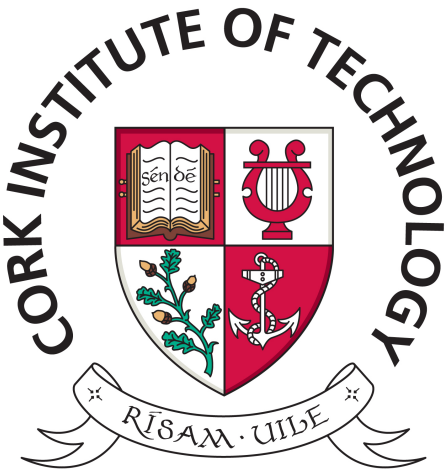

Taxi Trip Duration: Prediction and Analysis

by

Pooja Tyagi

This thesis has been submitted in partial fulfillment for the degree of Master of Science in Artificial Intelligence

in the
Faculty of Engineering and Science
Department of Computer Science

May 2019

Declaration of Authorship

I, Pooja Tyagi , declare that this thesis titled, 'Taxi Trip Duration: Prediction and Analysis' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for an masters degree at Cork Institute of Technology.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at Cork Institiute of Technology or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this project report is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.
- I understand that my project documentation may be stored in the library at CIT, and may be referenced by others in the future.

Signed: _____

Date: _____

CORK INSTITUTE OF TECHNOLOGY

Abstract

Faculty of Engineering and Science
Department of Computer Science

Master of Science

by Pooja Tyagi

New York city is a city of millions, where people travels more than a million miles a day. Taxis plays an important role not only in New York but in whole world where most of us do not own cars. In this era, where ride sharing applications are gaining popularity, it is important for people and taxi companies to retain transparency in trips details that includes trip duration, taxi fares etc. Predicting trip duration could help passengers to decide the optimal time to start their journey, also could help taxi drivers to decide which ride could make more profit. This could be helpful to Taxi services providing companies like Uber, My taxi , in retaining their consumers. To predict trip duration, data available at the beginning includes features such as pickup and drop-off coordinates (latitude and longitudes), number of passengers, date, month, time. In this project Machine Learning algorithms were used to predict trip duration. In particular, Linear Regression, Random Forests and XGBoost Regression algorithms were used...

Acknowledgements

I would like to thank my supervisor, Dr. Laura Climent for her goodwill, approach, and endless patience. More-over I would like to thank my parents and friends, for their unconditional support during my whole study.. . .

Contents

List of Figures

List of Tables

Abbreviations

NYC	**N**ew **Y**ork **C**ity
XGBoost	e**X**treme **G**radient **B**oosting
OSRM	**O**pen **S**ource **R**outing **M**achine
SSE	**S**um of **S**quare **E**rror
RF	**R**andom **F**orests
DNN	**D**eep **N**eural **N**etwork
OLS	**O**rdinary **L**east **S**quare
MAPE	**M**ean **A**bsolute **P**ercentage **E**rror
RMSE	**R**oot **M**ean **S**quare **E**rror

Dedicated to my mother...

Chapter 1

Introduction

The taxicabs of the New York make up a mature system. Cabs are a great compliment to transit. In New York City taxi rides portrays vibrant picture of life. Many ride sharing apps are gaining popularity because of increasing demands. It is important for companies to advance their systems with new methodology and get rid of inconveniences faced by the customers, drivers and itself taxi companies. They need to provide more visibility to their approximate fares and trip duration in order to compete with existing apps. Providing visibility to trip duration and taxi fares will give consumers an idea and they can chose optimal time to start their ride, drivers can select rides which will make more profit. Furthermore, it will provide insight into traffic patterns, road blockage or large-scale events that attract many people of New York.

To improve the efficiency of ride sharing apps, predicting trip duration is one of the important and required facility that a consumer expects. To achieve this goal many scholars and researchers worked in this field and gave significant contribution and are still working to improve the accuracy of predicting fare and trip duration. Researchers collected various historical data of different areas like New York, Singapore, Japan etc. Depending upon the area they used multiple techniques to predict the trip duration. Machine Learning has been proven to be very effective in this kind of situations. In my research I built three machine learning Regression models using Linear Regression, Random Forest and XGBoost Regression techniques. Finally evaluated the results of all three models and concluded the best model with maximum efficiency.

1.1 Motivation

The motivation behind this thesis came from real-time issues that we face in our daily life. There are two different points of view that can be seen to pick this problem as a project. First, since we all use taxi services in our routine and sometimes we have to wait for little long for taxi to reach our destination.

This may happen because of many reasons such as, traffic, weather, high taxi demand etc. On the other hand Taxi service providers currently has many issues regarding the prediction of duration and fare that in turns leads to severe loss and also at the cost of trust of passengers.

I selected this problem as my reasearch project so that I could contribute bit in this domain.

1.2 Executive Summary

Proposed project **"Taxi trip duration: Prediction and Analysis"** explores Machine Learning and Data Mining techniques. This project tries to achieve maximum accuracy in predicting trip duration of each and every taxi in New York so as to benefit ride-sharing cab companies, the consumers and drivers. To achieve this objective there are many constitutional parts. These parts are explained below.

First and foremost part includes data preparation. Informative and relevant featured data set builds more efficient Machine Learning model. Since the primary data set used was not sufficient to efficiently predict the ride duration. So, multiple data sets were used that includes information of traffic condition, weather and other dependent features.

Next crucial part of the project was to select the type of Machine learning algorithm that fits to predict given the type of data. In this project, three different machine learning regression models are build on the same data set. Regression techniques used are Linear regression, Random Forests and XGBoost Regression.

Linear Regression is used to study the linear relationship between a dependent variable Y and one or more independent variable X. The dependent variable Y must be continuous, while the independent variables may be either continuous, binary, or categorical.Benefits of linear regression are, it is widely used in industries, runs fast, easy to use,highly interpretable, lot of tuning is not required and it is a base for many other

methods.

Random Forest is a classier consisting of a collection of tree-structured classiers $t(x, \Theta b), b = 1, ..., B$ where the Θb are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input $\mathbf{x}$

In recent years, XGBoost Regression algorithm in the research also showed an excellent performance.Once the model was built and trained using training data set, test data set was predicted using all the three models. The next important part was to evaluate the efficiency of models and to determine the best model suitable for this data set, next parts deals with the evaluation of all the above stated models. Three different evaluation technique were used to evaluate the models. Finally this project concludes the best model suitable for predicting trip duration of each and every taxi trip in New York along with the per cent of accuracy obtained by the best model.

1.3 Contribution

Enumerate the main contributions achieved. Here try to zoom out, to talk from the perspective of a Computer Science graduate. In other words, imagine you are talking to a job panel, and you want to show your computer science skills by enumerating how they are reflected in your project work.

1.4 Structure of This Document

The goal of this thesis are following:

1. Study state-of-the-art of methods of taxi trip duration modeling
2. To select and analyse relevant and important dataset features
3. To design and implement machine learning models
4. To perform multiple experiments by tuning parameters
5. Evaluating the models

First part of this thesis will consist of basic concepts, with related literature. This should provide some insight about the problem mean, how is it currently being solved, and other relevant information.

Second part will consist of information about design and implementation of machine learning models, which should be able to train and process correct data.

Last part consists of experiments in which I use implemented machine learning models. I will talk about what methods are used, how the dataset is tested, and what results are achieved. Finally, I will summarize the work in the conclusion part.

Chapter 2

Background

2.1 Current State of the Art

To predict taxi trip duration, primary data set used are all subsets of New York City Taxi and Limousine Commissions trip data [1]. Total observations in this data set are around 1 billion taxi rides recorded between year 2009 and 2016. Original data set contains features such as pickup and drop-off locations, longitude and latitude coordinates, time and date of pickup and drop-off, ride fare, trip distance and passenger count and other less relevant variables.

In order to predict more accurately some derived features are also added in the data set that includes distance (Km), Date time features. For evaluating better fastest routes another data set is used that is, New York City Taxi with OSRM (Open Source Routing Machine) that includes multiple files such as Accidents in NY 2016, Fastest route for testing data, Fastest route for training data, Second fastest route for testing data and Second fastest route for training data

In one of the research papers Fare and Duration Prediction: A Study of New York City Rides [2], author used the same primary data set for predicting the fare and duration of taxi rides. This study used Machine Learning techniques for prediction using historical data. Models are built using only primary data set published by Limousine Commissions. Random subset of 10,000 observations from May 2016 were used. Training set included 8000 observations while validation set included 2000 observations. Author in [2] built two machine learning models Linear Regression and Random Forest. In my study I built multiple Machine Learning models.

2.1.1 Linear Regression

Regression Analysis is performed so as to determine the correlations between two or more variables having cause-effect relations, and to make predictions for the topic by using the relation. The regression using one single independent variable is called univariate regression analysis while the analysis using more than one independent variable is called multivariate regression analysis [3].

Linear Regression is used to study the linear relationship between a dependent variable Y and one or more independent variable X. The dependent variable Y must be continuous, while the independent variables may be either continuous, binary, or categorical. The initial agreement of a possible relationship between two continuous variables should always be made on the basis of scatter plot graph in order to show whether the relationship is linear or non-linear [4].

Below Figure 2.1 represents variables with linear relationships and non-linear relationships.

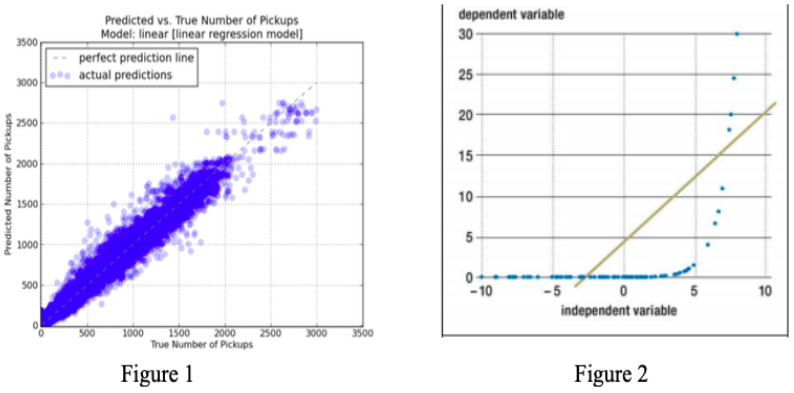

Figure 1 Figure 2

FIGURE 2.1: Linear relationship and Non-linear relationship

In the multi-variable regression model, the dependent variable is described as a linear function of the independent variables Xi , as follows:

$$\mathbf{Y = a + b1\ \ X1 + b2\ \ X2 + ... + b\_n\ \ X\_n.}$$

The model permits the computation of a regression coefficient bi for each independent variable Xi.

