Θεωρία Αποφάσεων

Εργαστηριακή Άσκηση

Αλέξανδρος Κουτσούκος, 1064882

Γεώργιος Νικολακόπουλος, 1059620

Νικόλαος Αβραντινής, 1059614

Χειμερινό Εξάμηνο 2021-2022

Περιεχόμενα

Data Loading and overview

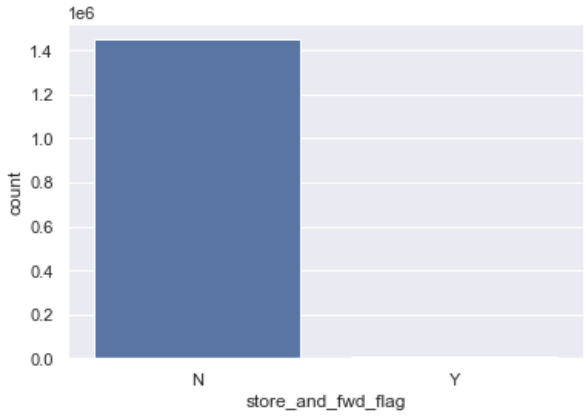
Σε αυτό το μέρος της εργασίας, φορτώνουμε τα δωσμένα datasets και κάνουμε μία πρώτη ανάλυση κυρίως για να εξάγουμε αποτελέσματα σχετικά με την κατανομή των τιμών σε ορισμένα πεδία.

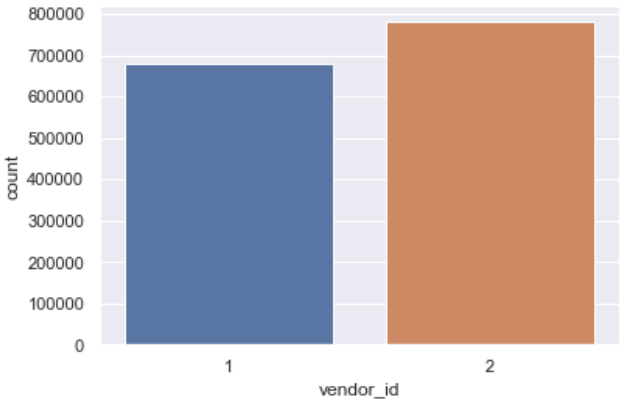
Διώχνουμε το πεδίο "id" διότι θα αλλοιώσει τα αποτέλεσματα των προβλέψεων στα μοντέλα μας και το πεδίο "dropoff_datetime" γιατί δεν υπάρχει στο αρχείο test.csv και θέλουμε τα αρχεία μας train και test να έχουν τα ίδια πεδία για να γίνει η εκπαίδευση των μοντέλων πάνω σε αυτά και αντίστοιχα η πρόβλεψη. Επιπλέον, είναι έτσι και αλλιώς περιττό αφού μας δίνεται η ώρα εκκίνησης (pickup_datetime) και η διάρκεια του ταξιδιού (trip_duration).

```
# Drop id column from both dataframes and dropoff_datetime from train
train.drop('id', axis=1, inplace=True)
test.drop('id', axis=1, inplace=True)

train.drop(['dropoff_datetime'], axis=1, inplace=True)
```

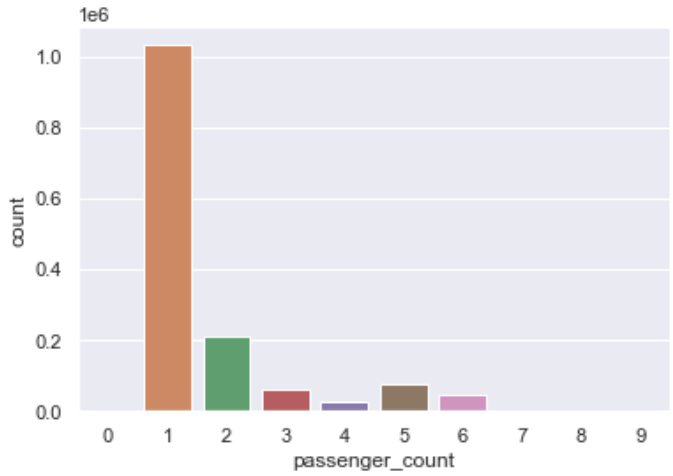
Παρατηρούμε πως στο πεδίο "store_and_fwd_flag", το οποίο υποδεικνύει εάν το αρχείο του ταξιδιού διατηρήθηκε στη μνήμη του οχήματος πριν προωθηθεί καθώς δεν υπήρχε σύνδεση με το διακομιστή, η τιμή N η οποία θεωρεί ότι αρχείο προωθήθηκε χωρίς να αποθηκευτεί, εμφανίζεται στη συντριπτική πλειοψηφία των εγγραφών.

Αντίθετα το πεδίο "vendor_id", το οποίο αναφέρεται στον πάροχο των υπηρεσιών, οι εγγραφές είναι σχεδόν μοιρασμένες ανάμεσα στους δύο παρόχους που υπάρχουν στα δεδομένα μας.





Επίσης, παρατηρούμε πως στις περισσότερες διαδρομές υπήρχε ένας πελάτης στο ταξί, σε αρκετά λιγότερες υπήρχαν δύο ενώ σε ελάχιστες διάδρομες υπήρχαν τρεις ή περισσότεροι, ενώ βλέπουμε πως υπάρχουν και εγγραφές με μηδέν πελάτες τις οποίες θα χειριστούμε παρακάτω.



Data Cleaning

Σε αυτό το μέρος της εργασίας διαχειριζόμαστε τις τιμές στο dataset που μπορεί να προκαλέσουν πρόβλημα στις προβλέψεις μας όπως ακραίες τιμές (outliers) ή κενές (null). Ως ακραίες θεωρούμε τις τιμές στο dataset που διαφέρουν πάρα πολύ από την πλειοψηφία και εμφανίζονται ελάχιστες φορές οπότε μπορεί να επηρεάσουν το αποτέλεσμα στην πρόβλεψη ενώ δεν θα έπρεπε.

Έλεγχος για ύπαρξη κενών τιμών, παρατηρούμε πως δεν υπάρχουν.

