ΕΠΛ 448: Εξόρυξη δεδομένων στον Παγκόσμιο Ιστό

Project Milestones:

- 1η Φάση: Ορισμός προβλήματος και Εξερεύνηση Δεδομένων
- 2η Φάση: Εκπαίδευση/Μάθηση αλγόριθμων (learning/training) και πρόβλεψη (prediction)
- 3η Φάση: Αξιολόγηση αλγορίθμου (testing)

1η Φάση: Ορισμός προβλήματος και Εξερεύνηση Δεδομένων:

Καταρχήν έπρεπε να καταλάβουμε το πρόβλημα. Αυτό που έπρεπε να κάνουμε ήταν να προβλέψουμε πόσο θα κοστίζει ένα εισιτήριο ταξιδιού με αεροπλάνο. Άρα, μιλάμε για supervised learning καθώς τα δεδομένα μας είναι labeled δηλαδή ξέρουμε την έξοδο και επίσης είναι regression problem αφού η έξοδος που θέλουμε αφορά ένα αριθμό (τιμή εισιτηρίου). Επίσης, γι' αυτήν την φάση, είχαμε στόχο να καταλάβουμε τα δεδομένα που συλλέξαμε από την ιστοσελίδα και μετά να τα φέρουμε σε μορφή που να μπορεί ένας αλγόριθμος μάθησης (supervised και regression) να εκπαιδευτεί.

Όσο αφορά το πρώτο βήμα, είδαμε ότι είχαμε ένα dataset το οποίο ήταν διαστάσεων 10683x11. Δηλαδή, είχε 10683 εγγραφές (εισιτήρια) και 11 στήλες, από τις οποίες τα 10 ήταν features ενός εισιτηρίου ενώ η τελευταία ήταν η τιμή. Αυτά τα features ήταν τα εξής:

- Airline: αερογραμμή που αφορούσε το συγκεκριμένο εισιτήριο
- Date_of_Journey: ημερομηνία αναχώρησης σε μορφή DD/MM/YYYY
- Source: πόλη αναχώρησης
- Destination: πόλη άφιξης
- Route: διαδρομή που θα ακολουθήσει το αεροπλάνο (περιέχει τουλάχιστον την πόλη αναχώρησης και άφιξης καθώς και άλλες πόλεις εφόσον υπάρχουν στάσεις)
- Dep_Time: ώρα αναχώρησης
- Arrival_Time: ημερομηνία και ώρα άφιξης
- Duration: διάρκεια πτήσης
- Total_Stops: πόσες στάσεις μέχρι τον προορισμό
- Additional_info: χρήσιμες πληροφορίες για την πτήση π.χ. αν είναι economy class, αν περιέχει φαγητό και άλλα

Προβλήματα που εντοπίστηκαν μόνο και μόνο από την παρατήρηση του dataset ήταν ότι το feature Arrival_ Time ήταν ημιτελής περιέχοντας μια από τις δύο πληροφορίες ενώ αρκετές φορές ήταν και λάθος (Λόγω του ότι δεν αντιστοιχούσε με το dep_time + duration). Ακόμα, μόνο δύο τιμές σε ολόκληρο το dataset έλειπαν και αυτές βρίσκονταν στην ίδια γραμμή. Οπότε δεν είχαμε επιπλέον missing values.

Έτσι προχωρήσαμε στο δεύτερο βήμα, όπου πήραμε όλα τα δεδομένα του dataset, τα διορθώσαμε εφόσον ήταν λανθασμένα και τα φέραμε στην μορφή που τα θέλαμε. Πιο συγκεκριμένα τα βήματα που ακολουθήσαμε για να προετοιμάσουμε τα δεδομένα μας ήταν:

- **Data Selection:** σε αυτό το κομμάτι ελέγξαμε ποια δεδομένα, και πιο συγκεκριμένα ποια features, όντως χρειαζόμασταν έτσι ώστε να διαγράψουμε ότι ήταν περιττό.

Έτσι θεωρήσαμε σωστό να διαγράψουμε το feature Arrival_Time αφού αρχικά υπήρχαν αρκετές καταχωρήσεις λανθασμένες (και δεν οφειλόταν στην διαφορά ώρας-time zone) και κατά δεύτερο μπορούσαμε να το υπολογίσουμε συνδυάζοντας τα features Date_of_Journey, Dep_Time και Duration. Επίσης, διαγράφηκε όλη η γραμμή στην οποίαν έλειπαν δύο τιμές, αφού ένα εισιτήριο ανάμεσα στις 10683 εγγραφές ήταν αμελητέο.
Κάναμε δύο δοκιμές όσον αφορά την στήλη route. Αρχικά δοκιμάσαμε να αφαιρέσουμε την στήλη και να εξαγάγουμε αποτελέσματα, και έπειτα δοκιμάσαμε εμπλουτίζοντας το dataset μας με την στήλη αυτή. Θα αναλύσουμε την διαδικασία χωρίς την στήλη route, αλλά η διαδικασία που ακολουθήθηκε και στις δύο περιπτώσεις ήταν ίδια. Θα ξεχωρίσουμε τα αποτελέσματα των δύο δοκιμών στην 3η φάση όπου θα μιλήσουμε για τα αποτελέσματα των αλγορίθμων.

