MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

TRAVEL TIME PREDICTION

Minor Project Report

Semester

Submitted by:

AKSHAY MISHRA	171112063
VISHANT GARG	171112091
SIDDHARTH GAUTAM	171112090
ARPIT PANDEY	171112049

Under the Guidance of
PROF. B.N.ROY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Session: 2019-20

MAULANA AZAD
NATIONAL INSTITUTE OF TECHNOLOGY
BHOPAL INDIA, 462003

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

<u>CERTIFICATE</u>

This is to certify that the project report carried out on "Travel Time Prediction " by the 3rd year students:

AKSHAY MISHRA	171112063
VISHANT GARG	171112091
SIDDHARTH GAUTAM	171112090
ARPIT PANDEY	171112049

Have successfully completed their project in partial fulfilment of their Degree in Bachelor of Technology in Computer Science and Engineering.

PROF. B.N. ROY
(Minor Project Mentor)

DECLARATION

We, hereby declare that the following report which is being presented in the Minor Project Documentation Entitled as “TRAVEL TIME PREDICTION” is an authentic documentation of our own original work and to best of our knowledge. The following project and its report, in part or whole, has not been presented or submitted by us for any purpose in any other institute or organization. Any contribution made to the research by others, with whom we have worked at Maulana Azad National Institute of Technology, Bhopal or elsewhere, is explicitly acknowledged in the report.

ARPIT PANDEY	171112049
AKSHAY MISHRA	171112063
SIDDHARTH GAUTAM	171112090
VISHANT GARG	171112091

ACKNOWLEDGEMENT

With due respect, we express our deep sense of gratitude to our respected guide and coordinator PROF. B.N. ROY, for his valuable help and guidance. We are thankful for the encouragement that he has given us in completing this project successfully.

It is imperative for us to mention the fact that the report of minor project could not have been accomplished without the periodic suggestions and advice of our project guide PROF. B.N. ROY and project coordinators Dr. Dhirendra Pratap Singh and Dr. Jaytrilok Choudhary.

We are also grateful to our respected director Dr. N. S. Raghuwanshi for permitting us to utilize all the necessary facilities of the college.

We are also thankful to all the other faculty, staff members and laboratory attendants of our department for their kind cooperation and help. Last but certainly not the least; we would like to express our deep appreciation towards our family members and batch mates for providing the much needed support and encouragement

ABSTRACT

Travel time prediction is crucial in developing mobility on demand systems and traveller information systems. Precise estimation of travel time supports the decision-making process for riders and drivers who use such systems

This project investigate the application of a machine learning algorithm to predict the time that will be spent by a vehicle between any two points in an approximated area. The prediction is based on a learning process based on historical data about the movements performed by the vehicles taking into account a set of semantics variable to estimate time accurately.

The project is about estimating ride duration without real time data, by analysing data collected from taxis.This project primarily focuses on the possible important factors that are used as attributes for the trip duration prediction in the New York City For prediction purposes factors such as pick up latitude, pick up longitude, drop off latitude, drop off longitude , pick up date, pick up time etc. are considered.

The project focuses on comparing the forecasting effectiveness of three machine learning models, namely Decision Tree Regression ,Random Forests Regression, K-Nearest Neighbour Regression - in addition to Multiple Linear Regression-using Jan 2015 NYC Yellow Cab trip record data

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. Introduction

Predicted travel time is very useful for drivers .Many users precede a trip by obtaining information on the travel time for their route using their computers, smart phones, or navigation systems.

Predicting a trip duration isn't something that has not been though upon. With the use of Google maps API one can find the estimated time it would take to move between two points in the city. However, a detailed analysis of the factors affecting a trip between two points in a city can be very useful for accurate and robust prediction.

Trip duration is not as simple as it seems. It is data dependent and is governed by a lot many factors apart from distance and speed. This project primarily focuses on the possible important factors that are used as attributes for the trip duration prediction in the New York City. This data can be used by taxi vendors for better services to the users. The project work not only uses a prediction model but also gives an in-depth analysis of the factors associated with the New York City taxi trips. A city like New York is expected to have various factors and variations with respect to the trip durations. The dataset used for training and testing purposes in multi-dimensional and requires a lot of pre-processing. For prediction purposes factors such as pick up latitude, pick up longitude, drop off latitude, drop off longitude etc. is considered. These geographical locations clubbed with other important factors such as pick up date, pick up time are used for the overall trip duration prediction

This project focuses on comparing the forecasting effectiveness of three machine learning models, namely Decision Tree Regression ,Random Forests Regression, K-Nearest Neighbour Regression - in addition to Multiple Linear Regression-using Jan 2015 NYC Yellow Cab trip record data.

PROBLEM DEFINITION

Travel time is the time taken by a vehicle, moving from one location to another including the effect of temporal conditions.

In simple words, one can think of this problem as to estimate the travel distance and time between an origin (o) and a destination (d) at a particular time (t) time-of-day.

2.Literature review

Travel time prediction has emerged as an active and intense research area nowadays. In the literature, there are a large number of researches that can deal with accurate prediction of travel time on road networks. In the following section, a wide-ranging literature review on the topic of travel time prediction is presented. Travel time prediction main feature general people consider are distance. But actually their are may factor which are responsible for the travel time. We have explore the some of features in this project like distance, holiday, week day, pickup tine and drop-off time etc. As Our dataset may contain some non usefull attributes. To identify these columns we have used the features selection so that our model results into good prediction. As good dataset always required for the good performance model. As many attributes values are very high and some are very low which shifts our measure of central value. So we have do the elimination for these valuea this known as outlier detection and removal. Before training any model we have normalize the values of each attributes so that each feature can have equal probability of being select. Their high and low does not affect on the features selection.

For every model we split our dataset into to dataset training and testing dataset . Training is used to train the model and testing is used to check the performance of the model. So that we check how model performance in real world. Model that we have used are simple linear regression,multiple regression,decsion tree regression, Random forest Regression and knn Regression.

And we have compare the performance of each model in which knn model results into highest score of 96% accuracy.

3. Gaps Identified

Now days travel time prediction is becomes very useful for business as well as for the people to achieve the work completion on time .our project mainly focused to predict the travel time more accurately and make the prediction reliable. As travel helpa us schedule our jobs and events so the optimal uses can be achieved. As accuracy of prediction depends on the how accurate our dataset. But in real life data contains noise and outliers. Missing values in the dataset also createa problems for us. To achieve a good dataset we do the preprocessing od the dataset. In preprocessing we remove some attributes and add some attributes. A good preprocessing of the dataset always results into good dataset which is ready to use data. As with time data is generated very faster and we need some advanced computational solution so for that machine learning comes in their are many framework for machine learning. The framework we have used is jupyter notebook and python as programming language. We have properly integrated the uberdataset with nyc taxi dataset. Now we have tested many different model so we can select the model with highest accuracy. As accurate results always makes us to take good decision.

4. Proposed work and methodology:

4.1 Background Information

4.1.1 Linear Regression

This analysis method is used for predicting the values of a variable based on the value of a different variable. The variable to be predicted, is called a dependent variable, and the variable which will be used to predict the other variable's value is known as the independent variable. This type of analysis calculates the coefficients of the linear equation, containing one or more independent variables which predict the value of the dependent variable, the best. Linear regression fits a straight line or surface that minimizes the differences between predicted and actual output values. Various simple linear regression calculators can be found, which use a “least squares” method to find the best-fit line for a collection of paired data. We then estimate the value of X (dependent variable) depending upon Y (independent variable).

Linear Regression sets-up a relationship between dependent variable (Y) and one or more independent variables (X) with the help of a best fit straight line (which is known as regression line).

a. When there is only one independent variable in the linear regression model, the model is generally termed as simple linear regression model.
b. When there are more than one independent variables in the model, then the linear model is termed as the multiple linear regression model.

c. Consider a simple linear regression model

$$y = \beta 0 + \beta 1 X + \varepsilon \ .$$

where y is termed as the dependent or study variable and X is termed as independent or explanatory variable.

The terms $\beta 0$, $\beta 1$ and ε are the parameters of the model.

The parameter $\beta 0$ is termed as intercept term and the parameter $\beta 1$ is termed as slope parameter. These parameters are usually called as regression coefficients.

The unobservable error component ε accounts for the failure of data to lie on the straight line and represents the difference between the true and observed realization of y. This is termed as disturbance or error term.

The independent variable X is viewed as controlled by the experimenter, so it is considered as non-stochastic

