# Project:

# CAB FARE PREDICTION


**SEPTEMBER-16, 2019**

**EDWISOR**

**Submittedby: SANJAY TALLOLLI**

| Sl. No | Table of Contents | Page No |
|---|---|---|
| | Index | 2 |
| | Pre requisite And Steps to Execute Codes | 3 |
| 1 | Introduction | |
| | 1.1 Problem Statement | 5 |
| | 1.2 Data | 6 |
| 2 | Methodology | 7 |
| | 2.1 CRISP-DM Process | 8 |
| | 2.2 Business Understanding | 9 |
| | 2.3 Data Understanding | 9 |
| | 2.4 Data Pre - Processing | 10 |
| | 2.4.1 Data Exploration and Cleaning | 11 |
| | 2.4.2 Missing Value Analysis | 21 |
| | 2.4.3 Outlier Analysis | 28 |
| | 2.4.4 Feature Engineering | 31 |
| | 2.4.5 Frequency Distribution Visualization | 44 |
| | 2.4.6 Feature Selection | 48 |
| | 2.4.7 Feature Scaling | 51 |
| | 2.5 Predictive Modeling | 51 |
| | 2.5.1 Model Selection | 52 |
| | 2.5.2 Multiple Linear Regression | 58 |
| | 2.5.3 Decision Tree | 60 |
| | 2.5.4 Random forest | 61 |
| 3 | Conclusion | 61 |
| | 3.1 MAE, MSE, RMSE, R-squared, MAPE Model Evaluation | 62 |
| | 3.2 Model Evaluation | 63 |
| | 3.3 Model Selection | 64 |
| | 3.4 Hyperparameter Tuning | |
| 4 | 4.1 Brief Insights about the Cab Fare prediction Project | 66 |
| | 4.2 Appendix A – R and Python Codes | 67 |
| 5 | Appendix B – References | 67 |

Pre requisite:

A. To run the R project in the system should have R &R-studio; Below are the versions of both which are used:
   1. R version 3.6.1 (64 bit).
   2. R-studio IDE (Version 1.2.1335).

B. To run the Python (ipynb) file in the system should have Anaconda Navigator to be installed Below are the versions of software used:
   1. Anaconda Navigator (Version -1.9.7)
   2. Jupyter Notebook (Version – 6.0.0)
   3. Python Version – 3.6.5

C. Download the file and save it in your local. (In my case):
   D:/Edwisor2019/Projects/Cab Fare Prediction /train_cab.csv
   D:/Edwisor2019/Projects/Cab Fare Prediction /test.csv
   D:/Edwisor2019/Projects/Cab Fare Prediction/Cab_Fare_Prediction_R.r
   (R-Script File)
   D:/Edwisor2019/Projects/Cab Fare Prediction / Cab_Fare_Prediction_Python.py
   (Python-Script File)
   For Notebook Files:
   Cab_Fare_Prediction_R.ipynb (R – Notebook)
   Cab_Fare_Prediction_Python.ipynb (Python – Notebook)
   Cab Fare Prediction Project Report.docx or .pdf format

Steps to execute the R-code:

i. Run the application by following the below steps using Command Prompt:
   - Go to the path where you have installed R, select the path of the file at top of the window then type “cmd” in that search bar and press enter.

- You will be directed to command prompt with R default location from where you executed the command.
- Next, Type the R-script filename path complete as mentioned above and press enter.
- In R-studio: For R code open R file and run the R-script to get the output in console and plots in Plot window.

Steps to execute the Python code

1) For Python please open Anaconda prompt type Jupyter notebook and load ipynb file and run the code
2) .py file also attached please download the same run in cmd suggested version of python is 3.6 because kNN fancy impute supported only in python 3.6.
3) For Complete project report Open world/ pdf document. Named. "Cab Fare Prediction Project Report. Pdf or . docx"

# 1. Introduction

## 1.1 Problem Statement:

"You are a cab rental start-up company. You have successfully run the pilot project and now want to launch your cab service across the country. You have collected the historical data from your pilot project and now have a requirement to apply analytics for fare prediction. You need to design a system that predicts the fare amount for a cab ride in the city ".

The objective of this Project is to predict the fare of the Cab rental in the city. This Fare prediction takes distance, date/time and other factors in to account from historical data which was gathered from the pilot project for the same. We would be building a model that can successfully predict the fare of rentals on relevant factors.

With technological advancements, there is a rise in the number of cabs one takes for commuting. One of the reasons for growing use of cab uses are how accurately the fair is predicted for a particular distance without charging extra by middlemen. With increasing number of cab rental startup, it is necessary to be able to accurately predict the cab fare for any distance, otherwise it would have inverse output.

A. Prediction < correct fare: results in loss of revenue to the company.
B. Prediction > correct fare: results in passenger being over charged and hence losing faith in the company resulting in loss of customers and creation of bad image in the market.

Hence it is of utmost necessity to be able to predict the cab fare accurately and precisely to ensure that both the company and the passengers are benefited.

