληπτικός αλγόριθμος βελτιστοποίησης πρώτης τάξης για την εύρεση ενός τοπικού ελάχιστου μιας παραγωγίσιμης συνάρτησης. Η ιδέα είναι να γίνονται επαναλαμβανόμενα βήματα προς την αντίθετη κατεύθυνση της κλίσης της συνάρτησης στο τρέχον σημείο καθώς είναι πιο απότομη προς το τοπικό ελάχιστο. Παρόμοια μεθοδολογία ακολουθείται για την εύρεση του μεγίστου, όπου απλώς ακολουθείται η κατεύθυνση της κλίσης και όχι αντίθετη αυτής.

$$\Theta_{n+1} = \Theta_n - \nabla J\left(\vec{x}, \overrightarrow{\theta_n}\right) \tag{2.2}$$

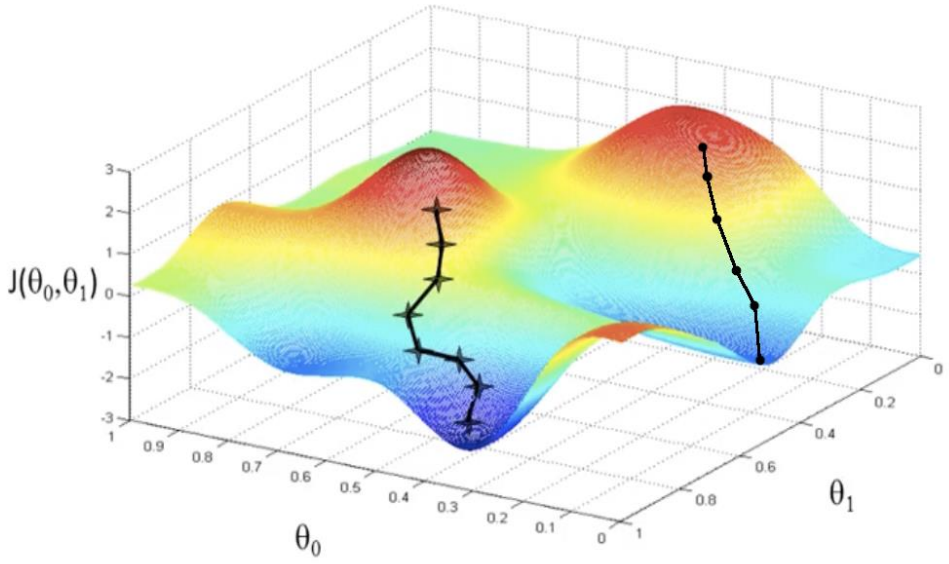
όπου:

- Θ_n οι παράμετροι που θέλουμε να βελτιστοποιήσουμε
- $\nabla J\left(\vec{x}, \overrightarrow{\theta_n}\right)$ η κλίση της συνάρτησης κόστους για τα δεδομένα $\vec{x}$



Διάγραμμα 2-1: Βήματα του αλγορίθμου κλίσης κατάβασης.

Στην παραπάνω εικόνα παρουσιάζεται πως λειτουργεί ο αλγόριθμος για μια απλή περίπτωση που μπορεί να απεικονισθεί στον χώρο R^3. Όπως μπορούμε να παρατηρήσουμε το σημείο που συγκλίνει ο αλγόριθμος δεν μπορεί να είναι αποκλειστικά ολικό μέγιστο καθώς οι αρχικές συνθήκες επηρεάζουν κατά πολύ το τελικό σημείο σύγκλισης.



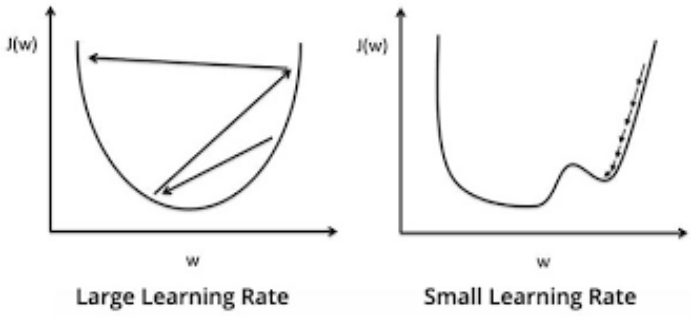
Διάγραμμα 2-2: Διαφορετικό σημείο σύγκλισης για διαφορετικό σημείο εκκίνησης.

Αν το αρχικό σημείο εκκίνησης του αλγορίθμου είναι για παράδειγμα η άλλη κορυφή στο παραπάνω σχήμα, τότε θα πραγματοποιηθεί σύγκλιση, αλλά το σημείο δεν θα είναι το ολικό ελάχιστο όπως παρατηρούμε. Ένας ακόμα παράγοντας που επηρεάζει την ποιότητα της σύγκλισης είναι το μέγεθος των βημάτων που πραγματοποιούνται σε κάθε επανάληψη. Για τον έλεγχο των βημάτων που λαμβάνονται σε κάθε βήμα προστίθεται ένας συντελεστής μάθησης (*learning rate*) α στην (2.2) (Kwiatkowski, 2021).

$$\Theta_{n+1} = \Theta_n - \alpha \nabla J\left(\overrightarrow{x}, \overrightarrow{\theta_n}\right) \tag{2.3}$$

- όπου: Θ_n οι παράμετροι που θέλουμε να βελτιστοποιήσουμε
- $\nabla J\left(\overrightarrow{x}, \overrightarrow{\theta_n}\right)$ η κλίση της συνάρτησης κόστους για τα δεδομένα $\vec{x}$
- α ο συντελεστής μάθησης

Ανάλογα με το πρόβλημα όπου αναζητείται σχετική επίλυση, θα πρέπει η επιλογή του συντελεστή μάθησης να γίνει με προσοχή καθώς, ο συντελεστής επηρεάζει την ταχύτητα σύγκλισης (τα μοντέλα θα χρειάζονται πολύ χρόνο εκπαίδευσης), την ποιότητα σύγκλισης (δεν επιτυγχάνεται ολικό ελάχιστο) και πολλές φορές οδηγεί σε αδυναμία σύγκλισης (απόκλιση από τον στόχο). Όπως βλέπουμε και στα διαγράμματα παρακάτω, για μεγάλους συντελεστές μάθησης είναι δυνατόν ο αλγόριθμος να μην μπορεί να συγκλίνει, με την μεταβολή να είναι πάρα πολύ μεγάλη σε κάθε βήμα αδυνατώντας να κάνει μικρά βήματα όταν χρειάζεται. Όμοια για την περίπτωση που έχουμε πολύ μικρό συντελεστή μάθησης, ο αλγόριθμος μπορεί να συγκλίνει σε σημείο το όποιο να είναι τοπικό ελάχιστο αλλά όχι το ολικό. Τα πολύ μικρά βήματα που κάνει τον καταδικάζουν στην σύγκλιση στο πιο κοντινό τοπικό ελάχιστο.



Διάγραμμα 2-3: Παραδείγματα άστοχης επιλογής συντελεστή μάθησης.

Η παράγωγος μπορεί να υπολογιστεί εύκολα με την βοήθεια του κανόνα της αλυσίδας (chain rule) ακόμα και για πεπλεγμένες συναρτήσεις αναλύοντας την σε γινόμενο γνωστών παραγώγων.

Η μορφή σύμφωνα με τον συμβολισμό κατά Leibniz:

$$\frac{dy}{dx} = \frac{dy}{dt}\frac{dt}{dx} \tag{2.4}$$

2.1.4 Στοχαστική κλίση κατάβασης (Stochastic Gradient Descent)

Η στοχαστική κλίση κατάβασης είναι μια βελτιωμένη εκδοχή τη κλασικής μεθόδου κλίσης κατάβασης. Η βασική φιλοσοφία της μεθόδου βασίζεται στην υλοποίηση μεγαλύτερου αριθμού επαναλήψεων σε σύγκριση με την κλασική μέθοδο, με αποτέλεσμα την μεγαλύτερη ακρίβεια του εκάστοτε βήματος. Η μέθοδος μπορεί να χαρακτηριστεί ως στοχαστική εκτίμηση της κλασικής κλίσης κατάβασης.

Η κλίση (gradient) σε αυτή την περίπτωση δεν υπολογίζεται από όλα τα δεδομένα αλλά από ένα τυχαίο υποσύνολο αυτών, μειώνοντας τον υπολογιστικό φόρτο κάθε βήματος. Λόγω ύπαρξης στοχαστικότητας η διασπορά σε κάθε βήμα θα αυξάνεται, για αυτό επιλέγεται ένας μεταβλητός συντελεστής μάθησης όπου μειώνεται με κάθε βήμα. Θέλουμε παρόλα αυτά την εναλλαγή των αποτελεσμάτων σε μεγάλους αριθμούς βημάτων. Για αυτό τον λόγο θέλουμε να ισχύουν οι παρακάτω περιορισμοί.

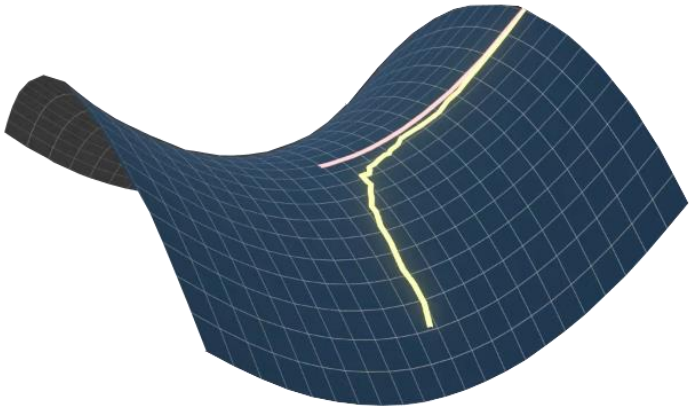
$$\sum_{t=1}^{\infty} n_t = \infty \sum_{t=1}^{\infty} {n_t}^2 < \infty \tag{2.5}$$

όπου:

- n_t ο συντελεστής μάθησης για το βήμα t

Για παράδειγμα μια συνάρτηση που ικανοποιεί τα παραπάνω κριτήρια της σχέσης (2.5) είναι: $n_t = \frac{1}{t+1}$

Η χρήση της προσέγγιση της κλίσης ακόμα δίνει λύση σε περιπτώσεις όπου η κλασσική μέθοδος θα αδυνατούσε.



Διάγραμμα 2-4: Διαφορά μεταξύ κλασσικής κλίσης κατάβασης και στοχαστικής.

Η απόλυτη καθοδήγηση της κλίσης κατάβασης όπως βλέπουμε και στο ανωτέρω διάγραμμα έχει ως αποτέλεσμα ο αλγόριθμος να συγκλίνει στο σημείο σέλας (το οποίο δεν είναι τοπικό ελάχιστο). Η προσέγγιση της κλίσης προσδίδει μία τυχαιότητα η οποία βοηθάει στην αποφυγή του σημείου σέλας, όπως φαίνεται και παραπάνω. Ο ίδιος μηχανισμός βοηθάει στην αποφυγή μικρών τοπικών ελαχίστων όπου η κλασική μέθοδος θα είχε καταλήξει.

Αυτή είναι η βασική μεθοδολογία εφαρμογής της στοχαστικής κλίσης κατάβασης, η οποία στην βιβλιογραφία αναφέρεται με πολλές παραλλαγές. Η κάθε παραλλαγή έχει διαφορετικές δυνατότητες και η επιλογή της εξαρτάται από το πρόβλημα που καλούμαστε να επιλύσουμε. Αναφορικά μερικές από τις πιο γνωστές είναι:

- Adam (Kingma & Lei, 2014)
- RMSProp (Hinton, Srivastava, & Swersky)
- AdaGrad (Duchi, Hazan, & Yoram, 2011)

Αυτές οι μέθοδοι είναι αρκετά περίπλοκες και η περεταίρω ανάλυσή τους ξεφεύγει από τα πλαίσια αυτής της εργασίας.

2.2 Στατιστική ανάλυση

Λόγω του πλήθους και του είδους των δεδομένων που χρειάστηκε να εξεταστούν καθώς και στατιστικής αβεβαιότητας κρίθηκε σκόπιμο να εξεταστούν στατιστικά μοντέλα για την πιο ολοκληρωμένη κατανόηση και περιγραφή του προβλήματος.

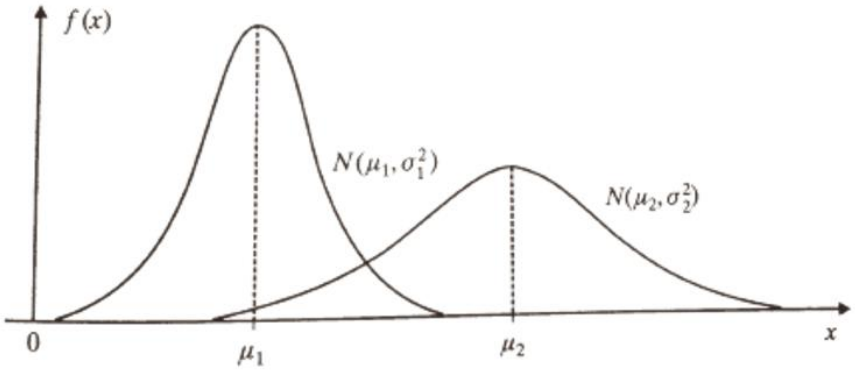
2.2.1 Κανονική κατανομή (Normal-Gaussian distribution)

Μια από τις πλέον χρήσιμες συνεχείς κατανομές στην θεωρία των πιθανοτήτων αλλά και στη στατιστική είναι η κανονική κατανομή. Η κατανομή είναι συνεχής και ορίζεται στο $(-\infty, +\infty)$. Οι παράμετροι που

την χαρακτηρίζουν είναι η μέση τιμή(mean) $-\infty < \mu < +\infty$ και η τυπική απόκλιση (standard deviation) $\sigma > 0$. Η Συνάρτηση Πυκνότητας Πιθανότητας(Probability Density Function) που την χαρακτηρίζει είναι:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \equiv N(x|\mu, \sigma^2), x \in R \tag{2.6}$$

Η παράμετρος μ καθορίζει τη θέση της κατανομής πάνω στον άξονα των x, γι' αυτό ονομάζεται και παράμετρος θέσης, ενώ η παράμετρος σ καθορίζει το πόσο απλώνεται η κατανομή πάνω στον άξονα των x, γι' αυτό ονομάζεται παράμετρος κλίμακας.



Διάγραμμα 2-5: Κανονικές κατανομές με μέσες τιμές μ_1, μ_2 και τυπικές αποκλίσεις σ_1, σ_2.

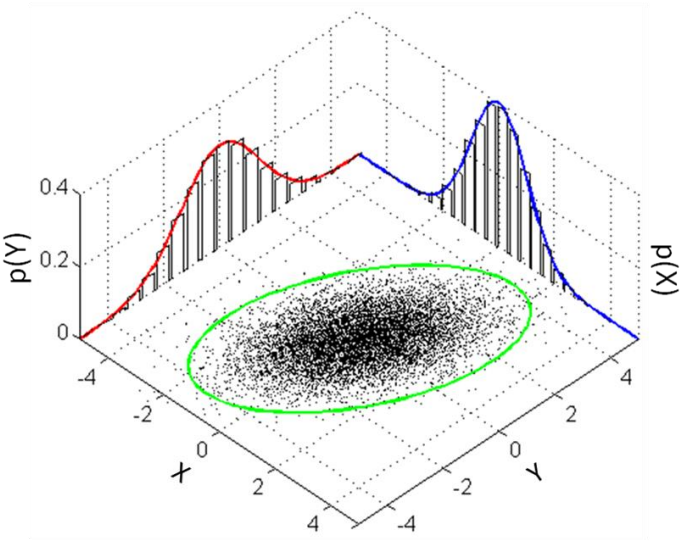
2.2.2 Πολυμεταβλητή κανονική κατανομή (Multivariate normal distribution)

Η πολυμεταβλητή κανονική κατανομή είναι μια γενίκευση της κλασικής κανονικής μεταβλητής σε μεγαλύτερες διαστάσεις. Η συνάρτηση πυκνότητας πιθανότητας που την χαρακτηρίζει είναι:

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} e^{\frac{-1}{2}(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})} \tag{2.7}$$

$$\text{όπου:} \vec{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix} \in R^n \kappa\alpha\iota\ \Sigma = \begin{bmatrix} {\sigma_1}^2 & \sigma_{12} & \cdots & \sigma_{1n} \\ \sigma_{21} & {\sigma_2}^2 & \cdots & \sigma_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{n1} & \sigma_{n2} & \cdots & {\sigma_n}^2 \end{bmatrix}$$

Είναι γνωστό από την γραμμική άλγεβρα, ότι υπάρχει ο αντίστροφος κάθε συμμετρικού και γνήσια θετικού πίνακα και είναι και αυτός επίσης συμμετρικός και γνήσια θετικός, αυτό σημαίνει ότι ο αναστραφός πίνακας θα υπάρχει πάντα . Η πολυμεταβλητή Κανονική κατανομή έχει ευρύτατες εφαρμογές. Αποτελεί το βασικότερο πιθανοθεωρητικό μοντέλο στην πολυμεταβλητή Ανάλυση, στην Στατιστική Συμπερασματολογία, στην Θεωρία Αποφάσεων και στην Πρόβλεψη.



Διάγραμμα 2-6: Πολυμεταβλητή κανονική κατανομή με συνδιασπορά.

Οι μη διαγώνιοι συντελεστές του μητρώου διασποράς αφορούν τις περιπτώσεις που έχουμε συσχέτιση μεταξύ των διαστάσεων. Στην περίπτωση που οι μη-διαγώνιοι συντελεστές δεν είναι μηδενικοί, δηλαδή θα έχουμε συσχέτιση των μεταβλητών, οι ισοϋψείς καμπύλες τις συνάρτησης πιθανότητας θα είναι ελλειψοειδούς μορφής. Στην περίπτωση που δεν υπάρχει συσχέτιση, οι ισοϋψείς καμπύλες είναι ομόκεντροι κύκλοι.

2.2.3 Μέγιστη Πιθανοφάνεια (Maximum likelihood)

Η συνάρτηση πιθανότητας ενός τυχαίου δείγματος $\vec{X} = (X_1, \dots, X_n)$ από πληθυσμό $\{\mathrm{X}, \Xi, p(x|\vec{\theta}), \vec{\theta} = (\theta_1, \dots, \theta_n) \in \Theta \subset R^m$ με $m \leq n$, είναι:

$$p(\vec{x}|\vec{\theta}) = \prod_{i=1}^{n} p(x_i|\vec{\theta}) \tag{2.8}$$

Όταν είναι γνωστές οι τιμές των τ.μ. $X_i, (i = 1, \dots, n)$, η συνάρτηση πιθανότητας $p(\vec{x}|\vec{\theta})$ μπορεί να θεωρηθεί ως συνάρτηση της παραμέτρου $\vec{\theta}$ με $\vec{x} = (x_1, \dots, x_n)$ σταθερό σημείο του χώρου $\Xi^n \subset R^n$. Έτσι έχουμε τη συνάρτηση πιθανοφάνειας.

$$L(\vec{\theta}|\vec{x}) = p(\vec{x}|\vec{\theta}) = \prod_{i=1}^{n} p(x_i|\vec{\theta}), \vec{\theta} \in \Theta \tag{2.9}$$

Η συνάρτηση πιθανοφάνειας $L(\vec{\theta}|\vec{x})$ εκφράζει το πόσο πιθανοφανείς, δηλαδή πόσο σύμφωνες με το συγκεκριμένο δείγμα $\vec{x}$, είναι οι διάφορες τιμές της παραμέτρου $\vec{\theta}$. Μεγιστοποιώντας την συνάρτηση πιθανοφάνειας ως προς το συντελεστή $\vec{\theta}$ βρίσκονται οι παράμετροι που χαρακτηρίζουν το δείγμα $\vec{x}$ βέλτιστα. Στο σημείο που θα έχουμε μεγιστοποίηση της πιθανοφάνειας οι μερικές παράγωγοι της $L(\vec{\theta}|\vec{x})$ ως προς $\theta_j = (j = 1, \dots, m)$, θα πρέπει να ισούνται με το μηδέν στο σημείο $\hat{\theta}$, εάν το $\hat{\theta}$ είναι η βέλτιστη παράμετρος. Έτσι έχουμε το σύστημα εξισώσεων:

$$\frac{\partial}{\partial \theta_j} L(\vec{\theta}|\vec{x}) = 0, (j = 1, \dots, m) \tag{2.10}$$

Παρόλα αυτά δεν συνηθίζεται η επίλυση με την παραπάνω μορφή. Η μορφή του γινομένου προσδίδει πολλά προβλήματα. Το πιο βασικό από αυτά είναι η παρουσία αριθμητικής αστάθειας. Παρόλο που δεν υπάρχει περίπτωση η πιθανότητα να είναι 0, αν η προσομοίωση γίνει με την βοήθεια υπολογιστικού μοντέλου (στην δική μας περίπτωση πακέτων στην γλώσσα προγραμματισμού Python), ο υπολογιστής στρογγυλοποιεί στο μηδέν τους παράγοντες που είναι κάτω από μια συγκεκριμένη τιμή. Η τιμή αυτή καθορίζεται από την ακρίβεια των αριθμών κινητής μεταβλητής (floating point). Για την αποφυγή των προβλημάτων αυτών γίνεται ανάλυση σε λογαριθμική κλίμακα, λογαριθμίζοντας την σχέση (2.9):

$$logL(\vec{\theta}|\vec{x}) = log \prod_{i=1}^{n} p(x_i|\vec{\theta}) \tag{2.11}$$

$$l(\vec{\theta}|\vec{x}) = logL(\vec{\theta}|\vec{x}) = \sum_{i=1}^{n} logp(x_i|\vec{\theta}) \tag{2.12}$$

Το l ονομάζεται λογαριθμική-πιθανοφάνεια (Log-Likelihood). Η διαδικασία εύρεσης της βέλτιστης παράμετρου $\hat{\theta}$ δεν αλλάζει, καθώς η συνάρτηση λογαρίθμου είναι γνήσια μονότονη και αύξουσα.

Η παραπάνω μέθοδος είναι ευρέως γνωστή για την αποτελεσματικότητά της για τις περιπτώσεις όπου το $\vec{\theta}$ είναι μικρής διάστασης (μικρός αριθμός παραμέτρων προς βελτιστοποίηση) για παράδειγμα . Στην περίπτωση που το $\vec{\theta}$ είναι μεγάλης διάστασης, η αναλυτική επίλυση ενός μεγάλου μη-γραμμικού συστήματος μπορεί να είναι και αδύνατη. Για τον λόγο αυτό θα πρέπει να επικαλεστούμε αριθμητικές μεθόδους για την εύρεση των βέλτιστων παραμέτρων.

2.2.4 Αλγόριθμος Προσδοκίας-Μεγιστοποίησης (Expectation-Maximization)

Στην επιστήμη της στατιστικής ο αλγόριθμος Προσδοκίας-Μεγιστοποίησης (E-M algorithm) είναι μια επαναληπτική μέθοδος για τη εύρεση του τοπικού μέγιστου της συνάρτησης πιθανοφάνειας. Η επαναληπτική διαδικασία χωρίζεται σε δυο μέρη, το βήμα Προσδοκίας και το βήμα Μεγιστοποίησης(E-step και M-step) τα οποία επαναλαμβάνονται μέχρι να επιτευχθεί σύγκλιση.

όπως σε κάθε επαναληπτική μέθοδο, το πρώτο βήμα είναι η επιλογή των αρχικών παραμέτρων. Όπως και στην μέθοδο κλίσης κατάβασης η σωστή επιλογή των αρχικών παραμέτρων επηρεάζει κατά πολύ το αποτέλεσμα της σύγκλισης καθώς και αν το ακρότατο είναι το μέγιστο. Η επιστημονική βιβλιογραφία προτείνει ο αλγόριθμος να τρέξει πολλαπλές φορές με τυχαίες αρχικές τιμές ή με την βοήθεια της μεθοδολογίας K-means (πραγματοποιώντας αρχικό clustering) από τις δοκιμές, επιλέγοντας τις παραμέτρους με την μέγιστη πιθανοφάνεια. (MacKay, 2003)

Το βήμα Προσδοκίας(E-step) είναι το πρώτο βήμα της επαναληπτικής μεθόδου. Στο βήμα αυτό υπολογίζεται η πιθανότητα κάθε στοιχείου να ανήκει στην εκάστοτε κατανομή.

$$p(j|i) = \frac{f_i\left(x^{(i)}, \overrightarrow{\theta_J}\right)}{p\left(x^{(i)}, \vec{\theta}\right)} \tag{2.13}$$

όπου:

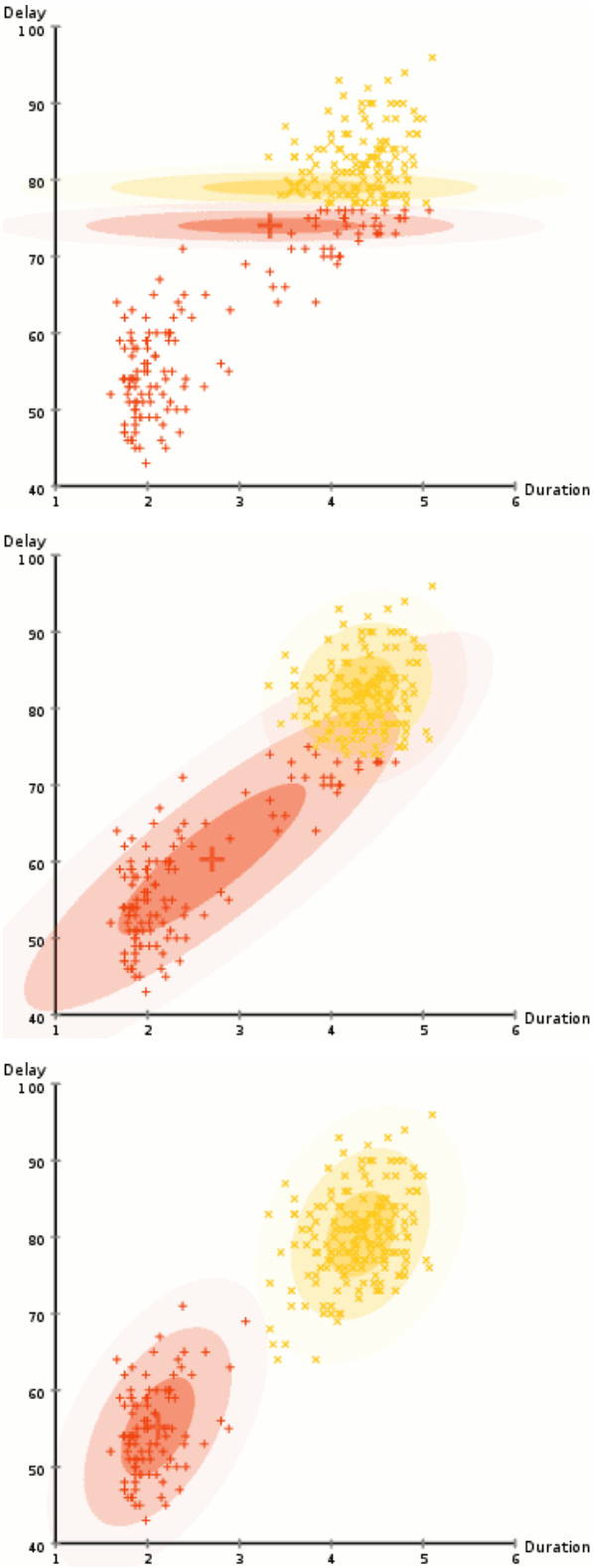
- $p(j|i)$ είναι η πιθανότητα το στοιχείο i να ανήκει στην κατανομή(υποομάδα) j.
- $f_i\left(x^{(i)}, \overrightarrow{\theta_J}\right)$ είναι η σ.π.π. της κατανομής (υποομάδας).
- $p\left(x^{(i)}, \vec{\theta}\right) = \sum_{j=1}^{J} f_i\left(x^{(i)}, \overrightarrow{\theta_J}\right)$.

Το βήμα Μεγιστοποίησης (M-step) είναι η διαδικασία όπου βελτιστοποιούνται οι παράμετροι σύμφωνα με το προηγούμενο βήμα Προσδοκίας(E-step).Οι τύποι για την βελτίωση των παραμέτρων προκύπτουν από τον μηδενισμό της μερικής παραγωγού της συνάρτησης πιθανοφάνειας ως προς την κάθε παράμετρο.

$$\frac{\partial}{\partial \theta_j} P\left(\vec{\theta}\middle|\vec{x}\right) = 0, (j = 1, \dots, m) \tag{2.14}$$

Στη συνέχεια θα γίνει εκτενής και αναλυτική απόδειξη για συγκεκριμένο στατιστικό μοντέλο για την κατανόηση των βημάτων Προσδοκίας-Μεγιστοποίησης.

Η επαναληπτική διαδικασία πραγματοποιείται έως ότου η διαφορά της πιθανοφάνειας μεταξύ δύο διαδοχικών επαναλήψεων είναι μικρότερη από την τιμή που επιθυμούμε(κριτήριο σύγκλισης).



Διάγραμμα 2-7: Διαδοχικά βήματα του ΕΜ αλγόριθμου μέχρι την σύγκλιση

Στο Διάγραμμα 2-7 βλέπουμε την διαδικασία σύγκλισης του αλγορίθμου καθώς προσπαθεί να προσαρμόσει τις δύο κατανομές στα δεδομένα. Έπειτα μπορούμε να χωρίσουμε τα δεδομένα ανάλογα με την κατανομή που τα χαρακτηρίζει.

Παρόλο που ο αλγόριθμός πάντα θα συγκλίνει σε τοπικά μέγιστα της πιθανοφάνειας, δεν προτείνεται για την βελτιστοποίηση μοντέλων μεγάλων διαστάσεων. Η αύξηση των διαστάσεων αυξάνει των αριθμό των τοπικών ελαχίστων που μπορεί εγκλωβιστεί ο αλγόριθμος οδηγώντας σε μειωμένη απόδοση. Ακόμα, με την αύξηση των διαστάσεων αυξάνεται ο αριθμός τον παραμέτρων που βελτιστοποιούμε αυξάνοντας την υπολογιστική πολυπλοκότητα, δηλαδή τον χρόνο εκτέλεση της εκπαίδευσης (training).

2.3 Κριτήρια ακρίβειας πρόβλεψης

Κύριος στόχος της συγκεκριμένης εργασίας είναι η σύγκριση των μεθόδων μεταξύ τους. Για να γίνει κάτι τέτοιο θα πρέπει να οριστεί ένα κοινό μέτρο σύγκρισης μεταξύ των μεθόδων. Εκ πρώτης όψεως η συνάρτηση της απόλυτης διαφοράς της πραγματικής τιμής με την τιμή εκτίμησης θα ήταν ικανή για το σκοπό αυτό. Ο στόχος του κάθε μοντέλου είναι να μηδενίσει το σφάλμα (πρόβλημα ελαχιστοποίησης) άρα όσο μικρότερη είναι η συνάρτηση τόσο πιο αποτελεσματική θα είναι η μέθοδος. Στις δύο περιπτώσεις που έχουμε π.χ. πραγματική τιμή 10 και 100 και πρόβλεψη 20 και 110 αντίστοιχα, η παραπάνω συνάρτηση θα είχε υπολογίσει το ίδιο σφάλμα ενώ στην πρώτη περίπτωση έχει κάνει συγκριτικά μια πάρα πολύ κακή πρόβλεψη. Για αυτό τον λόγο η επιλογή αυτή της συνάρτησης θα πρέπει να λαμβάνει υπόψη την διαφορά της κλίμακας.

2.3.1 Μέσο Απόλυτο Ποσοστιαίο Σφάλμα (Mean Absolute Percentage Error-MAPE)

Το μέσο απόλυτο ποσοστιαίο σφάλμα (MAPE) γνωστό και ως μέση απόλυτη ποσοστιαία απόκλιση είναι συνάρτηση σφάλματος για την αξιολόγησή μοντέλων προβλέψεων. Ο τύπος της συνάρτησης είναι:

$$MAPE = \frac{100\%}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right| \tag{2.15}$$

όπου:

- A_t: Η πραγματική τιμή (*Actual Value*)
- F_t: Η εκτίμηση (*Forecast Value*)

Το συνολικό σφάλμα προστίθεται για κάθε χρονοσειρά και διαιρείται από το πλήθος των προβλέψεων υπολογίζοντας το μέσο όρο τον σφαλμάτων των προβλέψεων.