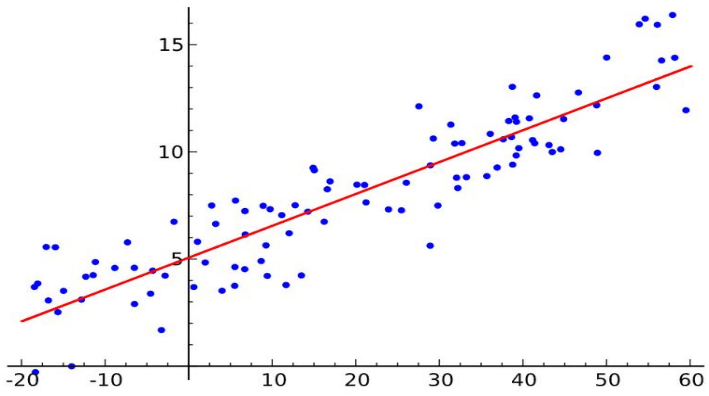
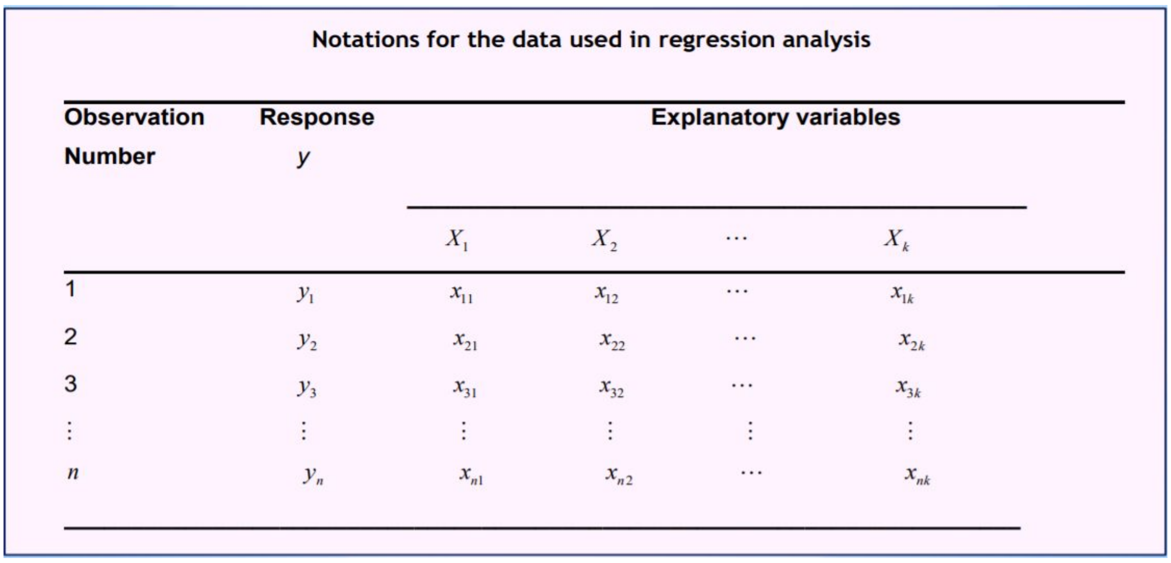


Fig 4.1:Fit a Line through given observation



Notations for the data used in regression analysis

Observation Number	Response y	Explanatory variables			
		X_1	X_2	$\cdots$	X_k
1	y_1	x_{11}	x_{12}	$\cdots$	x_{1k}
2	y_2	x_{21}	x_{22}	$\cdots$	x_{2k}
3	y_3	x_{31}	x_{32}	$\cdots$	x_{3k}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	y_n	x_{n1}	x_{n2}	$\cdots$	x_{nk}

Fig 4.2 : Terminology in Regression Analysis

In order to know the value of the parameters, n pairs of observations are observed/collected and are used to determine these unknown parameters.

Various methods of estimation can be used to determine the estimates of the parameters.

There can be three ways of calculating error function:

1. **Sum of residuals (∑(Yactual – Ypredict))** – It might result in cancelling out of positive and negative errors.
2. **Sum of the absolute value of residuals (∑ | Yactual - Ypredict |)** – Absolute value would prevent cancellation of errors
3. **Sum of square of residuals (∑ (Yactual - Ypredict $)^2$)** –We penalize higher error value much more as compared to a smaller one, so that there is a significant difference between making big errors and small errors, which makes it easy to differentiate and select the best fit line

Among them, the **Sum of Squares Residual (least squares estimation)** is the popular methods of estimation.

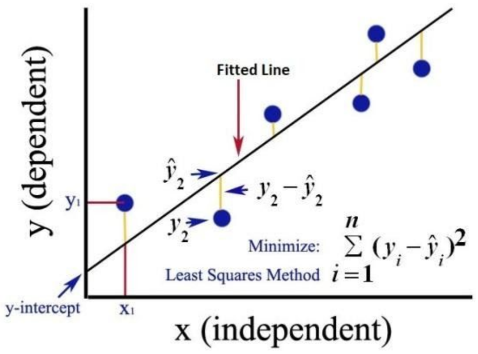


Fig 4.3 : Least Square Regression

4.1.2 Decision tree

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

A Decision tree is a tree structured classifier, where each internal node(decision node) denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.

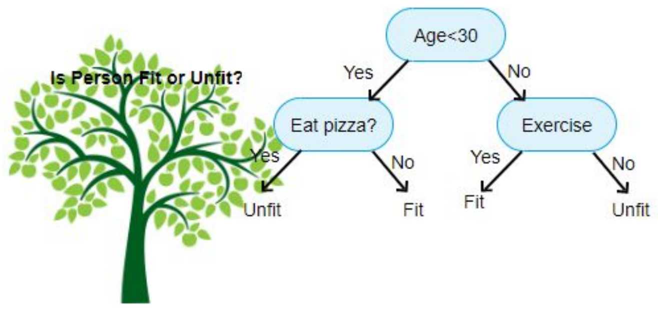


Fig 4.4 : Decision Tree

Construction of Decision Tree

A tree can be "learned" by splitting the source set into subsets based on an attribute value test(select best attribute). This process is repeated on each derived subset in a recursive manner called "recursive partitioning". The recursion is completed when the subset at a node all has the same value of the target variable, or when splitting no longer adds value to the predictions.

Choosing Best attribute for split-

There are two popular metrics used for selecting best feature for split

i. INFORMATION GAIN

The information gain is based on the decrease in entropy after a dataset is split on an attribute(feature). Constructing a decision tree is all about finding attribute that returns the highest information gain (i.e., the most homogeneous branches).

Entropy is basically the randomness in the decision tree.

Let S is a sample of training example

P+,P- is proportion of positive,negative examples of S respectively

$$\text{Entropy}(S) = -p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$I_H = -\sum_{j=1}^{c} p_j log_2(p_j)$$

p_j: proportion of the samples that belongs to class c for a particular node.

*This is the the definition of entropy for all non-empty classes ($p \neq 0$). The entropy is 0 if all samples at a node belong to the same class.

ii. Gini Index

Gini Index is another measure on the basis of which we decide on which feature to split on. This is used internally in the inbuild Decision tree in Sklearn library.

Gini Index favors larger partitions. It uses squared proportion of classes. For a perfectly classified i.e pure node ,Gini Index would be zero. We want a variable split that has a low Gini Index.

$$I_G = 1 - \sum_{j=1}^{c} p_j^2$$

p_j: proportion of the samples that belongs to class c for a particular node

Regression criteria

If the target is a continuous value, then for node m, representing a
region Rm with Nm observations, common criteria to minimise as for determining locations for future splits are Mean Squared Error, which minimizes the L2 error using mean values at terminal nodes, and Mean Absolute Error, which minimizes the L1 error using median values at terminal nodes.

Mean Squared Error:

$$\bar{y}_m = \frac{1}{N_m} \sum_{i \in N_m} y_i$$
$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - \bar{y}_m)^2$$

Mean Absolute Error:

$$median(y)_m = \underset{i \in N_m}{\mathrm{median}}(y_i)$$
$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} |y_i - median(y)_m|$$

where X_m is the training data in node m

4.1.3. Random Forest

In decision trees, overfitting is a major problem. As decision tree keep expanding till it runs out of features or till it finds a pure node, therefore there are very high chances of overfitting on the training data. Decision tree will split on the nodes that are not actually important if it is favourable. This causes decision trees to perform perfectly on training data but sometimes fail on testing data. Pruning definately helps to some extent but even it does not consider how important is a feature.

Random forest is a nice way to counter this situation.Random forest is another way to reduce overfitting in decision trees and it can also be used to find importance of features we are using.

Random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction. Each decision tree built will have a randomly selected set of features and randomly selected set of data points. Number of features in each decision tree in the forest will be less than the total number of features we have in our dataset. So if we have a feature 'A', it may appear in some of the decision trees of the forest and not in the other. Duplication is generally allowed in selecting the data-points from our data-set for the decision trees in the forest

Steps taken to implement Random forest:

i. Suppose there are **N observations** and **M features** in training data set. First, a sample from training data set is taken randomly with replacement.

ii. A subset of M features are selected randomly and whichever feature gives the best split is used to split the node iteratively to form the tree (as done in decision tree building).

iii. The tree is grown to the largest (without pruning).Above steps are repeated and prediction is given based on the aggregation of predictions from multiple trees which are built the same way.

Bagging training data & selecting features at random

Bagging (Bootstrap Aggregation) is used when our goal is to reduce the variance of a decision tree. Bagging is a general purpose procedure for reducing the variance of a predictive model. When applied to trees the basic idea is to grow multiple trees which are then combined to give a single prediction. Combining multiple trees help in improving precision and accuracy at the expense of some loss of interpretation. In bagging, we take multiple smaller data-sets in which we also allow repetition of data points and randomly select some features. Bagging is generally done in reference of data-points.

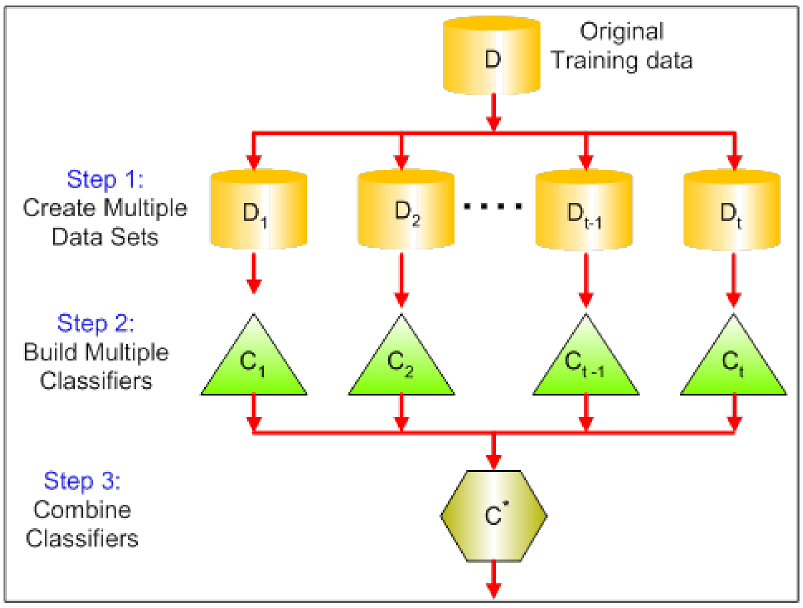

D
Original Training data
Step 1: Create Multiple Data Sets
D1
D2
Dt-1
Dt
Step 2: Build Multiple Classifiers
C1
C2
Ct-1
Ct
Step 3: Combine Classifiers
C*

Fig 4.5 Bagging

These smaller data-sets are obtained by choosing the data-points and the features in the following manner:

Features are selected at random **without** repetition

Data-points are selected at random **with** repetition(which is actually bagging)

4.1.4. K-Nearest Neighbours

KNN stands for K-Nearest Neighbours. KNN is **non-parametric** and **lazy** learning algorithm.