- **Data Preprocessing:** σε αυτό το κομμάτι έπρεπε να οργανώσουμε τα δεδομένα που πήραμε, να τα καθαρίσουμε και να τα κάνουμε encode.
  Πιο συγκεκριμένα, ακολουθήσαμε την πιο κάτω διαδικασία για κάθε feature:

  - Airline: οι τιμές στο συγκεκριμένο feature έδειχναν την αερογραμμή για το συγκεκριμένο εισιτήριο όπως για παράδειγμα “Indigo”, “Jet Airways” και άλλα. Έτσι, εφαρμόσαμε encoding καθώς για να λειτουργήσουν οι machine learning αλγόριθμοι πρέπει όλα να είναι νούμερα. Πιο συγκεκριμένα, εφαρμόσαμε Dummy Encoding παράγοντας για κάθε τιμή μια στήλη εκτός από μια. Πλέον έχουμε binary τιμές αυτών των στηλών. Διαλέξαμε αυτό τον τρόπο encoding, καθώς δεν θέλαμε να δώσουμε βάρος στις αερογραμμές (όπως Ordinal Encoding) και επίσης είναι μια βελτιστοποίηση του One-Hot Encoding σε θέμα μνήμης.
  - Date_of_Journey: στην συγκεκριμένη στήλη μπορούσαμε να δούμε την ημερομηνία αναχώρησης της πτήσης σε μορφή DD/MM/YYYY. Το preprocessing έγινε σε δύο στάδια. Στο πρώτο στάδιο, φέραμε όλες τις τιμές της στήλης στη μορφή που πρέπει, να έχει δηλαδή μόνο ακεραίους χωρίς ειδικούς χαρακτήρες. Στο δεύτερο στάδιο, εφαρμόσαμε Cyclical Feature Encoding τόσο στο μήνα όσο και στην μέρα της εβδομάδας (δηλ. Δευτέρα, Τρίτη κτλ. και όχι 14,15,16…).
    Με αυτό τον τρόπο, καταφέραμε να φέρουμε κοντά πτήσεις που έγιναν είτε σε κοντινές μέρες (Για παράδειγμα η Κυριακή είναι δίπλα με την Δευτέρα) είτε σε κοντινούς μήνες αφού ο χρόνος είναι σημαντικός παράγοντας. Και στις δύο περιπτώσεις (μέρες και μήνες) θέλουμε να αποδώσουμε αυτήν την κυκλική συμπεριφορά. Για παράδειγμα ο Δεκέμβρης είναι κοντά στον Ιανουάριο και οι τιμές των πτήσεων είναι υψηλές.

  - Source και Destination: οι τιμές στο συγκεκριμένο feature έδειχναν την πόλη αναχώρησης και άφιξης για το συγκεκριμένο εισιτήριο όπως για παράδειγμα “Banglore”, “New Delhi” και άλλα. Έτσι, εφαρμόσαμε

και εδώ Dummy Encoding ξεχωριστά για πόλη αναχώρησης και για πόλη άφιξης παράγοντας για κάθε τιμή μια στήλη (εκτός την τελευταία) και έχοντας binary τιμές αυτών των στηλών αντίστοιχα.

- Dep_Time: στην συγκεκριμένη στήλη μπορούσαμε να δούμε την ώρα αναχώρησης της πτήσης. Το preprocessing έγινε και εδώ σε δύο στάδια. Στο πρώτο στάδιο, αφαιρέσαμε όλους τους ειδικούς χαρακτήρες και τους μετατρέψαμε σε δεκαδικούς από το 00.00 μέχρι το 23.59 . Στο δεύτερο στάδιο, εφαρμόσαμε Cyclical Feature Encoding. Με αυτό τον τρόπο, καταφέραμε να φέρουμε κοντά πτήσεις που έγιναν σε κοντινές ώρες.

- Duration: η διάρκεια περιγραφόταν σε ώρες και λεπτά και για να ξεχωρίζουν αυτά υπήρχαν οι χαρακτήρες “h” και “m”. Έτσι, αυτό που κάναμε ήταν να τα αντικαταστήσουμε όλα με έναν ακέραιο αριθμό που αντιπροσώπευε την διάρκεια σε λεπτά.

