Διπλωματική
Τεχνητά Νευρωνικά Δίκτυα και Μηχανική Μάθηση σε logistics

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ

UNIVERSITY OF PIRAEUS

Ψηφιακά Συστήματα

Πρόγραμμα Μεταπτυχιακών Σπουδών

Μεγάλα Δεδομένα και Αναλυτική

Αντωνοπούλου Ελένη

Υπεύθυνος Καθηγητής:
Μιχαήλ Φιλιππάκης

Πρόλογος

Η παρούσα εργασία εκπονήθηκε στα πλαίσια του προγράμματος μεταπτυχιακών σπουδών “Μεγάλα Δεδομένα και Αναλυτική” του τμήματος Ψηφιακών Συστημάτων του Πανεπιστημίου Πειραιά και αποτελεί την Διπλωματική εργασία της φοιτήτριας Αντωνοπούλου Ελένης. Πραγματεύεται την εφαρμογή τεχνητών Νευρωνικών Δικτύων και Machine Learning στον τομέα των Logistics. Στόχος είναι η βελτιστοποίηση της αποστολής δεμάτων προβλέποντας τον μελλοντικό όγκο της δουλειάς ή το μελλοντικό κέρδος. Η παραπάνω πρόβλεψη, businesswise μπορεί να βοηθήσει στον καλύτερο χρονοπρογραμματισμό ή την αποστολή ομαδοποιημένων αντικειμένων. Η ομαδοποίηση κατά την αποστολή των προϊόντων μειώνει τόσο το μεμονωμένο κόστος αποστολής όσο και το κόστος σε ανθρώπινους πόρους που απαιτείται για τη συγκέντρωση και αποστολή των παραγγελιών. Η βελτιστοποίηση μιας τέτοιας διαδικασίας παρέχει μεγάλο business value για επιχειρήσεις που δραστηριοποιούνται στον τομέα των logistics - forwarding. Θα μελετήσουμε λοιπόν τη δομή και τη λειτουργία αλγορίθμων και θα εφαρμόσουμε κάποιους από αυτούς σε ένα σύνολο πραγματικών δεδομένων προκειμένου να επιχειρήσουμε να προβλέψουμε σε βάθος χρόνου μεγέθη όπως το κέρδος και το βάρος των shipment.

Abstract

The present thesis was implemented as part of the postgraduate program "Big Data and Analytics" of the Department of Digital Systems of the University of Piraeus, by the student Antonopoulou Eleni. The thesis studies the application of Artificial Neural Networks and Machine Learning in the field of Logistics. The goal is to optimize parcel delivery by predicting future job volume or future profit. The above forecast, businesswise can help in better scheduling or by consolidating shipments. Grouping on shipment of products reduces both the individual shipping cost and the human resources required to collect and ship multiple orders. Optimizing such a process provides great business value for companies operating in the field of logistics - forwarding. So we will study the structure and operation of algorithms and apply some of them to a real data set, in order to attempt to predict overtime the volumes like profit and shipment weight.

Περιεχόμενα

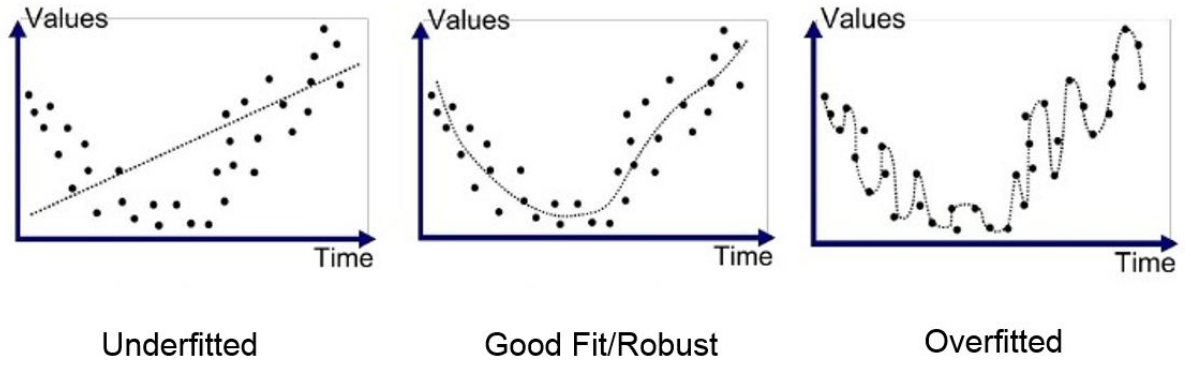
Κεφάλαιο 1 - Ορισμοί, Έννοιες

Algorithms

Σαν αλγόριθμος ορίζεται μια σειρά από βήματα που οδηγούν στην επίλυση ενός προβλήματος. Χρησιμοποιείται συνήθως για την επεξεργασία δεδομένων, κάποιον υπολογισμό και άλλες σχετικές και μαθηματικές ενέργειες. Σε μια μη τεχνική προσέγγιση, χρησιμοποιούμε αλγόριθμους σε καθημερινές εργασίες, όπως μια συνταγή για ψήσιμο ενός κέικ. Από τεχνικής άποψης, οι υπολογιστές χρησιμοποιούν αλγορίθμους για να απαριθμήσουν τις λεπτομερείς οδηγίες για τη διεξαγωγή μιας λειτουργίας. Ένας αλγόριθμος ορίζεται ως μια πεπερασμένη ακολουθία καλά καθορισμένων οδηγιών που μπορούν να υλοποιηθούν από τον υπολογιστή, συνήθως για να λυθεί μια τάξη προβλημάτων ή για να γίνει ένας υπολογισμός. Σε επόμενο στάδιο θα μελετήσουμε διάφορους αλγορίθμους και μοντέλα πρόβλεψης.

Overfitting

Ορίζεται ως το φαινόμενο που παρουσιάζεται όταν σε ένα μοντέλο πρόβλεψης δίνουμε τόση πληροφορία που παύει να προβλέπει αλλά ήδη γνωρίζει τα αποτελέσματα της πρόβλεψης. Μπορούμε να αναγνωρίσουμε ένα τέτοιο φαινόμενο όταν το ποσοστό σφάλματος του μοντέλου είναι πάρα πολύ μικρό.
Στη Στατιστική, το overfitting είναι η παραγωγή μιας ανάλυσης που αντιστοιχεί με πάρα πολύ ακρίβεια σε ένα συγκεκριμένο σύνολο δεδομένων και συνεπώς μπορεί να αποτύχει να προσαρμόσει πρόσθετα δεδομένα ή να προβλέψει αξιόπιστες μελλοντικές παρατηρήσεις. Ένα overfitted μοντέλο είναι επίσης ένα στατιστικό μοντέλο που περιέχει περισσότερες παραμέτρους από αυτές που δικαιολογούνται από τα δεδομένα.



[1]

[1] Anup Bhande, “What is underfitting and overfitting in machine learning and how to deal with it.”, *Greatom,* Mar 11, 2018

Underfitting

Το Underfitting συμβαίνει όταν ένα στατιστικό μοντέλο δεν μπορεί να καταγράψει επαρκώς την υποκείμενη δομή των δεδομένων. Ένα underfitted μοντέλο είναι ένα μοντέλο όπου λείπουν ορισμένες παράμετροι ή όροι που θα εμφανίζονταν σε ένα σωστά καθορισμένο μοντέλο. Ένα τέτοιο φαινόμενο για παράδειγμα, θα συνέβαινε κατά την τοποθέτηση ενός γραμμικού μοντέλου σε μη γραμμικά δεδομένα. Ένα τέτοιο μοντέλο θα έχει την τάση να έχει κακές προγνωστικές επιδόσεις και άρα πολύ μεγάλο σφάλμα.

Ground Truth

Στη μηχανική μάθηση, ο όρος “ground truth” αναφέρεται στην ακρίβεια της ταξινόμησης του εκπαιδευτικού συνόλου για τις εποπτευόμενες τεχνικές μάθησης. Αυτό χρησιμοποιείται σε στατιστικά μοντέλα για να αποδείξει ή να διαψεύσει ερευνητικές υποθέσεις. Ο όρος "ground truth" αναφέρεται στη διαδικασία συλλογής των κατάλληλων αντικειμενικών (αποδεδειγμένων) δεδομένων για τη δοκιμή αυτή.

Efficiency

Ο όρος αποδοτικότητα (Efficiency) είναι μια λέξη που θα χρησιμοποιηθεί πολυ στην παρούσα εργασία κατά τη μελέτη αλλά και περιγραφή των αλγορίθμων. Κατά κύριο λόγο ορίζεται ως ένα επίπεδο απόδοσης που περιγράφει τη χρήση της μικρότερης δυνατής προσπάθειας για να επιτευχθεί το υψηλότερο ποσό απόδοσης. Η αποδοτικότητα απαιτεί μείωση του αριθμού των περιττών πόρων που χρησιμοποιούνται για την παραγωγή ενός συγκεκριμένου προϊόντος, συμπεριλαμβανομένου του προσωπικού χρόνου και ενέργειας. Είναι μια μετρήσιμη έννοια που μπορεί να προσδιοριστεί χρησιμοποιώντας το λόγο της χρήσιμης εξόδου προς τη συνολική είσοδο. Σκοπός είναι η μείωση της απώλεια πόρων όπως φυσικά υλικά, ενέργεια και χρόνο, για την ολοκλήρωση της επιθυμητής ενέργειας. Στην περίπτωση εφαρμογής κάποιου αλγόριθμου, ο όρος αποδοτικότητα σχετίζεται με τον αριθμό των υπολογιστικών πόρων που χρησιμοποιούνται από τον αλγόριθμο. Ένας αλγόριθμος πρέπει να αναλυθεί για να προσδιοριστεί η χρήση των πόρων του, και η αποτελεσματικότητα ενός αλγορίθμου μπορεί να μετρηθεί με βάση τη χρήση διαφορετικών παραγόντων. Ωστόσο ο παράγοντας που κρίνει την αποδοτικότητα ενός αλγορίθμου διαφέρει κατά περίπτωση. Διαφορετικοί πόροι, όπως ο χρόνος και η πολυπλοκότητα, δεν μπορούν να συγκριθούν άμεσα, οπότε το ποιοι αλγόριθμοι θεωρούνται πιο αποτελεσματικοί, συχνά εξαρτάται από το ποιο μέτρο αποτελεσματικότητας θεωρείται σημαντικότερο.

Forecasting

Η πρόβλεψη είναι μια τεχνική που χρησιμοποιεί ιστορικά δεδομένα ως εισροές για να κάνει ενημερωμένες εκτιμήσεις που είναι προβλέψιμες για τον προσδιορισμό της κατεύθυνσης των μελλοντικών τάσεων. Οι επιχειρήσεις χρησιμοποιούν την πρόβλεψη για να καθορίσουν τον τρόπο κατανομής των προϋπολογισμών τους ή το σχέδιο για τις προβλεπόμενες δαπάνες για μια επόμενη χρονική περίοδο. Αυτό βασίζεται συνήθως στην προβλεπόμενη ζήτηση για τα αγαθά και τις υπηρεσίες που προσφέρονται.
Η πρόβλεψη αντιμετωπίζει ένα πρόβλημα ή ένα σύνολο δεδομένων. Γίνονται υποθέσεις σχετικά με την κατάσταση που αναλύεται κατά τις οποίες πρέπει να καθοριστούν οι μεταβλητές της πρόβλεψης. Με βάση τα προσδιορισμένα στοιχεία, επιλέγεται ένα κατάλληλο σύνολο δεδομένων και χρησιμοποιείται για τη χειραγώγηση των πληροφοριών. Τα δεδομένα αναλύονται και προσδιορίζεται η πρόβλεψη. Υπάρχει μια περίοδος επαλήθευσης όπου η πρόβλεψη συγκρίνεται με τα πραγματικά αποτελέσματα για να δημιουργηθεί ένα πιο ακριβές μοντέλο πρόβλεψης στο μέλλον. Όταν ο αριθμός των παραμέτρων είναι πολύ μεγάλος, δεν είναι δυνατή η υλοποίηση μιας πρόβλεψης χωρίς την χρήση κάποιου αλγορίθμου. Οι αλγόριθμοι πρόβλεψης χωρίζουν τα δεδομένα σε train και test set. Κατά την εκπαίδευση του αλγορίθμου, το μοντέλο χρησιμοποιεί το train set για να παράγει τον αλγόριθμο πρόβλεψης και το test set για να προσδιορίσει το ποσοστό σφάλματος του μοντέλου αλλά και να κρίνουμε την αποτελεσματικότητα ή μη της πρόβλεψης και κατ' επέκταση του παραγόμενου αλγορίθμου. Στην περίπτωση που το μοντέλο παράγει εξαιρετικά μικρό σφαλμα, μπορούμε να πούμε ότι στο μοντέλο έχει γίνει overfitting επομένως δεν αποτελεί αποδοτικό μοντέλο πρόβλεψη καθώς δεν γνωρίζουμε την αποδοτικότητα του σε άγνωστα δεδομένα. Ένας αλγόριθμος που παράγει μεγάλο σφάλμα δεν μπορεί να χαρακτηριστεί ως αποδοτικό μοντέλο πρόβλεψης καθώς μπορεί να παρουσιάζει underfitting.

Machine Learning Algorithms

Σαν μηχανική μάθηση ορίζεται η ιδέα ότι ένα πρόγραμμα υπολογιστή μπορεί να μάθει και να προσαρμόζεται σε νέα δεδομένα χωρίς ανθρώπινη παρέμβαση. Η μηχανική μάθηση είναι μια εφαρμογή της τεχνητής νοημοσύνης (AI) που παρέχει στα συστήματα τη δυνατότητα να μαθαίνουν και να βελτιώνουν αυτόματα από την εμπειρία χωρίς να έχουν προγραμματιστεί ρητά. Η μηχανική μάθηση επικεντρώνεται στην ανάπτυξη προγραμμάτων ηλεκτρονικών υπολογιστών που μπορούν να έχουν πρόσβαση στα δεδομένα και να τα χρησιμοποιούν για να "μάθουν για τον εαυτό τους". Η διαδικασία μάθησης ξεκινά με παρατηρήσεις ή δεδομένα, προκειμένου να αναζητήσουμε μοτίβα στα δεδομένα και να λάβουμε καλύτερες αποφάσεις στο μέλλον βάσει των παραδειγμάτων που παρέχουμε. Ο πρωταρχικός στόχος είναι να επιτρέπεται στους υπολογιστές να μαθαίνουν αυτόματα χωρίς ανθρώπινη παρέμβαση ή βοήθεια και να προσαρμόζουν ανάλογα τις διάφορες ενέργειες τους.

Οι αλγόριθμοι μηχανικής μάθησης συχνά κατηγοριοποιούνται ως **εποπτευόμενοι** ή **μη εποπτευόμενοι** αν και υπάρχουν και οι **ημι-εποπτευόμενοι**.

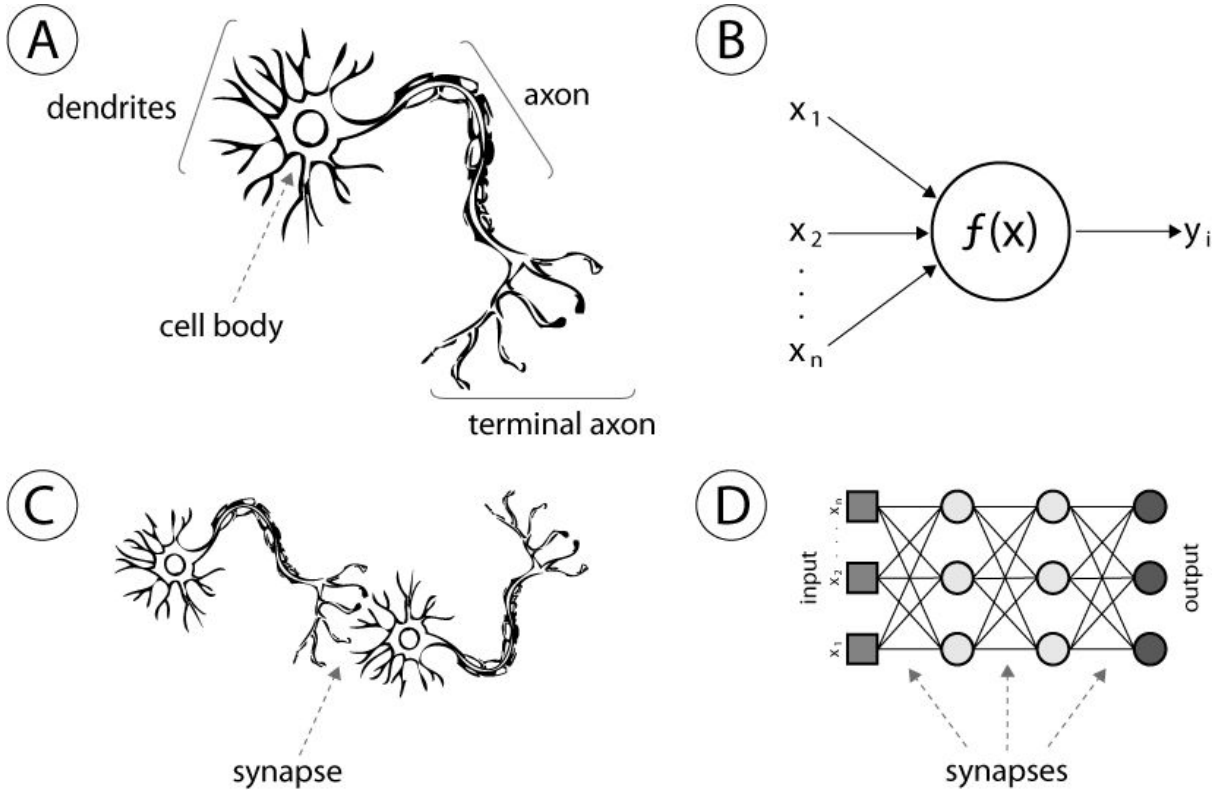
Οι **εποπτευόμενοι** (supervised) αλγόριθμοι μπορούν να εφαρμόσουν όσα έχουν μάθει στο παρελθόν σε νέα δεδομένα χρησιμοποιώντας παραδείγματα για να προβλέψουν μελλοντικά γεγονότα. Ξεκινώντας από την ανάλυση ενός γνωστού συνόλου δεδομένων (train set), ο αλγόριθμος εκμάθησης παράγει μια συνάρτηση για να κάνει προβλέψεις σχετικά με τις τιμές εξόδου. Το σύστημα είναι σε θέση να παρέχει προβλέψεις για τις τιμές εξόδου για κάθε νέα είσοδο μετά από επαρκή εκπαίδευση. Ο αλγόριθμος εκμάθησης επίσης συγκρίνει την έξοδο του με τη σωστή προβλεπόμενη έξοδο και να βρει λάθη για να τροποποιήσει ανάλογα το μοντέλο.
Αντίθετα, οι **μη εποπτευόμενοι** (unsupervised) αλγόριθμοι μηχανικής μάθησης επιλέγονται όταν οι πληροφορίες που χρησιμοποιούμε για την εκπαίδευση δεν ταξινομούνται ούτε επισημαίνονται. Οι μη εποπτευόμενες μελέτες μάθησης μπορούν να παράγουν μια συνάρτηση για να περιγράψουν μια κρυμμένη δομή από μη επισημασμένα δεδομένα. Το σύστημα δεν υπολογίζει τη σωστή έξοδο, αλλά διερευνά τα δεδομένα και μπορεί να αντλεί συμπεράσματα από σύνολα δεδομένων για να περιγράψει κρυφές δομές.

Οι **ημι-εποπτευόμενοι** (semi-supervised) αλγόριθμοι εκμάθησης εμπίπτουν κάπου μεταξύ της εποπτευόμενης και της μη εποπτευόμενης μάθησης, δεδομένου ότι χρησιμοποιούν τόσο επισημασμένα δεδομένα όσο και μη επισημασμένα για εκπαίδευση - συνήθως χρησιμοποιούν ένα μικρό πλήθος επισημασμένων δεδομένων και ένα μεγάλο μη επισημασμένων. Τα συστήματα που χρησιμοποιούν αυτή τη μέθοδο είναι σε θέση να βελτιώσουν σημαντικά την ακρίβεια της μάθησης.

Το machine learning επιτρέπει την ανάλυση μαζικών ποσοτήτων δεδομένων. Παρόλο που παρέχει γενικά ταχύτερα και ακριβέστερα αποτελέσματα για να παράξει μια πρόβλεψη, μπορεί επίσης να απαιτήσει επιπλέον χρόνο και πόρους για να εκπαιδεύσει σωστά ένα μοντέλο. Η μηχανική μάθηση αποτελεί μια αποτελεσματική μέθοδο για την επεξεργασία μεγάλου όγκου πληροφοριών. Οι αλγόριθμοι μηχανικής μάθησης είναι απίστευτα αποδοτικοί στο να κάνουν προβλέψεις ή να παρέχουν υπολογισμένες προτάσεις. Κάθε αλγόριθμος έχει μια διαφορετική προσέγγιση όσον αφορά τον τρόπο υπολογισμού του μοντέλου πρόβλεψης, για παράδειγμα κάποιοι αλγόριθμοι προσεγγίζουν το πρόβλημα σαν ένα γραμμικό μοντέλο ενώ άλλοι εφαρμόζουν μοντέλα απόφασης ή άλλες τεχνικές μάθησης. Επομένως δεν έχουν όλοι οι αλγόριθμοι μηχανικής μάθησης την ίδια εφαρμογή ή την ίδια αποδοτικότητα. Τα παραπάνω εξαρτώνται από το είδος του προβλήματος και την φύση των δεδομένων. Για να προσδιορίσουμε εάν ένα μοντέλο θεωρείται αποδοτικό ή όχι σε σχέση με τα υπόλοιπα πρέπει να χρησιμοποιήσουμε κάποια αντικειμενικά κριτήρια όπως το RMSE (Root Mean Squared error) ορίζοντας ίδιου μεγέθους train και resultset σε κάθε αλγόριθμο.

Neural Networks

Με τον όρο νευρωνικά δίκτυα συνήθως αναφερόμαστε στη δομή και τη λειτουργία του ανθρώπινου και όχι μονο εγκεφάλου. Ο ανθρώπινος εγκέφαλος αποτελείται από ένα σύνολο νευρώνων που συνδέονται μεταξύ τους διαμορφώνοντας ένα δίκτυο. Το οποιοδήποτε ερέθισμα περνάει από ένα μεγάλο αριθμό νευρώνων που όλοι μαζί συμβάλουν στην επεξεργασία του ερεθίσματος και την εξαγωγή κάποιου συμπεράσματος από τον εγκέφαλο. Με βάση το παραπάνω, μπορούμε να χαρακτηρίσουμε τα νευρωνικά δίκτυα ως ένα σύνολο αλγορίθμων οι οποίοι έχουν σχεδιαστεί για να αναγνωρίζουν και να ερμηνεύουν αισθητηριακά δεδομένα. Αυτα τα δεδομενα στα πλαίσια του ανθρώπινου εγκεφάλου προέρχονται από αισθήσεις όπως η οραση, η ακοή, η αφή κλπ. Τα μοτίβα που αναγνωρίζουν είναι αριθμητικά, που περιέχονται σε διανύσματα, στα οποία πρέπει να μεταφράζονται όλα τα δεδομένα του πραγματικού κόσμου, είτε πρόκειται για εικόνες, ήχους, κείμενο κ.α. Μπορούμε να παρομοιάσουμε τα Νευρωνικά Δίκτυα με ένα ένα στρώμα ομαδοποίησης και ταξινόμησης των δεδομένων σύμφωνα με τις ομοιότητες μεταξύ τους.

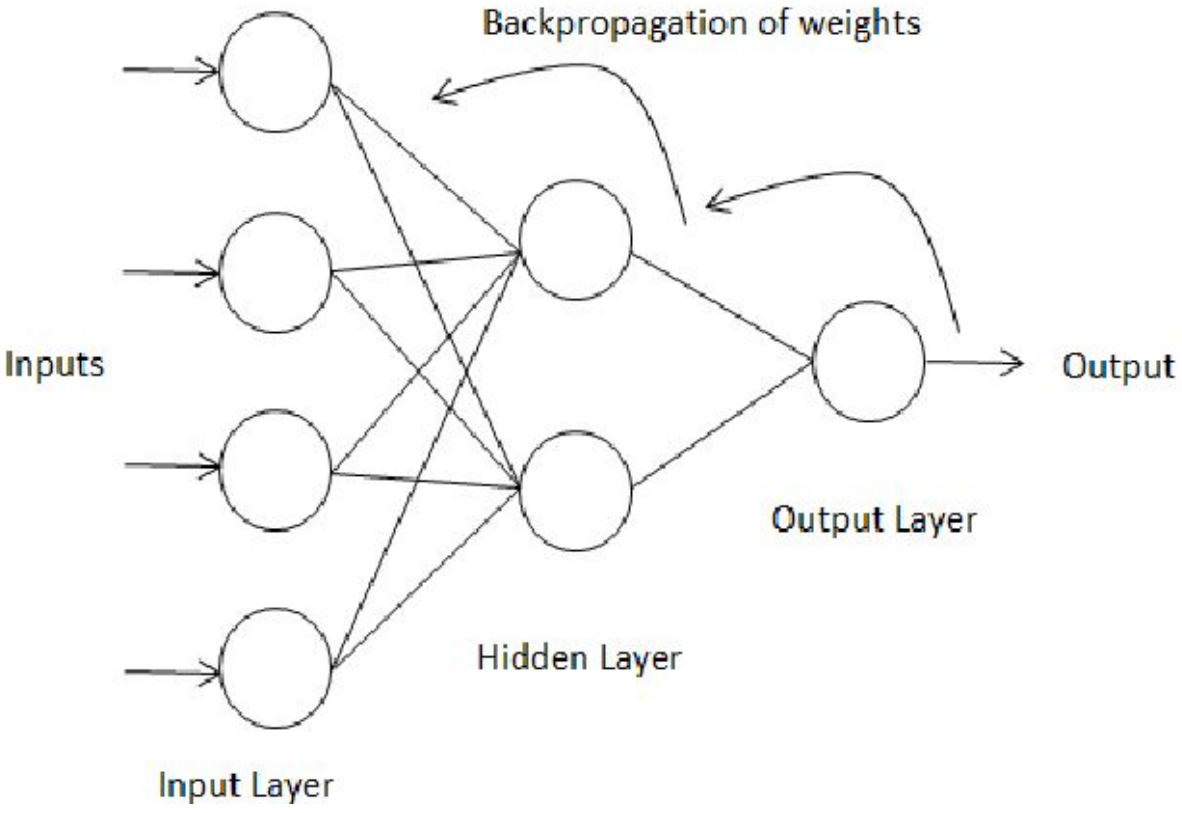


[2]

[2] Haadican, “How to Build a Simple Artificial Neural Network (ANN)”, *Medium,* Mar 28, 2019

Artificial Neural Networks

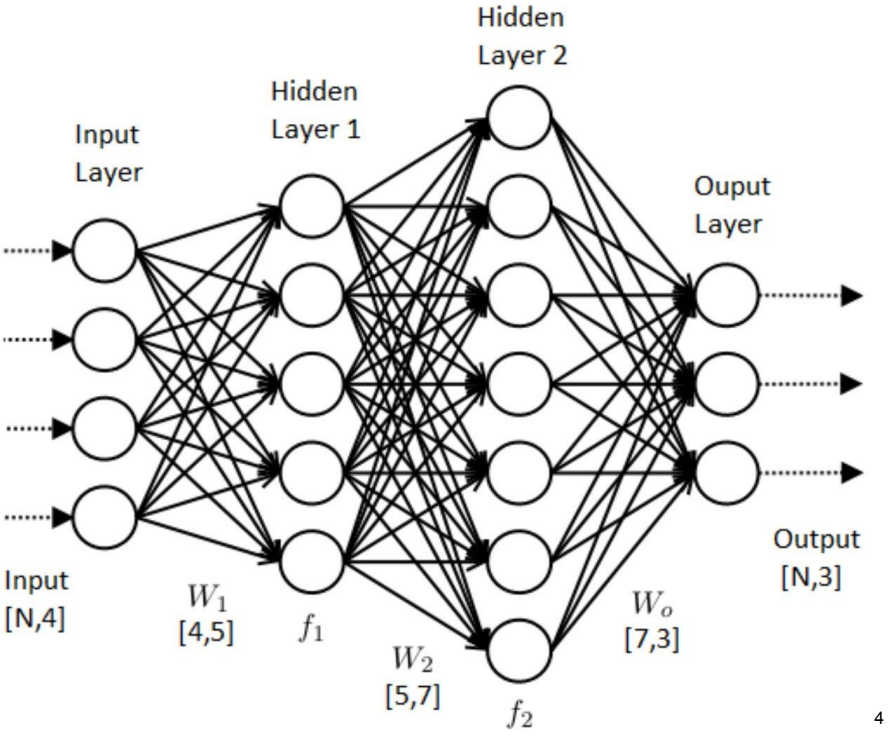
Τα τεχνητά νευρωνικά δίκτυα αποτελούν μεθόδους επεξεργασίας δεδομένων συντελώντας σε συνεργασία με άλλα εργαλεία στην εξαγωγή αποτελεσμάτων και συμπερασμάτων για την επίλυση διαφόρων προβλημάτων. Αποτελούν ένα από τα κύρια εργαλεία που χρησιμοποιούνται στη μηχανική μάθηση καθώς αναπαράγουν τον τρόπο με τον οποίο μαθαίνουμε οι άνθρωποι. Τα νευρωνικά δίκτυα αποτελούνται από στρώματα εισόδου και εξόδου, καθώς και (στις περισσότερες περιπτώσεις) από ένα κρυφό στρώμα που αποτελείται από μονάδες που μετατρέπουν την είσοδο σε κάτι που μπορεί να χρησιμοποιήσει το στρώμα εξόδου. Είναι εξαιρετικά εργαλεία για την εύρεση σχεδίων που είναι πάρα πολύ περίπλοκα ή περιέχουν πολλούς παράγοντες για έναν ανθρώπινο προγραμματιστή ή για έναν απλό αλγόριθμο μηχανικής μάθησης. Τις τελευταίες δεκαετίες έχουν γίνει σημαντικό μέρος της τεχνητής νοημοσύνης. Αυτό οφείλεται στην τεχνική **backpropagation**, η οποία επιτρέπει στα δίκτυα να προσαρμόζουν τα κρυμμένα στρώματα των νευρώνων τους.



[3]

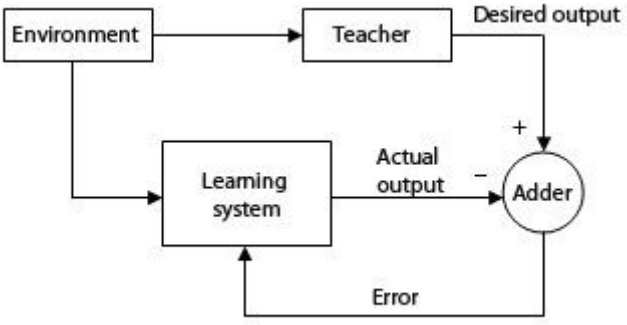
[3] “How do Neural Networks update weights and Biases during Back Propagation?”, *i2tutorials*, Sept 7 ,2019

Με λίγα λόγια ο αλγόριθμος σε καταστάσεις όπου το αποτέλεσμα δεν ταιριάζει με το αναμενόμενο, επιστρέφει σε προηγούμενους κόμβους - νευρώνες, και τους τροποποιεί προκειμένου να βελτιώσει το μοντέλο.



[4]

Τα νευρωνικά δίκτυα Feedforward είναι ο πρώτος και απλούστερος τύπος. Σε αυτά τα δίκτυα οι πληροφορίες μετακινούνται μόνο προς μία κατεύθυνση, από το στρώμα εισόδου κατευθείαν μέσω οποιουδήποτε κρυμμένου στρώματος στο στρώμα εξόδου χωρίς (backpropagation) όπως απεικονίζεται στο σχήμα Τα δίκτυα προώθησης μπορούν να κατασκευαστούν με διάφορους τρόπους, το απλούστερο από τα οποία είναι το Perceptron.



Μπορούμε επομένως να χαρακτηρίσουμε τα τεχνητά νευρωνικά δίκτυα με backpropagation ως πιο αποδοτικά καθώς σε αυτά όταν ο αλγόριθμος γυρνάει σε προηγούμενους κόμβους και τους τροποποιεί προκειμένου να βελτιώσει την αποδοτικότητα του μοντέλου. Σε αντίθεση με τους Feedforward αλγορίθμους που δεν τροποποιούνται όταν το σφάλμα που παρουσιάζουν είναι μεγάλο.

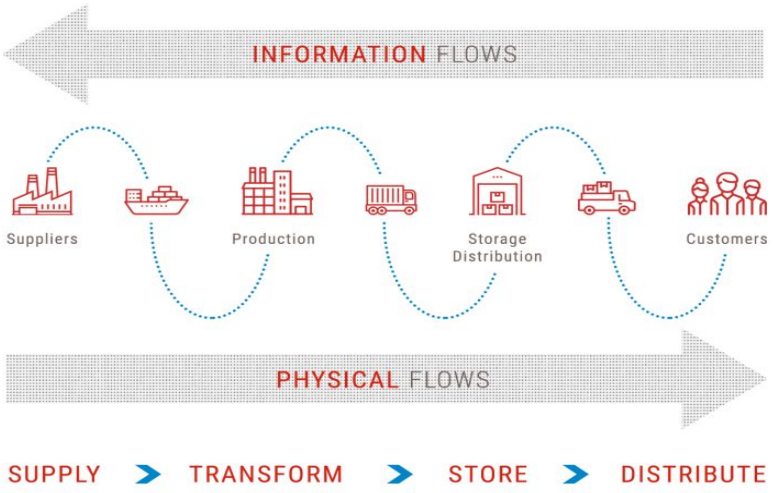
[4] Jayesh Bapu Ahire, “The Artificial Neural Networks handbook: Part 1”, *Medium*, Aug 24, 2018

Logistics - Forwarding

Με τον όρο logistics αναφερόμαστε στη διαδικασία του συντονισμού και της μετακίνησης πόρων - ανθρώπων, υλικών, αποθεμάτων και εξοπλισμού, από μια τοποθεσία έως την αποθήκευση στον επιθυμητό προορισμό. Ο όρος logistics προέρχεται από το στρατό, αναφερόμενος στην κίνηση του εξοπλισμού και των προμηθειών. Ο όρος χρησιμοποιείται πλέον ευρέως στον επιχειρηματικό τομέα, ιδίως από τις επιχειρήσεις στους τομείς της μεταποίησης, για να αναφερθεί ο τρόπος με τον οποίο χειρίζονται και μετακινούνται οι πόροι κατά μήκος της αλυσίδας εφοδιασμού.

Τα logistics αναφέρονται στη συνολική διαδικασία διαχείρισης της απόκτησης, αποθήκευσης και μεταφοράς των πόρων στον τελικό προορισμό τους. Η διαχείριση των logistics περιλαμβάνει τον εντοπισμό των μελλοντικών διανομέων και προμηθευτών και τον προσδιορισμό της αποτελεσματικότητας και της προσβασιμότητάς τους. Στόχος είναι να βρούμε τρόπους να βελτιστοποιήσουμε την διαδικασία μεταφοράς λαμβάνοντας υπόψην πολλαπλούς παράγοντες όπως το κόστος, ο χρόνος αλλά και ο όγκος της εκάστοτε μεταφοράς. Θα μελετήσουμε την εφαρμογή μοντέλων πρόβλεψης προκειμένου να πετύχουμε το παραπάνω.

Supply Chain



Μια αλυσίδα εφοδιασμού (Supply Chain) είναι ένα συνδεδεμένο δίκτυο μεταξύ ατόμων, οργανισμών, πόρων, δραστηριοτήτων και τεχνολογιών που εμπλέκονται στην κατασκευή, πώληση και διανομή ενός προϊόντος ή μιας υπηρεσίας. Αρχίζει με την παράδοση πρώτων υλών από έναν προμηθευτή σε έναν κατασκευαστή και τελειώνει με την παράδοση του τελικού προϊόντος ή της υπηρεσίας στον τελικό καταναλωτή. για την παραγωγή και τη διανομή ενός συγκεκριμένου προϊόντος στον τελικό αγοραστή. Αυτό το δίκτυο περιλαμβάνει διάφορες δραστηριότητες, άτομα, οντότητες, πληροφορίες και πόρους. Η αλυσίδα εφοδιασμού αντιπροσωπεύει επίσης τα βήματα που γίνονται για να μεταβεί το προϊόν ή η υπηρεσία από την αρχική του κατάσταση στον πελάτη.[5]

[6]

Ουσιαστικά αποτελεί την όλη διαδικασία παραγωγής, μεταφοράς και πώλησης εμπορικών αγαθών, συμπεριλαμβανομένου κάθε σταδίου από την προμήθεια υλικών και την κατασκευή των αγαθών μέχρι τη διανομή και πώληση τους.

[5] "Our Supply Chain", *GroupeSeb*
[6] "Supply Chain Management – For a steady flow and an increase in productivity!", *The Supply Chain People*

Κεφάλαιο 2 - Machine learning algorithms

Εισαγωγή

Σε πρώτη φάση θα δούμε κάποιους αλγόριθμους, τη δομή και τη χρήση τους ώστε να επιλέξουμε και να υλοποιήσουμε μερικούς από αυτούς.

Prerequisites

Πριν ξεκινήσουμε να δούμε τους διαφορετικούς αλγορίθμους μηχανικής μάθησης καλό είναι να έχουμε μια εικόνα για τις έννοιες και τους δείκτες που θα συναντήσουμε σε αυτά και τι αντιπροσωπεύουν:

- R, είναι ο πολλαπλός συντελεστής συσχέτισης (**multiple correlation coefficient**), αποτελεί ένα μέτρο της ισχύος της γραμμικής σχέσης μεταξύ της μεταβλητής απόκρισης και του συνόλου επεξηγηματικών μεταβλητών. Για την απλή γραμμική παλινδρόμηση όπου έχουμε μόνο δύο μεταβλητές, αυτό είναι το ίδιο με την απόλυτη τιμή του συντελεστή συσχέτισης Pearson. Ωστόσο, σε πολλαπλή παλινδρόμηση, μας επιτρέπει να μετρήσουμε τη συσχέτιση που περιλαμβάνει τη μεταβλητή απόκρισης και περισσότερες από μία επεξηγηματικές μεταβλητές.
- R squared (R^2) είναι το ποσοστό μεταβολής της μεταβλητής απόκρισης που εξηγείται από το μοντέλο παλινδρόμησης. Το R^2 παίρνει τιμές από 0 μέχρι 1. Όταν τιμές του R^2 είναι είναι πολύ μικρές, είναι μία υπόδειξη ότι τα δεδομένα δεν ταιριάζουν στο μοντέλο που χρησιμοποιήσαμε.
- **Standard deviation**, η τυπική απόκλιση (SD) μετράει το μέγεθος της μεταβλητότητας ή της διασποράς για ένα υποκείμενο σύνολο δεδομένων από το μέσο όρο.

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$$

$\bar{x} =$ the sample's mean
$n =$ the sample size

- **Standard error of the mean**, το τυπικό σφάλμα του μέσου (SEM) μετρά πόσο μακριά το μέσο δείγμα των δεδομένων είναι πιθανό να είναι ο πραγματικός πληθυσμός και χρησιμοποιείται για τον υπολογισμό των διαστημάτων εμπιστοσύνης και των δοκιμών σημαντικότητας.

$$\text{standard error } (\sigma_{\bar{x}}) = \frac{\sigma}{\sqrt{n}}$$

- **Outliers**, αποκλίσεις δηλαδή παρατηρήσεις που δεν φαίνονται να ταιριάζουν καλά στο μοντέλο. Μια απόκλιση μπορεί να οφείλεται σε μεταβλητότητα στη μέτρηση ή μπορεί να υποδηλώνει πειραματικό σφάλμα και μπορεί να προκαλέσει σοβαρά προβλήματα στις στατιστικές αναλύσεις, γι' αυτο το λόγο τα outliers εξαιρούνται μερικές φορές από το σύνολο δεδομένων.

