Cab Fare Prediction Project

-By Rakshith R

07-09-2020

Contents

Chapter 1: Introduction

1.1 Problem Statement

The project is about a cab rental company who has its historical data. The objective of the project is to build a model which predicts the fare amount of the Cab rental in the city. This Fare prediction takes in consideration the parameters like distance, No of passengers, date/time and other factors from historical data which was collected from the pilot project for the same. These predicted values would help the business to fix the fare amount for particular number of passengers, peak time or busy days and for particular distance.

1.2 DATA

We have two datasets:
a. Train Data
b. Test Data

There are 7 variables in our train data in which 6 are independent variables and 1 is dependent variable.

Dependant Features: - Fare amount, which exist only in train data set.
Independent Features: - Pickup datetime, Pickup longitude, Pickup latitude, Dropoff longitude, Dropoff latitude, Passenger count. This features are common for both the data sets.

Longitude and Latitude are the geometrical coordinates

The details of data attributes in the dataset are as follows:-

- **Pickup Datetime** - timestamp value indicating when the cab ride started.
- **Pickup Longitude** - float for longitude coordinate of where the cab ride started.
- **Pickup Latitude** - float for latitude coordinate of where the cab ride started.
- **Dropoff Longitude** - float for longitude coordinate of where the cab ride ended.
- **Dropoff Latitude** - float for latitude coordinate of where the cab ride ended.
- **Passenger Count** - an integer indicating the number of passengers in the cab ride.

Size of Train Dataset provided: - 16067 observations, 7 Variables (including dependent variable)
Size of Test Dataset provided: - 9914 observations, 6 Variables

Chapter 2: Methodology

Methodology mainly consists of following processes,

1. Pre-processing:
   It includes data understanding, missing value analysis, outlier analysis, feature selection and feature scaling.
2. Model development:
   It includes identifying suitable Machine learning Algorithms and applying those algorithms in our given dataset.

2.1 Pre-Processing

Data pre-processing is the first stage of any type of project. In this stage we get the feel of the data. A predictive model requires that we look at the data before we start to create a model. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, Imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.

Further we will look into what pre-processing steps do this project was involved in.

2.1 .1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach used to analysing data sets to summarize the main characteristics.
A snapshot of the Train dataset is mentioned following.

	fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	4.5	2009-06-15 17:26:21 UTC	-73.844311	40.721319	-73.841610	40.712278	1.0
1	16.9	2010-01-05 16:52:16 UTC	-74.016048	40.711303	-73.979268	40.782004	1.0
2	5.7	2011-08-18 00:35:00 UTC	-73.982738	40.761270	-73.991242	40.750562	2.0
3	7.7	2012-04-21 04:30:42 UTC	-73.987130	40.733143	-73.991567	40.758092	1.0
4	5.3	2010-03-09 07:51:00 UTC	-73.968095	40.768008	-73.956655	40.783762	1.0

A snapshot of the Test dataset is mentioned following.

	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
0	2015-01-27 13:08:24 UTC	-73.973320	40.763805	-73.981430	40.743835	1
1	2015-01-27 13:08:24 UTC	-73.986862	40.719383	-73.998886	40.739201	1
2	2011-10-08 11:53:44 UTC	-73.982524	40.751260	-73.979654	40.746139	1
3	2012-12-01 21:12:12 UTC	-73.981160	40.767807	-73.990448	40.751635	1
4	2012-12-01 21:12:12 UTC	-73.966046	40.789775	-73.988565	40.744427	1

In the given data set there are 7 variables and data types of all variables are object, float64 or int64.

```
fare_amount           object
pickup_datetime       object
pickup_longitude     float64
pickup_latitude      float64
dropoff_longitude    float64
dropoff_latitude     float64
passenger_count      float64
dtype: object
```

We can observe that the "fare_amount" and "pickup_datetime" are of object data types which needs to be converted. So data type of pickup_datetime variable was changed from object to datetime. Also the "fare_amount" variable was converted from object to numeric

2.1.2 Missing value Analysis

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. If a columns has more than 30% of data as missing value either we ignore the entire column or we ignore those observations.
In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks or because of reasons like, incomplete submission, wrong input, manual error etc. Some missing values are in form of NA or Missing values left behind after outlier analysis; missing values can be in any form. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.

We observed No missing values present in the Test dataset. But we observed few missing values in Train Dataset.

Below is the table which shows the number of missing values present in Train dataset and also with the missing values percentage of each variable.

	Variables	Missing values	Missing_Value_Percentage
0	passenger_count	55	0.342317
1	fare_amount	24	0.149374
2	pickup_datetime	0	0.000000
3	pickup longitude	0	0.000000
4	pickup latitude	0	0.000000
5	dropoff_longitude	0	0.000000
6	dropoff_latitude	0	0.000000

From the above table we can observe that null values are very less in our data set i.e. less than 1%. So we can delete the observations having missing values.

2.1.3 Data extraction

The variable pickup_datetime was changed from object to datetime and picked up some important values from it. Later the "pickup_datetime" column was separated into fields like year, month, day, day of the week, hour, Minute etc.

Also, we tried to find out the distance using the **haversine** formula which says:
After checking on the internet I found a formula called The haversine formula, that determines the distance between two points on a sphere based on their given longitudes and latitudes. The haversine formula determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Calculating the distance based on latitude and longitude: we are having the values of latitude and longitude, hence we can calculate the distance travelled by a passenger so that we can have only one input feature instead of four. This helps in reduction of dimensions of input features which helps improving the model accuracy. We will calculate the distance using haversine formula. These formula calculates the shortest distance between two points in a sphere.