Το μέσο απόλυτο ποσοστό σφάλματος χρησιμοποιείται συνήθως ως συνάρτηση απώλειας για προβλήματα παλινδρόμησης και στην αξιολόγηση μοντέλων, λόγω της πολύ διαισθητικής ερμηνείας του ως προς το σχετικό σφάλμα. Αν και η έννοια του μέσου απόλυτου ποσοστού σφάλματος ακούγεται πολύ

απλή και πειστική, έχει σημαντικά μειονεκτήματα στην πρακτική εφαρμογή και υπάρχουν πολλές μελέτες σχετικά με ελλείψεις και παραπλανητικά αποτελέσματα. (Tofallis, 2015)

Η συγκεκριμένη μεθοδολογία δεν μπορεί να λειτουργήσει αν ή πραγματική τιμή είναι ίση με το μηδέν $A_t = 0$ (δεν είναι καθόλου απίθανο σε μετρήσεις καταναλώσεων), καθώς δεν γίνεται ο παρονομαστής να ισούται με το μηδέν. Ένα ακόμα πρόβλημα είναι ότι για μικρές προβλέψεις το ποσοστό σφάλματος δεν μπορεί να ξεπεράσει το 100%, αλλά για πολύ μεγάλες προβλέψεις δεν υπάρχει άνω όριο του ποσοστιαίου σφάλματος και αυτό μπορεί να δημιουργήσει μια ψευδαίσθηση ανομοιομορφίας.

Στην περίπτωση που έχουμε αυτά τα προβλήματα, μπορούμε να τα αποφύγουμε με πιο περίπλοκες συναρτήσεις οι οποίες είναι:

i. Μέσο απόλυτο σφάλμα κλίμακας - Mean Absolute Scaled Error (MASE).
ii. Συμμετρικό μέσο απόλυτο ποσοστό σφάλματος-Symmetric Mean Absolute Percentage Error (sMAPE).
iii. Μέση ακρίβεια κατεύθυνσης- Mean Directional Accuracy (MDA).

Συμφώνα με τα δεδομένα που διαθέτουμε και τις προβλέψεις που θέλουμε στην παρούσα κατάσταση η συνάρτηση MAPE λειτουργεί χωρίς κανένα θέμα, αφού δεν έχουμε παρουσία μηδενικών παροχών. Παρόλα αυτά για την ανάλυση πραγματικών δεδομένων όπου είναι σίγουρο ότι θα υπάρχουν πελάτες με μηδενική κατανάλωση θα έπρεπε ή με κάποιο τρόπο να φιλτραριστούν από τους άλλους πελάτες και να γίνει εκτίμηση με άλλη μεθοδολογία ή να χρησιμοποιήσουμε τις εναλλακτικές συναρτήσεις σφάλματος που αναφέρθηκαν ανωτέρω.

2.3.2 Μέγιστη διαφορά (Maximum Difference)

Άλλη μια συνάρτηση σφάλματος που χρησιμοποιήθηκε είναι η μέγιστη διαφορά. Ό μαθηματικός τύπος που την χαρακτηρίζει είναι :

$$MD = max\ (A_t - F_t) \tag{2.16}$$

όπου:

- A_t: Η πραγματική τιμή (*Actual Value*)
- F_t: Η εκτίμηση (*Forecast Value*)

Ουσιαστικά είναι η μέγιστη διαφορά (απόκλιση) μεταξύ των προβλέψεων και των πραγματικών τιμών από όλες τις δοκιμές που έχουν πραγματοποιηθεί. Ο δείκτης αυτός μας βοηθάει να κατανοήσουμε την μέγιστη απόκλιση που μπορεί να παρουσιάζει μια μεθοδολογία. Σύμφωνα με αυτά που αναλύσαμε προηγουμένως θα ήταν λογικό να συμπεράνουμε ότι από την στιγμή που δεν λαμβάνουμε υπόψη την κλίμακα, το αποτέλεσμα αυτής της μεθόδου θα επηρεάζεται κυρίως από προβλέψεις μεγάλων καταναλωτών καθώς η απόκλιση θα είναι μεγάλη. Για αυτό τον λόγο δεν θα αποτελέσει κριτήριο επιλογής μεθόδου, αλλά ένα μέτρο σύγκρισης όπου το μέσο ποσοστιαίο σφάλμα είναι πολύ κοντινό.

2.3.3 Ολική διαφορά (Total Difference)

Η λογική πίσω από την επιλογή της ολικής διαφοράς σαν συνάρτηση κόστους προκύπτει καθαρά από την πλευρά του παρόχου. Ο δείκτης αυτός εντοπίζει αν η μεθοδολογία υποεκτιμά ή υπερεκτιμά τις καταναλώσεις στο σύνολο. Ο τύπος της είναι:

$$TD = \sum_{t=1}^{n} A_t - F_t \tag{2.17}$$

όπου:

- A_t: Η πραγματική τιμή (Actual Value)
- F_t: Η εκτίμηση (Forecast Value)

Αν ο δείκτης TD είναι θετικός $TD > 0$, τότε ο μοντέλο υποεκτιμά τις καταναλώσεις κατά μέσο όρο. Όμοια όταν είναι αρνητικός $TD < 0$ το μοντέλο υπερεκτιμά τις καταναλώσεις.

Για την ομαλότητα των χρηματοροών του παρόχου θα θέλαμε όσο γίνεται ο δείκτης TD να πλησιάζει το μηδέν. Θα πρέπει να σημειωθεί πως η απόδοση της ολικής διαφοράς δεν έχει καμία σχέση με την απόδοση του Μέσου Απόλυτου Ποσοστιαίου Σφάλματος καθώς είναι δυνατό να έχουμε μηδενική ολική διαφορά αλλά μη μηδενικό Μέσο Απόλυτο Ποσοστιαίο Σφάλμα.

2.3.4 Μπεϋζιανό κριτήριο πληροφορίας (BIC)

Στη επιστήμη της στατιστικής, το Μπεϋζιανό κριτήριο πληροφορίας (Bayesian Information Criterion-BIC) ή το κριτήριο πληροφοριών Schwarz (επίσης SIC, SBC, SBIC) είναι ένα κριτήριο για την επιλογή μοντέλου μεταξύ ενός πεπερασμένου συνόλου μοντέλων. Το Μπεϋζιανό κριτήριο πληροφορίας ορίζεται ως:

$$BIC = kln(n) - 2ln(\hat{L}) \tag{2.17}$$

όπου:

- $\hat{L}$ είναι η μέγιστη πιθανοφάνεια του μοντέλου.
- k είναι ο αριθμός των ανεξάρτητων παραμέτρων.
- n είναι ο αριθμός των συνολικών δεδομένων.

Γενικά μοντέλα με μικρότερο αριθμό BIC προτιμώνται. Όπως μπορούμε να δούμε και στην σχέση παραπάνω το κριτήριο επιβραβεύει μοντέλα με μεγαλύτερη πιθανοφάνεια, αλλά λειτουργεί ανάποδα στην περίπτωση που έχουμε πολλούς συντελεστές στο μοντέλο μας. Είναι ένας τρόπος για να ελέγξουμε αν το στατιστικό μοντέλο έχει προβλήματα υπερβολικής προσαρμογής στο σύνολο δεδομένων εκπαίδευσης (Overfit).

3. Υπολογιστικά πακέτα

3.1 Ανάλυση δεδομένων σε προγραμματιστικό περιβάλλον

Λόγω του είδους των μοντέλων (περίπλοκα μαθηματικά μοντέλα) και του πλήθους των δεδομένων και των απαιτούμενων υπολογισμών η ανάλυση έγινε σε προγραμματιστικό περιβάλλον και πιο συγκεκριμένα σε γλώσσα προγραμματισμού Python.

3.2 Γλώσσα προγραμματισμού Python

Η Python είναι διερμηνευόμενη (interpreted), γενικού σκοπού (general-purpose), υψηλού επιπέδου και ανοιχτού κώδικα γλώσσα προγραμματισμού. Είναι δυναμική γλώσσα προγραμματισμού (dynamically typed) και υποστηρίζει τόσο διαδικαστικό (procedural programming), όσο και αντικειμενοστραφή προγραμματισμό (object-oriented programming).

Ο δημιουργός της Python είναι ο Ολλανδός Guido van Rossum και αναπτύχθηκε στο ερευνητικό κέντρο Centrum Wiskunde & Informatica (CWI) το 1989. Εμπνευσμένη από την γλώσσα προγραμματισμού ABC αρχικά η Python ήταν γλώσσα σεναρίων. Στις 16 Οκτωβρίου του 2000 κυκλοφόρησε η Python 2.0 και στις 3 Δεκεμβρίου του ίδιου έτους η έκδοση 3.0. Η python 3.0 κλήθηκε να διορθώσει τα λάθη του υπήρχαν στις προηγούμενες εκδόσεις. Ιστορικά ήταν η πρώτη γλώσσα που ήταν συμβατή με προηγούμενες εκδόσεις για αυτό τον λόγο πολλά καινούργια χαρακτηριστικά έχουν μεταφερθεί στις εκδόσεις 2.6 και 2.7. Η πιο πρόσφατη έκδοση την στιγμή που συντάσσεται η εργασία αυτή είναι η έκδοση 3.9.2 και κυκλοφόρησε στις 26 Φεβρουάριου 2021.

Το κύριο πλεονέκτημά της είναι η αναγνωσιμότητα του κώδικά της και η ευκολία στην χρήση της. Ο τρόπος σύνταξης επιτρέπει στους προγραμματιστές να εκφράσουν έννοιες σε λιγότερες γραμμές κώδικα από ό,τι θα ήταν δυνατό σε γλώσσες όπως η C και η Java. Όμως μειονεκτεί στο γεγονός ότι είναι διερμηνευόμενη και είναι πιο αργή από τις μεταγλωττιζόμενες (compiled) όπως η C. Για αυτό τον λόγο είναι από τίς πιο αργές γλώσσες προγραμματισμού και για αυτό δεν είναι κατάλληλη για γραφή λειτουργικών συστημάτων. Την Python μπορεί κάποιος να την εγκαταστήσει από την ιστοσελίδα (Python, 2022)

3.3 Πλεονεκτήματα της γλώσσας προγραμματισμού Python

Η python είναι μια γλώσσα υψηλού επιπέδου (High-level programming language) κάνοντας την σύνταξή της πολύ εύκολη όπως και την κατανόηση της. Η εύκολη σύνταξη επιτρέπει στον μηχανικό να συγκεντρωθεί στο πρόβλημα και να μην χρειάζεται πολύ χρόνο για την κατανόηση της προγραμματιστικής θεωρίας. Αυτό, σε συνδυασμό με την δημοτικότητα της, με μεγάλη κοινότητα υποστήριξης καθιστούν την Python την ιδανική επιλογή για τους χρήστες που δεν έχουν εμπειρία. Η εύρεση σφαλμάτων γίνεται εύκολα καθώς λόγω της δημοφιλίας της είναι πολύ πιθανό κάποιος στην

κοινότητα να έχει επιλύσει κάποιο παρόμοιο πρόβλημα. Ο τρόπος σύνταξης του προγράμματος γίνεται με την βοήθεια συναρτήσεων (functions) και κλάσεων (classes) όπου καλούνται από το κεντρικό πρόγραμμα (script). Ο προγραμματιστής μπορεί εύκολα να συντάξει τις συναρτήσεις για να προσαρμόσει την λειτουργία του προγράμματος σύμφωνα με τις απαιτήσεις του. Τέλος η δυνατότητα εκτέλεσης του ίδιου ακριβώς προγράμματος σε διαφορετικές πλατφόρμες (portability) έχει ενώσει την κοινότητα στην δημιουργία πάρα πολλών κοινόχρηστων ανοιχτού κώδικα (opensource) πακέτων/συναρτήσεων (modules) για την απλούστευση τον προβλημάτων. Το pypi είναι μια βιβλιοθήκη τέτοιων πακέτων, την δεδομένη στιγμή φιλοξενεί πάνω από τριακόσες πενήντα χιλιάδες βιβλιοθήκες. Ο καθένας μπορεί να αναπτύξει τέτοια πακέτα, να τα δημοσιεύσει και να δεχθεί βελτιώσεις από άλλους χρήστες. Στα πλαίσια αυτής της εργασίας χρησιμοποιήθηκε ο ευρέως διαδεδομένος διαχειριστής πακέτων (package manager) conda, όπου περιέχει σχεδόν όλα τα πακέτα που χρειάζονται για αυτή την μελέτη.

3.3 Πακέτα που χρησιμοποιήθηκαν

NumPy

Με την βοήθεια της βιβλιοθήκης NumPy έχουμε δυνατότητα για την δημιουργία και την επεξεργασία δεδομένων μητρωικής μορφής (μορφής πίνακα). Τα δεδομένα αποθηκεύονται στην μνήμη με την μορφή κλάσης (class) με το όνομα array. Η βιβλιοθήκη αυτή παρέχει παρόμοιες λειτουργίες με την γλώσσα προγραμματισμού MATLAB καθώς η διαχείριση των πινάκων και πράξεων γραμμικής άλγεβρας είναι παρόμοιες. Πλέον η μορφή δεδομένων NumPy είναι τόσο διαδεδομένη που έχει υλοποιηθεί για γλώσσες προγραμματισμού πέρα από την Python. Τα στοιχεία που απαρτίζουν τα μητρώα είναι μορφής κινητής υποδιαστολής ακρίβειας 64-bit (διπλής ακρίβειας- double precision). Η επίσημη ιστοσελίδα του πακέτου είναι: (Numpy, 2022)

Pandas

Το όνομα της βιβλιοθήκης βγαίνει από τους όρους “panel data” και είναι ιδανική για την διαχείριση δεδομένων, όχι μόνο χρονοσειρών όπως στην παρούσα εργασία. Η βιβλιοθήκη έχει την δυνατότητα διαβάσματος δεδομένων από άλλες μορφές όπως: Excel, SQL και άλλα. Με δυνατότητες όπως φιλτράρισμα δεδομένων (data filtration) και διαχείριση ελλειπόντων δεδομένων (missing data) η διαδικασία της προεπεξεργασίας των δεδομένων γίνεται πιο εύκολη από ποτέ. Η επίσημη ιστοσελίδα του πακέτου είναι: (Pandas, 2022)

Statsmodels

Το πακέτο Statsmodels περιέχει στατιστικά μοντέλα καθώς και διαδικασίες στατιστικής ανάλυσης. Στην μελέτη επιλέχθηκαν δύο μοντέλα (ARIMA,SARIMA) από αυτή την βιβλιοθήκη. Η εκτενής βιβλιογραφία και τα αμέτρητα παραδείγματα καθιστούν την χρήση του πολύ απλή. Τα δεδομένα εισόδου είναι συμβατά με την μορφή NumPy. Ο μηχανισμός λειτουργίας των μοντέλων ARIMA και SARIMA θα αναλυθεί στα επόμενα κεφάλαια. Η επίσημη ιστοσελίδα του πακέτου είναι: (statsmodels, 2022)

Sklearn

Παρόμοια με την βιβλιοθήκη statsmodels η βιβλιοθήκη sklearn περιέχει μοντέλα μηχανικής μάθησης για προβλήματα ταξινόμησης (classification), παρεμβολής (regression) και ομαδοποίησης (clustering) όπως Μηχανές Διανυσμάτων Υποστήριξης (SVM), Τυχαίο Δάσος (Random Forest), κ-Μέσοι (k-Means)

και Μίξη Γκαουσιανών Κατανομών (Gaussian mixture models). Η βιβλιοθήκη χρησιμοποιεί δομή NumPy για τα δεδομένα εισόδου καθώς και για την πραγματοποίηση των αλγεβρικών πράξεων. Η επίσημη ιστοσελίδα του πακέτου είναι: (Scikit-Learn, 2022)

TensorFlow

Η βιβλιοθήκη αυτή έχει πολλές δυνατότητες αλλά εξειδικεύεται στην δημιουργία και στην εκπαίδευση νευρωνικών δικτύων (Neural Networks). Το TensorFlow δημιουργήθηκε από την ομάδα έρευνας Google Brain η οποία ανήκει στην Google. Κατά την λειτουργία των μοντέλων τα δεδομένα έχουν μορφή τανυστή, από εκεί είναι εμπνευσμένο το όνομα του πακέτου. Η βιβλιοθήκη περιέχει και πιο περίπλοκα μοντέλα όπως Επανατροφοδοτούμενα Νευρωνικα Δίκτυα (Recurrent Neural Networks-RNN) και Συνελικτικά Νευρωνικά Δίκτυα (convolutional neural networks-CNN). Το πακέτο είναι συμβατό για την χρήση πυρήνων cuda, δηλαδή την αξιοποίηση της κάρτας γραφικών για την ταχύτερη εκπαίδευση των μοντέλων. Η διαφορά στην ταχύτητα εκπαίδευσης με την βοήθεια της κάρτας γραφικών μπορεί να φτάσει μέχρι και έξι (6) φορές πιο γρήγορα σε σχέση με την χρήση του επεξεργαστή. Στην αγορά κυκλοφορούν κάρτες με αυξημένους πυρήνες cuda ειδικά για αυτήν την δουλεία: TPU (Tensor Processing Unit). Στην περίπτωση που ο υπολογιστής έχει κάρτα γραφικών της εταιρίας NVIDIA με μια μικρή παραμετροποίηση στον κώδικα μπορεί να επιτύχει πολύ μεγαλύτερες ταχύτητες εκτέλεσης. Η επίσημη ιστοσελίδα του πακέτου είναι: (TensorFlow, 2022)

Numba

Όπως αναφέρθηκε και παραπάνω η Python είναι διερμηνευόμενη γλώσσα προγραμματισμού. Στην περίπτωση που ο κώδικάς δεν είναι έτοιμος από κάποιο πακέτο, δηλαδή δεν έχει βελτιστοποιηθεί όσο θα μπορούσε, μπορεί να κάνει πολύ μεγάλο χρονικό διάστημα για να τρέξει. Την λύση σε αυτό το πρόβληματην δίνει το πακέτο Numba και ο JIT compiler (Just In Time Compiler). Με την βοήθεια του JIT ο κώδικας υπό μορφή Python μεταφράζεται σε γλώσσα μηχανής. Η μετάφραση αυτή θέλει λίγα δευτερόλεπτα για να πραγματοποιηθεί για πρώτη φορά, από την στιγμή που ένα πρόγραμμα τρέχει μια φορά αποθηκεύεται στο υπολογιστή η μεταφρασμένη εκδοχή του κώδικα σε γλώσσα μηχανής. Επειδή η γλώσσα μηχανής δεν έχει τόσες πολλές δυνατότητες όσο η Python, δεν μπορούμε να χρησιμοποιήσουμε την βιβλιοθήκη numba για οποιοδήποτε script. Αν ο κώδικας περιέχει κλάσεις η τύπο μεταβλητής λίστα (list) δεν θα μπορέσει να γίνει μετάφραση σε γλώσσα μηχανής. Γενικά το κύριο σκεπτικό είναι ο κώδικας να μην περιέχει άλλες βιβλιοθήκες καθώς συνήθως δεν ξέρουμε πώς τρέχουν και θα είναι αδύνατον να βρούμε το πρόβλημα. Αντιθέτως επιθυμούμε η σύνταξη του κώδικα να γίνει όσο περισσότερο γίνεται σε μορφή NumPy, δηλαδή μητρωικά. Οι μητρωικές πράξεις μπορούν να γίνουν παράλληλα μειώνοντας την ταχύτητα που απαιτείται για να ολοκληρωθεί το πρόγραμμα. Σε γενικές γραμμές η μεθοδολογία αυτή μπορεί να εκτελέσει μέχρι και είκοσι (20) φορές πιο γρήγορα το πρόγραμμα σε σχέση με την κλασσική διαδικασία. Τέλος η βιβλιοθήκη αυτή όπως και το TensorFlow μπορούν να αξιοποιήσουν την κάρτα γραφικών για την ελαχιστοποίηση του χρόνου εκτέλεσης. Η επίσημη ιστοσελίδα του πακέτου είναι: (Numba, 2022)

Matplotlib

Για την σχεδίαση των διαγραμμάτων χρησιμοποιήθηκε η βιβλιοθήκη Matplotlib. Ο τρόπος παρουσίασης καθώς και ο τρόπος σύνταξης θυμίζει πολύ την MATLAB. Η βιβλιοθήκη περιέχει πολλά είδη γραφημάτων

όπως: 2D-3D διαγράμματα, Ιστογράμματα(Histogram), Πολικά διαγράμματα (Polar plot) και άλλα πολλά.Η επίσημη ιστοσελίδα του πακέτου είναι: (Matplotlib, 2022)

Darts

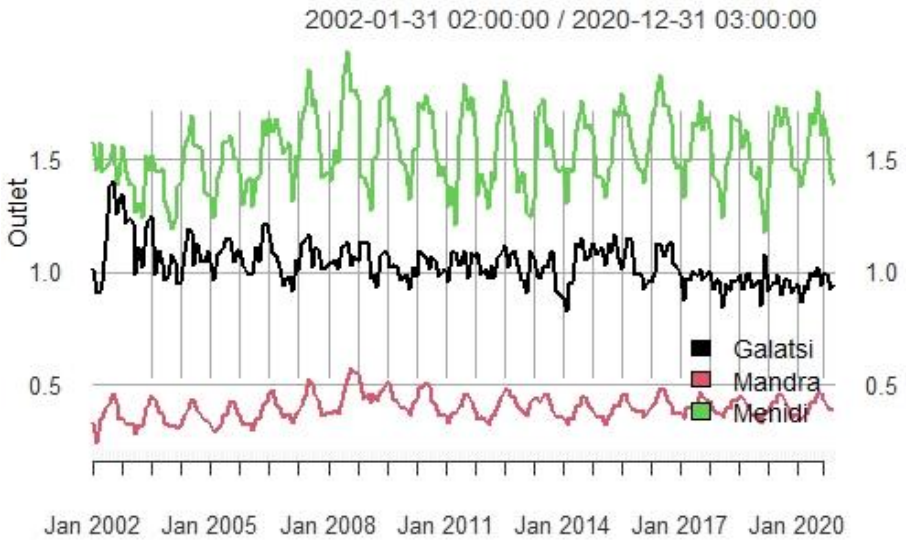
Η βιβλιοθήκη Darts έχει φτιαχτεί για τις ανάγκες χειρισμού χρονοσειρών. Περιέχει πληθώρα μοντέλων πρόβλεψης όπως ARIMA μέχρι και Νευρωνικά δίκτυα, αλλά και μοντέλα προεπεξεργασίας των δεδομένων όπως Γκαουσιανή διαδικασία (Gaussian Process), Φίλτρο Kalman (Kalman Filter) και άλλα. Ο τρόπος σύνταξης είναι πολύ εύκολος αφού για όλα τα μοντέλα που περιέχονται μπορούν να εκπαιδευτούν με την ίδια εντολή .fit() και η παραγωγή τον προβλέψεων με την εντολή .predict(). Τέλος τα μοντέλα της βιβλιοθήκης Darts υποστηρίζουν και πολυμεταβλητές χρονοσειρές ως δεδομένα εισόδου. Η επίσημη ιστοσελίδα του πακέτου είναι: (Darts, 2022)

Κάποια πακέτα όπως το Tensorflow χρησιμοποιούν άλλα πακέτα στον κώδικά τους (dependencies), στην συγκεκριμένη περίπτωση το TensorFlow θέλει το NumPy για να λειτουργήσει. Το πρόβλημα που μπορεί να αντιμετωπίσει κάποιος είναι ότι αυτές οι βιβλιοθήκες πρέπει να είναι συμβατές μεταξύ τους. Στο παραπάνω παράδειγμα η βιβλιοθήκη Tensorflow 2.6.0 δεν μπορεί να δουλέψει με μεγαλύτερες εκδόσεις NumPy 1.19.5. Για τον λόγο αυτό θέλει προσοχή στο στήσιμο του περιβάλλοντος προγραμματισμού που στήνουμε. Όλα τα πακέτα εγκαταστάθηκαν με την βοήθεια του package management system (pip) με την εντολή: -pip install <name_package> ή -conda install <name_package> από την γραμμή εντολών του conda όπου <name_package> το όνομα του πακέτου που θέλουμε να προσθέσουμε στο προγραμματιστικό περιβάλλον.

4. Δεδομένα

4.1 Συνθετικά δεδομένα

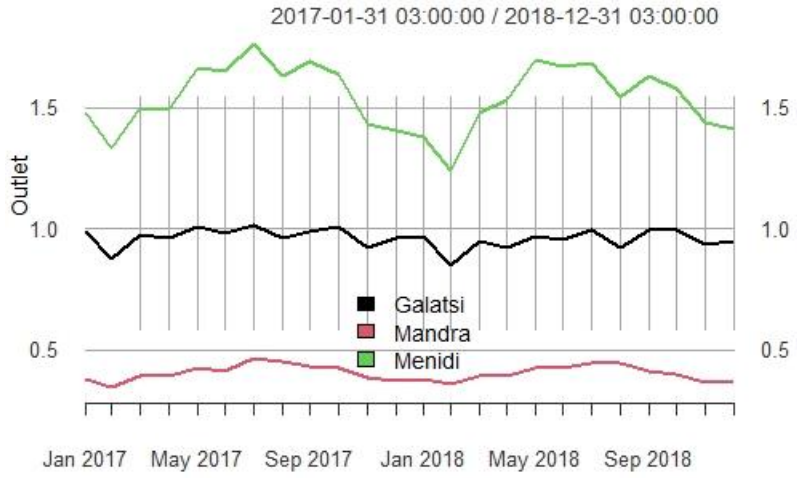
Με σκοπό να εξαχθούν ασφαλή συμπεράσματα για την προγνωστική ικανότητα των διαφορετικών μεθόδων που μελετώνται, έγινε παραγωγή και χρήση ενός μεγάλου σετ συνθετικών δεδομένων μηνιαίων καταναλώσεων νερού σε επίπεδο οικίας, δεδομένης της έλλειψης πραγματικών δεδομένων. Λόγω της έλλειψης αυτής, εκμεταλλευτήκαμε την διαθεσιμότητα μεγάλου μήκους χρονοσειρών μηνιαίων εκροών από 3 διυλιστήρια παραγωγής πόσιμου νερού (Γαλάτσι, Μάνδρα και Μενίδι) που εξυπηρετούν την περιοχή της Αττικής (Διάγραμμα 4-1). Συγκεκριμένα, για την παραγωγή συνθετικών οικιακών ζητήσεων νερού έγινε η υπόθεση ότι η μηνιαία ζήτηση νερού σε επίπεδο σπιτιού ακολουθεί την ετήσια περιοδικότητα της εκροής νερού από τα διυλιστήρια νερού.



Διάγραμμα 4-1: Εκροές διυλιστηρίων Γαλατσίου, Μάνδρας και Μενιδίου.

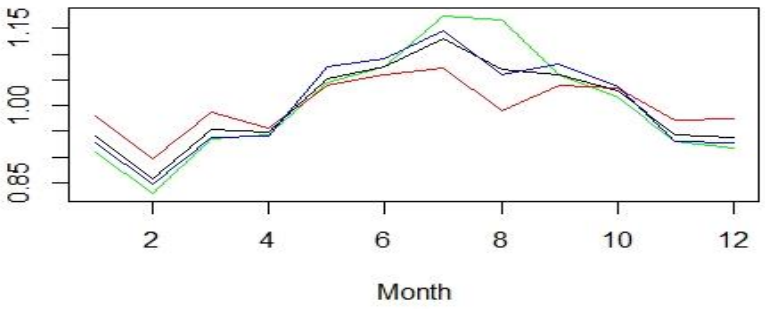
.

Το προφίλ μεταβολής της εκροής στα 3 υπό μελέτη διυλιστήρια (τυποποιημένο ως προς την αντίστοιχη μέση τιμή τους), από μήνα σε μήνα, φαίνεται στην εικόνα που ακολουθεί, για τα έτη 2017 - 2018.



Διάγραμμα 4-2: Τυποποιημένες ως προς τον μέσο όρο εκροές διυλιστηρίων Γαλατσίου, Μάνδρας και Μενιδίου.

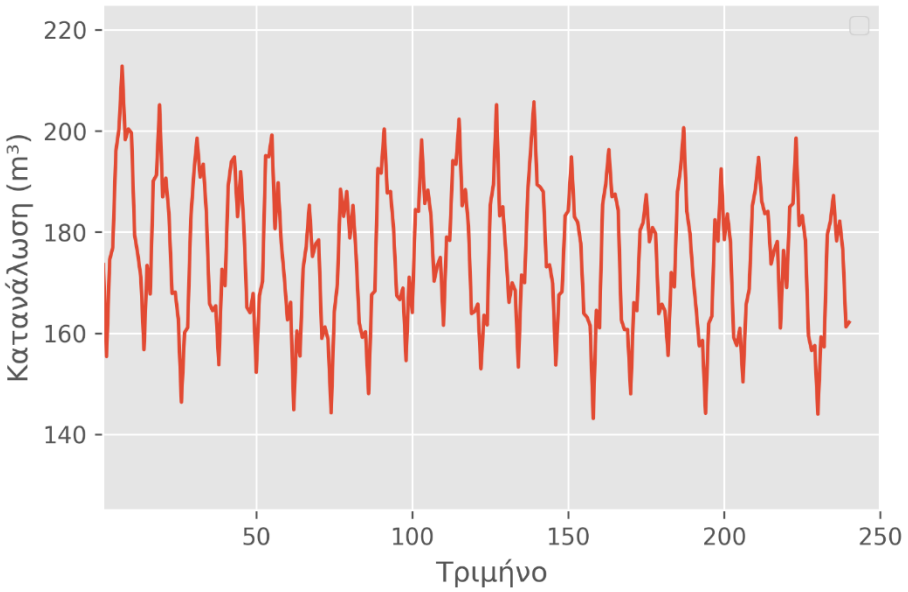
Για κάθε διυλιστήριο προέκυψε ένα τυποποιημένο προφίλ εκροής ανά μήνα, ενώ τα προφίλ τυποποιήθηκαν περαιτέρω για να προκύψει ένα κοινό προφίλ μεταβολής ανά μήνα.



Διάγραμμα 4-3: Πλήρως τυποποιημένες εκροές διυλιστηρίων Γαλατσίου, Μάνδρας και Μενιδίου.

Το ενιαίο προφίλ μεταβολής χρησιμοποιήθηκε για την τυποποίηση των χρονοσειρών από τα 3 διυλιστήρια και την αφαίρεση του ετήσιου κύκλου που εμφανίζει η μέση εκροή. Εν συνεχεία, έγινε στοχαστική προσομοίωση για την παραγωγή συνθετικών τυποποιημένων χρονοσειρών μέσω του πακέτου, γλώσσας προγραμματισμού R, anySim (Tsoukalas, Kossieris, & Makropoulos, 2020). Πιο συγκεκριμένα, η τυποποιημένη χρονοσειρά αντιμετωπίστηκε ως στάσιμη και για αυτό το λόγο η προσομοίωση σε μηνιαίο βήμα έγινε μέσω ενός στάσιμου γραμμικού μοντέλου αυτό-παλινδρόμησης, με χρήση της κατανομής Weibull για την μοντελοποίηση της περιθώριας συμπεριφοράς των τυποποιημένων δεδομένων. Συγκεκριμένα, παρήχθησαν 100.000 ανεξάρτητες συνθετικές χρονοσειρές με μήκος 20 χρόνια η κάθε μια. Εν συνεχεία, οι τυποποιημένες συνθετικές χρονοσειρές αυτές πολλαπλασιάστηκαν με μια ζήτηση βάσης, η οποία επιλέχθηκε μέσω τυχαίας δειγματοληψίας (διαδικασία Monte-Carlo). Συγκεκριμένα, η ζήτηση βάσης για 50.000 σπίτια επιλέγεται από το διάστημα [100, 300], για 25.000 σπίτια στο διάστημα [300, 600], και τέλος, για τα υπόλοιπα 25.000 σπίτια στο διάστημα [600, 1000]. Η επιλογή αυτή έγινε για να παραχθούν μηνιαίες συνθετικές χρονοσειρές που αντιστοιχούν σε καταναλωτές με διαφορετικά προφίλ μέσης ζήτησης. Οι συνθετικές χρονοσειρές

πολλαπλασιάστηκαν με το κοινό προφίλ μεταβολής της εκροής ανά μήνα, για να προκύψουν οι τελικές συνθετικές μηνιαίες χρονοσειρές με περιοδικά μεταβαλλόμενη μέση τιμή ανά μήνα. Ένα ενδεικτικό παράδειγμα των χρονοσειρών που προέκυψαν φαίνεται στην εικόνα που ακολουθεί.