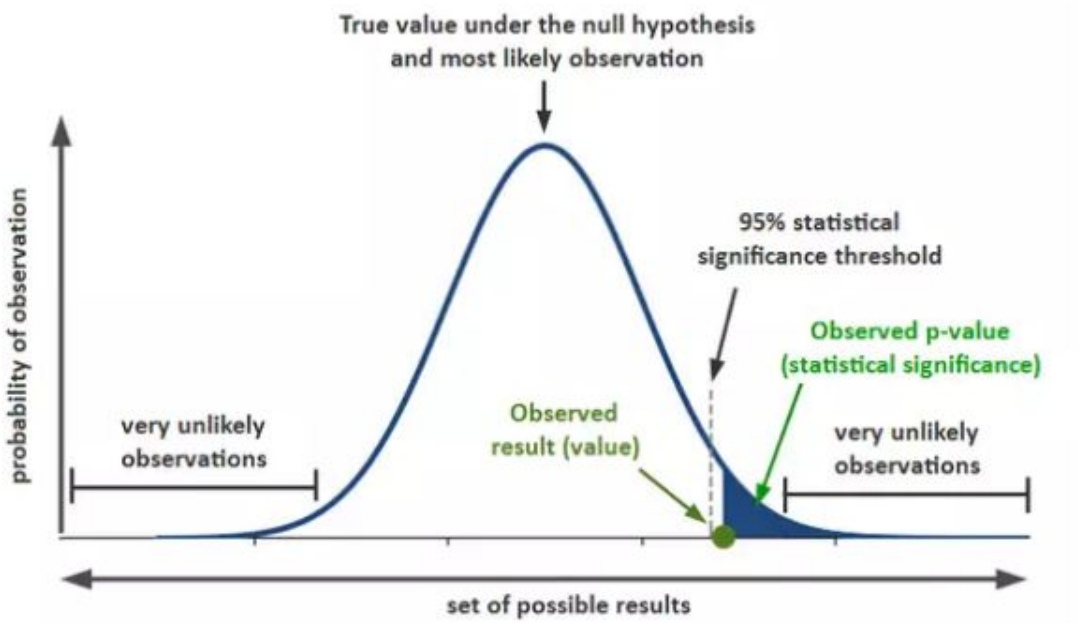
- **p value:** Στις στατιστικές υποθέσεις, η τιμή p ή η πιθανότητα είναι η πιθανότητα απόκτησης των αποτελεσμάτων των δοκιμών τουλάχιστον εξίσου ακραίων με τα αποτελέσματα που παρατηρήθηκαν κατά τη διάρκεια της δοκιμής, υποθέτοντας ότι η μηδενική υπόθεση είναι σωστή. Η τιμή p χρησιμοποιείται στο πλαίσιο δοκιμών null hypothesis προκειμένου να ποσοτικοποιηθεί η ιδέα της στατιστικής σημασίας των αποδείξεων.

Important:

Pr (observation | hypothesis) ≠ Pr (hypothesis | observation)

The probability of observing a result given that some hypothesis is true is *not equivalent* to the probability that a hypothesis is true given that some result has been observed.

Using the p-value as a "score" is committing an egregious logical error: **the transposed conditional fallacy.**



[7]

[7] Saul McLeod, "What a p-value tells you about statistical significance", *Simplypsychology*, 2019

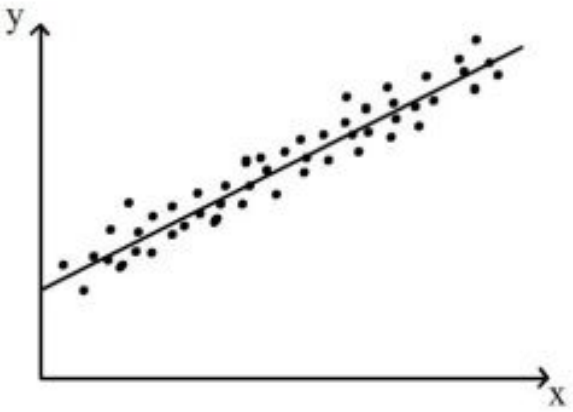
Simple linear regression

In simple linear regression we aim to predict the response for the ith individual, Yi , using the individual‟s score of a single predictor variable, X i . The form of the model is given by:
Η απλή γραμμική παλινδρόμηση είναι μια στατιστική μέθοδος για την απόκτηση μιας φόρμουλας για την πρόβλεψη τιμών μιας μεταβλητής από μια άλλη όπου υπάρχει μια αιτιώδης σχέση μεταξύ τους.

Σε απλή γραμμική παλινδρόμηση στοχεύουμε να προβλέψουμε την τιμής Yi χρησιμοποιώντας μια συνάρτηση της μορφής:

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

Περιλαμβάνει τους δύο συντελεστές παλινδρόμησης (0 και 1) και ένα τυχαίο στοιχείο που περιλαμβάνει τον υπολειπόμενο (σφάλμα) όρο (i). Το ντετερμινιστικό στοιχείο έχει τη μορφή μιας ευθείας γραμμής που παρέχει την προβλεπόμενη (μέση / αναμενόμενη) απόκριση για μια δεδομένη μεταβλητή τιμή πρόβλεψης. Οι υπολειπόμενοι όροι αντιπροσωπεύουν τη διαφορά μεταξύ της προβλεπόμενης τιμής και της παρατηρούμενης τιμής. Επομένως, τα δεδομένα μας θα πρέπει να είναι μια συλλογή σημείων που τυχαία διασκορπίζονται γύρω από μια ευθεία γραμμή με σταθερή μεταβλητότητα κατά μήκος της γραμμής. [8]



Η απλή γραμμική παλινδρόμηση είναι κατάλληλη για τη μοντελοποίηση γραμμικών τάσεων όπου τα δεδομένα κατανέμονται ομοιόμορφα γύρω από τη γραμμή. Εάν δεν συμβαίνει αυτό τότε θα πρέπει να χρησιμοποιούμε άλλες τεχνικές μοντελοποίησης ή και να μετασχηματίζουμε τα δεδομένα μας για να ανταποκριθούμε στις απαιτήσεις.

Στο απλό μοντέλο γραμμικής παλινδρόμησης, η έξοδος είναι ένας γραμμικός συνδυασμός ενώ οι είσοδοι έχουν μελετηθεί και χρησιμοποιηθεί εκτενώς. Είναι απλό στην εφαρμογή αλλά επιτρέπει μια περιορισμένη ευελιξία. Εάν η σχέση μεταξύ εισόδου και εξόδου δεν μπορεί να προσεγγιστεί με μεγάλη ακρίβεια, το μοντέλο θα δώσει φτωχές προβλέψεις άρα και θα εμφανίσει πολύ μεγάλο σφάλμα.

[8] “Simple Linear Regression”, *Statstutor*

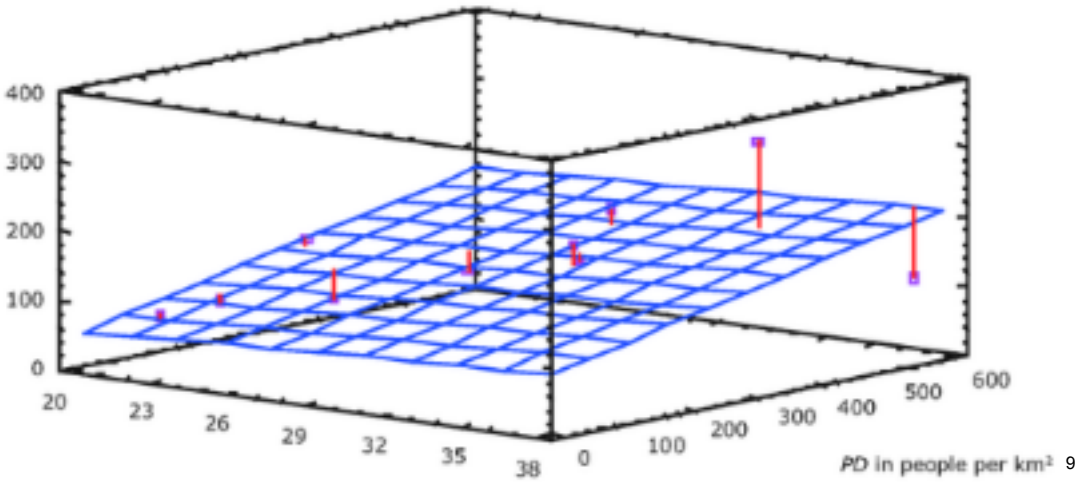
Multiple Linear Regression

Κατά την πολλαπλή γραμμική παλινδρόμηση προβλέπονται πολλαπλές συσχετισμένες εξαρτώμενες μεταβλητές, και όχι μια ενιαία μεταβλητή μεταβλητή σε σχέση με την απλή γραμμική παλινδρόμηση.

Και αυτό το μοντέλο έχει τη μορφή μιας ευθείας γραμμής όπου οι σχέσεις διαμορφώνονται χρησιμοποιώντας λειτουργίες γραμμικής πρόβλεψης των οποίων οι άγνωστες παράμετροι μοντέλου υπολογίζονται από τα δεδομένα. Σχεδόν όλα τα μοντέλα παλινδρόμησης του πραγματικού κόσμου περιλαμβάνουν πολλαπλούς παράγοντες πρόβλεψης. Η πολλαπλή γραμμική παλινδρόμηση παράγει μία συνάρτηση της μορφής:

y_i = dependent variable
x_i = expanatory variables
β_0 = y-intercept (constant term)
β_p = slope coefficients for each explanatory variable
ϵ = the model's error term (also known as the residuals)

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



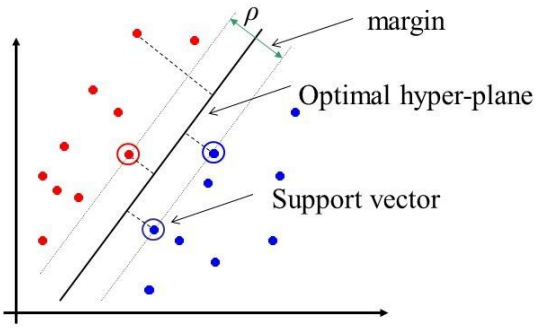
[9]

[9] "Illustration of the statistical analysis using multiple linear regression", *European Environmental Agency,* Sept 26, 2011

SVM (Support Vector Machines)

Τα SVM αναπτύχθηκαν για την επίλυση δυαδικών προβλημάτων ταξινόμησης, παρόλα αυτά δημιουργήθηκαν επεκτάσεις για την υποστήριξή τους για προβλήματα παλινδρόμησης πολλαπλών τάξεων ταξινόμησης.

Δέχονται σαν είσοδο αριθμητικές μεταβλητές ενώ τις ονομαστικές τις μετατρέπουν αυτόματα σε αριθμητικές και τα δεδομένα εισόδου κανονικοποιούνται πριν χρησιμοποιηθουν. Τα SVM βρίσκουν μια γραμμή η οποία διαχωρίζει καλύτερα τα δεδομένα εκπαίδευσης σε τάξεις.[10]



Ουσιαστικά δεν είναι ένα πιθανοτικό μοντέλο, δηλαδή δεν υποθέτει μια κατανομή πιθανότητας και συνεπώς δεν υποθέτει τυχόν τυχαιότητα. Απλώς προσπαθεί να σχεδιάσει μια απλή γραμμή για να διαχωρίσει τα σημεία δεδομένων σε δύο μέρη.

SVR (Support Vector Regression)

Το SVR (Support Vector Regression) σε αντίθεση με τα svm, λειτουργεί με την εύρεση μιας γραμμής καλύτερης προσαρμογής που ελαχιστοποιεί το σφάλμα της παραγόμενης συνάρτησης. Αυτό γίνεται χρησιμοποιώντας μια διαδικασία βελτιστοποίησης που θεωρεί μόνο τις περιπτώσεις δεδομένων που είναι πιο κοντά στη γραμμή με το ελάχιστο κόστος, σαν σύνολο δεδομένων κατάρτισης. Μπορεί επίσης να χρησιμοποιηθεί ως μέθοδος παλινδρόμησης, διατηρώντας όλα τα κύρια χαρακτηριστικά που χαρακτηρίζουν τον αλγόριθμο. Η SVR χρησιμοποιεί τις ίδιες αρχές με το SVM για ταξινόμηση, με ελάχιστες μόνο διαφορές.[11]

- Find a function, f(x), with at most ε-deviation from the target y

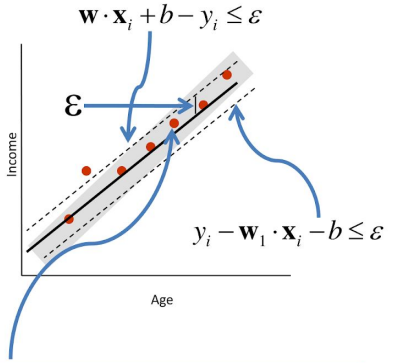
The problem can be written as a convex optimization problem

$$\min \frac{1}{2} \| \mathbf{w} \|^2$$

$$s.t.\ y_i - \mathbf{w}_1 \cdot \mathbf{x}_i - b \le \varepsilon;$$

$$\mathbf{w}_1 \cdot \mathbf{x}_i + b - y_i \le \varepsilon;$$

C: trade off the complexity



What if the problem is not feasible?

We can introduce slack variables (similar to soft margin loss function).

We do not care about errors as long as they are less than ε

Ουσιαστικά το SVM αποτελεί έναν αλγόριθμο ταξινόμηση ενώ το SVR εκτελεί παλινδρόμηση.

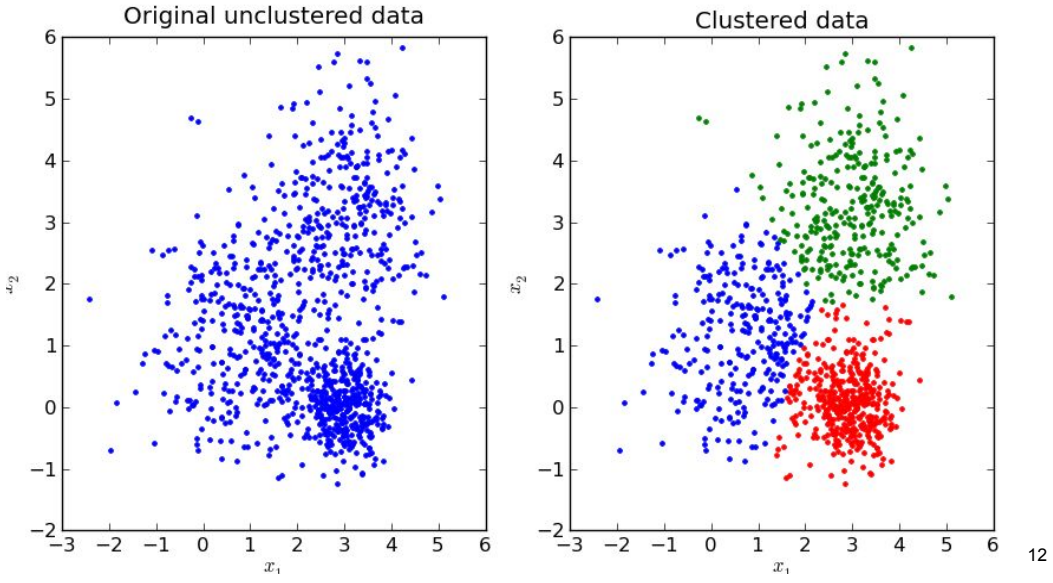
[10] Dr. Sathiya, "Support Vector Machine", *Mitosistech*, Mar 25, 2019

[11] Paul Paisitkriangkrai, "Linear Regression and Support Vector Regression", *The University of Adelaide*, Oct 24, 2012

k-means

Ο k-means είναι ένας διαιρετικός αλγόριθμος (partitioning algorithm) διαχωριστικής συσταδοποίησης, δηλαδή ο διαμερισμός των αντικειμένων γίνεται σε μη επικαλυπτόμενα υποσύνολα έτσι ώστε κάθε αντικείμενο να ανήκει σε ένα ακριβώς υποσύνολο. Ο αλγόριθμος αυτός είναι δημοφιλής εξαιτίας της απλότητας της υλοποίησης του και της γραμμικής πολυπλοκότητας του η οποία είναι της τάξης n (O(tkn)), όπου t το πλήθος των επαναλήψεων, k το πλήθος των συστάδων και n το σύνολο των στοιχείων.
Ουσιαστικά πρόκειται για το βασικό αλγόριθμο των Hartigan-Wong (1979), ο οποίος ορίζει τη συνολική εσωτερική διακύμανση (variation) της συστάδας ως το άθροισμα των τετραγώνων των Ευκλείδειων αποστάσεων μεταξύ των αντικειμένων και των αντίστοιχων κέντρων:

Κάθε παρατήρηση (xi) αντιστοιχίζεται σε μία συστάδα έτσι ώστε το άθροισμα των τετραγώνων (SS) των αποστάσεων των παρατηρήσεων από τα κέντρα των αντίστοιχων συστάδων k ελαχιστοποιείται.
Ορίζουμε τη συνολική εσωτερική διακύμανση μιας συστάδας ως:



[12]

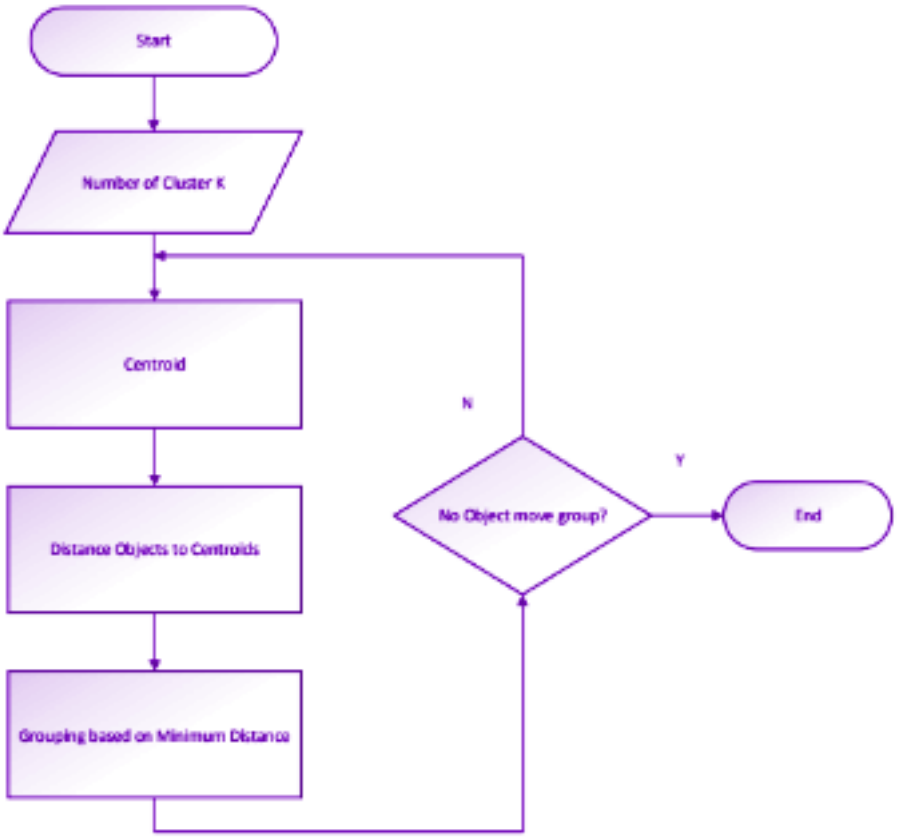
Το παραπάνω άθροισμα όσο πιο μικρό είναι, τόσο πιο κατάλληλη είναι η συσταδοποίηση που πραγματοποιήσαμε.

Ο αλγόριθμος k-means ξεκινάει με k τυχαία σημεία, τα οποία ονομάζονται κεντροειδή της συστάδας και δηλώνουν το κέντρο βάρους της συστάδας. Το k υποδηλώνει σε πόσες συστάδες θέλουμε ο αλγόριθμος να δημιουργήσει. Ο αλγόριθμος εκτελεί επαναληπτικά δύο βήματα. Το πρώτο βήμα αφορά την ανάθεση σε κάποια συστάδα, ενώ το δεύτερο βήμα αφορά τον επαναπροσδιορισμό και τη μετατόπιση του κεντροειδούς κάθε συστάδας.

[12] Niruhan Viswarupan, “K-Means Data Clustering”, *towardsdatascience*, Jul 10, 2017

Πιο αναλυτικά, όσον αφορά στο πρώτο βήμα, δηλαδή την ανάθεση σε κάποια συστάδα, ο αλγόριθμος εξετάζει κάθε δείγμα σε σχέση με τα κεντροειδή των συστάδων. Με χρήση κάποιου μέτρου απόστασης (Ευκλείδεια απόσταση, απόσταση Manhattan κα), αναθέτει το εξεταζόμενο δείγμα στη συστάδα, της οποίας το κεντροειδές είναι το πλησιέστερο ως προς το συγκεκριμένο δείγμα. Στο δεύτερο βήμα, παίρνοντας τον μέσο όρο των δειγμάτων κάθε συστάδας, επαναϋπολογίζονται τα κεντροειδή της κάθε συστάδας, ώστε το κεντροειδές να είναι πιο αντιπροσωπευτικό στην πρόσφατα διαμορφωμένη συστάδα.
Ο αλγόριθμος εκτελεί επαναληπτικά αυτά τα δύο βήματα, μέχρις ότου τα κεντροειδή των συστάδων να μετατοπίζονται ελάχιστα και σε απόσταση μικρότερη από κάποια δοθείσα τιμή κατωφλίου. Ως εναλλακτικό κριτήριο τερματισμού του αλγορίθμου μπορεί να χρησιμοποιηθεί και ο αριθμός επαναλήψεων του αλγορίθμου.

Ένα λογικό διάγραμμα του αλγορίθμου k-means φαίνεται παρακάτω:

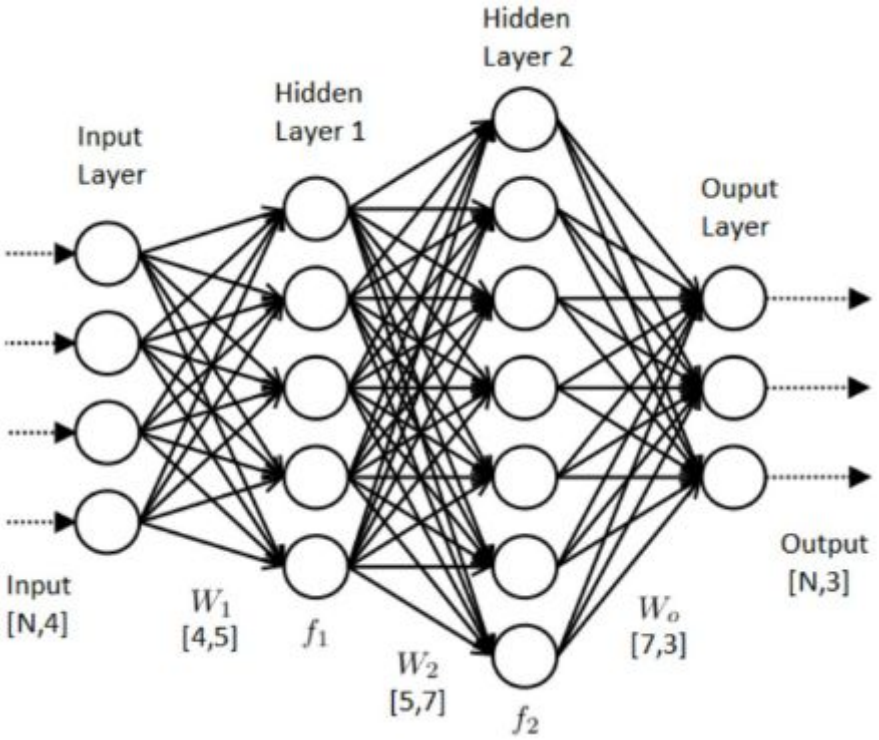


[13]

[13] Unnati R. Raval, Chaita Jani, “Implementing & Improvisation of K-means Clustering Algorithm”, *IJCSMC, Vol. 5, Issue. 5*, May 2016

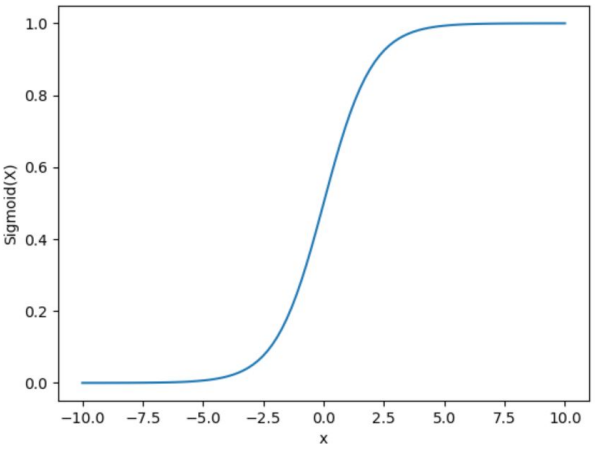
Multilayer perceptron

Το multilayer perceptron model είναι ένα νευρωνικό δίκτυο που είναι ικανό να ανταπεξέλθει σε πολύ απαιτητικά προβλήματα. Η δομή του αποτελείται από ένα input layer, ένα output layer και ένα ή περισσότερα hidden layers. Το Multilayer perceptron είναι ένα fully connected network, δηλαδή ο κόμβος σε κάθε layer συνδέεται με όλους τους κόμβους του επόμενου layer. Αποτελεί ένα αλγόριθμο με επίβλεψη και είναι ένα feed forward δίκτυο που σημαίνει ότι
τα δεδομένα ταξιδεύουν από τα αριστερά προς τα δεξιά.



[14]

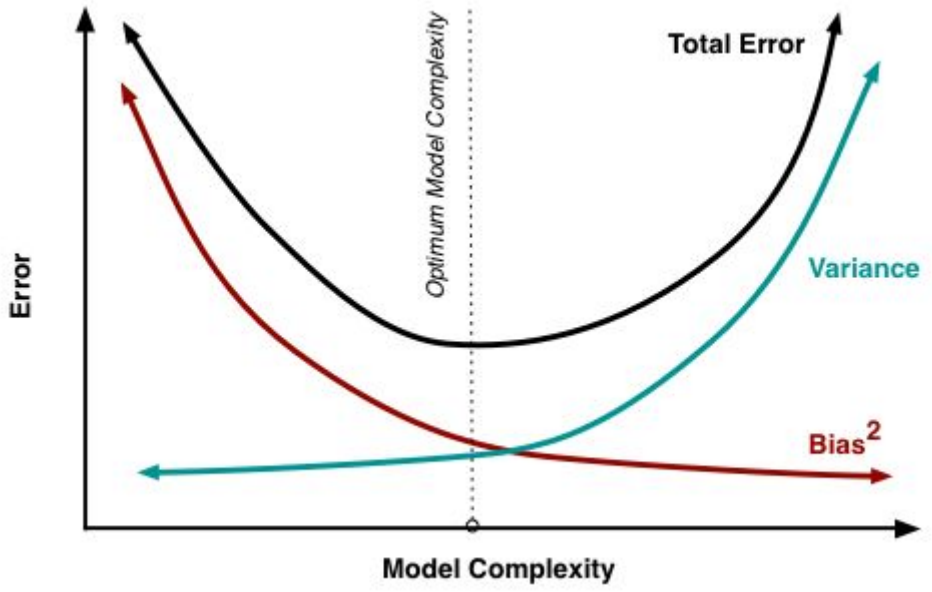
Ο κάθε κόμβος στα hidden layers είναι στην ουσία ένας perceptron ο οποίος δέχεται σαν input τα χαρακτηριστικά με τα βάρη τους, τα πολλαπλασιάζει και τα προσθέτει. Στο άθροισμα προστίθεται ακόμη και η σταθερά bias. Στην συνέχεια το άθροισμα περνάει από τη συνάρτηση ενεργοποίησης η οποία είναι μια σιγμοειδής συνάρτηση της μορφής:[15]



$$f(x) = \sigma(x) = \frac{1}{1 + e^{-x}}$$

[14] Jayesh Bapu Ahire, “The Artificial Neural Networks handbook: Part 1”, Medium, Aug 24, 2018
[15] “Implement sigmoid function using Numpy”, *Geeks for Geeks*

Ο αλγόριθμος που χρησιμοποιεί ο multilayer perceptron για να εκπαιδεύσει το δίκτυο είναι ο backpropagation. Κατά τον αλγόριθμο αυτό όταν έχουμε σήμα από έναν κόμβο output υπολογίζεται το σφάλμα του σε σχέση με την ground truth τιμή. Στη συνέχεια διασχίζουμε το δίκτυο από δεξιά προς τα αριστερά μεταβάλλοντας τις τιμές των βαρών ώστε να μειωθεί το σφάλμα. Αυτή η διαδικασία επαναλαμβάνεται για κάθε παράδειγμα οδηγώντας έτσι σε όσο το δυνατόν λιγότερο σφάλμα. Ο τρόπος με τον οποίο επιχειρεί να μειώσει το σφάλμα μπορεί να γίνει κατανοητός αν οπτικοποιήσουμε το σφάλμα από κάθε μεταβλητό βάρος.
Έτσι καταλήγουμε σε μια επιφάνεια σφάλματος (error surface) στην οποία μπορεί να εφαρμοστεί η μέθοδος gradient descent και να βρεθεί ένα ολικό ελάχιστο.



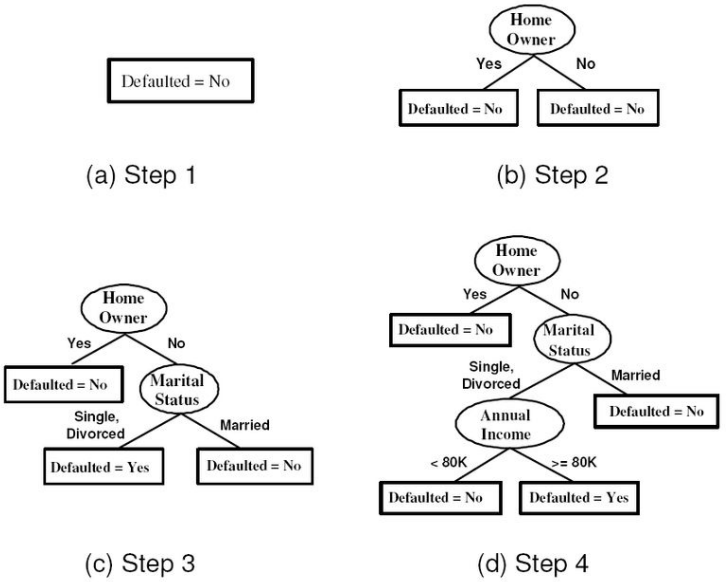
[16]

Η εκπαίδευση περιλαμβάνει την προσαρμογή των παραμέτρων ή των βαρών και του bias του μοντέλου ώστε να ελαχιστοποιηθεί το σφάλμα όπως φαίνεται στο παραπάνω διάγραμμα. Το Backpropagation χρησιμοποιείται για να κάνει αυτές τις ρυθμίσεις ζύγισης και μεροληψίας (bias) σε σχέση με το σφάλμα ενώ το ίδιο το σφάλμα μπορεί να μετρηθεί με διάφορους τρόπους, συμπεριλαμβανομένου του μέσου τετραγωνικού σφάλματος ρίζας (RMSE).

[16] Imad Dabbura, “Coding Neural Network — Regularization”, *Towards Data Science*, May 8, 2018

Decision Tree

Ένα δέντρο απόφασης είναι το δομικό στοιχείο ενός τυχαίου δάσους και είναι ένα διαισθητικό μοντέλο. Μπορούμε να σκεφτούμε ένα δέντρο απόφασης ως μια σειρά από ερωτήσεις ναι / όχι που ρωτήθηκαν για τα δεδομένα μας οδηγώντας τελικά σε μια προβλεπόμενη τάξη (ή συνεχή αξία σε περίπτωση παλινδρόμησης). Αυτό είναι ένα ερμηνεύσιμο μοντέλο επειδή κάνει τις ταξινομήσεις σαν να κάνουμε: ζητάμε μια σειρά ερωτημάτων σχετικά με τα διαθέσιμα δεδομένα που έχουμε μέχρι να φτάσουμε σε μια απόφαση (σε έναν ιδανικό κόσμο).



Οι τεχνικές λεπτομέρειες ενός δέντρου αποφάσεων αφορούν τον τρόπο με τον οποίο διαμορφώνονται οι ερωτήσεις σχετικά με τα δεδομένα. Στον αλγόριθμο CART, δημιουργείται ένα δέντρο απόφασης καθορίζοντας τις ερωτήσεις (αποκαλούμενες διαχωρίσεις κόμβων) που, όταν απαντηθούν, οδηγούν στη μεγαλύτερη μείωση της τιμής Gini. Αυτό σημαίνει ότι το δέντρο αποφάσεων προσπαθεί να σχηματίσει κόμβους που περιέχουν ένα μεγάλο ποσοστό δειγμάτων (σημεία δεδομένων) από μία κλάση, βρίσκοντας τιμές στις ιδιότητες που διαιρούν καθαρά τα δεδομένα σε κλάσεις.[17]

Ένα δέντρο απόφασης είναι ένα εργαλείο υποστήριξης αποφάσεων που χρησιμοποιεί ένα γράφημα τύπου δέντρου ή ένα μοντέλο αποφάσεων και τις πιθανές συνέπειες τους, συμπεριλαμβανομένων των εξελίξεων τυχαίων γεγονότων, του κόστους πόρων και της χρησιμότητας. Είναι ένας τρόπος προβολής ενός αλγορίθμου που περιέχει μόνο δηλώσεις υπό όρους ελέγχου. Είναι μια δομή που μοιάζει με flowchart στην οποία κάθε εσωτερικός κόμβος αντιπροσωπεύει μια "δοκιμή" σε ένα χαρακτηριστικό (π.χ. αν ένα flip νομισμάτων έρχεται πάνω από τις κεφαλές ή ουρές), κάθε κλάδος αντιπροσωπεύει το αποτέλεσμα της δοκιμής (η απόφαση λαμβάνεται αφού υπολογιστούν όλα τα χαρακτηριστικά). Οι διαδρομές από ρίζα σε φύλλο αντιπροσωπεύουν κανόνες ταξινόμησης.

[17] Hossein Hassani, Xu Huang, Emmanuel S. Silva, Mansi Ghodsi, “A Review of Data Mining Applicationsin Crime”

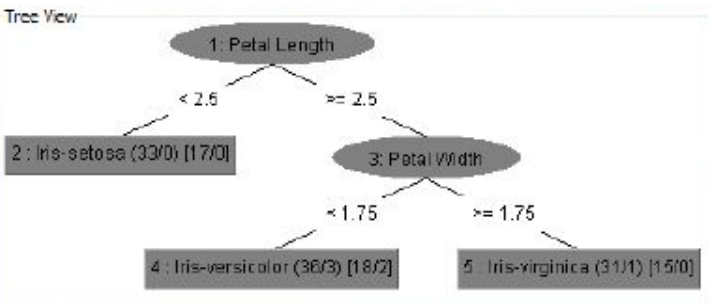
Οι αλγόριθμοι μάθησης που βασίζονται σε δέντρα θεωρούνται μια από τις καλύτερες μεθόδους μάθησης με επίβλεψη που χρησιμοποιούνται ως επί το πλείστον. Οι μέθοδοι που βασίζονται σε δέντρα ενισχύουν τα προγνωστικά μοντέλα με υψηλή ακρίβεια, σταθερότητα και ευκολία ερμηνείας. Σε αντίθεση με τα γραμμικά μοντέλα, χαρτογραφούν τις μη γραμμικές σχέσεις αρκετά καλά. Είναι προσαρμόσιμα στην επίλυση κάθε είδους προβλήματος στο χέρι (ταξινόμηση ή παλινδρόμηση). Οι αλγόριθμοι των δέντρων αποφάσεων αναφέρονται ως CART (δέντρα ταξινόμησης και παλινδρόμησης).
Μέθοδοι όπως τα δέντρα απόφασης, τα τυχαία δάση, η ενίσχυση της κλίσης χρησιμοποιούνται ευρέως σε όλα τα είδη των προβλημάτων της επιστήμης των δεδομένων.

Κοινές έννοιες που χρησιμοποιούνται στα δέντρα απόφασης είναι οι εξής:

- Root Node:Αντιπροσωπεύει ολόκληρο τον πληθυσμό ή το δείγμα και αυτό διαιρείται περαιτέρω σε δύο ή περισσότερα ομοιογενή σύνολα.
- Splitting: Είναι μια διαδικασία διαίρεσης ενός κόμβου σε δύο ή περισσότερους υποενοδικούς κόμβους.
- Decision Node: Όταν ένας κόμβος χωρίζεται σε άλλους κόμβους, τότε ονομάζεται κόμβος απόφασης.
- Leaf/ Terminal Node:Οι κόμβοι που δεν χωρίζονται ονομάζονται Κόμβος Leaf ή Terminal.
- Pruning: Όταν αφαιρούμε τους ενδιάμεσους κόμβους ενός κόμβου απόφασης, αυτή η διαδικασία ονομάζεται κλάδεμα.
- Branch / Sub-Tree: Ένα κομμάτι ολόκληρου του δέντρου ονομάζεται κλάδος ή υπο-δέντρο.
- Parent and Child Node: Ένας κόμβος, ο οποίος χωρίζεται σε άλλους κόμβους, ονομάζεται γονικός κόμβος υπο-κόμβων ενώ οι υποενοδικοί κόμβοι είναι το παιδί του γονικού κόμβου.

REPTree

Το Reduced Error Pruning Tree είναι ένας γρήγορος αλγόριθμος ο οποίος κατασκευάζει πολλαπλά δέντρα παλινδρόμησης σε διαφορετικές εκδοχές τους και από αυτά επιλέγει το βέλτιστο. Τα δέντρα κατασκευάζονται χρησιμοποιώντας πληροφορίες από το κέρδος Gain και τη διακύμανση των δεδομένων Variance, το οποίο στη συνέχεια κλαδεύει με τη μέθοδο κλαδέματος μειωμένου σφάλματος (με backfitting).
[18]



Η ρίζα και οι εσωτερικοί κόμβοι σε ένα δέντρο απόφασης αντιπροσωπεύουν τα χαρακτηριστικά και οι κόμβοι φύλλων αντιπροσωπεύουν τις κλάσεις. Ωστόσο, ο ταξινομητής δέντρων αποφάσεων μπορεί να δημιουργήσει μεγάλα δέντρα απόφασης και να γίνουν overfitted στο σετ εκπαίδευσης. Αυτό το αποτέλεσμα περιορίζει την απόδοση του μοντέλου και απαιτεί περισσότερους πόρους όσον αφορά την κατανομή της μνήμης. Αυτό το ζήτημα επιλύθηκε βελτιστοποιώντας το μέγεθος του δέντρου απόφασης.

Το RepTree χρησιμοποιεί τη λογική του δέντρου παλινδρόμησης και δημιουργεί πολλαπλά δέντρα σε διαφορετικές επαναλήψεις. Μετά από αυτό επιλέγει το καλύτερο από όλα τα δέντρα που παράγονται. Αυτό θα θεωρηθεί ως εκπρόσωπος. Κατά το κλάδεμα του δέντρου το χρησιμοποιούμενο μέτρο είναι το μέσο τετραγωνικό σφάλμα στις προβλέψεις που έγιναν.
Η διαδικασία βελτιστοποίησης επιτυγχάνεται διασχίζοντας τους εσωτερικούς κόμβους από κάτω προς τα πάνω, μια διαδικασία που ελέγχει και αντικαθιστά κάθε εσωτερικό κόμβο με την πιο συχνή κατηγορία χωρίς να επηρεάζεται η ακρίβεια των δένδρων. Η διαδικασία συνεχίζει να κλαδεύει τους κόμβους έως ότου υποστεί περαιτέρω περικοπή η οποία μειώνει την ακρίβεια.
Η μάθηση δέντρων αποφάσεων και χτίζει ένα δέντρο με βάση την αύξηση της πληροφορίας ή τη μείωση της διακύμανσης.

[18] W. Nor Haizan W. Mohamed, Mohd Najib Mohd Salleh, Abdul Halim Omar, “A Comparative Study of Reduced Error Pruning Method in Decision Tree Algorithms”, *2012 IEEE International Conference on Control System, Computing and Engineering*, Nov 23 - 25, 2012

MP5 Tree

Τα δέντρα MP5 αποτελούν δέντρα δυαδικής παλινδρόμησης στα οποία στα τελευταία στάδια - τελευταία φύλλα του δέντρου εφαρμόζεται γραμμική παλινδρόμηση. Για την κατασκευή ενός MP5 δέντρου χρησιμοποιείται ένας μετρητής απόκλισης σαν κριτήριο διαχωρισμού του δέντρου απόφασης που ονομάζεται Standard Deviation Reduction (SDR).