So our new extracted variables are:

- fare_amount
- pickup_datetime
- pickup_longitude
- pickup_latitude
- dropoff_longitude
- dropoff_latitude
- passenger_count
- year
- Month
- Date
- Day
- Hour
- Minute
- Distance

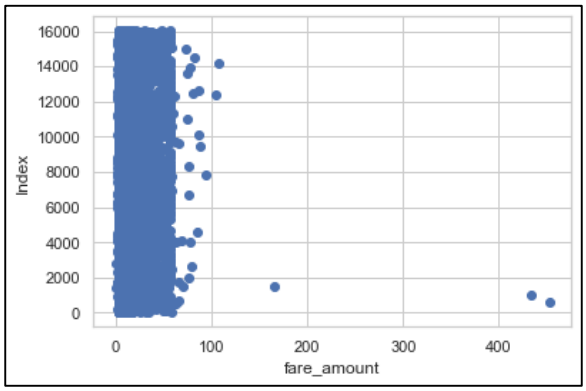
2.1.4 Outlier Analysis

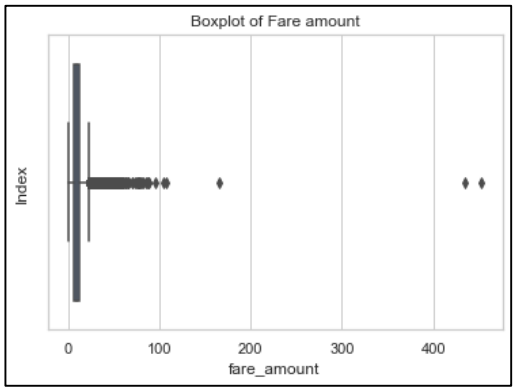
One of the other steps of pre-processing is to check the presence of outliers. Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error, poor quality of data and it is correct but exceptional data. This will create an error in predicting the target variables and can hamper our data model.
At first we remove the extreme values or values which are not practical manually.

- We know that Latitude ranges from (-90 to 90) and Longitude ranges from (-180 to 180). So we drop values which are not in that range.
- Any cab including SUV cannot have more than 6 passengers, so we are dropping rows which includes more than 6 passengers. Also we observe few values which are in decimal, so we drop them.
- We could observe above that there is huge difference in first three values of fare amount. So first two values seems to be outlier in fare_amount, so drop them initially. Also drop zero and negative values

- When we observe the distance variable after first 23 values, distance goes down suddenly to 129 km, so drop rows which includes distance above 130km. Also, distance cannot be 0 km, so drop the rows which includes distance 0 km.
- In Pick up time variable we observe one observation having random value which is not in format, so we drop it.

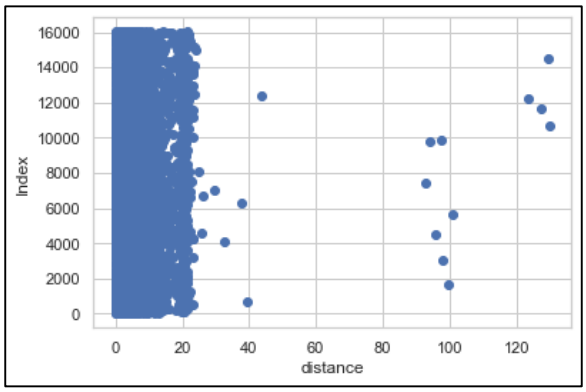
- Later the box plot and scatter plot of fare amount was plotted to check for the outliers.

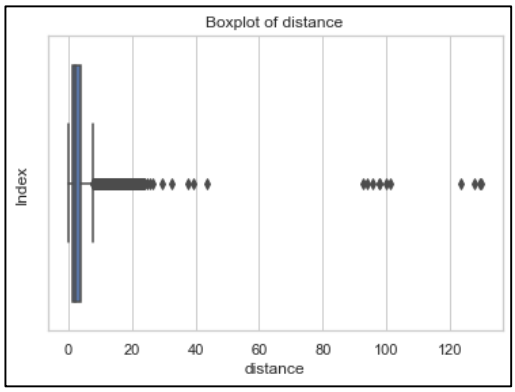




After observing the scatter and box plot, values above 100 were removed as outliers.

- Also the box plot and scatter plot of distance was plotted to check for the outliers.





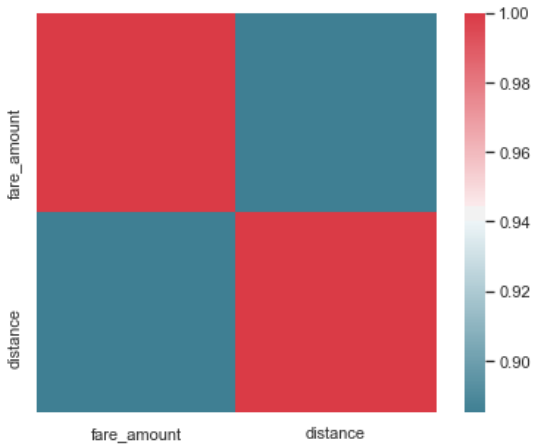
After observing the scatter and box plot, values above 30 were removed as outliers.

2.1.5 Feature Selection

Feature Selection is used to reduce the complexity of a model and make it easier to interpret. It even reduces over fitting. Sometimes in our data, all the variables may not be accurate enough or required to predict the target variable, in such cases we need to analyse our data, understand the data and select the variables which are most useful for our model. In such cases we apply feature selection. Feature selection helps in reducing computational time of model. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multi-collinearity between variables. The highly collinear variables are dropped and then the model is executed.

Since the pickup date time is splitted into different variables like month, year, day so on and also distance variable has been created using pickup and drop longitudes/latitudes, we will drop pickup date time, pickup and drop longitudes and latitudes variables on both Train and Test data.

- Correlation Analysis for Numerical Variables.



We could observe from the above plot that fare amount and distance variables are not highly correlated.

- **Checking VIF for multicollinerity**

```
const              1.175167e+06
passenger_count    1.002383e+00
year               1.015269e+00
Month              1.015250e+00
Date               1.001272e+00
Day                1.010594e+00
Hour               1.010571e+00
distance           1.003535e+00
dtype: float64
```

From the above VIF values since they are less than 10 for each variable, there is no multicolinerity exists.

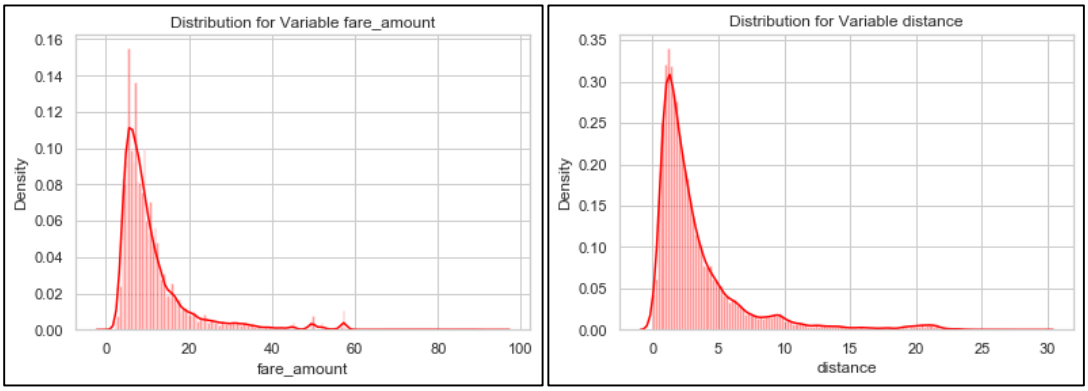
Finally after feature selection, following variables were dropped from both train and test data,

- pickup_datetime, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude and Minute.