Non-parametric basically means that the algorithm does not make any assumptions on the given data distribution. NON-parametric covers technique that do not rely on data belonging to particular distribution and do not assume the structure of model to be fixed. So, KNN is used as classification algorithm in cases where we do not have much information about the distribution of the data.

KNN is referred to as **Lazy** algorithm since is does not uses training points to do any generalization, which means that there is no separate training phase. KNN keeps all the training data and uses most of the training data during the testing phase. Thus, KNN does not learn any model, it make predictions on the fly, computing similarity between testing point and each training data point.

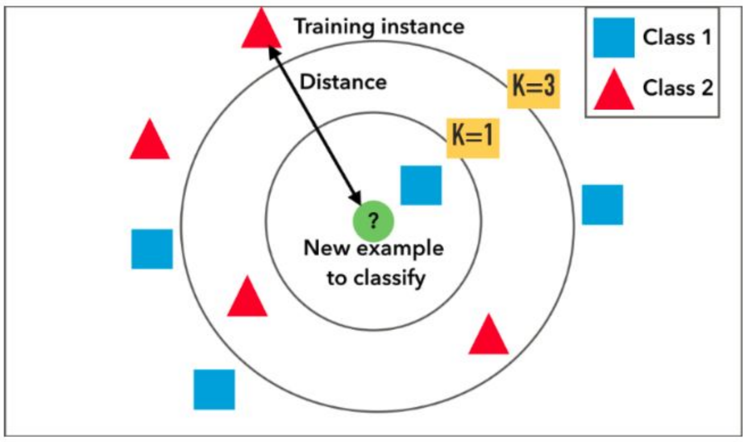


Fig 4.6 K Nearest Neighbour

KNN algorithm is based on **feature similarity**. We can classify the testing data point on the basis of resemblance of its features with that of the training data set.

KNN can be used for **Classification** as well as **Regression**.

In regression, output is the property value of object, this value is average or median of the value of its K nearest neighbors**.**

Distance Metric for KNN

There are various distance metrices that can be choosen are Manhattan Distance, Euclidian Distance, Minkowski distance

$$\textbf{MANHATTAN DISTANCE} = \left| \sum_{i=1}^{n} X_1^i - X_2^i \right|$$

$$\textbf{EUCLIDIAN DISTANCE} = \sqrt{\sum_{i=1}^{n} (X_1^i - X_2^i)^2}$$

$$\textbf{MINKOWSKI DISTANCE} = \left| \sum_{i=1}^{n} X_1^i - X_2^i \right|^p$$

Where X_1 and X_2 are two different data points and 'i' traverse over all the features in the given dataset.

Cross Validation

The best value of K is the one for which we get lowest error on testing data. To obtain the optimal value of K, we use a method known as CROSS VALIDATION.

Cross Validation basically means taking out the subset from the training data and not using this subset in the training process. This subset of training data is called the 'validation set'. There are various techniques available for cross validation, we will be using the most general one, known as **K-fold cross validation**.

In K fold cross validation, the training data is randomly split into K different samples(or folds). One of the sample is taken to be the validation set and the model is fitted on the remaining K-1 samples. The accuracy of the model is then computed. The same process is repeated K times, each time taking a different sample of points to be in the validation set. This results in K value for test error and these values are averaged out to obtain the overall result.

KNN- PROS :

Easy to understand and code.Works well for multi-class classification.
Insensitive to outliers for optimal choice for value of K, but accuracy can be affected from noise and irrelevant features.

KNN- CONS :

Computationally expensive, because the algorithm stores all the training data.Testing time is huge.
Sensitive to irrelevant features and scaling.

4.2 Methodology:

Trip duration is not as simple as it seems. It is data dependent and is governed by a lot many factors apart from distance and speed. This project primarily focuses on the possible important factors that are used as attributes for the trip duration prediction in the New York City. A city like New York is expected to have various factors and variations with respect to the trip durations. The dataset used for training and testing purposes in multi-dimensional and requires a lot of pre-processing. For prediction purposes factors such as pick up latitude, pick up longitude, drop off latitude, drop off longitude etc. is considered. These geographical locations clubbed with other important factors such as pick up date, pick up time are used for the overall trip duration prediction

This project focuses on comparing the forecasting effectiveness of three machine learning models, namely Decision Tree Regression ,Random Forests Regression, K-Nearest Neighbour Regression - in addition to Multiple Linear Regression-using Jan 2015 NYC Yellow Cab trip record data.

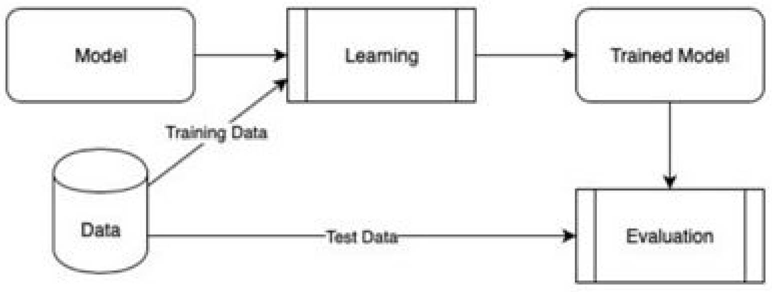


Fig 4.7 Architecture

4.3 DATABASE DESIGN

3.2.1. Database Source

The data used in this project are all subsets of New York City Taxi and Limousine Commission's trip data, which contains observations on around 1 billion taxi rides in New York City between 2009 and 2016. The total data is split between yellow taxis, which operate mostly in Manhattan, and green taxis, which operate mostly in the outer areas of the city. For the main analyses of this study, the data for yellow taxi rides during the month of January 2015 were used. Since each month consists of about 12 million observations, and there were computational limitations, subsets of the monthly data were used for model building, and other subsets were used for validation.

We have also used NYC Uber Pickups with Weather and Holidaysthat contains data about Uber Pickups in New York City, from 01/01/2015 to 30/06/2015 .
We have used this to get information about holiday on a given date

3.2.2. Data Structure

The given NYC_Taxi database is in CSV format.The original dataset contains features as pickup and drop-off locations, as longitude and latitude coordinates, time and date of pickup and drop-off, ride fare, tip amount, payment type, trip distance and passenger count (as well as other, for this study, less relevant variables).

Attribute	Meaning
'tpep_pickup_datetime'	The date and time when the meter was engaged.
'tpep_dropoff_datetime'	The date and time when the meter was disengaged.
'pickup_longitude	longitude where the meter was engaged
'pickup_latitude'	latitude where the meter was engaged
'dropoff_longitude'	longitude where the meter was disengaged
'dropoff_latitude'	latitude where the meter was disengaged
'trip_distance'	Trip distance in Km
'pickup_hrs'	Pickup time to nearest Hr
'dropoff_hrs'	Drop-off time to nearest Hr
'day_week'	Integer Representing Day of week
'duration'	Duration of the trip in seconds

Table 4.1 NYC_taxi Dataset

Uber_NYC dataset is in csv format and contains following attributes

Attribute	Meaning
pickup_dt	Time period of the observations
hday	Being a holiday (Y) or not (N).
borough	NYC's borough.
pickups	Number of pickups for the period
spd	Wind speed in miles/hour
vsb	
temp	temperature in Fahrenheit.
dewp	Dew point in Fahrenheit
slp	Sea level pressure
pcp01	1-hour liquid precipitation
pcp06	6-hour liquid precipitation
pcp24	24-hour liquid precipitation
sd	Snow depth in inches

Table 4.2 Uber_NYC Dataset

4.4 Task Design

The project work can be broadly divided into 2 Tasks

a. **Processing Input Dataset.**

i. Add/Remove Attributes

We have created a seperate new column ‘date’ whose value is extracted from ‘tpep_pickup_date_time’

We have used Uber_NYC dataset to map each date with holiday i.e. Using Uber_NYC dataset we have come up with a dictionary that has key as ‘date’ and value as ‘Y’ / ’N’ indicating whether or not there was holiday on given date in NYC

ii. Outlier Removal

We have used Inter-Quartile Range for Outlier Detection and Removal

This is done using these steps:

1. Calculate the inter-quartile range for the data.
2. Multiply the inter-quartile range (IQR) by 1.5 (a constant used to discern outliers).
3. Add 1.5 x (IQR) to the third quartile. Any number greater than this is a suspected outlier.
4. Subtract 1.5 x (IQR) from the first quartile. Any number less than this is a suspected outlier.

iii. Feature Selection and Feature Scaling

MinMax Scaler is used for feature Scaling to bring attribute value in range [0,1]

SelectKBestFeature from SKlearn is used for feature selection using Chi ^{2} as score metric

b. Building Machine Learning Models For Prediction

i. Simple Linear Regression Model

This Model uses trip_distance as the only input feature and predicts trip_duration

ii. Multiple Linear Regression Model

This model uses the all of the 'Best-Features' selected using feature selection.

iii. Decision Tree Regression Model

This model uses the all of the 'Best-Features' selected using feature selection.