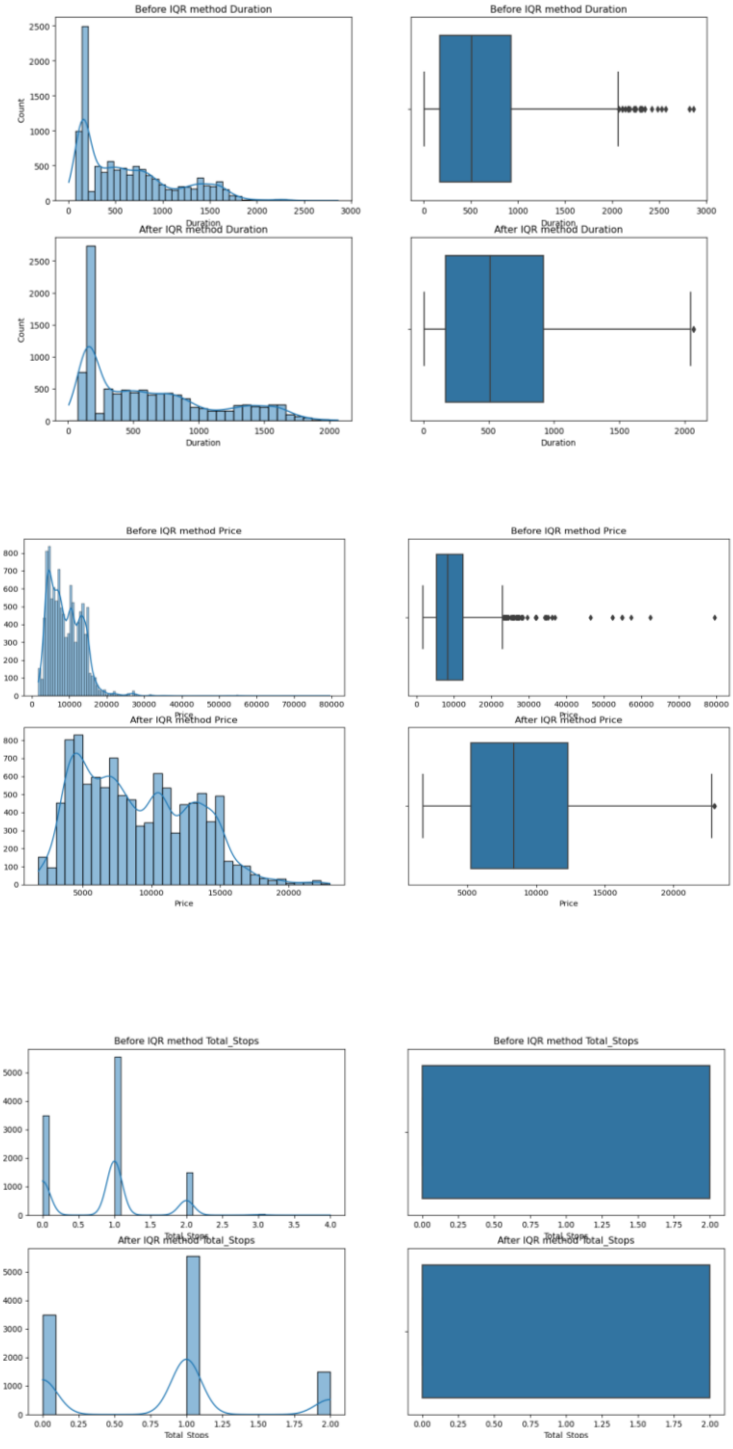
- Total_Stops: υπήρχαν και αριθμοί και λέξεις όπως για παράδειγμα “non-stop” ή “1 stop”. Έτσι, αυτό που κάναμε ήταν να βάλουμε απλά έναν αριθμό που να αντιπροσωπεύει τις στάσεις της κάθε πτήσης.