In my study, mean duration from training set were used to predict a constant value for the validation set. In order to select efficient covariates, author in [2] used Forward

selection to identify subset of covariates is best to use. Figure 2.2 represents the model selection. Author further used Lasso method to confirm best covariate and to shrink coefficients. Cross validation was used to find the lasso model with the lowest error and select the value of lambda. The linear regression model finds the set of the coefficients that minimize the sum of squared errors.

$y(i) = \Theta 0 + \Theta j\_x(i)$

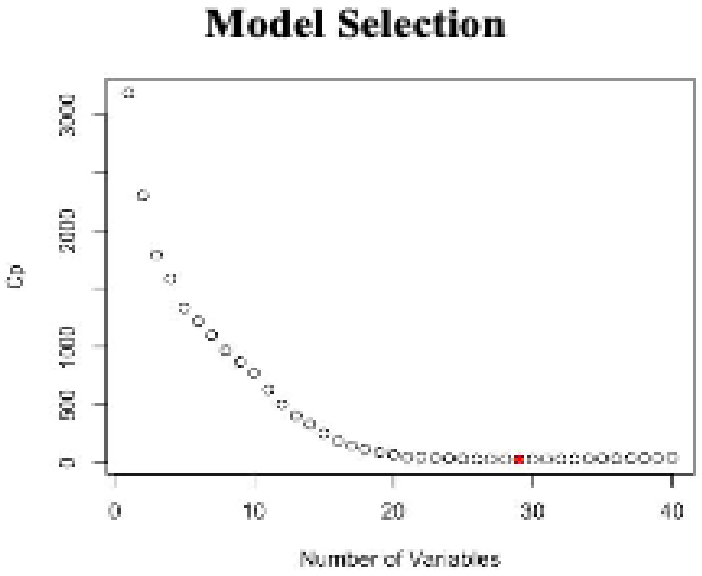

FIGURE 2.2: Model Selection

2.1.1.1 Sum of Square errors

If we model a collection of data (x1,y1), , (x_n, y_n) with a linear equation **y= mx + b** , then the residuals are the n quantities (Observed Value Predicted Value).
(y1- y1), (y2- y2), ... , (y_n- yn)

The sum of squares error (SSE) is the sum of the squares of the residuals:

SSE = (y1- y1)2 + (y2- y2)2 + + (y_n- yn)

In this model, locations effect on the trip duration is simply modeled by magnitude

of latitude and longitude coordinates. As traffic is not varying solely based on magnitude of the coordinates, Linear regression failed to account for the non-linear effect of location shave on traffic and hence trip duration.

2.1.2 Random Forest

Breiman [5] was the rst to provide a general denition of an RF and also proposed what is arguably the most popular RF algorithm, viz. Forest-RI. Concretely, Breiman denes an RF as follows:
Denition A random forest is a classier consisting of a collection of tree-structured classiers $t(x, \Theta b), b = 1, ..., B$ where the Θb are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input **x**.

The random forest algorithm aggregates many decision trees built on bootstrapped samples of the training data in order to reduce the high variance of a single decision tree and improve prediction accuracy [2]. Formally, a random forest is a predictor consisting of a collection of randomized base regression trees $rn(x, \Theta m, Dn), m1$, where$\Theta 1, \Theta 2, ...$ are i.i.d. outputs of a randomizing variable Θ [6]. These random trees are combined to form the aggregated regression estimate

$$rn(X, D\_n) = E\Theta[rn(X, \Theta, Dn)]$$

where $E\Theta$ denotes expectation with respect to the random parameter, conditionally on **X** and the data set Dn. In the following, to lighten notation a little, we will omit the dependency of the estimates in the sample, and write for example r n(X) instead of r n (X,D n). This expectation is evaluated by Monte Carlo, that is, by generating **M** (usually large) random trees, and taking the average of the individual outcomes. The randomizing variable Θ is used to determine how the successive cuts are performed when building the individual trees, such as selection of the coordinate to split and position of the split. Each of these decision trees aims to divide the predictor space, i.e. the set of all possible values for the features x1, x2, ..., x_n , in **J** distinct and non-overlapping regions R1, R2, ..., R_J. The predictor space is divided into high-dimensional rectangles, with the goal to find rectangles R1, R2, ..., R_J that minimize the RSS,

$$J\Sigma\Sigma(y(i)\_yR\_j)2j = 1i\epsilon R\_j$$

where $\_yR\_j$ is the mean response for the training observations within the jth rectangle. When building each tree, a top-down approach is taken. Beginning with all

points in the same region, the algorithm successively splits the predictor space into two halves, stopping when there are no more than five points in a region. At each split, a prediction **x_j** and cut-point s are chosen such that splitting the predictor space into the regions $x|x\_j < s$ and $x|x\_js$ leads to the biggest reduction in RSS. Defining the pair of halves as R1 (j, s) and R2 (j, s) , at each split we seek to find j and s that minimize the equation [2]. In this study Random forest outperforms all other models used and also managed the non-linearity of traffic and location effect. Below 2.3 shows the performance of Random forest in study [2].

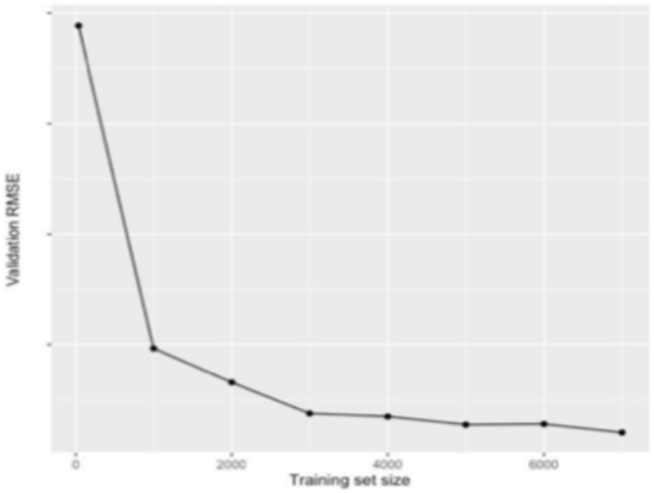

Figure 4: Random forest performance with larger training size

FIGURE 2.3: Performance of Random Forests

2.1.3 XGBoost Regression

Tree boosting is one of the most important and widely used machine learning models. Variants of the model have been applied to problems such as classification and ranking. These types of models are used by many winning solutions for machine learning challenges. They are also deployed into real world production systems, such as online advertising. Despite its great success, the existing public practice of tree boosting algorithms are still limited to million scale datasets. While there is some discussion on how to parallelize this type of algorithm, there is little discussion about optimizing system and algorithm jointly in order to build a reliable tree boosting system that handles billion scale problems. In this paper, we introduce XGBoost, a novel machine learning system that reliably scales tree boosting algorithms to billions of samples with fault tolerance

guarantees [4]. It is an ensemble technique that is capable of performing classification and regression. It makes use of multiple decision trees along with Bootstrap Aggregation which is known as bagging in layman language. Here, bagging refers to the process of training each decision tree with different samples of data. In this technique, sampling is performed with a replacement.

The idea behind this technique is to merge multiple decision trees to evaluate the final output instead of depending on the individual decision trees.

The first and most important parameter for selection is, **number of estimators**. This basically tells the number of trees in the forest. In most of the cases, higher the number of trees, better the precision. But if we increase the number of estimators, cost of time also gets increased. In every model, there is a limit beyond which increasing the number of estimators is wasteful. To determine that threshold level, I tried several numbers, in many runs.

In my model, I found that after 170 estimators, the results were constant. Limit initially was set to 300, then 250 and finally I was trying out values between 130 and 170, and 170 gave results which are satisfactory.

Next parameter considered was **min sample leaf**. It determines minimum number of samples required to be at any leaf node. I tried to build a model where this parameter was set from 10 to 100. I observed higher number of min samples resulted in unsatisfactory results. So, in the end the model was with value 25 in the leaf nodes.

Last parameter I considered was **max features**. It tells the number of features to be considered when model is looking for the best split. Here, three options were available to chose between, auto, sqrt and log2

Chapter 3

Data Analysis and Model Implementation

3.1 Dataset used

In my project, I have used datasets that are publicly open and available on the internet. The first dataset used is released by the NYC Taxi and Limousine Commission [1]. Data provided by the organization is huge, which comes up with certain disadvantages to deal with. Large dataset deliberately makes us use complete data difficult.

The given dataset is a collection of trips recorded in 6 months by the organization in 2016. There are two files in this dataset that is train.csv and test.csv. There are 1458644 records with 11 fields in the *train.csv* file while the test.csv file has 625134 records with 9 fields. Two variables missing in test file are ***trip_duration*** and ***dropoff_datetime***. The variable ***trip_duration*** is target feature, which we are trying to predict and is derived as the difference between ***dropoff_datetime*** and ***pickup_datetime***. Each record is differentiated by a unique trip id labeled as ***id***.

3.1.1 Features Description

In this section Interpretation of each feature is specified. There are 11 independent that includes id, vendor_id,*pickup_datetime*, *dropoff_datetime*, *passenger_count*, *pickup_longitude*, *pickup_latitude*, *dropoff_longitude*, *dropoff_latitude* and *store_and_fwd_flag*. The target feature is **trip_duration** which is to be predicted.

S.NO.	FEATURES	DATA TYPE	DESCRIPTION
1	id	character	A unique identifier for each trip
2	vendor_id	integer	A code indicating the provider associated with the trip record. There appears to be 2 taxi companies.
3	pickup_datetime	character	The date and time when the meter was engaged. This is currently a combination of date and time.
4	dropoff_datetime	character	The date and time when the meter was disengaged. This is currently a combination of date and time.
5	passenger_count	integer	The number of passengers in the vehicle (driver entered value).
6	pickup_longitude	numeric	The longitude where the meter was engaged. These are geographica coordinates and appear to be in correct format.
7	pickup_latitude	numeric	The latitude where the meter was engaged.
8	dropoff_longitude	numeric	The longitude where the meter was disengaged
9	dropoff_latitude	numeric	The latitude where the meter was disengaged.
10	store_and_fwd_flag	character	The flag indicates whether the trip record was held in vehicle memory before sending to the vendor because the vehicle did not gave a connection to the server.
11	trip_duration	integer	The duration of the trip in seconds.

FIGURE 3.1: Feature Description

3.1.2 Derived Features

To inject expert knowledge into the training process, I inserted a few derived features. Derived features are defined as features which are formulated from one or more features given in the dataset. There are 6 derived features labelled as *Weekday*, *month*, *weekday_num*, *pickup_hour*, *haversine_distance* and *speed*.