Various models have been created by experimenting with parameters of model like maxdepth

iv. Random Forest Regression Model

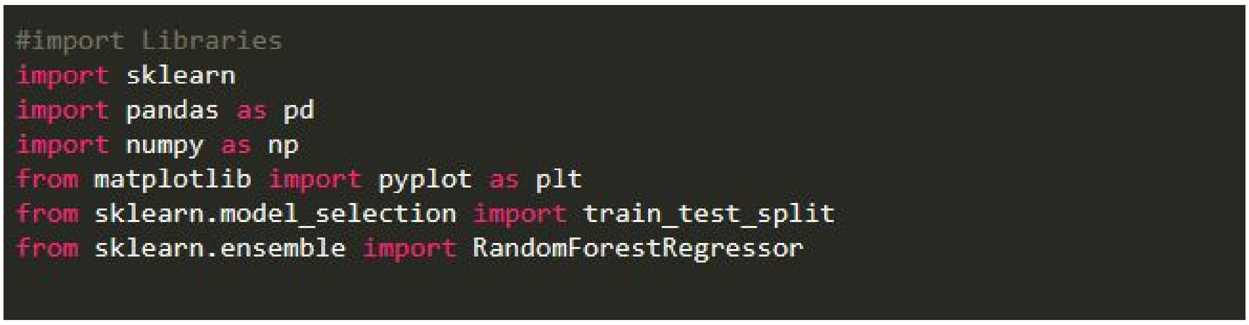
This model uses the all of the 'Best-Features' selected using feature selection.

Max-depth= 6 is used for optimal accuracy,Rest all parameters take default values.

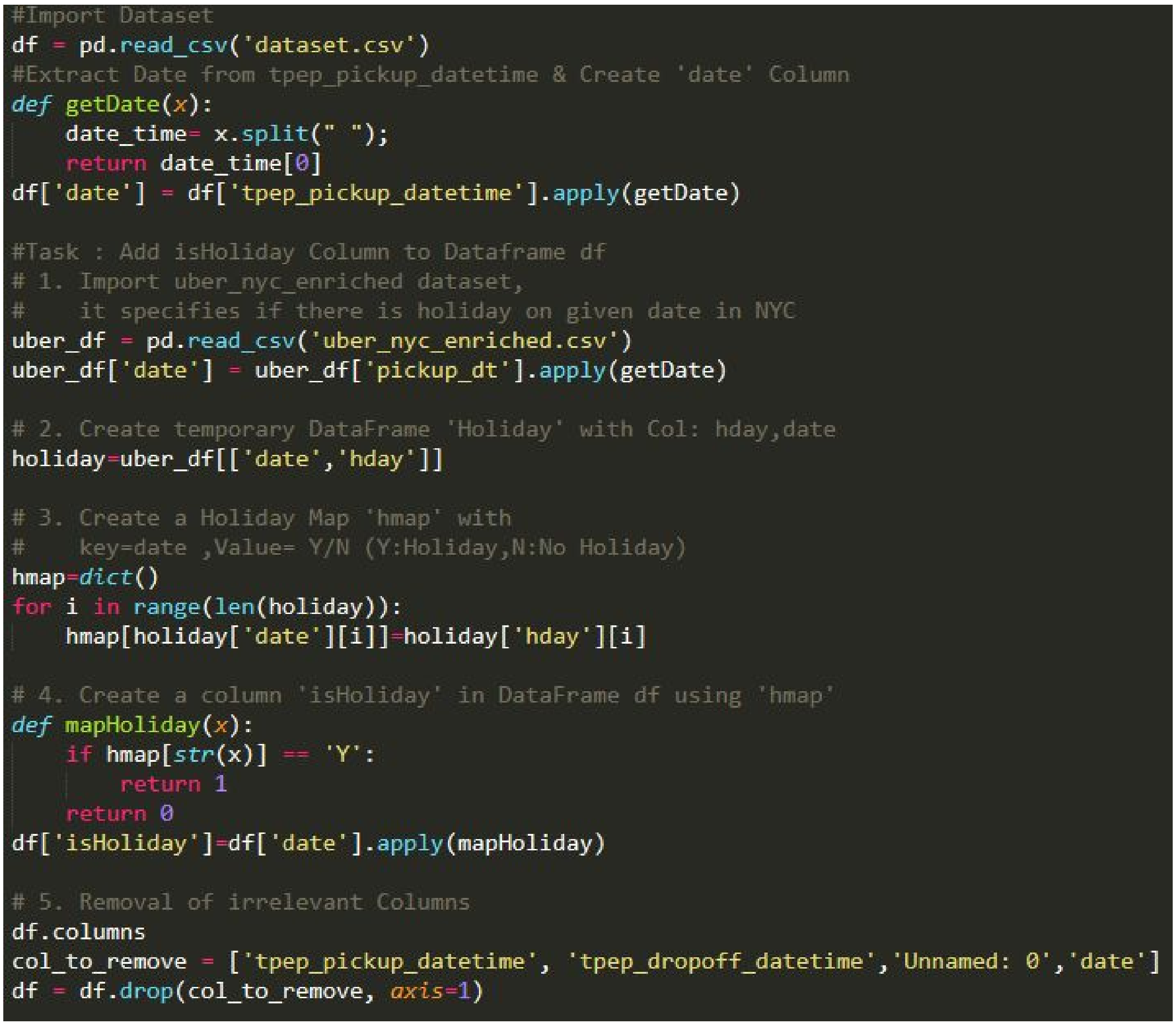
v. KNN Regression Model

This model uses the all of the 'Best-Features' selected using feature selection.