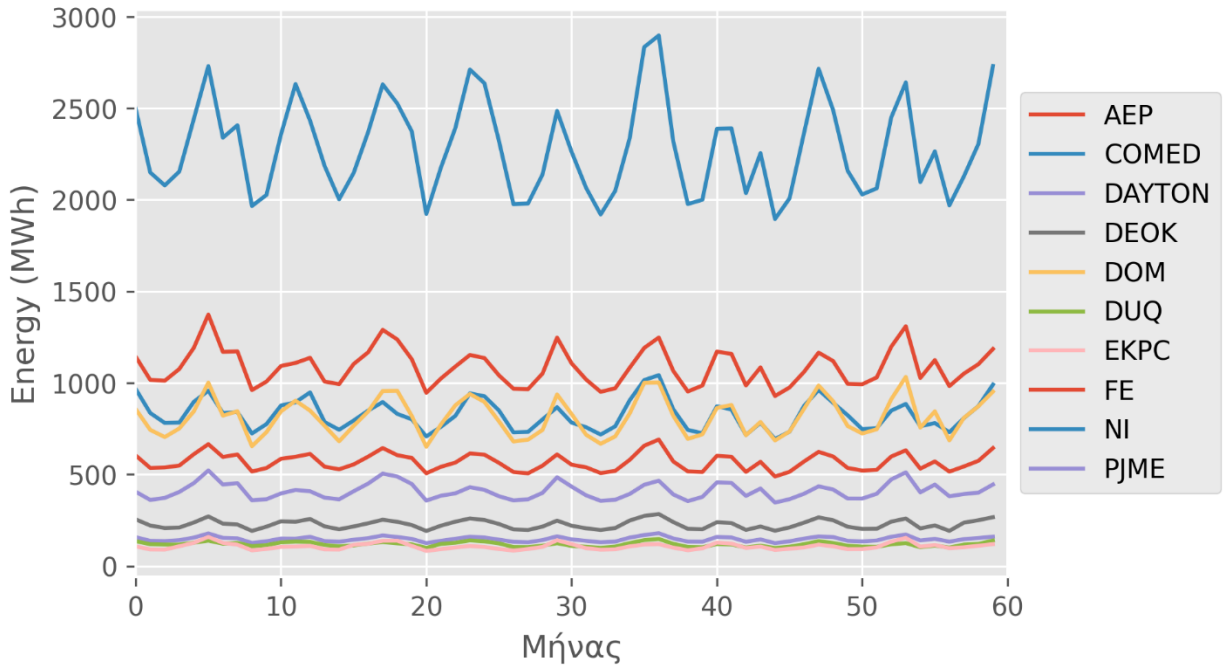
Διάγραμμα 4-4: Τυπική συνθετική χρονοσειρά

4.2 Δεδομένα ενέργειας

Γνωρίζοντας ότι τα συνθετικά δεδομένα δεν μπορούν να υποκαταστήσουν την εφαρμογή σε πραγματικά δεδομένα θεωρήθηκε σκόπιμο τα μοντέλα να δοκιμαστούν και με πραγματικά δεδομένα. Από την στιγμή που δεν υπήρχαν κατά την περίοδο ολοκλήρωσης της εργασίας διαθέσιμα δεδομένα από καταναλώσεις νερού, κρίθηκε σκόπιμη η εξέταση δεδομένων ενέργειας. Επιλέχθηκαν πραγματικά δεδομένα από την εταιρία PJM που παρέχονται από την ιστοσελίδα Kaggle : (Kaggle, 2022)Η Kaggle, θυγατρική της Google LLC, είναι μια διαδικτυακή κοινότητα επιστημόνων δεδομένων και επαγγελματιών μηχανικής μάθησης. Το Kaggle επιτρέπει στους χρήστες να βρίσκουν και να δημοσιεύουν σύνολα δεδομένων, να εξερευνούν και να δημιουργούν μοντέλα σε ένα διαδικτυακό περιβάλλον επιστήμης δεδομένων, να συνεργάζονται με άλλους επιστήμονες δεδομένων και μηχανικούς μηχανικής μάθησης και να συμμετέχουν σε διαγωνισμούς για την επίλυση προκλήσεων της επιστήμης δεδομένων. Η PJM είναι ένας περιφερειακός οργανισμός μετάδοσης ενέργειας στις Ηνωμένες Πολιτείες Αμερικής. Αποτελεί μέρος του δικτύου ανατολικής διασύνδεσης που λειτουργεί ένα σύστημα ηλεκτρικής μετάδοσης που εξυπηρετεί το σύνολο ή επιμέρους τμήματα των Delaware, Illinois, Indiana, Kentucky, Maryland, Michigan, New Jersey, North Carolina, Ohio, Pennsylvania, Tennessee, Virginia, West Virginia και της Περιφέρειας της Columbia. Στην διάθεση μας έχουμε δεδομένα από 11 περιοχές της ανατολικής Αμερικής και η κλίμακα αναφοράς είναι ωριαία.

4.2.1 Προεπεξεργασία δεδομένων

Η προεπεξεργασία δεδομένων μπορεί να αναφέρεται σε χειρισμό ή απόρριψη δεδομένων πριν χρησιμοποιηθούν προκειμένου να διασφαλιστεί ή να ενισχυθεί η απόδοση και είναι ένα σημαντικό βήμα στη διαδικασία εξόρυξης δεδομένων. Η φράση "garbage in, garbage out" ισχύει ιδιαίτερα για έργα εξόρυξης δεδομένων και μηχανικής μάθησης. Οι μέθοδοι συλλογής δεδομένων συχνά ελέγχονται χαλαρά, με αποτέλεσμα τιμές εκτός εύρους (π.χ. αρνητικές τιμές εισοδήματος), αδύνατοι συνδυασμοί δεδομένων (π.χ. Φύλο: Άνδρας, Έγκυος: Ναι) και ελλιπείς τιμές να εμφανίζονται αρκετά συχνά. Ανάλυση δεδομένων που δεν έχει ελεγχθεί προσεκτικά για τέτοια προβλήματα μπορεί να παράγει παραπλανητικά αποτελέσματα. Έτσι, η αναπαράσταση και η ποιότητα των δεδομένων έρχεται πρώτα και κύρια πριν από την εκτέλεση οποιασδήποτε ανάλυσης. Συχνά, η προεπεξεργασία δεδομένων είναι η πιο σημαντική φάση ενός έργου μηχανικής μάθησης. Στην δικιά μας περίπτωση τα δεδομένα των έντεκα σταθμών δεν αναπτύσσονταν στο ίδιο άξονα χρόνου (οι χρονοσειρές εξελίσσονταν σε διαφορετικές χρονικές στιγμές). Η προσαρμογή στον σωστό άξονα του χρόνου έγινε εύκολα με την βοήθεια του Excel καθώς η γραφική απεικόνιση που προσφέρει βοηθάει στην ευκολότερη επεξεργασία των δεδομένων. Είναι σημαντικό να σημειωθεί ότι τα δεδομένα ήταν δύο στήλες σε μορφή csv (coma separated values). Στην πρώτη στήλη αναγράφεται η ημερομηνία και η ώρα που έγινε η μέτρηση σε μορφή datetime (dd-mmm-yyyy hh:mm) ενώ στην δεύτερη στήλη αναγράφεται η ισχύς σε MW. Το επόμενο βήμα είναι η εξέταση των δεδομένων για την εύρεση προβλημάτων όπως κενά στις μετρήσεις ή τιμές που επαναλαμβάνονται. Λόγω του μεγάλου πλήθους τις χρονοσείρας (οι περιοχές μελέτης έχουν πάνω από εκατό χιλιάδες καταγεγραμμένες χρονικές στιγμές) ο έλεγχος αυτός είναι αδύνατο να γίνει μια προς μια. Λύση σε αυτό το πρόβλημα έχει προσφέρει ένας χρήστης του Kaggle όπου έχει προσφέρει κώδικα σε γλώσσα Python για την συμπλήρωση των χαμένων τιμών και την σωστή χρονική ταξινόμηση τις χρονοσειράς (ioanpier, 2020). Οι χαμένες τιμές συμπληρώνονται με τον μέσο όρο των προηγούμενων και επόμενων χρονικών στιγμών. Οι χρονοσειρές από την ωριαία μετατρέπονται σε μηνιαία κλίμακα με την εντολή .reshape την βιβλιοθήκης Pandas. Επειδή οι χρονοσειρές εξελίσσονται σε διαφορετικές χρονικές στιγμές επιλέγεται η περίοδος με τις περισσότερες ταυτόχρονα διαθέσιμες περιοχές. Το κριτήριο αυτό πραγματοποιείται για τις δέκα από τις έντεκα περιοχές καθώς και για χρονική διάρκεια των εξήντα μηνών. Για την βέλτιστη προσομοίωση των συνθηκών που θα κληθούμε να επιλύσουμε σε σχέση με πραγματικά δεδομένα καταναλώσεων νερού, μετασχηματίζουμε τα δεδομένα διαιρώντας δια δέκα χιλιάδες (10.000), ώστε να έχουν ανάλογη τάξη μεγέθους.



Διάγραμμα 4-5: Όλες οι πραγματικές χρονοσειρές που έχουμε στην διάθεση μας.

Όπως μπορούμε να παρατηρήσουμε οι περιοχές παρουσιάζουν συσχέτιση μεταξύ τους (οι χρονοσειρές παρουσιάζουν την ίδια χρονική στιγμή τοπικό μέγιστο). Λόγω της φύσης των δεδομένων αυτό είναι κάτι που θα περιμέναμε. Λόγω τις ίδιας τοπολογίας των περιοχών (ίδιο ημισφαίριο) παρουσιάζονται οι ίδιες εποχικότητες σε όλες τις περιοχές τις ανατολικής Αμερικής. Πιο συγκεκριμένα σε βάθος δωδεκάμηνου καταγράφονται σχεδόν πάντα δύο τοπικά μέγιστα, το καλοκαίρι όπου σχεδόν πάντα είναι και ολικό μέγιστο για τον χρόνο αυτό, και το χειμώνα.

4.3 Προετοιμασία δεδομένων για την εκπαίδευση των μοντέλων

Στη μηχανική μάθηση, μια τυπική εργασία είναι η μελέτη και η κατασκευή αλγορίθμων που μπορούν να μάθουν από δεδομένα και να κάνουν προβλέψεις σε δεδομένα. Τέτοιοι αλγόριθμοι λειτουργούν κάνοντας προβλέψεις ή παίρνοντας αποφάσεις βάσει δεδομένων, μέσω της κατασκευής ενός μαθηματικού μοντέλου από δεδομένα εισόδου. Αυτά τα δεδομένα εισόδου που χρησιμοποιούνται για την κατασκευή του μοντέλου συνήθως χωρίζονται σε πολλαπλά σύνολα δεδομένων. Συγκεκριμένα, τρία σύνολα δεδομένων χρησιμοποιούνται συνήθως σε διαφορετικά στάδια της δημιουργίας του μοντέλου: εκπαίδευση, επικύρωση και σύνολα δοκιμών.

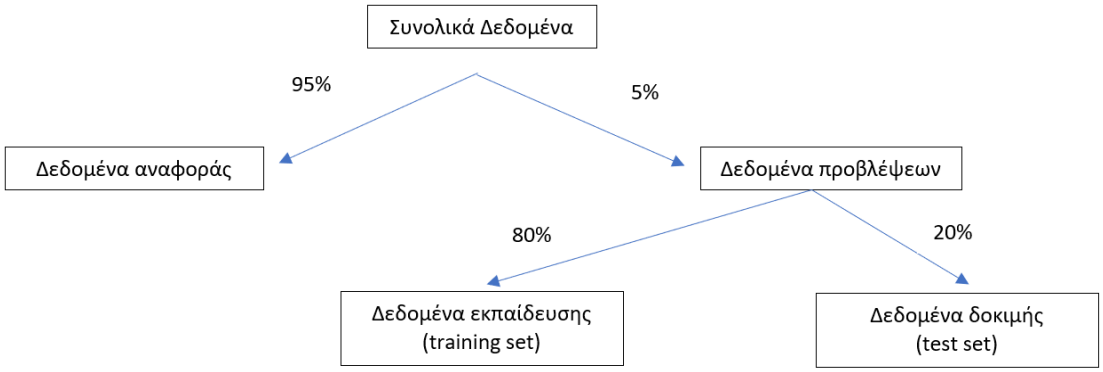
Το μοντέλο αρχικά προσαρμόζεται σε ένα σύνολο δεδομένων εκπαίδευσης (training set), που είναι ένα σύνολο παραδειγμάτων που χρησιμοποιούνται για την προσαρμογή των παραμέτρων (π.χ. βάρη συνδέσεων μεταξύ νευρώνων σε τεχνητά νευρωνικά δίκτυα) του μοντέλου. Το μοντέλο «εκπαιδεύεται» στο σύνολο δεδομένων εκπαίδευσης χρησιμοποιώντας μια μέθοδο εποπτευόμενης μάθησης, για παράδειγμα χρησιμοποιώντας μεθόδους βελτιστοποίησης όπως η κλίση κατάβασης. Στην πράξη, το σύνολο δεδομένων εκπαίδευσης αποτελείται συχνά από ζεύγη ενός διανύσματος εισόδου (ή αριθμού) και του αντίστοιχου διανύσματος εξόδου (ή αριθμού), όπου το κλειδί απάντησης συνήθως υποδηλώνεται ως στόχος (ή ετικέτα). Το τρέχον μοντέλο εκτελείται με το σύνολο δεδομένων

εκπαίδευσης και παράγει ένα αποτέλεσμα, το οποίο στη συνέχεια συγκρίνεται με τον στόχο για κάθε διάνυσμα εισόδου στο σύνολο δεδομένων εκπαίδευσης. Με βάση το αποτέλεσμα της σύγκρισης και τον συγκεκριμένο αλγόριθμο εκμάθησης που χρησιμοποιείται, προσαρμόζονται οι παράμετροι του μοντέλου. Η προσαρμογή του μοντέλου μπορεί να περιλαμβάνει τόσο την επιλογή μεταβλητής όσο και την εκτίμηση παραμέτρων (Brownlee, 2017).

Διαδοχικά, το προσαρμοσμένο (trained) μοντέλο χρησιμοποιείται για την πρόβλεψη των αποκρίσεων για τις παρατηρήσεις σε ένα δεύτερο σύνολο δεδομένων που ονομάζεται σύνολο δεδομένων επικύρωσης (validation set). Το σύνολο δεδομένων επικύρωσης παρέχει μια αμερόληπτη αξιολόγηση της προσαρμογής ενός μοντέλου στο σύνολο δεδομένων εκπαίδευσης κατά τον συντονισμό των υπερπαραμέτρων του μοντέλου (π.χ. τον αριθμό των κρυφών μονάδων—στρωμάτων και πλάτη στρώματος—σε ένα νευρωνικό δίκτυο). Τα σύνολα δεδομένων επικύρωσης μπορούν να χρησιμοποιηθούν για τακτοποίηση με πρόωρη διακοπή της εκπαίδευσης όταν αυξάνεται το σφάλμα στο σύνολο δεδομένων επικύρωσης, καθώς αυτό είναι σημάδι υπερβολικής προσαρμογής στο σύνολο δεδομένων εκπαίδευσης (Overfit). Αυτή η απλή διαδικασία περιπλέκεται στην πράξη από το γεγονός ότι το σφάλμα του συνόλου δεδομένων επικύρωσης μπορεί να παρουσιάζει διακυμάνσεις κατά τη διάρκεια της εκπαίδευσης, παράγοντας πολλαπλά τοπικά ελάχιστα. Αυτή η περιπλοκή οδήγησε στη δημιουργία πολλών μεθοδολογιών για την αναγνώριση του σημείου όπου αρχίζει να παρουσιάζεται υπερβολική προσαρμογή.

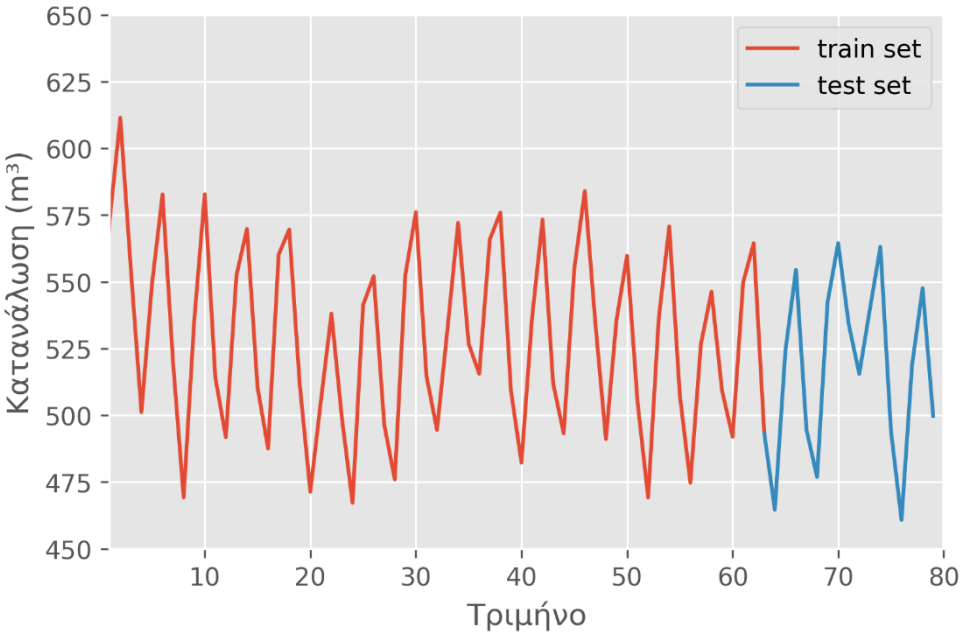
Τέλος, το σύνολο δεδομένων δοκιμής (test set) είναι ένα σύνολο δεδομένων που χρησιμοποιείται για να παρέχει μια αμερόληπτη αξιολόγηση ενός τελικού μοντέλου που ταιριάζει στο σύνολο δεδομένων εκπαίδευσης. Εάν τα δεδομένα του συνόλου δεδομένων δοκιμής δεν έχουν χρησιμοποιηθεί ποτέ στην εκπαίδευση (για παράδειγμα σε διασταυρούμενη επικύρωση, cross validation), το σύνολο δεδομένων δοκιμής ονομάζεται επίσης σύνολο δεδομένων κράτησης (*holdout data set*). Ο όρος "σύνολο επικύρωσης (*validation set*)" χρησιμοποιείται μερικές φορές αντί για "σύνολο δοκιμής (*test set*)" σε κάποια βιβλιογραφία (π.χ., εάν το αρχικό σύνολο δεδομένων χωρίστηκε σε δύο μόνο υποσύνολα, το σύνολο δοκιμής μπορεί να αναφέρεται ως σύνολο επικύρωσης). Η απόφαση για τα μεγέθη και τις στρατηγικές για τη διαίρεση συνόλων δεδομένων σε σύνολα εκπαίδευσης, δοκιμών και επικύρωσης εξαρτάται σε μεγάλο βαθμό από το πρόβλημα και τα διαθέσιμα δεδομένα.

Το πρώτο βήμα είναι η επιλογή του συνόλου των χρονοσειρών όπου θα επιχειρήσουμε να κάνουμε τις προβλέψεις. Αφού γίνει ο διαχωρισμός, χωρίζουμε τις χρονοσειρές σε δύο τμήματα. Πιο συγκεκριμένα για την περίπτωση όπου έχουμε συνθετικά δεδομένα:



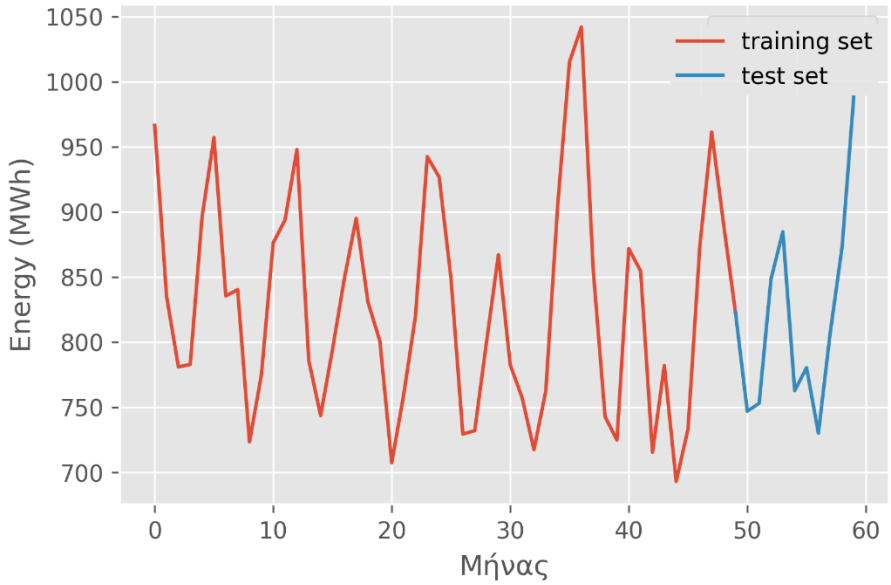
Εικόνα 4-1: Διάγραμμα διαχωρισμού δεδομένων.

Ο αρχικός αριθμός των χρονοσειρών ανέρχεται στις εκατό χιλιάδες. Λόγω του μεγάλου πλήθους των δεδομένων επιλέχθηκε μικρό ποσοστό της τάξης του 5%. Το κριτήριο επιλογής του 5% έγινε αποκλειστικά από το γεγονός ότι ορισμένες μεθοδολογίες θέλουν πολύ χρόνο για να εκπαιδευτούν. Η κάθε χρονοσειρά που ανήκει στα δεδομένα προβλέψεων χωρίζεται εκ νέου σε δεδομένα εκπαίδευσης και δοκιμής με ποσοστά 80% και 20% αντίστοιχα. Το μήκος της χρονοσειράς είναι 80 τρίμηνα εκ των οποίων τα εξήντα τέσσερα (64) αποτελούν τα δεδομένα εκπαίδευσης (*training*) και τα υπόλοιπα δεκαέξι (16) τα δεδομένα δοκιμής (*test*). Στο παρακάτω διάγραμμα φαίνεται ενδεικτικά ο διαχωρισμός.



Διάγραμμα 4-6: Καταχωρισμός σε δεδομένα εκπαίδευσης(*training*) και δεδομένα δοκιμής(test) για τα συνθετικά δεδομένα.

Για τα δεδομένα ενέργειας της ιστοσελίδας Kaggle ο διαχωρισμός δεν έγινε με τις ίδιες αναλογίες. Λόγω του πλήθους των δεδομένων, πρέπει να παρέχουμε επαρκή δεδομένα για εκπαίδευση. Οι διαθέσιμες χρονοσειρές είναι δέκα (10) εκ των οποίων οι δύο (2) επιλέγονται για προβλέψεις. Το μήκος της χρονοσειράς είναι οι εξήντα (60) μήνες. Σε αυτή την περίπτωση επιλέγουμε το χρονικό βήμα να είναι μήνας σε αντίθεση με το τρίμηνο στα συνθετικά δεδομένα, επειδή με την αναγωγή στην τριμηνιαία κλίμακα οι χρονοσειρές θα προέκυπταν πολύ μικρές και δεν θα μπορούσαν τα μοντέλα να εκπαιδευτούν κατάλληλα. Ο διαχωρισμός σε δεδομένα εκπαίδευσης και δοκιμής γίνεται σε αναλογία πενήντα – δέκα 50-10, δηλ. 50 μήνες εκπαίδευση και 10 μήνες δοκιμή. Στο παρακάτω διάγραμμα φαίνεται ενδεικτικά ο διαχωρισμός.



Διάγραμμα 4-6: Καταχωρισμός σε δεδομένα εκπαίδευσης(training) και δεδομένα δοκιμής(test) για τα δεδομένα ενέργειας.

5. Μοντέλα προβλέψεων

Στη στατιστική μοντελοποίηση, η ανάλυση παλινδρόμησης είναι ένα σύνολο στατιστικών διαδικασιών για την εκτίμηση των σχέσεων μεταξύ μιας εξαρτημένης μεταβλητής (συχνά αποκαλούμενη μεταβλητή «αποτέλεσμα» ή «απόκριση») και μίας ή περισσότερων ανεξάρτητων μεταβλητών (συχνά αποκαλούμενες «πρόβλεψη», «συμμεταβλητές», «επεξηγηματικές μεταβλητές» ή «χαρακτηριστικά»). Η πιο κοινή μορφή ανάλυσης παλινδρόμησης είναι η γραμμική παλινδρόμηση, στην οποία βρίσκουμε τη γραμμή (ή έναν πιο σύνθετο γραμμικό συνδυασμό) που ταιριάζει περισσότερο στα δεδομένα σύμφωνα με ένα συγκεκριμένο μαθηματικό κριτήριο. Για παράδειγμα, η μέθοδος των συνηθισμένων ελαχίστων τετραγώνων υπολογίζει τη μοναδική γραμμή (ή υπερεπίπεδο) που ελαχιστοποιεί το άθροισμα των τετραγωνικών διαφορών μεταξύ των αληθινών δεδομένων και αυτής της γραμμής (ή υπερεπίπεδου). Για συγκεκριμένους μαθηματικούς λόγους (βλ. γραμμική παλινδρόμηση), αυτό επιτρέπει στον ερευνητή να εκτιμήσει την υπό όρους προσδοκία (ή μέση τιμή πληθυσμού) της εξαρτημένης μεταβλητής όταν οι ανεξάρτητες μεταβλητές λαμβάνουν ένα δεδομένο σύνολο τιμών. Οι λιγότερο κοινές μορφές παλινδρόμησης χρησιμοποιούν ελαφρώς διαφορετικές διαδικασίες για να εκτιμήσουν εναλλακτικές παραμέτρους τοποθεσίας (π.χ. ποσοτική παλινδρόμηση ή Ανάλυση απαραίτητης συνθήκης) ή εκτιμούν την υπό όρους προσδοκία σε μια ευρύτερη συλλογή μη γραμμικών μοντέλων (π.χ. μη παραμετρική παλινδρόμηση).

Η ανάλυση παλινδρόμησης χρησιμοποιείται κυρίως για δύο εννοιολογικά διακριτούς σκοπούς. Πρώτον για την πρόβλεψη, όπου η χρήση της έχει ουσιαστική επικάλυψη με το πεδίο της μηχανικής μάθησης. Δεύτερον, σε ορισμένες περιπτώσεις, η ανάλυση παλινδρόμησης μπορεί να χρησιμοποιηθεί για να συμπεράνει αιτιώδεις σχέσεις μεταξύ των ανεξάρτητων και των εξαρτημένων μεταβλητών. Είναι σημαντικό ότι οι παλινδρομήσεις από μόνες τους αποκαλύπτουν μόνο σχέσεις μεταξύ μιας εξαρτημένης μεταβλητής και μιας συλλογής ανεξάρτητων μεταβλητών σε ένα σταθερό σύνολο δεδομένων. Για να χρησιμοποιήσει παλινδρομήσεις για πρόβλεψη ή για να συμπεράνει αιτιώδεις σχέσεις, αντίστοιχα, ένας ερευνητής πρέπει να αιτιολογήσει προσεκτικά γιατί οι υπάρχουσες σχέσεις έχουν προγνωστική δύναμη ή γιατί μια σχέση μεταξύ δύο μεταβλητών έχει αιτιολογική ερμηνεία. Το τελευταίο είναι ιδιαίτερα σημαντικό όταν οι ερευνητές επιζητούν να εκτιμήσουν τις αιτιώδεις σχέσεις χρησιμοποιώντας δεδομένα παρατήρησης. Στα πλαίσια της εργασίας αυτής θα επικεντρωθούμε αποκλειστικά στον πρώτο σκοπό, καθώς ο στόχος μας είναι η πρόβλεψη καταναλώσεων των πελατών.

Οι μεθοδολογίες που εφαρμόστηκαν μπορούν να χωριστούν σε δύο κατηγορίες ανάλογα με τον τρόπο που αξιοποιούν τα δεδομένα που διαθέτουμε. Τις οριζόντιες και τις κάθετες. Στη συνέχεια θα εξηγηθεί αναλυτικά η διαφορά ανάμεσα στις δύο προσεγγίσεις.

5.1: Οριζόντιες μεθοδολογίες

Ο ορισμός της οριζόντιας μεθοδολογίας προκύπτει από τον τρόπο όπου αξιοποιείται η πληροφορία. Πιο συγκεκριμένα η πληροφορία “ρέει” παράλληλα από τον άξονα του χρόνου δηλαδή οριζόντια. Τα μοντέλα εκπαιδεύονται με δεδομένα της ίδιας χρονοσειράς παλαιότερων χρονικών στιγμών.

5.1.1: Μέθοδος Naive

Η μέθοδος naive είναι το πιο οικονομικό (υπολογιστικά-cost effective) μοντέλο πρόβλεψης και παρέχει αποδόσεις που μπορούν να συγκριθούν και με πιο εξελιγμένα μοντέλα. Αυτή η μέθοδος πρόβλεψης είναι κατάλληλη μόνο για δεδομένα χρονοσειρών. Χρησιμοποιώντας την μέθοδο Naive, παράγονται προβλέψεις που είναι ίσες με την τελευταία παρατηρούμενη τιμή. Αυτή η μέθοδος λειτουργεί αρκετά καλά για οικονομικές και χρηματοοικονομικές χρονοσειρές, οι οποίες συχνά έχουν μοτίβα που είναι δύσκολο να προβλεφθούν αξιόπιστα και με ακρίβεια. Εάν η χρονοσειρά περιέχει εποχικότητα, η εποχιακή αφελής προσέγγιση (seasonal naive methodology) μπορεί να είναι πιο κατάλληλη όταν οι προβλέψεις είναι ίσες με την τιμή από την προηγούμενη περίοδο. Σε σημειογραφία χρονοσειρών η μέθοδος γράφεται ως εξής:

$$\hat{y}_{T+h\vee T} = y_T \tag{5.1}$$

όπου:

- y_t η τιμή της χρονοσειράς την χρονική στιγμή t

Παρόλα τα θετικά, η μέθοδος αυτή είναι πάρα πολύ επιρρεπής στον θόρυβο (noise), για παράδειγμα αν η προηγούμενη τιμή είναι μια μη αντιπροσωπευτική τιμή της χρονοσειράς (μπορεί και λάθος μέτρησης) τότε το αποτέλεσμα θα έχει πολύ μεγάλη απόκλιση από την πραγματικότητα. Για την αντιμετώπιση αυτού του προβλήματος μπορούμε να λαμβάνουμε πληροφορία όχι μόνο από την τελευταία τιμή της εποχικότητας, αλλά και προηγούμενων εποχικοτήτων με κάποιο συντελεστή βάρους. Ό τύπος της διαδικασίας είναι:

$$F_i = \alpha_1 * A_{i,t-T} + \alpha_2 * A_{i,t-2T} + \alpha_3 * A_{i,t-3T} \dots \tag{5.2}$$

όπου:

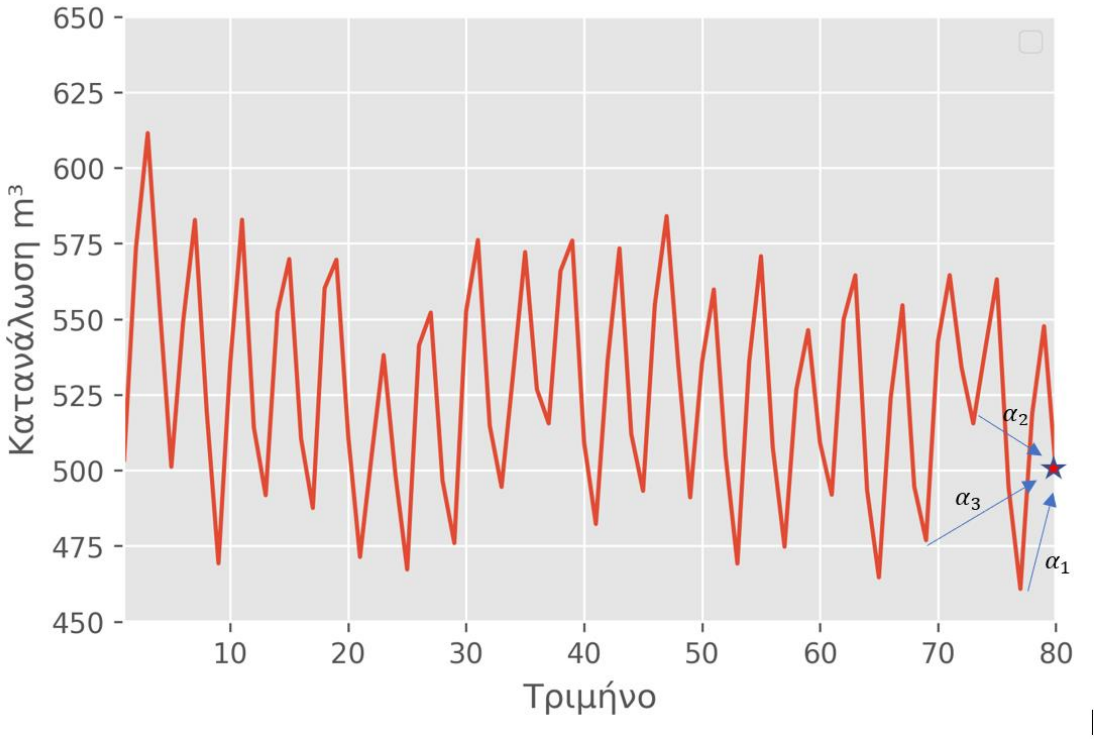
- $A_{i,t}$ η τιμή της χρονοσειράς την χρονική στιγμή t για τον πελάτη i.
- α_{i} ο συντελεστής βαρύτητας για την χρονική στιγμή i.