Weekday, *month*, weekday_num and *pickup_hour* features are derived from pickup_datetime. **Weekday** is of object datatype that contains weekday names (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday) as values. **Month** is of integer datatype that determines the month and has values ranging between 1 to 6 (6 months) because the dataset used is a collection of trips from January 2016 to June 2016. **Weekday_num** is of integer datatype that determines the day of a week and has values ranging between 0 to 6 (7 days of a week). Pickup_hour is of integer type that signifies the hour of the day and it has values from 0 to 23 (24 hours).

3.1.2.1 Haversine distance

It is derived from 4 features pickup_latitude, pickup_longitude, dropoff_latitude and dropoff_longitude. It is of float datatype. It determines the distance covered by taxi and I have used haversine formula to calculate distance because distance in this dataset can be only be calculated using geographical coordinates. Haversine formula determines the great-circle distance between a pair of points on a sphere given their longitudes and latitudes [7]. Let the central angle Θ between any 2 points on a sphere be: $\Theta = d/r$
Where, **d** is the distance between the two points and **r** is the radius of the sphere.
The haversine formula is shown in figure 3.2:

$$Hav\,\theta = \; hav(\varphi 2 - \varphi 1) + cos\,(\varphi 1)\,cos\,(\varphi 2)\,hav\,(\gamma 2 - \gamma 1)$$

Where,
$\varphi 1, \varphi 2$: latitudes of point 1 and point 2
$\gamma 1, \gamma 2$: longitudes of point1 and point 2
$hav(\theta)$: This is given by,

$$hav(\theta) = \; sin^2(\theta\,/2) = (1 - cos\,\theta)/2$$

FIGURE 3.2: Haversine formula

3.1.2.2 Speed

It is derived from haversine_distance and trip_duration features. It is of float datatype. It determines the speed of the taxi calculated using a formula,

Speed = Distance / Time

Where,**distance** is haversine_distance and **time** is trip_duration

3.2 What can be discerned visually from the data

In this section, I would like to talk about what can be discerned visually by the human eye in the dataset. The main objective is to check if there is any replicate pattern, significant differences, or critical features that could be recognized. To understand each feature, I have accomplished univariate analysis and to extract the correlation between variables, I experimented multivariate analysis.

3.2.1 Univariate Analysis

The univariate analysis is defined as the analysis of single feature at a time. In this section I have explored 5 features listed as *passenger_count*, *vendor_id*, *haversine_distance*, *trip_duration*, and *speed*.

3.2.1.1 Passenger_count

The first feature I considered is passenger_count. I have plotted a boxplot graph and summary for this feature (3.3 and 3.4). According to NYC Limousine Commission maximum limit of taxi passenger is 6 (5 adults and 1 child).
It is visible that there are trips where the passenger counts are 0 or more than 6 which is not correct. So, the trips having passenger count more than 6 and 0 are outliers and

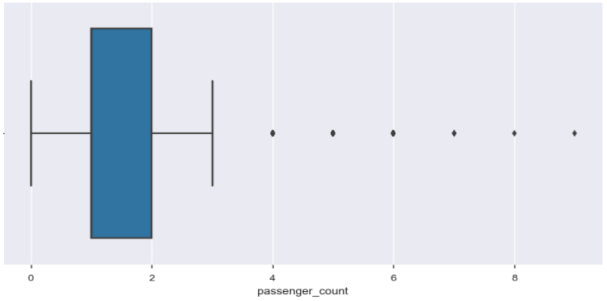

FIGURE 3.3: Passenger count

```
In [47]: train_data.passenger_count.describe()
Out[47]:
count    1.458644e+06
mean     1.664530e+00
std      1.314242e+00
min      0.000000e+00
25%      1.000000e+00
50%      1.000000e+00
75%      2.000000e+00
max      9.000000e+00
Name: passenger_count, dtype: float64
```

FIGURE 3.4: Summary Passenger count

need to be handled. I have replaced trips with passenger count 0 to 1 and removed trips with passenger count more than 6. The graph in figure 3.5 shows the number of trips with passenger_count.

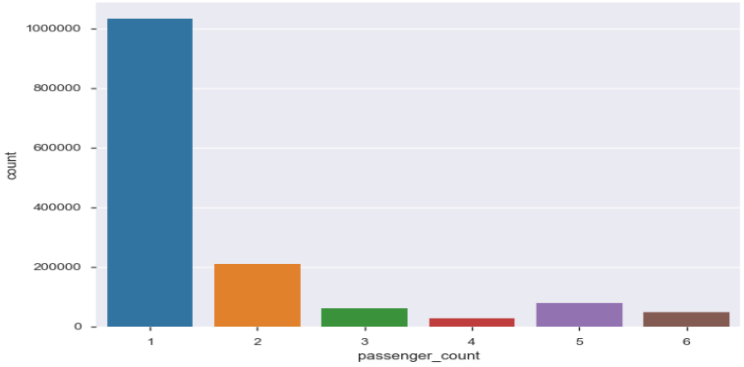

FIGURE 3.5: Trip count v/s Passenger count

3.2.1.2 Vendor_id

Second feature is **vendor_id**. There are 2 vendors, Vendor 1 and Vendor 2 providing taxi services. Number of taxi trips recorded by each vendor is shown in figure 3.6.
As it is visible from the graph, Vendor 2 is more popular than vendor 1 as number of trips carried by vendor 2 is more than as compared to trips carried by vendor 1.

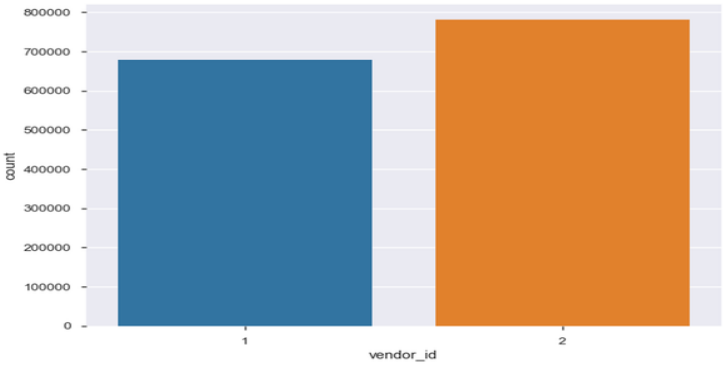

FIGURE 3.6: Vendor id

3.2.1.3 Haversine distance

Haversine distance is visually expressed using boxplot. There are few observations noted. In the Figure 3.7, it can be visually seen that maximum trips are between 0-10 Km. There are few trips above 100Km. Standard deviation is 4.30 so, maximum trip has distance between 0-10Km. To obtain better results I have considered trips with distance more than 100 Km as outliers and have not considered them in the final dataset.

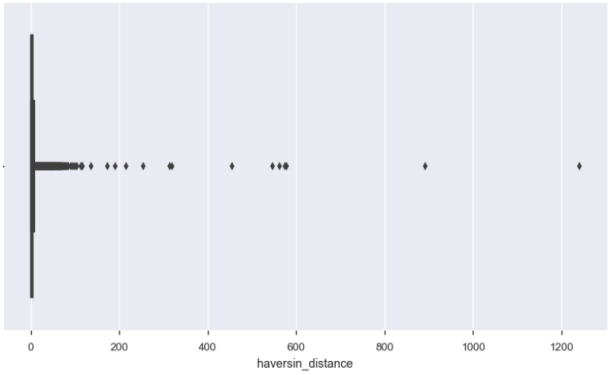

FIGURE 3.7: Haversine distance

After treating outliers, number of trips considered are shown below. The Figure 3.8 shows the graph between haversine_distance and number of trips.

3.2.1.4 Trip duration

To describe trip duration, I have plotted boxplot. There are few outliers detected in this feature as shown in 3.9.

From the above figure it is observed that there are few trips whose duration is beyond 86400 seconds i.e. 24 hours which are clearly stated as outliers. To deal with this, I

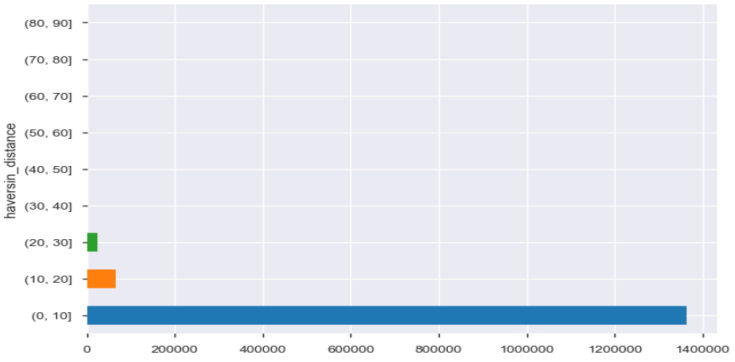

FIGURE 3.8: Number of trips v/s Haversine distance

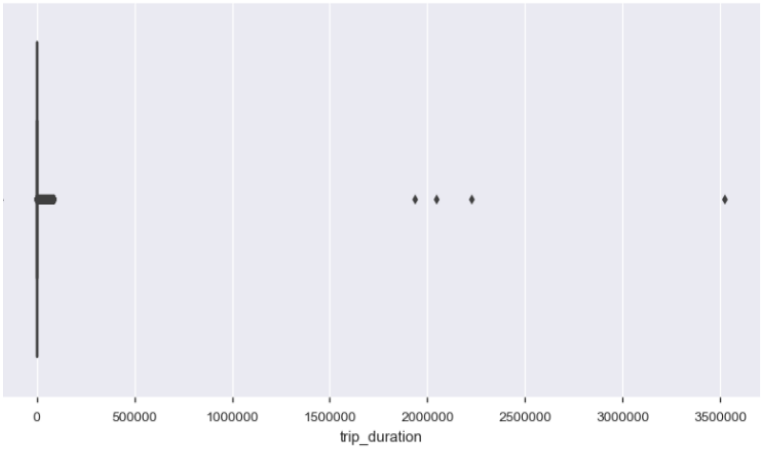

FIGURE 3.9: Trip duration

have considered trips only with duration less than 24 hours. I took a logarithimic value of trip duration so as to normalize the scale.

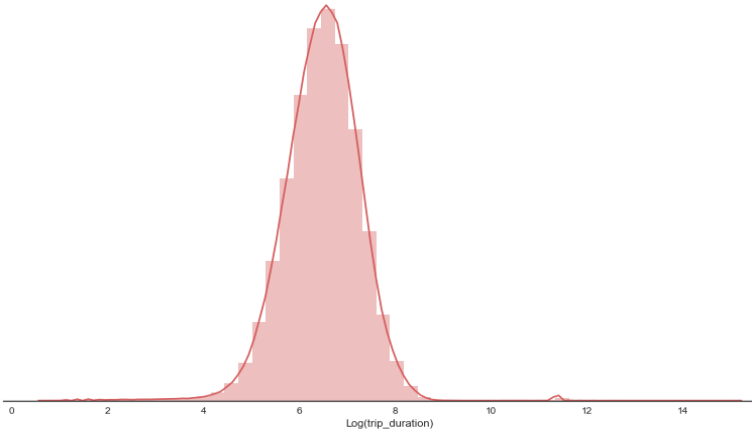

FIGURE 3.10: Log(Trip duration)

3.2.1.5 Speed

Maximum speed limit defined by NYC Limousine Commission is 25 mph in urban areas that is 40 Km/h and 65 mph on controlled state highways that is approximately 104 Km/h. In figure 3.11 , it is observed that there are trips where speed is going beyond 220 Km/h which is not acceptable and should be treated as outliers.