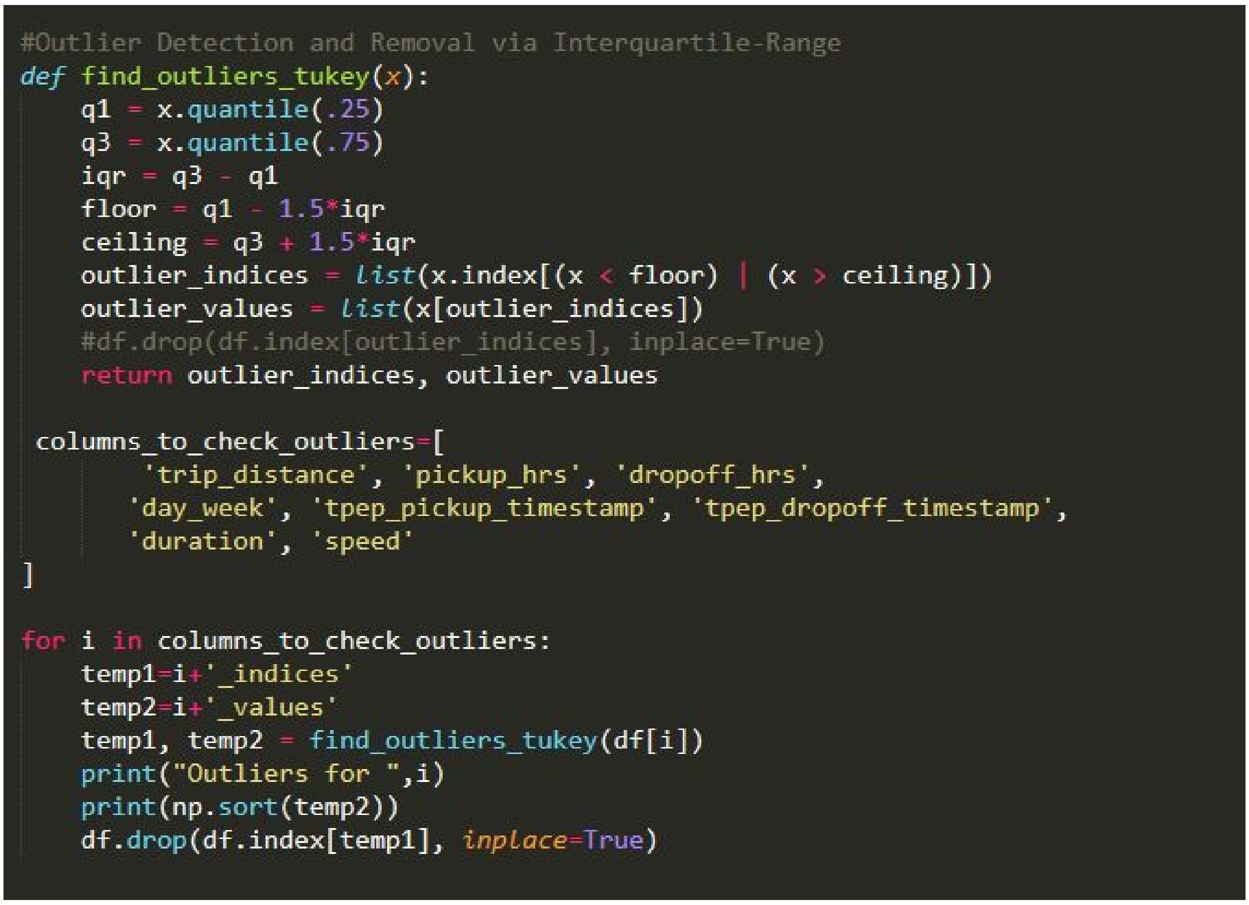
4.5 Implementation & Coding



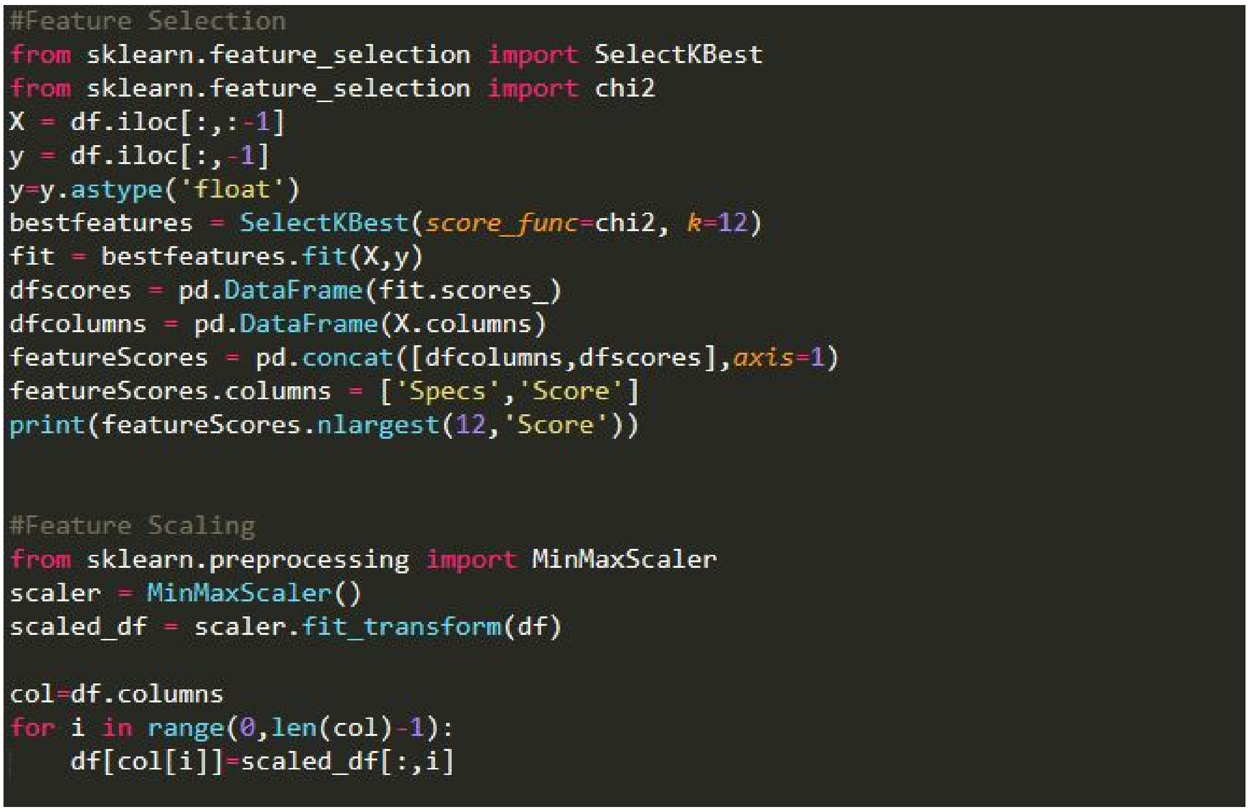
```python
#import Libraries
import sklearn
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```



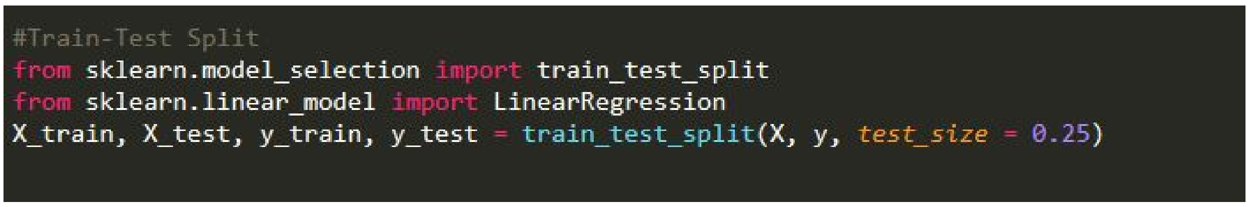
```python
#Import Dataset
df = pd.read_csv('dataset.csv')
#Extract Date from tpep_pickup_datetime & Create 'date' Column
def getDate(x):
    date_time= x.split(" ");
    return date_time[0]
df['date'] = df['tpep_pickup_datetime'].apply(getDate)

#Task : Add isHoliday Column to Dataframe df
# 1. Import uber_nyc_enriched dataset,
#    it specifies if there is holiday on given date in NYC
uber_df = pd.read_csv('uber_nyc_enriched.csv')
uber_df['date'] = uber_df['pickup_dt'].apply(getDate)

# 2. Create temporary DataFrame 'Holiday' with Col: hday,date
holiday=uber_df[['date','hday']]

# 3. Create a Holiday Map 'hmap' with
#    key=date ,Value= Y/N (Y:Holiday,N:No Holiday)
hmap=dict()
for i in range(len(holiday)):
    hmap[holiday['date'][i]]=holiday['hday'][i]

# 4. Create a column 'isHoliday' in DataFrame df using 'hmap'
def mapHoliday(x):
    if hmap[str(x)] == 'Y':
        return 1
    return 0
df['isHoliday']=df['date'].apply(mapHoliday)

# 5. Removal of irrelevant Columns
df.columns
col_to_remove = ['tpep_pickup_datetime', 'tpep_dropoff_datetime','Unnamed: 0','date']
df = df.drop(col_to_remove, axis=1)
```



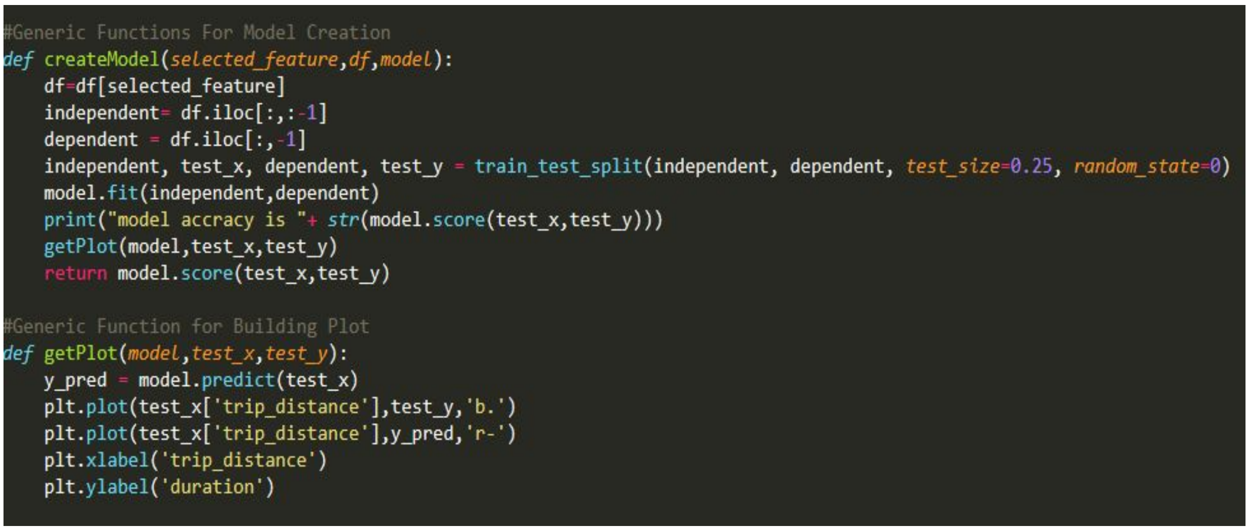
```
#Outlier Detection and Removal via Interquartile-Range
def find_outliers_tukey(x):
    q1 = x.quantile(.25)
    q3 = x.quantile(.75)
    iqr = q3 - q1
    floor = q1 - 1.5*iqr
    ceiling = q3 + 1.5*iqr
    outlier_indices = list(x.index[(x < floor) | (x > ceiling)])
    outlier_values = list(x[outlier_indices])
    #df.drop(df.index[outlier_indices], inplace=True)
    return outlier_indices, outlier_values

 columns_to_check_outliers=[
        'trip_distance', 'pickup_hrs', 'dropoff_hrs',
       'day_week', 'tpep_pickup_timestamp', 'tpep_dropoff_timestamp',
       'duration', 'speed'
]

for i in columns_to_check_outliers:
    temp1=i+'_indices'
    temp2=i+'_values'
    temp1, temp2 = find_outliers_tukey(df[i])
    print("Outliers for ",i)
    print(np.sort(temp2))
    df.drop(df.index[temp1], inplace=True)
```



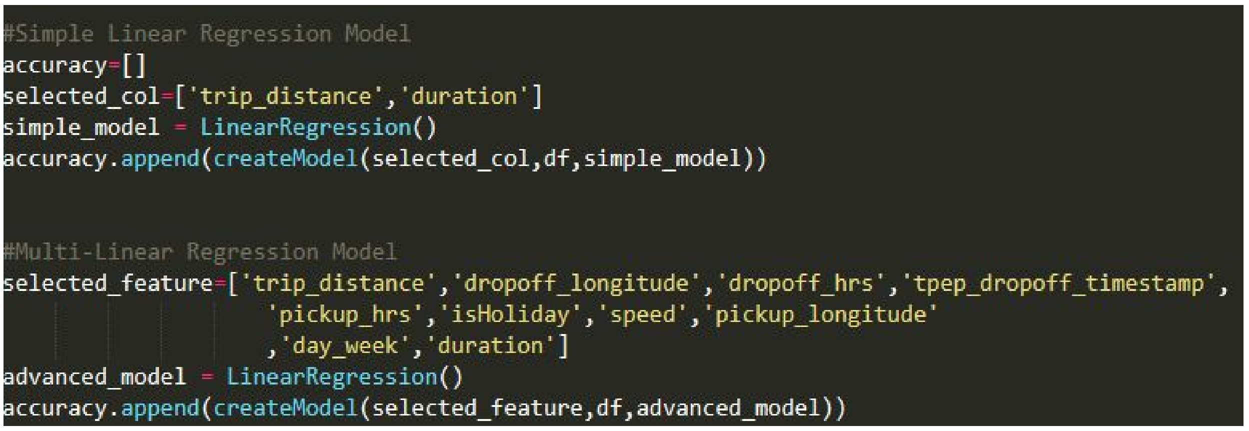
```
#Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
X = df.iloc[:,:-1]
y = df.iloc[:,-1]
y=y.astype('float')
bestfeatures = SelectKBest(score_func=chi2, k=12)
fit = bestfeatures.fit(X,y)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X.columns)
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Specs','Score']
print(featureScores.nlargest(12,'Score'))