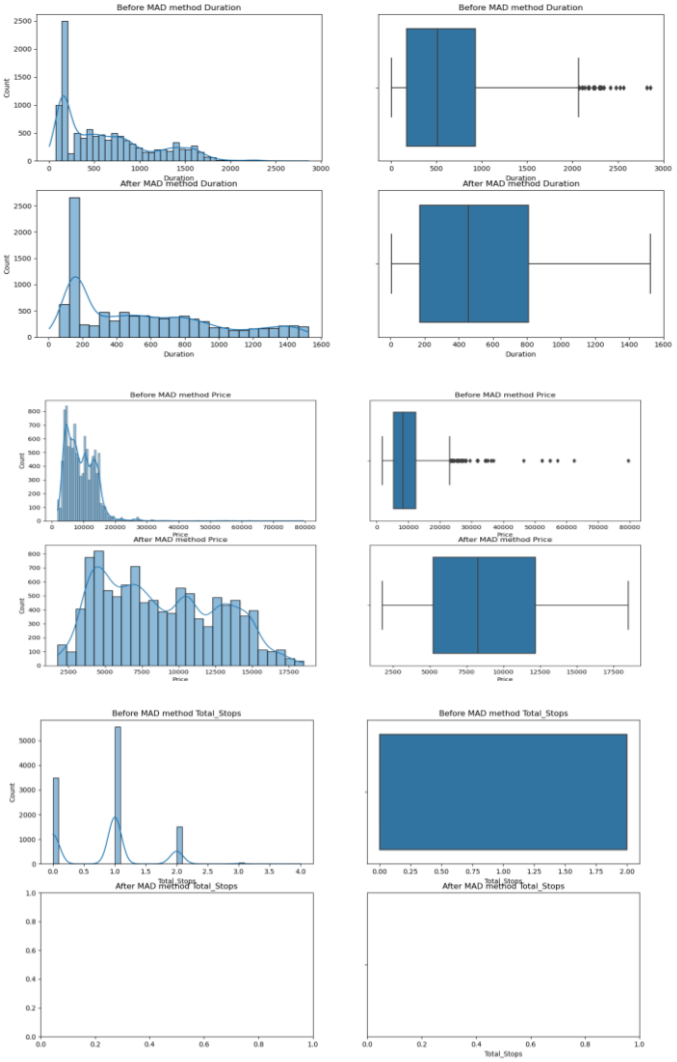
- Additional_info: οι τιμές στο συγκεκριμένο feature περιγράφονταν με κάποιες κατηγορίες όπως economy_class, food_included και άλλα δείχνοντας έτσι κάποιες επιπρόσθετες πληροφορίες για την συγκεκριμένη πτήση. Έτσι, εφαρμόσαμε encoding και πιο συγκεκριμένα Ordinal Encoding δίνοντας σημασία στην σειρά. Κάναμε ένα mapping αυτών των δεδομένων με τέτοιο τρόπο που ο αριθμός να δείχνει σε πόσο βαθμό επηρεάζει η συγκεκριμένη κατηγορία την τελική τιμή ξεκινώντας από το 0 ως αυτό που επηρεάζει την τιμή στον ελάχιστο βαθμό μέχρι και το 8 ως αυτό που ανεβάζει στο μέγιστο την τιμή
(0 → No food,
1 → in-flight meal not included,
2 → no check in baggage included,
3 → red-eye flight,
4 → 1 long layover,
5 → 1 short layover,
6 → 2 long layovers,
7 → change airports,
8 → business class).

- Route: την στήλη αυτή αρχικά την είχαμε διαγράψει αφού είχαμε τον αριθμό των στάσεων και πόλη αναχώρησης και άφιξης. Στην συνέχεια όμως αποφασίσαμε να δοκιμάσουμε να τρέξουμε τους αλγόριθμους μας και με αυτή την στήλη. Η στήλη αυτή περιείχε ακρώνυμα αεροδρομίων αφίξεων, προορισμών και γενικά στάσεων κατά την διάρκεια της πτήσης. Αρχικά η στήλη περιείχε βέλη τα οποία, δεν μπορούσαν να μηνούν στα δεδομένα μας. Έπειτα χρησιμοποιήσαμε την μέθοδο του TF-IDF score. Στην στήλη route είχαμε 43 μοναδικούς προορισμούς, έτσι δημιουργήσαμε 43 νέες στήλες, για κάθε ένα προορισμό που βρήκαμε. Με αυτό τον τρόπο εξαγάγαμε ένα score, για κάθε εμφάνιση της τοποθεσίας σε μια πτήση. Τα αποτελέσματα και για τις 2 περιπτώσεις παρουσιάζονται στην 3^{η} φάση.

Με το τέλος της φάσης αυτής (πρώιμο preprocessing) καταφέραμε να έχουμε τα δεδομένα μας σε αποδεκτή μορφή. Έτσι μελετήσαμε ξανά το dataset και τρέξαμε διάφορους αλγορίθμους με σκοπό να εντοπίσουμε outliers. Outliers εντοπίστηκαν στις στήλες “Duration”, “Total_Stops” και “Price”. Αυτό είχε ως

αποτέλεσμα, να εφαρμόσουμε για αυτές τις τρεις στήλες την μέθοδο IQR με σκοπό να εντοπίσει και να διαγράψει τους outliers. Συγκριτικά με την MAD μπορούσε να βρει τα outliers πολύ πιο αποδοτικά. Για παράδειγμα η Mad μέθοδος θεωρούσε αρκετές τιμές με πολύ μικρή απόκλιση από το median ως outliers. Σε χειριστή περίπτωση θεώρησε ένα ολόκληρο feature (total stops) ως outlier.