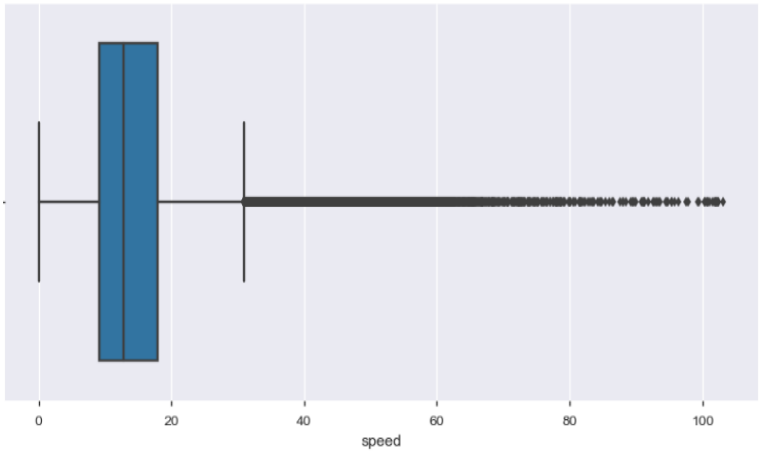

FIGURE 3.11: Speed

After treating outliers, it is observed that the speed of maximum trips is between 10-20 Km/h as shown in figure 3.12.

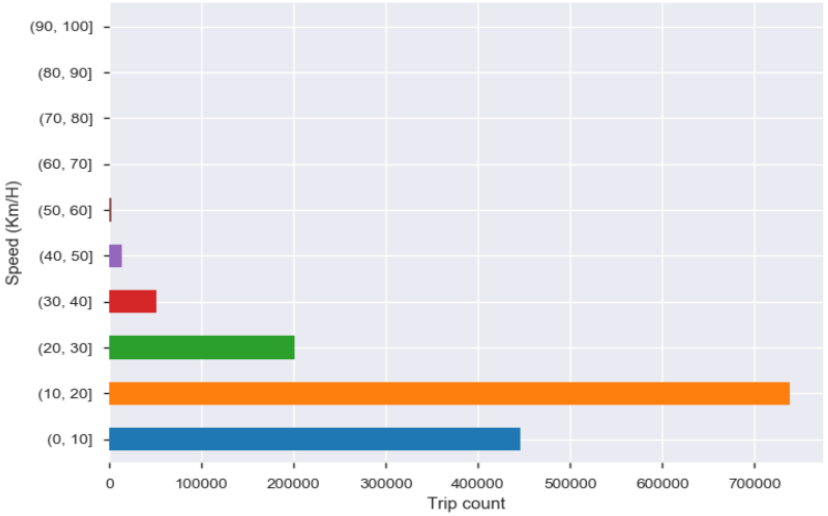

FIGURE 3.12: Speed v/s Trip count

3.2.2 Multivariate Analysis

Multivariate analysis is a statistical technique that determines the impact of one or more than one variable over the other variable. It involves the analysis of more than one

variable at a time. In this section I have explore the impact of independent features such as *pickup_hour*, *weekday*, *month*, *vendor_id*, *flag*, *haversine_distance* over target feature, *trip_duration*. Apart from this I have visualized the pattern of *Total trips per hour*, *Total trip per week* and *haversine_distance per hour*.

3.2.2.1 Total trips per hour

In this section I have explore the trend of number of trips in a whole day, from morning till night that is of 24 hours. It is observed that there is general inclination of taxi trips increasing from morning 5 AM till 8 PM. After 8 PM demand of taxi is started declining which is generally expected as shown in figure 3.13

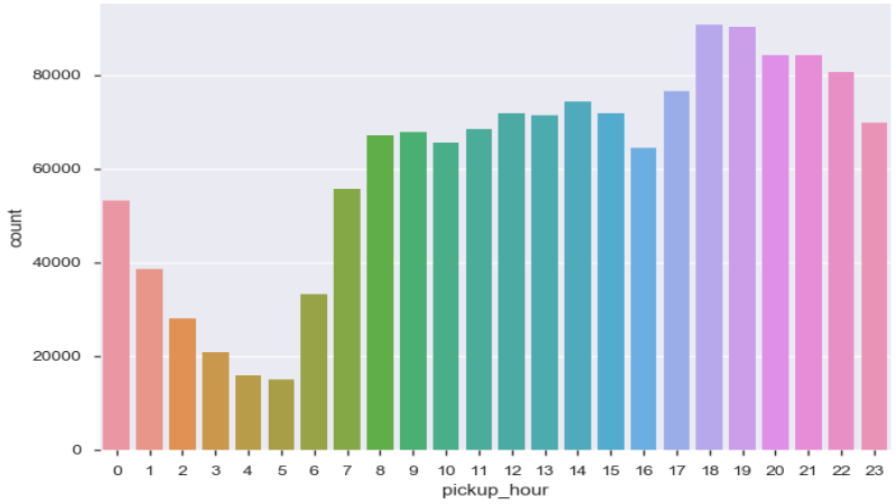

FIGURE 3.13: Total trips per hour

3.2.2.2 Total trips per week

To visualize the pattern of number of trips take place on every day of a week, I plotted a graph as shown in figure 3.14 where x-axis represents weekdays (0-Sunday, 1-Monday, ..., 6 - Saturday). It is observed that trip count is increasing from Monday till Friday and on weekend trip counts are less as compared to working days.

3.2.2.3 Trip duration per hour

It determines the average duration of the trip every hour. I plotted the graph showing the trip duration (seconds) for 24 hours. It is observed from the figure 3.16 that there is

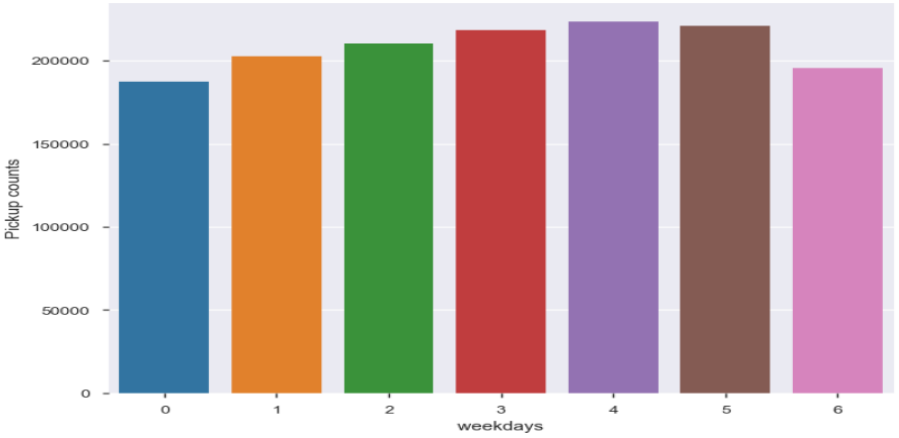

FIGURE 3.14: Total trips per Week

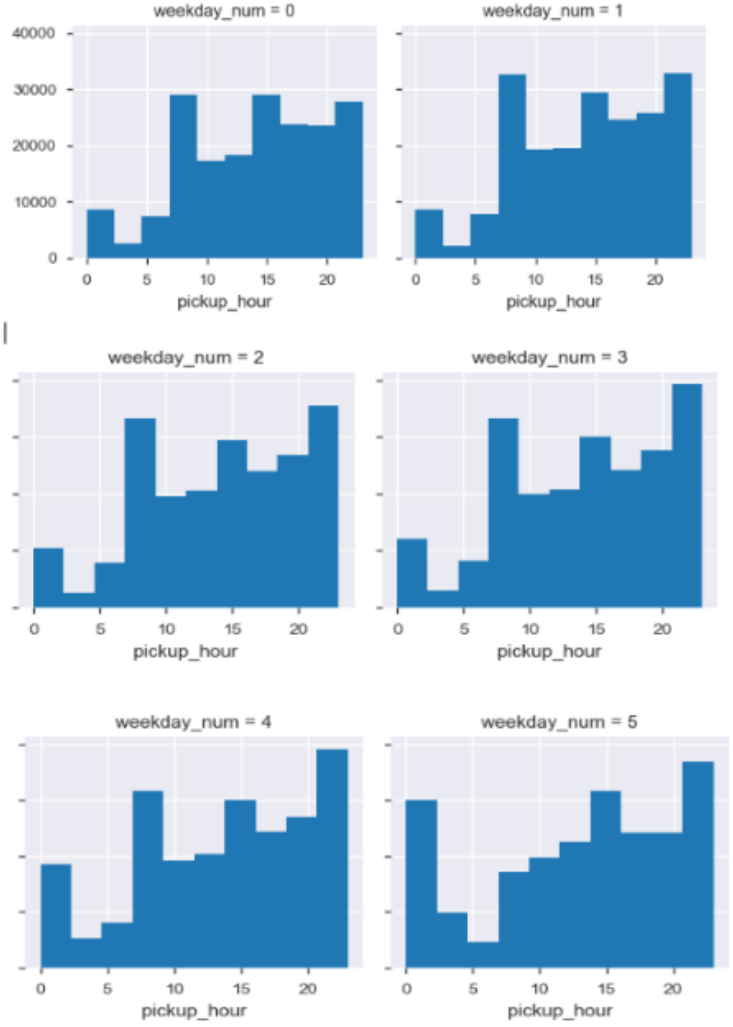

FIGURE 3.15: Total trips for each Weekday

a steep inclination in the duration of trips from 6:00 a.m. to 8:00 a.m. in the morning. Then again duration is increasing from 11:00 a.m. till 4:00 p.m. in the evening. After

this duration is declining till midnight.