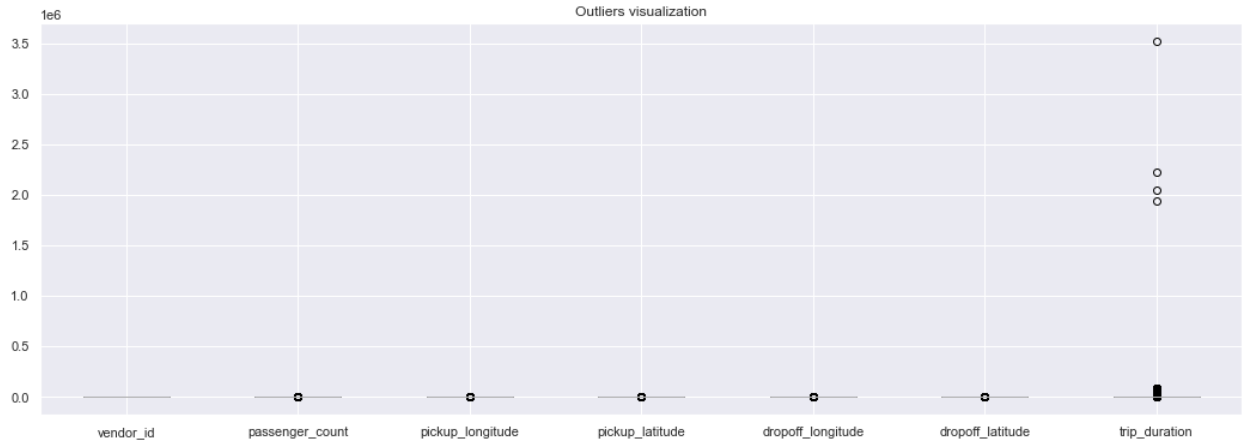
```
train.isna().sum()
```

```
vendor_id           0
pickup_datetime     0
passenger_count     0
pickup_longitude    0
pickup_latitude     0
dropoff_longitude   0
dropoff_latitude    0
store_and_fwd_flag  0
trip_duration       0
dtype: int64
```

```
test.isna().sum()
```

```
vendor_id           0
pickup_datetime     0
passenger_count     0
pickup_longitude    0
pickup_latitude     0
dropoff_longitude   0
dropoff_latitude    0
store_and_fwd_flag  0
dtype: int64
```

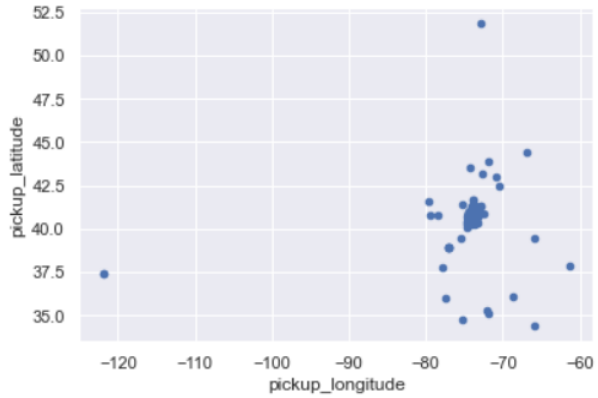
Το παρακάτω διάγραμμα ακραίων τιμών μας βοηθάει να παρατηρήσουμε την κατανομή των τιμών για κάθε χαρακτηριστικό και να εντοπίσουμε τις ακραίες. Στη συγκεκριμένη περίπτωση θα αφαιρέσουμε τις διαδρομές που διαρκούν παραπάνω από δύο ώρες και λιγότερο από ένα λεπτό και αυτές χωρίς κανέναν πελάτη γιατί αποτελούν ακραίες τιμές.

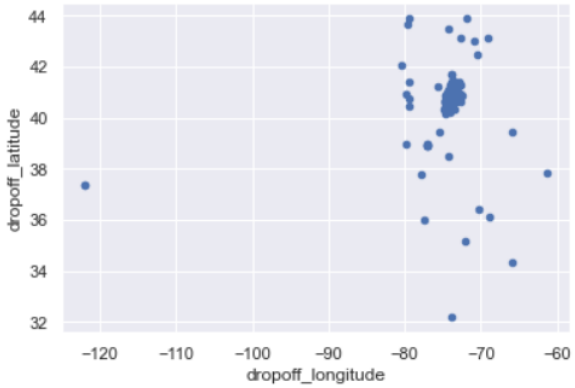



```
# Only keep trips that lasted less than 2 hours and more than 1 minute
train = train[train.trip_duration < 7200 and train.trip_duration > 60]
```

```
# Only keep trips with passengers
train = train[train['passenger_count'] > 0]
```

Στη συνέχεια σχεδιάζουμε τις τοποθεσίες που έχουν γίνει διαδρομές, βάσει των συντεταγμένων τους, ώστε να εντοπίσουμε και εδώ ακραίες τιμές.





Παρατηρώντας τα διαγράμματα περιορίσαμε τις συντεταγμένες στις παρακάτω περιοχές.

```
# Drop pickup and dropoff long lat that look like outliers
train = train[train['pickup_longitude'] > -85]
train = train[train['pickup_latitude'] < 45]

train = train[train['dropoff_longitude'] > -85]
train = train[train['dropoff_latitude'] > 34]
```

Feature Engineering and Data Analysis

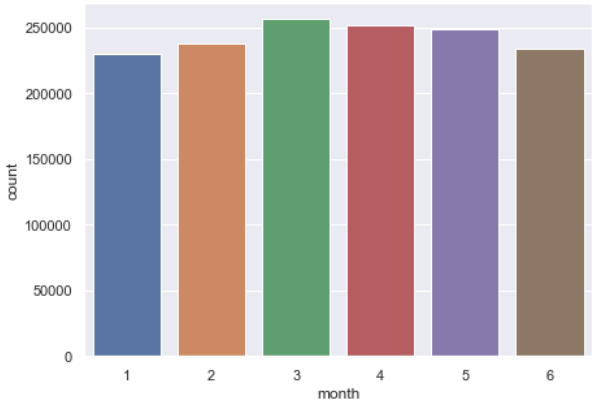
Σε αυτό το μέρος της εργασίας κάνουμε μία περαιτέρω επεξεργασία στα δεδομένα μας για μεγαλύτερη απόδοση στις προβλέψεις και μία επιπλέον ανάλυση για εξαγωγή χρήσιμων συμπερασμάτων.

Ορίζουμε τη συνάρτηση parse_dates με την οποία σπάμε το ενίαιο πεδίο “pickup_datetime” που περιλαμβάνει ολόκληρες ημερομηνίες, σε τέσσερα διαφορετικά πεδία που περιέχουν τον μήνα, την εβδομάδα, τη μέρα και την ώρα αντίστοιχα. Οι τιμές του πεδίου “weekday” (μέρα) επειδή αποτελούνται από χαρακτήρες, μετατρέπονται σε αριθμούς για να τις διαβάζουν τα μοντέλα μας.