Οι συντελεστές βάρους α_i δείχνουν το βάρος της τιμής και ο τελεστής i δηλώνει το χρονικό βήμα. Για παράδειγμα για i=2 αναφερόμαστε στην χρονική στιγμή $t = 2T$. Για την διατήρηση του μέσου όρου της χρονοσειράς είναι απαραίτητο οι συντελεστές να αθροίζουν στην μονάδα δηλαδη:

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{5.3}$$

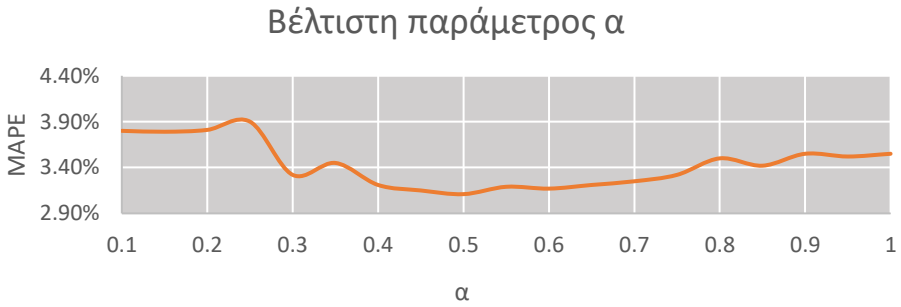
Αν η χρονοσειρά είχε σταθερό βαθμό αύξησης ή μείωσης θα μπορούσαμε να περιγράψουμε την μεταβολή αυτή με την μεταβολή της προηγούμενης ισότητας. Αν το άθροισμα είναι μεγαλύτερο της

μονάδας τότε η πρόβλεψη θα έχει αύξηση του μέσου όρου, ενώ αν είναι μικρότερο της μονάδας τότε η χρονοσειρά θα είναι φθίνουσα.



Διάγραμμα 5-1:Τρόπος πρόβλεψης με την μέθοδο Naive

Οι βέλτιστοί συντελεστές α_i υπολογίζονται με την μέθοδο εξαντλητικής δοκιμής (exhaustive testing) με μόνη υπερ-παράμετρο τον αριθμό i. Ο αριθμός i επιλέγεται από δοκιμές ίσος με τρία, δηλαδή το αποτέλεσμα θα το επηρεάζουν τα τρία τελευταία χρόνια. Πιο αναλυτικά έγινε δοκιμή όλων των δυνατών τιμών α_i ανά 5%. **Ο συνδυασμός που έδωσε τα καλύτερα αποτελέσματα είναι:** $\vec{\alpha} = (\mathbf{0.50}, \mathbf{0.25}, \mathbf{0.25})$ όπως φαίνεται και στο διάγραμμα 5-2.

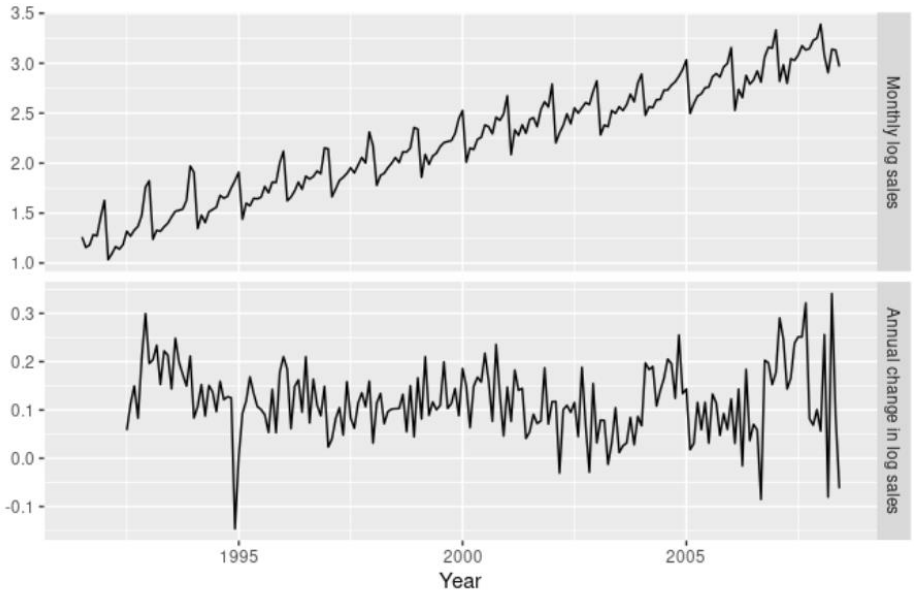


Διάγραμμα 5-2: Μεταβολή απόδοσης ανάλογα τον συντελεστή α.

5.1.2: Μέθοδος ARIMA

Στη στατιστική και στην οικονομετρία, και ιδιαίτερα στην ανάλυση χρονοσειρών, ένα μοντέλο αυτοπαλινδρομικού ολοκληρωμένου κινητού μέσου όρου (AutoRegressive Integrated Moving Average - ARIMA) είναι μια γενίκευση ενός μοντέλου αυτοπαλινδρομικού κινητού μέσου όρου (ARMA). Και τα δύο αυτά μοντέλα προσαρμόζονται σε δεδομένα χρονοσειρών είτε για την καλύτερη κατανόηση των δεδομένων είτε για την πρόβλεψη μελλοντικών σημείων της σειράς (πρόβλεψη). Τα μοντέλα ARIMA εφαρμόζονται σε ορισμένες περιπτώσεις όπου τα δεδομένα δείχνουν στοιχεία μη σταθερότητας με την έννοια του μέσου όρου (αλλά όχι διακύμανσης/αυτομεταβλητότητας), όπου μπορεί να εφαρμοστεί ένα αρχικό βήμα διαφοράς (που αντιστοιχεί στο "ολοκληρωμένο" μέρος του μοντέλου) ή περισσότερες φορές για να εξαλειφθεί η μη σταθερότητα της μέσης συνάρτησης (δηλ. η τάση). Το τμήμα AR του ARIMA υποδεικνύει ότι η εξελισσόμενη μεταβλητή ενδιαφέροντος υποχωρεί στις δικές της υστερούσες (δηλαδή, προηγούμενες) τιμές. Το τμήμα MA υποδεικνύει ότι το σφάλμα παλινδρόμησης είναι στην πραγματικότητα ένας γραμμικός συνδυασμός όρων σφάλματος των οποίων οι τιμές εμφανίστηκαν ταυτόχρονα και σε διάφορες χρονικές στιγμές στο παρελθόν. Το I (ήτοι "Integrated") υποδεικνύει ότι οι τιμές δεδομένων έχουν αντικατασταθεί με τη διαφορά μεταξύ των τιμών τους και των προηγούμενων τιμών (και αυτή η διαδικασία διαφοροποίησης μπορεί να έχει πραγματοποιηθεί περισσότερες από μία φορές). Ο σκοπός καθενός από αυτά τα χαρακτηριστικά είναι να κάνει το μοντέλο να ταιριάζει με τα δεδομένα όσο το δυνατόν καλύτερα. Οι συντελεστές του μοντέλου (p,d,q) περιγράφουν τον βαθμό των παραπάνω διαδικασιών. Όταν δύο από τους τρεις όρους είναι μηδέν, το μοντέλο μπορεί να αναφέρεται βάσει της μη μηδενικής παραμέτρου, αφαιρώντας τα "AR", "I" ή "MA" από το ακρωνύμιο που περιγράφει το μοντέλο. Για παράδειγμα, το ARIMA(1, 0, 0) είναι AR(1), ARIMA(0, 1, 0) είναι I(1), και το ARIMA(0, 0, 1) είναι MA (1).

Η διαδικασία εκμάθησης του μοντέλου ξεκινάει από τη σταθεροποίηση της χρονοσειράς, αυτό γίνεται με τον όρο I του μοντέλου. Δημιουργούμε μια νέα χρονοσειρά η οποία αποτελείται από την διαφορά δύο διαδοχικών χρονικών στιγμών. Η λειτουργία αυτή φαίνεται στο παρακάτω διάγραμμα.



Διάγραμμα 5-3: Μετατροπή χρονοσειράς από αύξουσα σε στάσιμη

Πολλές φορές για να πετύχουμε στασιμότητα στην χρονοσειρά θα χρειαστεί να εφαρμόσουμε το στάδιο της διαφοροποίησης πάνω από μια φορά. Ο αριθμός των διαφοροποιήσεων είναι ίσος με τον δεύτερο συντελεστή του μοντέλου (d). Μετά την σταθεροποίηση γίνεται προσαρμογή του μοντέλου ARMA(p',d) στην τροποποιημένη χρονοσειρά. Για δεδομένη χρονοσειρά Xt όπου t είναι ένας ακέραιος δείκτης και οι Xt είναι πραγματικοί αριθμοί. Το μοντέλο $ARMA(p', q)$ δίνεται από:

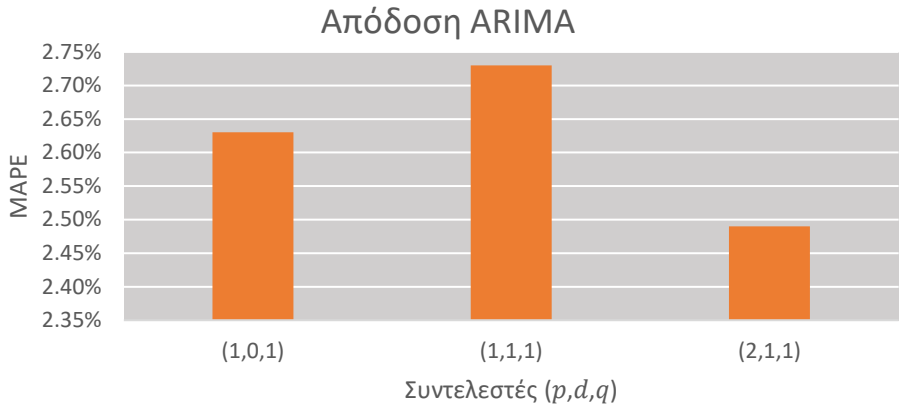
$$X_t - a_1 X_{t-1} - \cdots - a_{p'} X_{t-p'} = \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q} \rightarrow \quad (5.4)$$

$$\left(1 - \sum_{i=1}^{p'} a_i L^i\right) X_i = \left(1 + \sum_{i=1}^{q} \theta_i L^i\right) \varepsilon_t \quad (5.5)$$

Όπου L^i είναι ο τελεστής υστέρησης, a_i οι παράμετροι του αυτοπαλινδρόμενου τμήματος του μοντέλου, θ_i οι συντελεστές του κυλιόμενου μέσου όρου και ε_t ο όρος σφάλματος. Ο όρος ε_t γενικά θεωρείται ανεξάρτητος ακολουθώντας κανονική κατανομή με μηδενικό μέσο όρο. Από τα δεδομένα που διαθέτουμε βρίσκουμε τους συντελεστές των μοντέλων AR και MA.

Η επιλογή των συντελεστών (p,d,q) γίνεται πάλι με την δοκιμή (εξαντλητική μέθοδος) και επιλέγεται η τριάδα των μεταβλητών όπου έχει το μικρότερο σφάλμα. Θα πρέπει να σημειωθεί πως για τον λόγο όπου οι χρονοσειρές ήταν από την φύση τους στάσιμες, για $p > 3$ (όπου αρχίζει και παρουσιάζεται η στασιμότητα καθώς κλείνει ένας κύκλος εποχικότητας) ο αλγόριθμος δεν μπορεί να συγκλίνει. Πιο συγκεκριμένα η μέθοδος που χρησιμοποιείται για την επίλυση του γραμμικού συστήματος (LU-

Decomposition) δεν μπορεί να οδηγήσει σε πραγματική λύση καθώς η ορίζουσα των συντελεστών του συστήματος είναι ίση με το μηδέν.



Διάγραμμα 5-4: Απόδοση σε συνθετικά δεδομένα μοντέλου ARIMA συναρτήσει των παραμέτρων.

Για τα συνθετικά δεδομένα οι βέλτιστοι συντελεστές είναι (2,1,1) και από εδώ και πέρα κάθε συγκριτική αναφορά στο μοντέλο θα είναι για αυτούς τους συντελεστές. Ένας ανταλλακτικός τρόπος ελέγχου ακαταλληλότητας του μοντέλου είναι με την βοήθεια του Μπεϋζιανού κριτηρίου πληροφορίας BIC (Bayesian information criterion). Υπολογίζοντας την πιθανοφάνεια κάθε μοντέλου μπορεί να υπολογιστεί ο αριθμός BIC και επιλέγεται το μοντέλο με τον μικρότερο BIC.

Η εφαρμογή της μεθοδολογίας γίνεται με την βοήθεια της βιβλιοθήκης statsmodels. Για την κλασσική εκτέλεση του μοντέλου ο κώδικάς είναι πολύ απλός και μπορεί γραφθεί σε 3 γραμμές όπως φαίνεται στην ακόλουθη εικόνα.

```
>>> mod = sm.tsa.arima.ARIMA(endog, order=(1, 0, 0))
>>> res = mod.fit()
>>> print(res.summary())
```

Εικόνα 5-1: Τρόπος εκπαίδευσης μοντέλου ARIMA σε προγραμματιστικό περιβάλλον Python

Όπου: endog είναι τα γνωστά δεδομένα που θα εκπαιδευτεί το μοντέλο (endogenous variables) και order η τάξη του μοντέλου ARIMA, στην περίπτωση αυτή μιλάμε για το μοντέλο ARIMA(1,0,0) η το μοντέλο AR(1). Με την εντολή .predict() επιστρέφεται η πρόβλεψη για την αμέσως επόμενη τιμή από τα δεδομένα εκπαίδευσης.

5.1.3: Εποχιακή μέθοδος SARIMA

Μέχρι στιγμής, έχουμε περιορίσει την προσοχή μας σε μη εποχικά μοντέλα ARIMA. Ωστόσο, τα μοντέλα ARIMA είναι επίσης ικανά να μοντελοποιήσουν ένα ευρύ φάσμα εποχικών δεδομένων. Ένα εποχικό μοντέλο ARIMA διαμορφώνεται με τη συμπερίληψη πρόσθετων εποχιακών όρων στα μοντέλα. Ο τρόπος γραφής είναι:

$$ARIMA(p,d,q)(P,D,Q)_m \tag{5.6}$$

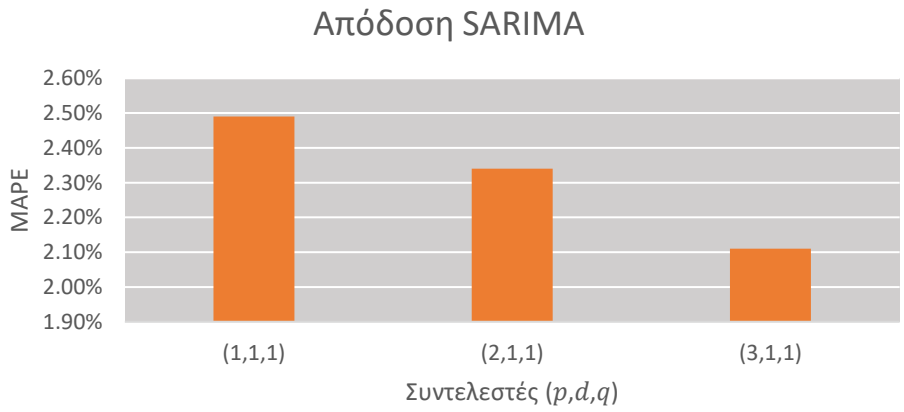
όπου m ο χρόνος της περιόδου ή της εποχικότητας,ενώ οι συντελεστές (P,D,Q) είναι οι εποχιακοί συντελεστές αυτοπαλινδρόμισης, διαφοράς και κινητού μέσου όρου. Η διαφορά του εποχικού μοντέλου με του κλασικού μοντέλου $ARIMA$ έχει να κάνει με τον τρόπο που γίνεται η διαφοροποίηση. Από την άποψη της επεξεργασίας σήματος, ειδικά της θεωρίας της φασματικής ανάλυσης Fourier, η τάση είναι το τμήμα χαμηλής συχνότητας στο φάσμα μιας μη στάσιμης χρονοσειράς, ενώ η εποχή είναι το τμήμα περιοδικής συχνότητας στο φάσμα αυτής. Επομένως, η διαφοροποίηση λειτουργεί ως φίλτρο υψηλής διέλευσης (*high-pass digital filter*)και η εποχιακή διαφοροποίηση ως φίλτρο χτενίσματος για την καταστολή της τάσης χαμηλής συχνότητας και της περιόδου περιοδικής συχνότητας στον τομέα του φάσματος (Shixiong, Chongshou, & Andrew, 2019). Μια σχετικά απλή μέθοδο περιοδικής διαφοροποίησης είναι:

$$\acute{y_t} = y_t - y_{t-m} \tag{5.7}$$

όπου:

- y_t η τιμή της χρονοσειρα την χρονική στιγμή t.
- m η περιοδικότητα του φαινομένου.

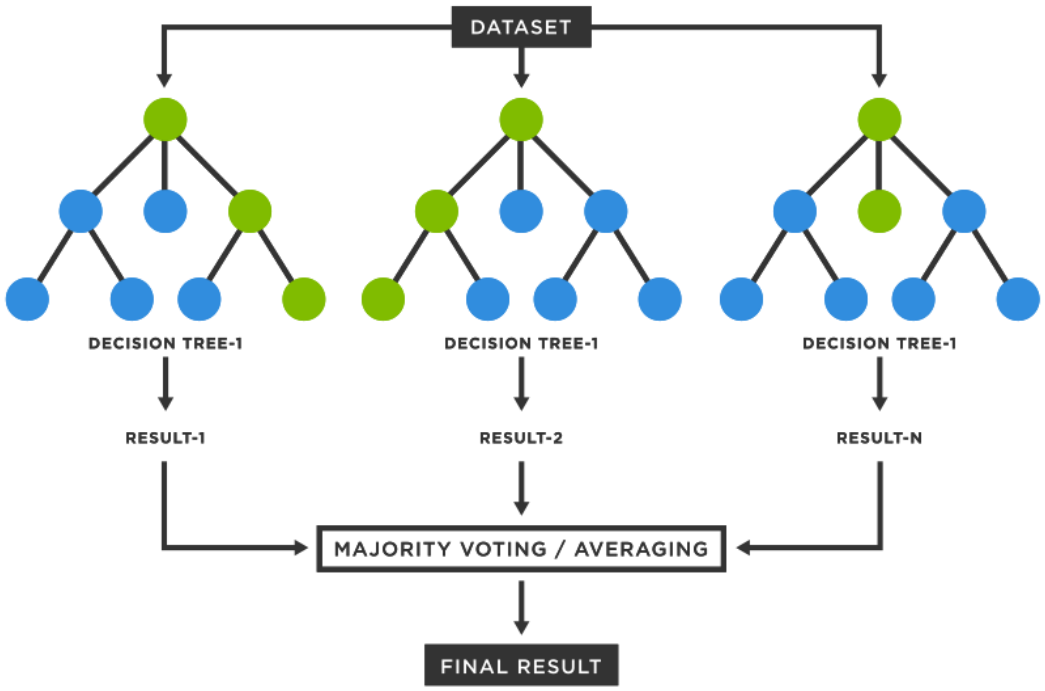
Σε γενικές γραμμές το μοντέλο $SARIMA$ είναι μια αναβάθμιση του κλασικού μοντέλου $ARIMA$ ικανό να προβλέψει και να περιγράψει πολύ πιο περίπλοκα φαινόμενα. Όσον αφορά τον τρόπο υλοποίησης του εποχιακού μοντέλου το μόνο που πρέπει να αλλαχτεί στον κώδικα σε σύγκριση με την μέθοδο $ARIMA$ είναι το όρισμα στην είσοδο της συνάρτησης και πιο συγκεκριμένα στο πεδίο seasonal_order -ο όποιος είναι μια τουπλέτα (tuplet)- και αποτελείται από τους αριθμούς (P,D,Q,m). Λόγω της τρίμηνης φύσης των συνθετικών δεδομένων η εποχικότητα επιλέγεται $m = 4$ (δηλαδή κάθε έτος αποτελείται από 4 τρίμηνα). Οι υπόλοιποι συντελεστές βρίσκονται με τον ίδιο τρόπο όπως και με την κλασσική μέθοδο ARIMA. **Στην περίπτωση των συνθετικών δεδομένων το βέλτιστο μοντέλο είναι (3,1,1)(0,0,0,4) ενώ στην περίπτωση των δεδομένων ενέργειας είναι (2,1,1)(0,0,0,3).**



Διάγραμμα 5-5: Απόδοση σε συνθετικά δεδομένα μοντέλου ARIMA συναρτήσει των παραμέτρων.

5.1.4: Μοντέλο τυχαίου Δάσους (Random Forest)

Τα τυχαία δάση ή τα δάση τυχαίας απόφασης (Random Forest η Random decision forest) είναι μια μέθοδος εκμάθησης συνόλου για ταξινόμηση, παλινδρόμηση και άλλες εργασίες που λειτουργεί κατασκευάζοντας ένα πλήθος δέντρων αποφάσεων κατά το χρόνο εκπαίδευσης. Για εργασίες ταξινόμησης, η έξοδος του τυχαίου δάσους είναι η κλάση που επιλέγεται από τα περισσότερα δέντρα. Για μοντέλα παλινδρόμησης, επιστρέφεται ο μέσος όρος ή ο μέσος όρος πρόβλεψης των μεμονωμένων δέντρων. Ο αλγόριθμος για τα δάση τυχαίας απόφασης δημιουργήθηκε από (Tin & AT and T Bell Laboratories, Inc., 1995)με την βοήθεια της μεθόδου του τυχαίου υποχώρου, η οποία, στη διατύπωση του Ho, είναι ένας τρόπος για την εφαρμογή της προσέγγισης της «στοχαστικής διάκρισης» στην ταξινόμηση που προτάθηκε από(Kleinberg, 1990). Τα τυχαία δάση χρησιμοποιούνται συχνά ως μοντέλα «μαύρου κουτιού (black box)» καθώς δημιουργούν λογικές προβλέψεις σε ένα ευρύ φάσμα δεδομένων, ενώ απαιτούν μικρή διαμόρφωση.



Εικόνα 5-2: Κλασσική αρχιτεκτονική μοντέλου Τυχαίου Δάσους

Αρχικά δημιουργείται ένας διακριτός αριθμός δέντρων (συνήθως τυχαίας αρχιτεκτονικής). Ο αριθμός των δέντρων είναι υπερπαράμετρος (δηλαδή εξαρτάται από τον χρήστη στην αρχή και δεν βελτιώνεται όσο εκπαιδεύεται το μοντέλο). Κατά την εκπαίδευση χωρίζονται τα δεδομένα σε υποομάδες (*batches*). Για δεδομένο σετ $X = x_1, \dots, x_n$με αποτελέσματα $Y = y_1, \dots, y_n$ εκπαιδεύουμε τα δέντρα f_b. Μετά την εκπαίδευση των δέντρων για την πρόβλεψη των δεδομένων που θα γίνει δοκιμή (*test set*) λαμβάνουμε υπόψη τον μέσο όρο όλων των δέντρων σύμφωνα με τον τύπο:

$$\mathrm{F} = \frac{1}{B}\sum_{b=1}^{B} f_b(\acute{x}) \tag{5.8}$$

όπου:

- $f_b(\acute{x})$ το αποτέλεσμα του δέντρου b για όρισμα εισόδου $\acute{x}$.
- B το πλήθος των δέντρων.
- F το αποτέλεσμα της μεθόδου.

Στην περίπτωση που έχουμε πρόβλημα ταξινόμησης, πρέπει να θεωρήσουμε την πλειοψηφία των δέντρων. Η διαδικασία αυτή ονομάζεται *bootstrapping* σύμφωνα με την διεθνή βιβλιογραφία. Η αρχιτεκτονική του τυχαίου δάσους περιγράφεται στην εικόνα 5-2. Αυτή η διαδικασία οδηγεί σε καλύτερη απόδοση του μοντέλου επειδή μειώνει τη διακύμανση του μοντέλου, χωρίς να αυξάνει την προκατάληψη (*bias*). Αυτό σημαίνει ότι ενώ οι προβλέψεις ενός μεμονωμένου δέντρου είναι πολύ ευαίσθητες στο θόρυβο στο σετ εκπαίδευσής τους του, ο μέσος όρος πολλών δέντρων δεν είναι, εφόσον τα δέντρα δεν συσχετίζονται. Η απλή εκπαίδευση πολλών δέντρων σε ένα μόνο σετ εκπαίδευσης θα έδινε δέντρα ισχυρά συσχετισμένα (ή ακόμα και το ίδιο δέντρο πολλές φορές, εάν ο αλγόριθμος εκπαίδευσης είναι ντετερμινιστικός). Η διασπορά των αποτελεσμάτων των δέντρων στην περίπτωση που έχουμε πρόβλημα παλινδρόμησης μπορεί να υπολογιστεί από την σχέση:

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}\left(f_b(\acute{x}) - \hat{f}\right)^2}{B-1}} \tag{5.9}$$

Λόγω της πολυπλοκότητας των μοντέλων τυχαίου δάσους χρησιμοποιούμε έτοιμες βιβλιοθήκες για την ευκολότερη υλοποίηση και επίλυση του προβλήματος. Η βιβλιοθήκη *sklearn* περιέχει έτοιμες συναρτήσεις για τυχαία δάση για όλα τα προβλήματα που μπορεί να αντιμετωπίσει ο μηχανικός (παλινδρόμησης και ταξινόμησης). Συγκεκριμένα αυτό επιτυγχάνεται με τη χρήση της μεθόδου *RandomForestClassifier* όπως φαίνεται στην επόμενη εικόνα.

```
>>> from sklearn.ensemble import RandomForestClassifier
>>> X = [[0, 0], [1, 1]]
>>> Y = [0, 1]
>>> clf = RandomForestClassifier(n_estimators=10)
>>> clf = clf.fit(X, Y)
```

Εικόνα 5-3: Κώδικας εκτέλεσης Τυχαίου Δάσους σε προγραμματιστικό περιβάλλον Python

Στην περίπτωση που θέλουμε να τρέξουμε το πρόγραμμα με τις προεπιλεγμένες ρυθμίσεις το μόνο δεδομένο που χρειάζεται να δώσουμε είναι το πλήθος των τυχαίων δέντρων(*n_estimators*). Γενικά οι προεπιλεγμένες ρυθμίσεις αποδίδουν αρκετά καλά ειδικά στην περίπτωση όπου θέλουμε να εξετάσουμε χονδροειδώς και γρήγορα την απόδοση του μοντέλου.

Για τις απαιτήσεις της παρούσας εργασίας πρώτα θα πρέπει να δημιουργήσουμε και να τροποποιήσουμε τα δεδομένα ώστε να μπορούν να τροφοδοτηθούν με τον τρόπο που απαιτείται από το σχετικό πακέτο.. Θα πρέπει τα δεδομένα να είναι κατάλληλα μορφοποιημένα συμφώνα με της απαιτήσεις του προγραμματιστή του μοντέλου. Για καλή μας τύχη για τις ανάγκες των απλών χρηστών η βιβλιοθήκη του

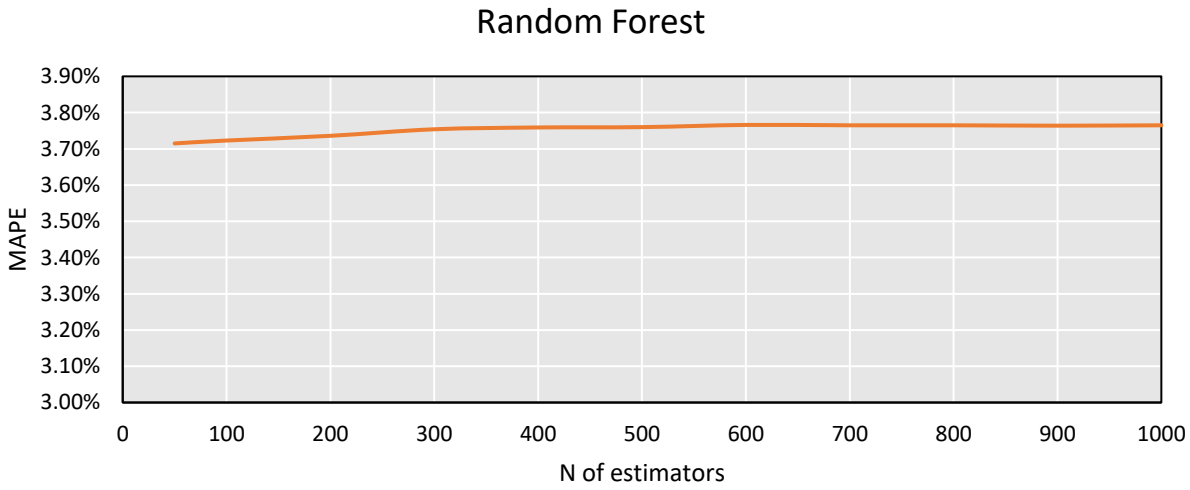
Keras παρέχει συνάρτηση η οποία δημιουργεί μια κλάση με την μορφοποίηση που θέλει το πακέτο του *RandomForestClassifier*. Η συνάρτηση αυτή ονομάζεται: *TimeseriesGenerator* όπως φαίνεται στην επόμενη εικόνα.

```
data_gen = TimeseriesGenerator(data, targets,
                               length=10, sampling_rate=2,
                               batch_size=2)
```

Εικόνα 5-4: Τρόπος εκτέλεσης της συνάρτησης TimeseriesGenerator σε προγραμματιστικό περιβάλλον Python.

Στα δεδομένα εισόδου της συνάρτησης εισάγουμε την χρονοσειρά στο πεδίο *data*, στο πεδίο *targets* μπαίνουν τα δεδομένα στόχου δηλαδή το αποτέλεσμα που θα θέλαμε να βγάλει το μοντέλο στην περίπτωση που του δοθούν τα δεδομένα (*data*). Στο πεδίο sampling_rate βάζουμε το πλήθος των δεδομένων που θέλουμε να πάρουμε πριν την πρόβλεψη. Στην δική μας περίπτωση επιλέγουμε το μήκος των οκτώ τριμήνων δηλαδή δύο ετών ώστε το μοντέλο να μπορεί να ανιχνεύσει το μοτίβο μεταξύ των τριμήνων. Τέλος η μεταβλητή *batch_size* χωρίζει συνολικά τα δεδομένα σε παρτίδες (*batches*), στην περίπτωση του παραπάνω κώδικα σε 2. Αυτό έχει ως στόχο την εκπαίδευση του μοντέλου σταδιακά σε μικρά κομμάτια δεδομένων βελτιστοποιώντας καλύτερη σύγκλιση. Στην δικιά μας περίπτωση που δεν έχουμε πολλά δεδομένα η μεταβλητή *batch_size* επιλέγεται ίση με την μονάδα καθώς δεν θα είχε νόημα η εκπαίδευση του μοντέλου με ένα προς ένα δεδομένο.

Ο καταλληλότερος αριθμός δέντρων επιλέγεται και πάλι με την εξαντλητική μέθοδο (συγκρίνουμε την απόδοση του μοντέλου συγκριτικά με τον αριθμών των τυχαίων δέντρων.



Διάγραμμα 5-6: Διάγραμα απόδοσης Τυχαίου Δάσους συναρτήσει του αριθμού των δέντρων

Όπως μπορούμε να παρατηρήσουμε η απόδοση του μοντέλου δεν επηρεάζεται σχεδόν καθόλου από το πλήθος των τυχαίων δέντρων. Η πολυπλοκότητα του μοντέλου αυξάνεται γραμμικά με την αύξηση των

τυχαίων δέντρων, όπως και ο χρόνος εκπαίδευσης και εκτέλεσης του προγράμματος. Για αυτό τον λόγο για το παρόν πρόβλημα προτείνεται η επιλογή μικρού πλήθους τυχαίων δέντρων (50-100). Ένας ακόμα λόγος που φαίνεται να έχουμε μείωση του σφάλματος (σχεδόν αμελητέα παρ' όλα αυτά) μπορεί να είναι ο μικρότερος αριθμός των παραμέτρων που καλούμαστε να προσαρμόσουμε για την επίλυση του προβλήματος. Ο μικρός αριθμός δεδομένων δεν μπορεί να εκπαιδεύσει μεγάλο αριθμό παραμέτρων.

Η ανάλυση των άλλων παραμέτρων του μοντέλου ξεφεύγει από τα όρια της εργασίας αυτής, το μόνο που θα γίνει είναι μια αναφορά στις πιο σημαντικές, για την καλύτερη κατανόηση του αποτελέσματος.

Για την επιβολή περιορισμών στην αρχιτεκτονική των τυχαίων δέντρων θα πρέπει να ρυθμιστούν οι παράμετροι: *max_depth* όπου ρυθμίζει το "βάθος" του δέντρου, *min_samples_split* όπου είναι ο ελάχιστος αριθμός κόμβων που μπορεί να έχει ένα τυχαίο δάσος ενώ *min_samples_leaf* ο ελάχιστος αριθμός φύλλων (τελευταίος κόμβος πριν το τέλος). Γενικά στην δικιά μας περίπτωση οι παράμετροι δεν πειράχτηκαν και το μοντέλο λειτούργησε με τις προεπιλεγμένες τιμές των παραπάνω παραμέτρων (Random Forest, 2022).