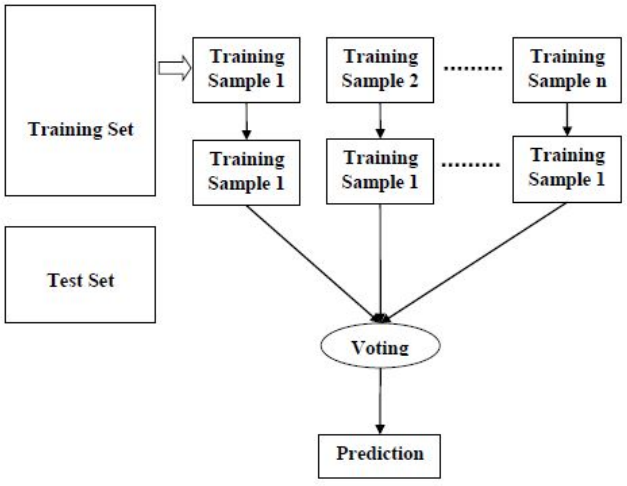
Για κάθε φύλλο δημιουργείται μία διαφορετική συνάρτηση πρόβλεψης. Αυτό έχει σαν στόχο μέσα από την ομαδοποίηση να μειώσει το σφάλμα Root Mean Squared Error.

$$SDR = sd(T) - \frac{\Sigma_{i=1}^{n}|T_i|}{|T|} * sd(T)$$

Random forest

Ο αλγόριθμος Random Forest είναι ένας αλγόριθμος ταξινόμησης με επίβλεψη. Όπως υποδηλώνει το όνομα, αυτός ο αλγόριθμος δημιουργεί το δάσος με διάφορα δέντρα. Όσο μεγαλύτερος ο αριθμός των δέντρων στο δάσος, τόσο υψηλότερη η ακρίβεια των αποτελεσμάτων.

Η βασική ιδέα είναι ο συνδυασμών πολλαπλών δέντρων απόφασης για να προσδιορίσει το τελικό αποτέλεσμα.



Σε πρώτη φάση επιλέγει τυχαία K σημεία από το σύνολο δεδομένων

Κατασκευάζει ένα δέντρο απόφασης με βάση τα παραπάνω σημεία.

Υπολογίζει ένα σκόρ για κάθε πρόβλεψη.

Τέλος επιλέγει το πιο ψηλό αποτέλεσμα πρόβλεψης ως το τελικό.
[19]

Ο αλγόριθμος Random Forest μπορεί να χρησιμοποιηθεί τόσο για την ταξινόμηση όσο και για την παλινδρόμησης, χειρίζεται τις τιμές που λείπουν και όταν έχουμε περισσότερα δέντρα στο δάσος, ο τυχαίος ταξινομητής δασών δεν θα κάνει overfit στο μοντέλο. Μπορεί επίσης να μοντελοποιήσει τον random forest classifier και για κατηγορικές τιμές.

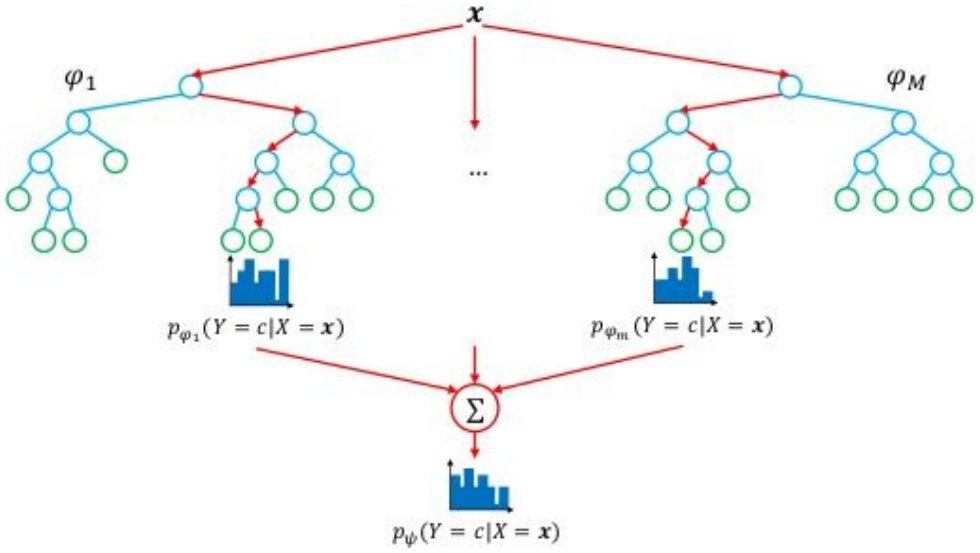
[19] “Classification Algorithms - Random Forest”, *Tutorials Point*

Πλεονεκτήματα:

- Αντιμετωπίζει το πρόβλημα του Overfitting με τον μέσο όρο ή το συνδυασμό των αποτελεσμάτων των διαφορετικών δέντρων αποφάσεων.
- Λειτουργεί καλύτερα για ένα μεγάλο εύρος δεδομένων σε σχέση με ένα ενιαίο δέντρο αποφάσεων.
- Έχει μικρότερη διακύμανση.
- Είναι πολύ ευέλικτο και έχει πολύ υψηλή ακρίβεια.
- Δεν απαιτεί τεράστιο όγκο δεδομένων, διατηρεί καλή ακρίβεια ακόμη και με δεδομένα χωρίς κλιμάκωση.
- Διατηρεί καλή ακρίβεια, ακόμη όταν λείπει μεγάλο μέρος δεδομένων.

Μειονεκτήματα:

- Μεγάλη πολυπλοκότητα.
- Δυσκολότερη και πιο χρονοβόρα η κατασκευή τους σε σχέση με τα απλά δέντρα αποφάσεων
- Απαιτούνται περισσότεροι υπολογιστικοί πόροι για την εφαρμογή του αλγόριθμου Random Forest.
- Είναι λιγότερο διαισθητικό σε περίπτωση που έχουμε μια μεγάλη συλλογή από δέντρα αποφάσεων.
- Η διαδικασία πρόβλεψης χρησιμοποιώντας τυχαία δάση είναι πολύ χρονοβόρα σε σύγκριση με άλλους αλγόριθμους.



Randomization

- Bootstrap samples
- Random selection of $K \leqslant p$ split variables } Random Forests
- Random selection of the threshold } Extra-Trees

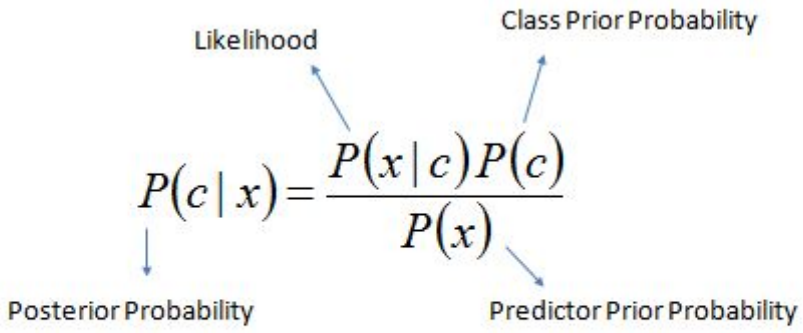
20

[20] Ilan Reinstein, “Random Forests, Explained”, *KDnuggets*

Bayes

Naïve Bayes classifiers είναι μια οικογένεια απλών "πιθανοτικών ταξινομητών" που βασίζονται στην εφαρμογή θεώρημα Bayes με ισχυρές (naïve) υποθέσεις ανεξαρτησίας μεταξύ των χαρακτηριστικών. Είναι μεταξύ των απλούστερων μοντέλων του Bayesian δικτύου.

Ο Naive Bayes είναι ένας πιθανοτικός αλγόριθμος εκμάθησης μηχανών βασισμένος στο Θεώρημα Bayes, που χρησιμοποιείται σε μια μεγάλη ποικιλία εργασιών ταξινόμησης.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$ [21]

Χρησιμοποιώντας το θεώρημα Bayes, μπορούμε να βρούμε την πιθανότητα να συμβεί ένα A ενδεχόμενο, δεδομένου ότι έχει ήδη συμβεί B. Εδώ, το B είναι η απόδειξη και η A είναι η υπόθεση. Η παραδοχή που γίνεται εδώ είναι ότι οι παράγοντες πρόβλεψης είναι ανεξάρτητοι. Οι ταξινομητές Naïve Bayes είναι εξαιρετικά κλιμακωτοί, απαιτώντας έναν αριθμό γραμμικών παραμέτρων στον αριθμό των μεταβλητών ενός προβλήματος μάθησης.

[21] Pumendu Das, “Naive Bayes Algorithm in Python”, *Codespeedy*

Multinomial Naive Bayes:

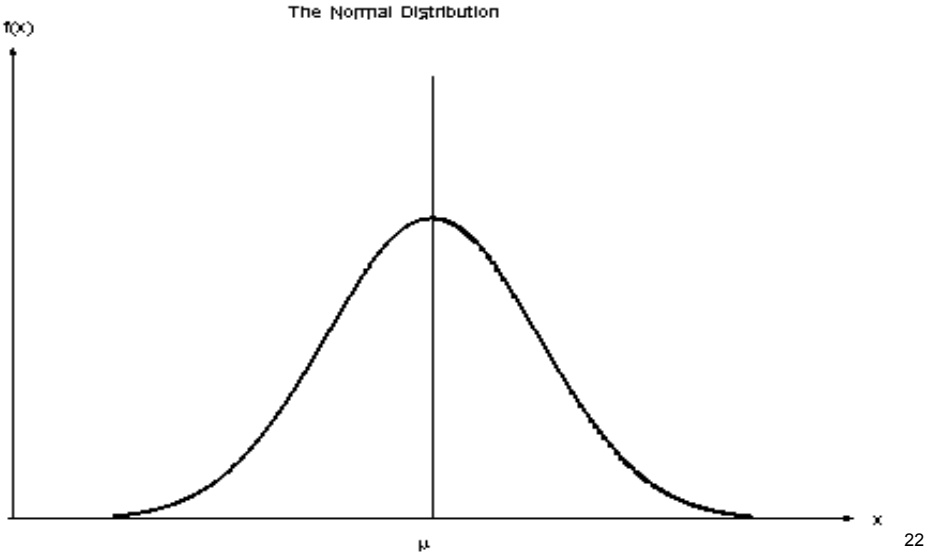
Χρησιμοποιείται κυρίως για το πρόβλημα της ταξινόμησης των εγγράφων, δηλ. Αν ένα έγγραφο ανήκει στην κατηγορία του αθλητισμού, της πολιτικής, της τεχνολογίας κ.λπ. Τα χαρακτηριστικά / οι προγνωστικοί παράγοντες που χρησιμοποιούνται από τον ταξινομητή είναι η συχνότητα των λέξεων που υπάρχουν στο έγγραφο.

Bernoulli Naive Bayes:

Είναι παρόμοιο με το multinomial naive bayes αλλά οι προγνωστικοί παράγοντες είναι οι boolean μεταβλητές. Οι παράμετροι που χρησιμοποιούμε για την πρόβλεψη της μεταβλητής κλάσης παίρνουν μόνο τιμές ναι ή όχι, για παράδειγμα, αν υπάρχει μια λέξη στο κείμενο ή όχι.

Gaussian Naive Bayes:

Όταν οι προγνωστικοί παράγοντες παίρνουν μια συνεχή τιμή και δεν είναι διακριτοί, υποθέτουμε ότι αυτές οι τιμές λαμβάνονται από μια Gaussian (κανονική) κατανομή.



[22]

Ανάλογα με τον τρόπο με τον οποίο οι τιμές βρίσκονται στο σύνολο δεδομένων αλλάζει και ο τύπος για την υπό όρους πιθανότητα.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

[22] Rohith Gandhi, “Naive Bayes Classifier”, *Towards Data Science*, May 5, 2018

Logistic Regression

Είναι ένα στατιστικό μοντέλο που στη βασική της μορφή χρησιμοποιεί μια λογική λειτουργία για να μοντελοποιήσει μια δυαδική εξαρτώμενη μεταβλητή, αν και υπάρχουν πολύ πιο πολύπλοκες επεκτάσεις. Στην παλινδρόμησης, η λογική παλινδρόμηση (ή η παλινδρόμηση logit) εκτιμά τις παραμέτρους ενός μοντέλου με τη μορφή δυαδικής παλινδρόμησης.

Το μοντέλο logistic (ή το μοντέλο logit) χρησιμοποιείται για να μοντελοποιήσει την πιθανότητα μιας συγκεκριμένης κατηγορίας ή συμβάντος όπως pass / fail, win / lose, ζωντανός / νεκρός ή υγιής / άρρωστος. Αυτό μπορεί να επεκταθεί σε μοντέλο αρκετών κατηγοριών συμβάντων όπως καθορισμός αν μια εικόνα περιέχει μια γάτα, σκύλο, λιοντάρι κλπ. Για κάθε αντικείμενο που ανιχνεύεται στην εικόνα θα έχει εκχωρηθεί μια πιθανότητα μεταξύ 0 και 1.

Το μοντέλο δυαδικής λογικής παλινδρόμησης έχει επεκτάσεις σε περισσότερα από δύο επίπεδα της εξαρτώμενης μεταβλητής: οι κατηγορικές εξόδους με περισσότερες από δύο τιμές μοντελοποιούνται με πολυωνυμική λογική παλινδρόμηση και οι πολλαπλές κατηγορίες με κανονική λογική παλινδρόμηση. Το ίδιο το μοντέλο απλώς μοντελοποιεί την πιθανότητα παραγωγής ως προς την είσοδο και δεν εκτελεί στατιστική ταξινόμηση (δεν είναι ταξινομητής), αν και μπορεί να χρησιμοποιηθεί για να κάνει έναν ταξινομητή, για παράδειγμα επιλέγοντας μια τιμή και ταξινομώντας εισόδους με πιθανότητα μεγαλύτερη από την τιμή αυτή ως μία τάξη και κάτω από την τιμή ως μία άλλη τάξη. Οι συντελεστές γενικά δεν υπολογίζονται από μια έκφραση κλειστής μορφής, σε αντίθεση με τα γραμμικά μοντέλα

Linear Discriminant Analysis

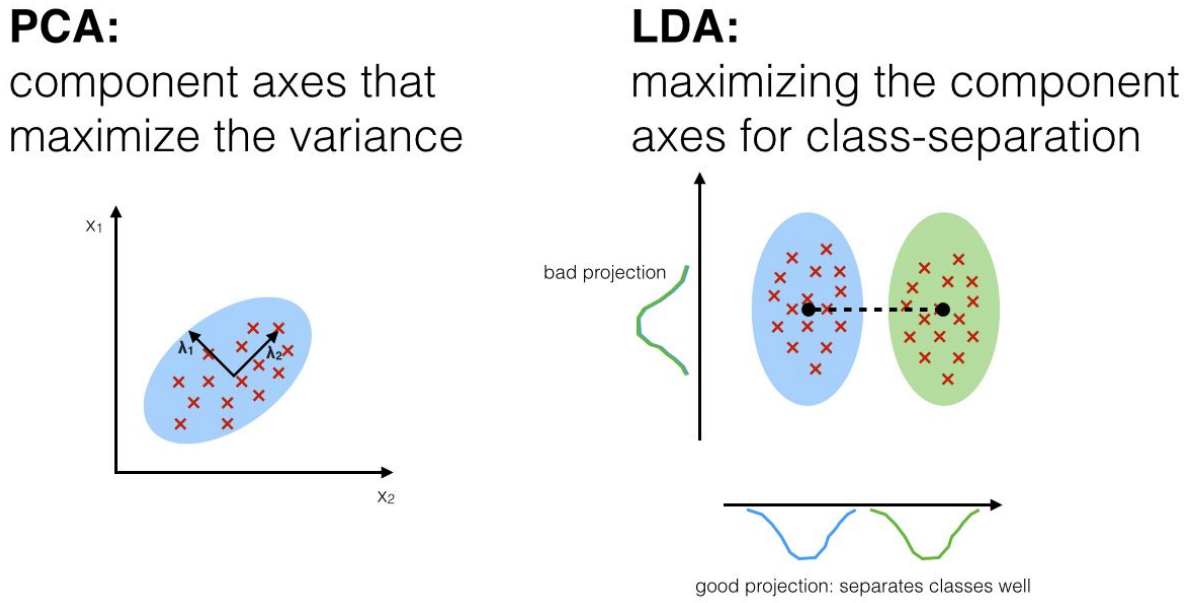
Η ανάλυση γραμμικής διακρίσεως (LDA) είναι μια τεχνική μείωσης των διαστάσεων. Καθώς μειώνονται οι μεταβλητές σε ένα σύνολο δεδομένων, διατηρώντας όσο το δυνατόν περισσότερες πληροφορίες.

Η ανάλυση γραμμικής διάκρισης (LDA), χρησιμοποιείται στην στατιστική, στην αναγνώριση προτύπων και στη μηχανική μάθηση για να βρεθεί ένας γραμμικός συνδυασμός χαρακτηριστικών που χαρακτηρίζει ή διαχωρίζει δύο ή περισσότερες κατηγορίες αντικειμένων ή συμβάντων. Ο συνδυασμός που προκύπτει μπορεί να χρησιμοποιηθεί ως ένας γραμμικός ταξινομητής ή, πιο συχνά, για τη μείωση των διαστάσεων πριν από μεταγενέστερη ταξινόμηση.

Η LDA σχετίζεται στενά με την ανάλυση της διακύμανσης (ANOVA) και την ανάλυση παλινδρόμησης, η οποία επίσης προσπαθεί να εκφράσει μια εξαρτώμενη μεταβλητή ως γραμμικό συνδυασμό άλλων χαρακτηριστικών ή μετρήσεων. Ωστόσο, η ANOVA χρησιμοποιεί κατηγορηματικές ανεξάρτητες μεταβλητές και μια συνεχή εξαρτώμενη μεταβλητή, ενώ η διακριτή ανάλυση (LDA) έχει συνεχείς ανεξάρτητες μεταβλητές και σαν κατηγοριοποιημένη εξαρτώμενη μεταβλητή την ετικέτα κλάσης. Αυτές οι μέθοδοι προτιμώνται σε εφαρμογές όπου δεν είναι λογικό να υποθέσουμε ότι οι ανεξάρτητες

μεταβλητές κατανέμονται κανονικά, γεγονός που αποτελεί θεμελιώδη παραδοχή της μεθόδου LDA.

Το LDA σχετίζεται επίσης στενά με την ανάλυση βασικών συστατικών (PCA), και οι δύο αναζητούν γραμμικούς συνδυασμούς μεταβλητών που εξηγούν καλύτερα τα δεδομένα. Η LDA προσπαθεί ρητά να μοντελοποιήσει τη διαφορά μεταξύ των κατηγοριών δεδομένων. Η PCA, αντίθετα, δεν λαμβάνει υπόψη τυχόν διαφορά στην τάξη και η ανάλυση παραγόντων βασίζει τους συνδυασμούς χαρακτηριστικών που βασίζονται σε διαφορές και όχι σε ομοιότητες. Η διακριτική ανάλυση είναι επίσης διαφορετική από την ανάλυση παράγοντα στο ότι δεν είναι μια τεχνική αλληλεξάρτησης: πρέπει να γίνει διάκριση μεταξύ ανεξάρτητων μεταβλητών και εξαρτημένων μεταβλητών (που ονομάζονται επίσης μεταβλητές κριτηρίων).



[23]

Το LDA λειτουργεί όταν οι μετρήσεις που έγιναν σε ανεξάρτητες μεταβλητές για κάθε παρατήρηση είναι συνεχείς ποσότητες. Κάθε σημείο πρέπει να έχει βαθμολογία σε ένα ή περισσότερα ποσοτικά μέτρα πρόβλεψης και μια βαθμολογία σε ένα μέτρο ομάδας. Με απλά λόγια, η ανάλυση διακριτικής λειτουργίας είναι η ταξινόμηση - η πράξη διανομής των δεδομένων σε ομάδες, κλάσεις ή κατηγορίες του ίδιου τύπου.

[23] Sebastian Raschka, “Linear Discriminant Analysis – Bit by Bit”, *Sebastianraschka,* Aug 3, 2014

Arima

Το ARIMA, συντομογραφία για το 'Auto Regressive Integrated Moving Average', είναι μια κλάση μοντέλων που εξηγεί μια χρονοσειρά με βάση τις προηγούμενες τιμές της, δηλαδή τις δικές της καθυστερήσεις και τα δικά της προηγούμενα σφάλματα πρόβλεψης, ώστε να μπορεί η εξίσωση να χρησιμοποιηθεί για την πρόβλεψη μελλοντικών τιμών.

Οποιαδήποτε «μη εποχιακή» χρονοσειρά που παρουσιάζει μοτίβα και δεν είναι τυχαίος λευκός θόρυβος μπορεί να διαμορφωθεί με μοντέλα ARIMA.

Ένα μοντέλο ARIMA χαρακτηρίζεται από 3 όρους:

- **p** είναι η σειρά του όρου AR 'Auto Regressive'. Αναφέρεται στον αριθμό των lags του Y που χρησιμοποιούνται ως πρόβλεπτες.
- **q** είναι η σειρά του MA 'Moving Average'. Αναφέρεται στον αριθμό των καθυστερημένων σφαλμάτων πρόβλεψης που πρέπει να εισαχθούν στο πρότυπο ARIMA.
- d είναι ο αριθμός των απαιτούμενων διαφοροποιήσεων για να είναι στάσιμη η χρονοσειρά, εάν η χρονοσειρά είναι ήδη ακίνητη τότε d = 0.

Εάν μια χρονολογική σειρά, έχει εποχιακά μοτίβα, τότε θα πρέπει να προσθέσουμε εποχιακούς όρους και να γίνει SARIMA, (Seasonal ARIMA). Περισσότερα για αυτό μόλις ολοκληρώσουμε την ARIMA.

Ουσιαστικά η ARIMA αποτελεί ένα μοντέλο γραμμικής παλινδρόμησης το οποίο προσπαθούμε να κάνουμε στατικό, η πιο κοινή προσέγγιση είναι να αφαιρέσουμε την προηγούμενη τιμή από την τρέχουσα. Μερικές φορές, ανάλογα με την πολυπλοκότητα της σειράς, μπορεί να χρειαστούν περισσότερες από μία διαφοροποιήσεις.

AR:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} + \epsilon_1$$

MA:

$$Y_t = \alpha + \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

ARIMA:

$$Y_t = \alpha + \beta_1 Y_{t-1} + \beta_2 Y_{t-2} + .. + \beta_p Y_{t-p} \epsilon_t + \phi_1 \epsilon_{t-1} + \phi_2 \epsilon_{t-2} + .. + \phi_q \epsilon_{t-q}$$

Σε πρώτη φάση ελέγχουμε αν η σειρά είναι ακίνητη χρησιμοποιώντας τη δοκιμή Augmented Dickey Fuller **(adfuller),** από το πακέτο statsmodels, επειδή χρειαζόμαστε διαφοροποίηση μόνο εάν η σειρά είναι μη στατική, διαφορετικά d = 0.

Εάν η τιμή p της δοκιμής είναι μικρότερη από το επίπεδο σημαντικότητας (0,05) τότε απορρίπτεται η μηδενική υπόθεση και συνάγεται ότι η χρονοσειρά είναι πράγματι ακίνητη. Εάν είναι μεγαλύτερη, βρίσκουμε το **d**.

Η μερική αυτοσυσχέτιση μπορεί να χαρακτηριστεί ως ο συσχετισμός μεταξύ της σειράς και του lag της. Έτσι, το PACF μεταφέρει την καθαρή συσχέτιση μεταξύ μιας καθυστέρησης και μιας σειράς. Όταν η καθυστέρηση PACF είναι 1 είναι αρκετά σημαντική καθώς είναι πολύ πάνω από τη γραμμή σημασίας.

Το ACF δηλώνει πόσους όρους MA είναι απαραίτητο για να αφαιρέσουμε οποιαδήποτε αυτοσυσχέτιση στη στατική χρονοσειρά.

Exponential Smoothing

Είναι μια τεχνική για την εξομάλυνση των δεδομένων χρονοσειρών χρησιμοποιώντας τη συνάρτηση του εκθετικού παραθύρου (exponential window function). Η εκθετική εξομάλυνση των δεδομένων χρονοσειρών εκχωρεί εκθετικά μειούμενα βάρη από τις νεότερες προς τις παλαιότερες παρατηρήσεις. Με άλλα λόγια, όσο μεγαλύτερα είναι τα δεδομένα, τόσο λιγότερη προτεραιότητα ("βάρος") δίνεται σε αυτα. Τα νεότερα δεδομένα θεωρούνται ως πιο συναφή και έχουν μεγαλύτερη βαρύτητα. Οι παράμετροι εξομάλυνσης (σταθερές εξομάλυνσης) καθορίζουν τα βάρη για παρατηρήσεις. Η εκθετική εξομάλυνση συνήθως χρησιμοποιείται για την πραγματοποίηση βραχυπρόθεσμων προβλέψεων, καθώς οι μακροπρόθεσμες προβλέψεις που χρησιμοποιούν αυτή την τεχνική μπορεί να είναι αρκετά αναξιόπιστες.

$$s_0 = x_0$$
$$s_t = \alpha x_t + (1-\alpha)s_{t-1},\ t > 0$$

where α is the *smoothing factor*, and $0 < \alpha < 1$.

Simple Exponential Smoothing

Η απλή (απλή) εκθετική εξομάλυνση χρησιμοποιεί σταθμισμένο κινητό μέσο με εκθετικά μειούμενα βάρη.
Η διορθωμένη με τάσεις διπλή εκθετική εξομάλυνση της Holt είναι συνήθως πιο αξιόπιστη για τον χειρισμό δεδομένων που παρουσιάζουν τάσεις, σε σύγκριση με την ενιαία διαδικασία.

Για το Exponential Smoothing χρησιμοποιούμε τα παρακάτω:

- SMA - Simple moving average

Ο απλός κινούμενος μέσος είναι ο μη σταθμισμένος μέσος όρος των προηγούμενων n δεδομένων. Αυτό διασφαλίζει ότι οι διακυμάνσεις του μέσου όρου ευθυγραμμίζονται με τις παραλλαγές στα δεδομένα αντί να μετατοπίζονται στο χρόνο.

$$\bar{p}_{\mathrm{SM}} = \frac{p_M + p_{M-1} + \cdots + p_{M-(n-1)}}{n} = \frac{1}{n}\sum_{i=0}^{n-1} p_{M-i}.$$

Το μέγεθος παραθύρου εξαρτάται από τον τύπο της πρόβλεψης που μας ενδιαφέρει να κάνουμε, όπως βραχυπρόθεσμη, ενδιάμεση ή μακροπρόθεσμη. Ένα χαρακτηριστικό του SMA είναι ότι εάν τα δεδομένα έχουν περιοδική διακύμανση, τότε η εφαρμογή ενός SMA αυτής της περιόδου θα εξαλείψει αυτήν την τάση.

- CMA

Κατα το cumulative moving average τα δεδομένα φτάνουν σε μια ταξινομημένη ροή δεδομένων και παίρνουμε το μέσο όρο όλων των δεδομένων μεχρι τη συγκεκριμένη χρονική στιγμή. Με τη προσθήκη νέων τιμών μεταβάλλεται και ο αθροιστικός μέσος. $CMA_n = \frac{x_1 + \cdots + x_n}{n}$

- EMA

Ο exponential moving average είναι ένας εκθετικός κινούμενος μέσος όρος, επίσης γνωστός ως ένας εκθετικά σταθμισμένος κινούμενος μέσος όρος ο οποίος εφαρμόζει συντελεστές στάθμισης που μειώνονται εκθετικά. Η στάθμιση για κάθε παλαιότερο στοιχείο μειώνεται εκθετικά, χωρίς να φτάσει ποτέ στο μηδέν.
Το EMA για μια σειρά Y μπορεί να υπολογιστεί αναδρομικά:

$$S_t = \begin{cases} Y_1, & t = 1 \\ \alpha Y_t + (1-\alpha)\cdot S_{t-1}, & t > 1 \end{cases}$$

Ο συντελεστής α αντιπροσωπεύει τον βαθμό μείωσης της στάθμισης, έναν σταθερό συντελεστή εξομάλυνσης μεταξύ 0 και 1. Ένας υψηλός α μειώνει τις παλαιότερες παρατηρήσεις γρηγορότερα.

Y_t είναι η τιμή σε μια χρονική περίοδο t, S_t είναι η τιμή του EMA σε οποιαδήποτε χρονική περίοδο t.

Triple Exponential Smoothing

Η τριπλή εκθετική εξομάλυνση (ονομάζεται επίσης Multiplicative Holt-Winters) είναι συνήθως πιο αξιόπιστη για παραβολικές τάσεις ή δεδομένα που δείχνουν τάσεις και εποχικότητα.

Gradient Boosting Regression

Ο αλγόριθμος Gradient Boosting είναι ένας αλγόριθμος μηχανικής μάθησης για προβλήματα παλινδρόμησης και ταξινόμησης, ο οποίος παράγει ένα μοντέλο πρόβλεψης με τη μορφή ενός συνόλου αδύναμων μοντέλων πρόβλεψης, συνήθως δέντρων αποφάσεων. Ο στόχος κάθε αλγορίθμου εποπτευόμενης μάθησης είναι να ορίσει μια συνάρτηση απώλειας και να την ελαχιστοποιήσει. Μετά την αξιολόγηση του πρώτου δέντρου, αυξάνουμε τα βάρη αυτών των παρατηρήσεων που είναι δύσκολο να ταξινομηθούν και να μειώνουμε τα βάρη για εκείνα που είναι εύκολο να ταξινομηθούν. Κατά συνέπεια, το δεύτερο δέντρο αναπτύσσεται με αυτά τα σταθμισμένα δεδομένα. Το νέο μας μοντέλο είναι ως εκ τούτου το Δέντρο 1 + Δέντρο 2. Στη συνέχεια υπολογίζουμε το σφάλμα ταξινόμησης από αυτό το νέο μοντέλο 2 δένδρων και δημιουργούμε ένα τρίτο δέντρο για να προβλέψουμε τα αναθεωρημένα υπολείμματα. Επαναλαμβάνουμε αυτή τη διαδικασία για έναν συγκεκριμένο αριθμό επαναλήψεων. Τα επόμενα δέντρα μας βοηθούν να ταξινομήσουμε τις παρατηρήσεις που δεν είναι καλά ταξινομημένες από τα προηγούμενα δέντρα. Οι προβλέψεις του τελικού μοντέλου του συνόλου είναι συνεπώς το σταθμισμένο άθροισμα των προβλέψεων που προέκυψαν από τα προηγούμενα μοντέλα δέντρων. Επομένως το Gradient Boosting εκπαιδεύει πολλά μοντέλα με βαθμιαίο, προσθετικό και διαδοχικό τρόπο.

Input: training set $\{(x_i, y_i)\}_{i=1}^n$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
   1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

   2. Fit a base learner (or weak learner, e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^n$.
   3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

   4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Κεφάλαιο 3 - Ανάλυση Προβλήματος

Ας δούμε λίγο την επιχειρησιακή διεργασία προκειμένου να κατανοήσουμε καλύτερα τα παραπάνω.

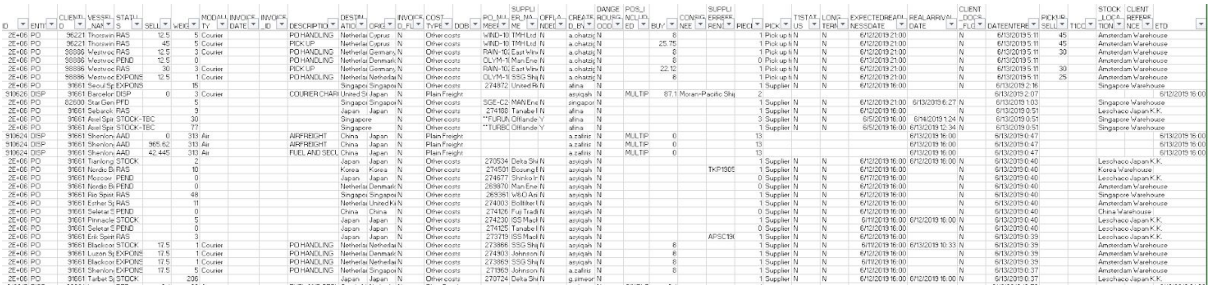
1. Ο πελάτης (πλοιοκτητρια εταιρεία) κάνει μια παραγγελία (**order**) σε ένα προμηθευτή.
2. Η εταιρία ενημερώνει τη συνεργαζόμενη μεταφορική για τα στοιχεία της παραγγελίας και του προμηθευτή. Η παραπάνω παραγγελία είτε θα παραδοθεί από τον προμηθευτή στη μεταφορική είτε θα την μαζέψει η ίδια η μεταφορική.
3. Όταν η παραγγελία έρθει στα χέρια της forwarding εταιρείας, αποθηκεύεται για κάποιο χρονικό διάστημα σε κάποια αποθήκη (κάθε πελάτης κάνει πολλές παραγγελίες από διαφορετικούς προμηθευτές για κάθε πλοίο με σκοπό όταν εκείνο πιάσει λιμάνι να στείλει πολλαπλές παραγγελίες σαν ένα **dispatch** μειώνοντας το κόστος μεταφοράς).
4. Ο πελάτης ζητάει τη μεταφορά μίας ή περισσότερων παραγγελιών από την αποθήκη σε κάποια άλλη αποθήκη/λιμάνι/καράβι, για ένα ή περισσότερα πλοία.
5. Η εταιρεία παίρνει προσφορές από αεροπορικές, κούριερ και άλλες εταιρείες και τις στέλνει στον πελάτη ανάλογα με το ζητούμενο χρόνο παράδοσης και τους εκάστοτε περιορισμούς (τελωνειακούς, dangerous goods κ.α.).
6. Κατόπιν αποδοχής της προσφοράς από τον πελάτη, κανονίζεται η μεταφορά είτε απευθείας είτε μέσω ενδιάμεσου σταθμού στον εκάστοτε προορισμό.

Σκοπός λοιπόν είναι να μπορέσουμε να προβλέψουμε το φόρτο εργασίας της εταιρείας forwarding ως προς το σύνολο του βάρους των φορτίων αλλά και ως προς το σύνολο του κέρδους για την επίτευξη καλύτερων συμφωνιών με διαμεσολαβητές και port agents. Τα παραπάνω είναι μερικά feature που μπορεί να έχουν business value για μια forwarding εταιρεία καθώς είναι σε θέση να διατηρεί ανταγωνιστικές τιμές μειώνοντας τα κόστη, άρα και να αυξάνει το profit.

Πιο συγκεκριμένα εάν μια εταιρεία Forwarding είναι σε θέση να προβλέψει τον “όγκο” της δουλειάς για το ερχόμενο έτος για ένα συγκεκριμένο πελάτη, είναι σε θέση να κάνει πιο ανταγωνιστικές συμφωνίες με τον παραπάνω άρα και είτε να κρατήσει τον πελάτη είτε να αυξήσει το συνολικό όγκο της δουλειας. Ομοίως εάν είναι σε θέσει να προβλέψει το συνολικό όγκο των μεταφορών, μπορεί να διαπραγματευτεί πιο συμφέρουσες συμφωνίες με agents σε λιμάνια, μεταφορικές, αεροπορικές κ.α. ώστε να μειώσει το κόστος (BUY) και κατ επέκταση να αυξήσει το κέρδος (BUY - SELL).

Κεφάλαιο 4 - Δεδομένα

Για την παρούσα εργασία χρησιμοποίησα ένα σύνολο δεδομένων που παρέχει πληροφορίες για τις ομαδοποιημενες μετακινήσεις πολλαπλών παραγγελιών, τα λεγόμενα dispatches για την εκάστοτε ναυτιλιακή εταιρεία.



Στήλες:
'ID', 'ENTITY', 'CLIENTID', 'VESSEL_NAME', 'STATUS', 'SELL', 'WEIGHT', 'MODALITY', 'INVOICEDATE', 'INVOICE_ID', 'DESCRIPTION', 'DESTINATION', 'ORIGIN', 'INVOICED_FLG', 'COST_TYPE', 'DOB', 'PO_NUMBER', 'SUPPLIER_NAME', 'OFFLANDED', 'CREATED_BY', 'DANGEROUSGOODS', 'POS_INCLUDED', 'BUY', 'CONSIGNEE', 'SUPPLIERREFERENCE', 'PIECES', 'PICKUP', 'T1STATUS', 'LONG_TERM', 'EXPECTEDREADINESSDATE', 'REALARRIVALDATE', 'CLIENT_DOCS_FLG', 'DATEENTERED', 'PICKUPSELL', 'T1COST', 'STOCK_LOCATION', 'CLIENTREFERENCE', 'ETD', 'CLIENT_NAME'

Το **entity** είναι είτε Order (PO) είτε Dispatch (DISP)
Το **status** παίρνει τις τιμές και δείχνει σε ποιο στάδιο βρίσκεται η παραγγελία. Κάποια status αφορούν αποκλειστικά orders και κάποια άλλα dispatches:

- EXPONS = expected on stock
- STOCK-TBC = stock to be confirmed
- PFD = prepare for dispatch
- RAS = ready at supplier
- PEND = pending
- STOCK
- DISP = dispatched
- AAD = arrived at destination
- DOB = delivery on board

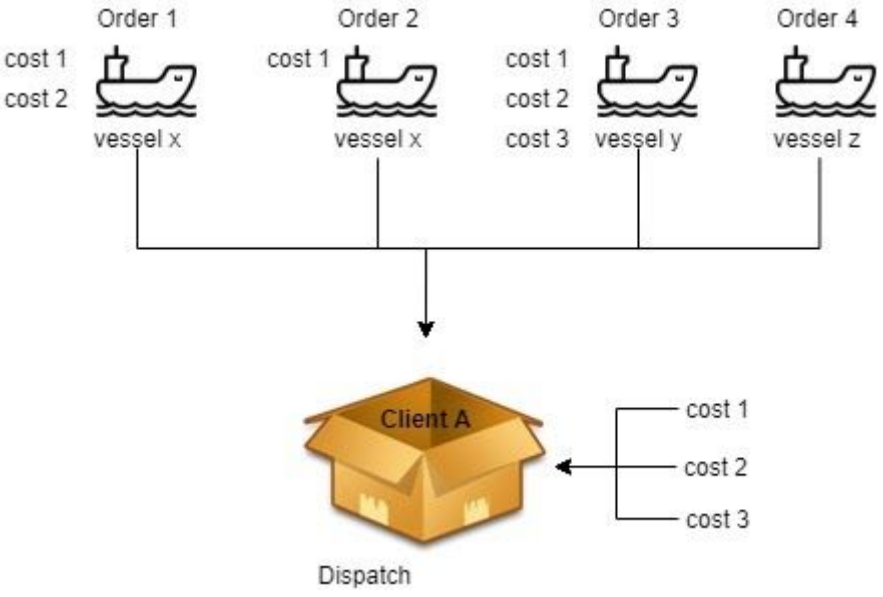
Το **modality** αφορά τον τρόπο μεταφοράς (airfreight, seafreigth, truck etc)
Το **description** είναι σε επίπεδο κόστους και δείχνει ουσιαστικά τι αντιπροσωπεύει η οποιαδήποτε χρέωση.
Τα **origin destination** αφορούν τη χώρα αφετηρία και προορισμό του order/dispatch.
Το **pos_included** παίρνει τις τιμές single , multiple και λειτουργεί σαν flag για να δείξει εάν το dispatch είναι μη συμφερον shipment (single) καθώς δεν γίνεται συμμεταφορά πολλαπλών φορτίων ώστε να μειώνεται συνολικό το κόστος των μεταφορικών.
Ως **consignee** ορίζεται ο ατζέντης (στο λιμάνι) ο οποίος παραλαμβάνει το dispatch από τη μεταφορική και παραδίδει τις παραγγελίες στο αντίστοιχο πλοίο.
Το **supplierreference** καθώς και το **clientreference** παρέχονται από το πελάτη για δική του χρήση και δεν παρέχουν κάποια χρήσιμη πληροφορία για εμάς.