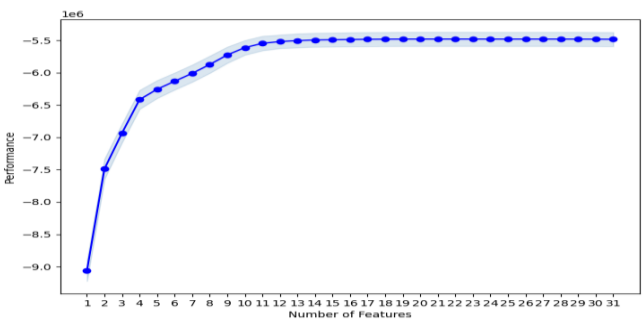


- **Data Transformation**: σε αυτό το βήμα σκοπός μας ήταν να μετατρέψουμε τα έτοιμα δεδομένα (preprocessed data) σε δεδομένα τα οποία θα μπορούσαν οι αλγόριθμοι machine learning να δουλέψουν μαζί τους.
  Με λίγα λόγια, θέλαμε να πάρουμε το dataset και να εφαρμόσουμε τεχνικές όπως Feature Selection και Feature Extraction για να μειώσουμε τις διαστάσεις δηλαδή τις στήλες που θα ήταν η είσοδος των αλγορίθμων. Με

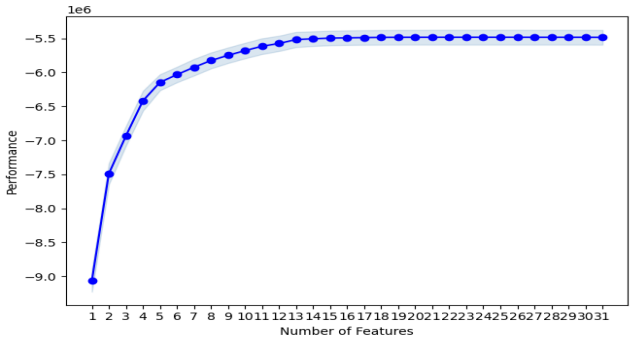
αυτό τον τρόπο θα χρειαζόμασταν πιο λίγο χρόνο για την εκπαίδευση των αλγορίθμων. Αλλά θα ήταν και πιο κατανοητά τα αποτελέσματα και θα αποτυπώνονταν πιο εύκολα σε μια γραφική. Από τις δύο τεχνικές διαλέξαμε την πρώτη, Feature Selection, καθώς οι στήλες δεν ήταν τόσες πολλές για να ακολουθήσουμε το Feature Extraction και να αρχίσουμε να παράγουμε νέες στήλες.
Στην αρχή, θέλαμε να καταλάβουμε τις στήλες και τι ρόλο παίζουν, δηλαδή πόσο "σημαντικές" είναι και πόσο συσχετισμένες είναι με το τελικό αποτέλεσμα. Για να πάρουμε μια γεύση της σημαντικότητας χρησιμοποιήσαμε δύο μεθόδους: Ensemble Regressors και Recursive Feature Elimination. Στην συνέχεια, δημιουργήσαμε και ένα heatmap (εικόνα 1.1) το οποίο έδειχνε τις συσχετίσεις μεταξύ των features.

Στη συνέχεια, αφού είχαμε μια πιο καθαρή εικόνα για τις στήλες που αντικατοπτρίζουν τα features, υλοποιήσαμε δύο τεχνικές. Σκοπός τους ήταν αφού πάρουν είσοδο όλες τις στήλες και το επιθυμητό αποτέλεσμα, να μας δώσουν σαν έξοδο το σύνολο των features το οποίο δίνει το καλύτερο αποτέλεσμα με βάση κάποιες μετρικές. Οι μετρικές αυτές ήταν το Mean Absolute Error και το Mean Squared Error. Οι δύο τεχνικές, που στο τέλος τις ημέρας έδωσαν παρόμοιες απαντήσεις με ελάχιστες διαφορές, ήταν Forward Selection και Backward Elimination.

SBS:



SFS:



Οι τεχνικές έτρεξαν με τους αλγόριθμους Lasso (διάφορες παραμέτρους) και Linear Regression. Είναι σημαντικό να πούμε ότι και στις δύο τεχνικές όταν εφαρμόζαμε τους αλγορίθμους χρησιμοποιήσαμε Cross Validation. Με το Cross Validation στην ουσία διασφαλίζουμε ότι τα αποτελέσματα μας δεν είναι τυχαία αφού τρέχουμε το πρόβλημα πολλές φορές και σπάζουμε κάθε φορά το training set παίρνοντας διαφορετικά training sets και validation sets.