2.1.6 Features Scaling

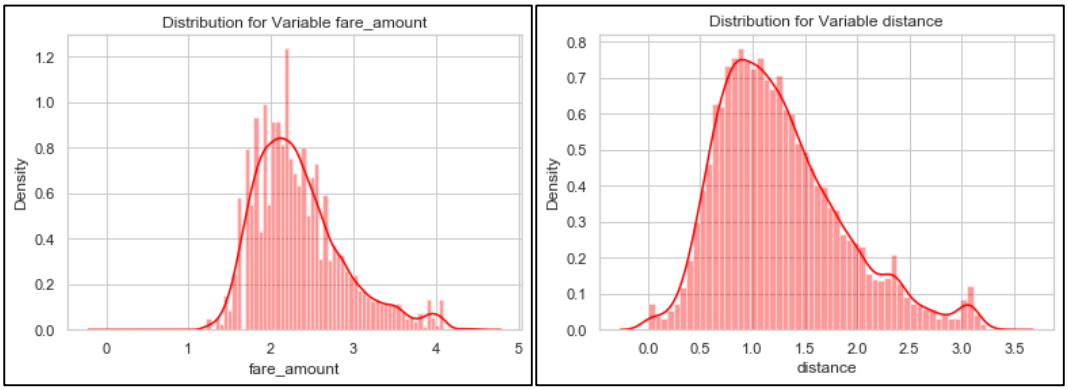
- Feature scaling is a method used to standardize the range of independent variables or features of data. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. When you normalize data you eliminate the units of measurement for the data, enabling you to more easily compare data from different places. Some of the more common ways to normalize data is by transforming data using a z-score or t-score. This is usually called as standardization.
- **Skewness** is asymmetry in the statistical distribution, where the curve appears to be distorted or skewed either to left or to right. Skewness can be quantified to define the extent to which the distribution differs from the normal distribution. Here we have tried to show the skewness of our variables and we find that our target variable is one sided skewed so by using **log transform** technique we tried to reduce the skewness of the same.

- Below mentioned graphs shows the probability distribution plot to check distribution before log transformation for fare amount and distance variables of train dataset:



- From the above graph we observe that the distribution of "fare_amount" and "distance" are right skewed. So to get right predictions we transform these values of two columns using logarithmic function.

- Below mentioned graphs shows the probability distribution plot to check distribution before log transformation for fare amount and distance variables of train dataset:

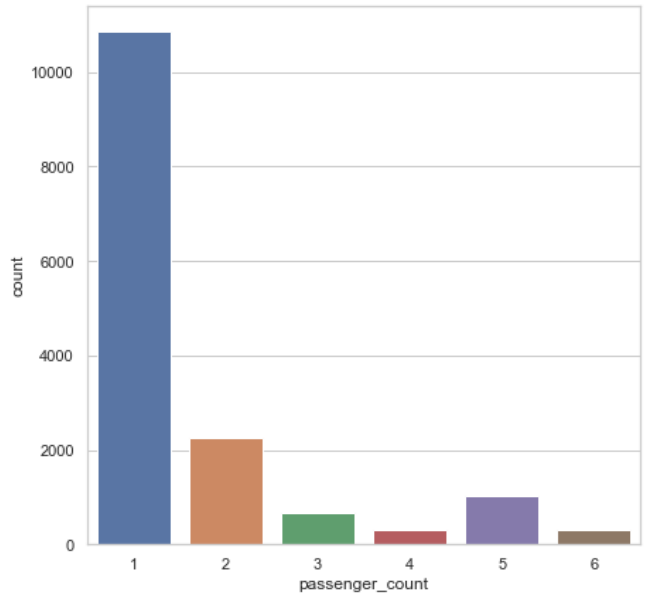


- From the above graph we observe that the distribution of "fare_amount" and "distance" are not skewed and hence they are ready for the Training of a model.

2.1.7 Data Visualization

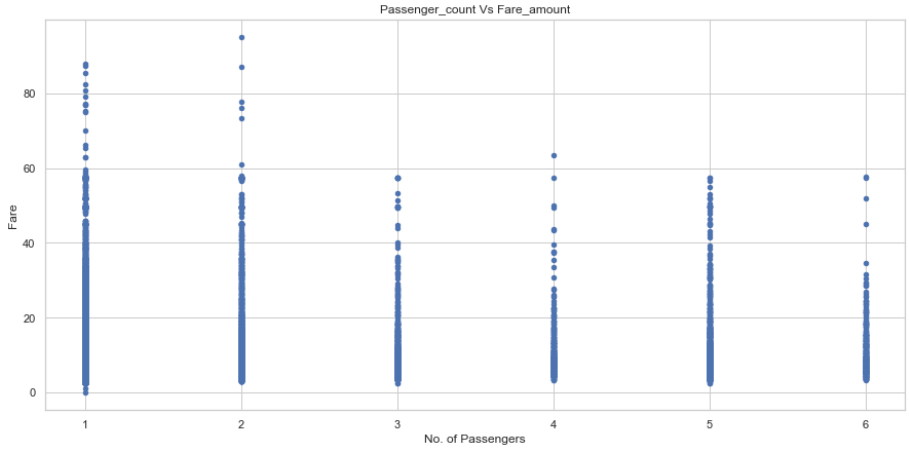
Data Understand is a process wherein we get know our data in a better way by the help of visual representations and come up with initial ideas for developing our model. Here, the specific variables are plotted with respect to the target variable. In some cases two variables are compared and plotted together for our better understanding and visualization.
For better understanding of data, here we have plotted some visualization for the variables.