Το **pickup** είναι σε επίπεδο order και δίνει πληροφορίες για το εάν θα γίνει παραλαβή της παραγγελίας από τον supplier ή αν θα φροντίσει ο ίδιος για τη μεταφορά της παραγγελίας στις αποθήκες της μεταφορικής
Το **t1status** αφορά τα τελωνειακά έγγραφα που απαιτούνται.
Τα **long_term** αφορούν την μακροχρόνια αποθήκευση των προϊόντων στις αποθήκες και λειτουργεί σαν flag, μπορεί να είναι μεταβλητή που επηρεάζει το κόστος καθώς επιφέρει κόστη στη μεταφορική αλλά τα παραπάνω κόστη δεν περνάνε απαραίτητα και στον πελάτη, ανάλογα με την εκάστοτε συμφωνία.
Το **stock_location** είναι η τοποθεσία αποθήκευσης των παραγγελιών

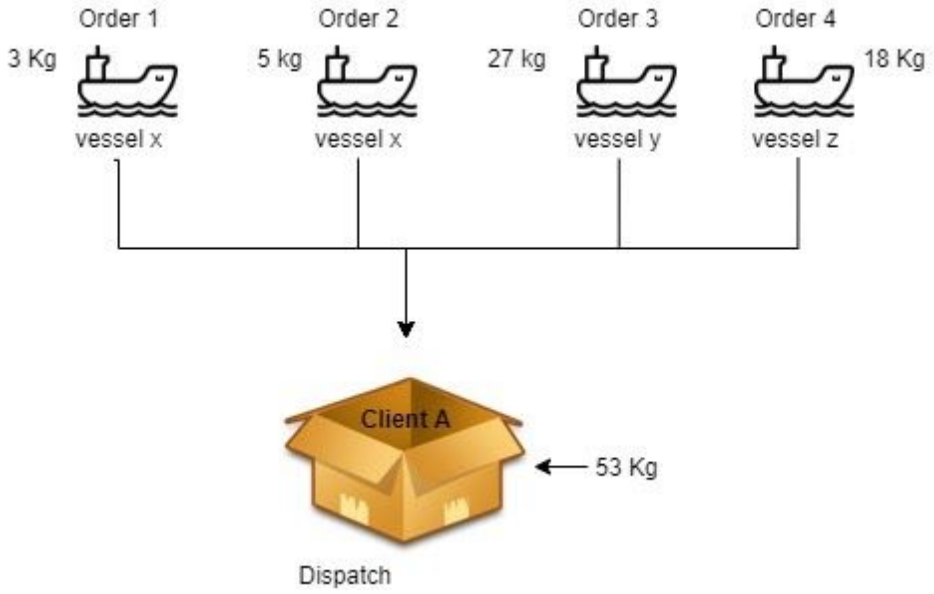
Οι στήλες Buy και Sell έχουν μετατραπεί όλες στο ίδιο νόμισμα.

Το παραπάνω dataset λοιπόν περιέχει πληροφορίες για τις μεταφορές και για τις παραγγελίες σαν σύνολο αλλά δεν δίνει πληροφορίες για το πώς ένα order δένεται με ένα dispatch, γεγονός που δυσκολεύει τη διαχείριση των δεδομένων σαν σύνολο. Εναλλακτικά θα μπορούσαμε να σπάσουμε τα δεδομένα σε order δεδομένα και dispatch δεδομένα, αλλά ούτε σε αυτήν την περίπτωση θα ήταν σωστή η διαχείριση από πλευράς κόστους καθώς κάποιες πλοιοκτητριες εταιρείες επιλέγουν να επιμερίζουν τα κόστη στα πλοία που περιέχονται στο dispatch ενώ κάποιες άλλες επιλέγουν να χρεώνουν το dispatch σαν μια οντότητα.

Συνοψίζοντας, το παραπάνω dataset περιέχει πληροφορίες που περιλαμβάνουν την αφετηρία και το προορισμό του κάθε φορτίου, το βάρος, το είδος της μεταφοράς αλλά και την φύση του. Τα φορτία (dispatches) έχουν σαν πελάτη ένα πλοίο και όχι την ναυτιλιακή που το διαχειρίζεται και κάθε φορτίο μπορεί να μεταφέρει παραγγελίες για περισσότερα από ένα πλοία της ίδιας ναυτιλιακής. Επίσης σε επίπεδο παραγγελίας έχουμε λίγο πολύ τις τις ίδιες πληροφορίες.



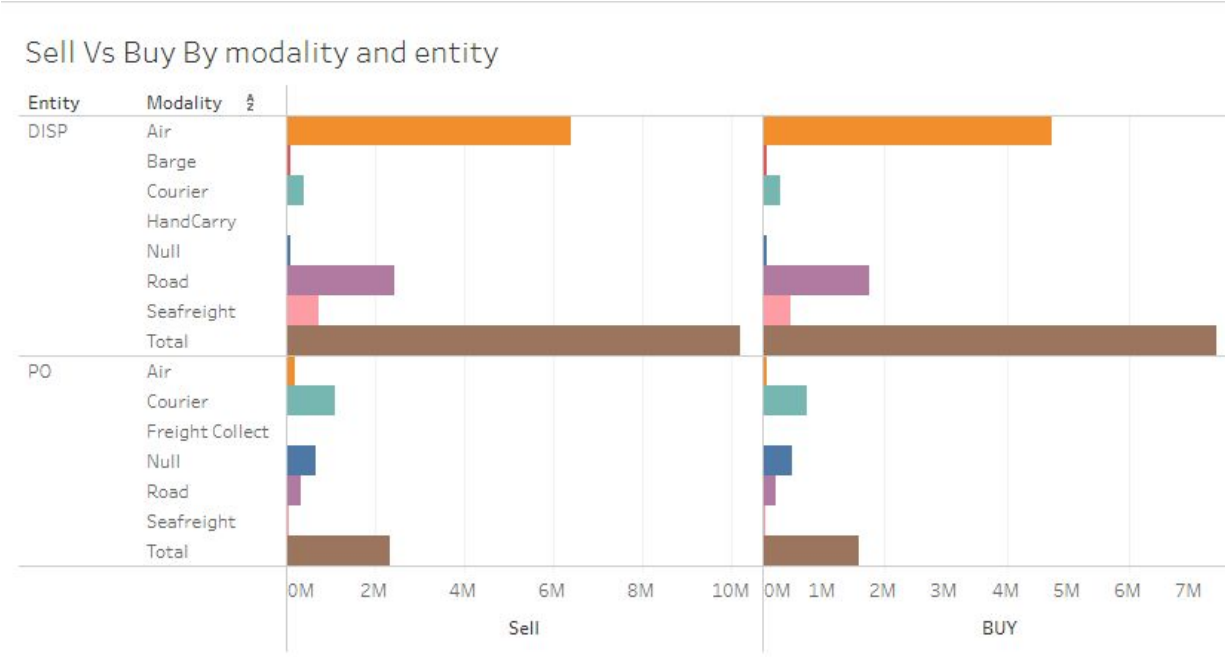
Το dataset είναι σε επίπεδο κόστους, επομένως το ίδιο ID υπάρχει περισσότερες από μια φορές. Είτε σε επίπεδο order (PO) είτε σε επίπεδο Dispatch (entity column) μπορούμε να έχουμε πολλα διαφορετικά είδη κόστους. Τέλος το Dispatch Weight αφορά το συνολικό βάρος του φορτίου, δηλαδή το άθροισμα των βαρών των παραγγελιών που συμπεριλαμβάνονται σε αυτό, ενώ σε επίπεδο order δείχνει το επιμέρους βάρος κάθε παραγγελίας. Καταλήγουμε λοιπόν στο συμπέρασμα ότι το να αθροίσουμε απλά τα δεδομένα θα έφερνε λάθος αποτελέσματα όπως πχ στην περίπτωση του βάρους όπως φαίνεται παρακάτω.

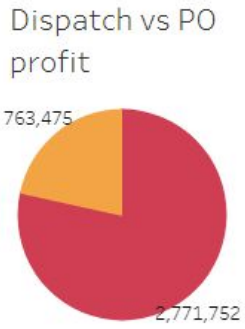


Το βάρος του dispatch είναι ήδη το άθροισμα των επιμέρους βαρών επομένως το άθροισμα των βαρών σαν απλά records θα έβγαζε λάθος νούμερο στο γενικό σύνολο.

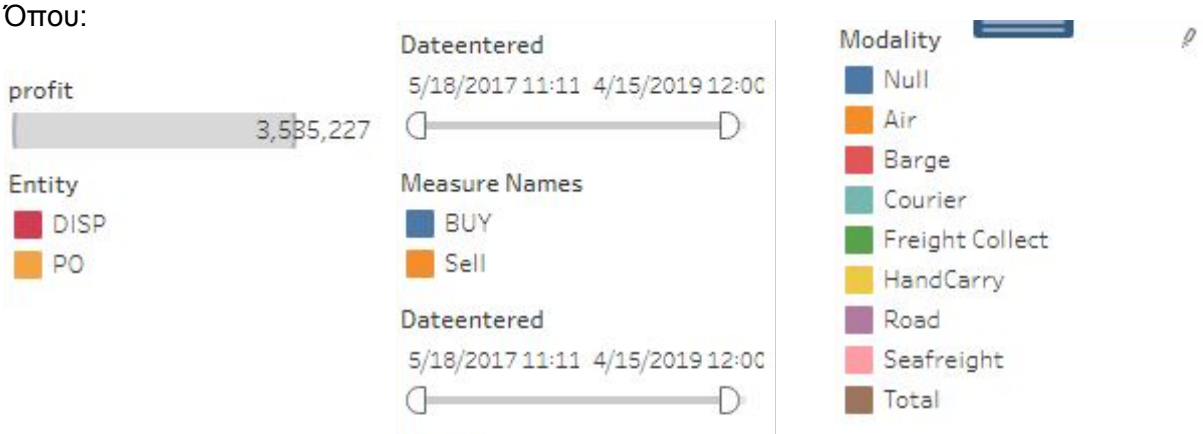
record id	Entity	Vessel	Weight
***01	Order 1	Vessel X	3 Kg
***02	Order 2	Vessel X	5 Kg
***03	Order 3	Vessel Y	27 Kg
***04	Order 4	Vessel Z	18 Kg
***05	Dispatch	-	53 Kg
		Summation	106 Kg

Για να έχουμε μια λίγο πιο πρακτική εικόνα της διαφοράς ανάμεσα στα orders και τα dispatches. Τα παρακάτω report αφορούν το σύνολο των δεδομένων και έχει γίνει φιλτράρισμα μόνο στο εύρος ημερομηνιών που χρησιμοποιήθηκαν στην ανάλυση. Στην ανάλυση έγινε και ένα φιλτράρισμα ως προς τα entries που έχουν τιμολογηθεί.

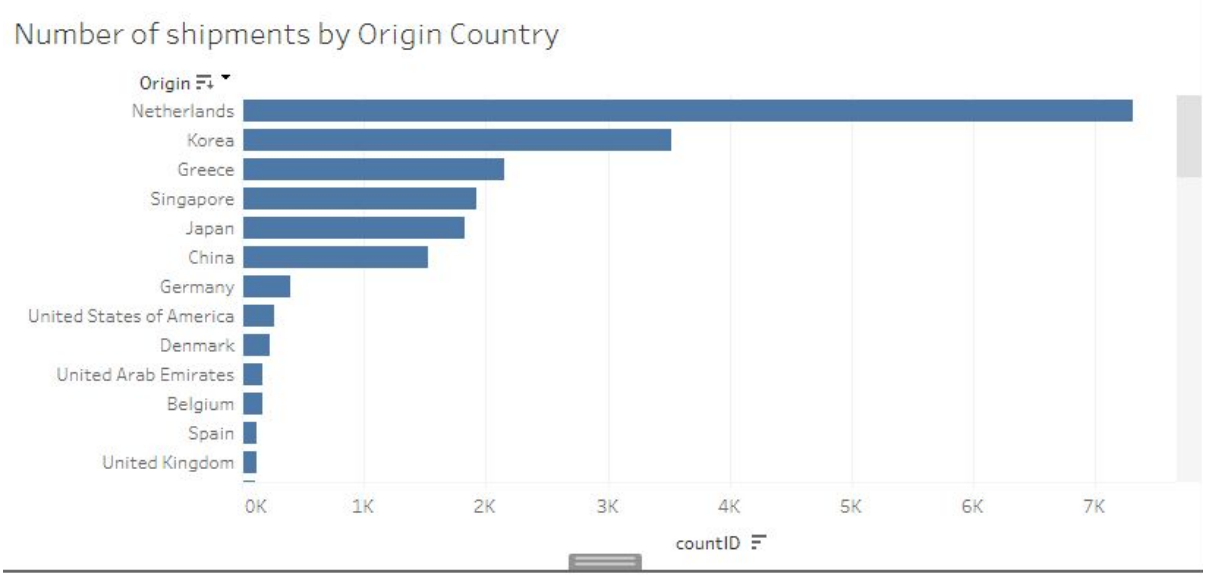


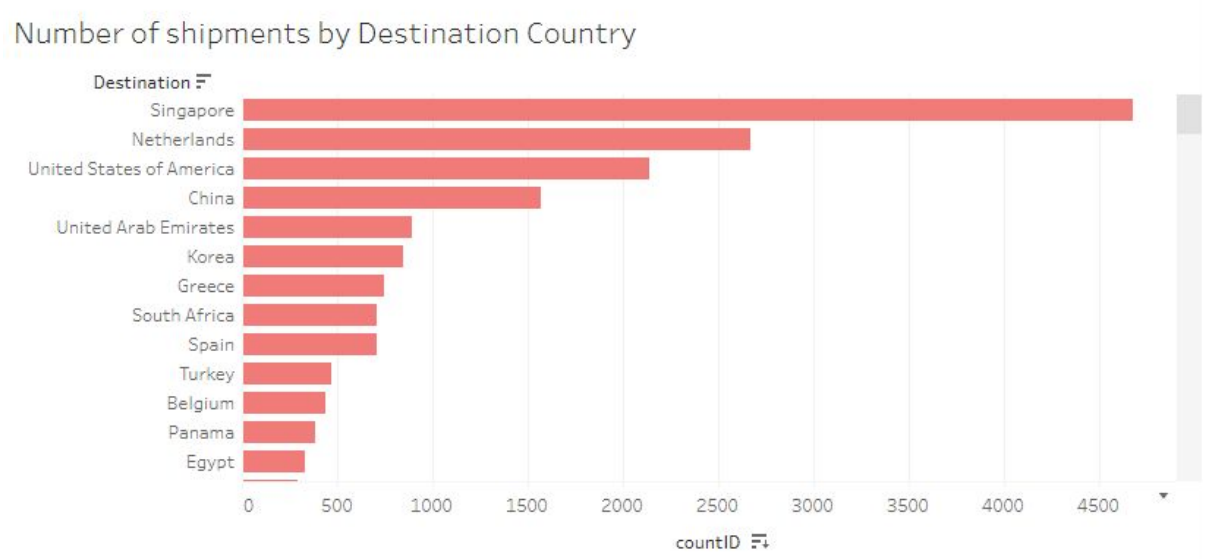


Όπου:

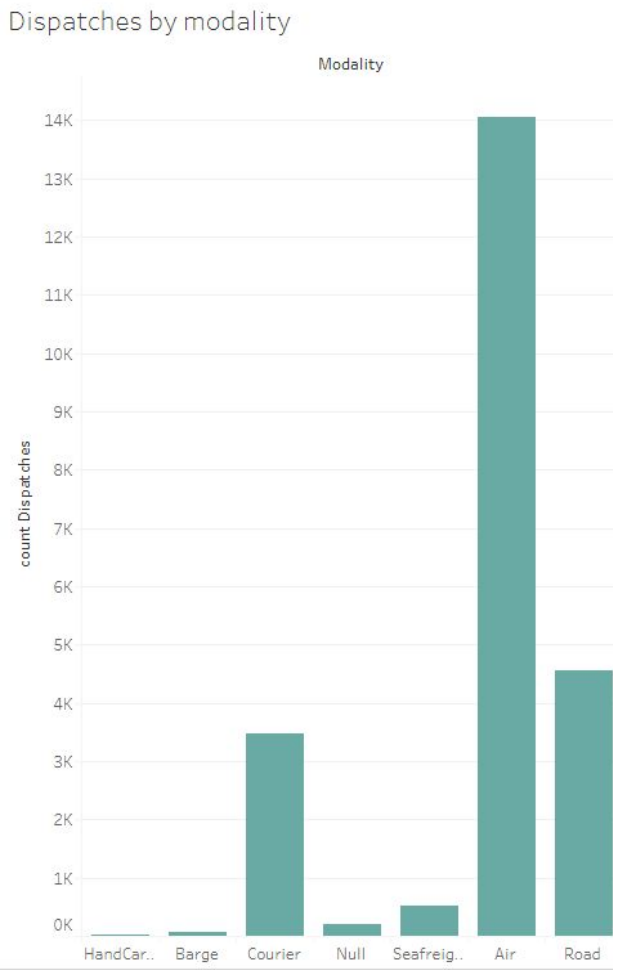


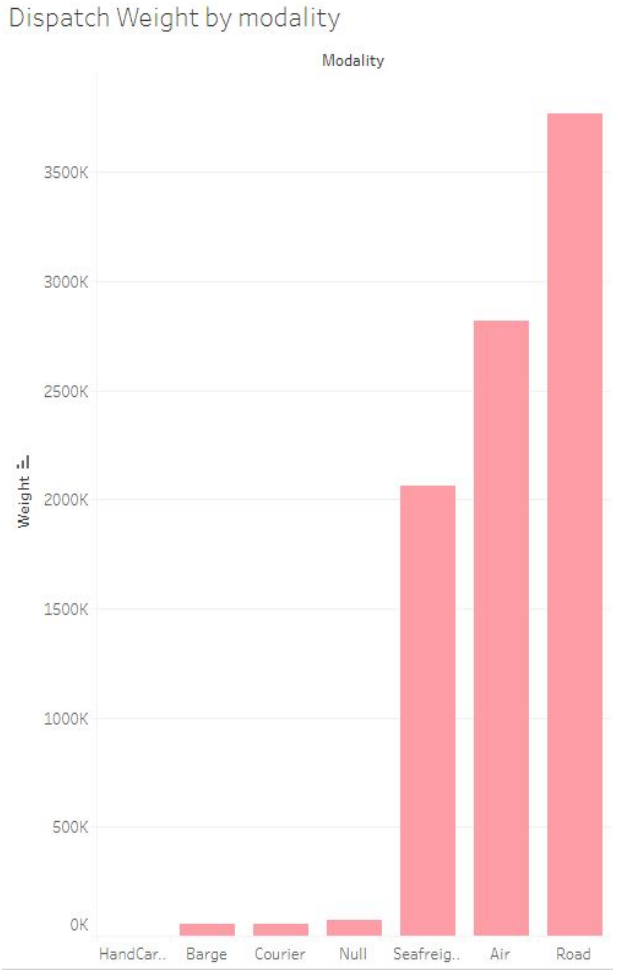
Σχετικά με το αριθμό των Dispatches που μετακινούνται από μια χώρα σε μια άλλη, κάθε χώρα έχει διαφορετικές τελωνειακές διαδικασίες που επηρεάζουν αυτές τις μετακινήσεις. Κάποιες χώρες έχουν συμφωνίες μεταξύ τους για τη μεταφορά αγαθών, επομένως έχει μεγάλη σημασία η αφετηρια αλλά και ο προορισμός ενός shipment. Για μεταφορά ανάμεσα σε ορισμένες χώρες δεν απαιτούνται επιπλέον δασμοί και έγγραφα ενώ σε αυτές που απαιτούνται, το κόστος μεταφοράς αυξάνεται.





Παρακάτω βλέπουμε ότι ποσοτικά τα περισσότερα dispatches μετακινούνται με αεροπορικά μέσα, αλλά το περισσότερο βάρος μεταφέρεται μέσω Road. Όπως είναι λογικό το τελικό κόστος ενός dispatch άρα και το profit έχουν άμεση σχέση και με το βάρος που μεταφέρεται αλλά και με τη μέθοδο μεταφοράς. Διαφορετικές κλίμακες βάρους έχουν διαφορετικά κόστη όπως επίσης και η αεροπορική μεταφορά έχει πολύ μεγαλύτερο από την οδική ή την ακτοπλοϊκή (Seafreight). Από την άλλη μια αεροπορική μεταφορά πέρα από το υψηλό κόστος, έχει και περιορισμούς ώς προς τον όγκο των μεταφερόμενων αντικειμένων αλλά αποτελεί τον ταχύτερο τρόπο μεταφοράς για μακρινές αποστάσεις. Η ακτοπλοϊκή μεταφορά δεν έχει περιορισμούς ως προς τον όγκο και σίγουρα έχει χαμηλότερο κόστος αλλά είναι πολύ χρονοβόρα, ενώ η οδική δεν ενδεικνυται για μεγάλες αποστάσεις.

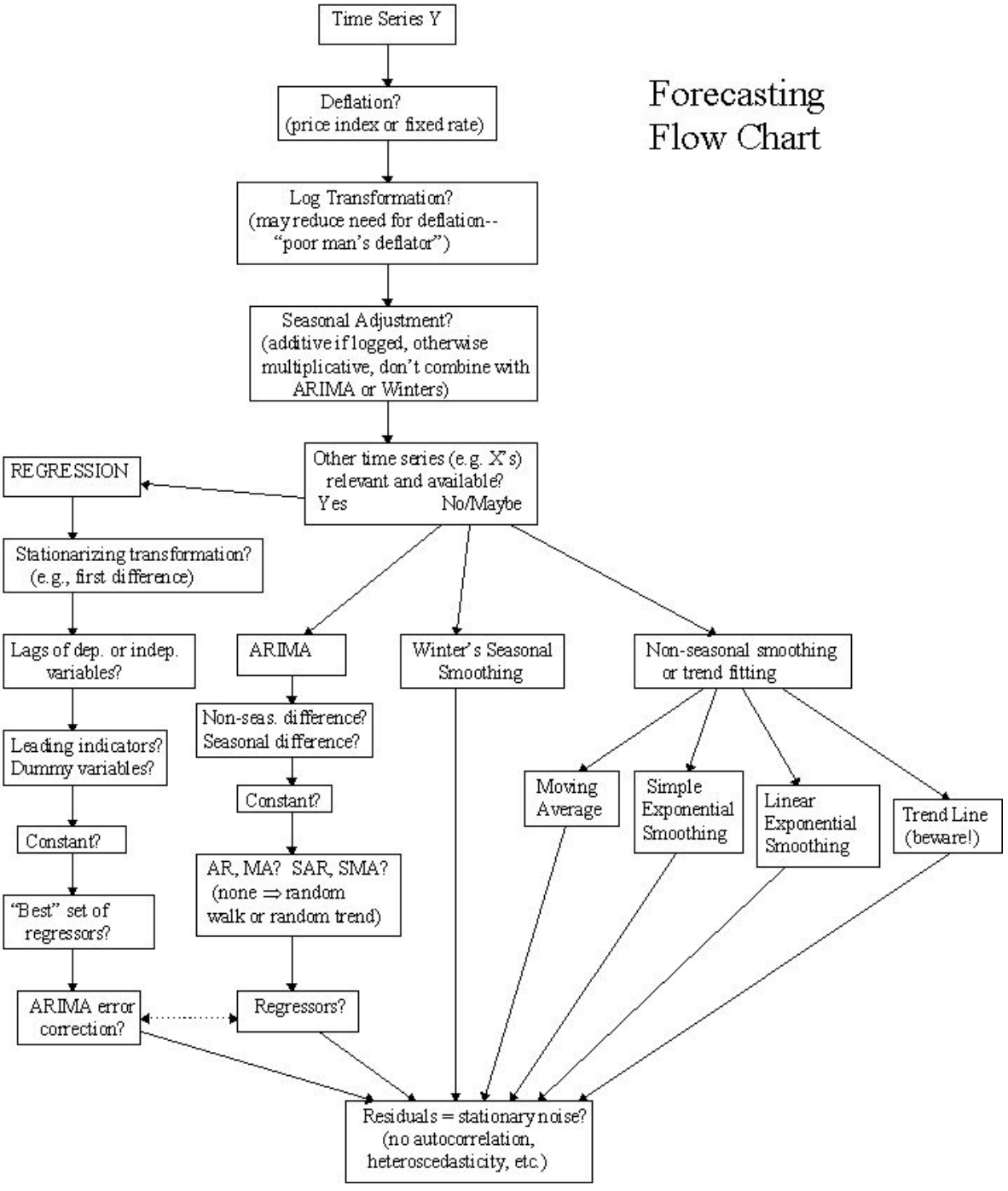




Συνεπώς, η επιλογή μεθόδου μεταφοράς είναι μια πολύπλοκη διαδικασία που επηρεάζει άμεσα το κόστος της μεταφοράς όπως επίσης και το συνολικό βάρος του φορτίου. Το τελικό κόστος μεταφοράς το επηρεάζει και η χώρα από την οποία αναχωρεί το φορτίο αλλά και από τον προορισμό του.

Κεφάλαιο 5 - Εφαρμογή Τεχνολογιών

Μεγάλο κομμάτι της Ανάλυσης είναι η επιλογή και εφαρμογή των σωστών τεχνολογιών. Σημαντικά κριτήρια είναι εποχικότητα το ειδος του μοντέλου τη φυση της ανάλυσης κ.α.



24

[24] "Steps in choosing a forecasting model", *Duke University*

Λόγω της ιδιαιτερότητας των δεδομένων επιλέξαμε να εφαρμόσουμε 2-3 διαφορετικές τεχνικές και κάποιες παραλλαγές τους για να δούμε πως θα τα διαχειριστούμε καλύτερα.

Είναι σημαντικό να τονίσουμε ότι τα δεδομένα είναι πραγματικά και όχι ένα πειραματικό σύνολο και επιλέξαμε να κόψουμε από το dataset περίπου τον τελευταίο ενάμιση μήνα καθώς μεγάλο μέρος του κόστους παραμένει ατιμολόγητο λόγω backlog της παραγωγής, επομένως δεν μπορούμε να έχουμε εικόνα για τα πραγματικά μεγέθη εκεινης της περιόδου.

Εφαρμογή μοντέλων με target το profit

Arima

Επιλέχθηκε το μοντέλο Arima για δεδομένα περίπου 2 ετών στα οποία έγινε άθροισμα του sell και του buy και υπολογίστηκε το profit το οποίο και ορίσαμε σαν target feature. Με τον ίδιο τρόπο μπορεί να γινει η ίδια ανάλυση σε επίπεδο buy ή sell. Ομοίως θα μπορούσαμε να μελετήσουμε το weight φιλτράροντας το dataset για dispatches μονο και με unique ID καθώς όπως είπαμε τα data είναι σε επίπεδο κόστους άρα καθε ID μπορεί να εμφανίζεται πολλαπλες φορες αρά αντίστοιχα και το βάρος του.

Φιλτράρουμε και μετασχηματίζουμε τα δεδομένα, υπολογίζουμε το profit per case. Μετά προσθέτουμε το Buy, Sell και Profit ανά ημέρα ταξινομώντας τα με βάση την ημερομηνία.

```
df['DATEENTERED'] = pd.to_datetime(df['DATEENTERED'],infer_datetime_format=True)
df['DATEENTERED'] =df['DATEENTERED'].dt.date

df2 = df.filter(items=['DATEENTERED', 'SELL', 'BUY']).sort_values(by=['DATEENTERED']).round({'SELL': 2}).round({'BUY': 2})
df2['PROFIT'] = df2['SELL'] - df2['BUY']

df2['SELL'].fillna(0, inplace=True)
df2['BUY'].fillna(0, inplace=True)
df3 = df2.dropna()

df3 = df3[(df3['DATEENTERED']<datetime.date(2019,4,15))]
```

```
df3.tail(6)
```

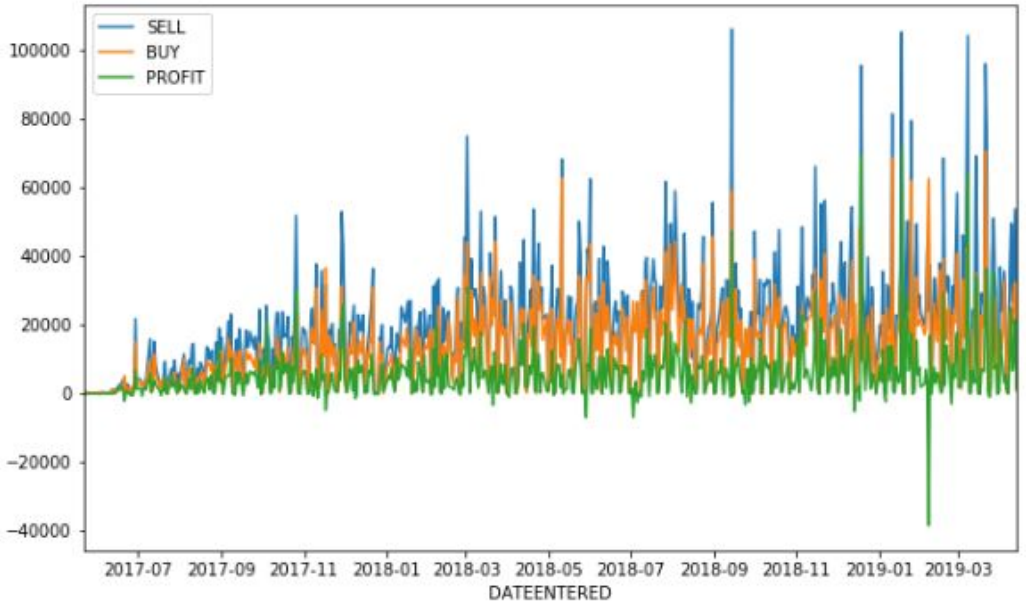
	DATEENTERED	SELL	BUY	PROFIT
204264	2019-04-13	110.89	66.53	44.36
24473	2019-04-13	320.00	229.90	90.10
24472	2019-04-13	50.00	0.00	50.00
204265	2019-04-13	15.97	0.00	15.97
216010	2019-04-13	120.00	0.00	120.00
216012	2019-04-13	0.00	111.33	-111.33

```
dftotals=pd.DataFrame(df3, columns=['DATEENTERED', 'SELL','BUY','PROFIT']).groupby('DATEENTERED').sum().sort_values(by=['DATEENT

dftotals.tail(15)
```

	DATEENTERED	SELL	BUY	PROFIT
611	2019-03-30	0.00	129.34	-129.34
612	2019-03-31	625.00	423.34	201.66
613	2019-04-01	36814.48	18353.65	18460.83
614	2019-04-02	24017.37	12821.37	11196.00

Μέσα από το plot παρατηρούμε το γενικό trend, δηλαδή εάν υπάρχει κάποια τάση. Τα δεδομένα παρόλου που δεν φαίνεται να παρουσιάζουν κάποιο μοτίβο, έχουν μια γενικά ανοδική τάση.

```
dftotals.plot(x="DATEENTERED", y=["SELL", "BUY", "PROFIT"])
plt.show()


dfprofit=pd.DataFrame(dftotals, columns=['DATEENTERED', 'PROFIT'])
dfprofit.reset_index(inplace=True)
dfprofit['DATEENTERED'] = pd.to_datetime(dfprofit['DATEENTERED'])
dfprofit = dfprofit.set_index('DATEENTERED')
```

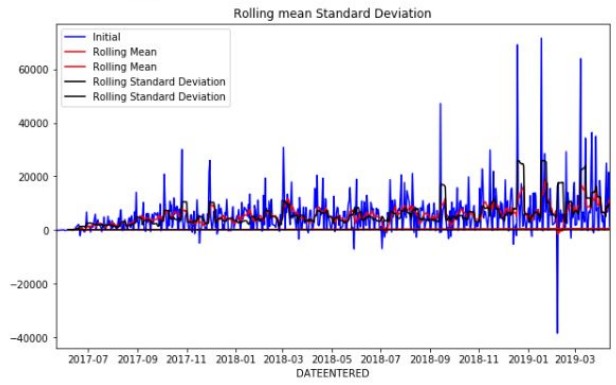


Για την παρακάτω ανάλυση χρησιμοποιείται το πεδίο dateentered και το profit για τα οποία υπολογίζονται η κινούμενη (rollling) τυπική απόκλιση και ο κινούμενος (rolling) μέσος με window=7

```
rolmean = dfprofit.rolling(window=7).mean()
rolstd = dfprofit.rolling(window=7).std()

print (rolmean, rolstd)
```

DATEENTERED	index	PROFIT
2017-05-22	NaN	NaN
2017-05-23	NaN	NaN
2017-05-29	NaN	NaN
2017-05-30	NaN	NaN
2017-06-01	NaN	NaN
2017-06-02	NaN	NaN
2017-06-06	3.0	-14.211429
2017-06-07	4.0	-6.942857
2017-06-08	5.0	12.960000
2017-06-09	6.0	18.441429
2017-06-12	7.0	-7.431429
2017-06-13	8.0	26.752857

```
dftotals.plot(x="DATEENTERED", y="PROFIT" , color='blue', label='Initial')
plt.plot(rolmean, color='red', label='Rolling Mean')
plt.plot(rolstd, color='black', label='Rolling Standard Deviation')
plt.legend(loc='best')
plt.title('Rolling mean Standard Deviation')
plt.show(block=False)
```



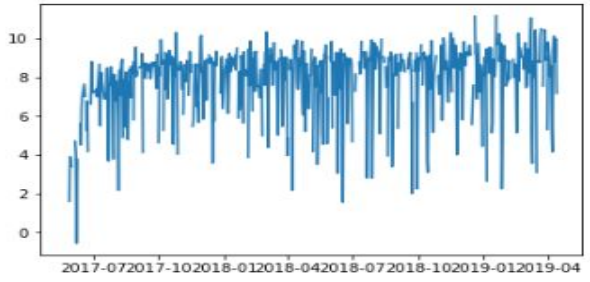
Στη συνέχεια πραγματοποιείται ένα adfuller test για να προσδιοριστεί εάν τα δεδομένα είναι στατικά, από το οποίο προέκυψε τιμή p μεγαλύτερη του 0,05 επομένως τα δεδομένα δεν είναι στατικά.

```
from statsmodels.tsa.stattools import adfuller

print ('Results of Dickey-Fuller Test:')
dftest = adfuller(dftotals['PROFIT'], autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statisti
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)
```

```
Results of Dickey-Fuller Test:
Test Statistic            -5.705598e+00
p-value                    7.501155e-07
Lags Used                  7.000000e+00
Nr of Observations Used    6.180000e+02
Critical Value (1%)       -3.440976e+00
Critical Value (5%)       -2.866228e+00
Critical Value (10%)      -2.569267e+00
dtype: float64
```

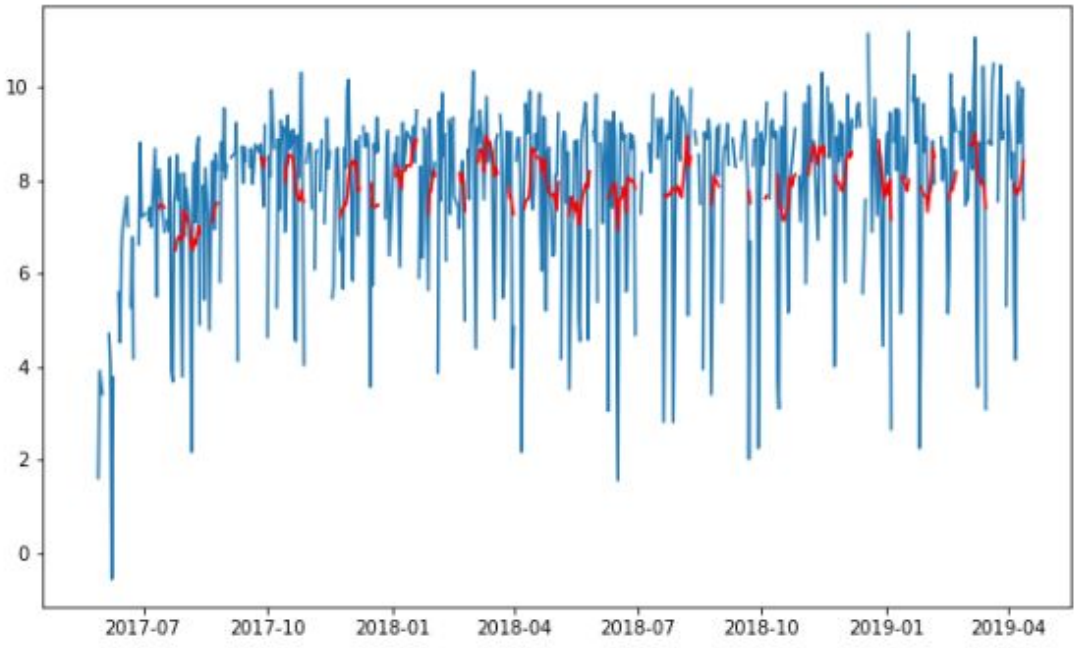
Εφαρμόζουμε το παρακάτω για να απορρίψουμε τις μη έγκυρες τιμές

```
dfprofit= dftotals.filter(items=['DATEENTERED', 'PROFIT']).set_index('DATEENTERED')


dfprofit.dropna(inplace=True)

with np.errstate(invalid='ignore'):
    dfprofit_logScale = np.log(dfprofit)

plt.plot(dfprofit_logScale)
```



Ομοίως υπολογίζεται ο κινούμενος μέσος και η κινούμενη τυπική απόκλιση πάλι με window = 7:

```
movingAverage = dfprofit_logScale.rolling(window=7).mean()
movingSTD = dfprofit_logScale.rolling(window=7).std()
plt.plot(dfprofit_logScale)
plt.plot(movingAverage, color='red')
```



και αφαιρούνται οι τιμές προκειμένου να γίνει η χρονοσειρά στατική:

```
dfprofit_logScale_Minus_MAverage = dfprofit_logScale - movingAverage

dfprofit_logScale_Minus_MAverage.dropna(inplace=True)
dfprofit_logScale_Minus_MAverage.head(10)
```

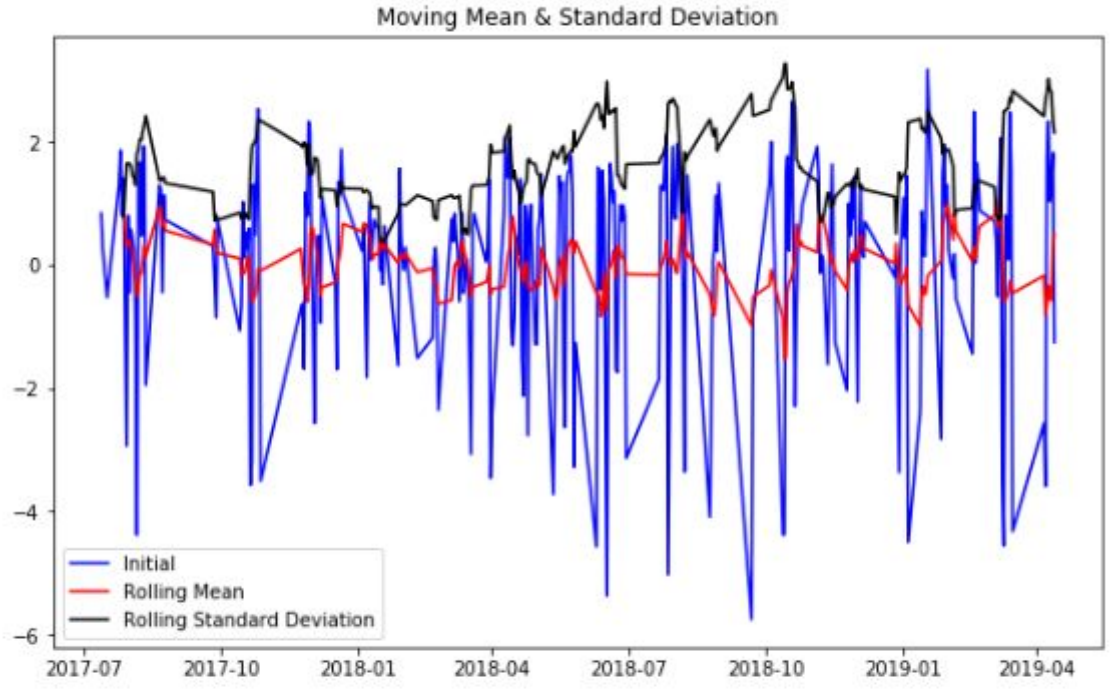
DATEENTERED	PROFIT
2017-07-13	0.831358
2017-07-14	0.266885
2017-07-17	-0.529295
2017-07-25	1.297113
2017-07-26	1.850394
2017-07-27	0.791491

Ομοίως φτιάχνουμε μια συνάρτηση που εφαρμόζει το adfuller test.

```
from statsmodels.tsa.stattools import adfuller
def test_stationarity(timeseries):
    movingAverage = timeseries.rolling(window=7).mean()
    movingSTD = timeseries.rolling(window=7).std()

    orig = plt.plot(timeseries, color='blue', label='Initial')
    mean = plt.plot(movingAverage, color='red', label='Rolling Mean')
    std = plt.plot(movingSTD, color='black', label='Rolling Standard Devia
    plt.legend(loc='best')
    plt.title('Moving Mean & Standard Deviation')
    plt.show(block=False)

    print ('Results for Dickey-Fuller Test:')
    dftest = adfuller(timeseries['PROFIT'], autolag='AIC')
    dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','
    for key,value in dftest[4].items():
        dfoutput['Critical Value (%s)'%key] = value
    print(dfoutput)
```

Σε αυτό το σημείο, τρέχοντας την, παρατηρούμε ότι έχουμε καλύτερο p value αυτή τη φορά αλλά και πάλι μεγαλύτερο του 0,05 η σειρά δεν είναι στατική.

```
test_stationarity(dfprofit_logScale_Minus_MAverage)
```