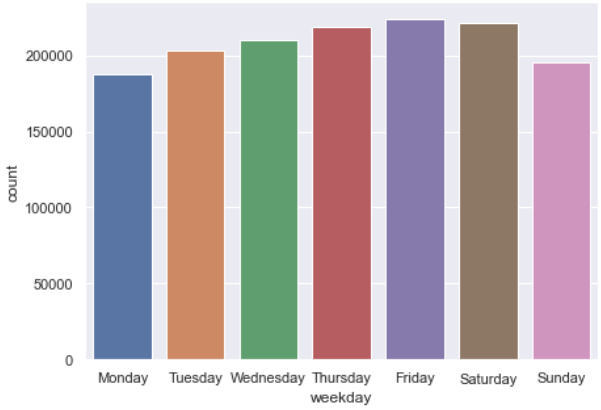
```
def parse_dates(data):
    data['pickup_datetime'] = pd.to_datetime(data.pickup_datetime)

    data['month'] = data.pickup_datetime.dt.month
    data['week'] = data.pickup_datetime.dt.week
    data['weekday'] = data.pickup_datetime.apply(lambda x: calendar.day_name[x.weekday()])
    data['hour'] = data.pickup_datetime.dt.hour


    data.drop(['pickup_datetime'], axis=1, inplace=True)
```

Αφού εφαρμόσουμε την παραπάνω συνάρτηση στα datasets που έχουμε, κάνουμε μια σχηματική ανάλυση για κάθε πεδίο που προκύπτει από την επεξεργασία των ημερομηνιών ώστε να παρατηρήσουμε το πλήθος των διαδρομών για κάθε μήνα, μέρα, ώρα αλλά και πως επηρεάζεται ο μέσος χρόνος της διαδρομής σε συνάρτηση με τα πεδία αυτά.

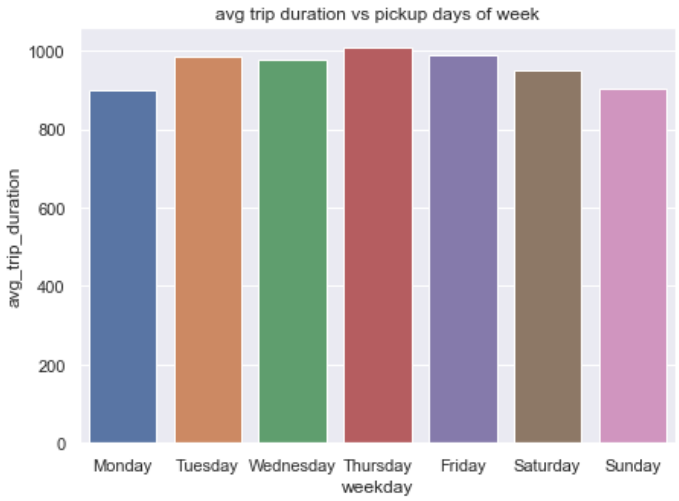
Πλήθος διαδρομών ανά μήνα



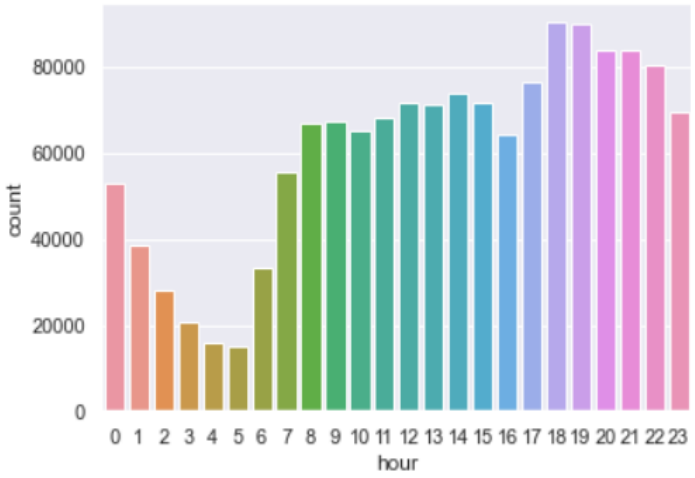
Πλήθος διαδρομών ανά μέρα



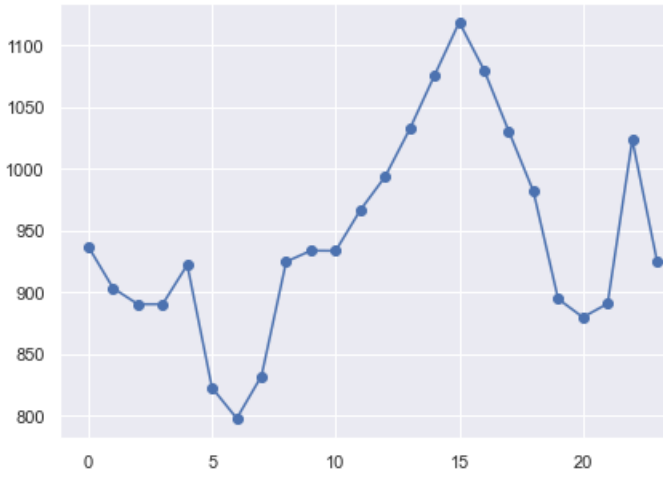
Μέσος χρόνος διαδρομής (σε δευτερόλεπτα) ανά μέρα