#Feature Scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_df = scaler.fit_transform(df)

col=df.columns
for i in range(0,len(col)-1):
    df[col[i]]=scaled_df[:,i]
```



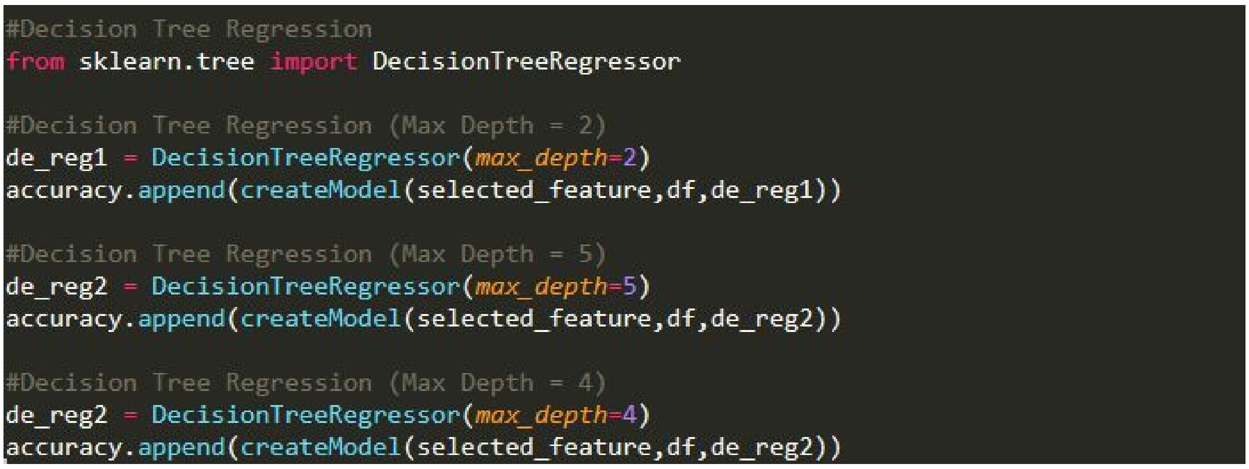
```
#Train-Test Split
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25)
```



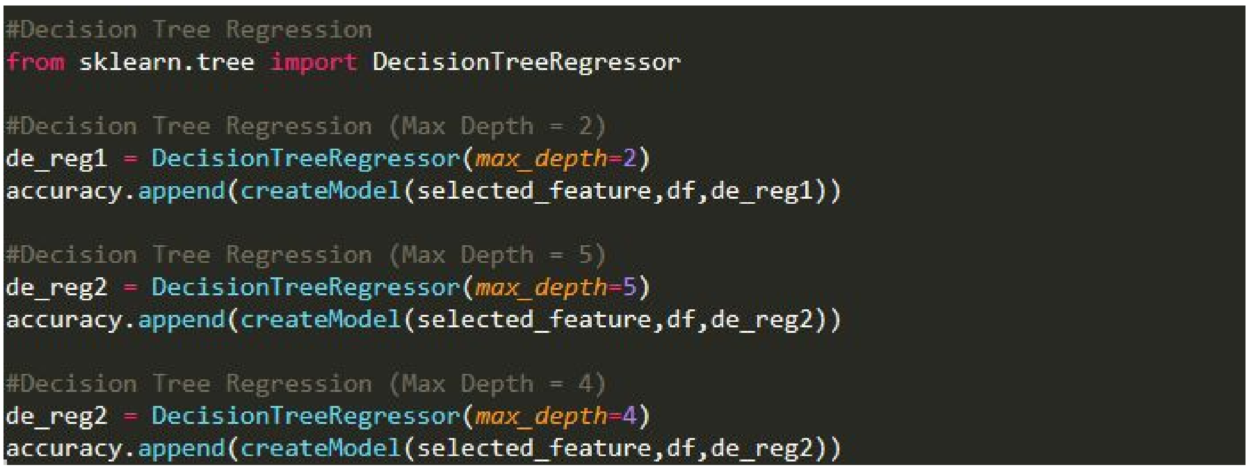
```
#Generic Functions For Model Creation
def createModel(selected_feature,df,model):
    df=df[selected_feature]
    independent= df.iloc[:,:-1]
    dependent = df.iloc[:,-1]
    independent, test_x, dependent, test_y = train_test_split(independent, dependent, test_size=0.25, random_state=0)
    model.fit(independent,dependent)
    print("model accracy is "+ str(model.score(test_x,test_y)))
    getPlot(model,test_x,test_y)
    return model.score(test_x,test_y)

#Generic Function for Building Plot
def getPlot(model,test_x,test_y):
    y_pred = model.predict(test_x)
    plt.plot(test_x['trip_distance'],test_y,'b.')
    plt.plot(test_x['trip_distance'],y_pred,'r-')
    plt.xlabel('trip_distance')
    plt.ylabel('duration')
```



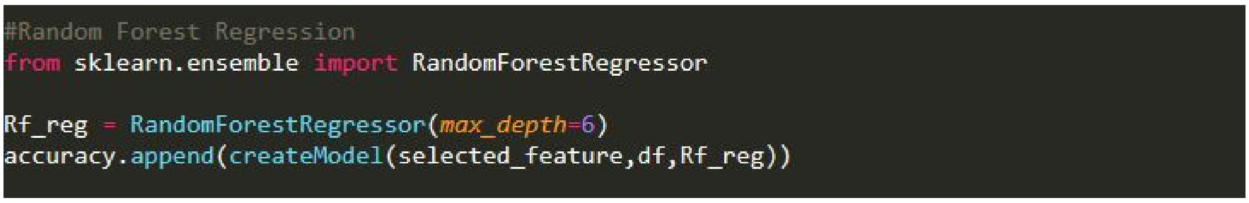
```
#Simple Linear Regression Model
accuracy=[]
selected_col=['trip_distance','duration']
simple_model = LinearRegression()
accuracy.append(createModel(selected_col,df,simple_model))


#Multi-Linear Regression Model
selected_feature=['trip_distance','dropoff_longitude','dropoff_hrs','tpep_dropoff_timestamp',
                  'pickup_hrs','isHoliday','speed','pickup_longitude'
                 ,'day_week','duration']
advanced_model = LinearRegression()
accuracy.append(createModel(selected_feature,df,advanced_model))
```



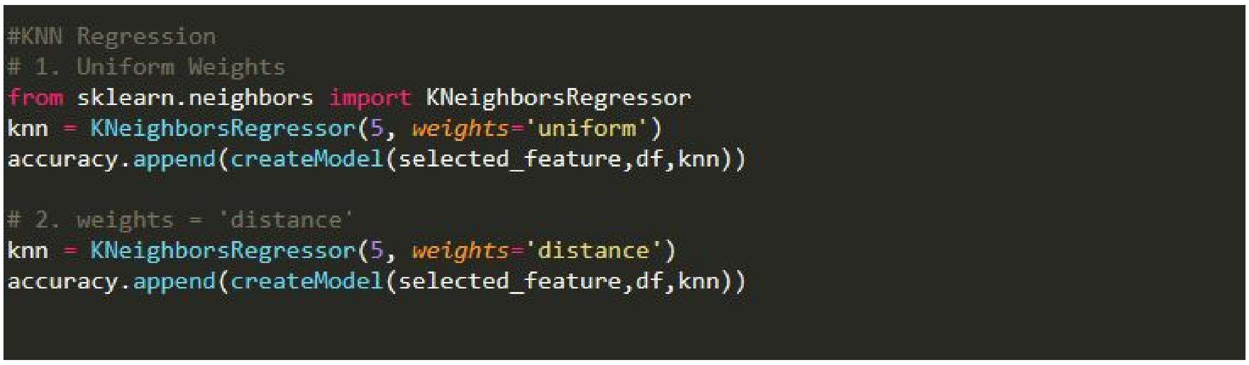
```
#Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor

#Decision Tree Regression (Max Depth = 2)
de_reg1 = DecisionTreeRegressor(max_depth=2)
accuracy.append(createModel(selected_feature,df,de_reg1))

#Decision Tree Regression (Max Depth = 5)
de_reg2 = DecisionTreeRegressor(max_depth=5)
accuracy.append(createModel(selected_feature,df,de_reg2))

#Decision Tree Regression (Max Depth = 4)
de_reg2 = DecisionTreeRegressor(max_depth=4)
accuracy.append(createModel(selected_feature,df,de_reg2))
```



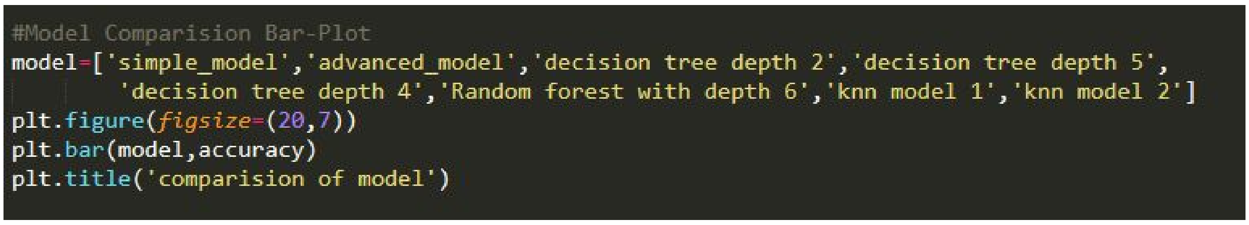
```python
#Decision Tree Regression
from sklearn.tree import DecisionTreeRegressor

#Decision Tree Regression (Max Depth = 2)
de_reg1 = DecisionTreeRegressor(max_depth=2)
accuracy.append(createModel(selected_feature,df,de_reg1))

#Decision Tree Regression (Max Depth = 5)
de_reg2 = DecisionTreeRegressor(max_depth=5)
accuracy.append(createModel(selected_feature,df,de_reg2))

#Decision Tree Regression (Max Depth = 4)
de_reg2 = DecisionTreeRegressor(max_depth=4)
accuracy.append(createModel(selected_feature,df,de_reg2))
```



```python
#Random Forest Regression
from sklearn.ensemble import RandomForestRegressor

Rf_reg = RandomForestRegressor(max_depth=6)
accuracy.append(createModel(selected_feature,df,Rf_reg))
```



```python
#KNN Regression
# 1. Uniform Weights
from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(5, weights='uniform')
accuracy.append(createModel(selected_feature,df,knn))

# 2. weights = 'distance'
knn = KNeighborsRegressor(5, weights='distance')
accuracy.append(createModel(selected_feature,df,knn))
```