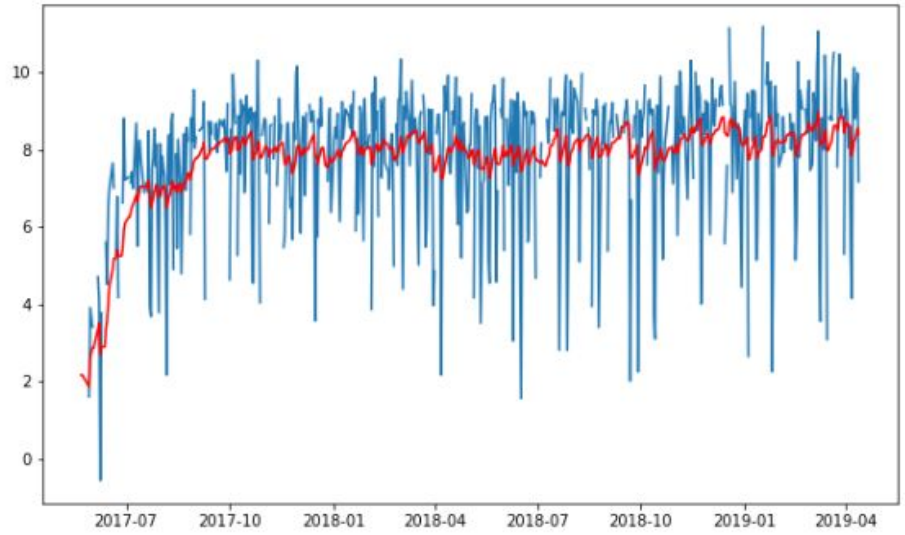



```
Results for Dickey-Fuller Test:
Test Statistic            -1.248159e+01
p-value                    3.082920e-23
Lags Used                  4.000000e+00
Nr of Observations Used    2.780000e+02
Critical Value (1%)       -3.454094e+00
Critical Value (5%)       -2.871993e+00
Critical Value (10%)      -2.572340e+00
dtype: float64
```

Στη συνέχεια υπολογίζεται το exponential decay του weighted moving average

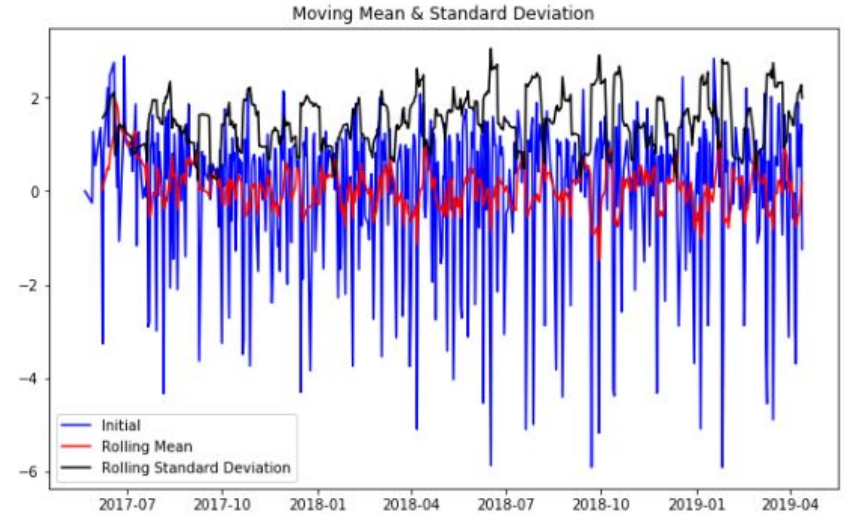
```
exponential_Decay_Weighted_Average = dfprofit_logScale.ewm(halflife=7, min_periods=0, adjust=True).mea
plt.plot(dfprofit_logScale)
plt.plot(exponential_Decay_Weighted_Average, color='red')
```

```
[<matplotlib.lines.Line2D at 0x1e1d46da128>]
```



και αφού το αφαιρέσουμε από το αρχικό, ξανα-ελέγχουμε τη στατικότητα

```
dfprofit_logScale_Minus_Moving_Exp_Decay_Avg = dfprofit_logScale - exponential_Decay_Weighted_Average

dfprofit_logScale_Minus_Moving_Exp_Decay_Avg.dropna(inplace=True)

test_stationarity(dfprofit_logScale_Minus_Moving_Exp_Decay_Avg)
```

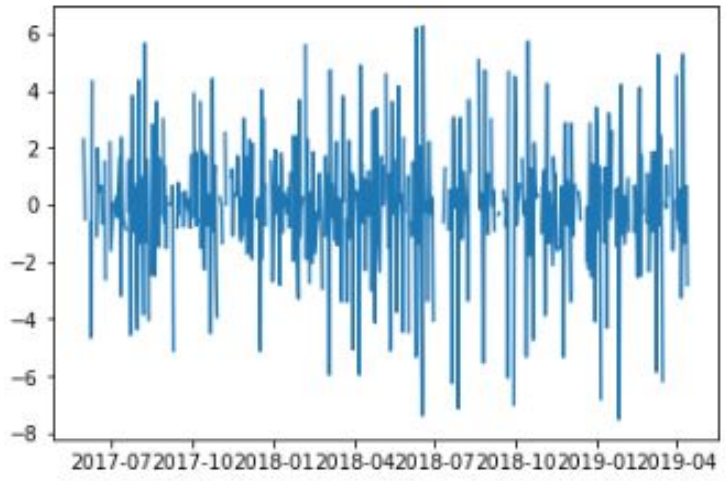



```
Results for Dickey-Fuller Test:
Test Statistic            -8.265586e+00
p-value                    4.948638e-13
Lags Used                  9.000000e+00
Nr of Observations Used    5.510000e+02
Critical Value (1%)       -3.442274e+00
Critical Value (5%)       -2.866800e+00
Critical Value (10%)      -2.569571e+00
dtype: float64
```

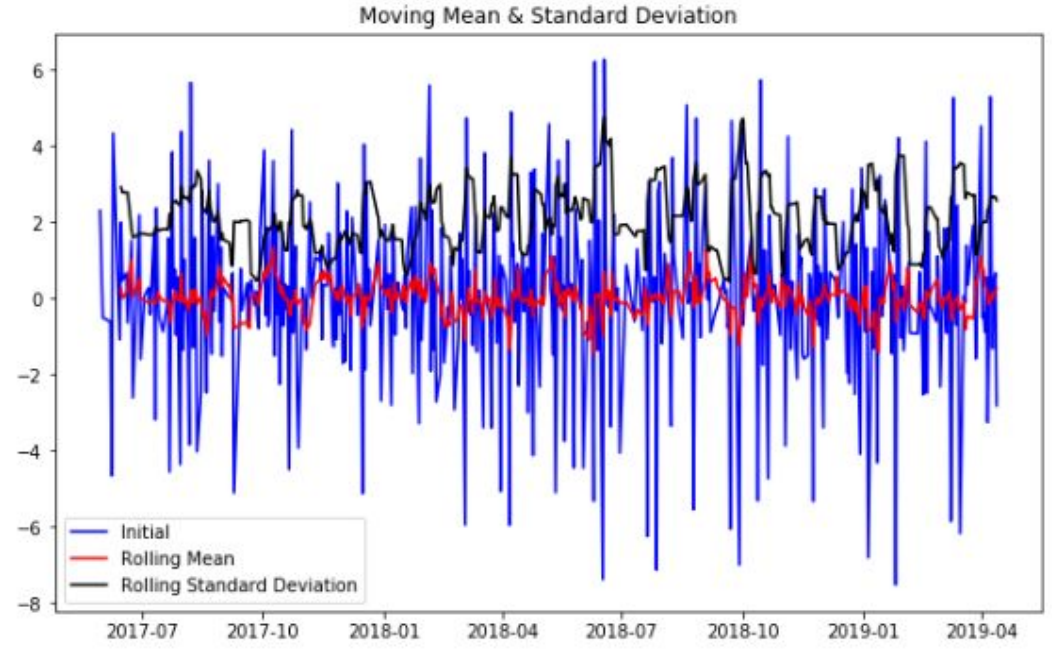
Σε αυτή τη περίπτωση το p είναι ακόμα μεγαλύτερο.
Ομοίως με τα shifted values:

```
df_log_diff_shifting = dfprofit_logScale - dfprofit_logScale.shift()
plt.plot(df_log_diff_shifting)
```

```
[<matplotlib.lines.Line2D at 0x1e4689ff2b0>]
```




```
df_log_diff_shifting.dropna(inplace=True)
test_stationarity(df_log_diff_shifting)
```

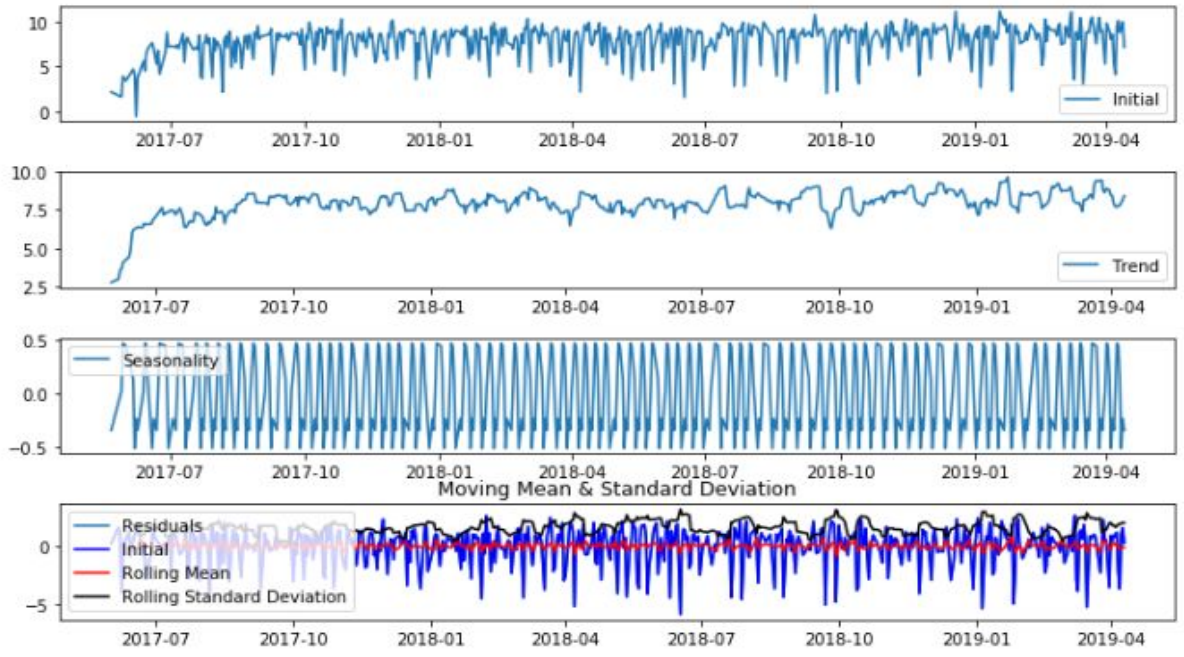



```
Results for Dickey-Fuller Test:
Test Statistic            -1.049482e+01
p-value                    1.122590e-18
Lags Used                  6.000000e+00
Nr of Observations Used    4.940000e+02
Critical Value (1%)       -3.443657e+00
Critical Value (5%)       -2.867408e+00
Critical Value (10%)      -2.569896e+00
dtype: float64
```

Στο επόμενο βήμα ελέγχουμε το seasonal decomposition:

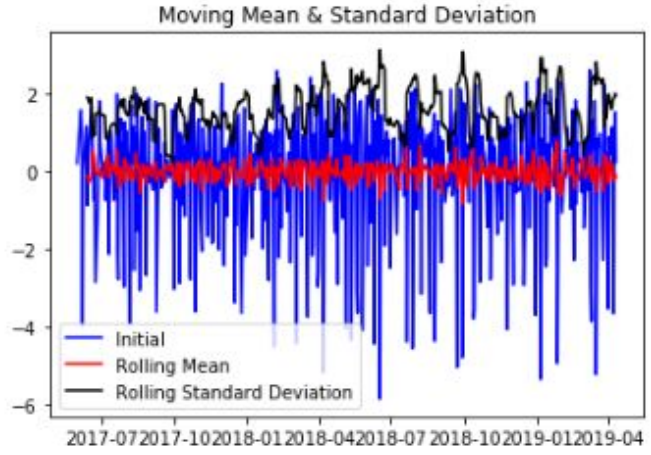
```
from statsmodels.tsa.seasonal import seasonal_decompose

dfprofit_logScale.dropna(inplace=True)
dfprofit_logScale.index.inferred_freq = 'd'

decomposition = seasonal_decompose(dfprofit_logScale)

trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

plt.subplot(411)
plt.plot(dfprofit_logScale, label='Initial')
plt.legend(loc='best')
plt.subplot(412)
plt.plot(trend, label='Trend')
plt.legend(loc='best')
plt.subplot(413)
plt.plot(seasonal, label='Seasonality')
plt.legend(loc='best')
plt.subplot(414)
plt.plot(residual, label='Residuals')
plt.legend(loc='best')
plt.tight_layout()

decomposedLogData = residual
decomposedLogData.dropna(inplace=True)
test_stationarity(decomposedLogData)
```

```
decomposedLogData = residual
decomposedLogData.dropna(inplace=True)
test_stationarity(decomposedLogData)
```



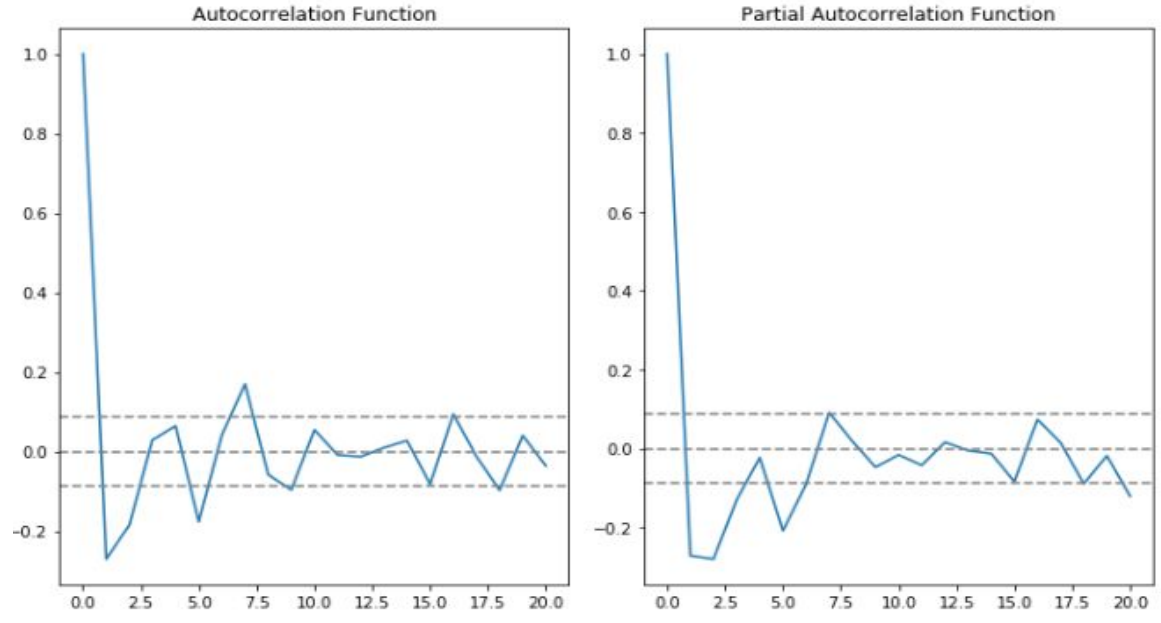

```
Results for Dickey-Fuller Test:
Test Statistic            -1.066523e+01
p-value                    4.304383e-19
Lags Used                  1.900000e+01
Nr of Observations Used    5.350000e+02
Critical Value (1%)       -3.442632e+00
Critical Value (5%)       -2.866957e+00
Critical Value (10%)      -2.569655e+00
dtype: float64
```

```
test_stationarity(decomposedLogData)
```



Υπολογίζουμε autocorrelation function (ACF) και partial autocorrelation function (pacf)

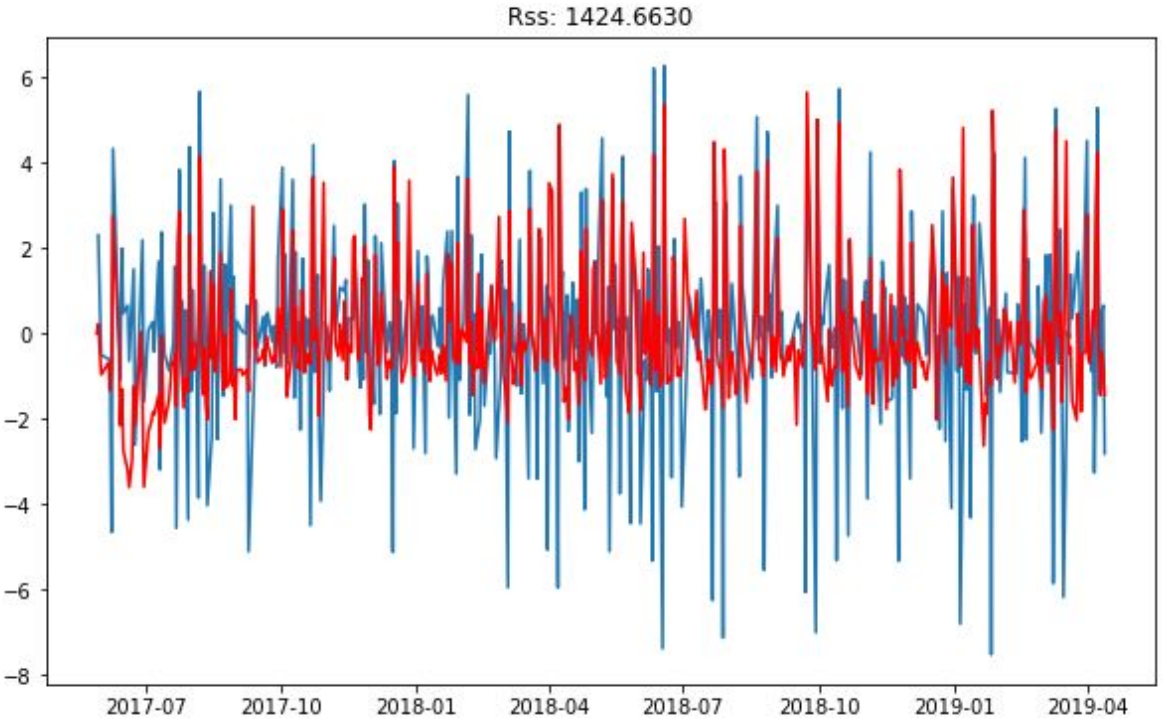
```
from statsmodels.tsa.stattools import acf , pacf

log_acf = acf(df_log_diff_shifting, nlags=20)
log_pacf = pacf(df_log_diff_shifting, nlags=20, method='ols')

#acf plots
plt.subplot(121)
plt.plot(log_acf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(df_log_diff_shifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(df_log_diff_shifting)), linestyle='--', color='gray')
plt.title('Autocorrelation Function')

#pcaf plot
plt.subplot(122)
plt.plot(log_pacf)
plt.axhline(y=0, linestyle='--', color='gray')
plt.axhline(y=-1.96/np.sqrt(len(df_log_diff_shifting)), linestyle='--', color='gray')
plt.axhline(y=1.96/np.sqrt(len(df_log_diff_shifting)), linestyle='--', color='gray')
plt.title('Partial Autocorrelation Function')
plt.tight_layout()
```



Με βάση τα παραπάνω υπολογίζουμε τις σειρές AR και MA

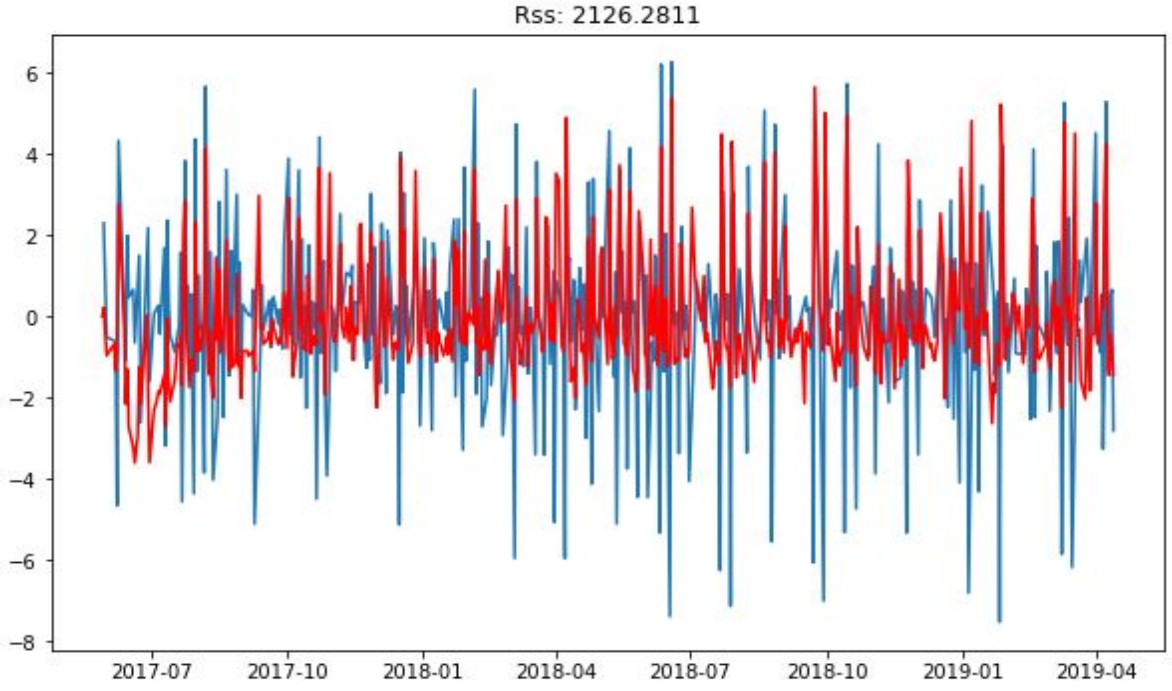
```
from statsmodels.tsa.arima_model import ARIMA

#AR
model = ARIMA(dfprofit_logScale, order=(1, 1, 3))
results_AR = model.fit(disp=-1)
plt.plot(df_log_diff_shifting)
plt.plot(results_AR.fittedvalues, color='red' )
a=(results_AR.fittedvalues-df_log_diff_shifting["PROFIT"])
a.dropna(inplace=True)
plt.title('Rss: %.4f'% sum((a)**2))
print ('Plot AR Model')
```

```
Plot AR Model
```

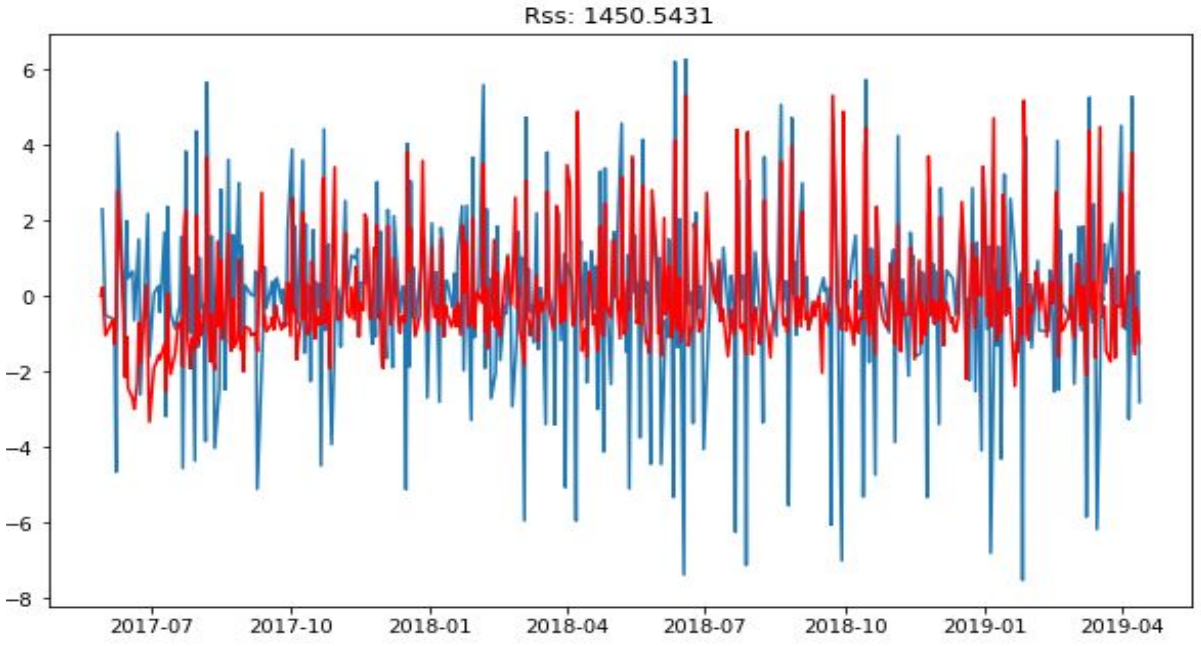


```
#MA
model = ARIMA(dfprofit_logScale, order=(1, 1, 0))
results_MA = model.fit(disp=-1)
plt.plot(df_log_diff_shifting)
plt.plot(results_AR.fittedvalues, color='red' )
b=(results_MA.fittedvalues-df_log_diff_shifting["PROFIT"])
b.dropna(inplace=True)
plt.title('Rss: %.4f'% sum((b)**2))
print ('Plot MA Model')
```

```
Plot MA Model
```



```
#ARMA
model = ARIMA(dfprofit_logScale, order=(1, 1, 2))
results_ARIMA = model.fit(disp=-1)
plt.plot(df_log_diff_shifting)
plt.plot(results_ARIMA.fittedvalues, color='red' )
c=(results_ARIMA.fittedvalues-df_log_diff_shifting["PROFIT"])
c.dropna(inplace=True)
plt.title('Rss: %.4f'% sum((c)**2))
```




```
predictions_ARIMA_diff = pd.Series(results_ARIMA.fittedvalues, copy=True)
print (predictions_ARIMA_diff.head())
```

```
DATEENTERED
2017-05-29     0.005022
2017-05-30     0.232060
2017-06-01    -1.040982
2017-06-06    -0.618316
2017-06-07    -1.261312
dtype: float64
```

```
#convert to cumulative sum
predictions_ARIMA_diff_cumsum = predictions_ARIMA_diff.cumsum()
print (predictions_ARIMA_diff_cumsum.head())
```

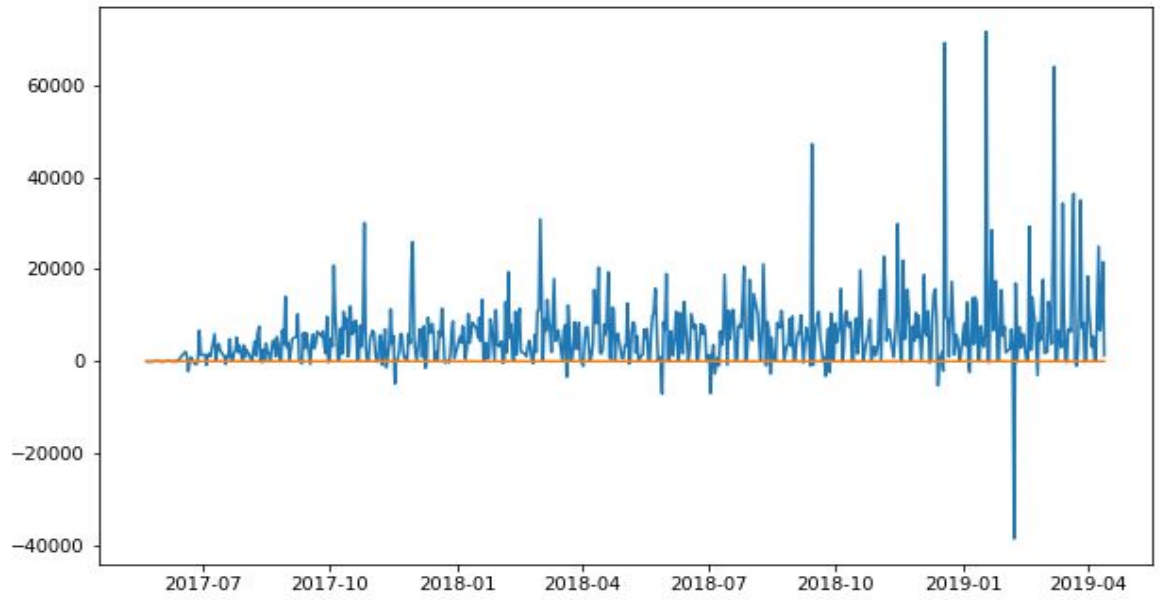
```
DATEENTERED
2017-05-29     0.005022
2017-05-30     0.237082
2017-06-01    -0.803900
2017-06-06    -1.422216
2017-06-07    -2.683528
dtype: float64
```

```
predictions_ARIMA_log = pd.Series(dfprofit_logScale['PROFIT'].iloc[0], index=dfprofit_logScale.index)
predictions_ARIMA_log = predictions_ARIMA_log.add(predictions_ARIMA_diff_cumsum, fill_value=0)
predictions_ARIMA_log.head()
```

```
DATEENTERED
2017-05-22    2.169054
2017-05-29    2.174076
2017-05-30    2.406136
2017-06-01    1.365154
2017-06-06    0.746838
dtype: float64
```

```
predictions_ARIMA = np.exp(predictions_ARIMA_log)
plt.plot(dfprofit)
plt.plot(predictions_ARIMA)
```

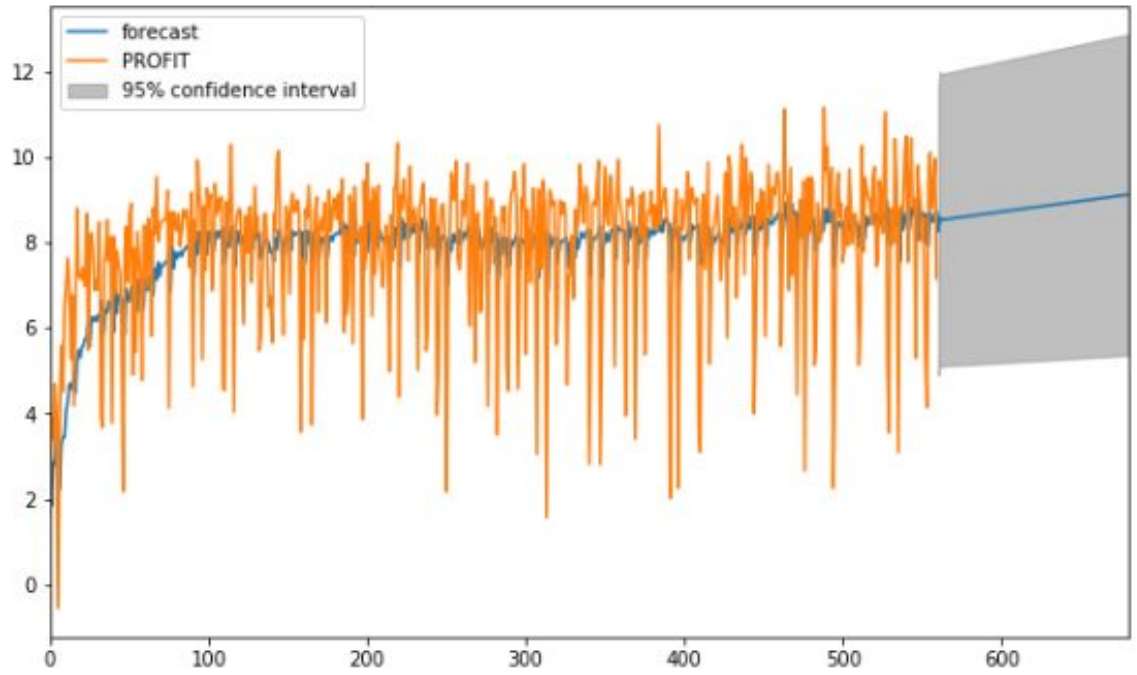
```
[<matplotlib.lines.Line2D at 0x1e457823668>]
```




```
dfprofit_logScale
```

DATEENTERED	PROFIT
2017-05-22	2.169054
2017-05-29	1.609438
2017-05-30	3.912023
2017-06-01	3.410157
2017-06-06	4.702751
2017-06-07	4.088159
2017-06-08	-0.562119
2017-06-09	3.769768
2017-06-13	5.596791
2017-06-14	4.519721
2017-06-15	6.512443
2017-06-16	6.989741

```
#data from 2017-05-22 to 2019-04-13 (2 years)

results_ARIMA.plot_predict(1,681)
```




```
results_ARIMA.forecast(steps=120)
```

```
(array([8.27587122, 8.60857606, 8.52585083, 8.5543704 , 8.5531006 ,
        8.55980786, 8.564379  , 8.56952216, 8.57451214, 8.57954314,
        8.58456315, 8.58958611, 8.59460828, 8.59963066, 8.60465299,
        8.60967532, 8.61469766, 8.61972   , 8.62474233, 8.62976467,
        8.634787  , 8.63980934, 8.64483167, 8.64985401, 8.65487635,
```

Παρατηρούμε ότι η παραπάνω πρόβλεψη παρουσιάζει μια ανοδική τάση αλλά το confidence interval είναι πολύ μεγάλο.

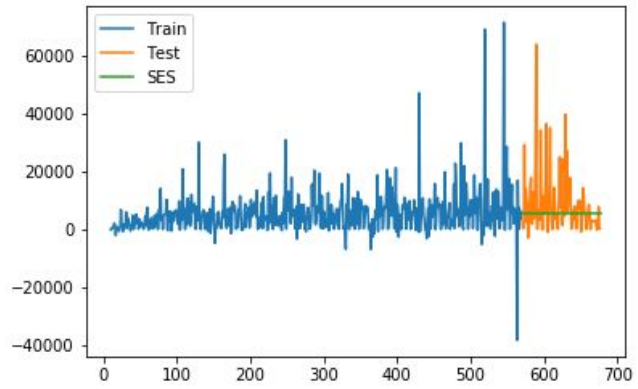
Exponential Smoothing

Συνεχίζοντας με χρονοσειρές η επεξεργασία των δεδομένων είναι ίδια, και σε αυτή τη περίπτωση τρέξαμε την ανάλυση ως προς το profit ομοίως θα μπορούσαμε να μελετήσουμε το sell ή το buy αλλά πάλι επιλέγουμε να συγκρίνουμε όμοια μεγέθη..

```
series= dfselling.iloc[:,3:4]
series2= dfselling.iloc[:-90,3:4]
Y_train = series['PROFIT'].values

print(len(series), len(series2))
```

```
667 577
```

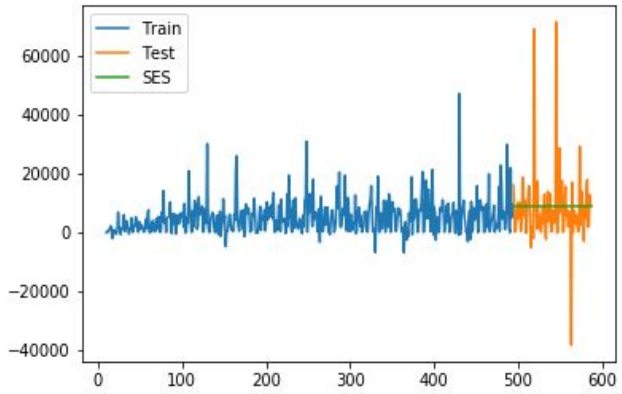
```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import SimpleExpSmoothing, Holt, ExponentialSmoothing

train_size = int(len(series) * 0.84)

train=series[0:train_size]
test=series[train_size:]

y_hat_avg = test.copy()
fit1 = SimpleExpSmoothing(np.asarray(train['PROFIT'])).fit(smoothing_level=0.6, optimized=False)
y_hat_avg['SES'] = fit1.forecast(len(test))
plt.plot(train['PROFIT'], label = 'Train')
plt.plot(test['PROFIT'], label = 'Test')
plt.plot(y_hat_avg['SES'], label = 'SES')
plt.legend(loc='best')
plt.show()
```




```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(test.PROFIT, y_hat_avg.SES))
print(rms)
```

```
10544.5521397427
```

```
train_size = int(len(series2) * 0.84)

train=series2[0:train_size]
test=series2[train_size:]

y_hat_avg = test.copy()
fit1 = SimpleExpSmoothing(np.asarray(train['PROFIT'])).fit(smoothing_level=0.6, optimized=False)
y_hat_avg['SES'] = fit1.forecast(len(test))
plt.plot(train['PROFIT'], label = 'Train')
plt.plot(test['PROFIT'], label = 'Test')
plt.plot(y_hat_avg['SES'], label = 'SES')
plt.legend(loc='best')
plt.show()
```




```
rms = sqrt(mean_squared_error(test.PROFIT, y_hat_avg.SES))
print(rms)
```

```
12323.671065558747
```

Χρησιμοποιώ Simple Moving Average με διαφορετικά μεγέθη παραθύρων (3, 5, 7 και 10).

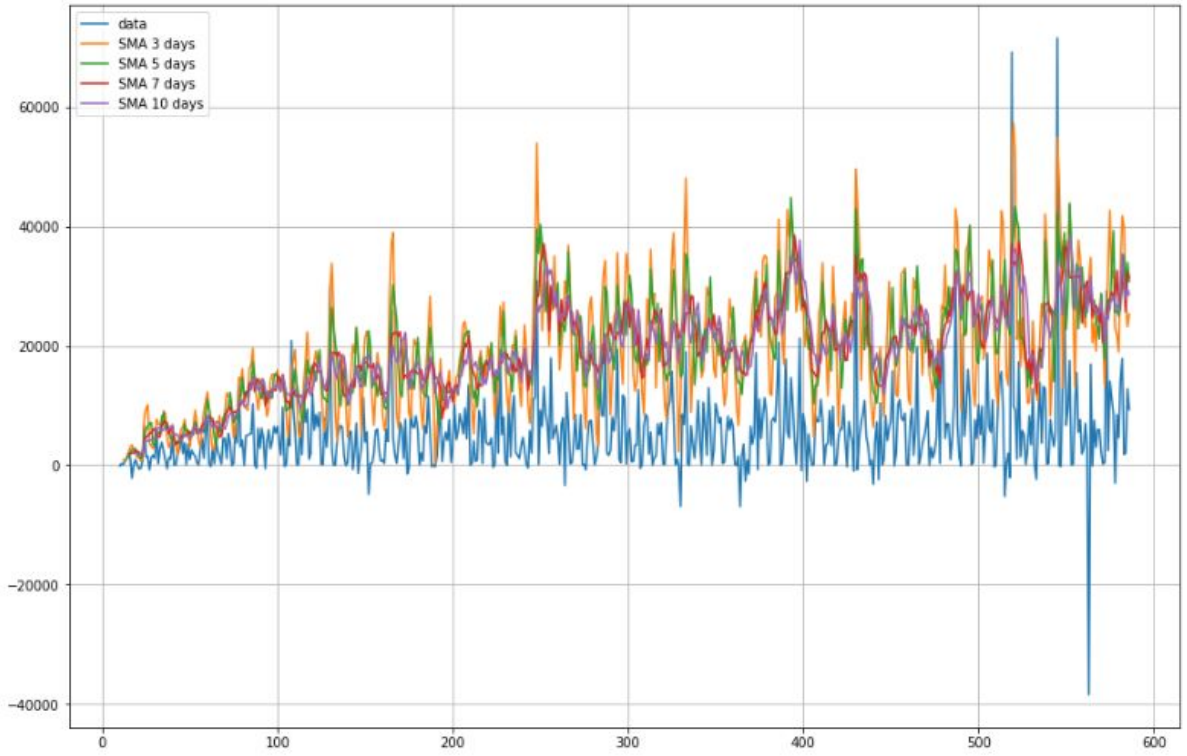
```
dfselling = dfselling.iloc[ :-90,:]

dfselling['SMA_3'] = dfselling.iloc[:,1].rolling(window=3).mean()

dfselling['SMA_5'] = dfselling.iloc[:,1].rolling(window=5).mean()

dfselling['SMA_7'] = dfselling.iloc[:,1].rolling(window=7).mean()

dfselling['SMA_10'] = dfselling.iloc[:,1].rolling(window=10).mean()

dfselling.head(5)
```

	DATEENTERED	SELL	BUY	PROFIT	SMA_3	SMA_5	SMA_7	SMA_10
10	2017-06-12	1099.03	1230.14000	-131.11000	NaN	NaN	NaN	NaN
11	2017-06-13	1200.00	930.44000	269.56000	NaN	NaN	NaN	NaN
12	2017-06-14	257.50	185.91492	71.58508	852.176667	NaN	NaN	NaN
13	2017-06-15	1184.50	511.03000	673.47000	880.666667	NaN	NaN	NaN
14	2017-06-16	1887.44	802.00000	1085.44000	1109.813333	1125.694	NaN	NaN

Και αντιστοιχα προβάλλουμε τα plot στα οποία παρουσιάζεται μια εξομάλυνση όσο μεγαλώνει το μέγεθος παραθύρου..