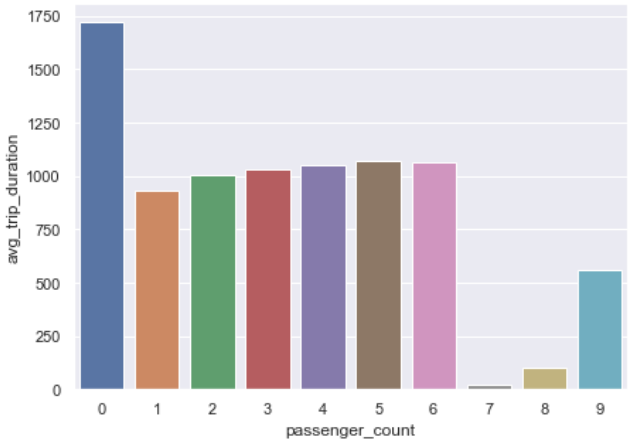
Πλήθος διαδρομών ανά ώρα



Μέσος χρόνος διαδρομής ανά ώρα

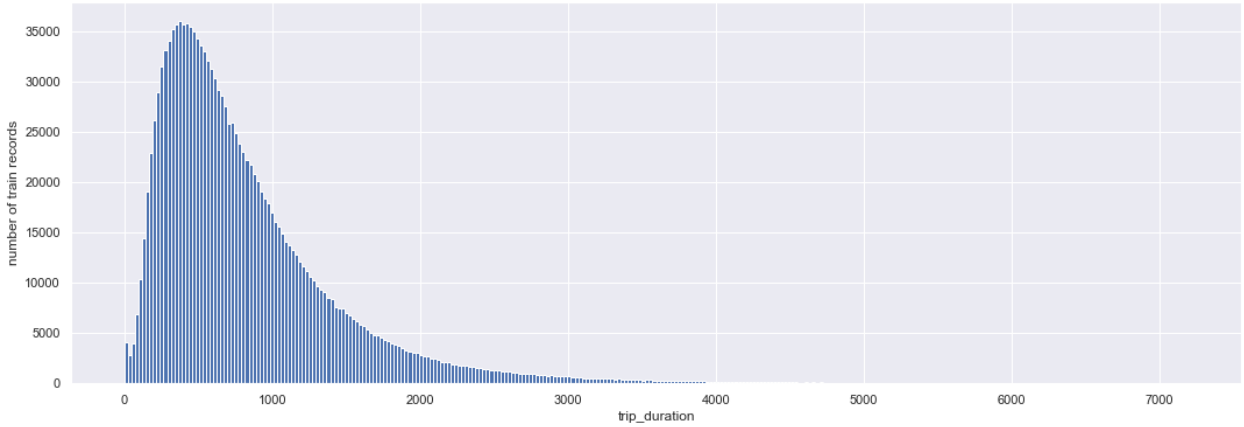


Μέσος χρόνος διαδρομής σε σχέση με το πλήθος πελατών στο ταξί.

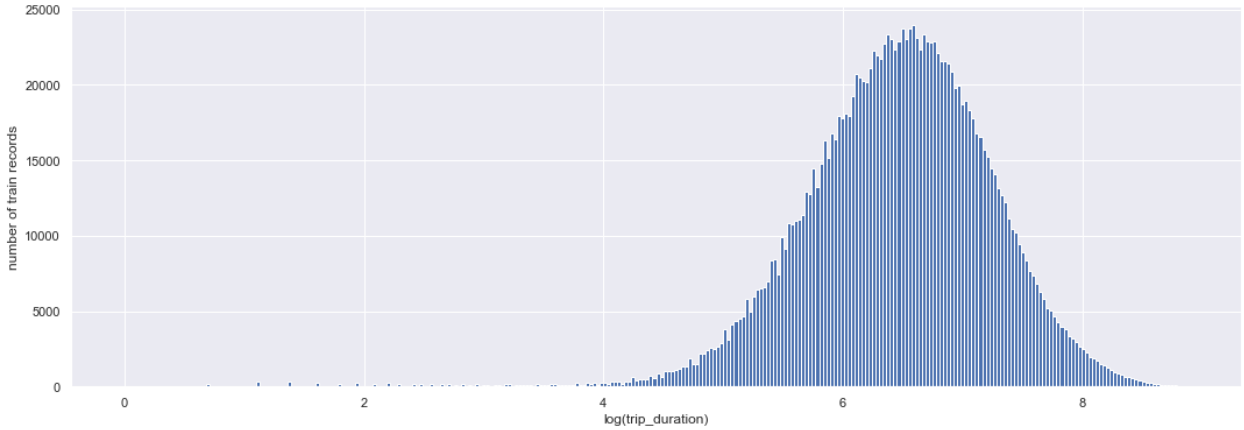


Παρατηρούμε ότι το πλήθος των διαδρομών ανά ημέρα επηρεάζει και το μέσο χρόνο διαδρομής την αντίστοιχη ημέρα, δηλαδή τις ημέρες που υπάρχουν περισσότερες διαδρομές αυξάνεται και ο μέσος χρόνος. Αντίστοιχη τάση παρατηρείται και στο μέσο χρόνο διαδρομής ανά ώρα αλλά όχι σε ολόκληρο το εύρος του διαστήματος. Η αυξανόμενη κίνηση τις ώρες 8 με 3 αποτυπώνεται και στο μέσο χρόνο αλλά στη συνέχεια τις ώρες 6 με 10 που επιτυγχάνεται το μέγιστο πλήθος διαδρομών ο μέσος χρόνος διαδρομής παρουσιάζει πτωτική τάση.

Κατανομή των τιμών του χαρακτηριστικού trip_duration (διάρκεια της διαδρομής)



Παρατηρούμε παραπάνω ότι η κατανομή της διάρκειας του ταξιδιού είναι ασύμμετρη και πιο συγκεκριμένα right-skewed οπότε θα χρησιμοποιήσουμε την λογαριθμική συνάρτηση για να πετύχουμε κανονική κατανομή, όπως φαίνεται παρακάτω. Η διάρκεια των περισσότερων διαδρομών όπως βλέπουμε είναι μεταξύ του exp(4) και exp(8) δηλαδή ανάμεσα σε 1 με 50 λεπτά.



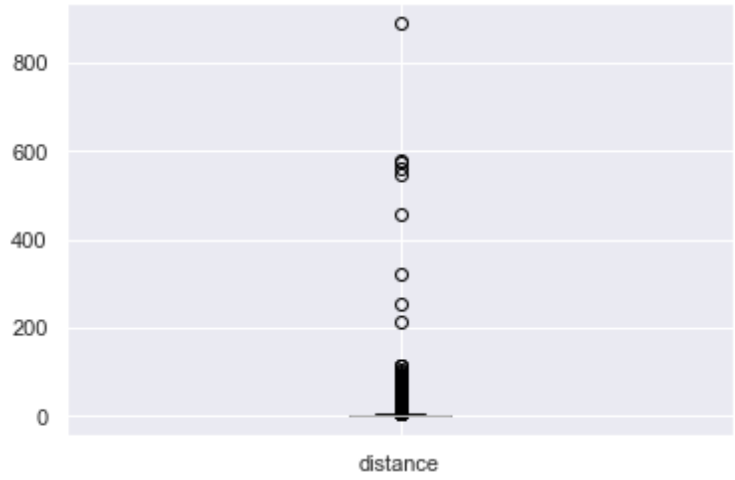
Το τελευταίο βήμα προεπεξεργασίας που κάνουμε στα δεδομένα είναι να μετατρέψουμε τα κατηγορικά δεδομένα (vendor_id, store_and_fwd_flag) σε δυαδικά, με τη διαδικασία one-hot-encoding, ώστε να είναι κατανοητά στα μοντέλα πρόβλεψης που θα χρησιμοποιήσουμε. Κάθε κατηγορία των χαρακτηριστικών θα είναι μια ξεχωριστή στήλη η οποία θα παίρνει την τιμή 0 ή 1 ανάλογα αν χρησιμοποιείτε ή όχι.

```
# One hot encoding binary categorical features
train = pd.concat([train, pd.get_dummies(train.store_and_fwd_flag, prefix='flag')], axis=1)
train = pd.concat([train, pd.get_dummies(train.vendor_id, prefix='vendor')], axis=1)

train.drop(['store_and_fwd_flag'], axis=1, inplace=True)
train.drop(['vendor_id'], axis=1, inplace=True)
```