5.1.6: Νευρωνικά δίκτυα (Neural Networks)

Τα τεχνητά νευρωνικά δίκτυα Artificial Neural Networks (ANN), που συνήθως ονομάζονται απλά νευρωνικά δίκτυα, είναι υπολογιστικά συστήματα εμπνευσμένα από τα βιολογικά νευρωνικά δίκτυα που αποτελούν τους εγκεφάλους κάποιων ζωντανών οργανισμών (π.χ. ανθρώπινος εγκέφαλος).

Για να μπορέσουμε να καταλάβουμε τον τρόπο λειτουργίας ενός νευρωνικού δικτύου θα πρέπει πρώτα να καταλάβουμε την λειτουργία ενός μόνο νευρώνα, γνωστός και ως νευρώνας αντίληπτρο (Perceptron). Ο perceptron είναι ένας δυαδικός ταξινομητής. Ορίζεται από μια συνάρτηση όπου έχει δεδομένα εισόδου το διάνυσμα $\vec{x}$ και μια και μοναδική τιμή εξόδου(στην περίπτωση που έχουμε πρόβλημα ταξινόμησης η τιμή είναι υποχρεωτικά δυαδική 0 η 1). Η συνάρτηση που περιγράφει την διαδικασία αυτή είναι:

$$f(\vec{x}) = \begin{cases} 1, if \vec{w} \cdot \vec{x} + b > 0 \\ 0, \quad else \end{cases} \tag{5.10}$$

όπου: w το διάνυσμα από βάρη πραγματικών αριθμών, η πράξη $\vec{w} \cdot \vec{x}$ είναι το εσωτερικό γινόμενο των διανυσμάτων και το b είναι ένας σταθερός αριθμός (bias). Γενικά ο νευρώνας Perceptron χρησιμοποιείται ως γραμμικός ταξινόμητης (linear classifier). Στην περίπτωση που θέλουμε να τον χρησιμοποιήσουμε για προβλήματα παλινδρόμησης η συνάρτηση έχει μια μικρή τροποποίηση:

$$f(\vec{x}) = \vec{w} \cdot \vec{x} + b \tag{5.11}$$

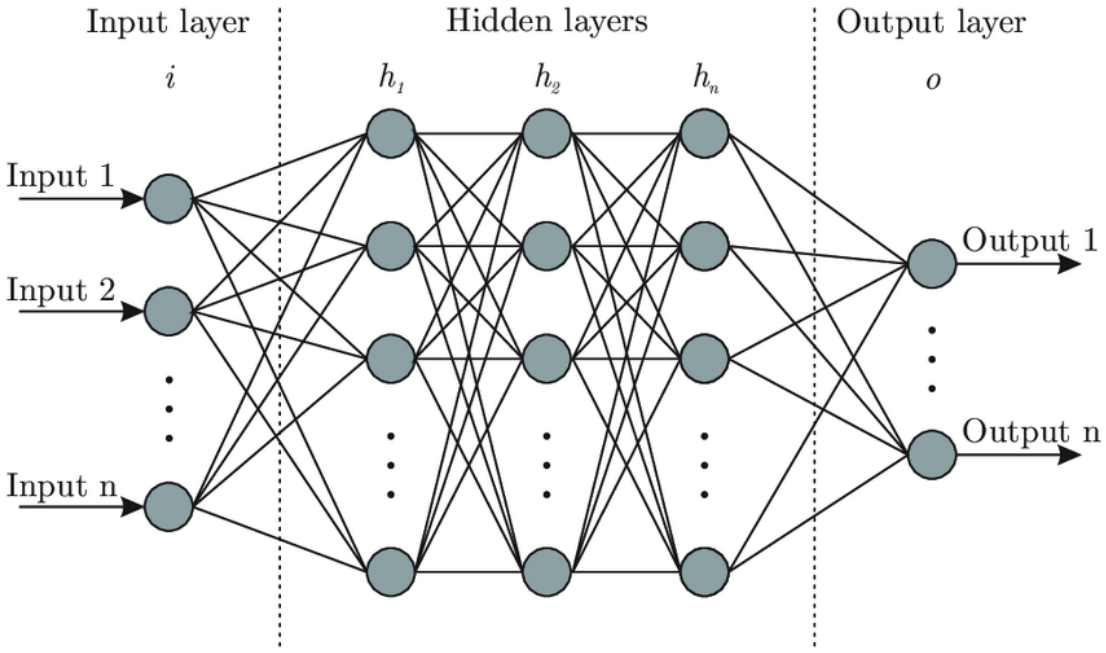
Η απόδοση του μοντέλου εξαρτάται αποκλειστικά από τους συντελεστές $\vec{w}$ και b. Για αυτό τον λόγο θέλουμε οι συντελεστές αυτοί να προσαρμοστούν όσο το δυνατόν καλύτερα για τα δεδομένα που διαθέτουμε για εκπαίδευση. Για την περίπτωση προβλήματος παλινδρόμησης υπάρχουν αναλυτικές μεθοδολογίες για την ακριβή εύρεση των καλύτερων παραμέτρων $\hat{w}$ και $\hat{b}$. Με την βοήθεια της κλασικής μεθοδολογίας μηδενισμού των μερικών παραγώγων προκύπτει το παρακάτω σύστημα για την περίπτωση όπου η διάσταση του $\vec{x}$ είναι μονάδα (μέθοδος ελαχίστων τετραγώνων):

$$w = \frac{m\sum x_i y_i - \sum x_i \sum y_i}{m\sum {x_i}^2 - (\sum x_i)^2} \tag{5.12}$$

$$b = \frac{\sum {x_i}^2 \sum y_i - \sum x_i \sum x_i y_i}{m\sum {x_i}^2 - (\sum x_i)^2} \tag{5.13}$$

Όπου m ο αριθμός των διαθέσιμων δεδομένων για εκπαίδευση, x_i οι ανεξάρτητες και y_i οι εξαρτημένες μεταβλητές.

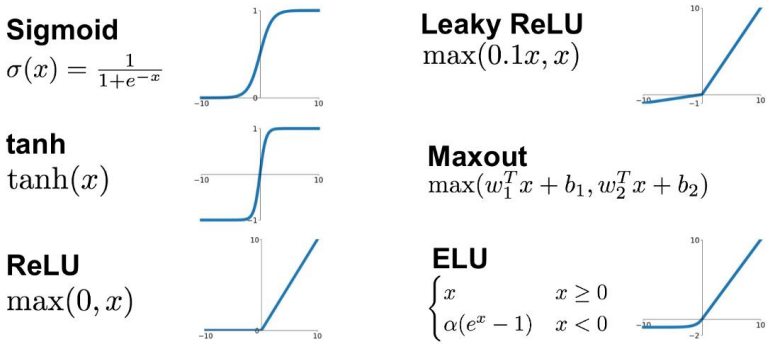
Ένα τεχνητό νευρωνικό δίκτυο (ANN) βασίζεται σε μια συλλογή συνδεδεμένων νευρώνων, οι οποίοι μοντελοποιούνται χαλαρά σε έναν βιολογικό εγκέφαλο. Κάθε σύνδεση, όπως οι συνάψεις σε έναν βιολογικό εγκέφαλο, μπορεί να μεταδώσει ένα σήμα σε άλλους νευρώνες. Ένας τεχνητός νευρώνας λαμβάνει ένα σήμα και στη συνέχεια το επεξεργάζεται και μπορεί να σηματοδοτήσει τους νευρώνες που συνδέονται με αυτόν. Το "σήμα" σε μια σύνδεση είναι ένας πραγματικός αριθμός και η έξοδος κάθε νευρώνα υπολογίζεται από κάποια μη γραμμική συνάρτηση του αθροίσματος των εισόδων του. Οι συνδέσεις ονομάζονται ακμές. Οι νευρώνες και τα άκρα έχουν συνήθως ένα βάρος που προσαρμόζεται καθώς προχωρά η μάθηση. Το βάρος αυξάνει ή μειώνει την ισχύ του σήματος σε μια σύνδεση. Οι νευρώνες μπορεί να έχουν ένα κατώφλι τέτοιο ώστε ένα σήμα να αποστέλλεται μόνο εάν το αθροιστικό σήμα υπερβαίνει αυτό το κατώφλι. Τυπικά, οι νευρώνες συγκεντρώνονται σε στρώματα.



Εικόνα 5-5: Κλασική αρχιτεκτονική νευρωνικού δικτύου με n ενδιάμεσα στρώματα.

Ένα νευρωνικό δίκτυο αποτελείται από πολλά στρώματα (*layers*). Πιο συγκεκριμένα από το στρώμα εισαγωγής το οποίο είναι υπεύθυνο για την εισαγωγή των δεδομένων στο μοντέλο (ο αριθμός των

κόμβων ισούται με τον αριθμό τον δεδομένων στην είσοδο), τα κρυφά στρώματα (*hidden layers*) τα οποία επιλέγονται από τον χρήστη και είναι υπεύθυνα για διάφορους μηχανισμούς ανάλογα με το πρόβλημα (οι κόμβοι των κρυφών στρωμάτων επιλέγονται και αυτές από τον χρήστη) και -τέλος- το τελευταίο στρώμα (εξόδου) (*output layer*) που παράγει το αποτέλεσμα του μοντέλου (το πλήθος των κόμβων εξαρτάται αποκλειστικά από τον αριθμό των εξόδων που θέλουμε να έχουμε).Η διάταξη φαίνεται καλύτερα στην Εικόνα 5-5. Στην περίπτωση που έχουμε πρόβλημα πρόβλεψης θα έχουμε μόνο ένα κόμβο στο τέλος, ενώ στην περίπτωση που είχαμε πρόβλημα ταξινόμησης ο αριθμός θα είναι ίσος με τον αριθμό των κλάσεων. Οι κόμβοι μεταξύ τους συνδέονται με ένα βάρος όπως και στην περίπτωση του μονού νευρώνα. Η μόνη διαφορά σε σχέση με τον μοναδικό νευρώνα είναι ότι το αποτέλεσμα θα περνάει μέσα από μια συνάρτηση ενεργοποίησης. Η λειτουργία της συνάρτησης ενεργοποίησης είναι να παρέχει μη γραμμικότητα στο μοντέλο μας, αφού ο γραμμικός συνδυασμός δύναται να περιγράψει μόνο γραμμικά φαινόμενα. Στην εικόνα παρακάτω βλέπουμε τις πιο συνηθισμένες συναρτήσεις ενεργοποίησης που χρησιμοποιούνται.



Διάγραμμα 5-7:Οι πιο γνωστές συναρτήσεις ενεργοποίησης Πηγή: Researchgate.net

Το αποτέλεσμα κάθε κόμβου i στο στρώμα j υπολογίζεται από την σχέση:

$$f_i^{(j)} = g(z), z = w_{0j} + \sum_{i=1}^{d} x_i w_i \quad (5.14)$$

όπου :

- g η συνάρτηση ενεργοποίησης
- x_i η τιμή του προηγούμενου κόμβου
- w_i το βάρος μεταξύ των κόμβων
- w_{0j} η σταθερά (*bias*)

Για την επίλυση προβλήματος ταξινόμησης θα πρέπει στο στρώμα εξόδου να εισάγουμε μια συνάρτηση ενεργοποίησης για να δημιουργήσουμε την πιθανότητα που έχει ένα στοιχείο να ανήκει στην κλάση. Η συνάρτηση αυτή από την βιβλιογραφία (GoodFellow, Bengio, Courville, & Aaron, 2016) ονομάζεται *softmax* και περιγράφεται:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5.15}$$

όπου :

- z_i το αποτέλεσμα στον τελικό στρώμα του κόμβου i
- $\sigma(\vec{z})_i$ η πιθανότητα το $\vec{z}$ να ανήκει στην υποομάδα i

Όπως και στην περίπτωση του απλού νευρώνα στόχος μας είναι να βρεθούν οι βέλτιστοι συντελεστές *w* για τα δεδομένα που έχουμε, ώστε να μπορούμε να κάνουμε όσο το δυνατόν καλύτερες προβλέψεις. Για την διαδικασία της εκπαίδευσης θα πρέπει να θεωρήσουμε ένα σφάλμα το όποιο θα λειτουργεί σαν οδηγός ως προς την βέλτιστη λύση. Οι πιο διαδεδομένοι δείκτες είναι:

- Μέσο Τετραγωνικό Σφάλμα(για προβλήματα παλινδρόμησης):

$$L = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{5.16}$$

όπου :

- y_i το αποτέλεσμα που θα θέλαμε να είχαμε.
- $\hat{y}_i$ το αποτέλεσμα που βγάζει το μοντέλο.
- N το σύνολο των δεδομένων.

- Σφάλμα διασταυρούμενης εντροπίας(για προβλήματα πιθανοτήτων):

$$L = \frac{-1}{N}\sum_{i=1}^{N} y_i log\hat{y}_i \tag{5.17}$$

όπου :

- y_i το αποτέλεσμα που θα θέλαμε να είχαμε.
- $\hat{y}_i$ το αποτέλεσμα που βγάζει το μοντέλο.
- N το σύνολο των δεδομένων.

Λόγω του μεγάλου αριθμού παραμέτρων η αναλυτική εύρεση των συντελεστών, όπως στο παράδειγμα του απλού νευρώνα είναι αδύνατη. Η επίλυση γίνεται με την μέθοδο στοχαστικής κλίσης κατάβασης όπως περιγράφεται στο δεύτερο κεφάλαιο.

$$w_{n+1} = w_n - \alpha\nabla J(\overrightarrow{x}, \overrightarrow{w_n}) \tag{5.18}$$

όπου :

- w_n η παράμετροι στο βήμα n.
- $\nabla J(\overrightarrow{x}, \overrightarrow{w_n})$ η παράγωγος της συνάρτησης κόστους για τα δεδομένα $\vec{x}$.
- α ο ρυθμός μάθησης (*learning rate*)

Η παράγωγος της σχέσης 5.18 στην συγκεκριμένη περίπτωση υπολογίζεται με την βοήθεια του κανόνα της αλυσίδας για κάθε συντελεστή βάρους και σταθερού όρου που καλούμαστε να βελτιστοποιήσουμε.

Εικόνα 5-6: Διαδοχικοί μονογραμμικοί νευρώνες. Πηγή: edX.org

Στην απλή περίπτωση όπου έχουμε ένα δίκτυο από πολλούς μονούς νευρώνες όπου x η είσοδος του μοντέλου, L το πλήθος των ενδιάμεσων στρωμάτων, w_i το βάρος κάθε νευρώνα f_i η συνάρτηση ενεργοποίησης και $L(g(x), \hat{y})$ η συνάρτηση σφάλματος. Η συνάρτηση που υπολογίζεται είναι:

$$g(x) = f_L \rightarrow f_{L-1} \cdot w_L \rightarrow f_{L-2} \cdot w_{L-1} \cdot w_L \dots \tag{5.19}$$

όπου:

- f_n το αποτέλεσμα της συνάρτησης ενεργοποίησης στο στρώμα n.
- w_n το βάρος της σύνδεσης στο στρώμα n.

Στην περίπτωση όπου η συνάρτηση ενεργοποίησης είναι η $tanh$ και συνάρτηση σφάλματος:

$$L = (y - f_L)^2 \tag{5.20}$$

όπου:

- f_L το αποτέλεσμα την συνάρτησης ενεργοποίησης στο τελευταίο στρώμα δηλαδή το αποτέλεσμα.
- y το αποτέλεσμα που θα θέλαμε να είχαμε.

Η παράγωγος της συνάρτησης ενεργοποίησης είναι:

$$f(x) = tanh(x) \tag{5.21}$$

$$\acute{f}(x) = \left(1 - tanh^2(x)\right) \rightarrow (1 - f(x)^2) \tag{5.22}$$

Η παράγωγος της συνάρτησης σφάλματος είναι:

$$\frac{dL}{df_L} = -2(y - f_L) \tag{5.23}$$

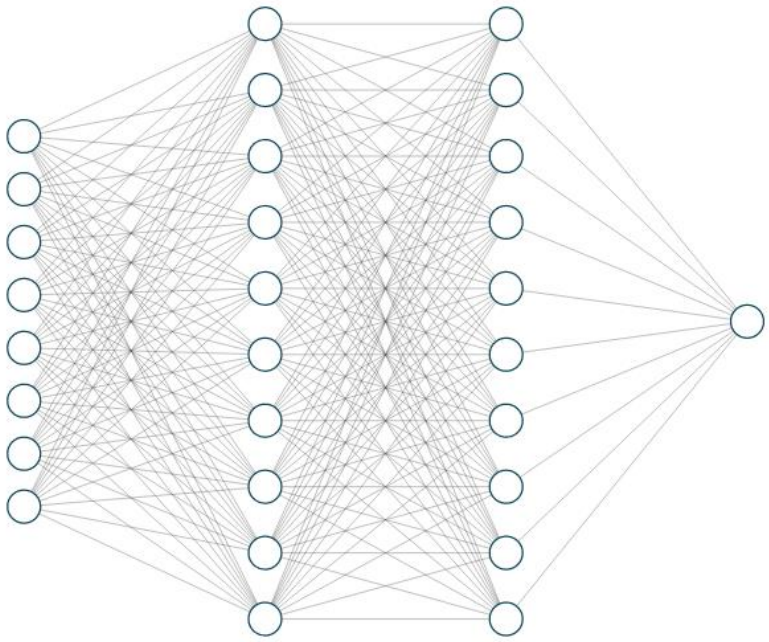
Για την μερική παράγωγο του σφάλματος με το βάρος του πρώτου νευρώνα:

$$\frac{\partial L}{\partial w_1} = x \cdot \frac{\partial L}{\partial z_1} \rightarrow \frac{\partial L}{\partial w_1} = x \cdot (1 - f_1)^2 \cdot w_2 \frac{\partial L}{\partial z_2} \rightarrow \dots \rightarrow \tag{5.24}$$

$$\rightarrow \frac{\partial L}{\partial w_1} = x \cdot (1 - f_1)^2 (1 - f_2)^2 \cdots (1 - f_L)^2 \cdot w_2 w_3 \cdots w_L \cdot 2(f_L - y) \tag{5.25}$$

Παρόμοια διαδικασία γίνεται και για τις άλλες μεταβλητές για την εφαρμογή ενός βήματος της στοχαστικής κλίσης κατάβασης.

Στα πλαίσια της εργασίας δοκιμάστηκαν δυο διαφορετικές αρχιτεκτονικές νευρωνικών δικτύων. Το στρώμα εισόδου και στα δύο μοντέλα αποτελούνται από οχτώ (8) κόμβους, όπως και στην περίπτωση των τυχαίων δέντρων (*Random-Forest*). Ο αριθμός δεν είναι τυχαίος καθώς αντιστοιχεί σε περίοδο δύο χρόνων, ευελπιστώντας ότι το μοντέλο θα μπορέσει να ανιχνεύσει την περιοδικότητα του φαινομένου. Και τα δύο μοντέλα έχουν από δύο ενδιάμεσα στρώματα, το ένα με δέκα (10) κόμβους και το άλλο με τριάντα (30) και αφού έχουμε πρόβλημα παλινδρόμησης θα έχουμε τελικό στρώμα με μόνο ένα κόμβο (δηλαδή την πρόβλεψη). Η επιλογή μικρού αριθμού κόμβων γίνεται για την αποφυγή ενώς πολύ σύνθετου μοντέλου δεδομένου ότι έχουμε πολύ μικρό αριθμό δεδομένων για εκπαίδευση. Η αρχιτεκτονική του μοντέλου για την περίπτωση φαίνεται παρακάτω:



Εικόνα 5-7: Η Αρχιτεκτονική του Νευρωνικού Δικτύου που καλούμαστε να εκπαιδεύσουμε.

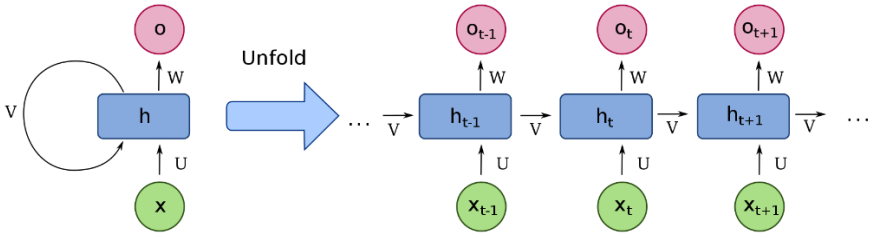
Οι παράμετροι που πρέπει να βελτιστοποιηθούν από το πρώτο στρώμα είναι ογδόντα μία (81) στο δεύτερο εκατό μία (101) και στο τελευταίο στρώμα έντεκα (11) δηλαδή στο σύνολο εκατό ενενήντα τρείς (193), μαζί με τους σταθερούς όρους. Κατά την εκπαίδευση χρησιμοποιήθηκε συνάρτηση σφάλματος μέσου τετραγωνικού σφάλματος και τρόπος βελτιστοποίησης στοχαστικής κλίσης κατάβασης Adam, καθώς είναι οι πιο ευρέως διαδεδομένοι παράμετροι για προβλήματα παλινδρόμησης (GoodFellow, Bengio, Courville, & Aaron, 2016)

5.1.7: Ανατροφοδοτούμενα Νευρωνικά δίκτυα (Recurrent Neural Networks)

Ένα ανατροφοδοτούμενο νευρωνικό δίκτυο (Recurrent Neural Network-RNN) είναι μια κατηγορία τεχνητών νευρωνικών δικτύων όπου οι συνδέσεις μεταξύ κόμβων σχηματίζουν ένα κατευθυνόμενο ή μη κατευθυνόμενο γράφο κατά μήκος μιας χρονικής ακολουθίας. Αυτό του επιτρέπει να επιδεικνύει χρονική δυναμική συμπεριφορά. Προερχόμενα από τροφοδοτικά νευρωνικά δίκτυα, μπορούν να χρησιμοποιήσουν την εσωτερική τους κατάσταση (μνήμη) για να επεξεργαστούν ακολουθίες μεταβλητού μήκους εισόδων. Αυτό τις καθιστά εφαρμόσιμες σε εργασίες όπως αναγνώριση προτύπων, αναγνώριση φωνής και γραφής. Τα ανατροφοδοτούμενα νευρωνικά δίκτυα είναι μοντέλα ικανά να ικανοποιήσουν τα κριτήρια Turing.

Ο όρος "ανατροφοδοτούμενο νευρωνικό δίκτυο" αναφέρεται στην κατηγορία δικτύων με άπειρη παλμική απόκριση, ενώ το "συνελικτικό νευρωνικό δίκτυο" αναφέρεται στην κατηγορία της απόκρισης πεπερασμένων παλμών. Και οι δύο κατηγορίες δικτύων παρουσιάζουν χρονική δυναμική συμπεριφορά. Ένα επαναλαμβανόμενο δίκτυο πεπερασμένου παλμού είναι ένας κατευθυνόμενος άκυκλος γράφος που μπορεί να ξετυλιχτεί και να αντικατασταθεί με ένα νευρωνικό δίκτυο αυστηράς τροφοδοσίας, ενώ ένα επαναλαμβανόμενο δίκτυο άπειρων παλμών είναι ένας κατευθυνόμενος κυκλικός γράφος που δεν μπορεί να ξετυλιχτεί.

Τόσο τα πεπερασμένα παλμικά όσο και τα ανατροφοδοτούμενα δίκτυα άπειρων παλμών μπορούν να έχουν πρόσθετες αποθηκευμένες καταστάσεις και η αποθήκευση μπορεί να είναι υπό άμεσο έλεγχο από το νευρωνικό δίκτυο. Ο χώρος αποθήκευσης μπορεί επίσης να αντικατασταθεί από άλλο δίκτυο ή γράφημα εάν ενσωματώνει χρονικές καθυστερήσεις ή έχει βρόχους ανάδρασης. Τέτοιες ελεγχόμενες καταστάσεις αναφέρονται ως πυλωτή κατάσταση ή κλειστή μνήμη και αποτελούν μέρος των δικτύων μακράς βραχυπρόθεσμης μνήμης (*LSTM*) και των περιφραγμένων επαναλαμβανόμενων μονάδων. Ο τρόπος δόμησης ενός τέτοιου μοντέλου φαίνεται στην επόμενη εικόνα.

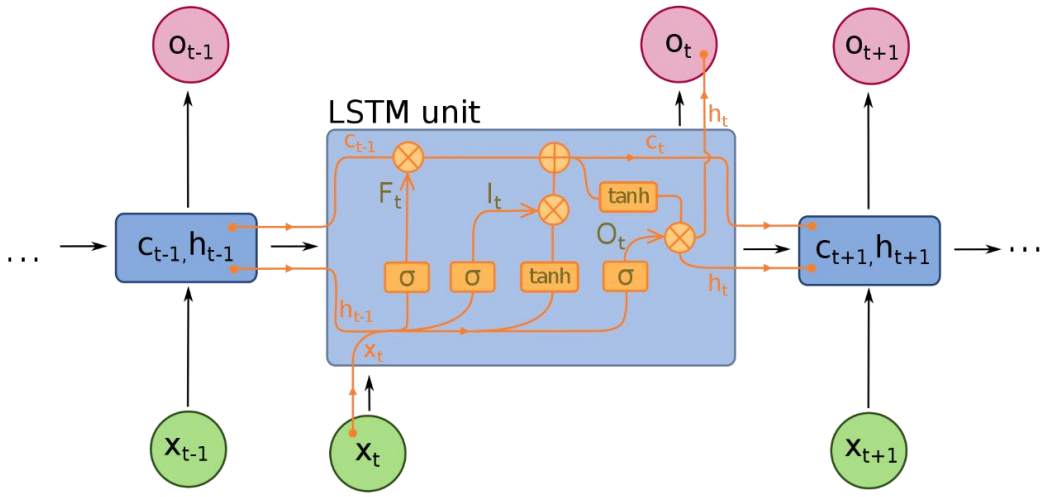


Εικόνα 5-8: Διάγραμμα εκτέλεσης Ανατροφοδοτούμενου Νευρωνικού Δικτύου.Πηγή: Wikipedia.com

Όπου h_t το νευρωνικό δίκτυο που ανατροφοδοτείται σε κάθε βήμα έχοντας 2 ορίσματα εξόδου: το o_t που είναι το αποτέλεσμα του νευρωνικού για την παρούσα χρονική στιγμή και το h_t όπου είναι η πληροφορία που ρέει από το ένα νευρωνικό στο επόμενο.

Η πιο γνωστή κατηγορία ανατροφοδοτούμενων νευρωνικών δίκτυών -κρίνοντας από την αποτελεσματικότητα με την οποία μπορούν να αποδώσουν- είναι τα Δίκτυα Μακράς Βραχύχρονης Μνήμης (*Long Short Term Memory- LSTM*). Η επινόηση των Δικτύων Μακράς Βραχύχρονης Μνήμης έγινε

για την επίλυση των προβλημάτων των κλασσικών επανατροφοδοτούμεων νευρωνικών δικτύων. Το κύριο πρόβλημα προερχόταν από την φύση του κατά την εκπαίδευση μεγάλων χρονικών στιγμών. Το βήμα της οπισθοδιάδοσης (*backpropagation*) δεν είναι αποτελεσματικό. Ο κανόνας της αλυσίδας υπολογίζει την μερική παράγωγο της συνάρτησης σφάλματος σε μορφή γινομένου πολλών παραγώγων. Παρόλα τα θετικά που προβλέπονται από αυτήν την διαδικασία όπως απλότητα και ταχύτητα υπολογισμών, τα μεγάλα γινόμενα εμπεριέχουν τον κίνδυνο απότομου μηδενισμού. Καθώς μόνο ένα από αυτά τα γινόμενα χρειάζεται να μηδενιστεί για να μηδενίσει όλο το γινόμενο. Η πιθανότητα να γίνει κάτι τέτοιο προφανώς είναι ανάλογη με το πόσο βαθύ είναι το δίκτυο. Στην επιστημονική βιβλιογραφία το πρόβλημα αυτό αναφέρεται ως πρόβλημα εξαφανιζόμενων κλίσεων (*vanishing gradient problem*). Η διαδικασία εκπαίδευσης των LSTM λύνει εν μέρει το πρόβλημα αυτό καθώς επιτρέπει τις κλίσεις να μην μεταβάλλονται απαραίτητα (Hochreiter & Schmidhuder, 1997).



Εικόνα 5-9: Εσωτερική Αρχιτεκτονική μιας μονάδας LSTM. Πηγή Wikipedia.com

Στην εικόνα προηγουμένως βλέπουμε την αρχιτεκτονική μίας βασικής μονάδας LSTM. Αυτή η εικόνα ουσιαστικά είναι ο τρόπος λειτουργείας των h_t στην εικόνα 5-8. Η βασική μονάδα περιέχει τις πύλες εισόδου όπου εισάγεται η νέα πληροφορία i_t , την πύλη λήθης (*forget gate*) που είναι υπεύθυνη για την διαγραφή μίας άχρηστης πληροφορίας , f_t, και -τέλος- την πύλη εξόδου (*output gate*) όπου είναι υπεύθυνη για την έξοδο των αποτελεσμάτων. Το αποτέλεσμα y_t τροφοδοτείται πάλι στην επόμενη ως μεταβλητή h_t μαζί με την μεταβλητή C_t που αποτελεί την μνήμη από προηγούμενες μονάδες. Όπως μπορούμε να δούμε και στο παραπάνω διάγραμμα το κάθε στρώμα έχει και την δικιά του συνάρτηση ενεργοποίησης: στην κλασική περίπτωση όλες οι πύλες έχουν την συνάρτηση *sigmoid* εκτός από την πύλη ενεργοποίησης $\hat{C}_t$ όπου χρησιμοποιείται η συνάρτηση υπερβολικής εφαπτομένης (*hyperbolic tangent*) . Τέλος για προβλήματα παλινδρόμησης όπως και στα κλασικά νευρωνικά δίκτυα θα πρέπει να έχουμε στο στρώμα της εξόδου ένα μόλις κόμβο όπου θα βγάζει το αποτέλεσμα. Οι πράξεις που πραγματοποιούνται σε κάθε πύλη είναι:

$$
\begin{aligned}
f_t &= \sigma_g\big(W_f x_t + U_f h_{t-1} + b_f\big) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
\hat{C}_t &= \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \hat{C}_t \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
\tag{5.26}
$$

όπου:

- $\circ$ πολλαπλασιασμός μητρώου στοιχείο με στοιχείο
- $+$ άθροιση μητρώου στοιχείο με στοιχείο

Στα πλαίσια της εργασίας εξετάστηκαν δύο σενάρια ανάλογα με τον αριθμό των κόμβων που απαρτίζουν κάθε στρώμα των πυλών. Πιο συγκεκριμένα τα δύο σενάρια είναι για εκατό (100) και τριάντα (30) κόμβους. Ο λόγος της επιλογής των σεναρίων δεν είναι καθόλου τυχαίος καθώς δοκιμάζουμε την ανταπόκριση από μοντέλα μικρής πολυπλοκότητας και μέτριας. Τα μοντέλα μεγάλης πολυπλοκότητας είναι καταδικασμένα να μην δουλέψουν λόγο της μικρής ποσότητας δεδομένων για την εκπαίδευση.

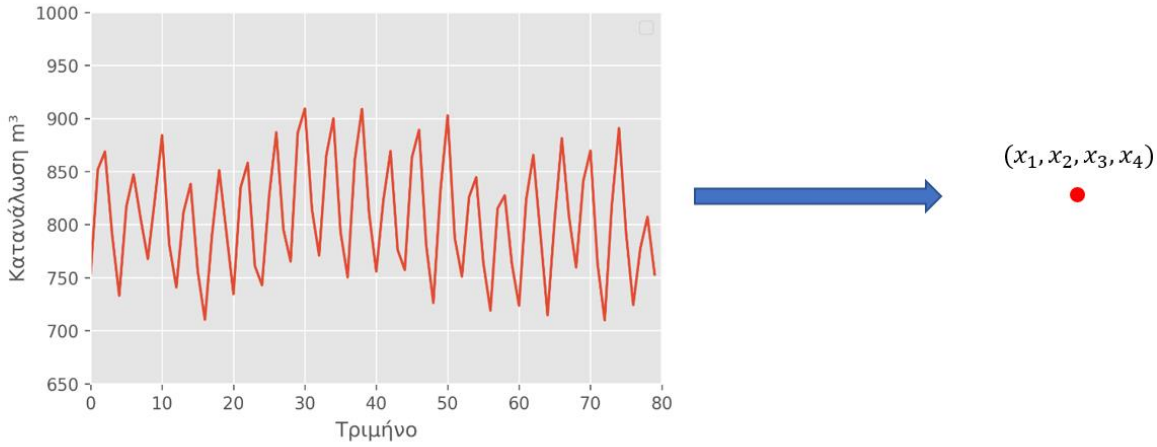
5.2: Κατακόρυφες μεθοδολογίες

Σε αντίθεση με τις μεθοδολογίες που εξετάστηκαν παραπάνω, όπου η πληροφορία προχωράει κατά μήκος της χρονοσειράς, οι κατακόρυφες μεθοδολογίες εκμεταλλεύονται τις μετρήσεις από πελάτες οι οποίες έχουν ήδη πραγματοποιηθεί (εκμετάλλευση των ελλιπών μετρήσεων που συναντάει η ΕΥΔΑΠ από τους εργολάβους μέτρησης). Στην συγκεκριμένη περίπτωση η πληροφορία ρέει κάθετα στην χρονοσειρά, δηλαδή από τον ένα πελάτη στον άλλο, για αυτό και οι σχετικές μεθοδολογίες χαρακτηρίστηκαν ως κάθετες.