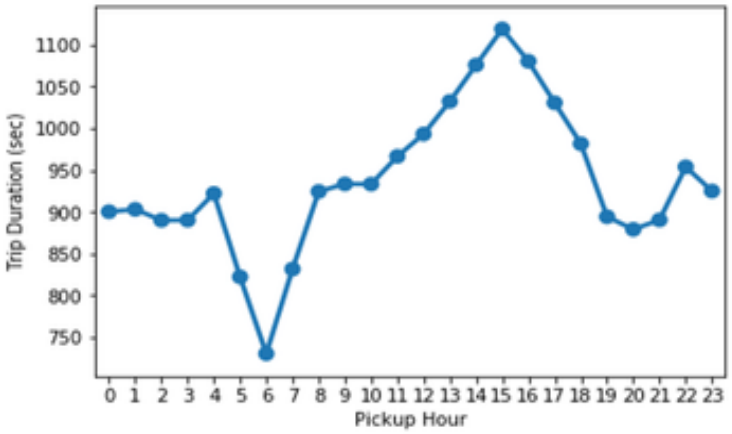

FIGURE 3.16: Trip duration for every Hour

3.2.2.4 Trip duration per week

It tells us the pattern of trips duration in a whole week. It is visualized that the duration trips are increasing from Monday till Friday. As these are workdays so this was expected. From Friday trips duration is declining at a high rate that is on weekends there are usually short trips.

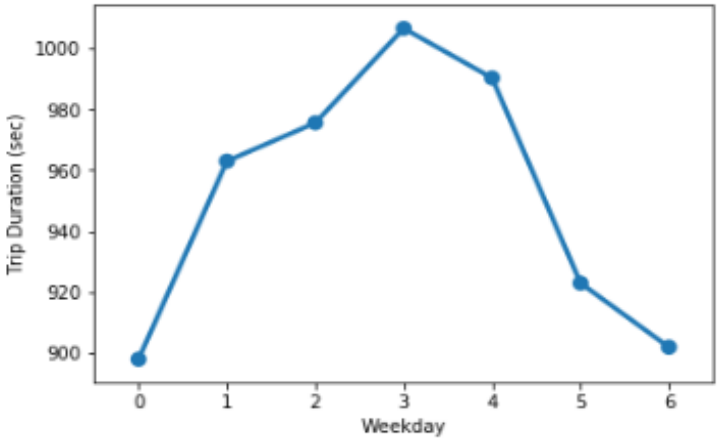

FIGURE 3.17: Trip duration per week

3.2.2.5 Trip duration per month

This helped to get an idea of the pattern of trips in 6 months that is from January to June. January and February cover the winter season and the duration of trips are very less. After winter comes Spring (March, April, May) and there is a steep rise in the

duration of trips. Also, in Summer (June) trips duration are maximum. So, it can be concluded that longer trips in New York occurs in Summer and then it gradually starts decreasing as winter arises.

To get more information from this combination of features, I plot individual graph for each day weekday and for each hour showing trips count as shown in figure 3.18.

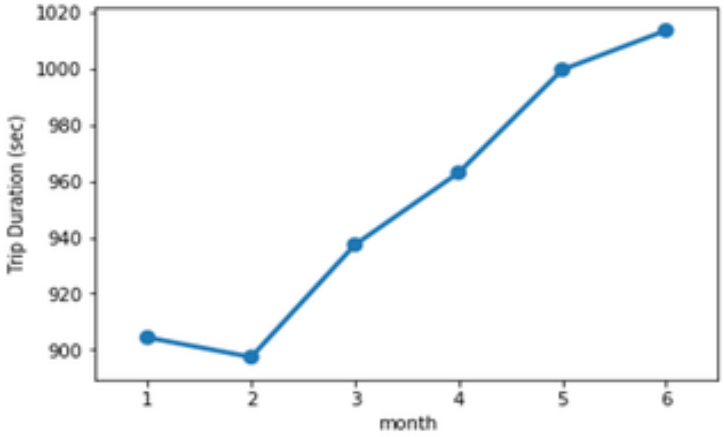

FIGURE 3.18: Trip duration per month

3.2.2.6 Trip duration per Distance

It is expected that there should have been a linear relationship between the distance covered and trip duration on an average but as we can see there is a dense collection of trips in the lower right corner which determines there are many trips with inconsistent readings.

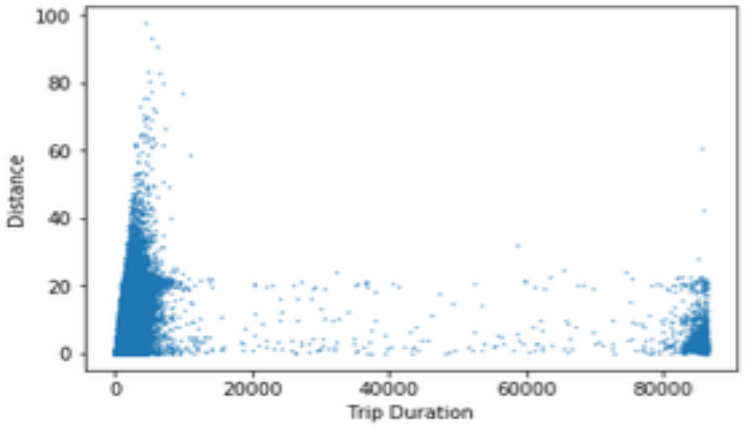

FIGURE 3.19: Trip duration per Distance

We should remove those trips which covered 0 Km distance but cloaked more than

a minute to make our data more consistent for a predictive model. If the trip was canceled after booking, then that should not have taken more than a minute. This is what I assumed to make our model more realistic. After removing the unreal trips, I plotted the graph as shown in figure 3.20.

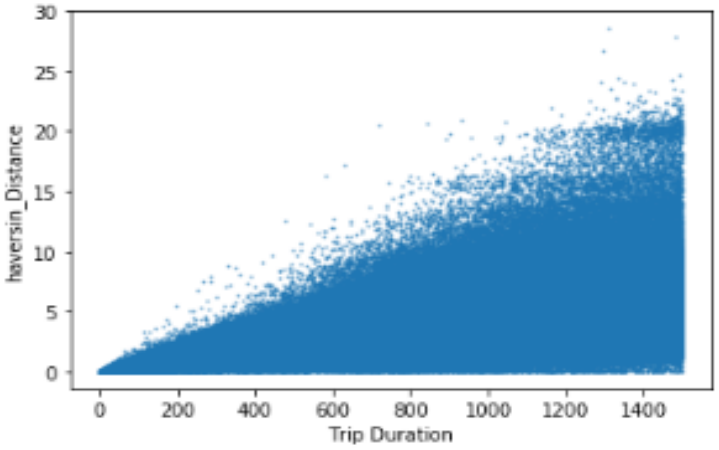

FIGURE 3.20: Trip duration per Distance(without outliers)

3.2.2.7 Traffic heatmap

In this section I have plotted a map showing the traffic situation in New York. Red points shows pick locations and yellow points shows dropoff locations. Thus, from this we can observe the traffic situation in New York as shown in figure 3.21.

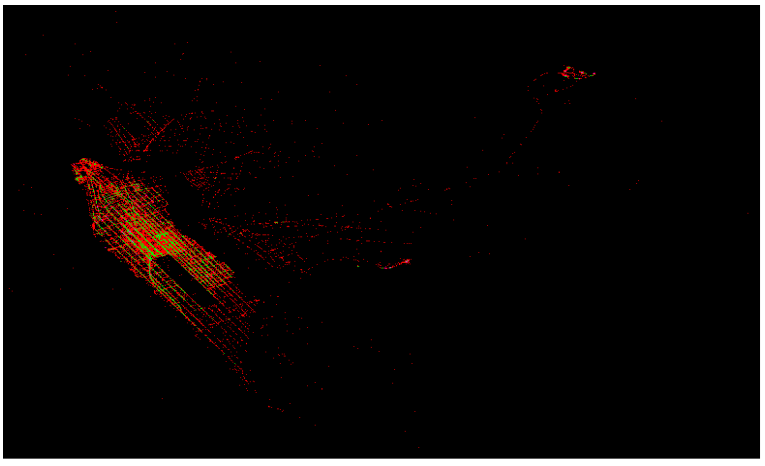

FIGURE 3.21: Traffic heatmap

3.3 Model Implementation

This section includes implementation of regression models that I have used to predict trip duration and its implementation. The implementation is programmed in Python language 3.6.1, in the Scientific Python Development Environment Spyder. For model implementation, I used open source libraries for Machine Learning in Python, scikit-learn.

3.3.1 Feature Engineering

The first phase of implementation is dedicated to **Feature engineering**. It is important to make sure that all of the data used is relevant and correct. To build an efficient model I performed Feature Engineering more specific Feature selection.There are two ways to do Feature engineering, Feature Selection, and Feature Extraction.

3.3.1.1 Feature Selection

Feature Selection is also known as Variable selection. It is a dimensionality reduction technique defined as a method that requires a search strategy to select the subset of attributes and an objective function to evaluate the selected subset of features [8]. Feature Selection is a very critical component in a model workflow. When the given dataset is of high dimensionality, models usually choke because Training time increases exponentially with the number of features also models have an increased risk of overfitting with an increasing number of features.

Feature Selection methods help with these problems by reducing the dimensions without much loss of the total information. It also helps to make sense of the features and its importance. In my project I have used Backward Elimination method to perform feature selection.

Backward Elimination Technique
In my dataset, there are 62 features and I have used the Backward elimination technique. This methodology eliminates worst feature variables on a specific model one at a time till the most effective set of options are renowned. In order to select the best features to train my model, the level of significance is set to 0.05 that is 5% which means features whose p-value goes below 0.05 will be removed one at a time. Results obtained after Backward elimination are displayed as OLS summary shown in the figure 3.22.

OLS Regression Results

Dep. Variable:	y	R-squared:	0.034
Model:	OLS	Adj. R-squared:	0.034
Method:	Least Squares	F-statistic:	6488.
Date:	Thu, 23 May 2019	Prob (F-statistic):	0.00
Time:	15:14:23	Log-Likelihood:	-1.3648e+07
No. Observations:	1453814	AIC:	2.730e+07
Df Residuals:	1453805	BIC:	2.730e+07
Df Model:	8		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	319.0693	5.544	57.550	0.000	308.203	329.936
x1	135.3696	0.606	223.285	0.000	134.181	136.558
x2	185.4508	4.805	38.593	0.000	176.033	194.869
x3	22.3190	7.119	3.135	0.002	8.367	36.271
x4	50.0954	7.156	7.000	0.000	36.070	64.121
x5	77.2956	7.189	10.752	0.000	63.206	91.385
x6	89.1818	7.331	12.165	0.000	74.814	103.550
x7	31.5219	7.030	4.484	0.000	17.743	45.301
x8	46.7897	6.927	6.755	0.000	33.213	60.367

Omnibus:	3656024.154	Durbin-Watson:	2.000
Prob(Omnibus):	0.000	Jarque-Bera (JB):	40753758989.288
Skew:	28.273	Prob(JB):	0.00
Kurtosis:	821.277	Cond. No.	24.3

FIGURE 3.22: OLS summary

3.3.2 Machine Learning models

For this task, I tried three different machine learning models. Linear Regression, Random Forest Regression and XGBoost Regression. Results obtained from all these models varies at high scale and this helped me in getting clear understanding of which constraints leads to better outcome or prediction. I decided to explore each one of them with different parameters to obtain best possible result.