Ορίζουμε τη συνάρτηση haversine_distance για να υπολογίσουμε την απόσταση κάθε διαδρομής μέσω των συντεταγμένων εκκίνησης και άφιξης, με τη βοήθεια της βιβλιοθήκης haversine της python.

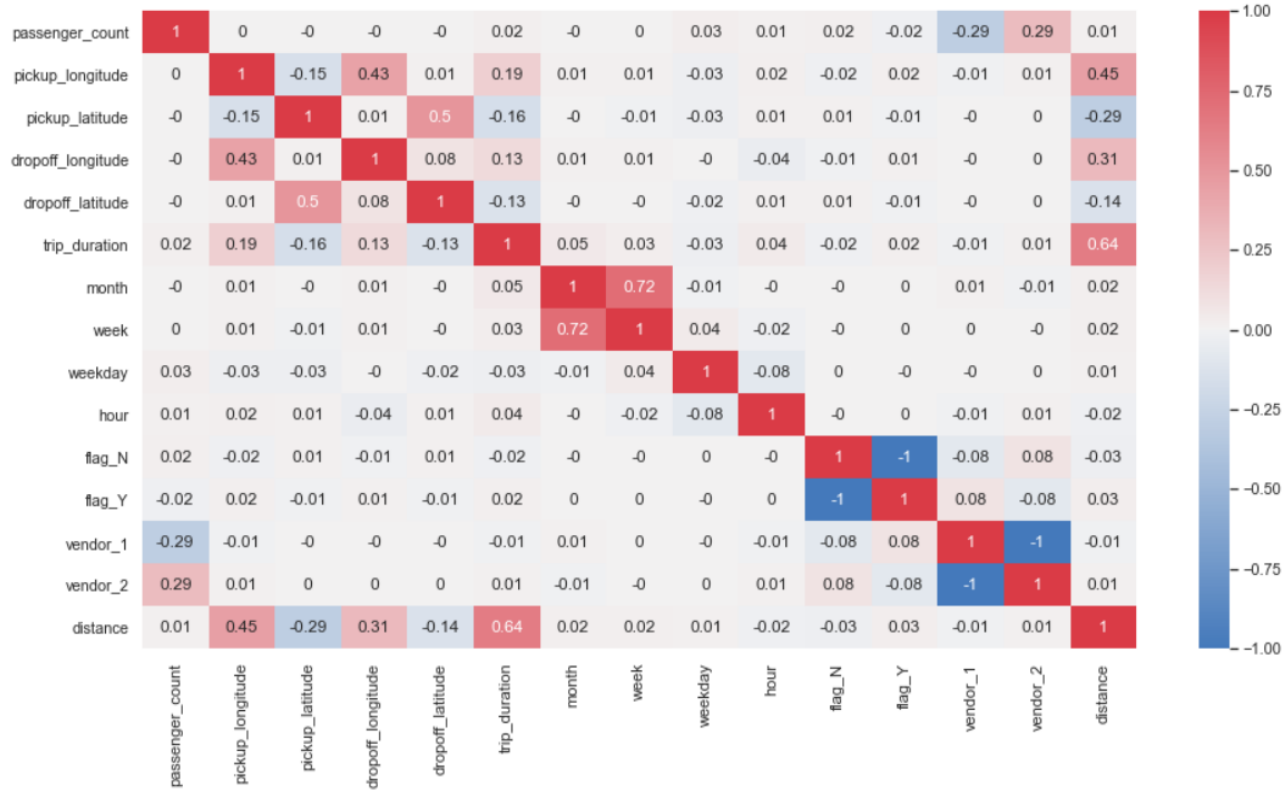
```
def haversine_distance(data):
    pickup_lon = data['pickup_longitude'].values
    pickup_lat = data['pickup_latitude'].values

    dropoff_lon = data['dropoff_longitude'].values
    dropoff_lat = data['dropoff_latitude'].values

    distance_list = []

    for i in range(len(data)):
        pickup = (pickup_lat[i], pickup_lon[i])
        dropoff = (dropoff_lat[i], dropoff_lon[i])
        distance = hs.haversine(pickup, dropoff)
        distance_list.append(distance)

    data['distance'] = distance_list
```

Από το παρακάτω διάγραμμα με το πλήθος εγγραφών για κάθε απόσταση, συμπεραίνουμε πως η συντριπτική πλειοψηφία των διαδρομών είναι κάτω από τα 200 χλμ με εξαίρεση ελάχιστες περιπτώσεις, τις οποίες και θεωρούμε ακραίες τιμές και τις αφαιρούμε από το dataset.




```
# Only keep trips with distances under 200km
train = train[train['distance'] < 200]
```

Πίνακας συσχετίσεων μεταξύ των μεταβλητών του dataset.



Ως πίνακας συσχέτισης ορίζεται ένας πίνακας που εμφανίζει τους συντελεστές συσχέτισης για διαφορετικές μεταβλητές. Ο παραπάνω πίνακας απεικονίζει τη συσχέτιση μεταξύ όλων των πιθανών ζευγών τιμών και είναι ένα ισχυρό εργαλείο για το εντοπισμό μοτίβων στα δεδομένα. Στην περίπτωση μας βλέπουμε ότι η διάρκεια του ταξιδιού (trip_duration), την οποία θέλουμε αργότερα να προβλέψουμε, εμφανίζει αρκετά μεγάλη θετική συσχέτιση με την απόσταση (distance), δηλαδή όσο μεγαλώνει η απόσταση τόσο μεγαλύτερη γίνεται και η διάρκεια του ταξιδιού.