5.2.1: Διαδικασία κωδικοποίησης χρονοσειράς

Οι μεθοδολογίες αυτές λειτουργούν με την διαδικασία ομαδοποίησης (*clustering*). Οι χρονοσειρές περιέχουν μεγάλου μήκους πληροφορία, στις συνθετικές χρονοσειρές για παράδειγμα έχουμε ανάπτυγμα εξήντα τεσσάρων τιμών. Γενικά κατά την ανάλυση των δεδομένων η επιστημονική βιβλιογραφία προτείνει η διάσταση μελέτης να μην είναι πολύ μεγάλη (Yiu, 2019). Το κοινό θέμα αυτών των προβλημάτων είναι ότι όταν αυξάνεται η διάσταση, ο όγκος του χώρου αυξάνεται τόσο γρήγορα που τα διαθέσιμα δεδομένα γίνονται αραιά. Προκειμένου να επιτευχθεί ένα αξιόπιστο αποτέλεσμα, ο όγκος των δεδομένων που απαιτούνται συχνά αυξάνεται εκθετικά με τη διάσταση. Επίσης, η οργάνωση και η αναζήτηση δεδομένων συχνά βασίζεται στον εντοπισμό περιοχών όπου τα αντικείμενα (*features*) σχηματίζουν ομάδες με παρόμοιες ιδιότητες. Σε δεδομένα υψηλών διαστάσεων, ωστόσο, όλα τα αντικείμενα φαίνεται να είναι αραιά και ανόμοια από πολλές απόψεις, γεγονός που εμποδίζει τις κοινές στρατηγικές οργάνωσης δεδομένων να είναι αποτελεσματικές. Το φαινόμενο αυτό ονομάζεται η κατάρα

της διαστατικότητας (curse of dimensionality). Για αυτό τον λόγο επιλέγεται η ανάλυση στο χώρο R^4. Η επιλογή δεν είναι αυθαίρετη καθώς τέσσερις είναι και οι εποχές (τρίμηνα) με βάση την εποχικότητα των δεδομένων που διαθέτουμε. Η διαδικασία αυτή γίνεται παρόμοια με το μοντέλο Naive με βάρη. Κάθε διάσταση απεικονίζει μια εποχή, όπου υπολογίζεται ως άθροισμα των τελευταίων χρόνων με τον δικό του βάρος . Πιο απλά κάθε διάσταση αντιπροσωπεύει την μέση τιμή της χρονοσειράς δίνοντας έμφαση στα πιο πρόσφατα δεδομένα.Με αυτό τον τρόπο καταφέρνουμε να ανάγουμε ολόκληρη την χρονοσειρά σε ένα σημείο στον χώρο R^4.



Εικόνα 5-10: Μετατροπή χρονοσειράς σε ένα σημείο στον R^4

Πιο συγκεκριμένα ο κάθε x_i υπολογίζεται από την ακόλουθη σχέση, σε χρονοσειρά y_t με μήκος k:

$$x_i = \sum_{n=0}^{N} a_n \, y_{k-nT} \tag{5.27}$$

όπου:

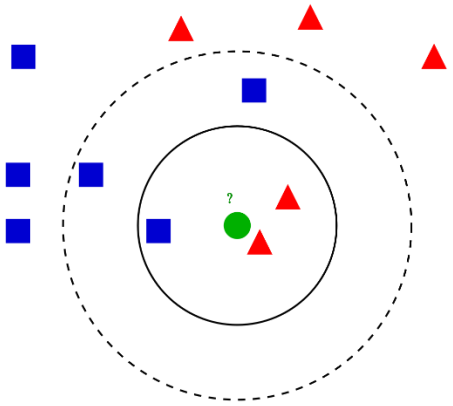
- a_n ο συντελεστής βαρύτητας για την n χρονιά.
- y_i η τιμή την χρονοσειράς την χρονική στιγμή i και k ο δείκτης της τελευταίας τιμής.

Όπως και στην περίπτωση της μεθόδου Naive το άθροισμα των συντελεστών a_n πρέπει να ισούται με την μονάδα.

Οι συντελεστές a_n επηρεάζουν πολύ την ποιότητα της κωδικοποίησης. Τα μοντέλα που εξετάζονται παρακάτω είναι αρκετά περίπλοκα και θέλουν πολύ χρόνο για να τρέξουν, κάνοντας αδύνατη την επιλογή των συντελεστών με την εξαντλητική μέθοδο (*exhaustive*). Για αυτό επιλέγονται οι βελτιστοποιημένοι συντελεστές από την μέθοδο Naive (δεδομένου ότι εφόσον ήταν οι βέλτιστοι για την πρόγνωση θα είναι και βέλτιστοι για κωδικοποίηση) όπου είναι $\vec{\alpha} = (0.50, 0.25, 0.25)$. Η διαδικασία αυτή πραγματοποιείται για κάθε χρονοσειρά.

5.2.2: Μοντέλο πλησιέστερων γειτόνων (nearest neighbors)

Στην επιστήμη της στατιστικής, ο αλγόριθμος k-πλησιέστερων γειτόνων (k nearest neighbors k-NN) είναι μια μη παραμετρική μέθοδος ταξινόμησης που αναπτύχθηκε για πρώτη φορά από τους (Fix & Hodges, 1951). Χρησιμοποιείται για ταξινόμηση και παλινδρόμηση. Και στις δύο περιπτώσεις, η είσοδος αποτελείται από τα k πιο κοντινά παραδείγματα εκπαίδευσης σε ένα σύνολο δεδομένων. Στην ταξινόμηση k-NN, η έξοδος είναι μια ιδιότητα μέλους κλάσης. Ένα αντικείμενο ταξινομείται με πλήθος ψηφοφοριών των γειτόνων του, με το αντικείμενο να εκχωρείται στην κατηγορία που είναι πιο κοινή μεταξύ των k πλησιέστερων γειτόνων του (k είναι ένας θετικός ακέραιος, συνήθως μικρός). Αν $k = 1$, τότε το αντικείμενο απλώς εκχωρείται στην κλάση αυτού του απλού πλησιέστερου γείτονα. Στην παλινδρόμηση k-NN, η έξοδος είναι η τιμή ιδιότητας για το αντικείμενο. Αυτή η τιμή είναι ο μέσος όρος των τιμών των k πλησιέστερων γειτόνων. Το k-NN είναι ένας τύπος ταξινόμησης όπου η συνάρτηση προσεγγίζεται μόνο τοπικά και όλοι οι υπολογισμοί αναβάλλονται μέχρι την αξιολόγηση της συνάρτησης. Δεδομένου ότι αυτός ο αλγόριθμος βασίζεται στην απόσταση για ταξινόμηση, εάν τα χαρακτηριστικά αντιπροσωπεύουν διαφορετικές φυσικές μονάδες ή βρίσκονται σε πολύ διαφορετικές κλίμακες, τότε η κανονικοποίηση των δεδομένων εκπαίδευσης μπορεί να βελτιώσει δραματικά την ακρίβειά τους. Για αυτό τον λόγο ο αλγόριθμος είναι ευαίσθητος στην τοπική δομή των δεδομένων και πολλές φορές μπορεί να προβεί σε λανθασμένες αποφάσεις. Η φιλοσοφία λειτουργείας περιγράφεται με την βοήθεια του παρακάτω διαγράμματος.



Εικόνα 5-11: Διαφορά λειτουργείας του αλγορίθμου KNN ανάλογα με την τον αριθμό k.

Στον χώρο R^2 η ταυτότητα του πράσινου σημείου θα χαρακτηριστεί από την ταυτότητα των k κοντινών σημείων, για την περίπτωση όπου $k = 3$ το σημείο θα χαρακτηριζόταν κόκκινο ενώ αν $k = 5$ το σημείο θα χαρακτηριζόταν μπλε. Για αυτό τον λόγο θα πρέπει να γίνει κατάλληλη αναζήτηση για τον αριθμό k που θα επιλεχθεί για το μοντέλο αυτό. Η προγραμματιστική διαδικασία είναι πολύ απλή. Χωρίζεται σε δύο βήματα. Το πρώτο είναι η μέτρηση της απόστασης του σημείου που θέλουμε να εξετάσουμε με τα υπόλοιπα σημεία που έχουμε ως σύνολο μάθησης. Αυτό γίνεται με την εύρεση της μαθηματικής νόρμας της διαφοράς των δύο σημείων. Η πιο ευρέως διαδεδομένη νόρμα είναι η ευκλείδεια, δηλαδή η ευκλείδεια απόσταση στο καρτεσιανό επίπεδο. Η νόρμα υπολογίζεται ως ακολούθως:

$$\|\vec{x}\|_2 := \sqrt{{x_1}^2 + \cdots + {x_n}^2} \tag{5.28}$$

όπου:

- x_i η τιμή της διάστασης i του σημείου.

Η νόρμα του Μανχάταν είναι εμπνευσμένη από την χωροθέτηση των κτιρίων στην περιοχή του Μανχάταν της Νέα Υόρκης. Ο υπολογισμός γίνεται με το άθροισμα των απολύτων τιμών των διαστάσεων. Δηλαδή την απόσταση που θα διανύσει ένα ταξί στο κέντρο του Μανχάταν.

$$\|\vec{x}\|_1 = \sum_{i=1}^{N} |x_i| \tag{5.29}$$

όπου:

- x_i η τιμή της διάστασης i του σημείου.

Προφανώς η επιλογή της κάθε νόρμας έχει διαφορετική επίδραση στην συμπεριφορά του μοντέλου. Γενικά για προβλήματα μεγάλων διαστάσεων η επιστημονική βιβλιογραφία προτείνει την εφαρμογή της νόρμας Μανχάταν διότι είναι πιο σταθερή και δεν έχουμε φαινόμενα εξαφάνισης μικρών συνιστωσών (Zijie & Mengtian, 2020). Στα πλαίσια της εργασίας έγινε εξέταση μόνο της ευκλείδειας νόρμας για την πιο εύκολη κατανόηση και οπτικοποίηση του προβλήματος, ενώ είναι σημαντικό το γεγονός πως δεν έχουμε σημεία μεγάλης διάστασης (η ανάλυση έγινε στον R^4) όπως αναλύθηκε παραπάνω. Το δεύτερο βήμα είναι η κατάταξη των σημείων σε αύξουσα σειρά ανάλογα με τις αποστάσεις που υπολογίστηκαν στο προηγούμενο βήμα, και η επιλογή των k πρώτων, όπου k η υπερπαράμετρος των γειτόνων. Θα πρέπει να σημειωθεί στην περίπτωση όπου έχουμε την επίλυση προβλήματος ομαδοποίησης η υπερπαράμετρος k θα πρέπει να είναι περιττός αριθμός για να είναι αδύνατο να έχουμε ισοψηφία των ομάδων όπου ανήκουν οι γείτονες. Για την περίπτωση όπου έχουμε πρόβλημα παλινδρόμησης δεν έχουμε τέτοιο θέμα καθώς η τιμή θα υπολογιστεί από τον μέσο όρο των k κοντινότερων ή από μιας μορφής συντελεστή βαρύτητας ανάλογα με την απόσταση.

Για τα δεδομένα του προβλήματος μας έπρεπε να γίνει τροποποίηση της μεθοδολογίας για να μπορεί να ανταπεξέλθει στις απαιτήσεις των χρονοσειρών. Τα δεδομένα χωρίζονται σε δύο κατηγορίες: (i) στους πελάτες που γνωρίζουμε την τελευταία μέτρηση και (ii) στους πελάτες για τους οποίους μας είναι άγνωστη η τελευταία τιμή και θα θέλαμε να κάνουμε μια πρόγνωση. Οι πελάτες με γνωστή την τελευταία μέτρηση θα έχουν σημείο στο χώρο με την μορφή:

$$y_i = (x_1, x_2, x_3, x_4) \tag{5.30}$$

Ενώ για τους πελάτες που δεν γνωρίζουμε την τελευταία τιμή το σημείο στο χώρο θα ανήκει στον R^3 αφού η τελευταία τιμή είναι η τιμή που θέλουμε να προβλέψουμε και είναι άγνωστη. Το σημείο θα έχει την μορφή:

$$\hat{y}_i = (x_1, x_2, x_3) \tag{5.31}$$

Η ιδέα της τροποποίησης της μεθόδου είναι εμπνευσμένη από την λογική ότι οι πελάτες που έχουν σχεδόν ίδιες καταναλώσεις τους προηγούμενους μήνες έχουν μεγάλη πιθανότητα να έχουν σχεδόν κοινές καταναλώσεις τον τελευταίο μήνα. Ο υπολογισμός της απόστασης γίνεται από τα τρία πρώτα

τρίμηνα όπου έχουμε τιμές για όλους τους καταναλωτές. Αφού γίνει ο υπολογισμός των αποστάσεων και γίνει η ταξινόμηση κατά αύξουσα σειρά μπορούμε να υπολογίσουμε τις άγνωστες καταναλώσεις με τις δύο παρακάτω παραλλαγές.

(i) Η πρώτη παραλλαγή είναι με τον υπολογισμό του μέσου όρου των k πιο κοντινών γειτόνων. Ο τρόπος υπολογισμού περιγράφεται ως:

$$F = \frac{1}{K}\sum_{k=1}^{K} {x_4}^{(k)} \tag{5.32}$$

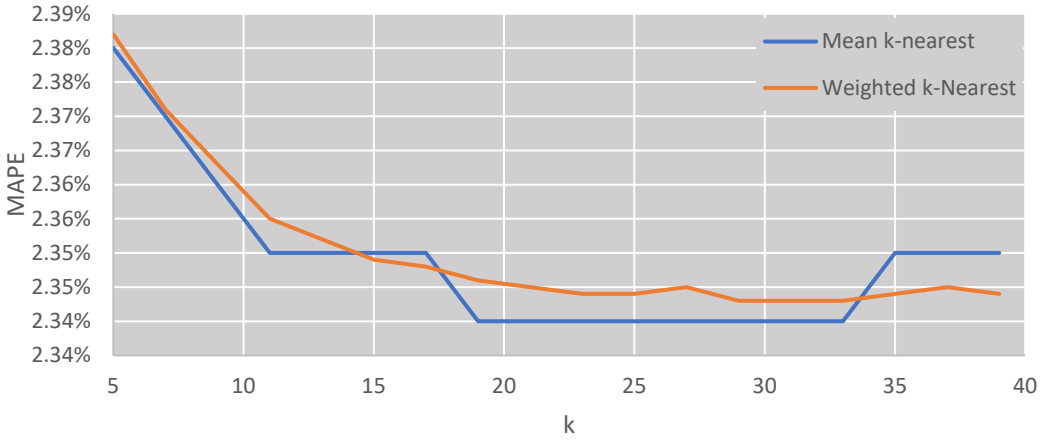
(ii) Η δεύτερη παραλλαγή δίνει μεγαλύτερη βαρύτητα στους καταναλωτές οι οποίοι είναι πιο κοντά στον καταναλωτή για τον οποίο θέλουμε να κάνουμε την πρόβλεψη. Ο συντελεστής βαρύτητας υπολογίζεται με την βοήθεια του νόμου των ανάστροφων αποστάσεων και περιγράφεται ως:

$$F = \sum_{k=1}^{K} \frac{{r_k}^{-n}}{R} {x_4}^{(k)}, \text{όπου} R = \sum_{k=1}^{K} {r_k}^{-n} \tag{5.33}$$

Όπου:

- n, ο βαθμός της σταθμισμένης αντίστροφης απόστασης.
- r_k η απόσταση του k-ου γείτονα από το σημείο.
- ${x_4}^{(k)}$ η τιμή της 4ης διάστασης του k-ου γείτονα.

Όπως αναφέρθηκε και παραπάνω το μοντέλο έχει μόνο μια υπερπαράμετρο k όπου είναι ο αριθμός των γειτόνων. Η βέλτιστή υπερπαράμετρος υπολογίζεται με την εξαντλητική μέθοδο και για τις δύο παραλλαγές. Για την περίπτωση των συνθετικών χρονοσειρών όπου διαθέτουμε ενενήντα πέντε χιλιάδες δεδομένα εκπαίδευσης, ο χρόνος υλοποίησης του αλγορίθμου για μία περίπτωση υπερπαράμετρου k είναι σχεδόν μία ώρα. Ο χρόνος υπολογισμού όλων των πιθανών περιπτώσεων k είναι πάρα πολύ μεγάλος (περίπου 2 μέρες). Ο λόγος που θέλει τόσο πολύ χρόνο για να τρέξει το πρόγραμμα είναι η ίδια γλώσσα προγραμματισμού, καθώς η Python υστερεί σε ταχύτητα υπολογισμών. Η υπολογιστική ταχύτητα μπορεί να αυξηθεί πολύ με την βοήθεια του πακέτου *numba* όπου μετατρέπει τον κώδικα της Python κατευθείαν σε γλώσσα μηχανής. Το κόστος αυτής της διαδικασίας βρίσκεται στον τρόπο που πρέπει να γραφτεί ο κώδικας αποκλείοντας την χρήση αντικειμενοστραφή προγραμματισμού καθώς και μερικές δομές δεδομένων όπως τα λεξικά (*dictionaries*) και οι λίστες (*lists*). Η ομάδα που έχει αναλάβει την ανάπτυξη και την συντήρηση αυτού του πακέτου, συστήνει οι χρήστες του πακέτου κατά την σύνταξή του κώδικα να κάνουν όσο το δυνατόν περισσότερη χρήση μητρωικών συναρτήσεων όπως αυτές της βιβλιοθήκης *Numpy*. Το αποτέλεσμα όλων αυτών είναι ο χρόνος εκτέλεσης κάθε δοκιμής να είναι περίπου τρία λεπτά, **είκοσι φορές πιο γρήγορα σε σύγκριση με την κλασσική υλοποίηση κώδικα**. Μειώνοντας έτσι τον χρόνο για δοκιμές για παράδειγμα από δύο μέρες σε μόλις δύο ώρες.



Διάγραμμα 5-8: Απόδοση των μοντέλων KNN συναρτήσει των αριθμών γειτόνων k.

Από το διάγραμμα παραπάνω μπορούμε να επιλέξουμε την βέλτιστη υπερπαράμετρο k για δύο παραλλαγές. Για την παραλλαγή του μέσου όρου έχουμε μέγιστη απόδοση για k=19, ενώ στην περίπτωση με συντελεστές βάρους έχουμε βέλτιστη για k=21. Θα πρέπει να σημειωθεί πως ο βαθμός των ανάστροφων αποστάσεων πάρθηκε ίσος με δύο $n = 2$ (δηλαδή το ανάστροφο του τετραγώνου της απόστασης). Έγινε επιπλέον διερεύνηση για διαφορετικούς βαθμούς, αλλά λόγω του πλήθους των δεδομένων οι αποστάσεις των k πρώτων είναι σχεδόν ίσες, με αποτέλεσμα να μην έχουμε διαφορά στα αποτελέσματα. Στην περίπτωση όπου τα δεδομένα είναι λιγότερα θα περιμέναμε μεγάλη αλλαγή στα αποτελέσματα καθώς με την αύξηση του βαθμού μειώνουμε κατά πολύ την επιρροή των πιο απομακρυσμένων σημείων.

5.2.3: Μοντέλα μίξης (Mixture Models)

Στην επιστήμη της στατιστικής, ένα μοντέλο μείγματος είναι ένα πιθανολογικό μοντέλο για την αναπαράσταση παρουσίας υποπληθυσμών σε έναν συνολικό πληθυσμό. Τυπικά, ένα μοντέλο μείγματος αντιστοιχεί στην κατανομή του μείγματος που αντιπροσωπεύει την κατανομή πιθανοτήτων των παρατηρήσεων στο συνολικό πληθυσμό. Ωστόσο, ενώ τα προβλήματα που σχετίζονται με τις «κατανομές μιγμάτων» σχετίζονται με την εξαγωγή των ιδιοτήτων του συνολικού πληθυσμού από εκείνες των υποπληθυσμών, τα «μοντέλα μιγμάτων» χρησιμοποιούνται για την εξαγωγή στατιστικών συμπερασμάτων σχετικά με ιδιότητες των υποπληθυσμών, δίνοντας μόνο παρατηρήσεις για συγκεντρωτικό πληθυσμό, χωρίς στοιχεία ταυτότητας υποπληθυσμού. Θεωρητικά μπορούμε να αναμίξουμε όποια κατανομή θέλουμε εμείς είτε είναι συνεχής είτε διακριτή. Στην περίπτωση μας τα μοντέλα που θα “μειχθούν” είναι γκαουσιανές κατανομές δηλαδή, θα έχουμε μια μίξη γκαουσιανων κατανομών (Gaussian Mixture models). Η σύνθετη κατανομή πυκνότητας πιθανότητας είναι:

$$p(\theta|x) = \sum_{i=1}^{K} \pi_i N(x|\mu_i, \sigma_i) \tag{5.34}$$

όπου:

- π_i βάρος υποομάδας ως τον συνολικό πληθυσμό.

- μ_i η μέση τιμή της υποομάδας
- σ_i το μητρώο διασποράς της υπομάδας

Για να θεωρηθεί μία συνάρτηση ως πυκνότητας πιθανότητας θα πρέπει η συνάρτηση κατανομής πιθανότητας που τη συνοδέυει να ικανοποιεί την ανισότητα:

$$0 \leq F(x) \leq 1$$

Τούτο είναι προφανές αφού $F(x) = P(X \leq x)$ και η P αποτελεί μέτρο πιθανότητας. Στην περίπτωση που η συνάρτηση είναι συνεχείς (*continuous function*) :

$$\lim_{x \to -\infty} F(x) = F(-\infty) = 0 \text{ και } \lim_{x \to \infty} F(x) = F(\infty) = 1$$

Από την στιγμή που έχουμε γραμμικό συνδυασμό Γκαουσιανών κατανομών μπορούμε να αποδείξουμε ότι:

$$F(x) = \int_{-\infty}^{x} p(\theta|x)dx = \int_{-\infty}^{x} \sum_{i=1}^{K} \pi_i N(x|\mu_i, \sigma_i)\, dx = \sum_{i=1}^{K} \pi_i \int_{-\infty}^{x} N(x|\mu_i, \sigma_i)\, dx \Rightarrow \quad (5.35)$$

$$\lim_{x \to \infty} F(x) = \sum_{i=1}^{K} \pi_i \int_{-\infty}^{\infty} N(x|\mu_i, \sigma_i)\, dx \quad (5.36)$$

όπου:

- π_i βάρος υποομάδας ως τον συνολικό πληθυσμό.
- μ_i η μέση τιμή της υποομάδας
- σ_i το μητρώο διασποράς της υπομάδας

Από την θεωρία της κλασσικής Γκαουσιανής κατανομής:

$$\int_{-\infty}^{\infty} N(x|\mu_i, \sigma_i)\, dx = 1 \quad (5.37)$$

Άρα από τις σχέσεις 5.36 και 5.37 προκύπτει:

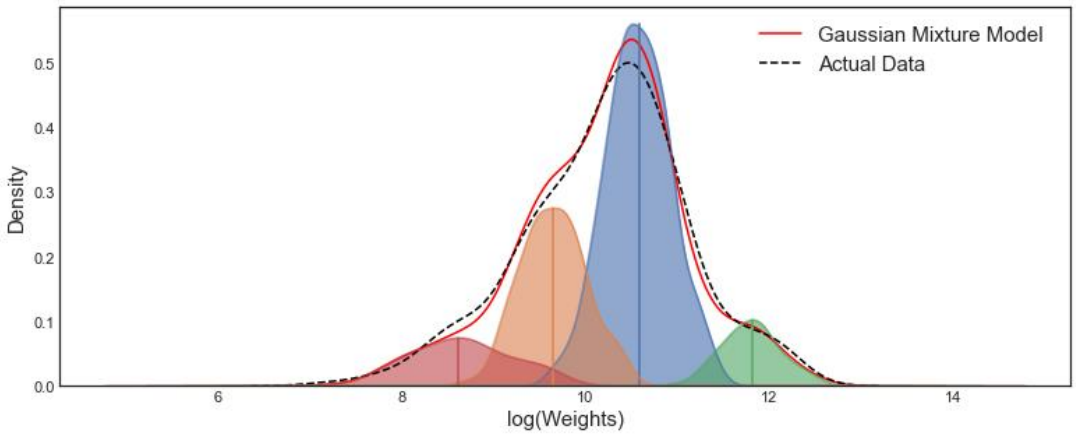
$$\lim_{x \to \infty} F(x) = \sum_{i=1}^{K} \pi_i \quad (5.38)$$

Οπότε ο μόνος περιορισμός για να μπορεί η συνάρτηση $p(\theta \vee x)$ να θεωρηθεί συνάρτηση πυκνότητας πιθανότητάς είναι πως το άθροισμα των συντελεστών θα πρέπει να ισούται με την μονάδα:

$$\sum_{i=1}^{K} \pi_i = 1 \quad (5.39)$$

Γενικά οι συντελεστές π_i παρουσιάζουν το βάρος κάθε υποομάδας ενώ οι συντελεστές μ_i και σ_i είναι τα στατιστικά χαρακτηριστικά της υποομάδας και πιο συγκεκριμένα η μέση τιμή και η τυπική απόκλιση.

Το θετικό αυτής της μεθοδολογίας είναι ότι μπορούν να περιγράφουν πολύ περίπλοκες κατανομές με τον γραμμικό συνδυασμό απλών Γκαουσιανων κατανομών. Όπως φαίνεται και στο διάγραμμα παρακάτω μπορούμε να περιγράψουμε ικανοποιητικά την κατανομή των πραγματικών δεδομένων με την βοήθεια μόλις τεσσάρων απλών κανονικών κατανομών.



Διάγραμμα 5-9: Προσαρμογή περίπλοκης κατανομής ως μίξης απλών κανονικών. Πηγή: Stack Exchange

Η διαδικασία αυτή αναγράφεται στην βιβλιογραφία από τον (Pearson, 1894) ο οποίος ήταν ο πρώτος που αντιμετώπισε ρητά το πρόβλημα της αποσύνθεσης στον χαρακτηρισμό των μη φυσιολογικών χαρακτηριστικών των αναλογιών μήκους μετώπου προς μήκος σώματος σε θηλυκούς πληθυσμούς καβουριών της ακτής. Το κίνητρο για αυτήν την εργασία δόθηκε από τον ζωολόγο Walter Frank Raphael Weldon, ο οποίος είχε υποθέσει το 1893 ότι η ασυμμετρία στο ιστόγραμμα αυτών των αναλογιών θα μπορούσε να σηματοδοτήσει εξελικτική απόκλιση. Η προσέγγιση του Pearson ήταν να προσαρμόσει ένα μονομεταβλητό μείγμα δύο κανονικών στα δεδομένα επιλέγοντας τις πέντε παραμέτρους του μείγματος έτσι ώστε οι εμπειρικές στιγμές να ταιριάζουν με αυτές του μοντέλου.

Ενώ η δουλειά του ήταν επιτυχής στον εντοπισμό δύο δυνητικά διακριτών υποπληθυσμών και στην επίδειξη της ευελιξίας των μειγμάτων ως εργαλείου ταιριάσματος ροπών, η διατύπωση απαιτούσε τη λύση ενός πολυωνύμου 9ου βαθμού που τότε αποτελούσε μια σημαντική υπολογιστική πρόκληση. Οι επόμενες εργασίες επικεντρώθηκαν στην αντιμετώπιση αυτών των προβλημάτων, αλλά η έρευνα πραγματικά απογειώθηκε μόνο με την εμφάνιση του σύγχρονου υπολογιστή και τη διάδοση των τεχνικών παραμετροποίησης της μεθόδου Μέγιστης Πιθανοφάνειας.

Ο τρόπος λειτουργείας της μεθοδολογίας είναι εμπνευσμένος από τον τρόπο χρήσης του (Pearson, 1894). Στο σύνολο όλων τον καταναλωτών σίγουρα θα υπάρχουν ομάδες καταναλωτών όπου θα έχουν σχεδόν κοινά στατιστικά χαρακτηριστικά. Στην περίπτωση που θέλουμε να κάνουμε πρόβλεψη το μόνο που χρειαζόμαστε να γνωρίζουμε είναι η πιθανότητα που έχει ο καταναλωτής να ανήκει στην κάθε υποομάδα. Θα πρέπει να σημειωθεί πως από την στιγμή που οι χρονοσειρές μετασχηματίζονται σε

σημεία στον R^4 χρησιμοποιούμε την πολυμεταβλητή Γκαουσιανή κατανομή. Η διαδικασία πρόβλεψής χωρίζεται σε δύο σκέλη: (i) Το πρώτο σκέλος όπου θέλουμε να βρούμε τις υποομάδες του πληθυσμού καθώς και τα στατιστικά χαρακτηριστικά του και (ii) το δεύτερο σκέλος όπου γίνεται η κατάταξη του καταναλωτή στις υποομάδες που εξεταστήκαν στο προηγούμενο βήμα και γίνεται η πρόβλεψη ανάλογα με την κατάταξη αυτή.

Ο υπολογισμός των στατιστικών χαρακτηριστικών των ομάδων γίνεται με τον αλγόριθμο Προσδοκίας-Μεγιστοποίησης (*Expectation-Maximization*) του οποίου ο τρόπος λειτουργίας έχει αναλυθεί στο Δεύτερο κεφάλαιο. Αναφορικά με την περίπτωση όπου η κατανομή είναι μίξη γκαουσιανών κατανομών στο βήμα της Μεγιστοποίησης (*M step*), ο μηδενισμός των παραγώγων για την μεγιστοποίηση των παραμέτρων γίνεται ως ακολούθως:

$$\hat{\mu}^{(j)} = \frac{\sum_{i=1}^{n} p(j|i)\vec{x}^{(i)}}{\sum_{i=1}^{n} p(j|i)} \tag{5.40}$$

$$\hat{p}_j = \frac{1}{n}\sum_{i=1}^{n} p(j|i) \tag{5.41}$$

$$\hat{\sigma^2}_j = \frac{\sum_{i=1}^{n} p(j|i)\left\|\vec{x}^{(i)} - \hat{\mu}^{(j)}\right\|^2}{d\sum_{i=1}^{n} p(j|i)} \tag{5.42}$$

όπου:

- $\hat{\mu}^{(j)}$ νέα βελτιωμένη μέση τιμή υποομάδας
- $\hat{p}_j$ νέα βελτιωμένη τιμή βάρους υποομάδας
- $\hat{\sigma^2}_j$ νέα βελτιωμένη διασπορά υποομάδας
- $p(j|i)$ η πιθανότητα του σημείου i να ανοίκει στην υποομάδα j
- d η διάσταση του σημείου (στην περίπτωση μας d =4)

Οι τύποι αυτοί είναι για την περίπτωση όπου έχουμε σφαιρική Γκαουσιανή κατανομή δηλαδή έχουμε κοινή διασπορά και μηδενική συνδιασπορά μεταξύ των διαστάσεων της κατανομής.

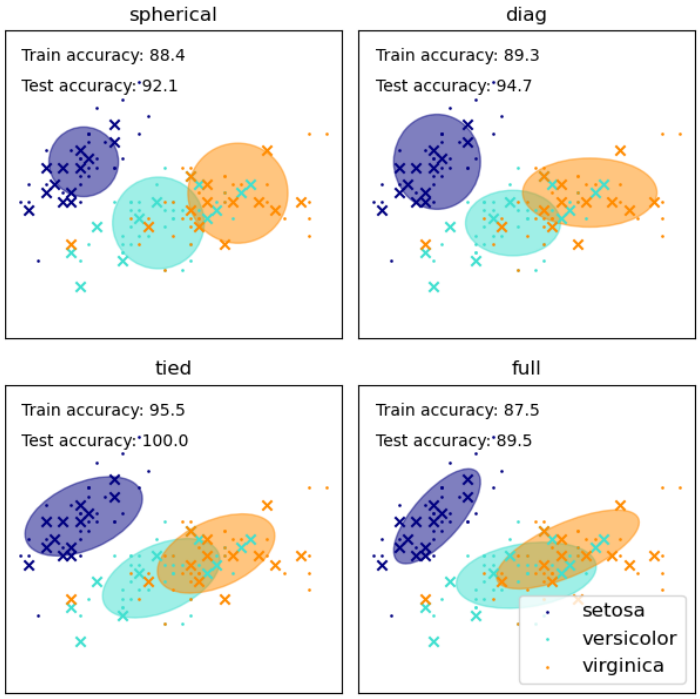
Η υλοποίηση του πρώτου βήματος γίνεται με την βοήθεια του υπολογιστικού πακέτου sklearn που διαθέτει ειδική συνάρτηση για την εκπαίδευση μοντέλου μίξης Γκαουσιανων κατανομών.

```
>>> import numpy as np
>>> from sklearn.mixture import GaussianMixture
>>> X = np.array([[1, 2], [1, 4], [1, 0], [10, 2], [10, 4], [10, 0]])
>>> gm = GaussianMixture(n_components=2, random_state=0).fit(X)
```

Εικόνα 5-12: Τρόπος εκπαίδευσης μοντέλου Μίξης Γκαουσιανών Κατανομών σε περιβάλλον Python.