```python
#Model Comparision Bar-Plot
model=['simple_model','advanced_model','decision tree depth 2','decision tree depth 5',
        'decision tree depth 4','Random forest with depth 6','knn model 1','knn model 2']
plt.figure(figsize=(20,7))
plt.bar(model,accuracy)
plt.title('comparision of model')
```

4.6 Results

4.6.1 Simple Linear Regression Model

```
Accuracy is 0.8034512948173701
```

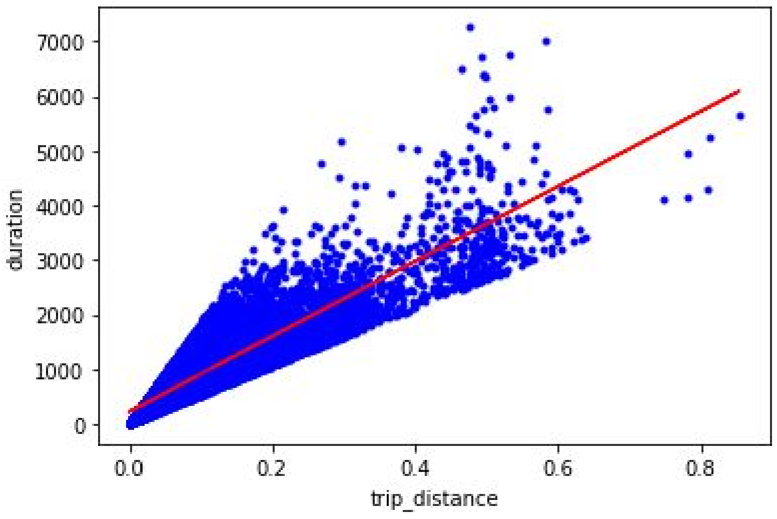


Fig 4.8 : Simple Linear Regression Model

4.6.2 Multiple Linear Regression Model

```
Accuracy is 0.903990850047781
```

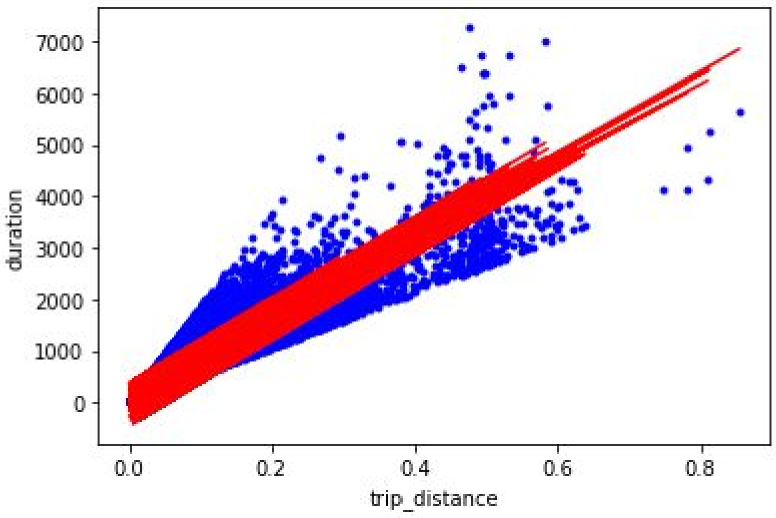
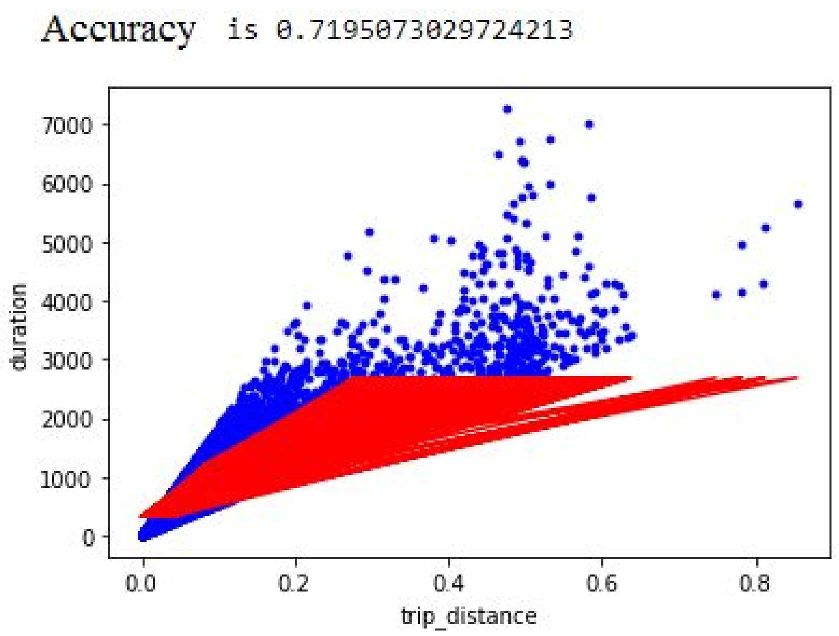


Fig 4.9 : Multiple Linear Regression Model

4.6.3 Decision Tree Regression Model

4.6.3.1 Decision Tree with MaxDepth=2



```
Accuracy is 0.7195073029724213
```

Fig 4.10 : Decision Tree with MaxDepth=2

4.6.3.1 Decision Tree with MaxDepth=5

```
Accuracy  is 0.9369273250258211
```

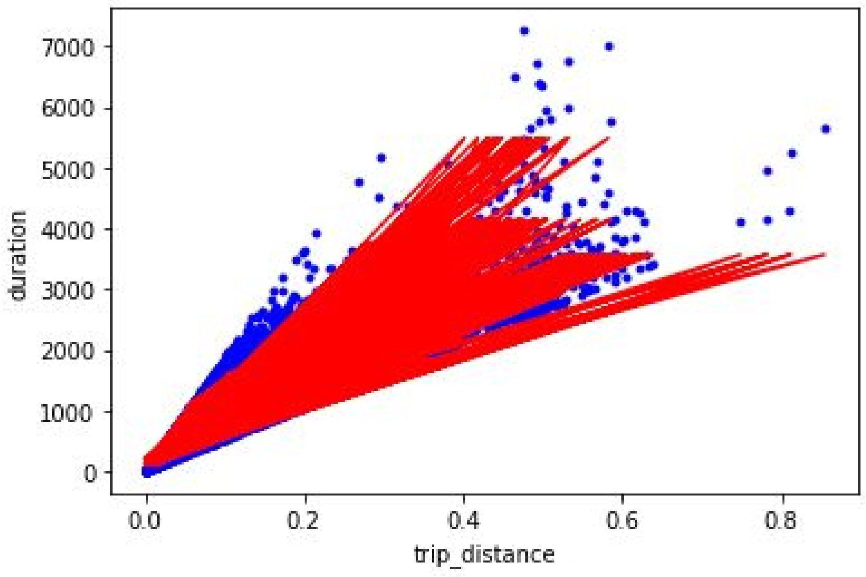


Fig 4.11 : Decision Tree with MaxDepth=5

4.6.3.1 Decision Tree with MaxDepth=4

```
Accuracy  is 0.897962661841011
```

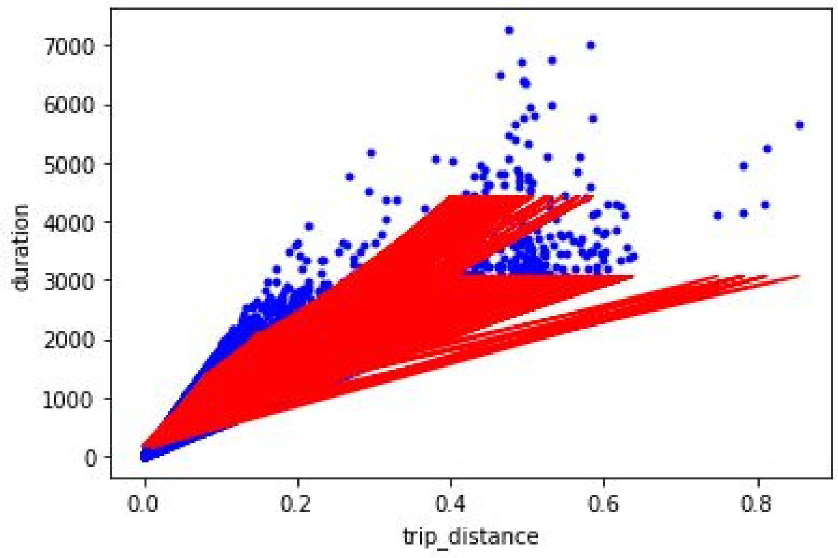


Fig 4.12 : Decision Tree with MaxDepth=4

4.6.4 Random Forest Regression Model

```
Accuracy is 0.9531350962385482
```

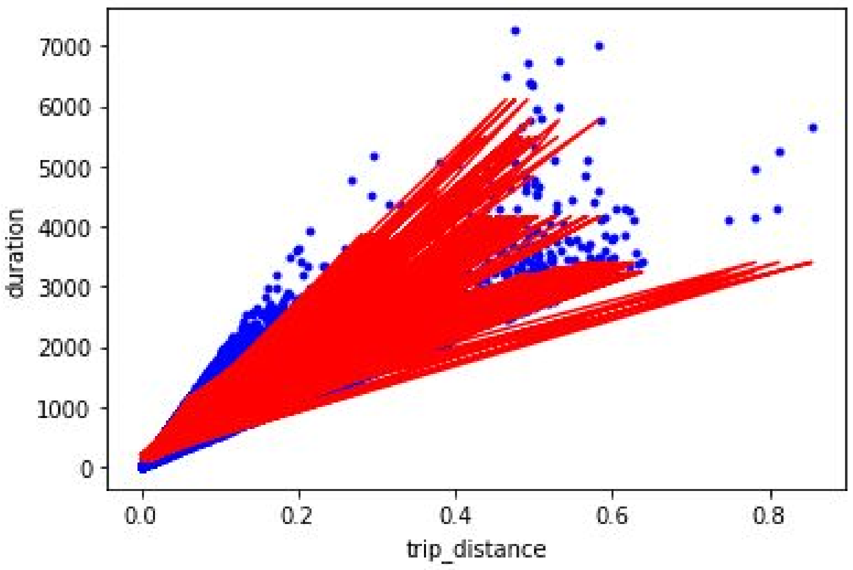


Fig 4.13 : Random Forest Regression with MaxDepth=6

4.6.5 KNN Regression

4.6.5.1 KNN Regression (Uniform Weight)