Building models

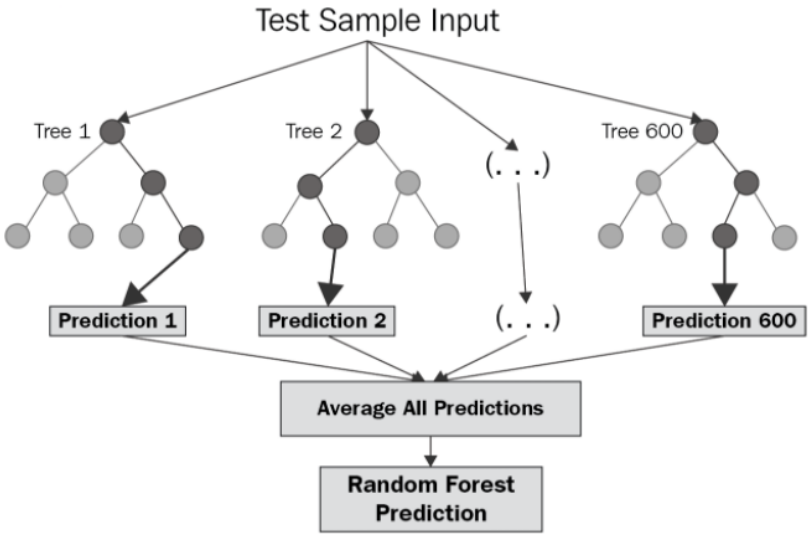
Σε αυτό το μέρος της εργασίας, κατασκευάζουμε διάφορα μοντέλα παλινδρόμησης τα οποία εκπαιδεύουμε με τα δεδομένα που έχουμε προεπεξεργαστεί παραπάνω και τέλος τα αξιολογούμε χρησιμοποιώντας κατάλληλες μετρικές. Στόχος μας είναι να βρούμε το μοντέλο που επιτυγχάνει τις καλύτερες επιδόσεις για το πρόβλημα μας.

Αρχικά χωρίζουμε τα δεδομένα σε δεδομένα εκπαίδευσης και αξιολόγησης με αναλογία 80-20 και έπειτα εκπαιδεύουμε το κάθε μοντέλο ξεχωριστά στα δεδομένα εκπαίδευσης. Στη συνέχεια, μετράμε την απόδοση του κάθε μοντέλου κάνοντας προβλέψεις για τα δεδομένα αξιολόγησης και τις συγκρίνουμε με τις πραγματικές τιμές. Οι μετρικές που χρησιμοποιούμε για να μετρήσουμε αυτή τη διαφορά είναι το Μέσο Απόλυτο Σφάλμα (MAE), το Μέσο Τετραγωνικό Σφάλμα (MSE) και η ρίζα του Μέσου Τετραγωνικού σφάλματος (RMSE).

Παρακάτω θα παρουσιάσουμε τα 4 μοντέλα που υλοποιήσαμε και τις επιδόσεις τους.

Random Forest model

Ο αλγόριθμος Random Forest είναι ένας ικανός αλγόριθμος που μπορεί να χρησιμοποιηθεί για προβλήματα παλινδρόμησης αλλά και ταξινόμησης, και βασίζεται στη χρήση πολλαπλών δέντρων αποφάσεων. Η βασική ιδέα πίσω από αυτό είναι ο συνδυασμός πολλαπλών δέντρων αποφάσεων, ως μοντέλα εκμάθησης, για τον προσδιορισμό της τελικής εξόδου αντί να βασίζεται σε μεμονωμένα δέντρα απόφασης.

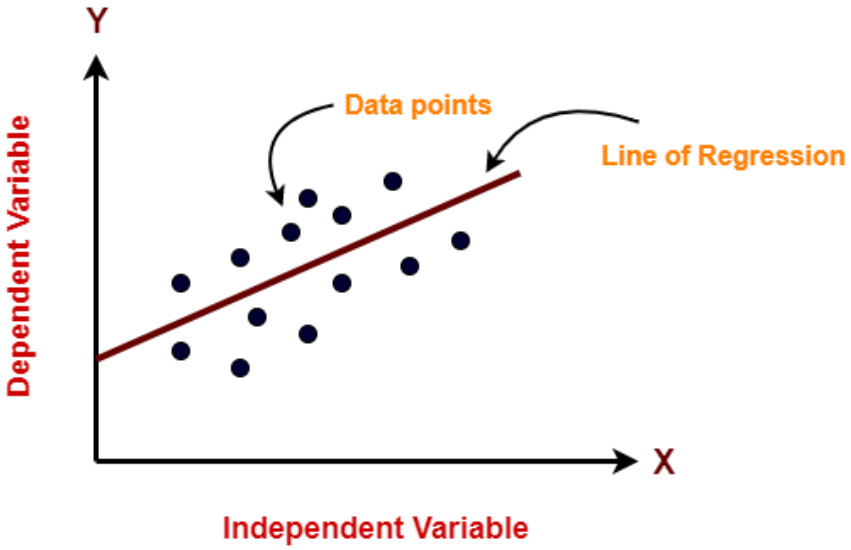


Αποτελέσματα

```
Mean Absolute Error  0.2366715815660458
Mean Squared Error:  0.10503864653376876
Root Mean Squared Error:  0.3240966623304979
```

Linear Regression model

Ο αλγόριθμος Linear Regression είναι ένας απλός στην υλοποίηση αλγόριθμος μηχανικής μάθησης που χρησιμοποιείτε για προβλήματα παλινδρόμησης. Η βασική ιδέα πίσω από αυτόν είναι εκτέλεση βημάτων για την πρόβλεψη της εξαρτημένης μεταβλητής με βάση κάποιες δεδομένες ανεξάρτητες μεταβλητές. Οπότε καταλαβαίνουμε αυτή η τεχνική παλινδρόμησης ανακαλύπτει μια γραμμική σχέση μεταξύ μιας εξαρτημένης μεταβλητής και των άλλων δεδομένων ανεξάρτητων μεταβλητών.