- Passenger count visualization



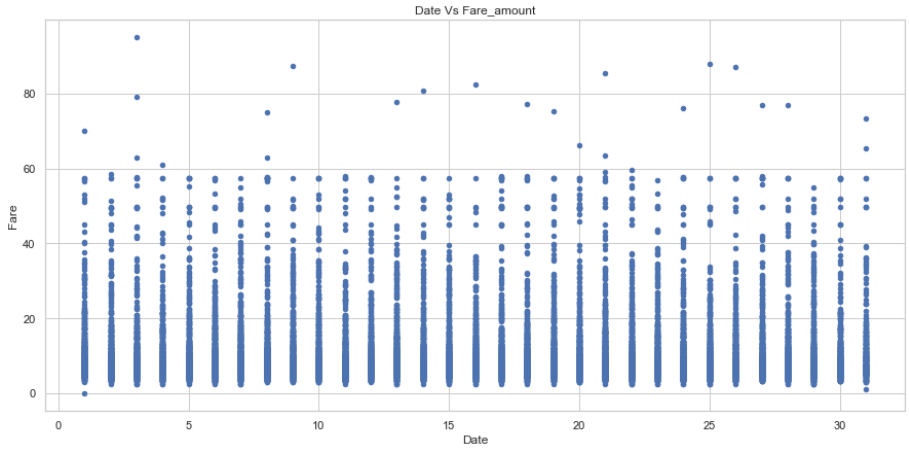
From the above graph it can be observe that the most of the rides were availed by one or two passengers at a time

- Relationship between number of passengers and Fare amount



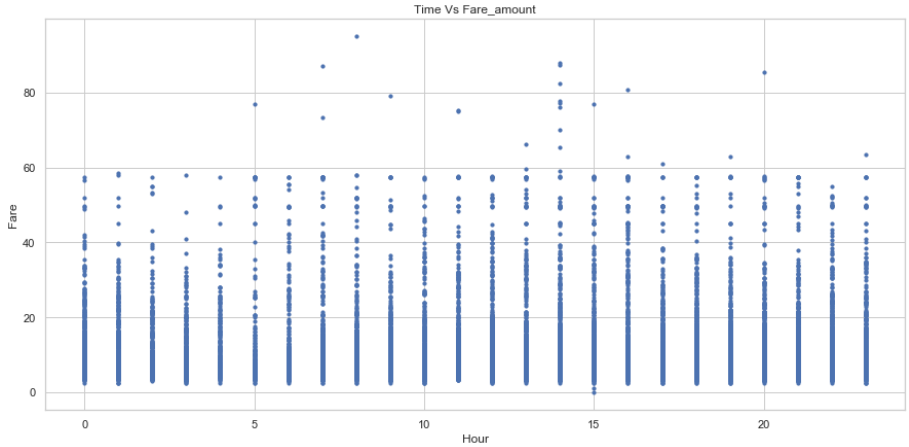
From this graph it is observed that the revenue is more from the rides that are availed by one or two passengers at a time

- Relationship between date and Fare amount



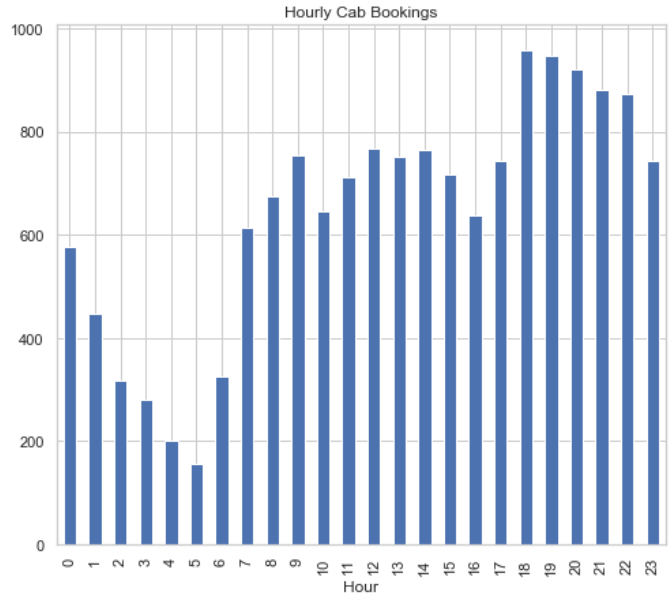
From the above graph it is seen that highest fare was charged on 3rd and 24th of the month

- Relationship between Time and Fare amount



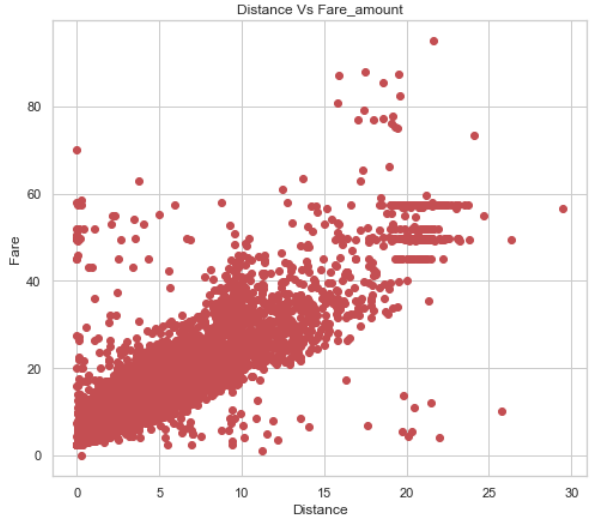
The highest fare was 8AM in the morning, 2pm in afternoon and 8PM in the night of a day

- Visualisation for hourly cab bookings



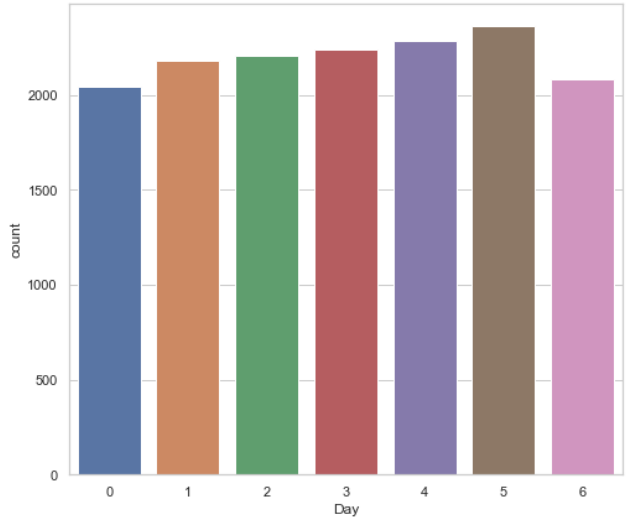
We can confirm that least number of rides were at 5AM and more number of rides were taken at 6PM and 7PM, hence the high number of cars can be arranged at those peak hours