Οι συναρτήσεις όπως και στα άλλα έτοιμα πακέτα μπορούν να παραμετροποιηθούν σε πολύ μεγάλο βαθμό ανάλογα με τις απαιτήσεις του κάθε προβλήματος που καλούμαστε να επιλύσουμε. Για το λόγο αυτό όπως και στα προηγούμενα μοντέλα θα αναφερθούμε στις βασικότερες παραμέτρους.

Η πιο βασική παράμετρος όχι μόνο της συνάρτησης αλλά και του μοντέλου είναι ο αριθμός των υποομάδων που θα επιλέξουμε στο μοντέλο και γίνεται με την μεταβλητή *n_components*. Στο παραπάνω παράδειγμα λόγου χάρη έχουν επιλεχθεί δύο υποομάδες. Η μεταβλητή *covariance_type* περιγράφει την μορφή που θα έχουν οι Γκαουσιανές κατανομές και πιο συγκεκριμένα την μορφή που θα έχει το μητρώο διασπορών. Οι διαθέσιμες επιλογές είναι Πλήρες ("Full"), Εξαρτημένες ("Tied"), Διαγώνιες ("Diag") και σφαιρικές ("spherical").



Εικόνα 5-13: Σχηματικές αλλαγές ανάλογα του μητρώου διασποράς που θα επιλέξουμε.Πηγή: Scikit-learn.org

Οι διαφορετικές περιπτώσεις έχουν να κάνουν με τα σενάρια των περιορισμών που θέτουμε στο μητρώο διασποράς της Γκαουσιανής κατανομής. Όπως αναφέρθηκε και πριν, στην περίπτωση που έχουμε σφαιρική κατανομή περιορίζουμε το μητρώο διασποράς να έχει στοιχεία μόνο στην κύρια διαγώνιο και όλα ίσα. Μαθηματικά αυτό σημαίνει ότι όλες οι διαστάσεις είναι ανεξάρτητες και έχουν την ίδια διασπορά. Στις διαγώνιες κατανομές περιορίζουμε και πάλι το μητρώο στο να έχει στοιχεία μόνο στην διαγώνιο, παρόλα αυτά τα στοιχεία της διαγωνίου είναι διαφορετικά. Όπως βλέπουμε και στο διάγραμμα παραπάνω οι κατανομές δεν έχουν ίδιες διασπορές μεταξύ των διαστάσεων με αποτέλεσμα τα αποτυπώματα των ισοϋψών να δημιουργούν ελλείψεις. Στην περίπτωση όπου επιλέγουμε να μην βάζουμε κάποιο περιορισμό στο μητρώο διασποράς του μοντέλου επιλέγουμε το μοντέλο ως Πλήρες. Στην περίπτωση που το μητρώο είναι πλήρες, έχουμε παρουσία συνδιασποράς μεταξύ των διαστάσεων με αποτέλεσμα το μοντέλο αυτό να μπορεί να περιγράψει καλύτερα καταστάσεις όπου υπάρχει συσχέτιση μεταξύ των διαστάσεων των σημείων. Τέλος η βιβλιοθήκη με την εντολή περιορισμού *("Tied")* παρέχει την δυνατότητα όλες οι κατανομές του μοντέλου να έχουν κοινό μητρώο διασποράς όπως φαίνεται και στο παραπάνω διάγραμμα. Κατά τον σχεδιασμό του μοντέλου θα πρέπει να γίνουν δοκιμές για την εξακρίβωση της βέλτιστης μορφής του μητρώου διασποράς. Η επιλογή πλήρους μητρώου

διασποράς μπορεί εκ πρώτης όψεως να φαίνεται ως η πιο καλή περίπτωση λόγω της μεταβλητότητας του μοντέλου και λόγω των περισσότερων μεταβλητών αλλά η ύπαρξη περισσότερων μεταβλητών (στην περίπτωση που δουλεύουμε στον R^4 έχουμε δώδεκα παραπάνω παραμέτρους ανά υποομάδα μεταξύ διαγώνιας και πλήρους κατανομής) μπορεί να καταστήσει αδύνατη την εύρεση του ολικού μέγιστου κατά την εκπαίδευση του μοντέλου. Για τον λόγο αυτό το μοντέλο που απεικονίζεται στα διαγράμματα παραπάνω έχει μειωμένη απόδοση στην περίπτωση που έχουμε πλήρες μοντέλο.

Η παράμετρος *random_state* έχει εισαχθεί για την μερική επίλυση των προβλημάτων που έχει ο αλγόριθμός Προσδοκίας-Μεγιστοποίησης (*Expectation Maximization*). Όπως αναφέραμε και στο δεύτερο κεφάλαιο ο αλγόριθμος Προσδοκίας-Μεγιστοποίησης είναι μια επαναληπτική διαδικασία για την εύρεση των βέλτιστων στατιστικών παραμέτρων καταλήγοντας σε τοπικό μέγιστο της συνάρτησης πιθανοφάνειας. Το αποτέλεσμα της διαδικασίας επηρεάζεται εξ ολοκλήρου από τις αρχικές συνθήκες, για αυτόν τον λόγο δοκιμάζουμε την διαδικασία με πολλές διαφορετικές και τυχαίες αρχικές τιμές και επιλέγουμε το μοντέλο με την μέγιστη πιθανοφάνεια. Ο αριθμός των δοκιμών δίνεται από την μεταβλητή *random_state*. Στην περίπτωση που δεν συμπληρωθεί το πεδίο γίνεται μόνο μια δοκιμή. Για την ταχύτερη εκτέλεση πολλές φορές συνίσταται οι αρχικές μέσες τιμές να υπολογίζονται με την βοήθεια του αλγορίθμου *K-means*.

Τέλος οι τελευταίες παράμετροι του μοντέλου αφορούν την σύγκλιση του μοντέλου. Πιο συγκεκριμένα η μεταβλητή *"tol"* είναι το όριο σύγκλισης της επαναληπτικής διαδικασίας (δηλαδή θα έχουμε διακοπή διαδικασίας στην περίπτωση όπου η διαφορά της πιθανοφάνειας δύο διαδοχικών επαναλήψεων είναι μικρότερη ή ίση από την τιμή που έχει οριστεί). Η μεταβλητή *max_iter* δηλώνει το άνω όριο των επαναλήψεων του αλγορίθμου, δηλαδή στην περίπτωση που ο αλγόριθμος δεν έχει συγκλίνει στις επαναλήψεις αυτές τότε η διαδικασία σταματάει και λαμβάνονται οι συντελεστές από την τελευταία επανάληψη.

Από την στιγμή που έχουμε τα στοιχεία για τις υποομάδες από τα γνωστά δεδομένα μπορούμε να κατατάξουμε και να κάνουμε προβλέψεις για τους πελάτες που δεν γνωρίζουμε την κατανάλωσή τους. Η κατάταξη των αγνώστων καταναλωτών θα μπορούσε να γίνει και με το πρώτο βήμα του αλγορίθμου Προσδοκίας-Μεγιστοποίησης (*Expectation-Maximization*) εάν δεν είχαμε άγνωστη την τέταρτη διάσταση του σημείου. *Για τον λόγο αυτό η κατάταξη γίνεται αποκλειστικά από τα διαθέσιμα δεδομένα, δηλαδή τις τρεις πρώτες διαστάσεις*. Η μεθοδολογία αυτή μπορεί να λειτουργήσει και σε περιπτώσεις που λείπουν και άλλες τιμές και μπορεί να κάνει και διαδοχικές προβλέψεις με κόστος πάντα την ακρίβεια του αποτελέσματος. Η πιθανότητα ο καταναλωτής i να ανήκει στην υποομάδα j δίνεται από την σχέση:

$$p(j|i) = \frac{\pi_j N\left(x_{C_u}^{(i)}, \mu_{C_u}^{(j)}, \sigma_{C_u}^{(j)}\right)}{\sum_{j=1}^{K} \pi_j N\left(x_{C_u}^{(i)}, \mu_{C_u}^{(j)}, \sigma_{C_u}^{(j)}\right)} \quad (5.43)$$

όπου:

- π_j το βάρος την υπομάδας
- $\mu_{C_u}^{(j)}$ η μέση τιμή αφαιρώντας την διάσταση την οποία δεν γνωρίζουμε.
- $\sigma_{C_u}^{(j)}$ το μητρώο διασποράς αφαιρώντας την διάσταση που δεν γνωρίζουμε.

Η διάσταση που δεν γνωρίζουμε στην περίπτωση της εργασίας αυτής είναι η τέταρτη.

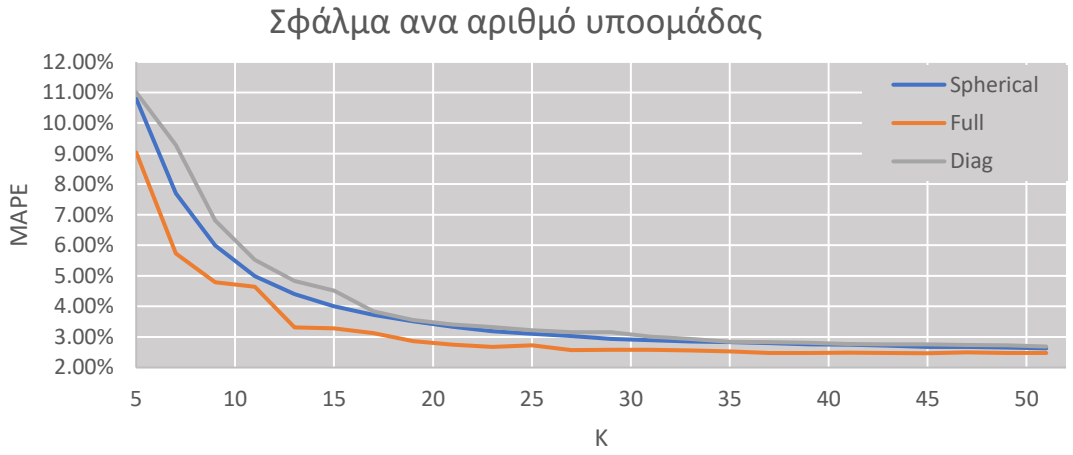
Από την στιγμή που γνωρίζουμε την πιθανότητα του καταναλωτή να ανήκει στην κάθε υποομάδα μπορούμε να υπολογίσουμε την εκτιμώμενη τιμή ως άθροισμα του γινομένου της πιθανότητας του σημείου να ανήκει στην υποομάδα με την μέση τιμή της υποομάδας αυτής για την άγνωστη διάσταση. Ο τρόπος υπολογισμού περιγράφεται για την περίπτωση που δεν γνωρίζουμε μόνο την τελευταία τιμή (δηλαδή γίνεται η χρήση μόνο της 4ης διάστασης της μέσης τιμής) γίνεται ως ακολούθως:

$$F_i = \sum_{j=1}^{K} p(j|i) * \mu_4^{(j)} \tag{5.44}$$

όπου:

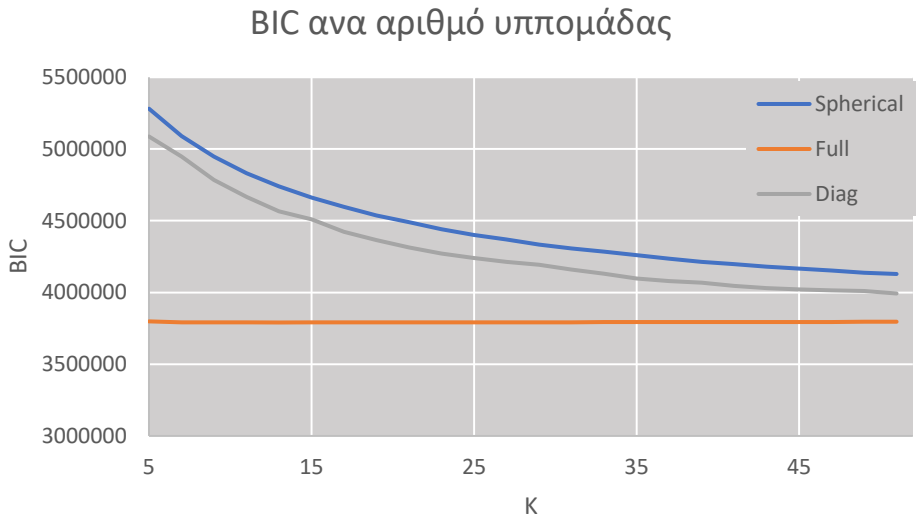
- $p(j|i)$ η πιθανότητα του στοιχείου i να ανήκει στην υποομάδα j.
- $\mu_4^{(j)}$ η μέση τιμής της 4ης διάστασης της υποομάδας j.

Λόγω της φύσης των παραμέτρων η διαδικασία εύρεσης των βέλτιστων παραμέτρων δεν είναι τόσο απλή. Με την αλλαγή της μορφής του μητρώου αλλάζει και ο αριθμός των παραμέτρων του. Τα κυκλικά και τα διαγώνια μοντέλα έχουν πολύ λιγότερες παραμέτρους σε αντίθεση με το πλήρες με αποτέλεσμα να έχουν διαφορετικές επιδόσεις για τον ίδιο αριθμό υποομάδων. Για τον λόγο αυτό έγινε εξαντλητική αναζήτηση για όλους του πιθανούς συνδυασμούς. Οι συναρτήσεις κόστους που χρησιμοποιήθηκαν για την αντικειμενική σύγκριση των μοντέλων είναι η συνάρτηση ποσοστιαίου απόλυτου μέσου σφάλματος (*MAPE*) καθώς και το Μπεϋζιανό κριτήριο πληροφορίας (*Bayesian Information Criterion*).



Διάγραμμα 5-10: Απόδοση μοντέλων συναρτήσει των αριθμών υποομάδας.

Θα πρέπει να σημειωθεί πως για τις περιπτώσεις όπου έχουμε περιορισμό στο μητρώο διασποράς (Σφαιρικό και Διαγώνιο μοντέλο) ο χρόνος σύγκλισης είναι πολύ μεγάλος σε σύγκριση με το Πλήρες μοντέλο. Η εξήγηση μπορεί να αποδίδεται στο γεγονός ότι με περισσότερες μεταβλητές υπάρχουν περισσότερα τοπικά ελάχιστα με αποτέλεσμα ο αλγόριθμος να συγκλίνει πιο γρήγορα. Για το Μπεϋζιανό κριτήριο πληροφορίας έχουμε τα παρακάτω διάγραμμα:



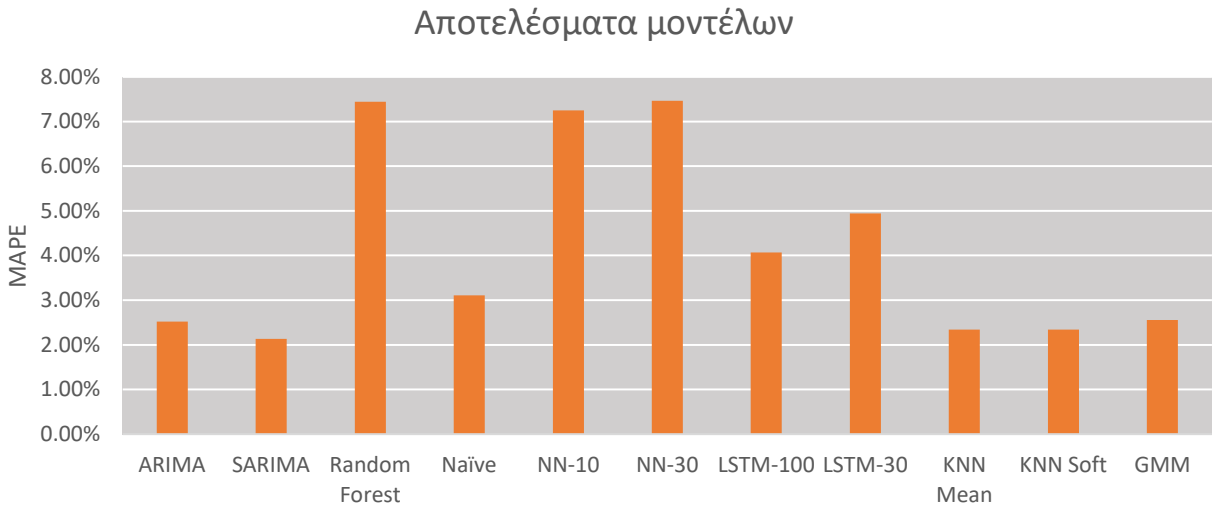
Διάγραμμα 5-11: Ο αριθμός BIC συναρτήσει των αριθμών υποομάδας.

Όπως μπορούμε να δούμε και από τα δύο διαγράμματα το Πλήρες μοντέλο είναι πιο ικανό να περιγράψει το φαινόμενο. Θα πρέπει να σημειωθεί πως για πολύ μεγάλο αριθμό υποομάδων είναι πολύ πιθανό τα άλλα μοντέλα να έχουν ακόμα μικρότερα σφάλματα αλλά απορρίπτονται καθώς απαιτούν πολύ μεγάλη υπολογιστική ισχύ. Το σφάλμα του πλήρους μοντέλου για αριθμό υποομάδων $K = 29$ δεν αλλάζει σχεδόν καθόλου, για αυτόν τον λόγο επιλέγεται και ως η βελτιστοποιημένη παράμετρος του μοντέλου για την περίπτωση των συνθετικών δεδομένων.

6. Αποτελέσματα και συμπεράσματα

6.1: Σύγκριση αποτελεσμάτων μοντέλων πρόβλεψης για τα δεδομένα συνθετικών χρονοσειρών

Για την περίπτωση των συνθετικών χρονοσειρών τα αποτελέσματα ανά μοντέλο απεικονίζονται στο παρακάτω διάγραμμα.



Διάγραμμα 6-1: Αποτελέσματα όλων των μοντέλων που εξετάστηκαν πάνω στα συνθετικά δεδομένα.

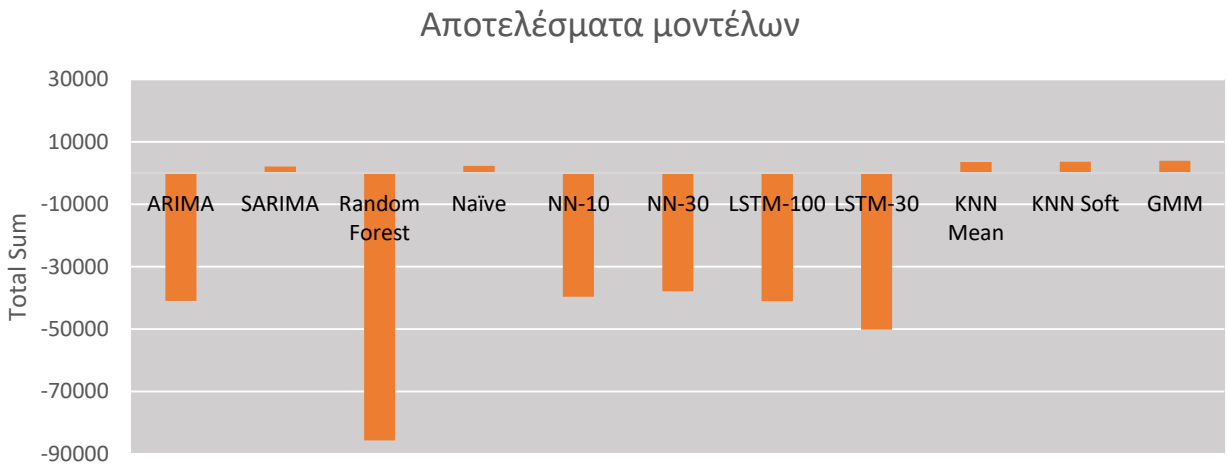
Τα μοντέλα που απέδωσαν λιγότερο στο μέσο απόλυτο ποσοστιαίο σφάλμα (*MAPE*) με διαφορά ήταν το Random Forest και τα απλά νευρωνικά δίκτυα (και οι δύο παραδοχές) με σφάλμα πάνω από 7%. Η εξήγηση πίσω από την τόσο χαμηλή απόδοση κρύβεται στην φύση των μοντέλων και τον τρόπο εκπαίδευσης, τα μοντέλα αυτά χρειάζονται πάρα πολλά δεδομένα για να μπορούν να περιγράψουν ένα περίπλοκο πρόβλημα και οι ογδόντα τιμές εκπαίδευσης που έχουμε στην διάθεση μας είναι πολύ λίγες για να μπορέσουν να προσαρμοστούν στις απαιτήσεις του μοντέλου. Η απόδοση των μοντέλων *LSTM* ήταν πολύ καλή αν κάποιος αναλογιστεί τον περιορισμένο αριθμό δεδομένων που διαθέτουμε, δείχνοντας έτσι ότι τα μοντέλα *LSTM* έχουν την δυνατότητα να πρωταγωνιστήσουν στην επίλυση τέτοιων προβλημάτων αν τους δοθεί ικανοποιητικός αριθμός δεδομένων. Η μέθοδος Naive είναι η πιο απλή και απαιτεί την λιγότερη υπολογιστική ισχύ καθώς κάνει μόλις ένα δευτερόλεπτο για τον υπολογισμό πέντε χιλιάδων αποτελεσμάτων. Παρόλο την απλότητα της μεθόδου τα αποτελέσματα ήταν εκπληκτικά καλά και αυτό αποδίδεται στην μορφή τον δεδομένων (σταθερή περιοδικότητα χωρίς αύξηση μέσου όρου). Στην περίπτωση που τα πραγματικά δεδομένα έχουν τις παραπάνω ιδιότητες η μέθοδος naive είναι ιδανική για την πρώτη γρήγορη αναγνωριστική εκτίμηση. Τα μοντέλα κάθετης λειτουργίας (χρονοσειράς) λειτούργησαν πολύ αποτελεσματικά. Πιο συγκεκριμένα, οι μεθοδολογίες *KNN* παρείχαν σχεδόν την ίδια ακρίβεια. Αυτό αποδίδεται στο πολύ μεγάλο πλήθος δεδομένων στο training set καθώς οι k πρώτες αποστάσεις έχουν μεγάλη πιθανότητα να είναι σχεδόν ίσες. Η υπόθεση αποδεικνύεται και μαθηματικά:

$$r_k = c \Longrightarrow \frac{{r_k}^{-n}}{\sum_{k=1}^{K} {r_k}^{-n}} = \frac{1}{n} \tag{6.1}$$

όπου:

- r_k η απόσταση του σημείου που θέλουμε να κάνουμε πρόγνωση με το k κοντινό σημείο.
- n ο βαθμός βαρύτητας της ανάστροφης απόστασης.

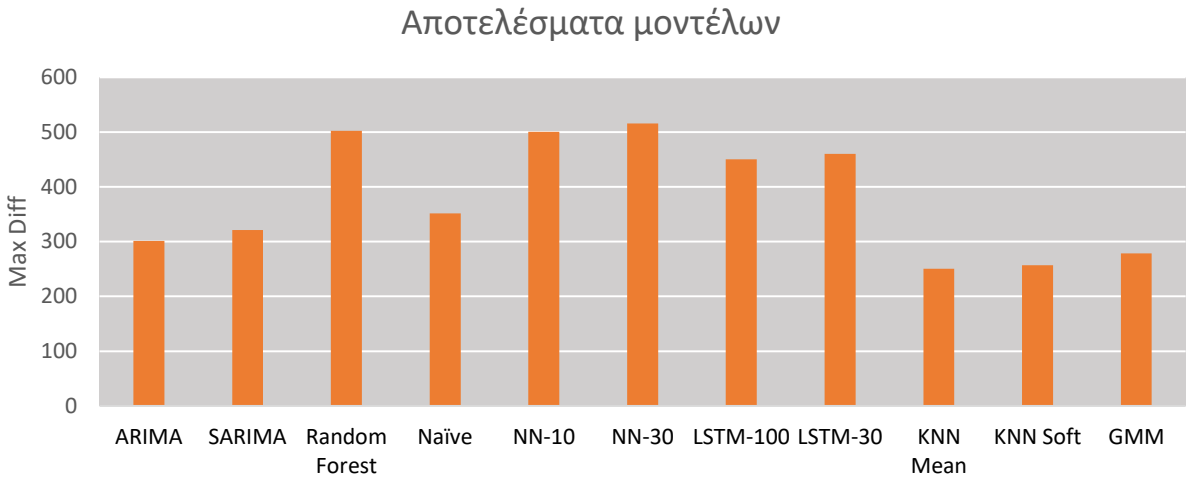
Ο συντελεστής αυτός είναι ουσιαστικά ο συντελεστής του μέσου όρου. Ένας άλλος λόγος που τα μοντέλα χρονοσειρών παράγουν καλά αποτελέσματα είναι το πολύ μεγάλο πλήθος χρονοσειρών που γνωρίζουμε την τελευταία τιμή (ενενήντα πέντε χιλιάδες). Το μοντέλο μίξης γκαουσιανών είχε λίγο χειρότερη απόδοση από το KNN. Τέλος, η ομάδα των μοντέλων *ARIMA* είχαν την καλύτερη συμπεριφορά με καλύτερο όλων το μοντέλο *SARIMA*. Η συμπεριφορά αυτή ήταν αρκετά αναμενόμενη από την στιγμή που τα δεδομένα έχουν παραχθεί από μεθοδολογία που στηρίζεται στα μοντέλα *ARIMA*. Η καλύτερη απόδοση που μπορέσαμε να πετύχουμε είχε σφάλμα μόλις 2.13%.



Διάγραμμα 6-2: Απόδοση μοντέλων κατά το συνολικό άθροισμα για τα συνθετικά δεδομένα.

Η συνάρτηση σφάλματος Total Sum αποτυπώνει αν το μοντέλο έχει προδιάθεση για υπερεκτίμηση η υποεκτίμηση των αποτελεσμάτων. Στην περίπτωση που ο δείκτης είναι αρνητικός το μοντέλο υπερεκτιμά τις καταναλώσεις. Ο δείκτης αυτός έχει υπολογιστεί διότι έχει μεγάλη αξία από πλευράς του παρόχου, ο οποίος θέλει να έχει ένα σταθερό Cash flow (χρηματορροή). Το χειρότερο μοντέλο με διαφορά ήταν και πάλι το *Random Forest*. Τα μοντέλα νευρωνικών δικτύων σε αντίθεση με την κακή απόδοση που είχαν στον προηγούμενο δείκτη κατάφεραν να έχουν καλύτερη απόδοση από τα μοντέλα *LSTM*. Μεγάλη έκπληξη προκαλεί η επίδοση του μοντέλου *ARIMA* αν αναλογιστούμε την καλή απόδοση που είχε στον πρώτο δείκτη. Η καλύτερη απόδοση σημειώθηκε από το μοντέλο *SARIMA*.Τα υπόλοιπα μοντέλα (KNN,GMM,Naive) σημείωσαν πανομοιότητες πολύ καλές αλλά πανομοιότητες αποδόσεις.

Ο τελευταίος δείκτης που χρησιμοποιήθηκε είναι της μέγιστης διαφοράς.

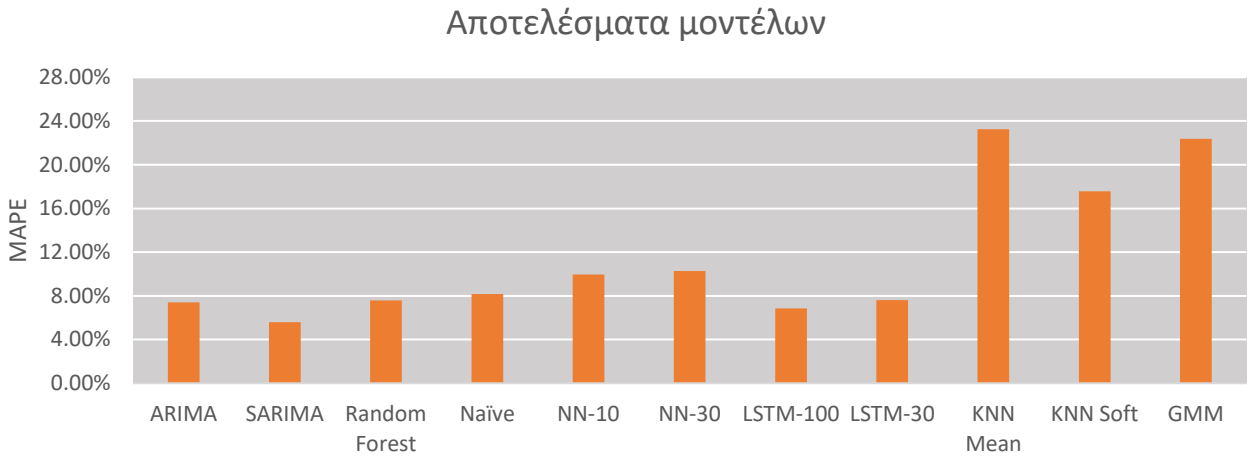


Διάγραμμα 6-3: Η μέγιστη απόκλιση κάθε μοντέλου για τα συνθετικά δεδομένα.

Τα αποτελέσματα στην περίπτωση της μέγιστης διαφοράς ποιοτικά δεν αλλάζουν σε σχέση με τα αποτελέσματα του μέσου απόλυτου ποσοστιαίου σφάλματος. *Τα μοντέλα κατακόρυφης λειτουργίας έχουν την καλύτερη απόδοση σε σχέση με τα υπόλοιπα.* Βέβαια το μόνο συμπέρασμα που μπορούμε να βγάλουμε για τα μοντέλα με αυτή την συνάρτηση σφάλματός είναι ένα άνω όριο για την διασπορά των προβλέψεων.

6.2: Σύγκριση αποτελεσμάτων μοντέλων πρόβλεψης για τα δεδομένα ενεργειακής κατανάλωσης

Από την στιγμή που δεν είχαμε πραγματικά δεδομένα καταναλώσεων για να επαληθεύσουμε τα συνθετικά δεδομένα χρειάστηκε να δοκιμάσουμε τα μοντέλα σε δεδομένα ενέργειας τα οποία είναι πιο εύκολα διαθέσιμα. Τα αποτελέσματα για το μέσο απόλυτο ποσοστιαίο σφάλμα (MAPE) αποτυπώνονται στο παρακάτω διάγραμμα.:



Διάγραμμα 6-4: Σφάλμα MAPE όλων των μοντέλων για τα δεδομένα Ενέργειας

Κατά τον διαχωρισμό Γενικά τα σφάλματα είναι αρκετά μετατοπισμένα προς τα πάνω σε σχέση με τα συνθετικά δεδομένα. Αυτό είναι κάτι που περιμέναμε αφού πήραμε την απόφαση να κάνουμε ανάλυση ανά μήνα και όχι σε τρίμηνο όπως κάναμε και με τα συνθετικά δεδομένα, ενώ τα μοντέλα που έχουν κατασκευαστεί είναι για δεδομένα με τριμηνιαία χρονική βάση . Παρ' όλα αυτά οι παράμετροι δεν έμειναν ίδιες καθώς έγινε τροποποίηση και βελτιστοποίηση για τα νέα δεδομένα.

Για τις οριζόντιες μεθοδολογίες την καλύτερη απόδοση την έχει το μοντέλο *SARIMA* με σφάλμα 5.58%. Δεύτερησε κατάταξη μέθοδος έρχονται τα μοντέλα *LSTM*. Το μόνο μοντέλο με σταθερή απόδοση είναι το *Random Forest* έχοντας σχεδόν την ίδια απόδοση και στις δύο περιπτώσεις δεδομένων. Στην περίπτωση των κάθετων μεθοδολογιών η απόδοση ήταν πολύ χαμηλή. Το αποτέλεσμα αυτό ήταν αναμενόμενο λόγω χαμηλού αριθμού χρονοσειρών που γνωρίζουμε την τελευταία μέτρηση. Στην περίπτωση που δεν έχουμε πολλά δεδομένα μπορούμε να δούμε την διαφορά στην μέθοδο *KNN* μεταξύ μέσης τιμής και με συντελεστές βαρών.

Λόγω του μικρού πλήθους δεδομένων κρίνεται άσκοπη η ανάλυση των άλλων σφαλμάτων καθώς τα αποτελέσματα διαφοροποιούνται ανάλογα ποια χρονοσειρά έχει χρησιμοποιηθεί ως δεδομένο δοκιμής.