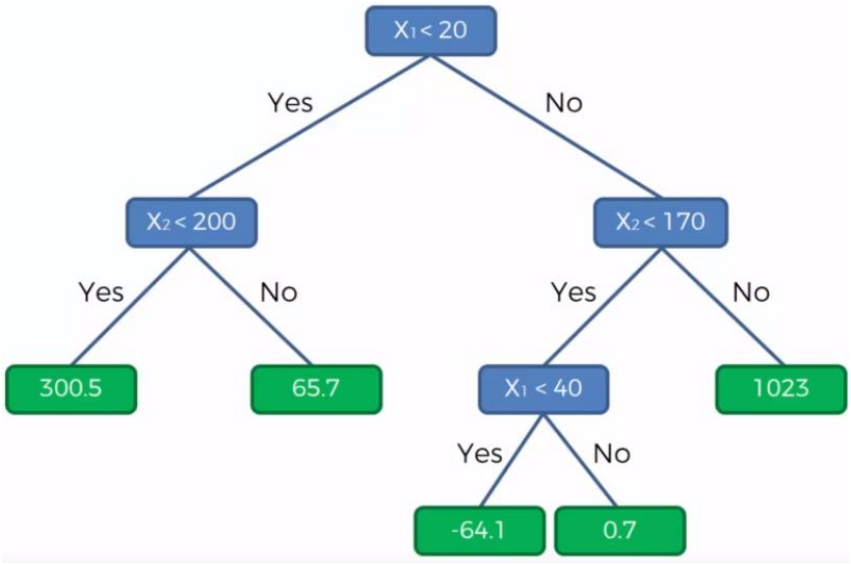


Αποτελέσματα

```
Mean Absolute Error  0.4189735186030883
Mean Squared Error:  0.29503369632903303
Root Mean Squared Error:  0.5431700436594723
```

Decision Tree model

Ο αλγόριθμος Decision Tree είναι ένας αλγόριθμος που μπορεί να εφαρμοστεί σε δεδομένα που περιέχουν αριθμητικά και κατηγορικά δεδομένα. Τα δέντρα αποφάσεων είναι καλά στην καταγραφή της μη γραμμικής αλληλεπίδρασης μεταξύ των ανεξάρτητων μεταβλητών και της μεταβλητής στόχου. Εν συντομία ένα δέντρο αποφάσεων είναι ένα δέντρο όπου κάθε κόμβος αντιπροσωπεύει ένα χαρακτηριστικό, κάθε κλάδος αντιπροσωπεύει μια απόφαση και κάθε φύλλο αντιπροσωπεύει ένα αποτέλεσμα.

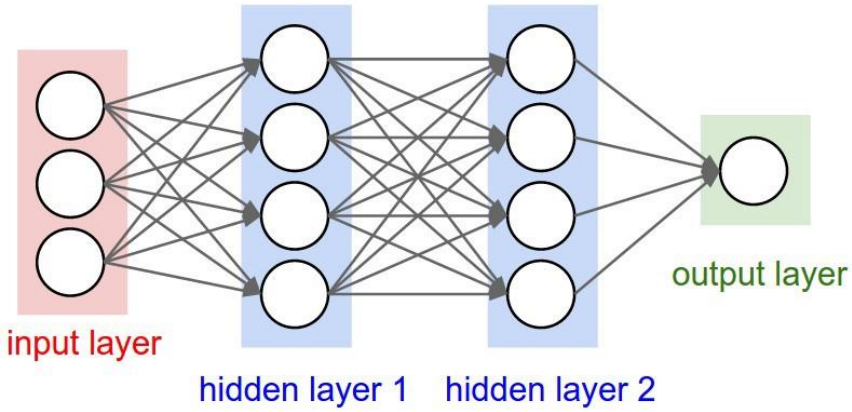


Αποτελέσματα

```
Mean Absolute Error  0.3432888304534125
Mean Squared Error:  0.21877179391336876
Root Mean Squared Error:  0.4677304714398761
```

Neural Network model

Ο σκοπός της χρήσης νευρωνικών δικτύων για προβλήματα παλινδρόμησης είναι η δυνατότητα που έχουν να μαθαίνουν τη σύνθετη μη γραμμική σχέση μεταξύ των χαρακτηριστικών και του στόχου λόγω της συνάρτησης ενεργοποίησης που υπάρχει σε κάθε επίπεδο του δικτύου.



Για το συγκεκριμένο μοντέλο πριν το εκπαιδεύσουμε κανονικοποιούμε όλα τα δεδομένα του dataset X διότι τα νευρωνικά δίκτυα επιτυγχάνουν καλύτερες επιδόσεις όταν δουλεύουν με δεδομένα οι τιμές των οποίων βρίσκονται στο διάστημα 0-1.

Η αρχιτεκτονική που ακολουθήσαμε στο συγκεκριμένο δίκτυο φαίνεται παρακάτω.

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_4 (Dense)             (None, 64)                960