- Relationship between distance and fare amount



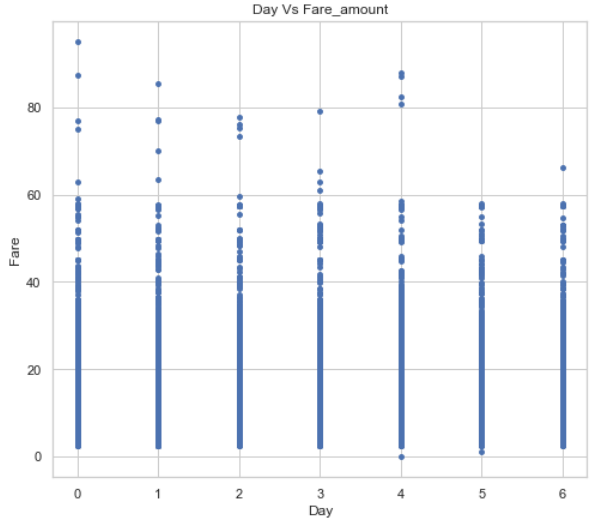
Most no of rides were taken between the distances 0 to 30kms, And also the highest fare being charged with in this limit.

- Impact of Day on the number of cab rides



We can see that the day is not impacting much on the number of rides

- Relationships between day and Fare amount



The highest fare was charged on Monday, Thursday and Friday.

2.2 Model Development

After cleaning the data by removing outliers and missing values and pre-processing the data, next step would be Model Development. Now we are having data ready to be implemented for developing a model. There are number of models and machine learning algorithms that can used to develop model. It includes decision tree, random forest, KNN, Naïve Bayes, Linear regression, Logistic Regression etc. So, before implementing any model we should choose precisely our model. So, the first step in the Model Development is selection of model.

2.2.1 Model Selection

First step in selecting the suitable machine learning algorithm for a problem statement is to categorize, analysing and understanding the data. There are many categories in which a problem may lie like forecasting, classification, optimisation, unsupervised learning etc. If the output of the model is a number, it's a regression problem. If the output of the model is a class, it's a classification problem. If the output of the model is a set of input groups, it's a clustering problem.

Choosing a right machine learning algorithm depends on many factors including, but not limited to: data size, quality and diversity, also what answers businesses want to derive from data.

The process of selecting suitable model depends on our goal and the problem statement. In this project the goal of the Project is to build a model which predicts the fare amount of the Cab rental in the city. Thus, the problem statement is an identified as regression problem and it will fall under the category of forecasting where we need to forecast a numeric data or continuous variable for the target.

The dependent variable in our model is a continuous variable i.e., fare amount of the cab rental. Therefore the models that we choose in this project are Linear Regression, Decision Tree and Random Forest.

2.2.1.1 Linear Regression

Linear Regression is one of the statistical method of prediction. It is most common predictive analysis algorithm. It is used to predict the value of variable Y based on one or more input predictor variables X. The goal of this method is to establish a linear relationship between the predictor variables and the response variable. Such that, we can use this formula to estimate the value of the response Y, when only the predictors (X- Values) are known.

2.2.1.2 Decision tree

Decision Tree is a supervised learning predictive model which is used to predict the data for classification and regression. It uses a set of binary rules in order to calculate the target value/dependent variable. It uses a tree-like model of decisions. A decision tree can be used to visually and also explicitly represent decision making. It accepts both continuous and categorical variables.

Decision trees are divided into three main parts this are:

- Root Node : performs the first split
- Terminal Nodes : that predict the outcome which are also called leaf nodes
- Branches: arrows connecting nodes which shows the flow from root to other leaves.

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with "and" and multiple branches are connected by "or". It provides its output in the form of rule, which can easily understood by a non-technical person also

2.2.1.3 Random Forest

The next model to be followed in this project is Random forest. Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build 'n' number of trees in order to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data.

The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important. Random decision forests correct for decision tree's habit of over fitting to their training set.

CHAPTER 3: EVALUATION OF THE MODEL

Once the models are developed for predicting the target variable, next step is evaluate the models and identify which one is most suitable for deployment. To evaluate the model, error metrics are used. In this project, we will be using MAPE, RMSE and R Square as error metrics

3.1 Error Metrics

- **MAPE (Mean Absolute Percent Error)**
  MAPE is a measure of prediction accuracy of a forecasting method. It is the measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.
  MAPE is calculated using below expression,

$$\left(MAPE = \frac{1}{n}\sum \frac{|Actual - Forecast|}{|Actual|}\right) * 100$$

- **Accuracy**
  It is the ratio of number of correct predictions to the total number of predictions made.

  Accuracy= number of correct predictions / Total predictions made

  It can also be calculated from MAPE as
  Accuracy = 1- MAPE

- **RSME**
  RMSE (Root Mean Square Error) is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted value. RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

$$RSME = \sqrt{\frac{\sum_{i=1}^{n} (X_{obs,i} - X_{model,i})^2}{n}}$$

- **R Square**
  R Square is another metric that helps us to know the Correlation between original and predicted values. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared explains how much variance of dependent variable explained by the independent variable. It is a measure of goodness of fit in regression line. R-squared is a relative measure of fit.
  R-squared has the useful property that its scale is intuitive: it ranges from zero to one, where 0 means independent variable is unable to explain the target variable and 1 means the target variable is completely explained by the independent variable. Improvement in the regression model results in proportional increases in R-squared.

- We have shown both train and test data results, the main reason behind showing both the results is to check whether our data is over fitted or not.

- Lower values of MAPE and higher value of R-Squared Value indicate better fit of model.

- Lower values of RMSE indicate better fit and that the model can relatively predict the data accurately.