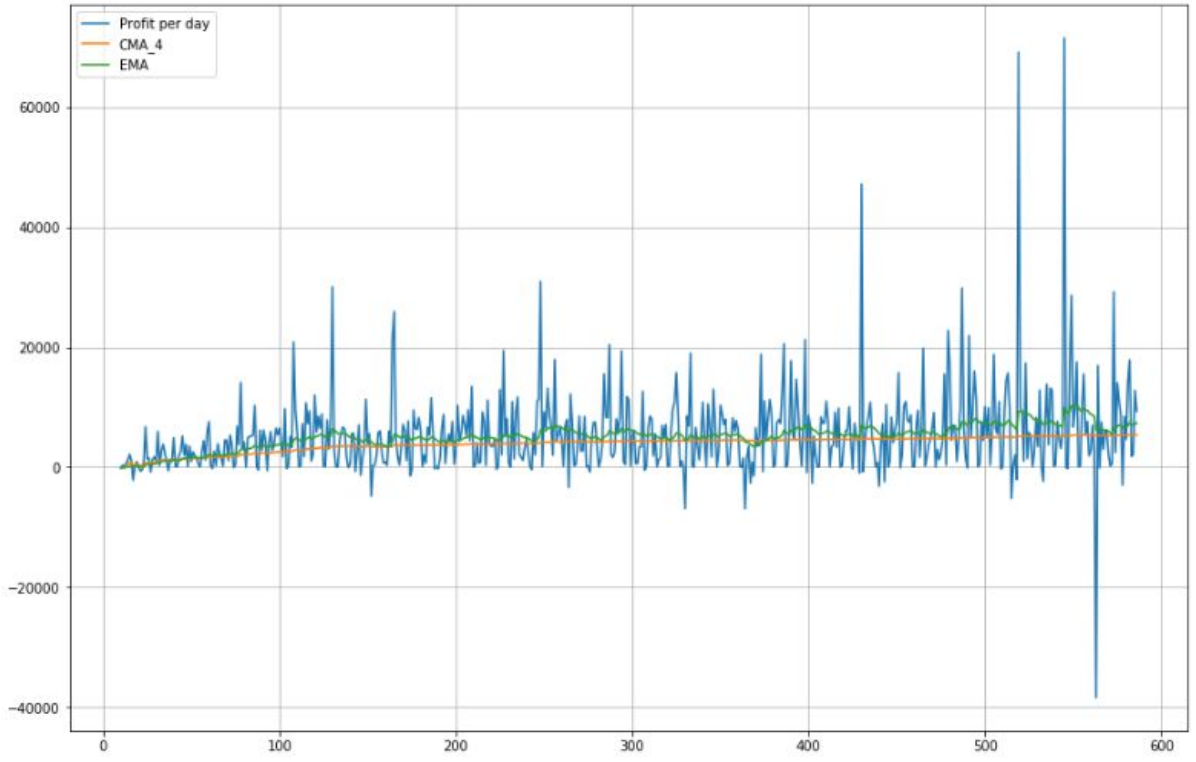
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=[15,10])
plt.grid(True)
plt.plot(dfselling['PROFIT'],label='data')
plt.plot(dfselling['SMA_3'],label='SMA 3 days')
plt.plot(dfselling['SMA_5'],label='SMA 5 days')
plt.plot(dfselling['SMA_7'],label='SMA 7 days')
plt.plot(dfselling['SMA_10'],label='SMA 10 days')
plt.legend(loc=2)
```

```
<matplotlib.legend.Legend at 0x1b330ad6828>
```



Υπολογίσαμε το Simple moving average (SMA), και στη συνεχεια υπολογίζουμε το Cumulative moving average (CMA) με παράθυρο 4 και το exponential moving average (EMA). Παρατηρούμε ότι παράγεται αρκετά μεγάλο rms.

```
#Cumulative Moving Average

df_selling_cma=pd.DataFrame(dfselling.iloc[:,3])
df_selling_cma['CMA_4'] = df_selling_cma.expanding(min_periods=4).mean()

# Exponential Moving Average
df_selling_cma['EMA'] = df_selling_cma.iloc[:,0].ewm(span=40,adjust=False).mean()

plt.figure(figsize=[15,10])
plt.grid(True)
plt.plot(df_selling_cma['PROFIT'],label='Profit per day')
plt.plot(df_selling_cma['CMA_4'],label='CMA_4')
plt.plot(df_selling_cma['EMA'],label='EMA')
plt.legend(loc=2)
```




```
from sklearn.metrics import mean_squared_error
rms_error = np.sqrt(mean_squared_error(test['PROFIT'], y_hat_avg.SES))
print("RMS Error is ",rms_error)
```

```
RMS Error is  12323.671065558747
```

Gradient Boosting (dateentered as index, profit as target)

Σε πρώτη φάση καθαρίζουμε τα δεδομένα και τα χωρίζουμε σε train set και test set σε αναλογία 80-20%. Υπολογίζουμε και προσθέτουμε το profit ανά date, την οποία και χρησιμοποιούμε σαν index καθώς τα δεδομένα μας είναι πλέον ομαδοποιημένα ως προς αυτήν.

```
df = df[df['INVOICEDATE'].notna()]

df = df[df['INVOICEDATE'].notna()]
df['DATEENTERED'] = pd.to_datetime(df['DATEENTERED'],infer_datetime_format=True)
df['DATEENTERED'] = df['DATEENTERED'].dt.date

df = df[(df['DATEENTERED']<datetime.date(2019,4,15))]
```

```
### id entity vessel status sell weight modality origin destination
df['PROFIT'] = df['SELL'] - df['BUY']

df2 = df.filter(items=['DATEENTERED','DANGEROUSGOODS','ENTITY','CLIENTID','STATUS','ORIGIN','DE

mydata = df2.groupby(['DATEENTERED','DANGEROUSGOODS','ENTITY','CLIENTID','STATUS','ORIGIN','DES

mydata.head(5)
```

	DATEENTERED	DANGEROUSGOODS	ENTITY	CLIENTID	STATUS	ORIGIN	DESTINATION	PROFIT
0	2017-05-23	N	PO	84335	AAD	Netherlands	Netherlands	-138.75
1	2017-05-29	N	PO	84335	AAD	Netherlands	Netherlands	5.00
2	2017-05-30	N	DISP	84335	AAD	Netherlands	China	50.00
3	2017-06-01	N	PO	84358	AAD	China	Netherlands	30.27
4	2017-06-02	N	DISP	84335	AAD	Netherlands	United Arab Emirates	-165.00

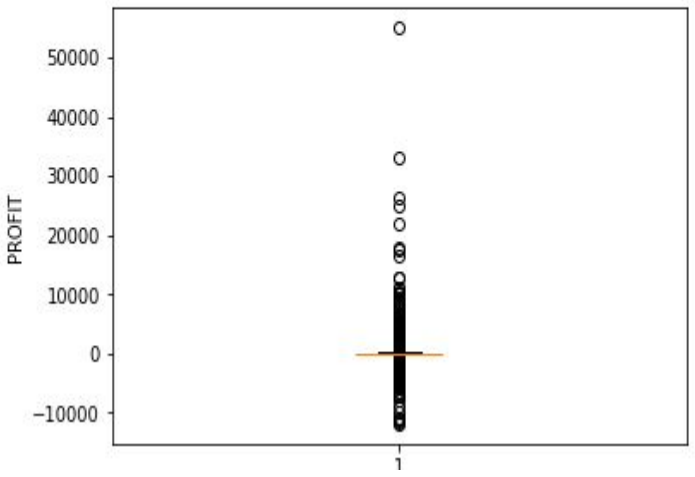
```
train_size = int(len(mydata) * 0.80)

train=mydata[0:train_size].reset_index(drop=True)
test=mydata[train_size:].reset_index(drop=True)

print(test.shape)
print(test.head())
```

```
(9753, 8)
  DATEENTERED DANGEROUSGOODS ENTITY  CLIENTID STATUS                ORIGIN  \
0  2019-01-07              N     PO     84113    DOB  United Arab Emirates
1  2019-01-07              N     PO     84113    AAD                Cyprus
2  2019-01-07              N     PO     83713    AAD                 Korea
3  2019-01-07              N     PO     82600    DOB           Netherlands
4  2019-01-07              N     PO     82600    DOB               Belgium

   DESTINATION  PROFIT
0  Netherlands   21.50
1  Netherlands    9.50
2  Netherlands   25.48
3       Brazil   -1.95
4    Singapore   -6.61
```

```
train["PROFIT"].describe()
```

```
count    39010.000000
mean        66.932235
std        593.319763
min     -11869.454118
25%         8.750000
50%        20.750000
75%        57.660330
max     54999.084000
Name: PROFIT, dtype: float64
```

```
plt.figure()
plt.boxplot(train["PROFIT"])
plt.ylabel("PROFIT")
plt.show()
```

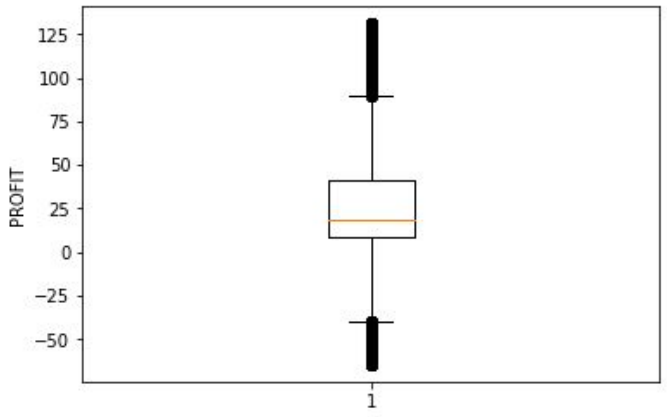


Βλέπουμε τα δεδομένα και κόβουμε πιθανά outliers:

```
q75, q25 = np.percentile(train["PROFIT"], [75, 25])
iqr = q75 - q25

minimum = q25 - (iqr * 1.5)
maximum = q75 + (iqr * 1.5)

print("Minimum = %.2f" % minimum)
print("Maximim = %.2f" % maximum)
```

```
Minimum = -64.62
Maximim = 131.03
```

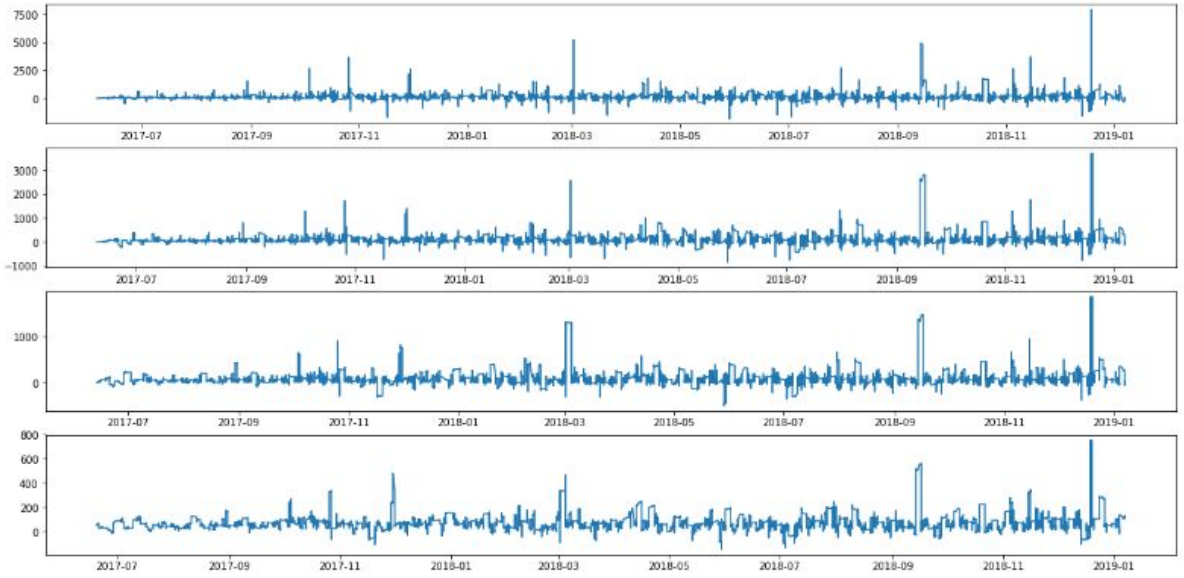
```
plt.figure()
plt.boxplot(train["PROFIT"][(train["PROFIT"] >= minimum) & (train["PROFIT"] <= maximum)])
plt.ylabel("PROFIT")
plt.show()
```



Σε αυτή την ανάλυση επιλέχθηκε μεγαλύτερο παράθυρο για τον υπολογισμό κινούμενων μέσων.

```
periods = [7,15,30,90]

fig = plt.figure(figsize=(20,10))

for n in periods:
    col = "MA" + str(n)
    train[col] = train["PROFIT"].rolling(window=n).mean()

ax1 = fig.add_subplot(411)
ax1.plot(train["DATEENTERED"], train["MA7"])

ax2 = fig.add_subplot(412)
ax2.plot(train["DATEENTERED"], train["MA15"])

ax3 = fig.add_subplot(413)
ax3.plot(train["DATEENTERED"], train["MA30"])

ax4 = fig.add_subplot(414)
ax4.plot(train["DATEENTERED"], train["MA90"])
```

```
[<matplotlib.lines.Line2D at 0x17c2bd38ba8>]
```




```
for col in ["MA7","MA15","MA30","MA90"]:
    train[col].fillna(train["PROFIT"].mean(), inplace=True)
```

Ελέγχουμε τα είδη των δεδομένων ώστε σε δεύτερη φάση να μετατρέψουμε δεδομένα τα κατηγορικα σε αριθμητικά κλπ.

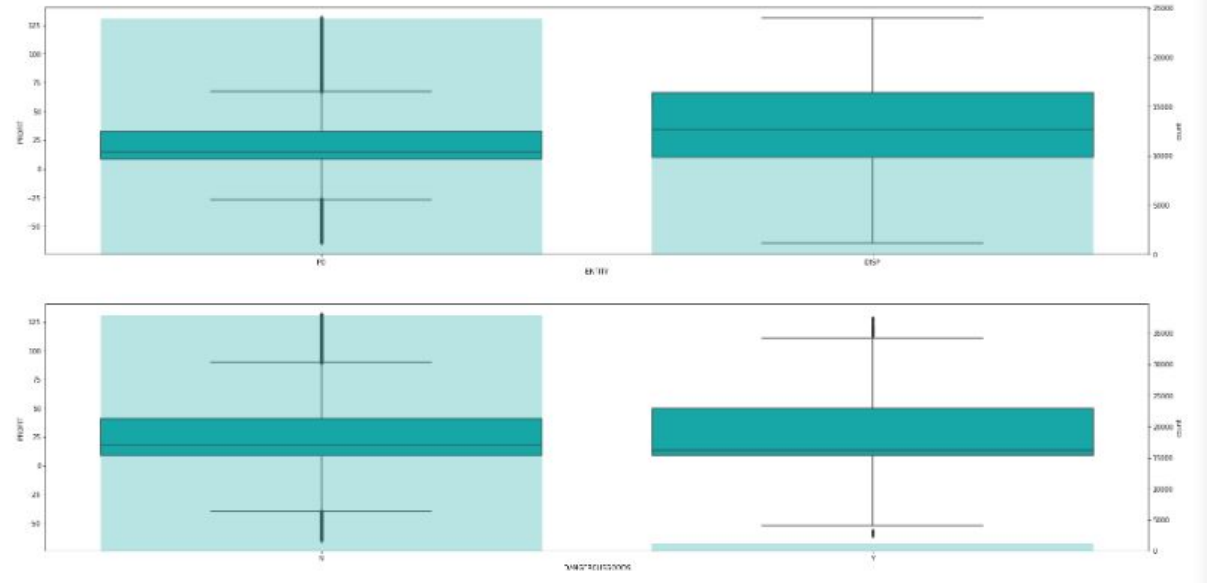
```
counts = [[], [], []]
for c in cols:
    typ = train[c].dtype
    uniq = len(np.unique(train[c]))
    if uniq == 1: counts[0].append(c)
    elif uniq == 2 and typ == np.int64: counts[1].append(c)
    else: counts[2].append(c)

print('Constant features: {}\nBinary features: {}\nCategorical features: {}\n'.format(*[len(c) for c i

print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

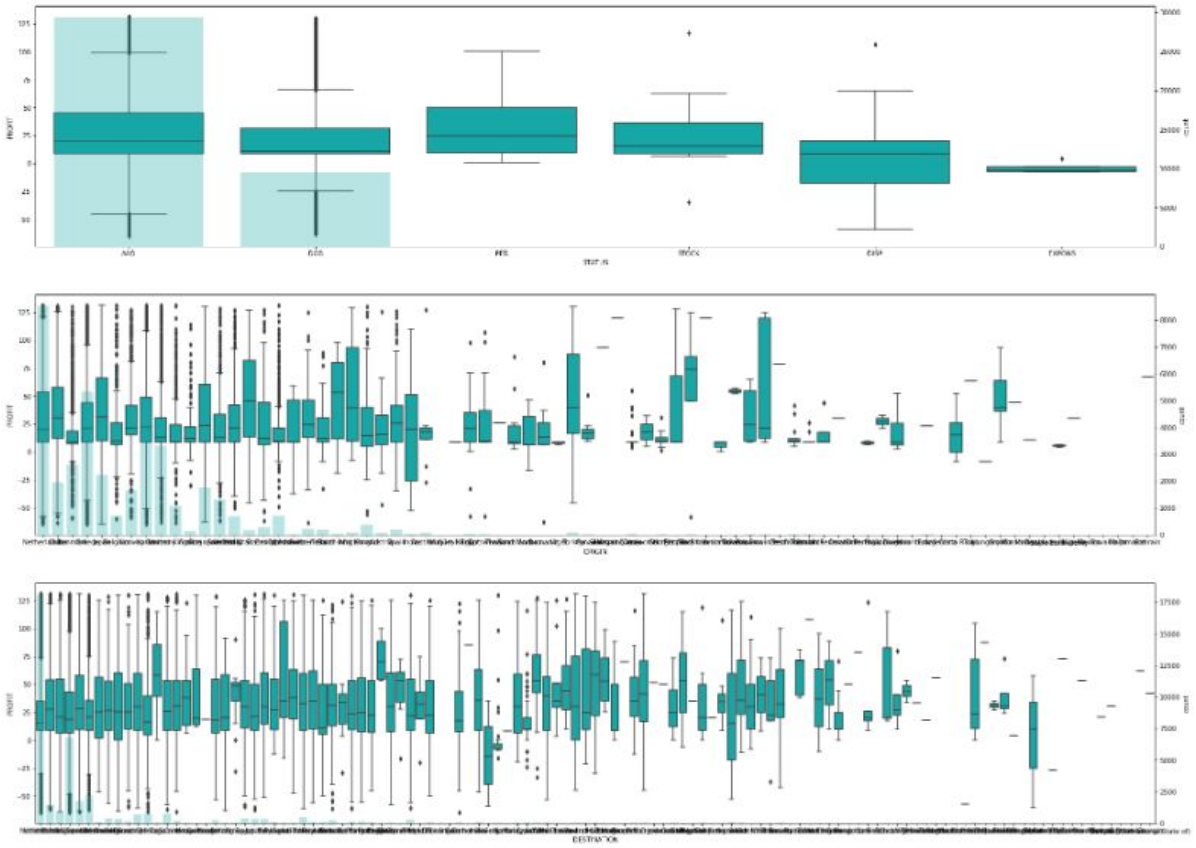
```
Constant features: 0
Binary features: 0
Categorical features: 12

Constant features: []
Categorical features: ['DATEENTERED', 'DANGEROUSGOODS', 'ENTITY', 'CLIENTID', 'STATUS', 'ORIGIN', 'DE
STINATION', 'PROFIT', 'MA7', 'MA15', 'MA30', 'MA90']
```

```
cat_feat =  ['ENTITY','DANGEROUSGOODS', 'STATUS', 'ORIGIN', 'DESTINATION']
```

Παρατηρούμε ότι οι στήλες `DANGEROUSGOODS, ENTITY, STATUS, ORIGIN, DESTINATION` περιέχουν κατηγορικές μεταβλητές τις οποίες θα μετατρέψουμε σε αριθμητικές μέσω μιας dummyfi συνάρτησης.

```
fig, ax = plt.subplots(5, 1, figsize=(30,40))
for c in cat_feat:
    axis = ax[cat_feat.index(c)]
    ax2 = axis.twinx()
    sns.boxplot(x=train[c], y=train["PROFIT"][(train["PROFIT"] >= minimum) & (train["PROFIT"] <= maxim
    sns.countplot(x=train[c], alpha=0.3, color="c", ax=ax2)
```





```
def dummify(df, columns, drop=True):
    for column in columns:
        df_dummies = pd.get_dummies(df[column], prefix=column)
        df = pd.concat([df,df_dummies], axis=1)
        if drop == True:
            df.drop([column], inplace=True, axis=1)

    return df


def add_missing_dummy_columns(df, columns):
    missing_cols = set(columns) - set(df.columns)
    for c in missing_cols:
        df[c] = 0
```

```
old_col_train = list(train.drop(cat_feat, axis=1).columns)
old_col_test = list(test.drop(cat_feat, axis=1).columns)
```

```
train = dummify(train, cat_feat, True)
test = dummify(test, cat_feat, True)
```

```
new_col_train = [c for c in list(train.columns) if c not in old_col_train]
new_col_test = [c for c in list(test.columns) if c not in old_col_test]
```

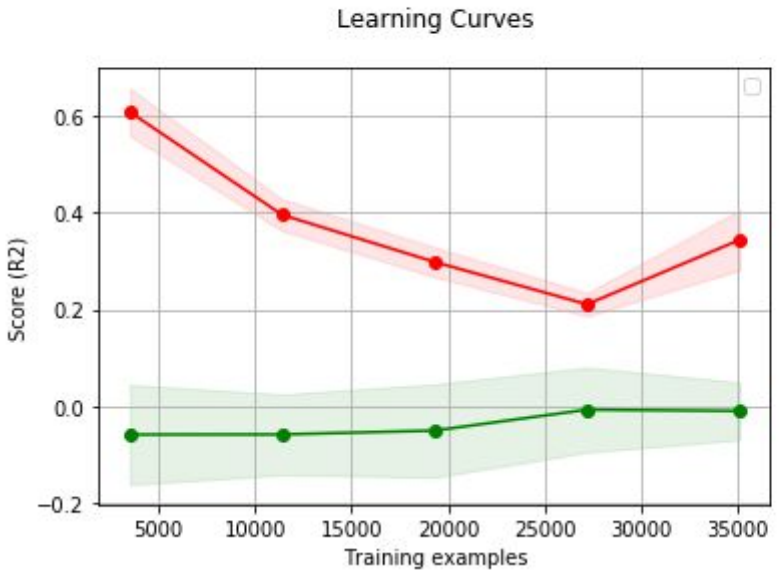
```
add_missing_dummy_columns(test, new_col_train)
add_missing_dummy_columns(train, new_col_test)
```

Φτιάξαμε μια συνάρτηση που για κάθε input οταν τη καλέσεις υπολογίζει και κάνει plot τα learning curves

```
def plot_learning_curves(estimator, X, y, scoring="accuracy", cv=None, n_jobs=1, train_sizes=np.linspa
    plt.figure()
    plt.title("Learning Curves\n")
    plt.xlabel("Training examples")
    plt.ylabel("Score ({})".format(scoring))
    plt.legend(loc="best")
    plt.grid()

    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs, tra

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")

    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1,
                     color="g")

    plt.plot(train_sizes, train_scores_mean, "o-", color="r",
             label="Training score")

    plt.plot(train_sizes, test_scores_mean, "o-", color="g",
            label="Cross-validation score")

    plt.show()
```

```
plot_learning_curves(model, X, y, scoring="R2", cv=10, n_jobs=4)
```

```
No handles with labels found to put in legend.
```



```
model.fit(X, y)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, n_iter_no_change=None, presort='auto',
             random_state=None, subsample=1.0, tol=0.0001,
             validation_fraction=0.1, verbose=0, warm_start=False)
```

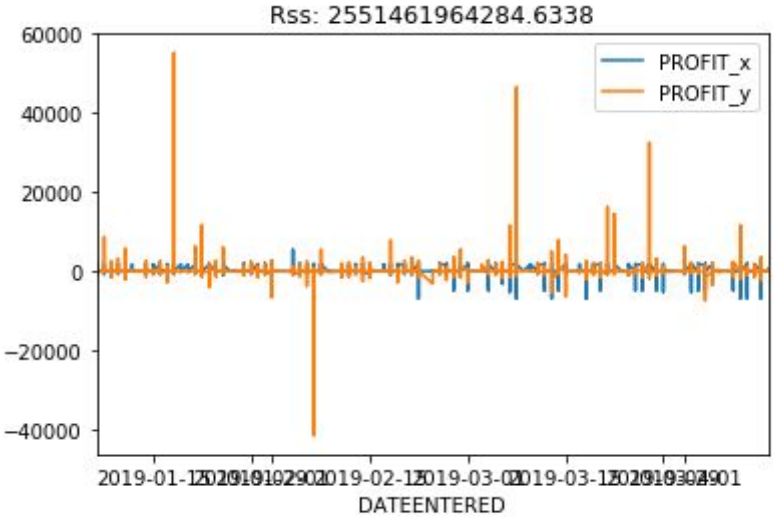
```
y_pred = model.predict(test.drop("DATEENTERED", axis=1))
```

```
y_pred = model.predict(test.drop("DATEENTERED", axis=1))

submission = pd.DataFrame()
submission["DATEENTERED"] = test["DATEENTERED"].values
submission["PROFIT"] = y_pred
print (submission)
```

```
     DATEENTERED       PROFIT
0     2019-01-07    36.269824
1     2019-01-07    34.781390
2     2019-01-07    34.781390
3     2019-01-07  1695.511368
4     2019-01-07    23.856669
5     2019-01-07    22.368235
6     2019-01-07    22.368235
```

```
compare.plot(x="DATEENTERED", y=["PROFIT_x", "PROFIT_y"])
c=(compare.PROFIT_x-compare.PROFIT_y)
plt.title('Rss: %.4f'% sum((c)**2))
```

```
Text(0.5, 1.0, 'Rss: 2551461964284.6338')
```



Gradient Boosting (index as id, profit as target)

Εφαρμόζουμε τα παραπάνω χρησιμοποιώντας το index του κάθε entry σαν id και όχι το date.

```
df['PROFIT'] = df['SELL'] - df['BUY']

df2 = df.filter(items=['ID','ENTITY','DANGEROUSGOODS', 'DATEENTERED','CLIENTID','ST

mydata = df2.groupby(['ID','ENTITY','DATEENTERED', 'DANGEROUSGOODS','CLIENTID','STA

mydata.set_index('ID', inplace=True)
mydata.reset_index(inplace = True)

myavg=mydata.groupby(['DATEENTERED'], as_index=False).count()

mydata = mydata.reset_index()

mydata.drop(['DATEENTERED','ID'], inplace=True, axis=1)

mydata.head(5)
```

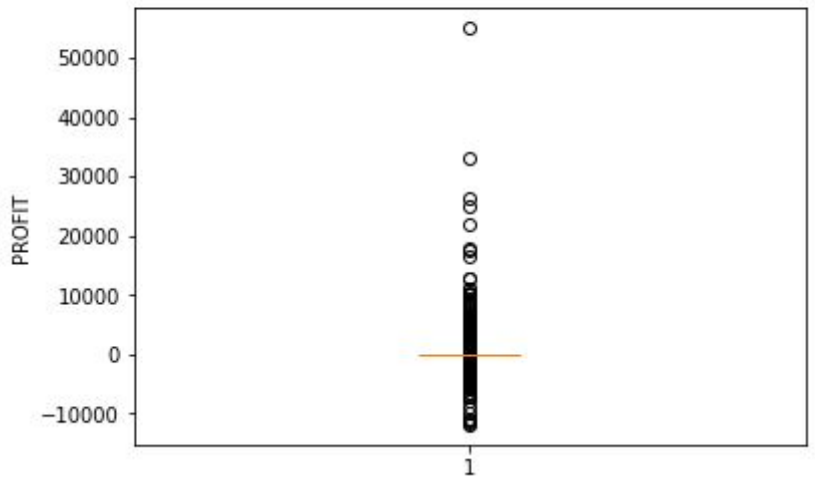
	index	ENTITY	DANGEROUSGOODS	CLIENTID	STATUS	ORIGIN	DESTINATION	PROFIT
0	0	PO	N	84335	AAD	Netherlands	Netherlands	-153.75
1	1	PO	N	84335	AAD	Netherlands	Netherlands	3.75
2	2	PO	N	84335	AAD	Netherlands	Netherlands	3.75
3	3	PO	N	84335	AAD	Netherlands	Netherlands	3.75
4	4	PO	N	84335	AAD	Netherlands	Netherlands	3.75

Και πάλι χωρίζουμε τα δεδομένα σε train και test set σε αναλογία 80-20%.

```
train_size = int(len(mydata) * 0.80)

train=mydata[0:train_size].reset_index(drop=True)
test=mydata[train_size:].reset_index(drop=True)
test2=test.filter(items=['index','PROFIT'])

print(test.shape)
print(test.head())
```

```
(11834, 8)
   index ENTITY DANGEROUSGOODS  CLIENTID STATUS       ORIGIN  DESTINATION  \
0  47335     PO               N     90918    AAD       Greece    Singapore
1  47336     PO               N     13376    AAD       Greece  Netherlands
2  47337     PO               N      8802    AAD       Greece  Netherlands
3  47338     PO               N     84872    AAD      Germany  Netherlands
4  47339     PO               N     84254    AAD  Netherlands  Netherlands

   PROFIT
0   18.82
1    9.50
2   28.55
3    9.50
4   40.50
```

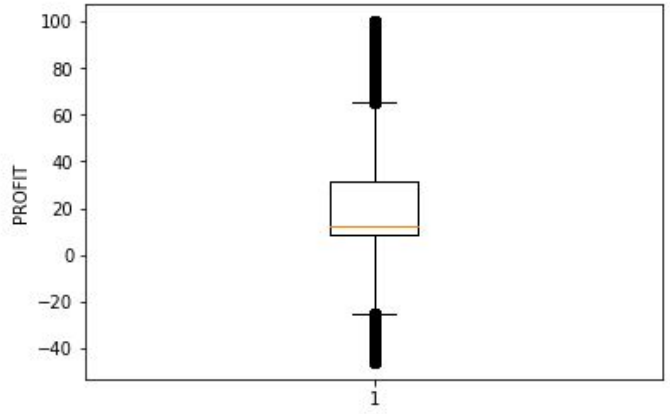
```
plt.figure()
plt.boxplot(train["PROFIT"])
plt.ylabel("PROFIT")
plt.show()
```



```
q75, q25 = np.percentile(train["PROFIT"], [75, 25])
iqr = q75 - q25

minimum = q25 - (iqr * 1.5)
maximum = q75 + (iqr * 1.5)

print("Minimum = %.2f" % minimum)
print("Maximim = %.2f" % maximum)
```

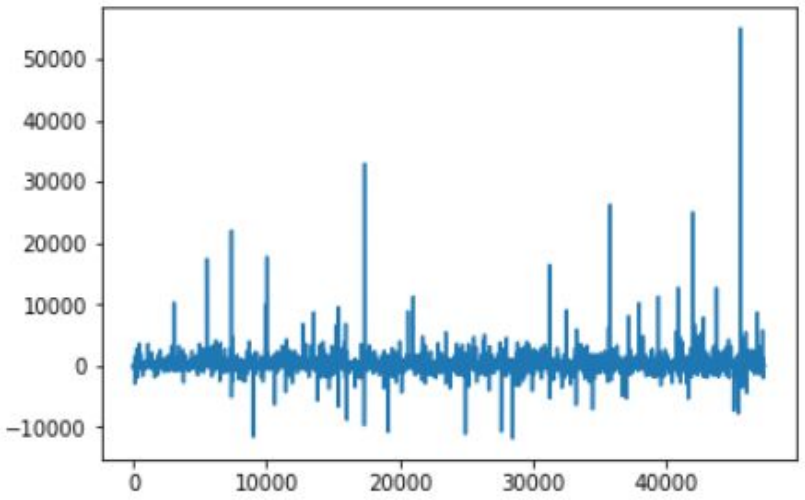
```
Minimum = -45.93
Maximim = 99.88
```

```
plt.figure()
plt.boxplot(train["PROFIT"][(train["PROFIT"] >= minimum) & (train["PROFIT"] <= maximum)])
plt.ylabel("PROFIT")
plt.show()
```



```
plt.plot(train["index"], train["PROFIT"])
```

```
[<matplotlib.lines.Line2D at 0x17400a09978>]
```

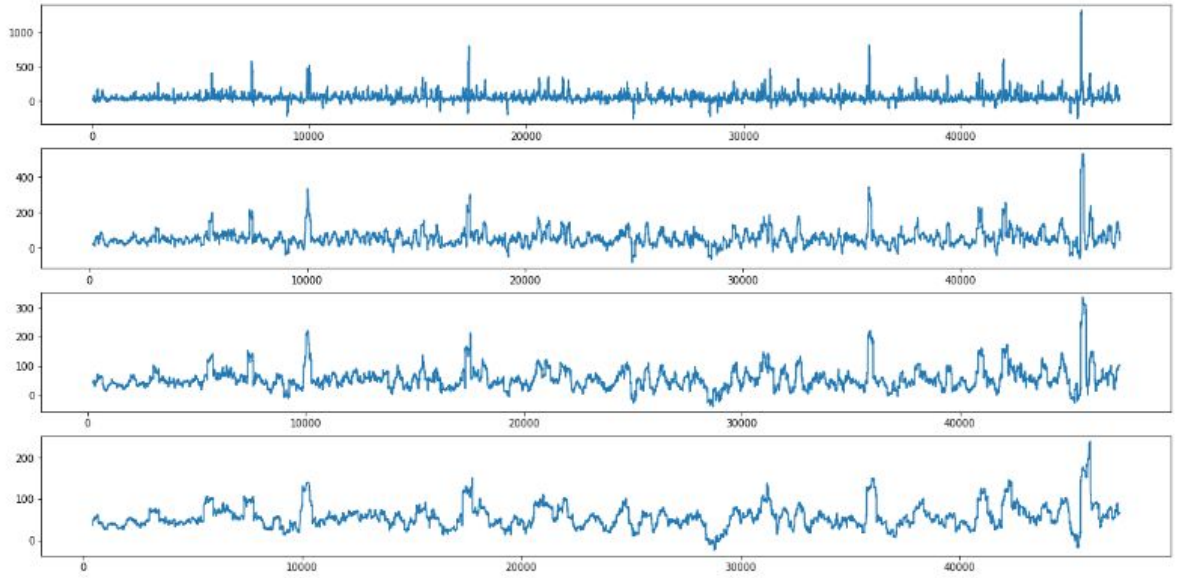



```
print (myavg.mean())
print (myavg)
```

```
ID                94.974318
ENTITY            94.974318
DANGEROUSGOODS    94.974318
CLIENTID          94.974318
STATUS            94.974318
ORIGIN            94.974318
DESTINATION       94.974318
PROFIT            94.974318
dtype: float64
    DATEENTERED   ID  ENTITY  DANGEROUSGOODS  CLIENTID  STATUS  ORIGIN
0    2017-05-23    5       5               5         5       5       5
1    2017-05-29    1       1               1         1       1       1
2    2017-05-30    1       1               1         1       1       1
3    2017-06-01    1       1               1         1       1       1
4    2017-06-02    1       1               1         1       1       1
5    2017-06-06    4       4               4         4       4       4
6    2017-06-07    3       3               3         3       3       3
7    2017-06-08    2       2               2         2       2       2
```

```
periods = [50,150,250,450]

fig = plt.figure(figsize=(20,10))

for n in periods:
    col = "MA" + str(n)
    train[col] = train["PROFIT"].rolling(window=n).mean()

ax1 = fig.add_subplot(411)
ax1.plot(train["index"], train["MA50"])

ax2 = fig.add_subplot(412)
ax2.plot(train["index"], train["MA150"])

ax3 = fig.add_subplot(413)
ax3.plot(train["index"], train["MA250"])

ax4 = fig.add_subplot(414)
ax4.plot(train["index"], train["MA450"])
```

```
[<matplotlib.lines.Line2D at 0x1746a9c0f98>]
```



```
for col in ["MA50","MA150","MA250","MA450"]:
    train[col].fillna(train["PROFIT"].mean(), inplace=True)
```

```
cols = [c for c in train.columns]
print('Number of features: {}'.format(len(cols)))

print('Feature types:')
train[cols].dtypes.value_counts()
```

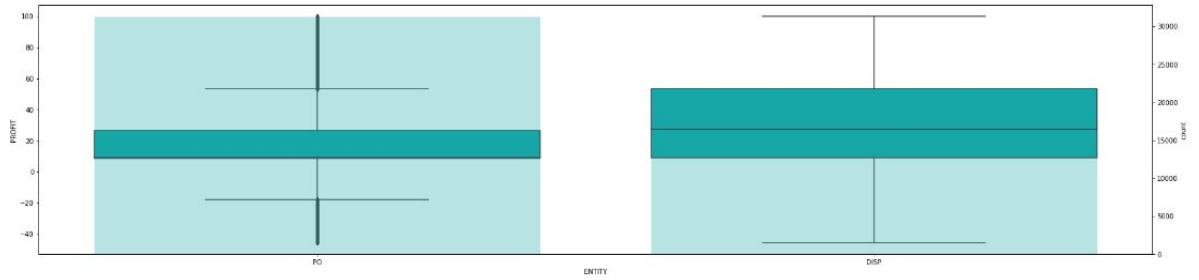
```
Number of features: 12
Feature types:

object     5
float64    5
int64      2
dtype: int64
```

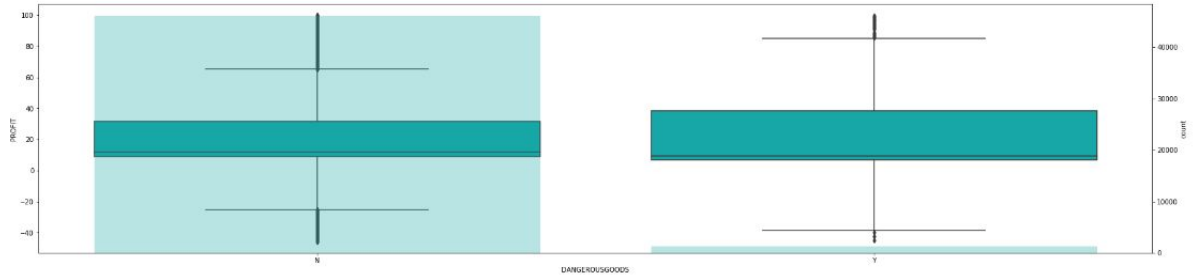
```
counts = [[], [], []]
for c in cols:
    typ = train[c].dtype
    uniq = len(np.unique(train[c]))
    if uniq == 1: counts[0].append(c)
    elif uniq == 2 and typ == np.int64: counts[1].append(c)
    else: counts[2].append(c)

print('Constant features: {}\nBinary features: {}\nCategorical features: {}\n'.format(*[len(c) for c in counts]))

print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

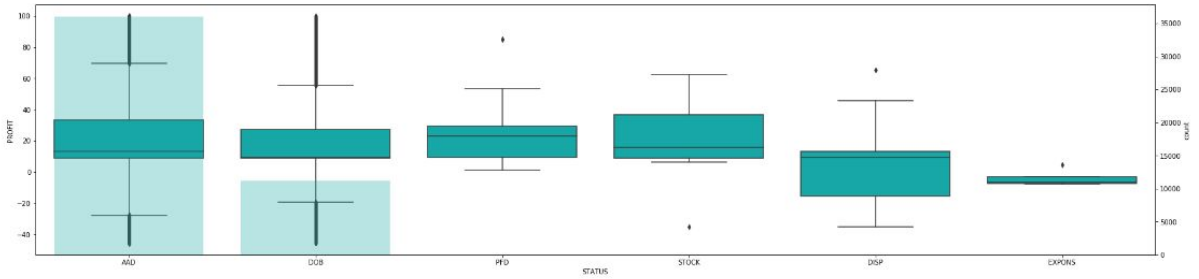
```
Constant features: 0
Binary features: 0
Categorical features: 12