 dense_5 (Dense)             (None, 64)                4160

 dense_6 (Dense)             (None, 32)                2080

 dense_7 (Dense)             (None, 1)                 33

=================================================================
Total params: 7,233
Trainable params: 7,233
Non-trainable params: 0
```

Αποτελέσματα

Παρακάτω φαίνονται οι τιμές του MSE για τα δεδομένα εκπαίδευσης αλλά και αξιολόγησης για κάθε ένα από τα 20 epochs(=επαναλήψεις) που εκπαιδεύσαμε το μοντέλο. Με την τιμή για τα δεδομένα αξιολόγησης στην τελευταία επανάληψη να είναι MSE = 0.1114

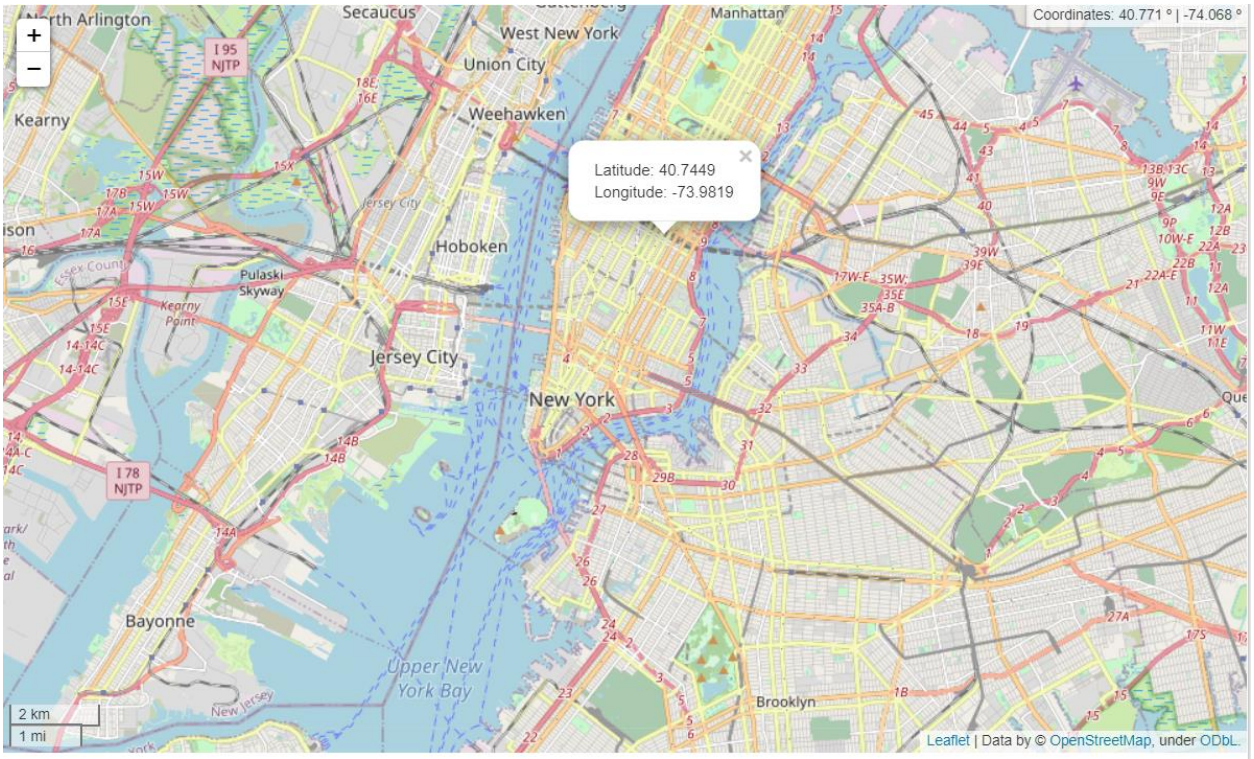
Συμπέρασμα

Συγκρίνοντας τα παραπάνω αποτελέσματα καταλήγουμε στο συμπέρασμα ότι το μοντέλο Random Forest επιτυγχάνει τις καλύτερες επιδόσεις για το πρόβλημα μας, με το μοντέλο Neural Network να ακολουθεί με σχεδόν παρόμοια επίδοση αλλά μεγαλύτερη πολυπλοκότητα στην υλοποίηση. Το μοντέλο Linear Regression παρουσιάζει τις χειρότερες επιδόσεις καθώς απλοποιεί το πρόβλημα θεωρώντας ότι υπάρχει γραμμική σχέση ανάμεσα στις μεταβλητές και ως εκ τούτου δεν προτείνεται για πρακτικές εφαρμογές. Τέλος, η επίδοση του Decision Tree αλγορίθμου είναι καλύτερη από αυτή του Linear Regression αλλά και πάλι δεν πλησιάζει τους καλύτερους αλγορίθμους.

Mini prediction app

Στο τελευταίο μέρος της εργασίας, κατασκευάζουμε ένα πρόγραμμα το οποίο δέχεται ως είσοδο συντεταγμένες αρχής και προορισμού, οι οποίες μπορούν να αντληθούν από τον χάρτη που έχουμε ενσωματώσει στο πρόγραμμα, και προβλέπει την διάρκεια της διαδρομής χρησιμοποιώντας το μοντέλο Random Forest που επιτυγχάνει τις καλύτερες επιδόσεις. Για να έχουμε μέτρο σύγκρισης για την ποιότητα της πρόβλεψης χρησιμοποιούμε ως ground truth την διάρκεια της διαδρομής μέσω οδικών δικτύων που υπολογίζει το API της Google που καλούμε. Παρακάτω παρουσιάζονται τα βασικότερα σημεία της υλοποίησης.

Ένας χάρτης παραθέτετε στον χρήστη με αρχική θέση την κάτοψη της Νέας Υόρκης, δίνοντας του όλες τις βασικές λειτουργίες ενός χάρτη καθώς και την δυνατότητα με ένα κλικ του ποντικιού του να λαμβάνει σε αναδυόμενο μήνυμα της συντεταγμένες της συγκεκριμένης τοποθεσίας.
Στην συνέχεια, ζητείται από τον χρήστη να εισάγει τις συντεταγμένες της τοποθεσίας που βρίσκεται και ύστερα τις συντεταγμένες του προορισμού του. Η είσοδος ελέγχεται από το σύστημα ώστε όντως να βρίσκονται τα δύο σημεία εντός Νέας Υόρκης, ενώ αν εισαχθεί τοποθεσία που βρίσκεται σε νερό (θάλασσα, ποτάμι, λίμνη) το σύστημα βρίσκει το κοντινότερο σημείο στεριάς και λαμβάνει αυτό ως εισαχθέν.



Ο χρήστης αφού εισάγει τις συντεταγμένες για τα δύο σημεία, έχει την επιλογή να προσθέσει χρόνο αναχώρησης για την διαδρομή σε μορφή λεπτών. Η είσοδος και πάλι ελέγχεται να είναι ένα θετικός αριθμός ενώ για είσοδο μηδέν ο χρόνος αναχώρησης δεν λαμβάνεται υπόψιν ενώ γίνονται και οι απαραίτητες μετατροπές για να προστεθεί ο χρόνος αναχώρησης στην ώρα που εισάγεται.

```
Enter pickup longitude: -73.9819
Enter pickup latitude: 40.7449
Location's coordinates have been set correctly!
Enter dropoff longitude: -73.9789
Enter dropff latitude: 40.7112
Destination's coordinates have been set correctly!
Provide with departure time in minutes: 5
```

Μετά την σωστή εισαγωγή όλων των στοιχείων από τον χρήστη το σύστημα παίρνει την σκυτάλη και έχοντας συλλέξει όλα τα στοιχεία κάνει ένα request με τα απαραίτητα ορίσματα στο Distance Matrix API της Google για να λάβει ένα response και να το εκτυπώσει αυτά που περιλαμβάνει δηλαδή την απόσταση των σημείων οδικά, την διάρκεια που χρειάζεται η μεταφορά αλλά και μια εκτίμηση της διάρκειας για την πιθανότητα κυκλοφοριακής συμφόρησης.

```
Google API trip duration: 5 mins
Google API trip duration in traffic: 6 mins
Rf model predicted trip duration: 6 mins
```

Σημαντική σημείωση είναι πως το σύστημα αφού καταλήξει στην ημερομηνία και ώρα που θα γίνει η διαδρομή αλλάζει τον χρόνο της σε 2016 για να μπορεί να γίνει η πρόβλεψη από το πρόγραμμα μας.