```
Accuracy is 0.9586222815537633
```

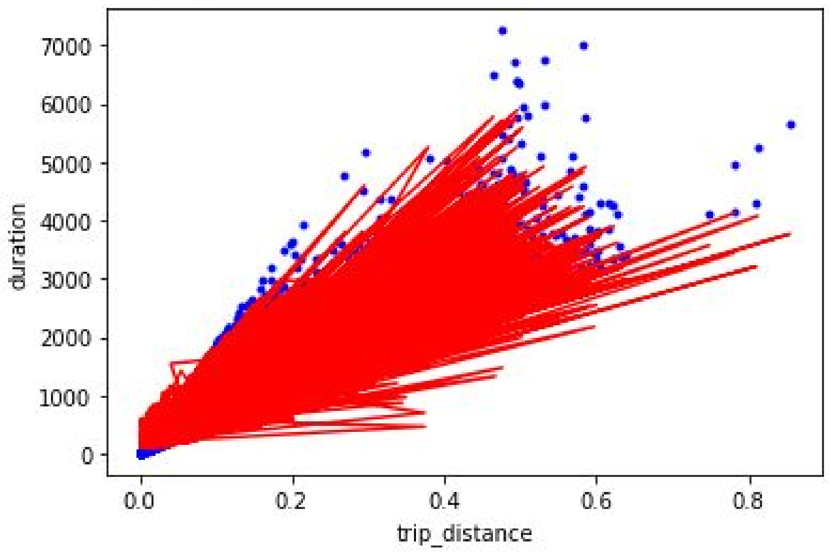


Fig 4.14 : KNN with Uniform Weight

4.6.5.1 KNN Regression (Weight=distance)

```
Accuracy is 0.9678863739882441
```

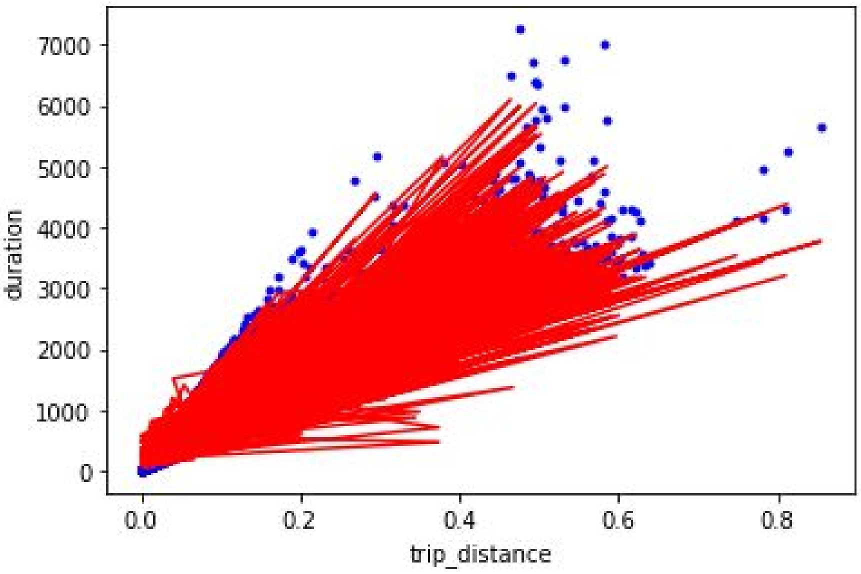


Fig 4.15 : KNN with Non Uniform Weight

4.7 Comparision

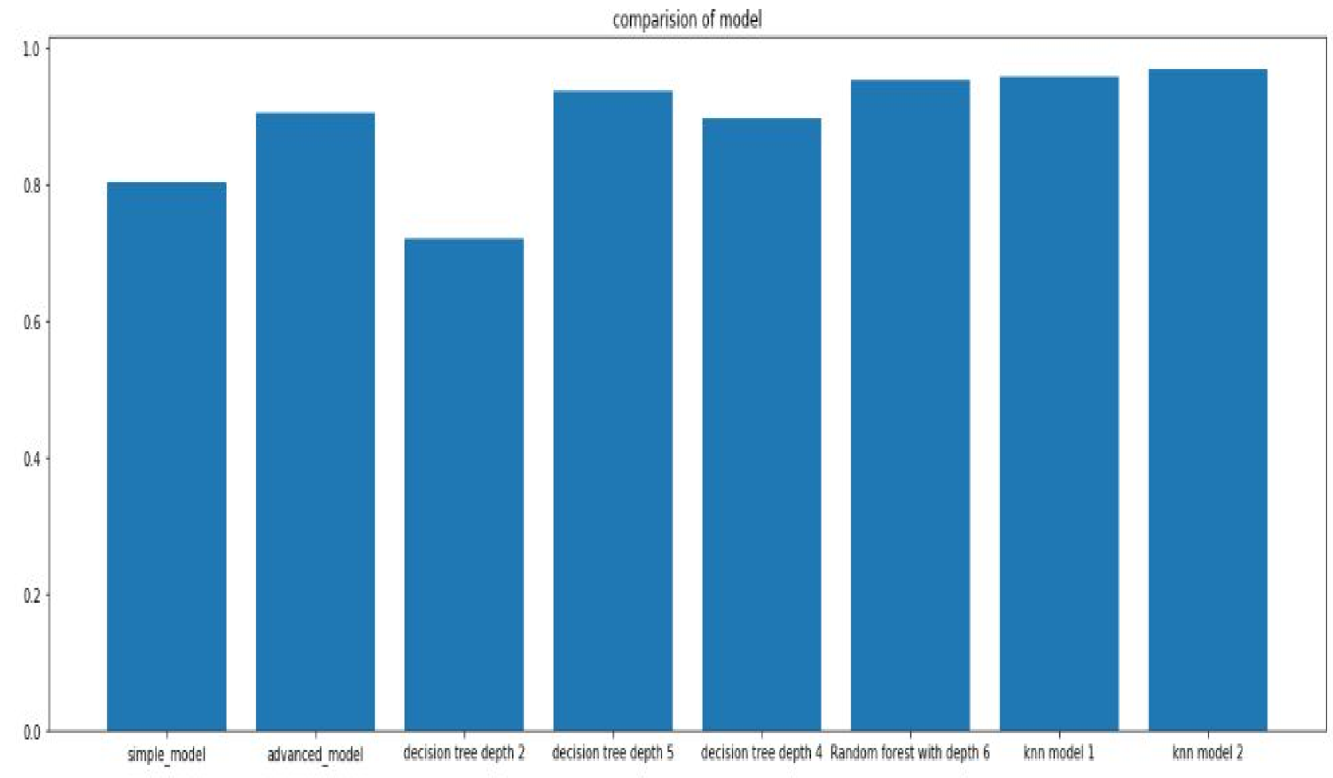


Fig 4.15 : Comparision of Various Models

5. Tools and technology to be used (hardware and software)

5.1. Software Requirements

The following software's were used for this project:

Operating System	:	Microsoft® Windows® 10, Ubuntu 16.04 or above.
Python 3.8	:	Programming Language for execution of the code.
Pandas	:	Library for the Python for data Manipulation and Analysis.
Numpy	:	Mathematical and logical operations on arrays.
SKLearn	:	Machine learning library for Python.
MatPlotLib	:	Object-oriented API for embedding plots in applications.

5.2. Hardware Requirements

The following hardware configuration were used and at least required to run the various software's for this project:

Processor	**:**	**Intel® Core™ i5 CPU and above**
Memory	**:**	**4GB RAM**
Storage required	**:**	**Maximum of 1GB**

6. Conclusion

On Comparing all the various predictions models described in this project,The accuracy Comparision Table is show Below:

Table 6.1 Model Comparision

Model	**Accuracy**
Simple Linear Regression	0.8034512948173701
Multiple Linear Regression	0.903990850047781,
Decision Tree Regression (maxdepth:2)	0.7195073029724213
Decision Tree Regression (maxdepth:5)	0.9369273250258211
Decision Tree Regression (maxdepth:4)	0.897962661841011
Random Forest Regression	0.9531350962385482
K Nearest Neighbour (Uniform Weight)	0.9586222815537633
K Nearest Neighbour (Non Uniform Weight)	0.9678863739882441

We find that Random Forest and K Nearest Neighbour Regression gives the Best Result (Accuracy ~= 96%)

However a more realistic approach to solve the problem statement would be to get dynamic data or real data via Cab service provider's API. We aim to carry this work ahead using

dynamic data sets via API's, getting real data and using other algorithms such as Stochastic gradient
descent to train the model and make predictions.
We can also try to consider other factors that affect travel time such as the traffic information ,weather information ,Road Conditions subjected to availability of dataset.

7. Referecnces

[1] Chen, M., Chien, S.: Dynamic freeway travel time prediction using probe vehicle data: Link-based vs. Path-based. J. of Transportation Research Record, TRB Paper No. 01-2887, Washington, DC, pp. 157-161 (2001)

[2] Chun-Hsin, W., Chia-Chen, W., Da-Chun, S., Ming-Hua, C., Jan-Ming, H.: Travel Time Prediction with Support Vector Regression. In: IEEE Intelligent Transportation Systems Conference, vol. 2, pp. 1438–1442 (2003)

[3] Kwon, J., Petty, K.: A travel time prediction algorithm scalable to freeway networks with many nodes with arbitrary travel routes. In: Transportation Research Board 84th Annual Meeting, Washington, DC, pp. 147–153 (2005)