3.1.1 MAPE (Mean Absolute Percent Error) and Accuracy

We evaluated the above metrics in both R and python and are compiled and shown below,

- **In python**

Method	MAPE	Accuracy
Linear Regression	7.38	92.62
Decision Tree	9.41	90.59
Random Forest	2.78	97.22

- **In R**

Method	MAPE	Accuracy
Linear Regression	7.07	92.93
Decision Tree	8.34	91.66
Random Forest	3.85	96.15

The model which has lowest MAPE should be chosen as a suitable Model. Lower the MAPE value, higher the accuracy of the model. Here, from Python and R values we can observe that Random Forest having lower MAPE value as a better model As, Accuracy is based on MAPE percentage, here also it is found that Random Forest is good models for the given data set. But we need more error metrics to cross check this. So, we go for RMSE and R Square which is a better error metric.

3.1.2 RMSE

We evaluated the above metrics in both R and python and are compiled and shown below,

- **In python**

Method	Train data	Test data
Linear Regression	0.257	0.235
Decision Tree	0.291	0.281
Random Forest	0.093	0.239

- **In R**

Method	Train data	Test data
Linear Regression	0.246	0.249
Decision Tree	0.266	0.268
Random Forest	0.124	0.242

Lower values of RMSE indicate better fit. Here, from Python and R values we can observe that Random Forest having lower RMSE value as a better model. But we need more error metrics to cross check this. So, we go for R Square as next error metric.

3.1.3 R Square

We evaluated the above metrics in both R and python and are compiled and shown below,

- **In python**

Method	Train data	Test data
Linear Regression	0.77	0.81
Decision Tree	0.71	0.72
Random Forest	0.97	0.80

- **In R**

Method	Train data	Test data
Linear Regression	0.79	0.78
Decision Tree	0.75	0.75
Random Forest	0.95	0.80

The model which has highest R Square value should be chosen as a suitable Model. R Square is identified as a better error metric to evaluate models. When we observe the values from above table, we choose the model with highest R Square as a suitable Model. Here, from both R and Python it is found that Random Forest is a best fit model for the given data.

We can see that in both R and Python **Random Forest Model** fits the best out of Decision Tree and Linear Regression. RMSE Value is the lowest for Random Forest. Also R square value being highest and with good accuracy. So, to improve the model and enhance its performance, we apply **Hyperparameter tuning** techniques on Random Forest model in next section.

3.2 Hyper Parameters Tunings for optimizing the results

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. Hyperparameter optimization finds a tuple of hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data. The objective function takes a tuple of hyperparameters and returns the associated loss. Model hyperparameters are set by the data scientist ahead of training and control implementation aspects of the model. The weights learned during training of a linear regression model are parameters while the number of trees in a random forest is a model hyperparameter because this is set by the data scientist.
Hyperparameters can be thought of as model settings. These settings need to be tuned for each problem because the best model hyperparameters for one particular dataset will not be the best across all datasets. The process of hyperparameter tuning (also called hyperparameter optimization) means finding the combination of hyperparameter values for a machine learning model that performs the best - as measured on a validation dataset - for a problem.

Here we have used two hyper parameters tuning techniques

- Random Search CV
- Grid Search CV

3.2.1 Random Search CV

Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. It is similar to grid search, and yet it has proven to yield better results comparatively. The drawback of random search is that it yields high variance during computing. Since the selection of parameters is completely random; and since no intelligence is used to sample these combinations, luck plays its part.

As random values are selected at each instance, it is highly likely that the whole of action space has been reached because of the randomness, which takes a huge amount of time to cover every aspect of the combination during grid search. This works best under the assumption that not all hyperparameters are equally important. In this search pattern, random combinations of parameters are considered in every iteration. The chances of finding the optimal parameter are comparatively higher in random search because of the random search pattern where the model might end up being trained on the optimised parameters without any aliasing.

Below is the table of values from python,

Error Metrics	Values
RMSE	0.236
R square	0.81

```
Randomized Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'n_estimators': 15, 'max_depth': 9}
```

3.2.2 Grid Search CV

Grid Search CV algorithm set up a grid of hyperparameter values and for each combination, train a model and score on the validation data. In this approach, every single combination of hyperparameters values is tried which can be very inefficient.

In this tuning technique, we simply build a model for every combination of various hyperparameters and evaluate each model. The model which gives the highest accuracy wins. The pattern followed here is similar to the grid, where all the values are placed in the form of a matrix. Each set of parameters is taken into consideration and the accuracy is noted. Once all the combinations are evaluated, the model with the set of parameters which give the top accuracy is considered to be the best.

Below is the table of values from python,

Error Metrics	Values
RMSE	0.234
R square	0.81

```
Grid Search CV Random Forest Regressor Model Performance:
Best Parameters = {'max_depth': 7, 'n_estimators': 18}
```

3.3 Conclusion

Upon evaluating the above error metrics, we can conclude that **Random Forest** is the better model for our analysis. And we choose **Grid search CV Random Forest model** for fare amount prediction for cab rental.

3.4 Output

Below is the snap of Output data (from python) with predicted fare amount,

Index	passenger_count	year	Month	Date	Day	Hour	distance	Predicted_fare
0	1	2015	1	27	1	13	2.323259	9.947734
1	1	2015	1	27	1	13	2.425353	10.275658
2	1	2011	10	8	5	11	0.618628	4.952214
3	1	2012	12	1	5	21	1.961033	8.124924
4	1	2012	12	1	5	21	5.387301	15.083241