6.3: Συμπεράσματα και επέκταση μελέτης

Για τις οριζόντιες μεθοδολογίες τα μοντέλα που σίγουρα ξεχώρισαν ήταν το μοντέλο *SARIMA* όπου κατάφερε να πετύχει την βέλτιστη απόδοση και στις δύο περιπτώσεις δεδομένων και το μοντέλο *LSTM* όπου τα κατάφερε εξαιρετικά και χωρίς μεγάλο όγκο δεδομένων. Κατάφερε την δεύτερη μεγαλύτερη απόδοση στα δεδομένα ενέργειας. Ίσως με ακόμα καλύτερη παραμετροποίηση και την σωστή επιλογή μέσων βελτιστοποίησης η απόδοση θα ήταν ακόμα καλύτερη. Για τις κατακόρυφες μεθοδολογίες ενώ κατέγραψαν πολύ καλή απόδοση στα συνθετικά δεδομένα και με πολύ καλή ταχύτητα εκτέλεσης, στην περίπτωση των πραγματικών δεδομένων οι προβλέψεις που κρίθηκαν να πάρουν ήταν άστοχες. Το αποτέλεσμα αυτό μπορεί να δικαιολογείται από το γεγονός ότι τα δεδομένα των πλήρων χρονοσειρών ήταν πολύ λίγα. . Η πλήρης αξιολόγηση θα πρέπει να πραγματοποιηθεί όταν θα υπάρχουν διαθέσιμα τα πραγματικά δεδομένα. Στην περίπτωση όπου τα δεδομένα αυτά είναι παρόμοια με τα συνθετικά τότε ενδέχεται να σημειώσουν μεγάλη απόδοση.

Όσον αφορά την επέκταση της εργασίας αυτής, ενδιαφέρον θα έχει η εξέταση της ανάμιξης των κατακόρυφων και οριζόντιων μεθοδολογιών για την κατασκευή ενός υβριδικού μοντέλου. Η μεθοδολογία αυτή θα προσπαθεί να επιλύσει τα αρνητικά του κάθε μοντέλου. Ο αρχικός τρόπος σκέψης είναι να χρησιμοποιηθεί το μοντέλο μίξης γκαουσιανων κατανομών για την ομαδοποίηση των καταναλωτών δημιουργώντας μια μεγάλη βάση δεδομένων για κάθε υποομάδα. Από τα δεδομένα κάθε υποομάδας θα εκπαιδεύεται ένα μοντέλο για κάθε υποομάδα και όχι για κάθε καταναλωτή. Το κύριο θετικό της μεθοδολογίας αυτής είναι η ύπαρξη πολλών δεδομένων και δίνει την δυνατότητα μοντέλα όπως LSTM να δείξουν την πραγματική τους ισχύ. Ένα ακόμα θετικό στοιχείο είναι η μείωση χρόνου εκπαίδευσης, ειδικά σε σχέση με την κλασσική μεθοδολογία *LSTM* που ακολουθήθηκε στα πλαίσια αυτής της εργασίας λόγω της μείωσης των αριθμών των μοντέλων.

7. Βιβλιογραφία

7.1 Ξενόγλωσση βιβλιογραφία

Dempster, Arthur P., Nan M. Laird, and Donald B. Rubin. "Maximum likelihood from incomplete data via the EM algorithm." Journal of the Royal Statistical Society: Series B (Methodological) 39.1 (1977): 1-22.

Moon, Todd K. "The expectation-maximization algorithm." *IEEE Signal processing magazine* 13.6 (1996): 47-60.

Hardt, Moritz, Ben Recht, and Yoram Singer. "Train faster, generalize better: Stability of stochastic gradient descent." *International conference on machine learning*. PMLR, 2016.

Makridakis, Spyros. "Accuracy measures: theoretical and practical concerns." *International journal of forecasting* 9.4 (1993): 527-529.

Bhat, Harish S., and Nitesh Kumar. "On the derivation of the bayesian information criterion." *School of Natural Sciences, University of California* 99 (2010).

Ho, Tin Kam. "Random decision forests." *Proceedings of 3rd international conference on document analysis and recognition*. Vol. 1. IEEE, 1995.

Hochreiter, Sepp, and Jürgen Schmidhuber. "Long short-term memory." *Neural computation* 9.8 (1997): 1735-1780.

Altman, Naomi S. "An introduction to kernel and nearest-neighbor nonparametric regression." *The American Statistician* 46.3 (1992): 175-185.

Reynolds, Douglas A. "Gaussian mixture models." Encyclopedia of biometrics 741.659-663 (2009).

Lam, Siu Kwan, Antoine Pitrou, and Stanley Seibert. "Numba: A llvm-based python jit compiler." Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC. 2015.

Améndola, Carlos, Jean-Charles Faugere, and Bernd Sturmfels. "Moment varieties of Gaussian mixtures." arXiv preprint arXiv:1510.04654 (2015).

De Boer, Pieter-Tjerk, et al. "A tutorial on the cross-entropy method." Annals of operations research 134.1 (2005): 19-67.

Contreras, Javier, et al. "ARIMA models to predict next-day electricity prices." IEEE transactions on power systems 18.3 (2003): 1014-1020.

Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

Brownlee, J. (2017, 8 14). *Machine Learning Mastery*. Retrieved from https://machinelearningmastery.com/difference-test-validation-datasets/

Darts. (2022, 2). Retrieved from https://unit8co.github.io/darts/

Duchi, J., Hazan, E., & Yoram, S. (2011). Adaptive Subgradient Methods for. 39. Retrieved from https://jmlr.org/papers/volume12/duchi11a/duchi11a.pdf

Fix, E., & Hodges, J. (1951). Discriminatory Analysis. Nonparametric Discrimination: Consistency Properties. *USAF School of Aviation Medicine*, 22.

GoodFellow, I., Bengio, Y., Courville, & Aaron. (2016). Deep Learning. MIT Press.

Hinton, G., Srivastava, N., & Swersky, K. (n.d.). *Overview of mini-batch gradient descent .* Retrieved from http://www.cs.toronto.edu: http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Hochreiter, S., & Schmidhuder, J. (1997). Long Short-Term Memory. *MIT Press*, 32.

ioanpier. (2020, 8 19). Retrieved from GitHub: https://gist.github.com/ioanpier/e231b22bb9f705ef6280c8b73e40b4a1

Kaggle. (2022, 2). Retrieved from https://www.kaggle.com/

Kingma, D. P., & Lei, J. B. (2014). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. 15.

Kleinberg, E. (1990). Stohastic discrimination. *Annals of Mathematics and Artificial intelligence*, 33.

Kwiatkowski, R. (2021, 5). *Towards Data Science*. Retrieved from https://towardsdatascience.com/gradient-descent-algorithm-a-deep-dive-cf04e8115f21

MacKay, D. (2003). *Information Theory, Inference and Learning Algorithms.* Cambridge University Press.

Matplotlib. (2022, 2). Retrieved from https://matplotlib.org/

Numba. (2022, 2). Retrieved from https://numba.pydata.org/

Numpy. (2022, 2). Retrieved from https://numpy.org/

Pandas. (2022, 2). Retrieved from https://pandas.pydata.org/

Pearson, K. (1894). Contributions to the Mathematical Theory of Evolution.

Python. (2022, 2). Retrieved from https://www.python.org/

Random Forest. (2022). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

Scikit-Learn. (2022, 2). Retrieved from https://scikit-learn.org/stable/

Shixiong, W., Chongshou, L., & Andrew, L. (2019). Why Are the ARIMA and SARIMA not Sufficient. 10.

statsmodels. (2022, 2). Retrieved from https://www.statsmodels.org/stable/index.html

TensorFlow. (2022, 2). Retrieved from https://www.tensorflow.org/

Tin, H. K., & AT and T Bell Laboratories, Inc. (1995). Random Decision Forests. *Random decision forests*, 5.

Tofallis, C. (2015). A Better Measure of Relative Prediction Accuracy for Model Selection and Model Estimation. *Journal of the Operational Research Society*, 25.

Tsoukalas, I., Kossieris, P., & Makropoulos, C. (2020). Simulation of Non-Gaussian Correlated Random Variables, Stochastic Processes and Random Fields: Introducing the anySim R-Package for Environmental Applications and Beyond . 41.

Tsoukalas, I., Makropoulos, C., & Koutsoyiannis, D. (2018). Simulation of Stochastic Processes Exhibiting Any-Range Dependence and Arbitrary Marginal Distributions. *AGU*.

Kossieris, P., Tsoukalas, I., Makropoulos, C., & Savic, D. (2019). Simulating Marginal and Dependence Behaviour of Water Demand Processes at Any Fine Time Scale. Water, 11(5), 885. https://doi.org/10.3390/w11050885

Yiu, T. (2019, 7). *Towards Data Science*. Retrieved from https://towardsdatascience.com/the-curse-of-dimensionality-50dc6e49aa1e

Zijie, Z., & Mengtian, Z. (2020). K-Nearest Neighbors(KNN) Classification with Different. 14.

7.2 Ελληνόγλωσση βιβλιογραφία

Κοκκολάκης, Γ., and Ι. Σπηλιώτης. "Θεωρία Πιθανοτήτων και Στατιστική με εφαρμογές." Εκδόσεις Συμεών, Αθήνα (2010): 320-331.

Ν.Καδιανάκης, Σ.Καρανάσιος and Α.Φελλούρης. "Λογισμός συναρτήσεων πολλών μεταβλητων." *Ιδιωτική Έκδοση* Αθήνα (2015): ISBN 978-960-90120-1-0

Ι.Σαρρής, Θ.Καρακασίδης "Αριθμητικές Μέθοδοι και Εφαρμογές για Μηχανικούς." Εκδόσεις Τζιόλα Αθήνα (2017): ISBN 978-960-418-520-7

Παράρτημα

Ο κώδικας που συντάχθηκε για τις μεθοδολογίες είναι:

NAIVE, ARIMA, SARIMA

```
import pickle
import numpy as np
import utils as u

"""DATA READ FOR SYNTHETIC"""
data=1
with open("data"+str(data)+".pkl","rb") as f:
    [train_set,test_set]=pickle.load(f)

"""DATA READ FOR KAGGLE"""
kaggle = np.genfromtxt('data.csv', delimiter=',')
test_set=kaggle[:,[1,3]]
train_set=kaggle[:,[0,2,4,5,6,7,8,9]]

"""Dataset Divition"""
x=test_set.shape[0]
xt=int(5/6*x)
trainn_set=test_set[:xt,:]
testt_set=test_set[xt:,:]


"""ROLLING NAIVE"""
coef=(0,0.5,0.25,0.25)
results_Naive=u.rolling_naive(trainn_set, testt_set,coef=coef)


"""ROLLING ARIMA"""
results_ARIMA=u.rolling_arima(trainn_set, testt_set,order=(2,0,1))

"""ROLLING SARIMA"""
results_SARIMA=u.rolling_sarima(trainn_set, testt_set,order=(2,0,1),season=3)


"""print results"""
```

```
print("\n\n           Naive")
print("\nMAPE:                {:.3%} \nTotal Sum:           {:.2f} \nMax
difference:
{:.2f}".format(results_Naive[0],results_Naive[1],results_Naive[2]))

print("\n           ARIMA")
print("\nMAPE:                {:.3%} \nTotal Sum:           {:.2f} \nMax
difference:
{:.2f}".format(results_ARIMA[0],results_ARIMA[1],results_ARIMA[2]))

print("\n           SARIMA")
print("\nMAPE:                {:.3%} \nTotal Sum:           {:.2f} \nMax
difference:
{:.2f}".format(results_SARIMA[0],results_SARIMA[1],results_SARIMA[2]))
```

Με την βοήθεια του αρχείου:

```
import numpy as np
from statsmodels.tsa.arima.model import ARIMA
from tqdm import tqdm
import dataprep as da


def rolling_arima(train_set,test_set,order=(1,1,1)):
    """
    Parameters
    ----------
    train_set : Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    order : Tuple, optional
        The order of arima model. The default is (1,1,1).

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """

    x,y=test_set.shape
```

```
    #ERRORS#
    MAPE=np.zeros([y,])
    ToS=np.zeros([y,])
    MaxDif=0.0

    for i in tqdm(range(y)):
        train=list(train_set[:,i])
        for ii in range(x):

            model=ARIMA(train,order=order)
            m_f=model.fit()
            F=m_f.forecast()
            #ERRORS RECORD#
            MAPE[i] += da.MAPE(F,test_set[ii,i])
            ToS[i] += da.TotalSum(F,test_set[ii,i])
            MaxDif= max(MaxDif,np.absolute(F-test_set[ii,i]))

            train.append(test_set[:,i][ii])
    return np.average(MAPE/x) , np.sum(ToS) , float(MaxDif)


def rolling_sarima(train_set,test_set,order=(1,1,1),season=4 ):
    """


    Parameters
    ----------
    train_set :  Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    order : Tuple, optional
        The order of sarima model. The default is (1,1,1).
    season : Tuple, optional
        The seasonality of the model. The default is 4.

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """
    x,y=test_set.shape
    #ERRORS#
    MAPE=np.zeros([y,])
```

```
    ToS=np.zeros([y,])
    MaxDif=0.0

    for i in tqdm(range(y)):
        train=list(train_set[:,i])
        for ii in range(x):


model=ARIMA(train,order=order,seasonal_order=(1,1,1,season),enforce_invertibi
lity=False)
            m_f=model.fit()
            F=m_f.forecast()
            #ERRORS RECORD#
            MAPE[i] += da.MAPE(m_f.forecast(),test_set[ii,i])
            ToS[i] += da.TotalSum(F,test_set[ii,i])
            MaxDif=max(MaxDif,np.absolute(F-test_set[ii,i]))

            train.append(test_set[:,i][ii])
    return np.average(MAPE/x), np.sum(ToS) , float(MaxDif)




def rolling_naive(train_set,test_set,coef=(0,1)):
    """


    Parameters
    ----------
    train_set : Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    coef : Tuple, optional
        The weight of Naive !SOS! the first always be 0. The default is
(0,1).

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """
    x,y=test_set.shape
```

```
    #ERRORS
    MAPE=np.zeros([y,])
    ToS=np.zeros([y,])
    MaxDif=0.0

    time_series=np.concatenate((train_set,test_set),axis=0)
    for _ in range(x):
        naived=da.naive(time_series, coef)
        actual=time_series[-1,:]
        #ERRORS RECORD#
        MAPE += da.MAPE(naived[-1,:],actual)
        ToS += da.TotalSum(naived[-1,:],actual)
        MaxDif=max(MaxDif,np.max(np.absolute(naived[-1,:]-actual)))

        time_series=np.delete(time_series,-1,axis=0)

    return np.average(MAPE/x) , np.sum(ToS) , float(MaxDif)
```

Neural Networks, LSTM, Random Forest

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.preprocessing.sequence import TimeseriesGenerator
from sklearn.ensemble import RandomForestRegressor
from tqdm import tqdm
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import LSTM
import pandas as pd
import pickle
import dataprep as da

def timeseries(timeseries,n_series):
    n=len(timeseries)-n_series
    x=np.zeros([n,n_series])
    y=np.zeros([n,1])
    for i in range(n):
```

```
        x[i,:]=timeseries[i:i+n_series].ravel()
        y[i]=timeseries[i+n_series]

    return x,y

def LSTMRolling(train_set,test_set,n_pri_steps=8):
    """
    Long Short Term Memory

    Parameters
    ----------
    train_set : Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    n_pri_steps : Int, optional
        The number of steps before to took into account. The default is 8.

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """
    x,y=test_set.shape
    #Errors
    MAPE=np.zeros([y,])
    ToS=np.zeros([y,])
    MaxDif=0.0

    for i in tqdm(range(y)):
        #Data prep
        train=pd.Series(train_set[:,i]).to_frame()
        #Scale data
        scaler = MinMaxScaler()
        scaler.fit(train)
        scaled_train = scaler.transform(train)
        #Batchs

generator=TimeseriesGenerator(scaled_train,scaled_train,length=n_pri_steps,ba
tch_size=1)
        test=pd.Series(test_set[:,i]).to_frame()
        scaled_test=scaler.transform(test)
        time_series=np.concatenate((scaled_train,scaled_test),axis=0)
        actual_results=test_set[:,i].copy()
```

```
        #define model
        model=Sequential()
        model.add(LSTM(100,input_shape=(n_pri_steps,1)))
        model.add(Dense(1))
        model.compile(optimizer='adam',loss='mse')
        #model fit
        model.fit(generator,epochs=20,verbose=0,use_multiprocessing=False)


        for _ in range(scaled_test.shape[0]):
            sc_predict=model.predict(     time_series[-n_pri_steps-1:-
1].reshape((1,n_pri_steps,1))    )
            F=scaler.inverse_transform(sc_predict)
            #ERRORS RECORD#
            MAPE[i] += da.MAPE(F, actual_results[-1])
            ToS[i] += da.TotalSum(F,actual_results[-1])
            MaxDif= max(MaxDif,np.absolute(F-actual_results[-1]))


            time_series=np.delete(time_series,-1,axis=0)
            actual_results=np.delete(actual_results,-1,axis=0)

    return np.average(MAPE/x),np.sum(ToS) , float(MaxDif)


def DLMRolling(train_set,test_set,n_pri_steps=8):
    """
    Deep Learning Model

    Parameters
    ----------
    train_set : Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    n_pri_steps : Int, optional
        The number of steps before to take into account. The default is 8.

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """
```

```
    x,y=test_set.shape
    #Errors
    MAPE=np.zeros([y,])
    ToS=np.zeros([y,])
    MaxDif=0.0

    for i in tqdm(range(y)):
        #Data prep
        train=pd.Series(train_set[:,i]).to_frame()
        scaler = MinMaxScaler()
        scaler.fit(train)
        scaled_train = scaler.transform(train)
        xx,yy=timeseries(scaled_train,n_pri_steps)
        test=pd.Series(test_set[:,i]).to_frame()
        scaled_test=scaler.transform(test)
        time_series=np.concatenate((scaled_train[-
n_pri_steps:],scaled_test),axis=0)
        x_test,y_test=timeseries(time_series, n_pri_steps)

        #define model
        model=Sequential()
        model.add(Dense(10,activation='relu',input_shape=(n_pri_steps,)))
        model.add(Dense(10,activation='relu'))
        model.add(Dense(1))
        model.compile(optimizer='adam',loss='mse')
        #model fit

model.fit(x=xx,y=yy,epochs=20,batch_size=1,verbose=0,use_multiprocessing=True
)

        F=scaler.inverse_transform(model.predict(x_test))
        #ERRORS RECORD#
        MAPE[i]=da.MAPE(F,test_set[:,i])
        ToS[i] = da.TotalSum(F,test_set[:,i])
        MaxDif= max(MaxDif,max(np.absolute(F-test_set[:,i].reshape(-1,1))))
    return np.average(MAPE),np.sum(ToS) , float(MaxDif)


def Randomfor_rolling(train_set,test_set,n_est=100):
    """
    Random Forest Model

    Parameters
    ----------
    train_set : Numpy Array
```

```
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    n_est : Int, optional
        The number of trees . The default is 100.

    Returns
    -------
    Tuple
        MAPE,Total Sum,Max Difference.

    """
    x,y=test_set.shape
    #Errors
    MAPE=np.zeros([y,])
    ToS=np.zeros([y,])
    MaxDif=0.0
    #Training Prosses
    for i in tqdm(range(y)):
        #Training Prosses

data=TimeseriesGenerator(train_set[:,i],train_set[:,i],length=8,batch_size=
20)
        regressor=RandomForestRegressor(n_est,random_state=0)
        regressor.fit(data[0][0],data[0][1])
        time_series=np.concatenate((train_set,test_set),axis=0)
        #Test Prosses
        for ii in range(x):
            F=regressor.predict(time_series[-8:,i].reshape(1,-1))
            #Errors RECORD#
            MAPE[i] +=da.MAPE(F,time_series[-1,i])
            ToS[i] += da.TotalSum(F,time_series[-1,i])
            MaxDif= max(MaxDif,np.absolute(F-time_series[-1,i]))

            time_series=np.delete(time_series,-1,axis=0)

    return np.average(MAPE/x),np.sum(ToS) , float(MaxDif)



# data=1
# with open("data"+str(data)+".pkl","rb") as f:
#     [train_set,test_set]=pickle.load(f)
```

```
kaggle = np.genfromtxt('data.csv', delimiter=',')
test_set=kaggle[:,[1,3,7]]
train_set=kaggle[:,[0,2,4,5,6,8,9]]

"""Dataset Divition"""
x=test_set.shape[0]
xt=int(0.8*x)
trainn_set=test_set[:xt,:]
testt_set=test_set[xt:,:]

"""Rolling Random Forest"""

trainn_set=test_set[:xt,:]
testt_set=test_set[xt:,:]
results_RF=Randomfor_rolling(trainn_set, testt_set)

"""Rolling LSTM"""

trainn_set=test_set[:xt,:]
testt_set=test_set[xt:,:]
results_LSTM=LSTMRolling(trainn_set, testt_set)
"""Rolling DLM"""

trainn_set=test_set[:xt,:]
testt_set=test_set[xt:,:]
results_DLM=DLMRolling(trainn_set, testt_set)



"""print results"""
print("\n\n          Rolling RF")
print("\nMAPE:                {:.3%} \nTotal Sum:          {:.2f} \nMax
difference:     {:.2f}".format(results_RF[0],results_RF[1],results_RF[2]))

print("\n          Rolling LSTM")
print("\nMAPE:                {:.3%} \nTotal Sum:          {:.2f} \nMax
difference:
{:.2f}".format(results_LSTM[0],results_LSTM[1],results_LSTM[2]))

print("\n          Rolling DLM")
print("\nMAPE:                {:.3%} \nTotal Sum:          {:.2f} \nMax
difference:     {:.2f}".format(results_DLM[0],results_DLM[1],results_DLM[2]))
```

KNN

```
import numpy as np
from numba import jit,prange
import pickle
import dataprep as da
from tqdm import tqdm

@jit(nopython=True,parallel=True)
def KNNregressor(train_set,test_set,k=15,mean=False,n=1):
    """
    KNN timeseries regressor

    Parameters
    ----------
    train_set : Numpy Array
        Train set of the data.
    test_set : Numpy Array
        Test set of the data.
    k : Int, optional
        Number of k - neighboors. The default is 15.
    mean : Boolean, optional
        if True -> Mean weight, if false -> Weighted . The default is False.
    n : Int, optional
        The degree of the weigth . The default is 1.

    Returns
    -------
    predict : Numpy Array
        The forcast for the last Values.

    """
    #data separation
    compare=train_set[:3,:]
    knowmed_result=train_set[3,:]
    compared=test_set[:3,:]

    #size
    x=compared.shape[1]
    y=compare.shape[1]
```

```
    rangematrix  = np.zeros((y, x))
    nearest      = np.full((y, x), -1)
    predict      = np.zeros((x,))

    if mean == True:
    #HARDCLUSTER THE AVERAGE OF K NEAREST
        for i in prange(x):
            for ii in range(y):
                rangematrix[ii,i]=np.linalg.norm(compare[:,ii]-compared[:,i])
            nearest[:,i]=np.argsort(rangematrix[:,i])
            #Mean
            predict[i]=np.sum(knowmed_result[nearest[:k,i]])/k

    else:
    #SOFT-CLUSTER R/SUM(R) for each neighboor
        for i in prange(x):
            for ii in range(y):
                rangematrix[ii,i]=np.linalg.norm(compare[:,ii]-compared[:,i])
            nearest[:,i]=np.argsort(rangematrix[:,i])
            #soft clusting coef R/Rtot for each neighbor
            nearest_temp=nearest[:k,i]

predict[i]=np.sum(knowmed_result[nearest_temp]*(rangematrix[nearest_temp,i]**
(-n)/np.sum(rangematrix[nearest_temp,i]**(-n)) ))


    return predict

###MAIN###

if __name__=="__main__":
    """DATA READ FOR SYNTHETIC"""
    data=1
    with open("data"+str(data)+".pkl","rb") as f:
        [train_set,test_set]=pickle.load(f)

    """DATA READ FOR KAGGLE"""
    # kaggle = np.genfromtxt('data.csv', delimiter=',')
    # test_set=kaggle[:,[1,3]]
    # train_set=kaggle[:,[0,2,4,5,6,7,8,9]]

    """Dataset Divition"""
    coef=np.array([0.5,0.25,0.25])
    x=test_set.shape[0]
    x,y=test_set.shape
```

```
    xt=int(0.8*x)
    k=19


    MAPE=np.zeros([y,])
    ToS=0.0
    MaxDif=0.0

    for _ in tqdm(range(x-xt)):
        #Code the Timeseries
        naived_train=da.naive(train_set,coef)
        naived_test=da.naive(test_set,coef)
        F=KNNregressor(naived_train,naived_test,k=k,mean=False)

        #ERRORS RECORD#
        MAPE += da.MAPE(F,test_set[-1,:])
        ToS += da.TotalSum(F,test_set[-1,:])
        MaxDif= max(MaxDif,np.max(np.absolute(F-test_set[-1,:])))

        train_set=np.delete(train_set,-1,axis=0)
        test_set=np.delete(test_set,-1,axis=0)

    results_KNN=( np.average(MAPE/(x-xt)), ToS,  MaxDif     )




    """print results"""
    print("\n\n           KNN")
    print("\nMAPE:                {:.3%} \nTotal Sum:           {:.2f} \nMax
difference:     {:.2f}".format(results_KNN[0],results_KNN[1],results_KNN[2]))
```

GMM

```
import numpy as np
from scipy.special import logsumexp
import dataprep as da
import pickle
from tqdm import tqdm
from sklearn.mixture import GaussianMixture
```

```
import math as m



def fill_matrix(X, mixture,cov):
    """Fills an incomplete matrix according to a mixture model

    Args:
        X: (n, d) array of incomplete data (incomplete entries =0)
        mixture: a mixture of gaussians

    Returns
        np.ndarray: a (n, d) array with completed data
    """



    def log_norm_pdf_multivariate(x, mu, sigma):
        size = x.shape[0]
        if size == mu.shape[0] and (size, size) == sigma.shape:
            det = np.linalg.det(sigma)
            if det == 0:
                raise NameError("The covariance matrix can't be singular")

            norm_const = -np.log( m.pow((2*m.pi),float(size)/2) *
m.pow(det,1.0/2) )
            x_mu = x - mu
            inv = np.linalg.inv(sigma)
            result =  -0.5 * (x_mu @ inv @ x_mu.T)
            return norm_const + result
        else:
            raise NameError("The dimensions of the input don't match")


    def log_gaussian( x, mean , var ):
        """Computes the log probablity of vector x under a normal
distribution

        Args:
            x: (d, ) array holding the vector's coordinates
            mean: (d, ) mean of the gaussian
            var: variance of the gaussian

        Returns:
            float: the log probability
```

```
        """
        d = len(x)
        log_prob = -d / 2.0 * np.log(2 * np.pi * var)
        log_prob -= 0.5 * ((x - mean)**2).sum() / var
        return log_prob




    n, d = X.shape
    X_pred = X.copy()
    K, _ = mixture[0].shape


    if cov=="spherical":
        for i in range(n):
            post=np.zeros(K)
            for j in range(K):

log_likelihood=log_gaussian(X[i,:3],mixture[0][j,:3],mixture[1][j])
                post[j]=np.log(mixture[2][j])+log_likelihood
            post=np.exp(post-logsumexp(post))
            X_pred[i,3]=np.dot(post,mixture[0][:,3])

        return X_pred


    if cov=="tied":
        for i in range(n):
            post=np.zeros(K)
            for j in range(K):

log_likelihood=log_norm_pdf_multivariate(X[i,:3],mixture[0][j,:3],mixture[1][
:3,:3])
                post[j]=np.log(mixture[2][j])+log_likelihood
            post=np.exp(post-logsumexp(post))
            X_pred[i,3]=np.dot(post,mixture[0][:,3])
        return X_pred

    if cov=="diag":
        temp=np.zeros((K,d,d))
        for i in range(K):
            temp[i,:,:]=mixture[1][i]*np.identity(d)
        mixture=(mixture[0],temp,mixture[2])
```

```
    if cov=="diag" or cov== "full":
        for i in range(n):
            post=np.zeros(K)
            for j in range(K):

log_likelihood=log_norm_pdf_multivariate(X[i,:3],mixture[0][j,:3],mixture[1][
j][:3,:3])
                post[j]=np.log(mixture[2][j])+log_likelihood
            post=np.exp(post-logsumexp(post))
            X_pred[i,3]=np.dot(post,mixture[0][:,3])

        return X_pred




if __name__ == '__main__':
    data=1
    with open("data"+str(data)+".pkl","rb") as f:
        [train_set,test_set]=pickle.load(f)

    # kaggle = np.genfromtxt('data.csv', delimiter=',')
    # test_set=kaggle[:,[1,3,7]]
    # train_set=kaggle[:,[0,2,4,5,6,8,9]]

    x,y=test_set.shape
    xt=int(0.8*x)
    coef=np.array([0.5,0.25,0.25])
    x,y=test_set.shape
    K=29
    cov="diag"   #full=compete matrix  diag=only the diagonal matrix
spherical=single variance tied-same cov matrix for all K



    MAPE=np.zeros([y,])
    ToS=0.0
    MaxDif=0.0

    for _ in tqdm(range(x-xt)):

        naived_train=da.naive(train_set,coef).T
```

```
        naived_test=da.naive(test_set,coef).T
        Filled=naived_test.copy()
        Filled[:,3]=0

gmm=GaussianMixture(n_components=K,n_init=1,max_iter=10000000000000000000,tol
=1e-5,covariance_type=cov).fit(naived_train)
        mixture=(gmm.means_,gmm.covariances_,gmm.weights_)
        results=fill_matrix(Filled,mixture,cov)
        F=results[:,3]

        #ERRORS RECORD#
        MAPE+=da.MAPE2(results[:,3],test_set[-1,:])
        ToS += da.TotalSum(F,test_set[-1,:])
        MaxDif= max(MaxDif,np.max(np.absolute(F-test_set[-1,:])))

        train_set=np.delete(train_set,-1,axis=0)
        test_set=np.delete(test_set,-1,axis=0)

    results_GMM=( np.average(MAPE/(x-xt)), ToS,  MaxDif     )


    """print results"""
    print("\n\n          GMM")
    print("\nMAPE:               {:.3%} \nTotal Sum:          {:.2f} \nMax
difference:     {:.2f}".format(results_GMM[0],results_GMM[1],results_GMM[2]))
```

Και βοηθητικό αρχείο:

```
import math as m
import numpy as np

class CustomError(Exception):
    pass

def MAPE(expected,actual):
    value= np.average(np.absolute((expected-actual)/actual)  ) #TODO fix with
average
    return value

def MAPE2(expected,actual):
    value=np.absolute((expected-actual)/actual)
    return value
```

```
def TotalSum(expected,actual):
    value=np.sum(actual-expected)
    return value



def arima_def(time_series,pososto):
    """
    Parameters
    ----------
    time_series : numpy array
        time series
        x-axis:consumers
        y-axis:time.
    pososto : float
        The percentage of test data.

    Returns
    -------
    train_set : numpy array
    test_set : numpy array
    """

    num_of_colum=time_series.shape[1]
    size=m.floor(num_of_colum*pososto)
    random_colums=np.random.choice(num_of_colum,size,replace=False)

    test_set=time_series[:,random_colums]
    train_set=np.delete(time_series,random_colums,axis=1)

    return train_set,test_set

def mto3m(data_set):
    """
    Parameters
    ----------
    data_set : numpy array
        data set

    Returns
    -------
    new_set : numpy array
        new set anagogi sto 3mhno
```

```
    """
    a=data_set.shape[0]
    gamma= a/3

    new_size=m.floor(gamma)

    new_set=np.zeros([new_size,data_set.shape[1]])
    for i in range(new_size):
        new_set[new_size-i-1,:]=np.sum(data_set[a-3*(i+1):a-3*i,:],axis=0 )


    return new_set


def naive(time_series,coef):
    """
    Parameters
    ----------
    time_series : Numpy Array
        Timeseries to be naived.
    coef : Tuple
        The weights of the naive.

    Raises
    ------
    CustomError
        When the Sum of the weight doe not sum to 1.

    Returns
    -------
    result : Numpy Array
        The Naived Timeseries.

    """
    if sum(coef)>=1.001 or sum(coef)<=0.999:
        raise CustomError("Πρέπει το sum(coef) να είναι =1")
    if len(coef)*4>time_series.shape[0]:
        raise CustomError("Πολυ μικρή χρονοσειρά για τέτοιο coef")
    result=np.zeros([4,time_series.shape[1]])
    for i in range(len(coef)):
        result += time_series[-4:,:]* coef[i]
        time_series=np.delete(time_series,-1,axis=0)
        time_series=np.delete(time_series,-1,axis=0)
        time_series=np.delete(time_series,-1,axis=0)
        time_series=np.delete(time_series,-1,axis=0)
```

```
    return result

def anti_naive(time_series,results,coef):
    """
    Parameters
    ----------
    time_series : Numpy Array
        The started Timeseries before it got Naived.
    results : Numpy Array
        The timeseries we want to Anti-Naive.
    coef : TYPE
        The weigths of Naive.

    Returns
    -------
    anti_naived : Numpy Array
        The Anti-Naive Timeseries.

    """
    n=results.shape[0]
    v=time_series.shape[0]
    d=len(coef)
    anti_naived=np.zeros([n,])
    for i in range(n):
        temp=results[i]
        for ii in range(1,d):
            temp -= time_series[v-1-4*ii,i]*coef[ii]
        anti_naived[i]=temp/coef[0]


    return anti_naived
```