2η Φάση: Εκπαίδευση/Μάθηση αλγόριθμων (learning/training) και πρόβλεψη (prediction)

Στην φάση αυτή αρχικά αποφασίσαμε τους αλγόριθμους τους οποίους θα χρησιμοποιούσαμε. Οι αλγόριθμοι αυτοί είναι οι ακόλουθοι:

a) Linear Regression
b) Polynomial Regression
c) Support Vector Regression (SVR)
d) Random Forest Regression
e) Lasso Regression
f) Decission Tree Regression
g) Gradient Boost Regression
h) Light Gradient Boosting Machine (ένα optimized version του decision tree.)

Επιλογή αλγορίθμων:
Linear, polynomial, lasso regressions, και οι 3 αλγόριθμοι ακολουθούν την ίδια φιλοσοφία. Ο polynomial είναι ένα είδος multi linear regressor, όπου η σχέση μεταξύ του mean και του value είναι ένα πολυώνυμο βαθμού n. Ο lasso από την άλλη επεκτείνει την λειτουργία του linear regressor, με την χρήση shrinkage. Δηλαδή τα δεδομένα μας μειώνονται προς το κέντρο πχ. mean. Θεωρήσαμε καλό αυτοί οι 3 να συγκριθούν μεταξύ τους ως προς τα αποτελέσματα αφού λειτουργούν με παρόμοιο τρόπο.
Ο support vector regressor, είναι ένας αλγόριθμος που μπορεί να χρησιμοποιηθεί με πολλά διαφορετικά kernels, για αυτό θεωρήσαμε αναγκαίο να τον συμπεριλάβουμε αφού θα βρούμε «ιδανικό», kernel, μέσω grid search. Στο grid search χρησιμοποιήσαμε linear, poly, rbf, sigmoid kernels. Μετά την ερευνά του grid search, βρήκαμε ως ιδανικό kernel, το polynomial.
Έπειτα επιλέξαμε τον random forest regressor, ο οποίος χρησιμοποιεί ensemble learning. Ο αλγόριθμος συνδυάζει πολλά predictions, από διάφορους αλγόριθμους, ώστε να κερδίσει ακρίβεια. Ο random forest, χτίζει πολλαπλά decision trees. Έτσι είναι φυσικό ότι θα συμπεριλαμβάναμε και τον αλγόριθμο decision tree regression, ώστε να συγκριθούν. Ο Decision tree χτίζει ένα δέντρο σπάζοντας το dataset σε πιο μικρά υποσύνολα.
Επιλέξαμε επίσης τον gradient boost regressor ο οποίος και αυτός συνδυάζει άλλους αδύναμους αλγόριθμους (ensemble learning) για να δημιουργήσει ένα δυνατό αλγόριθμο μάθησης. Χρησιμοποίει και αυτός decision trees, και στην θεωρεία πρέπει να ξεπερνά τον random forest. Επιλέξαμε επίσης να χρησιμοποιήσουμε το light gradient boosting library της Microsoft, το οποίο περιέχει optimized αλγόριθμους. Χρησιμοποιήσαμε τον gradient boost version του ώστε να τον συγκρίνουμε στα αποτελέσματα.

Διαδικασία grid search:Αρχικά δημιουργήσαμε parameter_grid για τον κάθε αλγόριθμο με διάφορες πιθανές παραμέτρους που θεωρήσαμε ότι πιθανόν να είναι οι ιδανικές. Έπειτα χρησιμοποιήσαμε το GridSearchCV με cv=5 για τον κάθε αλγόριθμο με σκοπό να βρούμε τις κατάλληλες παραμέτρους για την κάθε περίπτωση. Η διαδικασία αυτή για κάποιους αλγορίθμους (όπως ο SVR) ήταν πολύ χρονοβόρα.