APPENDIX A – R Code

```
#Clean the environment
rm(list=ls())

#Set Working Directory
setwd("C:/Users/User/Desktop/edwisor/New folder")

#get Working directory
getwd()

#Load the librarires
libraries = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information", "MASS", "rpart", "gbm", "ROSE", 'sampling',
'DataCombine',  'inTrees',"tidyr")


install.packages(libraries)
lapply(X = libraries,require, character.only = TRUE)
rm(libraries)

train = read.csv("train_cab.csv", header = T, na.strings = c(" ", "", "NA"))
test = read.csv("test.csv")

###############EXPLORATARY DATA ANALYSIS ######################
# Summarizing  data

#Verify first five rows of data

head(train)

head(test)

#Dimensions of data
dim(train)

dim(test)

#Column names
names(train)

names(test)

#Structure of variables
str(train)

str(test)

#Verify  summary of data
summary(train)

summary(test)
```

```
# Changing the data types of variables
train$fare_amount = as.numeric(as.character(train$fare_amount))
train$passenger_count = round(train$passenger_count)

####Missing Value analysis
# Checking the Missing data
apply(train, 2, function(x) {sum(is.na(x))})

#Creating data frame of missing values present in each variable
value = data.frame(apply(train, 2, function(x){sum(is.na(x))}))
value$Columns = row.names(value)
names(value)[1] = "null_percentage"

# missing value percentage
value$null_percentage = (value$null_percentage/nrow(train)) * 100

# Sorting in Descending order
value = value[order(-value$null_percentage),]
row.names(value) = NULL

# Reordering the columns
value = value[,c(2,1)]

value

#We have seen that null values are very less in our data set i.e. less than 1%.
#So we can delete the columns having missing values
# delete all the rows with missing values,
train = drop_na(train)

# Verifying of missing value upon deletion
sum(is.na(train))

#Splitting Date and time on train data
train$pickup_date = as.Date(as.character(train$pickup_datetime))
train$weekday = as.factor(format(train$pickup_date,"%u"))
train$month = as.factor(format(train$pickup_date,"%m"))
train$year = as.factor(format(train$pickup_date,"%Y"))
pickup_time = strptime(train$pickup_datetime,"%Y-%m-%d %H:%M:%S")
train$hour = as.factor(format(pickup_time,"%H"))

#Splitting Date and time on test data
test$pickup_date = as.Date(as.character(test$pickup_datetime))
test$weekday = as.factor(format(test$pickup_date,"%u"))
test$month = as.factor(format(test$pickup_date,"%m"))
test$year = as.factor(format(test$pickup_date,"%Y"))
pickup_time_test = strptime(test$pickup_datetime,"%Y-%m-%d %H:%M:%S")
test$hour = as.factor(format(pickup_time_test,"%H"))

# Now drop the column pickup_datetime and pickup_date from train data
train = subset(train, select = -c(pickup_datetime))
train = subset(train, select = -c(pickup_date))
```

```
# Now drop the column pickup_datetime and pickup_date from test data
test = subset(test, select = -c(pickup_datetime))
test = subset(test, select = -c(pickup_date))


########### data cleaning  ############
#working on fare amount variable

# fare amount cannot be less than one
# considering fare amount 453 as max and removing all the fare amount greater than 500
nrow(train[which(train$fare_amount < 1 ),])

train = train[-which(train$fare_amount < 1 ),]

nrow(train[which(train$fare_amount >500 ),])
train = train[-which(train$fare_amount >500 ),]

# passenger count cannot be Zero
# even if we consider suv max seat is 6, so removing passenger count greater than 6.
nrow(train[which(train$passenger_count < 1 ),])
train=train[-which(train$passenger_count < 1 ),]

nrow(train[which(train$passenger_count >6 ),])
train=train[-which(train$passenger_count >6 ),]

# Latitudes range from -90 to 90.Longitudes range from -180 to 180.
# Removing which does not satisfy these ranges.
summary(train)
train = train[-which(train$pickup_latitude > 90),]

# Also we will see if there are any values equal to 0.
nrow(train[which(train$pickup_longitude == 0 ),])
nrow(train[which(train$pickup_latitude == 0 ),])
nrow(train[which(train$dropoff_longitude == 0 ),])
nrow(train[which(train$pickup_latitude == 0 ),])

# removing those data points.
train=train[-which(train$pickup_longitude == 0 ),]
train=train[-which(train$dropoff_longitude == 0),]

# checking for missing values.
sum(is.na(train))
sum(is.na(test))

train=na.omit(train) # we have removed the missing values...as they are less
sum(is.na(train))

# Calculate the distance travelled using longitude and latitude
haversine = function(long1, lat1, long2, lat2) {
  rad = pi/180
  a1 = lat1*rad
```

```
  a2 = long1*rad
  b1 = lat2*rad
  b2 = long2*rad
  dlon = b2 - a2
  dlat = b1 - a1
  a = (sin(dlat/2))^2 + cos(a1)*cos(b1)*(sin(dlon/2))^2
  c = 2*atan2(sqrt(a), sqrt(1 - a))
  Radius = 6371
  distance = Radius*c
  return(distance)
}
# Using the haversine formula to calculate distance variable in both train and test data set
train$distance =
haversine(train$pickup_longitude,train$pickup_latitude,train$dropoff_longitude,train$dropoff_l
atitude)

test$distance =
haversine(test$pickup_longitude,test$pickup_latitude,test$dropoff_longitude,test$dropoff_latit
ude)

# We will remove the variables which were used to feature engineer new variables
train = subset(train,select = -
c(pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude))
test = subset(test,select = -
c(pickup_longitude,pickup_latitude,dropoff_longitude,dropoff_latitude))

summary(train)

# creating boxplot of the continous variables to check outlier
ggplot(data = train, aes(x = "", y = distance)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 150))

#Working on distance variable
nrow(train[which(train$distance ==0 ),])
nrow(test[which(test$distance==0 ),])
nrow(train[which(train$distance >30 ),])
nrow(test[which(test$distance >30 ),])

# considering the distance 30 as max and considering rest as outlier.
train=train[-which(train$distance ==0 ),]
train=train[-which(train$distance >30 ),]
test=test[-which(test$distance ==0 ),]

#working on fare amount variable
ggplot(data = train, aes(x = "", y = fare_amount)) +
  geom_boxplot() +
  coord_cartesian(ylim = c(0, 150))

# considering the fare amount 100 as max and considering rest as outlier.
nrow(train[which(train$fare_amount >100 ),])
train = train[-which(train$fare_amount >100 ),]
```

```
########## feature selection##########