3.3.2.1 Linear Regression

Linear Regression is commonly used for predictive analysis. It is used to determine the linear relationship between a dependent variable Y and one or more independent variables X. The dependent variable must be continuous, while the independent variable may be either continuous, binary or categorical. The simplest form of the regression equation with one dependent and one independent variable is defined by the formula, $\mathbf{y = c + b*x}$, where

$\mathbf{y}$ = estimated dependent variable score,

$\mathbf{c}$ = constant,

$\mathbf{b}$ = regression coefficient,

and $\mathbf{x}$ = score on the independent variable.

The initial judgment of a possible relationship between two continuous variables should

always be made on the basis of a scatter plot.[4]. In case of Multiple Linear Regression following equation as shown in figure 3.23 works.

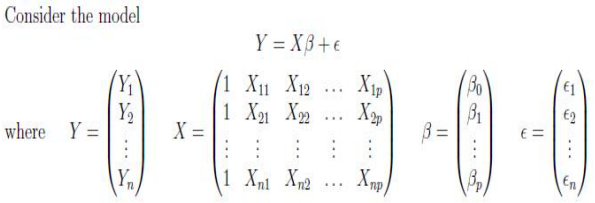

FIGURE 3.23: Multiple Linear Regression

Assumptions I have made while running this model are, Probability distribution means of the error is 0, Probability distribution variance of the error term is constant, Error term associated with one of the values of the independent variable has no effect on any of the values of the error associated with any other independent value.

3.3.2.2 Random Forests Regression

It is an ensemble technique that is capable of performing classification and regression. It makes use of multiple decision trees along with Bootstrap Aggregation which is known as bagging in layman language. Here, bagging refers to the process of training each decision tree with different samples of data. In this technique, sampling is performed with a replacement.

The idea behind this technique is to merge multiple decision trees to evaluate the final output instead of depending on the individual decision trees.

The first and most important parameter for selection is, **number of estimators**. This basically tells the number of trees in the forest. In most of the cases, higher the number of trees, better the precision. But if we increase the number of estimators, cost of time also gets increased. In every model, there is a limit beyond which increasing the number of estimators is wasteful. To determine that threshold level, I tried several numbers, in many runs.

In my model, I found that after 170 estimators, the results were constant. Limit initially was set to 300, then 250 and finally I was trying out values between 130 and 170, and 170 gave results which are satisfactory.

Next parameter considered was **min sample leaf**. It determines minimum number

of samples required to be at any leaf node. I tried to build a model where this parameter was set from 10 to 100. I observed higher number of min samples resulted in unsatisfactory results. So, in the end the model was with value 25 in the leaf nodes.

Last parameter I considered was **max features**. It tells the number of features to be considered when model is looking for the best split. Here, three options were available to chose between, auto, sqrt and log2 shown in figure 3.24.

$$auto: \max_{features} = n\,features$$

$$sqrt: \max_{features} = sqrt(n)\,features$$

$$log2: \max_{features} = log2(n)\,features$$

FIGURE 3.24: Options for selecting Max features parameter

3.3.2.3 XGBoost Regression

XGBoosting stands for eXtreme Gradient Boosting. This technique is basically an implementation of gradient boosting decision trees which are designed specifically to improve speed and performance of the model.

Three major forms of gradient boosting supported by this technique are Gradient Boosting also known as gradient boosting machine, Stochastic Gradient Boosting along with sub-sampling at the column, row and column per split levels, and Regularized Gradient Boosting. That includes both L1 and L2 regularization.

The First parameter I considered was **n_estimators**, It is expected that more the number of estimators better is the result. I tried multiple values for this from 50 to 200, but in my case by keeping the value to 100 result obtained was satisfactory.

Next parameter was **learning rate**. This parameter basically makes the model more robust by diminishing weights on each step. Standard value varies between 0.05 to 0.3. I have used learning rate as 0.08.

The third parameter was **gamma**. When nodes split then there is a positive reduction in the loss function. Gamma determines the minimum loss reduction and its value

can vary depending on the loss function used and it should be tuned.

Next is **subsample**, it defines in which fraction observations will be treated as random samples for each tree. In this case, lower the value, lesser are the chances of over-fitting and at the same too small value can lead to under-fitting. I have used 0.75.

The last parameter I considered was **max_depth**. It is used to control over-fitting as higher the depth, model learns the relations more specific but higher value requires cross-validation to be performed. In my case, I used 7 as max_depth.

3.3.3 Evaluation metrics

In order to test the predictions made by the above stated model I have used R2 scoring, Mean Absolute Percentage Error and Root Mean Squared Error. To perform comparative analysis of all the three models I have used these three common metrics to evaluate the predictions made by each model.

Chapter 4

Experiments and Evaluation

This section includes implementation of regression models that I have used to predict trip duration and its implementation. The implementation is programmed in Python language 3.6.1, in the Scientific Python Development Environment Spyder.

4.1 Architecture

For model implementation, I used open source libraries for Machine Learning in Python, scikit-learn. The packages I imported to build my models are listed below.

- NumPy, It is the foundation package for performing computation in Python. It consists of a powerful N-dimensional array object, broadcasting features, tools for integrating Fortran, C/C++ code.
- Pandas, It is an open source library that provides easy to use data structures and also provides efficient data analysis for Python language.
- seaborn, Matplot Library, Haversine
- Linear model for Linear Regression
- Ensemble model for Random Forests Regression
- Model selection for spliting train and test data
- Metrics for R2 score

4.2 Types of experiments and evaluation

There was some room for different types of experiments taking availability of data set into consideration. In my data set there are two files training and test data file. In all the experiments I have considered training file and later splitting into training and test data. Training model with split train data and testing it with split test data.

Other things that need to be considered during experiments was use of features. For each model I have performed experiments where model were initially trained on all the features and later, all the three models were trained on features resulted after processing Feature Selection.

In total, there are total 8 experiments that I have performed. Experiment 1 and Experiment 2 are building Linear Regression model, Experiment 3,4 and 5 builds Random Forests model and, Experiment 6 and 7 builds XGBoost Regression model.

For evaluation I decided to use three metrics by which I compared my predictions. Metrics used were R2 score, Mean Absolute Percentage Error and Root Mean Squared Error. Also, for each experiment I displayed the predicted trip duration showing how close or far the predicted value is with the actual duration.

4.2.1 Experiments

4.2.1.1 Experiment 1

I started with simple experiments, where I trained Linear Regression model with all the features to observe the efficiency of the model and the result were recorded. I fit the model with training data and validate the model with test data with default parameters. The results obtained from the model is shown in Table 4.1 which shows the value of R2 score, MAPE, and RMSE.

Metrics	Experiment 1
RMSE	0.5041
MAPE	5.92%
R2 score	0.60

Table 4.1: RESULTS: Experiment 1

The Table 4.2 shows predicted trip duration and the observed trip duration of first

five instances of test data.

Instances	Observed value	Predicted value
0	477.0	542.71
1	759.0	620.46
2	422.0	480.23
3	995.0	685.99
4	255.0	450.23
5	1755.0	999.86
6	485.0	551.98

TABLE 4.2: Experiment 1: Predicted and Observed value

4.2.1.2 Experiment 2

In this section, I trained the Linear Regression model with selected features obtained after feature selection which are eight features. The result obtained is shown in Table 4.3 with all the metrics.

Metrics	Experiment 2
RMSE	0.6148
MAPE	7.53%
R2 score	0.40

TABLE 4.3: RESULTS: Experiment 2

In Table 4.4, predicted and observed trip duration are displayed showing the accuracy of the model.

Instances	Observed value	Predicted value
0	477.0	504.83
1	759.0	602.53
2	422.0	515.21
3	995.0	585.31
4	255.0	469.02
5	1755.0	687.92
6	485.0	489.56

TABLE 4.4: Experiment 2: Predicted and Observed value

4.2.1.3 Experiment 3

In this experiment, first Random Forests model was build. I started with simple model by training complete data set including all the features. the results were recorded as shown in Table 4.5.

Metrics	Experiment 3
RMSE	0.030
MAPE	0.06%
R2 score	1.00

TABLE 4.5: RESULTS: Experiment 3

The table 4.6 shows predicted trip duration and the observed trip duration of first five instances of test data obtained from experiment 3.

Instances	Observed value	Predicted value
0	477.0	504.83
1	759.0	602.53
2	422.0	515.21
3	995.0	585.31
4	255.0	469.02
5	1755.0	687.92
6	485.0	489.56

TABLE 4.6: Experiment 3: Predicted and Observed value

4.2.1.4 Experiment 4

In this experiment, Random Forests model was build.This time I trained model with eight selected features of the data set. Parameters in this model were taken as default and the results were recorded as shown in Table 4.7.

Metrics	Experiment 4
RMSE	0.561
MAPE	6.67%
R2 score	0.50

TABLE 4.7: RESULTS: Experiment 4

The table 4.8 shows predicted trip duration and the observed trip duration of first five instances of test data obtained from experiment 4.

Instances	Observed value	Predicted value
0	477.0	236.99
1	759.0	744.23
2	422.0	475.89
3	995.0	789.63
4	255.0	341.39
5	1755.0	855.95
6	485.0	385.63

TABLE 4.8: Experiment 4: Predicted and Observed value

4.2.1.5 Experiment 5

In this experiment also another Random Forests model was build. Model was trained with selected eight features. Parameters in this model were set as,
number of estimators, n_estimator = 200
maximum depth of tree, max_depth= 22
sample split, min_sample_split = 9
The results obtained were recorded and is shown in Table 4.9.

Metrics	Experiment 5
RMSE	0.48
MAPE	5.74%
R2 score	0.62

TABLE 4.9: RESULTS: Experiment 5

The table 4.10 shows predicted trip duration and the observed trip duration of first five instances of test data obtained from experiment 5.

Instances	Observed value	Predicted value
0	477.0	450.04
1	759.0	733.27
2	422.0	507.71
3	995.0	756.20
4	255.0	335.79
5	1755.0	996.36
6	485.0	400.59

TABLE 4.10: Experiment 5: Predicted and Observed value

4.2.1.6 Experiment 6

In this experiment, first XGBoost Regression model was build. Model was trained with all the features. Parameters in this model were set as,
number of estimators, n_estimator = 100
learning rate,learning_rate= 22
sample split, subsample = 0.75,
maximum depth, max_depth = 7,
colsample_bytree= 1
The results obtained were recorded and is shown in Table 4.11.

Metrics	Experiment 6
RMSE	0.02
MAPE	0.17%
R2 score	0.98

TABLE 4.11: RESULTS: Experiment 6

The Table 4.12 shows predicted trip duration and the observed trip duration of first five instances of test data obtained from experiment 6.