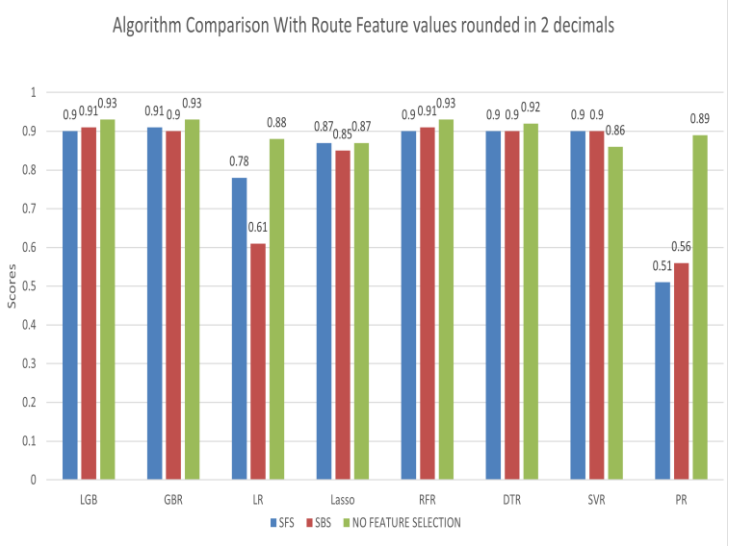
Στην συνέχεια αφού πήραμε τις κατάλληλες παραμέτρους για τον κάθε αλγόριθμο ξεκινήσαμε την διαδικασία εκπαίδευσης και δοκιμής του, για να μπορέσουμε να κρίνουμε και την αποτελεσματικότητα του.

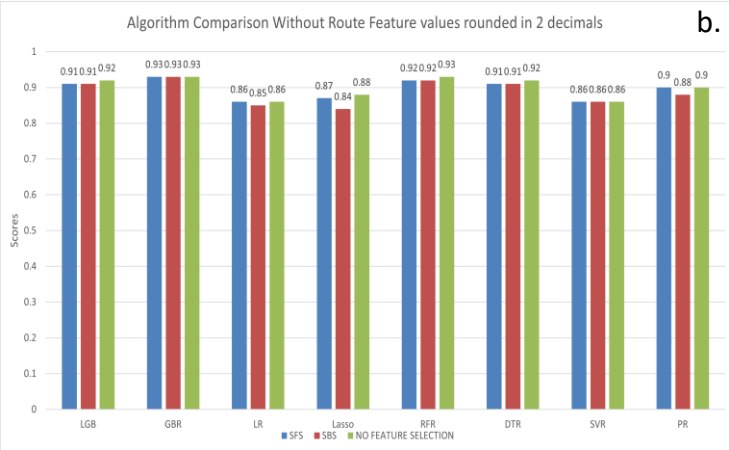
Η μετρική για την αποτελεσματικότητα κάθε αλγορίθμου δίνεται από την εκφώνηση του προβλήματος και είναι η ακόλουθη:

1 - np.sqrt(np.square(np.log10(y_pred +1) - np.log10(y_true +1)).mean())

Τα αποτελέσματα των αλγορίθμων φαίνονται στις πιο κάτω γραφικές.

3η Φάση: Αξιολόγηση αλγορίθμου (testing):





Τα αποτελέσματα των πιο πάνω γραφικών χωρίζονται σε δύο τομείς. Τα αποτελέσματα των αλγορίθμων χωρίς την χρήση του route feature, και με την χρήση του. Κάθε τομέας χωρίζεται σε 3 διαφορετικά στάδια ανάλογα με την χρήση του feature selection. Χρήση feature selection με Backward elimination, με forward selection και χωρίς χρήση feature selection.

1. Χωρίς το route feature:

   Σαν γενικό σχόλιο, βλέπουμε ότι τα καλυτέρα αποτελέσματα είχε ο gradient boost regressor, παρόλα αυτά και ο random forest χωρίς κάποιο feature selection κατάφερε να έχει score 0.93.

   a. SBS & SFS feature selection:

      Βλέπουμε ότι ο SFS έχει καλύτερα αποτελέσματα από το SBS σε μερικούς από τους αλγορίθμους (linear regressor, polynomial regressor , lasso regressor), και αυτό το πετυχαίνει με την χρήση λιγότερων features από τον SBS. Επίσης, οι καλύτεροι αλγόριθμοι είτε με SFS είτε με SBS είναι οι Gradient Boost Regression και Random Forest Regression με scores 0.93 και 0.92 αντίστοιχα.

   b. No feature selection:

      Χωρίς κάποιο feature selection, βλέπουμε ότι όλοι οι αλγόριθμοι έχουν καλύτερο αποτέλεσμα. Αυτό μας οδηγεί στο συμπέρασμα ότι η διαδικασία του feature selection αφαίρεσε κάποια features που πρόσθεταν ένα ελάχιστο προβάδισμα στους αλγόριθμους. Αξίζει να πούμε ότι δεν υπάρχει το 1% που χώριζε τους δύο καλύτερους αλγορίθμους όταν χρησιμοποιούσαμε SFS ή SBS, ενώ παράλληλα ανεβαίνουν στο 0.92 οι Decission Tree Regression και ο Light Gradient Boosting Machine.