Constant features: []
Categorical features: ['index', 'ENTITY', 'DANGEROUSGOODS', 'CLIENTID', 'STATUS', 'ORIGIN', 'DESTINATION', 'PROFIT', 'MA50', 'M
A150', 'MA250', 'MA450']
```

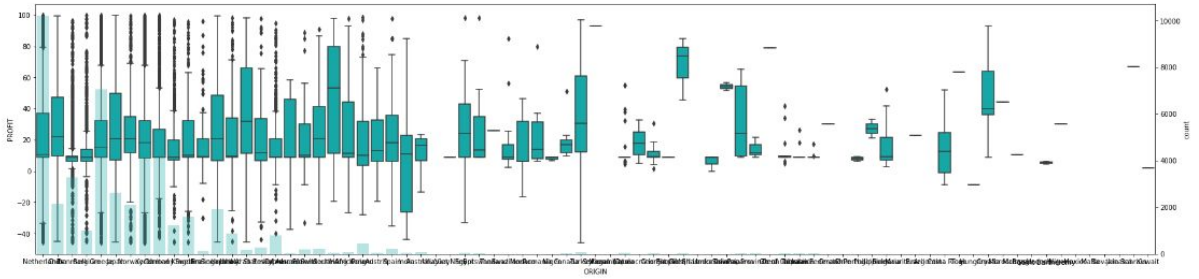
```
cat_feat =  ['ENTITY','DANGEROUSGOODS', 'STATUS', 'ORIGIN', 'DESTINATION']
```

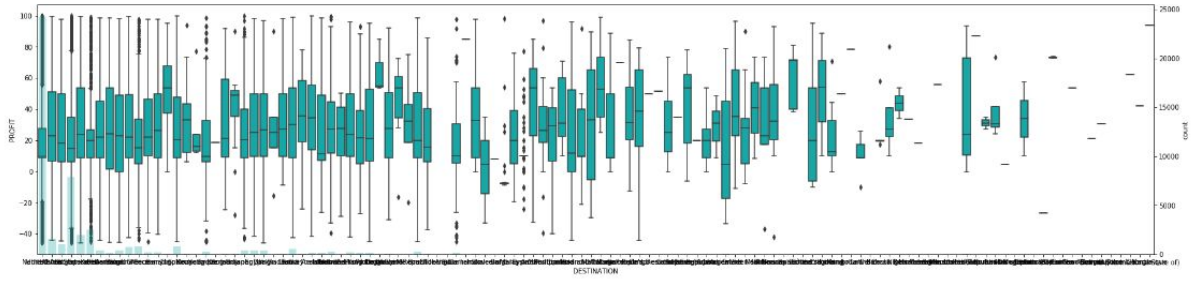
```
fig, ax = plt.subplots(5, 1, figsize=(30,40))
for c in cat_feat:
    axis = ax[cat_feat.index(c)]
    ax2 = axis.twinx()
    sns.boxplot(x=train[c], y=train["PROFIT"][(train["PROFIT"] >= minimum) & (train["PROFIT"] <= maximum)], color="c", ax=axis)
    sns.countplot(x=train[c], alpha=0.3, color="c", ax=ax2)
```











```
train.head()
```

	index	ENTITY	DANGEROUSGOODS	CLIENTID	STATUS	ORIGIN	DESTINATION	PROFIT	MA50	MA150	MA250	MA450
0	0	PO	N	84335	AAD	Netherlands	Netherlands	-153.75	56.213093	56.213093	56.213093	56.213093
1	1	PO	N	84335	AAD	Netherlands	Netherlands	3.75	56.213093	56.213093	56.213093	56.213093
2	2	PO	N	84335	AAD	Netherlands	Netherlands	3.75	56.213093	56.213093	56.213093	56.213093
3	3	PO	N	84335	AAD	Netherlands	Netherlands	3.75	56.213093	56.213093	56.213093	56.213093
4	4	PO	N	84335	AAD	Netherlands	Netherlands	3.75	56.213093	56.213093	56.213093	56.213093

```
train.drop(["MA50","MA150","MA250","MA450"],axis=1, inplace=True)
```

```
def dummify(df, columns, drop=True):
    for column in columns:
        df_dummies = pd.get_dummies(df[column], prefix=column)
        df = pd.concat([df,df_dummies], axis=1)
        if drop == True:
            df.drop([column], inplace=True, axis=1)

    return df


def add_missing_dummy_columns(df, columns):
    missing_cols = set(columns) - set(df.columns)
    for c in missing_cols:
        df[c] = 0
```

```
print("Train: {}".format(train.shape))
print("Test: {}".format(test.shape))
```

```
Train: (47335, 219)
Test: (11834, 219)
```

```
test.drop(['PROFIT'], inplace=True, axis=1)

print("Train: {}".format(train.shape))
print("Test: {}".format(test.shape))
```

```
Train: (47335, 219)
Test: (11834, 218)
```

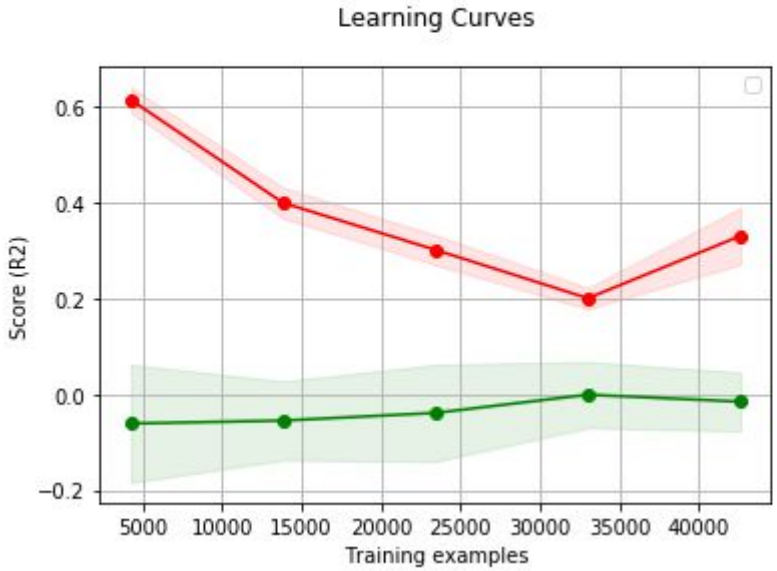
```
model = GradientBoostingRegressor()
X = train.drop(["index","PROFIT"], axis=1)
y = train["PROFIT"]
```

```
def plot_learning_curves(estimator, X, y, scoring="accuracy", cv=None, n_jobs=1, train_sizes=np.linspace(0.1,1.0,5)):
    plt.figure()
    plt.title("Learning Curves\n")
    plt.xlabel("Training examples")
    plt.ylabel("Score ({})".format(scoring))
    plt.legend(loc="best")
    plt.grid()

    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")

    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1,
                     color="g")

    plt.plot(train_sizes, train_scores_mean, "o-", color="r",
             label="Training score")

    plt.plot(train_sizes, test_scores_mean, "o-", color="g",
            label="Cross-validation score")

    plt.show()
```

Ακολουθώ την ίδια διαδικασία για το dummification των κατηγορικών δεδομένων.

```
plot_learning_curves(model, X, y, scoring="R2", cv=10, n_jobs=4)
```

```
No handles with labels found to put in legend.
```




```
model.fit(X, y)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, n_iter_no_change=None, presort='auto',
             random_state=None, subsample=1.0, tol=0.0001,
             validation_fraction=0.1, verbose=0, warm_start=False)
```

```
y_pred = model.predict(test.drop("index", axis=1))
```

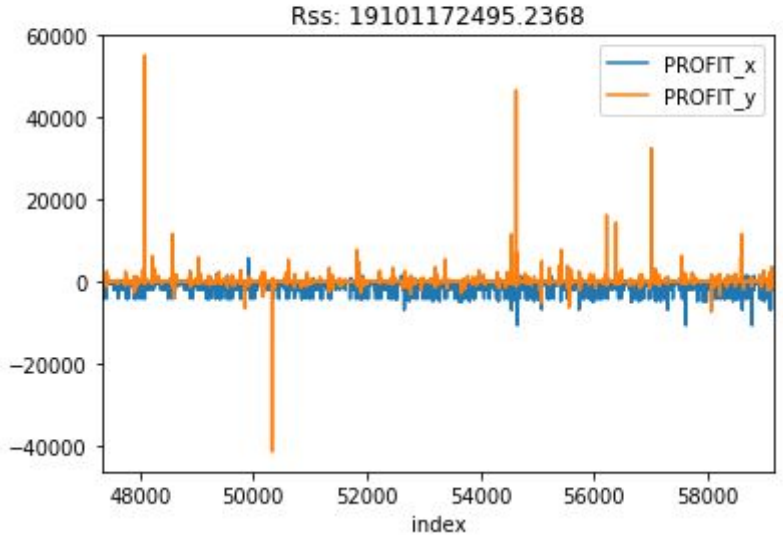
```
y_pred = model.predict(test.drop("index", axis=1))

submission = pd.DataFrame()
submission["index"] = test["index"].values
submission["PROFIT"] = y_pred
print (submission)
```

	index	PROFIT
0	47335	-4188.704368
1	47336	23.190971
2	47337	21.700208
3	47338	10.282809
4	47339	25.762596
5	47340	58.731734
6	47341	58.731734
7	47342	58.731734
8	47343	58.731734
9	47344	58.731734
10	47345	-661.758638
11	47346	3.987252

Και σε αυτή τη περίπτωση έχουμε μεγάλο error αλλα μικρότερο σε σχέση με τη προηγούμενη εφαρμογη του μοντέλου.

```
compare.plot(x="index", y=["PROFIT_x", "PROFIT_y"])
c=(compare.PROFIT_x-compare.PROFIT_y)
plt.title('Rss: %.4f'% sum((c)**2))
```

```
Text(0.5, 1.0, 'Rss: 19101172495.2368')
```



Εφαρμογή μοντέλων με target το Weight

Σε αυτή τη περίπτωση εφαρμόζουμε όλα τα παραπάνω προσπαθώντας αυτή τη φορά να προβλέψουμε το συνολικό βάρος μεταφοράς. Αυτό έχει ώς αποτέλεσμα να μειωθεί σημαντικά το dataset καθώς βλέπουμε τα δεδομένα σε επίπεδο dispatch και όχι πολλαπλών κόστων, αλλά να έχουμε μια πιο αντιπροσωπευτική εικόνα του τι συμβαίνει ανά dispatch.

Arima (weight as target)

Σε πρώτη φάση γίνεται διαχωρισμός του συνόλου δεδομένων φιλτράροντας ως προς τα dispatches

```
df = df[df['INVOICEDATE'].notna()]
df = df[df['ENTITY']=='DISP']
```

```
df['DATEENTERED'] = pd.to_datetime(df['DATEENTERED'],infer_datetime_format=True)
df['DATEENTERED'] =df['DATEENTERED'].dt.date

df1 = df.filter(items=['DATEENTERED','ID', 'WEIGHT'])

df2 = df1.drop_duplicates()

df2 = df.filter(items=['DATEENTERED', 'WEIGHT']).sort_values(by=['DATEENTERED'])


df2['WEIGHT'].fillna(0, inplace=True)

df3 = df2.dropna()

df3 = df3[(df3['DATEENTERED']<datetime.date(2019,4,15))]
```

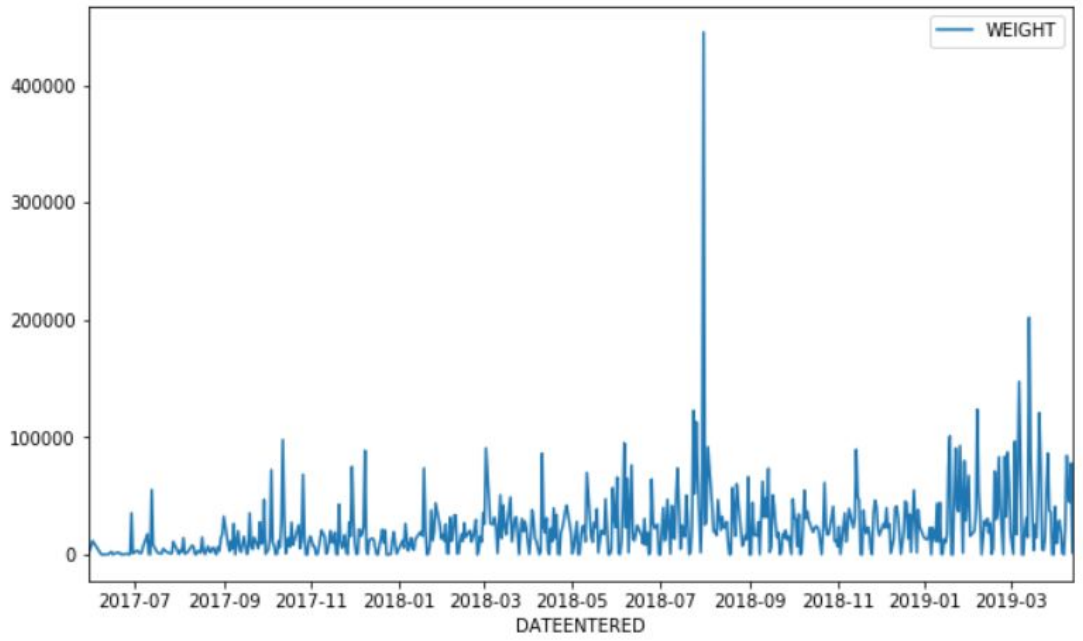
```
df3.tail(6)
```

	DATEENTERED	WEIGHT
24472	2019-04-13	30.0
204268	2019-04-13	477.0
24473	2019-04-13	30.0

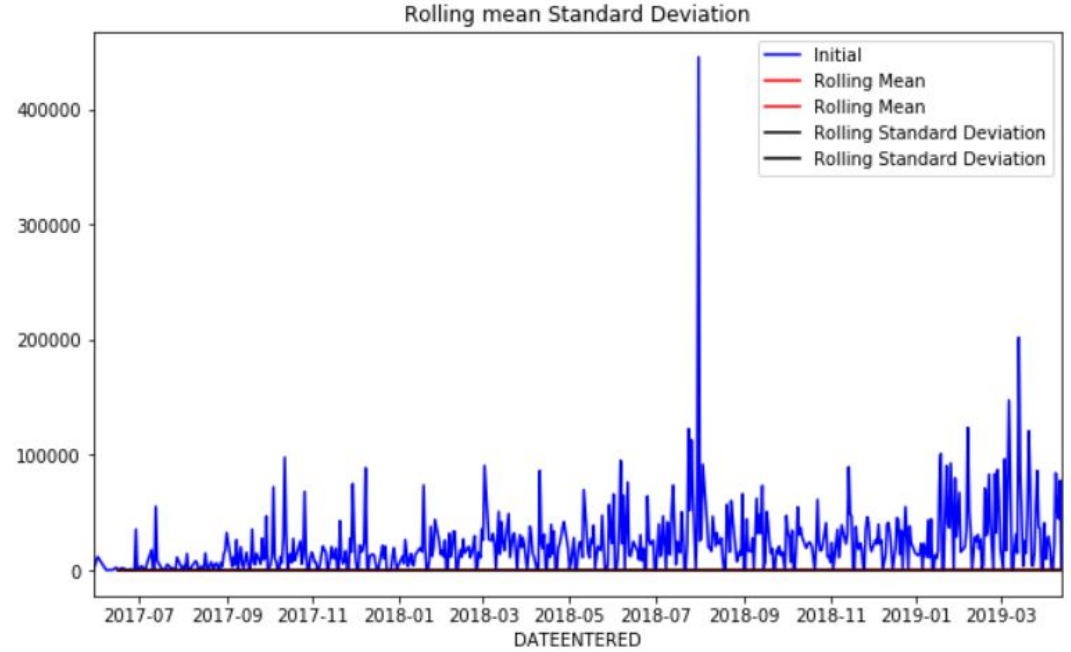
```
dftotals=pd.DataFrame(df3, columns=['DATEENTERED', 'WEIGHT']).groupby('DATEENTERED').sum().sort_values

dftotals.tail(15)
```

	DATEENTERED	WEIGHT
572	2019-03-30	258.00
573	2019-03-31	1.20
574	2019-04-01	40873.00
575	2019-04-02	9657.20
576	2019-04-03	24100.00

```
dftotals.plot(x="DATEENTERED", y=["WEIGHT"])
plt.show()

dfprofit=pd.DataFrame(dftotals, columns=['DATEENTERED', 'PROFIT'])
dfprofit.reset_index(inplace=True)
dfprofit['DATEENTERED'] = pd.to_datetime(dfprofit['DATEENTERED'])
dfprofit = dfprofit.set_index('DATEENTERED')
```



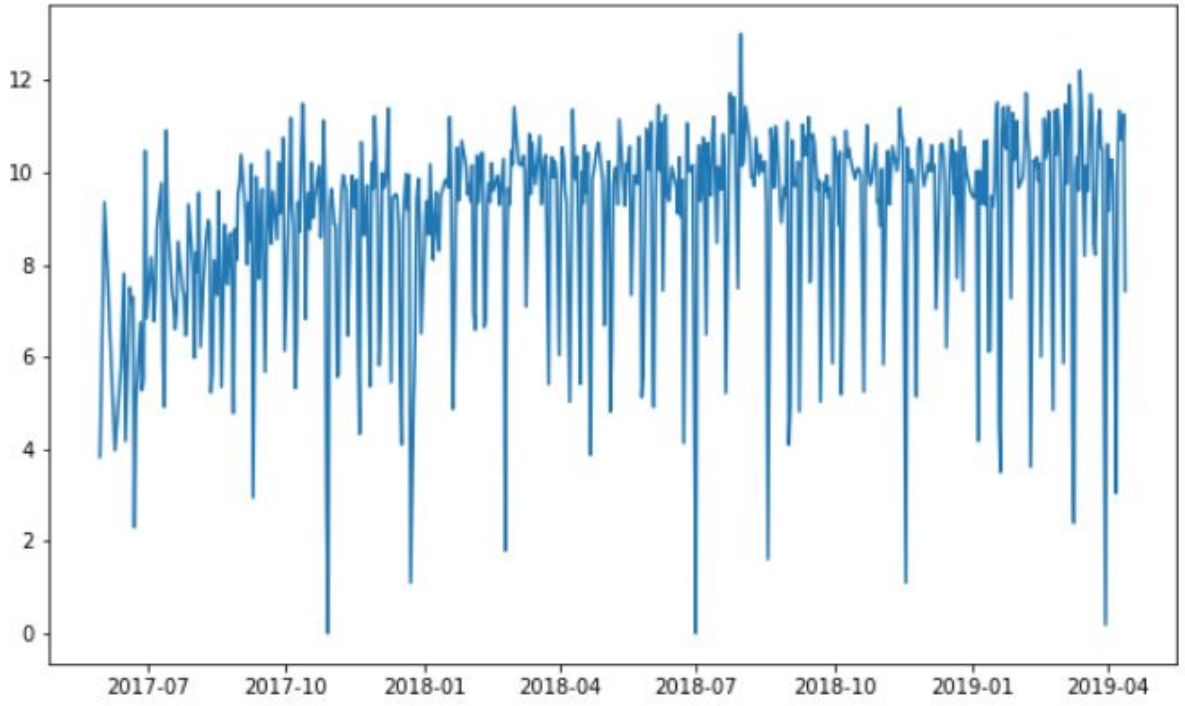

```
dftotals.plot(x="DATEENTERED", y="WEIGHT" , color='blue', label='Initial')
plt.plot(rolmean, color='red', label='Rolling Mean')
plt.plot(rolstd, color='black', label='Rolling Standard Deviation')
plt.legend(loc='best')
plt.title('Rolling mean Standard Deviation')
plt.show(block=False)
```



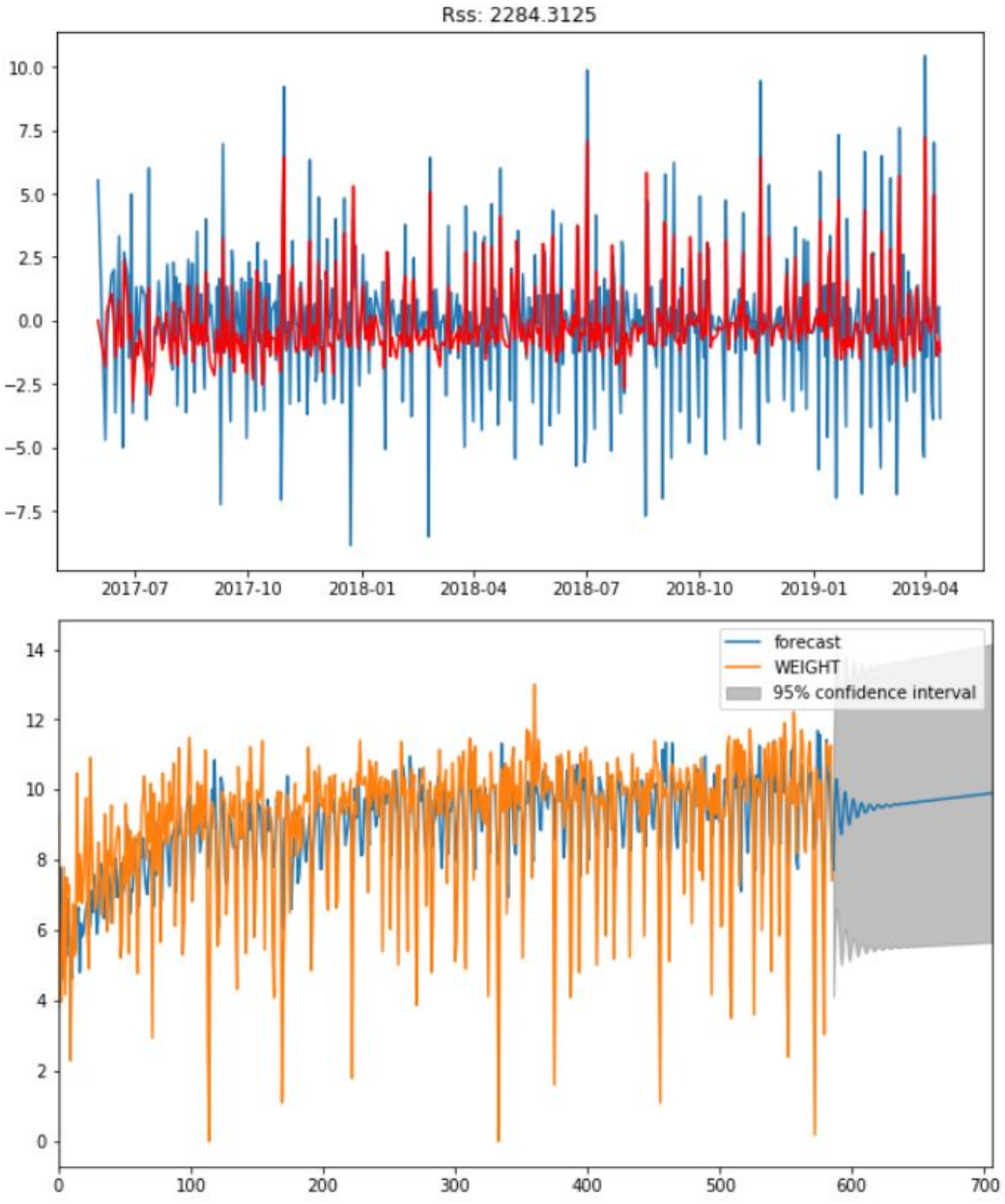
```
from statsmodels.tsa.stattools import adfuller

print ('Results of Dickey-Fuller Test:')
dftest = adfuller(dftotals['WEIGHT'], autolag='AIC')

dfoutput = pd.Series(dftest[0:4], index=['Test Statistic','p-value','L
for key,value in dftest[4].items():
    dfoutput['Critical Value (%s)'%key] = value

print(dfoutput)
```

```
Results of Dickey-Fuller Test:
Test Statistic                -5.156304
p-value                        0.000011
Lags Used                      6.000000
Nr of Observations Used      580.000000
Critical Value (1%)           -3.441675
Critical Value (5%)           -2.866536
Critical Value (10%)          -2.569431
dtype: float64
```

Στο παραπάνω διάγραμμα παρατηρούμε οτι το p value είναι πολύ μικρό (<0.05) πράγμα που σημαίνει ότι η χρονοσειρά είναι στατική.



```
from statsmodels.tsa.arima_model import ARIMA

#AR
model = ARIMA(dfprofit_logScale, order=(0, 1, 2))
results_AR = model.fit(disp=-1)
plt.plot(df_log_diff_shifting)
plt.plot(results_AR.fittedvalues, color='red' )
a=(results_AR.fittedvalues-df_log_diff_shifting["WEIGHT"])
a.dropna(inplace=True)
plt.title('Rss: %.4f'% sum((a)**2))
print ('Plot AR Model')
```



Exponential Smoothing (weight as target)

Τα δεδομένα σε κάθε περίπτωση φιλτράρονται ως προς distinct ID με entity DISP

```
import pandas as pd
df = pd.read_csv('C:/Users/uocen/Downloads/thesis_project/mb_tr.csv')

df['DATEENTERED'] = pd.to_datetime(df['DATEENTERED']).dt.date
df = df[df['INVOICEDATE'].notna()]
df = df[df['ENTITY']=='DISP']
df2 = df.filter(items=['DATEENTERED', 'ID', 'WEIGHT' ]).sort_values(by=['DATEENTERED'])
df2 = df2.drop_duplicates()

df2['WEIGHT'].fillna(0, inplace=True)
df2.tail(5)

df3 = df2.filter(items=['DATEENTERED', 'WEIGHT' ])
```

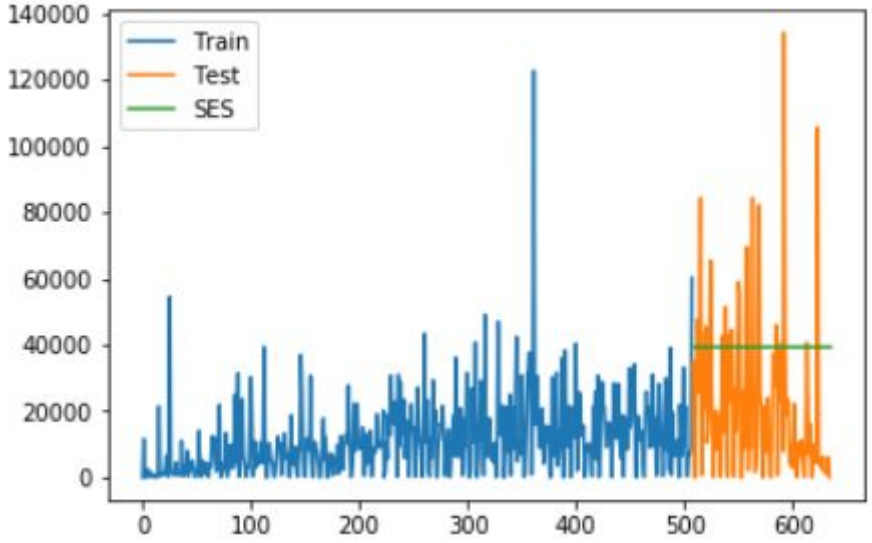
```
dfselling=pd.DataFrame(df3, columns=['DATEENTERED', 'WEIGHT']).groupby('DATEENTERED').sum().sort_value

dfselling.tail(5)
```

	DATEENTERED	WEIGHT
630	2019-06-04	2259.0
631	2019-06-05	959.5
632	2019-06-06	996.0
633	2019-06-07	6113.0
634	2019-06-10	3.0

```
series= dfselling.iloc[:,1:2]
series2= dfselling.iloc[:-90,1:2]
Y_train = series['WEIGHT'].values

print(len(series), len(series2))
```

```
635 545
```

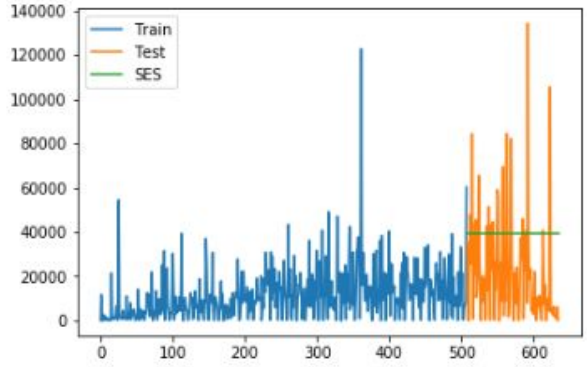
```
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import SimpleExpSmoothing, Holt, ExponentialSmoothing

train_size = int(len(series) * 0.8)

train=series[0:train_size]
test=series[train_size:]

y_hat_avg = test.copy()
fit1 = SimpleExpSmoothing(np.asarray(train['WEIGHT'])).fit(smoothing_level=0.6, optimized=False)
y_hat_avg['SES'] = fit1.forecast(len(test))
plt.plot(train['WEIGHT'], label = 'Train')
plt.plot(test['WEIGHT'], label = 'Test')
plt.plot(y_hat_avg['SES'], label = 'SES')
plt.legend(loc='best')
plt.show()
```




```
from sklearn.metrics import mean_squared_error
from math import sqrt

rms = sqrt(mean_squared_error(test.WEIGHT, y_hat_avg.SES))
print(rms)
```

```
29338.911229829584
```

```
train_size = int(len(series) * 0.8)

train=series[0:train_size]
test=series[train_size:]

y_hat_avg = test.copy()
fit1 = SimpleExpSmoothing(np.asarray(train['WEIGHT'])).fit(smoothing_level=0.6, optimized=False)
y_hat_avg['SES'] = fit1.forecast(len(test))
plt.plot(train['WEIGHT'], label = 'Train')
plt.plot(test['WEIGHT'], label = 'Test')
plt.plot(y_hat_avg['SES'], label = 'SES')
plt.legend(loc='best')
plt.show()
```




```
rms = sqrt(mean_squared_error(test.WEIGHT, y_hat_avg.SES))
print(rms)
```

```
15646.957097335542
```

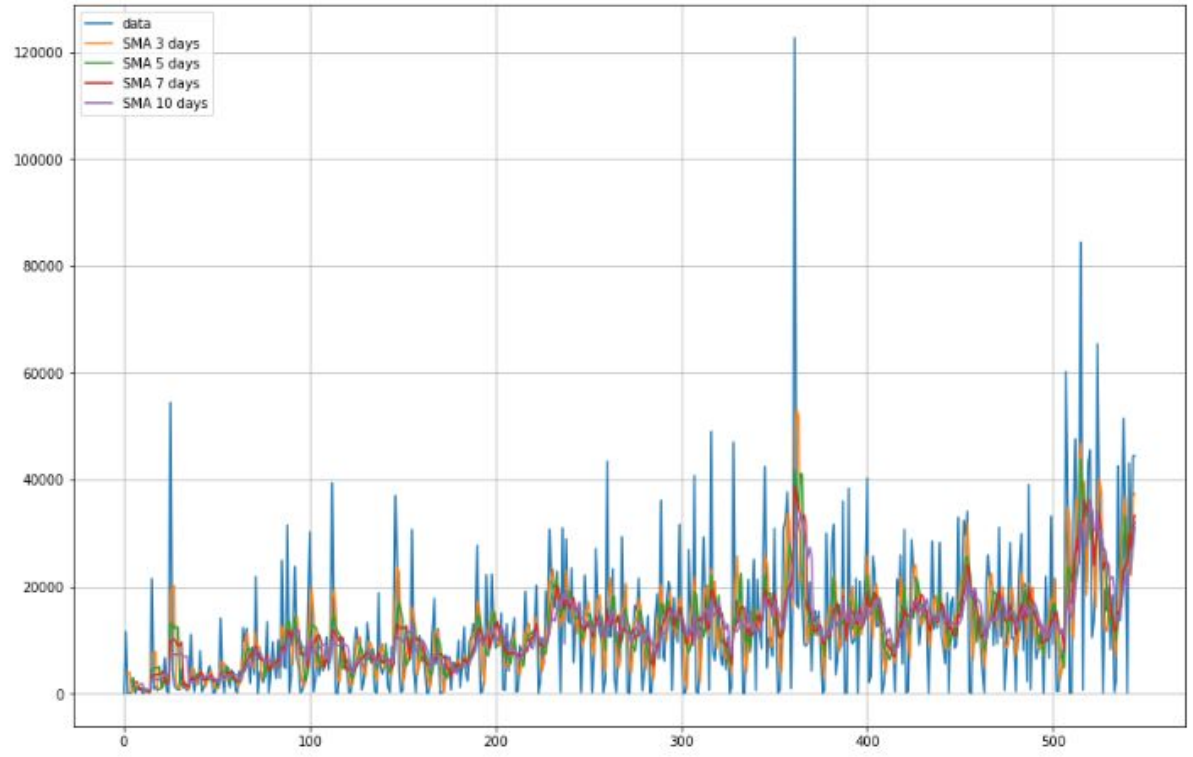
```
dfselling = dfselling.iloc[ :-90,:]

dfselling['SMA_3'] = dfselling.iloc[:,1].rolling(window=3).mean()

dfselling['SMA_5'] = dfselling.iloc[:,1].rolling(window=5).mean()

dfselling['SMA_7'] = dfselling.iloc[:,1].rolling(window=7).mean()

dfselling['SMA_10'] = dfselling.iloc[:,1].rolling(window=10).mean()

dfselling.head(5)
```

	DATEENTERED	WEIGHT	SMA_3	SMA_5	SMA_7	SMA_10
0	2017-05-30	23.0	NaN	NaN	NaN	NaN
1	2017-06-02	11596.0	NaN	NaN	NaN	NaN
2	2017-06-08	53.0	3890.666667	NaN	NaN	NaN
3	2017-06-09	53.0	3900.666667	NaN	NaN	NaN
4	2017-06-13	329.0	145.000000	2410.8	NaN	NaN

Υπολογίζεται Simple Moving Average για 3, 5, 7 και 10 μέρες αντίστοιχα

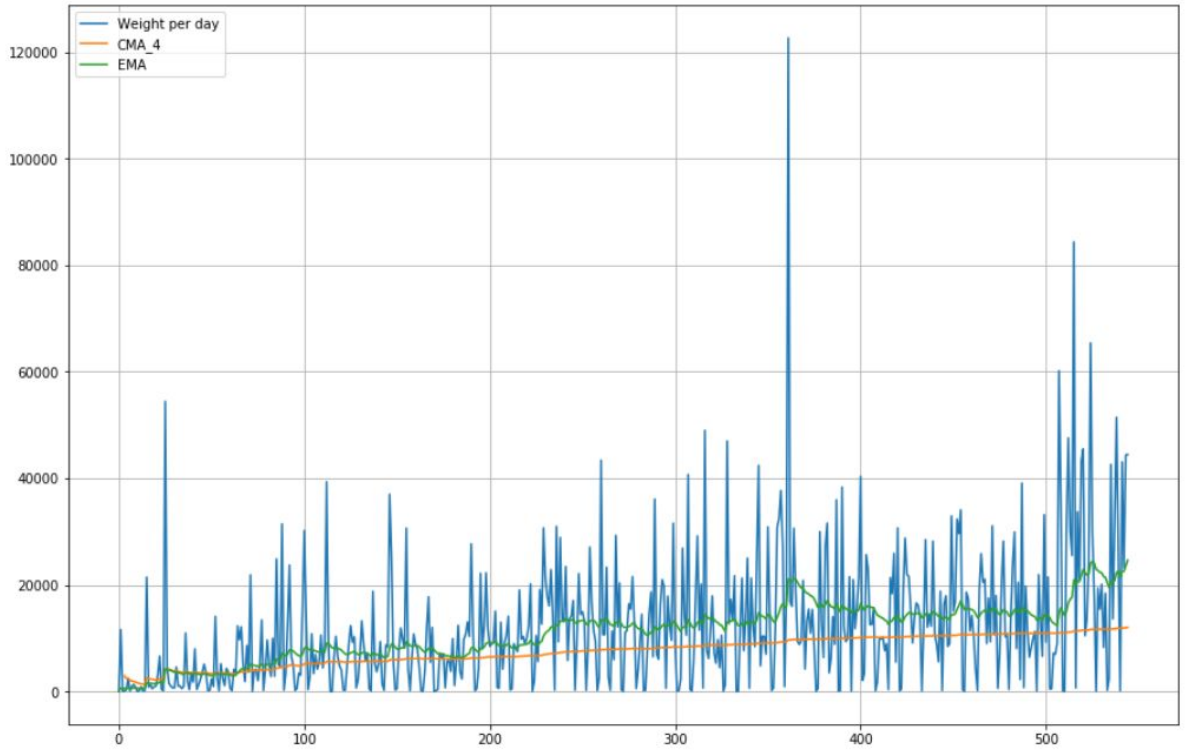
```
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=[15,10])
plt.grid(True)
plt.plot(dfselling['WEIGHT'],label='data')
plt.plot(dfselling['SMA_3'],label='SMA 3 days')
plt.plot(dfselling['SMA_5'],label='SMA 5 days')
plt.plot(dfselling['SMA_7'],label='SMA 7 days')
plt.plot(dfselling['SMA_10'],label='SMA 10 days')
plt.legend(loc=2)
```

```
<matplotlib.legend.Legend at 0x1f3f9fcb748>
```



Και ακολουθεί το Cumulative Moving Average

```
#Cumulative Moving Average

df_selling_cma=pd.DataFrame(dfselling.iloc[:,1])
df_selling_cma['CMA_4'] = df_selling_cma.expanding(min_periods=4).mean()

# Exponential Moving Average
df_selling_cma['EMA'] = df_selling_cma.iloc[:,0].ewm(span=40,adjust=False).mean()

plt.figure(figsize=[15,10])
plt.grid(True)
plt.plot(df_selling_cma['WEIGHT'],label='Weight per day')
plt.plot(df_selling_cma['CMA_4'],label='CMA_4')
plt.plot(df_selling_cma['EMA'],label='EMA')
plt.legend(loc=2)
```




```
from sklearn.metrics import mean_squared_error
rms_error = np.sqrt(mean_squared_error(test['WEIGHT'], y_hat_avg.SES))
print("RMS Error is ",rms_error)
```

```
RMS Error is  15646.957097335542
```

Gradient Boosting dateentered - weigth (weight as target)

```
df = pd.read_csv('C:/Users/uocen/Downloads/thesis_project/mb_tr.csv')

df = df[df['INVOICEDATE'].notna()]

df = df[df['ENTITY']=='DISP']
df['DATEENTERED'] = pd.to_datetime(df['DATEENTERED'],infer_datetime_format=True)
df['DATEENTERED'] = df['DATEENTERED'].dt.date

df = df[(df['DATEENTERED']<datetime.date(2019,4,15))]
```

```
df1 = df.filter(items=['DATEENTERED','DANGEROUSGOODS','ENTITY','CLIEN
df2 = df1.drop_duplicates()
mydata = df2.groupby(['DATEENTERED','DANGEROUSGOODS','ENTITY','CLIENT
```

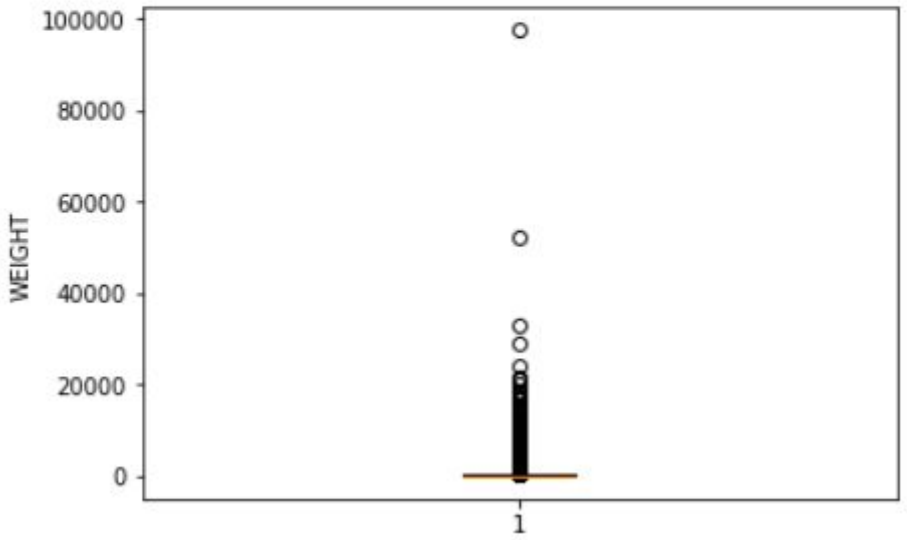
	DATEENTERED	DANGEROUSGOODS	ENTITY	CLIENTID	STATUS	ORIGIN	DESTINATION	WEIGHT
0	2017-05-30	N	DISP	84335	AAD	Netherlands	China	23.0
1	2017-06-02	N	DISP	84335	AAD	Netherlands	United Arab Emirates	11596.0
2	2017-06-08	N	DISP	84358	AAD	Japan	United States of America	53.0
3	2017-06-09	N	DISP	12880	DOB	Korea	Korea	53.0
4	2017-06-13	N	DISP	12377	AAD	Netherlands	Panama	329.0