Instances	Observed value	Predicted value
0	477.0	475.49
1	759.0	769.59
2	422.0	425.16
3	995.0	910.79
4	255.0	254.37
5	1755.0	1753.05
6	485.0	483.08

TABLE 4.12: Experiment 6: Predicted and Observed value

4.2.1.7 Experiment 7

In this experiment, another XGBoost Regression model was build. Model was trained with selected eight features. Parameters in this model were set as,
number of estimators, n_estimator = 100
learning rate,learning_rate= 22
sample split, subsample = 0.75,
maximum depth, max_depth = 7,
colsample_bytree= 1

The results obtained were recorded and is shown in Table 4.13.

Metrics	Experiment 7
RMSE	0.48
MAPE	5.64%
R2 score	0.63

TABLE 4.13: RESULTS: Experiment 7

The Table 4.14 shows predicted trip duration and the observed trip duration of first five instances of test data obtained from experiment 7.

Instances	Observed value	Predicted value
0	477.0	489.53
1	759.0	770.04
2	422.0	493.89
3	995.0	708.83
4	255.0	326.63
5	1755.0	986.40
6	485.0	430.33

TABLE 4.14: Experiment 7: Predicted and Observed value

To visualize the evaluation metrics of all the above Experiments and to get better understanding of working of the model, I plotted three different bar graphs showing the all the three metrics for all the different models I experimented.
In figure 4.1, R2 scores of all the experiments are presented. The outcome of this metric indicates that Experiment 6 shows the best result as it is having variance score close to 1 which is satisfactory. Apart from this, experiment 5 and 7 have variance score above 0.5 which is acceptable and also there is less over-fitting in these cases. In figure 4.2, Root Mean Squared Error for all the experiments are presented. The outcome of this metric indicates that Experiment 6 shows the best result as it is having minimum RMSE which is satisfactory. Apart from this, experiment 5 and 7 having RMSE value which is acceptable.
In figure 4.3, Mean Absolute Percentage Error for all the experiments are presented. The outcome of this metric indicates that Experiment 6 shows the best result as it is having minimum MAPE, which is satisfactory. Apart from this, experiment 5 and 7 having MAPE value which is acceptable.

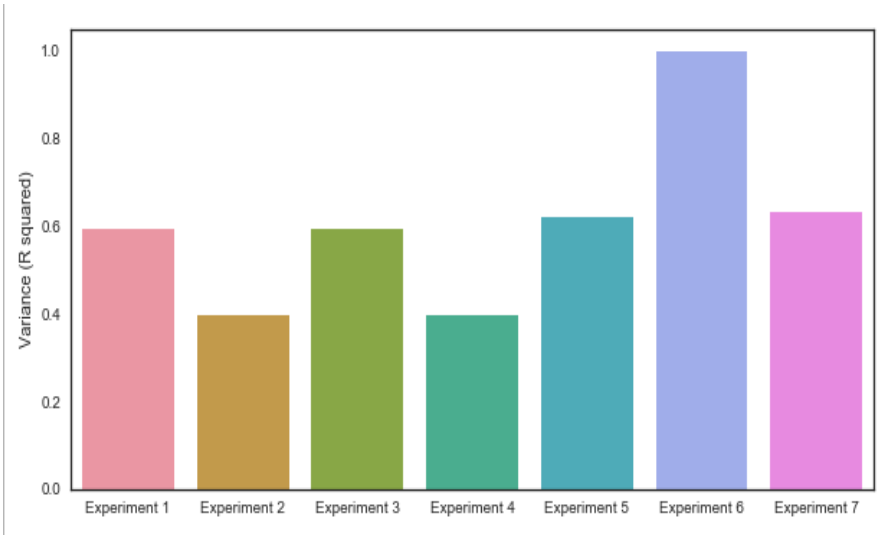

FIGURE 4.1: Results: R2 Scores

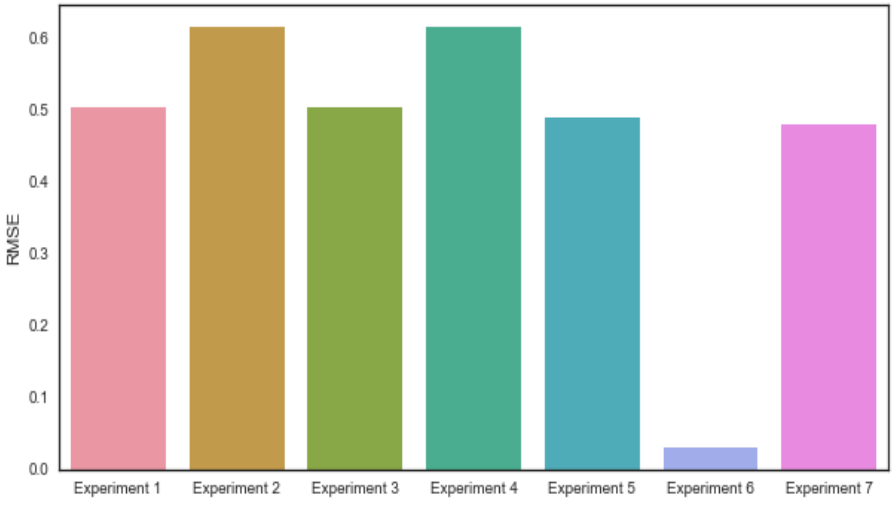

FIGURE 4.2: Results: Root Mean square Error

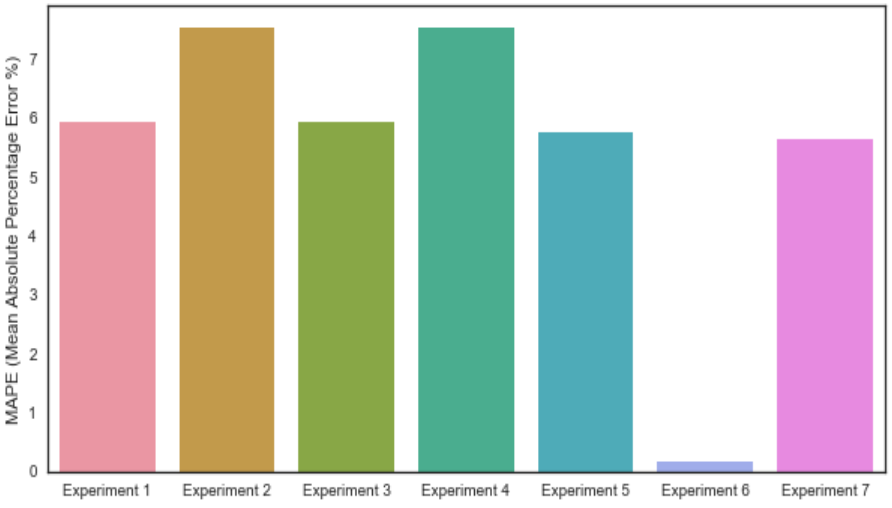

FIGURE 4.3: Results: Mean Absolute Percentage Error

Chapter 5

Conclusions and Future Work

5.1 Discussion

The objective of my thesis is to predict trip duration of taxis in New York. To complete my objective there were many hurdles I faced. Initially I was struggling with the data set collection. Initially the data set available had many missing value, to deal with that I refer to other New York traffic sources available online.

Next, while examining the data set features and dealing with the traffic features there were not enough information that could help me out to achieve my objective. In the initial phase of my project I was working on Support Vector Regression machine learning algorithm to build a model, but because the data set available was large and has set of files, the model was not working efficiently.

Later, I switched to XGBoost Regression model, which worked really well to achieve my objective. Finally I built three different models and all the models somehow were able to achieve the goal.

5.2 Conclusion

In this work, methods of predicting trip duration modeling were introduced, briefly shown how they work and what they do. Available data sets were thoroughly examined, and explained, trying to describe all important and relevant features and patterns that can be discerned by a human.

The objective of this thesis was to design and create a data-driven model of predicting taxi trip duration and its analysis. Features and attributes were carefully examined and after careful examination, I was able to design and implement the data processing model, with a wide selection of learning techniques and parameters. Along with this, traffic data was implemented, in an effort to improve the efficiency of the model to predict the taxi trip duration.

Numerous experiments and their evaluation shows, that there is a possibility to predict approximate taxi trip duration. If it is considered that the prediction does not have to be exactly the same or 100 percent correct, considering the difference in pickup and drop-off latitudes and longitudes, the results seem much better. In my view, the predictions made could be improved by adding more features, like Weather, accident details, where it could learn patterns each trip duration makes. Also considering traffic data available, it was limited only to the general history, this would have obstructed the learning process and model capability.

To summarize, Random Forests and XGBoost Regression models worked really well as compared to the Linear Regression model for solving this complex problem, but it is not fully used to transfer to different areas because of some missing and important features as discussed earlier. It can work really work by adding some new features from data sets which can improve its learning process.

5.3 Future Work

Acknowledging all the work that is done and that is not done for in the models built in this thesis, the results obtained are fairly accurate and precise. In order to improve further accuracy of the models, more vulnerabilities need to be added and modeled.

Although taxi trips in hour and average speed in hour worked as intermediary for traffic, there is lot of scope of modeling on the effect of location. These features could be used to calculate duration of trips in different areas and further to model local effects of traffic.

All these steps could be considered either by analyzing larger data sets so as to intrude relationships and effects of traffic and location at different location and at different times, in addition to this aggregation with different data sets, traffic, speed threshold, accidents etc.

Results obtained by implementing machine learning algorithms can also be improved

if we switch to neural networks. In this field, there are researchers working towards building Neural Networks and by providing complete and relevant features, deep neural network could show significant improvement on real-time data.

Bibliography

[1] NYC.gov, "Nyc taxi limousine commission," *NYC Taxi Limousine Commission official government site*, December 2019. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[2] A. F. A. J. Christophoros Antoniades, Delara Fadavi, "Fare and duration prediction: A study of new york city rides," *Stanford education*, no. 3, pp. 1236–1239, December 2016. [Online]. Available: http://cs229.stanford.edu/proj2016/report/AntoniadesFadaviFobaAmonJuniorNewYorkCityCabPricing-report.pdf

[3] G. kaya Uyanik and N. Guler, "A study on multiple linear regression analysis," *Elsevier*, no. 1, pp. 234–240, January 2013. [Online]. Available: https://core.ac.uk/download/pdf/82526651.pdf

[4] A. Schneider, G. Hommel, and M. Blettner, "Linear regression analysis: part 14 of a series on evaluation of scientific publications." *Deutsches Arzteblatt international*, vol. 107 44, pp. 776–82, 2010.