2. Με το route feature:

   Ένα γενικό συμπέρασμα, βλέποντας την συγκεκριμένη γραφική παράσταση είναι ότι δεν εκμεταλλεύονται όλοι την έξτρα πληροφορία που δίνει αυτό το feature. Παρόλα αυτά, άλλοι λαμβάνουν υπόψη το νέο feature και φτάνουν μέχρι και 0.93 score. Επίσης ο linear regressor και polynomial regressor, χάνουν πληροφορία με την χρήση feature selection και έχουν χειρότερα αποτελέσματα.

   a. SBS & SFS feature selection:

      Είτε με SBS είτε με SFS, οι πλείστοι αλγόριθμοι συμπεριφέρονται το ίδιο, όμως αυτό που παρατηρούμε είναι ότι ο συνδυασμός SBS/SFS και route feature δίνει το πολύ 0.91 score και το μικρότερο score ήταν 0.51.

   b. No feature selection:

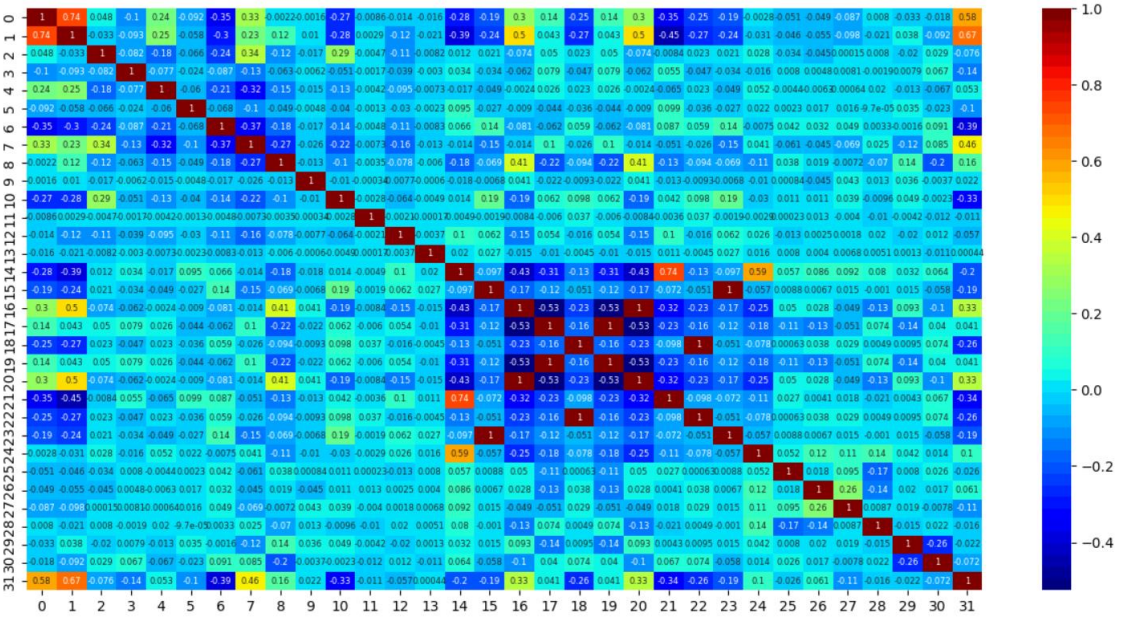
Χωρίς κάποιο feature selection, βλέπουμε ότι οι περισσότεροι αλγόριθμοι έχουν καλύτερο αποτέλεσμα. Είναι σημαντικό ότι οι δύο καλύτεροι αλγόριθμοι συνεχίζουν να είναι οι Gradient Boost Regression και Random Forest Regression με scores 0.93, score το οποίο καταφέρνει να πετύχει και ο Light Gradient Boosting Machine.

Συμπεράσματα:

- Ο gradient boost regressor είναι ο καλύτερος με score 0.93 , με και χωρίς την χρήση feature selection.
- Στα περισσότερα πειράματα το feature selection process έχει χειρότερα αποτελέσματα
- Συμπερασματικά το πείραμα με την στήλη route δεν ήταν και τόσο χρήσιμο , αφού μόνο ο Light gradient boost regressor κατάφερε να φτάσει σε πιο ψηλά σκορ τάξης του 93% κάτι που πετύχαμε ήδη.

Υπόμνημα:

Εικόνα 1.1:



Λόγω της φύσης του προβλήματος μας (πρόβλημα regression) βάλαμε και την στήλη Price που είναι το αποτέλεσμα που θέλουμε προβλέψουμε, ώστε να δούμε πώς συσχετίζεται με τα υπόλοιπα features. Το Duration με τα total stops έχουν την μεγαλύτερη συσχέτιση, όπως επίσης και κάποια δρομολόγια (source-destination). Επίσης βλέποντας την συσχέτιση price-duration (0.577), αλλά και total stops-price (0.670), παίρνουμε αρκετές πληροφορίες για το ζητούμενο (price).