```
train_size = int(len(mydata) * 0.80)

train=mydata[0:train_size].reset_index(drop=True)
test=mydata[train_size:].reset_index(drop=True)
test2=mydata[train_size:].reset_index(drop=True)
```

```
train["WEIGHT"].describe()
```

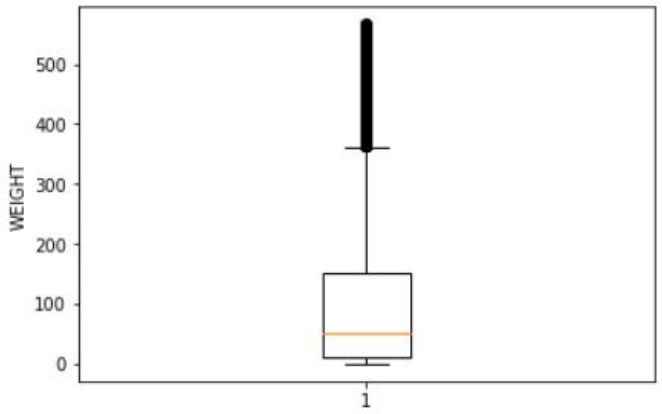
```
count    15198.000000
mean       358.033353
std       1505.327461
min          0.000000
25%         15.000000
50%         70.000000
75%        236.000000
max      97629.000000
Name: WEIGHT, dtype: float64
```

```
plt.figure()
plt.boxplot(train["WEIGHT"])
plt.ylabel("WEIGHT")
plt.show()
```



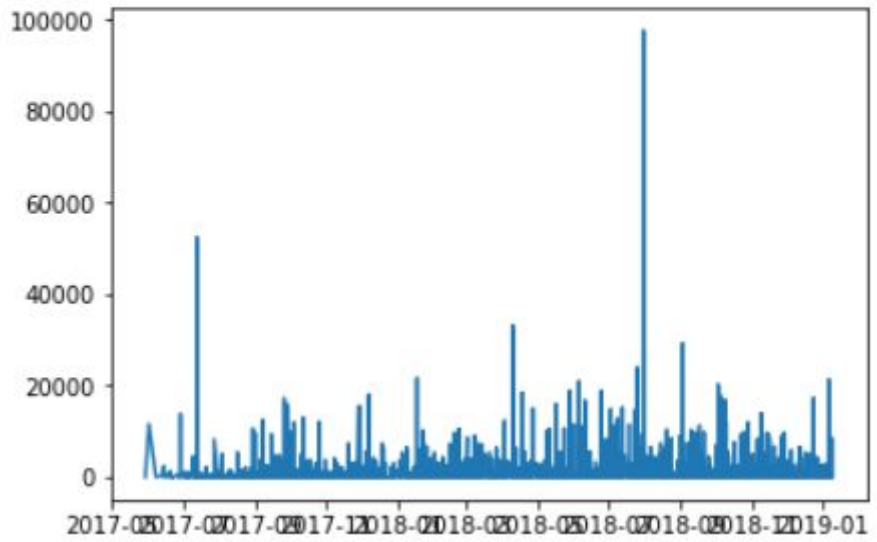

```
q75, q25 = np.percentile(train["WEIGHT"], [75, 25])
iqr = q75 - q25

minimum = q25 - (iqr * 1.5)
maximum = q75 + (iqr * 1.5)

print("Minimum = %.2f" % minimum)
print("Maximim = %.2f" % maximum)
```

```
plt.figure()
plt.boxplot(train["WEIGHT"][(train["WEIGHT"] >= minimum) & (train["WEIGHT"] <= maximum)])
plt.ylabel("WEIGHT")
plt.show()
```



```
plt.plot(train["DATEENTERED"], train["WEIGHT"])
```

```
[<matplotlib.lines.Line2D at 0x1f5c0aa3eb8>]
```




```
periods = [7,15,30,90]

fig = plt.figure(figsize=(20,10))

for n in periods:
    col = "MA" + str(n)
    train[col] = train["WEIGHT"].rolling(window=n).mean()

ax1 = fig.add_subplot(411)
ax1.plot(train["DATEENTERED"], train["MA7"])

ax2 = fig.add_subplot(412)
ax2.plot(train["DATEENTERED"], train["MA15"])

ax3 = fig.add_subplot(413)
ax3.plot(train["DATEENTERED"], train["MA30"])

ax4 = fig.add_subplot(414)
ax4.plot(train["DATEENTERED"], train["MA90"])
```


```
for col in ["MA7","MA15","MA30","MA90"]:
    train[col].fillna(train["WEIGHT"].mean(), inplace=True)
```

```
cols = [c for c in train.columns]
print('Number of features: {}'.format(len(cols)))

print('Feature types:')
train[cols].dtypes.value_counts()
```

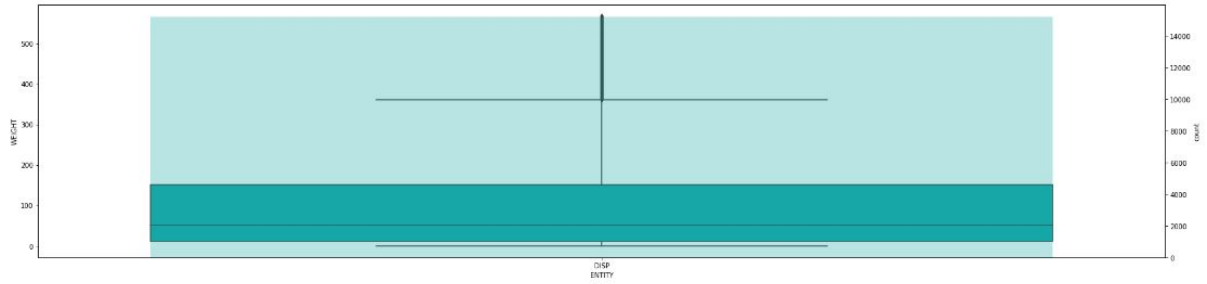
```
Number of features: 12
Feature types:

object     6
float64    5
int64      1
dtype: int64
```

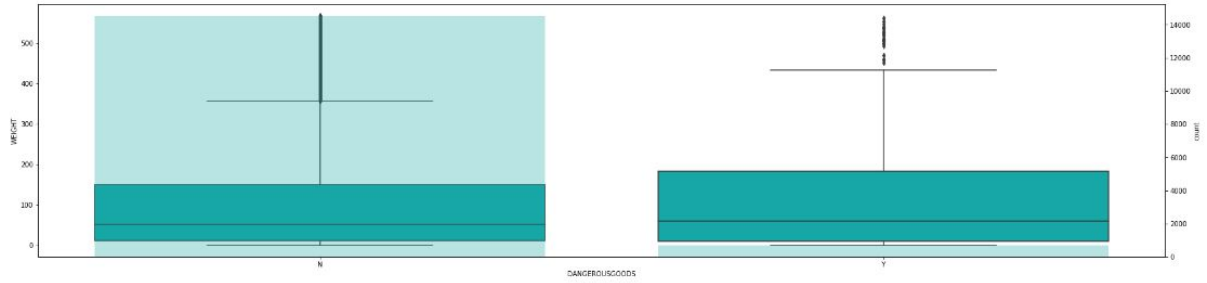
```
counts = [[], [], []]
for c in cols:
    typ = train[c].dtype
    uniq = len(np.unique(train[c]))
    if uniq == 1: counts[0].append(c)
    elif uniq == 2 and typ == np.int64: counts[1].append(c)
    else: counts[2].append(c)

print('Constant features: {}\nBinary features: {}\nCategorical features: {}\n'.format(*[len(c) for c in counts]))

print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

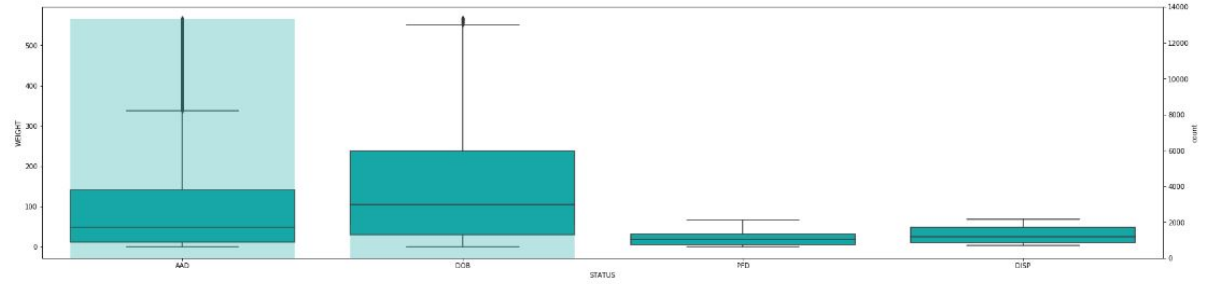
```
Constant features: 1
Binary features: 0
Categorical features: 11

Constant features: ['ENTITY']
Categorical features: ['DATEENTERED', 'DANGEROUSGOODS', 'CLIENTID', 'STATUS', 'ORIGIN', 'DESTINATION', 'WEIGHT', 'M
'MA30', 'MA90']
```

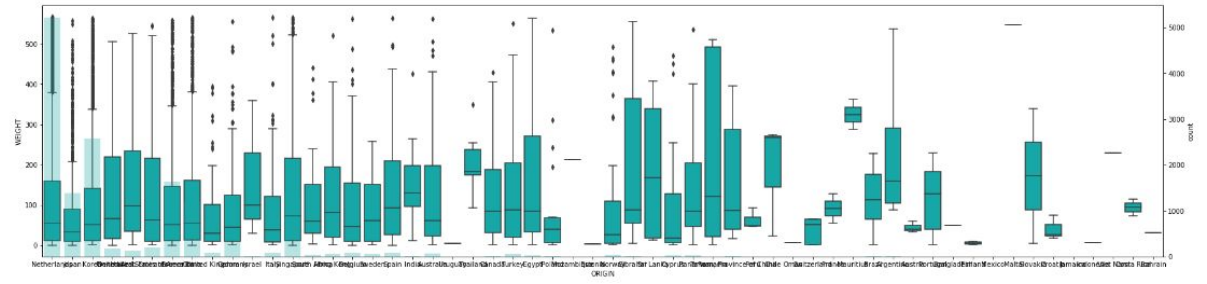
```
cat_feat =  ['ENTITY','DANGEROUSGOODS', 'STATUS', 'ORIGIN', 'DESTINATION']
```

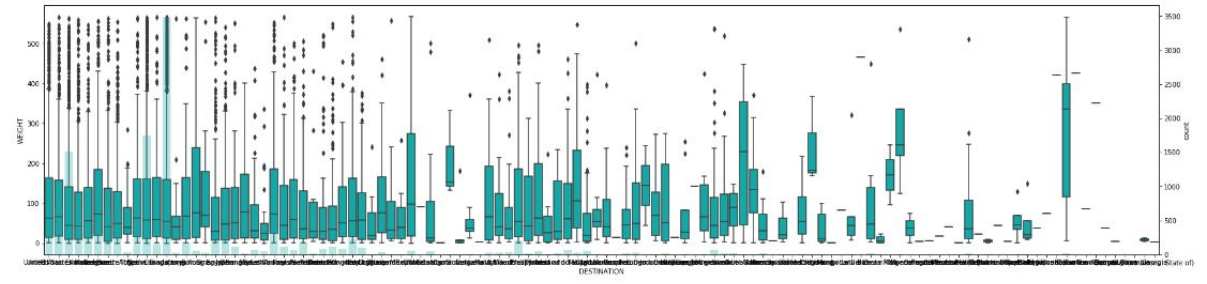
```
fig, ax = plt.subplots(5, 1, figsize=(30,40))
for c in cat_feat:
    axis = ax[cat_feat.index(c)]
    ax2 = axis.twinx()
    sns.boxplot(x=train[c], y=train["WEIGHT"][(train["WEIGHT"] >= minimum) & (train["WEIGHT"] <= maximum)], color="c", ax=axis)
    sns.countplot(x=train[c], alpha=0.3, color="c", ax=ax2)
```











```
train.drop(["MA7", "MA15", "MA30", "MA90"],axis=1, inplace=True)
```

```
def dummify(df, columns, drop=True):
    for column in columns:
        df_dummies = pd.get_dummies(df[column], prefix=column)
        df = pd.concat([df,df_dummies], axis=1)
        if drop == True:
            df.drop([column], inplace=True, axis=1)

    return df


def add_missing_dummy_columns(df, columns):
    missing_cols = set(columns) - set(df.columns)
    for c in missing_cols:
        df[c] = 0
```

```
old_col_train = list(train.drop(cat_feat, axis=1).columns)
old_col_test = list(test.drop(cat_feat, axis=1).columns)
```

```
train = dummify(train, cat_feat, True)
test = dummify(test, cat_feat, True)
```

```
new_col_train = [c for c in list(train.columns) if c not in old_col_train]
new_col_test = [c for c in list(test.columns) if c not in old_col_test]
```

```
add_missing_dummy_columns(test, new_col_train)
add_missing_dummy_columns(train, new_col_test)
```

```
test2=test.filter(items=['DATEENTERED','WEIGHT'])

test.drop(['WEIGHT'], inplace=True, axis=1)

print("Train: {}".format(train.shape))
print("Test: {}".format(test.shape))
```

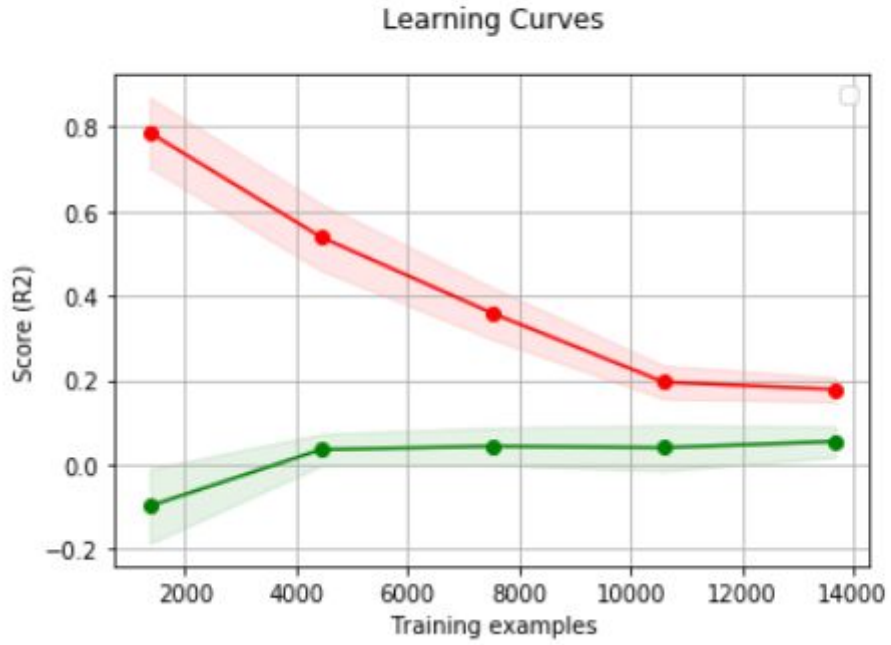
```
Train: (15198, 195)
Test: (3800, 194)
```

```
model = GradientBoostingRegressor()
X = train.drop(["DATEENTERED","WEIGHT"], axis=1)
y = train["WEIGHT"]
```

```python
def plot_learning_curves(estimator, X, y, scoring="accuracy", cv=None, n_jobs=1, train_sizes=np.linspace(0.1,1.0,5)):
    plt.figure()
    plt.title("Learning Curves\n")
    plt.xlabel("Training examples")
    plt.ylabel("Score ({})".format(scoring))
    plt.legend(loc="best")
    plt.grid()

    train_sizes, train_scores, test_scores = learning_curve(estimator, X, y, cv=cv, n_jobs=n_jobs, train_sizes=train_sizes)

    train_scores_mean = np.mean(train_scores, axis=1)
    train_scores_std = np.std(train_scores, axis=1)
    test_scores_mean = np.mean(test_scores, axis=1)
    test_scores_std = np.std(test_scores, axis=1)

    plt.fill_between(train_sizes, train_scores_mean - train_scores_std,
                     train_scores_mean + train_scores_std, alpha=0.1,
                     color="r")

    plt.fill_between(train_sizes, test_scores_mean - test_scores_std,
                     test_scores_mean + test_scores_std, alpha=0.1,
                     color="g")

    plt.plot(train_sizes, train_scores_mean, "o-", color="r",
             label="Training score")

    plt.plot(train_sizes, test_scores_mean, "o-", color="g",
            label="Cross-validation score")

    plt.show()
```

```python
plot_learning_curves(model, X, y, scoring="R2", cv=10, n_jobs=4)
```

```
No handles with labels found to put in legend.
```



```
model.fit(X, y)
```

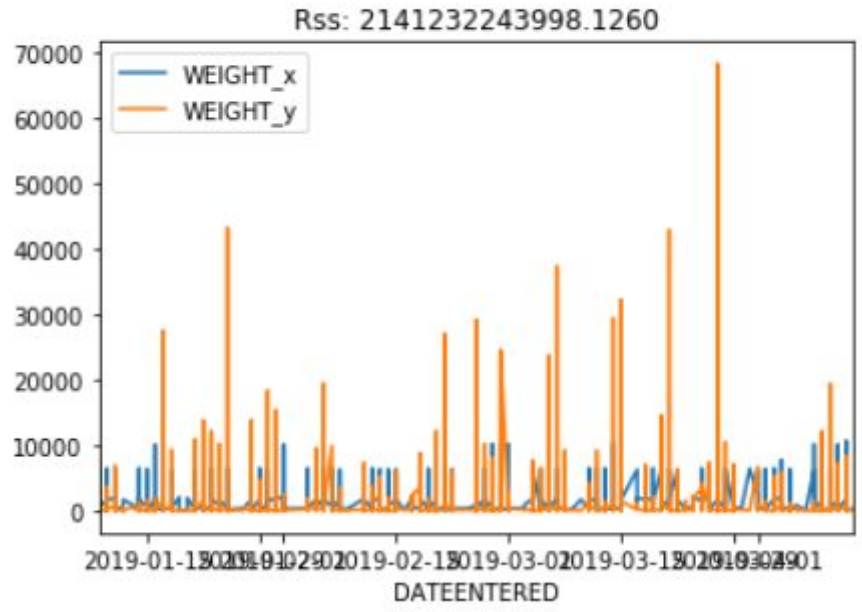
```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, n_iter_no_change=None, presort='auto',
             random_state=None, subsample=1.0, tol=0.0001,
             validation_fraction=0.1, verbose=0, warm_start=False)
```

```
y_pred = model.predict(test.drop("DATEENTERED", axis=1))
```

```
y_pred = model.predict(test.drop("DATEENTERED", axis=1))

submission = pd.DataFrame()
submission["DATEENTERED"] = test["DATEENTERED"].values
submission["WEIGHT"] = y_pred
```

```
compare = pd.merge(submission , test2, left_on=['DATEENTERED'], right_on=['DATEENTERED'])

compare.plot(x="DATEENTERED", y=["WEIGHT_x", "WEIGHT_y"])
c=(compare.WEIGHT_x-compare.WEIGHT_y)
plt.title('Rss: %.4f'% sum((c)**2))
```

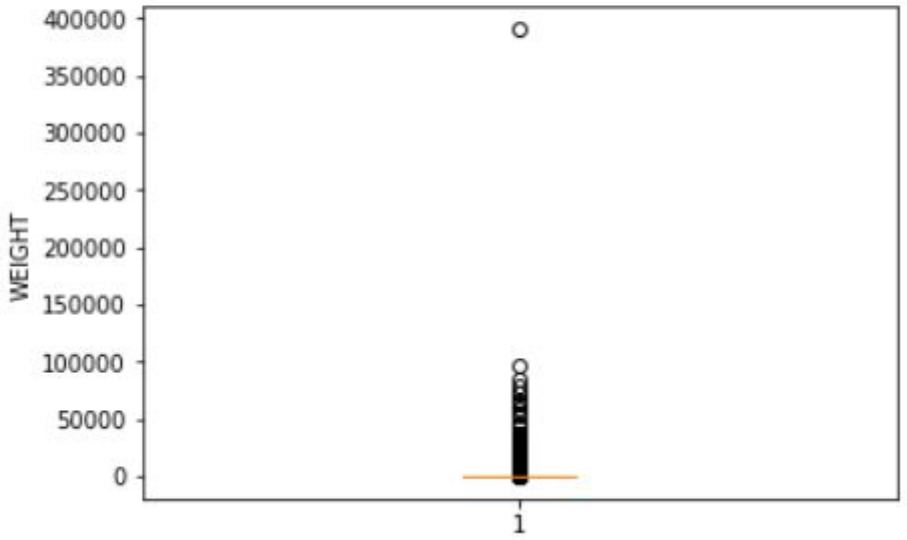
```
Text(0.5, 1.0, 'Rss: 2141232243998.1260')
```



Gradient Boosting (index as id, weight as target)

‘Ιδια μεθοδολογία ακολουθείται όπως και στις προηγούμενες εφαρμογές. Αφού φιλτράρουμε τα δεδομένα με βάση το dispatch και χρησιμοποιήσουμε το index σαν id, προχωράμε στην εφαρμογή του μοντέλου.

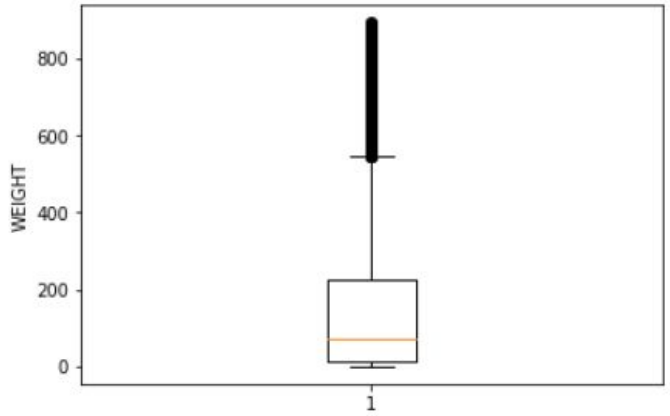
```
plt.figure()
plt.boxplot(train["WEIGHT"])
plt.ylabel("WEIGHT")
plt.show()
```




```
q75, q25 = np.percentile(train["WEIGHT"], [75, 25])
iqr = q75 - q25

minimum = q25 - (iqr * 1.5)
maximum = q75 + (iqr * 1.5)

print("Minimum = %.2f" % minimum)
print("Maximim = %.2f" % maximum)
```

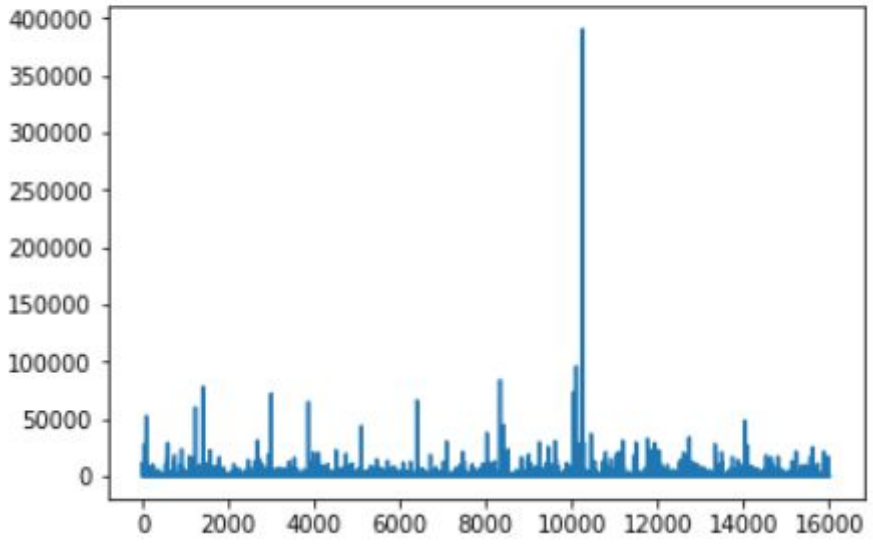
```
Minimum = -505.00
Maximim = 895.00
```

```
plt.figure()
plt.boxplot(train["WEIGHT"][(train["WEIGHT"] >= minimum) & (train["WEIGHT"] <= maximum)])
plt.ylabel("WEIGHT")
plt.show()
```

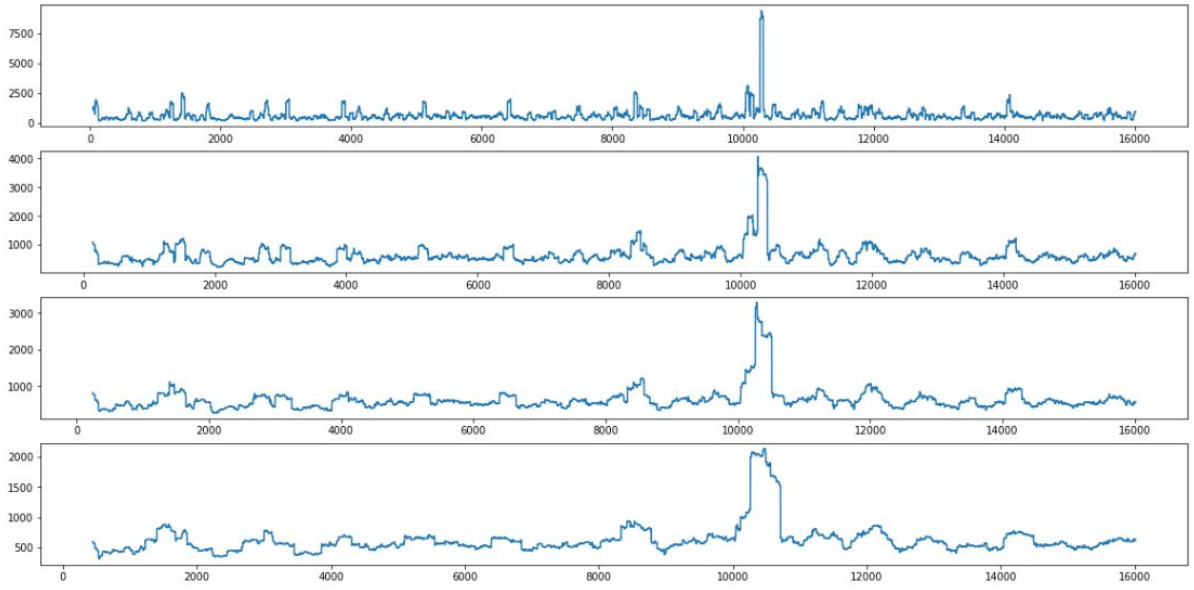



```
plt.plot(train["index"], train["WEIGHT"])
```

```
[<matplotlib.lines.Line2D at 0x1e71846b048>]
```



```
periods = [50,150,250,450]

fig = plt.figure(figsize=(20,10))

for n in periods:
    col = "MA" + str(n)
    train[col] = train["WEIGHT"].rolling(window=n).mean()

ax1 = fig.add_subplot(411)
ax1.plot(train["index"], train["MA50"])

ax2 = fig.add_subplot(412)
ax2.plot(train["index"], train["MA150"])

ax3 = fig.add_subplot(413)
ax3.plot(train["index"], train["MA250"])

ax4 = fig.add_subplot(414)
ax4.plot(train["index"], train["MA450"])
```



```
for col in ["MA50","MA150","MA250","MA450"]:
    train[col].fillna(train["WEIGHT"].mean(), inplace=True)
```

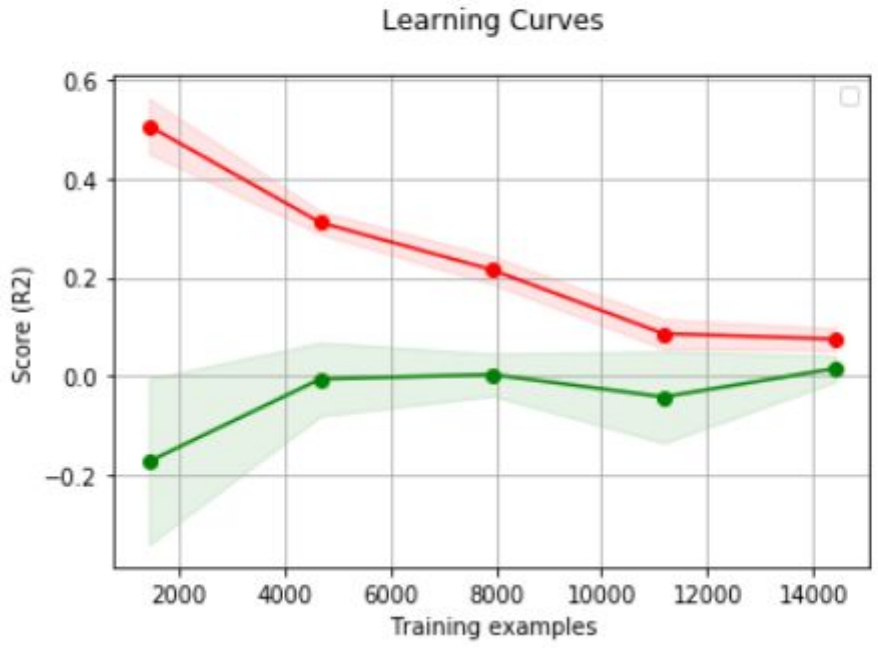
```
cols = [c for c in train.columns]
print('Number of features: {}'.format(len(cols)))

print('Feature types:')
train[cols].dtypes.value_counts()
```

```
Number of features: 12
Feature types:

object     5
float64    5
int64      2
dtype: int64
```

Ομοίως με πρίν κάνουμε dummify τις στήλες που περιέχουν κατηγορικές τιμές.

```
plot_learning_curves(model, X, y, scoring="R2", cv=10, n_jobs=4)
```

```
No handles with labels found to put in legend.
```

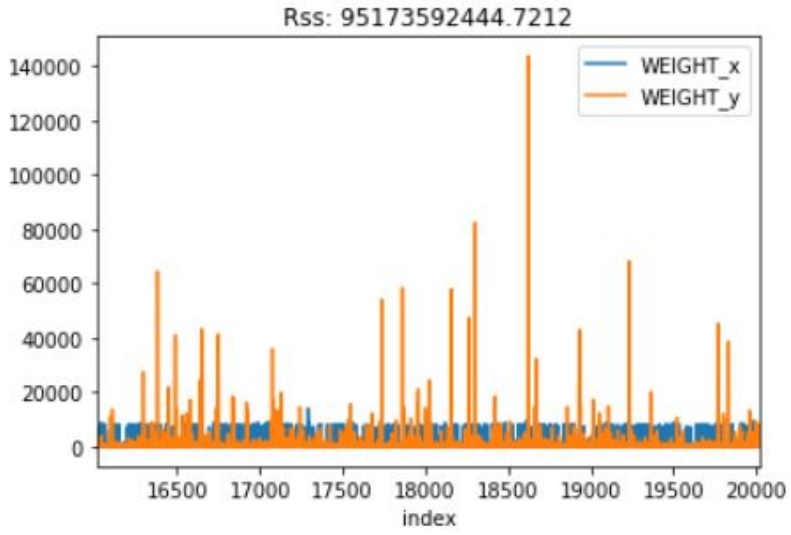


```
model.fit(X, y)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
             min_impurity_split=None, min_samples_leaf=1,
             min_samples_split=2, min_weight_fraction_leaf=0.0,
             n_estimators=100, n_iter_no_change=None, presort='auto',
             random_state=None, subsample=1.0, tol=0.0001,
             validation_fraction=0.1, verbose=0, warm_start=False)
```

```
compare = pd.merge(submission , test2, left_on=['index'], right_on=['index'])

compare.plot(x="index", y=["WEIGHT_x", "WEIGHT_y"])
c=(compare.WEIGHT_x-compare.WEIGHT_y)
plt.title('Rss: %.4f'% sum((c)**2))
```

```
Text(0.5, 1.0, 'Rss: 95173592444.7212')
```



Κεφάλαιο 6- Συμπεράσματα

Γνωρίζοντας ότι στα δεδομένα μου το βάρος ενός shipment είναι άμεσα συσχετιζόμενο με το κόστος, το modality καθώς και το buy και sell, καταλήγουμε στο ότι δεν μπορούμε να εφαρμόσουμε αποδοτικά μοντέλα πρόβλεψης ως προς το weight με τα δεδομένα ως έχουν, πράγμα που φαίνεται και από τα αποτελέσματα των παραπάνω μοντέλων.

Πιο αναλυτικά:

arima	target: Profit		target: weight	
test p-value lags used	-5.70 7.50 7	p>0.05 non-stationary	0.00	Stationary
logscale - moving average test p-value lags used	 -1.24 3.08 4	 p>0.05 non-stationary		
logscale - exponential decay weighted average test p-value lags used	 -8.26 4.95 9	 p>0.05 non-stationary		
shifted values test p-value lags used	 -1.05 1.12 6	 p>0.05 non-stationary		
seasonal decompose test p-value lags used	 -1.07 4.3 1.9	 p>0.05 non-stationary		
AutoRegression (**AR**): Rss	1424.66		2284.31	
Moving Average (**MA**): Rss	2126.28			
ARIMA: Rss	1450.54			

Exponential Smoothing	target: Profit	target: Profit
Simple Smoothing rms	12323.67	15646.95

Το μοντέλο με τη χειρότερη απόδοση φαίνεται να είναι το Gradient boosting πράγμα λογικό καθώς μετά τον dumification των κατηγορικών μεταβλητών ο αριθμός των στηλών άρα και μεταβλητων που λαμβάνει υπόψη το μοντέλο μεγαλώνει δραματικά. Ίσως σε ένα πολύ μεγαλύτερο dataset να μπορούσε να έχει καλύτερη απόδοση.

Gradient boosting	date-profit	id-profit	date-weight	id-weight
nr of columns before dummification	8	8	8	8
nr of columns after dummification	219	219	195	195
rss	2551461964284.63	19101172495.24	2141232243998.12	95173592444.72

Από τα παραπάνω συμπεραίνουμε ότι μοντέλα χρονοσειρών έχουν πολύ καλύτερη απόδοση στο συγκεκριμένο πείραμα με το πρόβλεψη να παρουσιάζει μια σχετική ανοδική τάση αλλά την ίδια τη χρονοσειρά να είναι σχετικά στατική. Η πρόβλεψη έχει καλύτερα αποτελέσματα με το weight σαν target σε κάθε περίπτωση, με την ARIMA να βγάζει καλύτερα αποτελέσματα σε σχέση με το Exponential Smoothing.

Βιβλιογραφία

[1] Andres Munoz, “Machine Learning and Optimization”, *Courant Institute of Mathematical Sciences*, New York, NY

[2] Anoop Vasant Kumar, “What is the difference between Support Vector Machine and Support Vector Regression?”, 2016

[3] Anup Bhande, “What is underfitting and overfitting in machine learning and how to deal with it.”, *Greatom*, Mar 11, 2018

[4] A.D.Dongare, R.R.Kharde, Amit D.Kachare, ”Introduction to Artificial Neural Network”, *International Journal of Engineering and Innovative Technology (IJEIT), Volume 2, Issue 1*, July 2012

[5] C. E. Rasmussen, C. K. I. Williams, “Gaussian Processes for Machine Learning”, *The MIT Press*, 2006

[6] Chris Thornton, Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, "Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms", *Department of Computer Science, University of British Columbia*

[7] Cory Maklin, “Linear Discriminant Analysis In Python”

[8] Dr. Sathiya, “Support Vector Machine”, *Mitosistech*, Mar 25, 2019

[9] Gunseerat Kaur Brar, "Comparison of Machine learning algorithms in Anomaly detection”, *Central University of Punjab*, 2018

[10] Haadican, “How to Build a Simple Artificial Neural Network (ANN)”, *Medium,* Mar 28, 2019

[11] Hossein Hassani, Xu Huang, Emmanuel S. Silva, Mansi Ghodsi, “A Review of Data Mining Applicationsin Crime”

[12] Ilan Reinstein, “Random Forests, Explained”, *KDnuggets*

[13] Imad Dabbura, “Coding Neural Network — Regularization”, *Towards Data Science*, May 8, 2018

[14] James Furbush, “Machine learning: A quick and simple definition. Get a basic overview of machine learning and then go deeper with recommended resources”

[15] Jayesh Bapu Ahire, “The Artificial Neural Networks handbook: Part 1”, *Medium*, Aug 24, 2018

[16] Jason Brownlee, “A Gentle Introduction to Exponential Smoothing for Time Series Forecasting in Python”

[17] Jason Brownlee, “How to Configure the Number of Layers and Nodes in a Neural Network”, *Data Science Central*, Feb 17, 2019

[18] Kenneth Chu, Claude Poirier, “Machine Learning Documentation Initiative", *Workshop on the Modernisation of Statistical Production Meeting*, 2015

[19] L´eon Bottou, Chih-Jen Lin, “Support Vector Machine Solvers”

[20] Lakshmi Devasena C, “Comparative Analysis of Random Forest, REP Tree and J48 Classifiers for Credit Risk Prediction”, 2014

[21] M. Bremer, “Supplement 5 - multiple regression”, 2012

[22] M. W. Gardnera, S. R. Dorlinga, “Artificial neural networks (the multilayer perceptron) - a review of the applications in the atmospherice sciences”

[23] Mark A. Hall, “Correlation-based feature selection for machine learning”

[24] Mustapha Belouch, Salah El Hadaj, Mohamed Idhammad, “A Two-Stage Classifier Approach using RepTree Algorithm for Network Intrusion Detection”, *International Journal of Advanced Computer Science and Applications*, 2017
[25] Nicolas Vandeput, “Exponential Smoothing”
[26] Niels Landwehr, Mark Hall, Eibe Frank, “Logistic Model Trees”
[27] Niruhan Viswarupan, “K-Means Data Clustering”, *Towards Data Science*, Jul 10, 2017
[28] Paul Paisitkriangkrai, “Linear Regression and Support Vector Regression”, *The University of Adelaide*, October 2012
[29] Pumendu Das, “Naive Bayes Algorithm in Python”, *Codespeedy*
[30] Rohith Gandhi, “Naive Bayes Classifier”, *Towards Data Science*, May 5, 2018
[31] Saimadhu Polamuri, “How the Random Forest Algorithm works in Machine learning”, 2017
[32] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, Somesh Jha, “Privacy Risk in Machine Learning: Analyzing the Connection to Overfitting”
[33] Saul McLeod, “What a p-value tells you about statistical significance”, *Simply psychology*, 2019
[34] Sebastian Raschka, “Linear Discriminant Analysis – Bit by Bit”, Aug 3, 2014
[35] Selva Prabhakaran, “ARIMA Model – Complete Guide to Time Series Forecasting in Python”
[36] Siva S. Sivatha Sindhu, S. Geetha, A. Kannan, “Decision tree based light weight intrusion detection using a wrapper approach”
[37] Sushilkuma Kalmegh, “Analysis of WEKA Data Mining Algorithm REPTree, Simple Cart and RandomTree for Classification of Indian News”, *International Journal of Innovative Science, Engineering & Technology*, 2015
[38] Trevor Hastie, Robert Tibshirani, Jerome Friedman, “The elements of Statistical Learning: Data Mining, Inference, and Prediction”
[39] Unnati R. Raval, Chaita Jani, “Implementing & Improvisation of K-means Clustering Algorithm”, *IJCSMC, Vol. 5, Issue. 5*, May 2016
[40] Will Kenton, ”Logistics”, *Investopedia*
[41] W. Nor Haizan W. Mohamed, Mohd Najib Mohd Salleh, Abdul Halim Omar, “A Comparative Study of Reduced Error Pruning Method in Decision Tree Algorithms”, *2012 IEEE International Conference on Control System, Computing and Engineering*, Nov 23 - 25, 2012
[42] “A.I. Wiki: A Beginner's Guide to Neural Networks and Deep Learning”, *Pathmind*
[43] “Classification Algorithms - Random Forest”, *Tutorials Point*
[44] “Everything You Need to Know About Artificial Neural Networks”, *Medium*
[45] “Forecasting”, *Investopedia*
[46] “How do Neural Networks update weights and Biases during Back Propagation?”, *i2tutorials*, Sept 7 ,2019
[47] “Illustration of the statistical analysis using multiple linear regression”, *European Environmental Agency,* Sept 26, 2011
[48] “Implement sigmoid function using Numpy”, *Geeks for Geeks*
[49] “Introduction to exponential Smoothing for Time Series Forecasting using Python”, *Uncoolai*
[50] “Machine Learning”, *Deep AI*

[51] “Simple linear regression”, *Statstutor*
[52] “Steps in choosing a forecasting model”, *Duke University*
[53] “Supply chain”, *Collins Dictionary*
[54] “Supply Chain Management – For a steady flow and an increase in productivity!”, *The Supply Chain People*
[55] “What is underfitting and overfitting in machine learning and how to deal with it”, *Greyatom*