[5] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[6] G. Biau, "Analysis of a random forests model," *J. Mach. Learn. Res.*, vol. 13, pp. 1063–1095, Apr. 2012. [Online]. Available: http://jmlr.org/papers/volume13/biau12a/biau12a.pdf

[7] Wikipedia contributors, "Haversine formula wikipedia, the free encyclopedia," https://en.wikipedia.org/w/index.php?title=Haversine_formula&oldid=892532744, 2019, [Online; accessed 12-May-2019].

[8] J. Miao and L. Niu, "A survey on feature selection," *Procedia Computer Science*, vol. 91, pp. 919–926, 12 2016.

Appendix A

Code Snippets

Complete code is present in the below link mentioned.
https://github.com/PoojaTyagi-AI/Master-project

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from haversine import haversine
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
import xgboost as xgb
from sklearn import metrics
from sklearn.model_selection import train_test_split
import statsmodels.formula.api as sm
from bokeh.palettes import Spectral4
from bokeh.plotting import figure, output_notebook, show


train_data = pd.read_csv("/Users/poojatyagi/Desktop/AI PROJECT/DATA SOURCES/train.csv")
test_data = pd.read_csv("/Users/poojatyagi/Desktop/AI PROJECT/DATA SOURCES/test.csv")
train_data.head()
train_data.shape

#numer of unique Id'S in dataset
print("There are %d unique id's in Training dataset, which is equal to the number of records"%(train_data.id.nunique()))

#checking for NaN values in a dataset
train_data.isnull().sum()

#checking teh datatype of each input feature
train_data.dtypes

#converting the datatype of "pickup_datetime" and "dropoff_datetime" into datetime format
train_data['pickup_datetime'] = pd.to_datetime(train_data['pickup_datetime'])
train_data['dropoff_datetime'] = pd.to_datetime(train_data['dropoff_datetime'])

#Adding more features i.e. "weekday", "month", "weekday_num" and "pickup_hour"
train_data['weekday'] = train_data.pickup_datetime.dt.weekday_name
train_data['month'] = train_data.pickup_datetime.dt.month
train_data['weekday_num'] = train_data.pickup_datetime.dt.weekday
train_data['pickup_hour'] = train_data.pickup_datetime.dt.hour

train_data.head()

#Defining dist_calc() function for Calculating distance between  pickup and dropoff coordinates using Haversine formula
def dist_calc(df):
    pickup = (df['pickup_latitude'], df['pickup_longitude'])
    drop = (df['dropoff_latitude'], df['dropoff_longitude'])
    return haversine(pickup, drop)
```

FIGURE A.1: Importing libraries and training model

```
#Calculate distance and assign new feature "haversin_-distance" to the dataframe.
train_data['haversin_distance'] = train_data.apply(lambda x: dist_calc(x), axis = 1)


#Calculate Speed in km/h for further insights
train_data['speed'] = (train_data.haversin_distance/(train_data.trip_duration/3600))

#Check the type of each input feature
train_data.dtypes.reset_index()


'''Treating categorical features Dummify all the categorical features like "store_and_fwd_flag, vendor_id,
   month, weekday_num, pickup_hour, passenger_count" except the label i.e. "trip_duration" '''

dummy = pd.get_dummies(train_data.store_and_fwd_flag, prefix='flag')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)

dummy = pd.get_dummies(train_data.vendor_id, prefix='vendor_id')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)

dummy = pd.get_dummies(train_data.month, prefix='month')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)

dummy = pd.get_dummies(train_data.weekday_num, prefix='weekday_num')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)

dummy = pd.get_dummies(train_data.pickup_hour, prefix='pickup_hour')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)

dummy = pd.get_dummies(train_data.passenger_count, prefix='passenger_count')
dummy.drop(dummy.columns[0], axis=1, inplace=True) #avoid dummy trap
train_data = pd.concat([train_data,dummy], axis = 1)
```

FIGURE A.2: Part 2

```
############################ UNIVARIATE ANALYSIS ##########################################################################################

'''[1.] PASSENGER_COUNT (According to NY Limousine commsion 5 adults and 1 child is the max limit of taxi passenger)'''
train_data.passenger_count.value_counts()

#Detecting outliers (here passenge count = 7,8 ,9)
plt.figure(figsize = (20,5))
sns.boxplot(train_data.passenger_count)
plt.show()
'''observations --1. Instances with 0 passengers exists
                  2. Trips with 7,8 and 9 passengers count is less
                  3. Maximum trips has passenger count 1 or 2 '''


#Treating outliers "passenger_count" feature
train_data.passenger_count.describe()
#coverting all the taxi trips with passenger count = 0 to passenger count = 1. since mean value is approx 1
train_data['passenger_count'] = train_data.passenger_count.map(lambda x: 1 if x == 0 else x)
#Considering trips with maximum passenger 6
train_data = train_data[train_data.passenger_count <= 6]

sns.countplot(train_data.passenger_count)
plt.show() #shows max trips with passenger 1 and max passenger count is 6


'''[2.] VENDOR '''
sns.countplot(train_data.vendor_id)
plt.show() #shows vendor 2 is more famous than vendor 1


'''[ 3.] Distance'''
print(train_data.haversin_distance.describe())

#Boxplot presentation of haversin_distance
plt.figure(figsize = (20,5))
sns.boxplot(train_data.haversin_distance)
plt.show()
'''OBSERVATION--1. There are trips with 0 km distance
                2. Maximum trips are btw 0 -10 km
                3. There are few trips above 100 km
                4. Mean is 3.44
                5. Standard deviation is 4.30 so max trips have distance btw 0-10 km'''

print("Number of trips with zero distance are: ",train_data.haversin_distance[train_data.haversin_distance == 0 ].count())

#Treating outliers removing trips beyond 100 km
train_data = train_data[train_data.haversin_distance <= 100]
train_data.haversin_distance.groupby(pd.cut(train_data.haversin_distance, np.arange(0,100,10))).count().plot(kind='barh')
plt.show()
```

FIGURE A.3: Part 3

```
'''[4.] Trip duration'''
train_data.trip_duration.describe()

#Detecting outliers
plt.figure(figsize = (20,5))
sns.boxplot(train_data.trip_duration)
plt.show()

#OBSERVATIONS -----  1. There are few trips whose duration is beyond 86400 seconds i.e. 24 hours which is clear outliers
train_data = train_data[train_data.trip_duration <= 86400]

#visualizing the trip duration
train_data.trip_duration.groupby(pd.cut(train_data.trip_duration, np.arange(1,7200,600))).count().plot(kind='barh')
plt.xlabel('Trip Counts')
plt.ylabel('Trip Duration (seconds)')
plt.show()

'''OBSERVATIONS -- 1. Maximum trips have duration between 1-601 sec i.e. (0 -30 mins )'''



'''[5.] SPEED'''
'''Maximum speed limit in NYC is as follows:
   25 mph in urban area i.e. 40 km/h
   65 mph on controlled state highways i.e. approx 104 km/h'''

train_data.speed.describe()
plt.figure(figsize = (20,5))
sns.boxplot(train_data.speed)
plt.show()

'''Observation -- 1. Some trips are going beyond a aspeed of 220km which is not acceptable
                    So these are outliers and we need to remove them '''

train_data = train_data[train_data.speed <= 104]
plt.figure(figsize = (20,5))
sns.boxplot(train_data.speed)
plt.show()
```

FIGURE A.4: Part 4

```
print('Experiment 5 \t Variance score(R squared): %.2f' % metrics.r2_score(y_test_fs,y_pred_rf1))
print("\n")
print("\n \n Evaluating XGBoost Regression :\n")
print("EXPERIMENT 6:")
print("Experiment 6 \t RMSE: ", rmse(y_test,predictions_raw))
print("Experiment 6 \t MAPE(%): ",mean_absolute_percentage_error(y_test,predictions_raw))
print('Experiment 6 \t Variance score(R squared):',metrics.r2_score(y_test, predictions_raw))
print("\n")
print("EXPERIMENT 7:")
print("Experiment 7 \t RMSE: ", rmse(y_test_fs,predictions))
print("Experiment 7 \t MAPE(%): ",mean_absolute_percentage_error(y_test_fs,predictions))
print('Experiment 7 \t Variance score(R squared): %.2f' % metrics.r2_score(y_test_fs,predictions))

#########################################################################################################################
########### VISUALIZATION OF ACCURACY ###############

#DISPLAYING THE IMPROVED TABLE OF ACCURACY OF ALL MODELS AFTER HYPER PARAMETER OPTIMIZATION
Summary_table_1 = pd.DataFrame([rmse(y_test,y_pred),rmse(y_test_fs,y_pred_fs),
                                rmse(y_test,y_pred),rmse(y_test_fs,y_pred_fs),
                                rmse(y_test_fs,y_pred_rf1),rmse(y_test,predictions_raw),
                                rmse(y_test_fs,predictions) ],
                                index=['Experiment 1', 'Experiment 2', 'Experiment 3','Experiment 4','Experiment 5',
                                'Experiment 6','Experiment 7'], columns=['RMSE'])


print("Summary Table for RMSE")
Summary_table_1

#PLOTTING GRAPGH OF ACCURACY
plt.figure(figsize=(10,5))
sns.barplot(Summary_table_1.index,Summary_table_1['RMSE'])
plt.show()


Summary_table_2 = pd.DataFrame([mean_absolute_percentage_error(y_test,y_pred),mean_absolute_percentage_error(y_test_fs,y_pred_fs),
                                mean_absolute_percentage_error(y_test,y_pred),mean_absolute_percentage_error(y_test_fs,y_pred_fs),
                                mean_absolute_percentage_error(y_test_fs,y_pred_rf1),mean_absolute_percentage_error(y_test,predict
                                mean_absolute_percentage_error(y_test_fs,predictions) ],
                                index=['Experiment 1', 'Experiment 2', 'Experiment 3','Experiment 4','Experiment 5',
                                'Experiment 6','Experiment 7'], columns=['MAPE (Mean Absolute Percentage Error %)'])
print("Summary Table for MAPE")
Summary_table_2

#PLOTTING GRAPGH OF ACCURACY
plt.figure(figsize=(10,5))
sns.barplot(Summary_table_1.index,Summary_table_2['MAPE (Mean Absolute Percentage Error %)'])
plt.show()

Summary_table_3 = pd.DataFrame([metrics.r2_score(y_test,y_pred),metrics.r2_score(y_test_fs,y_pred_fs),
                                metrics.r2_score(y_test,y_pred),metrics.r2_score(y_test_fs,y_pred_fs),
                                metrics.r2_score(y_test_fs,y_pred_rf1),metrics.r2_score(y_test,predictions_raw),
                                metrics.r2_score(y_test_fs,predictions) ],
                                index=['Experiment 1', 'Experiment 2', 'Experiment 3','Experiment 4','Experiment 5',
                                'Experiment 6','Experiment 7'], columns=['Variance (R squared)'])

print("Summary Table for R2 Score")
Summary_table_3
```

FIGURE A.5: Part 5