#selecting only numeric
numeric_index = sapply(train,is.numeric)
numeric_data = train[,numeric_index]
cnames = colnames(numeric_data)

#Correlation analysis for numeric variables
library(corrgram)
corrgram(train[,numeric_index],upper.panel=panel.pie, main = "Correlation Plot")


########3# feature scaling ##########
library(car)
library(MASS)
qqPlot(train$fare_amount)
truehist(train$fare_amount)
lines(density(train$fare_amount))

d=density(train$fare_amount)
plot(d,main="distribution")
polygon(d,col="purple",border="red")

D=density(train$distance)
plot(D,main="distribution")
polygon(D,col="purple",border="red")

A=density(test$distance)
plot(A,main="distribution")
polygon(A,col="green",border="red")

#make a copy
df_train1 = train
df_test1 = test

###########Normalisation
# log transformation.
train$fare_amount=log1p(train$fare_amount)
train$distance=log1p(train$distance)

# checking back features after transformation.
d=density(train$fare_amount)
plot(d,main="distribution")
polygon(d,col="purple",border="red")

D=density(train$distance)
plot(D,main="distribution")
polygon(D,col="green",border="black")

############### Data Visualization ########################
# Scatter plot b/w passenger and fare on train data
ggplot(data = train, aes_string(x = train$passenger_count, y = train$fare_amount))+
```

```
  geom_point()

# Scatter plot b/w distance and fare on train data
ggplot(data = train, aes_string(x = train$distance, y = train$fare_amount))+
  geom_point()

# Scatter plot b/w hour and fare on train data
ggplot(data = train, aes_string(x = train$hour, y = train$fare_amount))+
  geom_point()

# Scatter plot b/w weekday and fare on train data
ggplot(data = train, aes_string(x = train$weekday, y = train$fare_amount))+
  geom_point()

#make a copy
df_train = train
df_test = test



############################ Model Development ####################

########## Split train data into train and test #################
set.seed(123)
split_index = createDataPartition(train$fare_amount, p = 0.8, list = FALSE)
train_data = train[split_index,]
test_data = train[-split_index,]

#############  Linear regression Model  #################
lm_model = lm(fare_amount ~., data=train_data)

# summary of trained model
summary(lm_model)

# residual plot
plot(lm_model$fitted.values,rstandard(lm_model),main = "Residual plot",
     xlab = "Predicted values of fare amount",
     ylab = "standardized residuals")

# prediction on test data
lm_predictions = predict(lm_model,test_data[,2:7])
# prediction on train data
lm_predictions_train = predict(lm_model,train_data[,2:7])
qplot(x = test_data[,1], y = lm_predictions, data = test_data, color = I("blue"), geom =
"point")

regr.eval(test_data[,1],lm_predictions)
#        mae        mse       rmse       mape
#    0.16237321 0.06233015 0.24966007 0.07057444

regr.eval(train_data[,1],lm_predictions_train)
#       mae        mse       rmse       mape
```

```
#    0.16189595 0.06081755 0.24661215 0.07072041

# compute r^2 on test data
r2_lr = sum((lm_predictions - test_data$fare_amount) ^ 2)
t2_lr = sum((test_data$fare_amount - mean(test_data$fare_amount)) ^ 2)
rsq_lr = 1 - r2_lr/t2_lr
rsq_lr
#     r^2 - 0.788882

# compute r^2 on train data
r2_lr1 = sum((lm_predictions_train - train_data$fare_amount) ^ 2)
t2_lr1 = sum((train_data$fare_amount - mean(train_data$fare_amount)) ^ 2)
rsq_lr1 = 1 - r2_lr1/t2_lr1
rsq_lr1
#     r^2 - 0.7928228

################### Decision Tree Regressor Model ####################
DT_model = rpart(fare_amount ~ ., data=train_data, method = "anova" , minsplit=5)

# summary on train data
summary(DT_model)

#Prediction on test data
prediction_DT = predict(DT_model, test_data[,2:7])

#Prediction on train data
prediction_DT_train = predict(DT_model, train_data[,2:7])

qplot(x = test_data[,1], y = prediction_DT, data=test_data, color = I("blue"), geom =
"point")

regr.eval(test_data[,1], prediction_DT)
#       mae        mse       rmse       mape
#    0.19141056 0.07190210 0.26814567 0.08440643

regr.eval(train_data[,1], prediction_DT_train)
#    mae        mse       rmse       mape
# 0.18934990 0.07080609 0.26609414 0.08347341

# compute r^2 on test data
r2_dt = sum((prediction_DT - test_data$fare_amount) ^ 2)
t2_dt = sum((test_data$fare_amount - mean(test_data$fare_amount)) ^ 2)
rsq_dt = 1 - r2_dt/t2_dt
rsq_dt
#     r^2 - 0.7587964

# compute r^2 on train data
r2_dt1 = sum((prediction_DT_train - train_data$fare_amount) ^ 2)
t2_dt1 = sum((train_data$fare_amount - mean(train_data$fare_amount)) ^ 2)
rsq_dt1 = 1 - r2_dt1/t2_dt1
rsq_dt1
#     r^2 - 0.7564609
```

```
################  Random forest Regressor Model ####################
rf_model = randomForest(fare_amount ~., data=train_data ,  ntree = 100 ,
importance=TRUE)

# summary on trained model
summary(rf_model)

# prediction of test data
rf_prediction = predict(rf_model, test_data[,2:7])

# prediction of train data
rf_prediction_train = predict(rf_model, train_data[,2:7])

qplot(x = test_data[,1], y = rf_prediction, data=test_data, color = I("blue"), geom = "point")

regr.eval(test_data[,1], rf_prediction)
#          mae        mse       rmse       mape
#       0.17062625 0.05869819 0.24227709 0.07632930

regr.eval(train_data[,1], rf_prediction_train)
#     mae        mse       rmse       mape
# 0.08646949 0.01528412 0.12362895 0.03859882

# compute r^2 on test data
r2_rf = sum((rf_prediction - test_data$fare_amount) ^ 2)
t2_rf = sum((test_data$fare_amount - mean(test_data$fare_amount)) ^ 2)
rsq_rf = 1 - r2_rf/t2_rf
rsq_rf
#    r^2 - 0.8011838

# compute r^2 on train data
r2_rf1 = sum((rf_prediction_train - train_data$fare_amount) ^ 2)
t2_rf1 = sum((train_data$fare_amount - mean(train_data$fare_amount)) ^ 2)
rsq_rf1 = 1 - r2_rf1/t2_rf1
rsq_rf1
#    r^2 - 0.9479341


#######################Selection of model########

#Predict for test data with best fit model - Random forest
# we have already clean the test data
# we use whole training Dataset to predict the fare on test dataset
train=df_train1
rf_model1 = randomForest(fare_amount ~., data=train ,  ntree = 100 , importance=TRUE)
pred_RF_test = predict(rf_model1,df_test)
pred_RF_test_R_DF = data.frame(df_test$passenger_count, df_test$distance,"fare_amount"
= pred_RF_test )
write.csv(pred_RF_test_R_DF,"Test_Prediction_R.csv",row.names = FALSE)
```

Thank you