## 1.2 Data

Understanding of data is the very first and important step in the process of finding solution of any business problem. Let's have a quick preview of the train and test data. Will discuss in detail about the data understanding in the CRISP-DM process section. Let's understand shape of training and test dataset:

```
Shape of training data is:  (16067, 7)
Shape of test data is:  (9914, 6)
```

Preview of Train data:

| | 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 |

Preview of Test data:

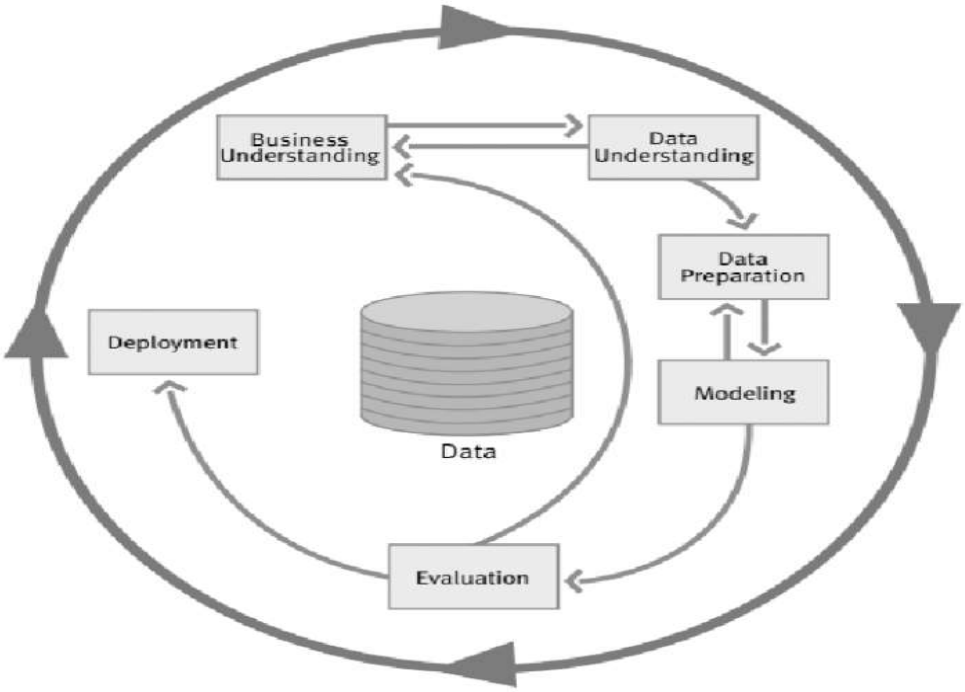
| | pickup_datetime | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count |
|---|---|---|---|---|---|---|
| 0 | 2015-01-27 13:08:24 UTC | -73.973320 | 40.763805 | -73.981430 | 40.743835 | 1 |
| 1 | 2015-01-27 13:08:24 UTC | -73.986862 | 40.719383 | -73.998886 | 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 |

# 2 METHODOLOGY:

## 2.1 CRISP-DM Process:

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining works. The project follows CRISP DM process to develop the model for the given problem. It involves the following major steps:

1. Business understanding.
2. Data understanding.
3. Data preparation.
4. Modeling.
5. Evaluation.
6. Deployment.




## 2.2 Business Understanding:

**Set objectives** - This primary objective is what we want to accomplish from a business perspective which uncover important factors that could influence the outcome of the project, that would be "How to accurately predict the Cab fare for a particular cab ride? Whether it's a fare or distance or location etc., are the influential factors !?"

**Produce project plan** – The plan should specify the steps to be performed during the rest of the project, including the initial selection of tools and techniques i.e, in our case the plan will be involved Understanding the available data and clean the data by converting into a proper shape, i.e., a data free from anomalies like missing values, outliers that can possibly produce errors; scaling the variables; applying various stat or ML models and choosing the best model for a cab fare prediction.

**Business success criteria** – Here we'll lay out the criteria that we'll use to determine whether the project has been successful from the business point of

view ideally be specific and measurable. In our case the success would be related to able to accurately predict the cab fare for a particular cab ride input, with maximum accuracy including possible feature variables. On being able to predict it accurately.

### 2.3 Data Understanding:

We must predict cab fare amount for our test data and the dataset given has dependent & independent variables, since our target variable is a continuous variable therefore this problem comes under supervised machine learning Regression problem. Data was given in two sets, train and test set, Train set was used to train the models whereas test set was used to test the models. There are total 16067 observations, 7 variables in train_cab data and 9914 observations, 6 variables in test data. Missing Values – Yes, Outliers – Yes

**Data Dictionary:** - The details of data attributes in the dataset are as follows let's understand each attribute in detail.

**Pickup_datetime:** – Timestamp value indicating when the cab ride started. This variable explains about cab pickup date and time for a ride like which hour, day passenger booked the cab.

**Pickup_longitude**: – Float for longitude coordinate of where the cab ride started.

**Pickup_latitude**: – Float for latitude coordinate of where the cab ride started. Pickup longitude and latitude are variables represents the (Co-ordinates) where the cab ride was 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. Drop off longitude and latitude are variables represents the location where the cab ride was ended.

**Passenger_count**: – An integer indicating the number of passengers in the cab. No. of passenger travelled in the cab.

**Preview of train_cab dataset:**

| | 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 |

**Preview of test_cab dataset:**

| | 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 |

Here our response / target variable is (fare_amount) and predictor/independent variables are (pickup_datetime), (pickup_logitude), (pickup_latitude), (dropoff_longitude), (dropoff_lattitude), (passenger_count).

```
Shape of training data is:  (16067, 7)
Shape of test data is:  (9914, 6)
```

## 2.4 Data Pre - Processing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data preprocessing is defining the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model. As we already know the quality of our inputs decide the quality of our output. So, once we

have got our business hypothesis ready, it makes sense to spend lot of time and efforts here. Approximately, data exploration, cleaning and preparation can take up to 60-70% of our total project time. This process is often called as Exploratory Data Analysis

Listed below are data pre-processing techniques used for this Project

1) Data Exploration and Cleaning
2) Missing Value Analysis
3) Outlier analysis
4) Analyzing Distribution of each variable with respect to target variable using Visualization
5) Feature Selection
6) Feature Scaling

Let's discuss /explain this technique in detail.

### 2.4.1 Data Exploration and Cleaning :

Data Pre-processing is the very first step which comes with any data science project is data exploration and cleaning, according to our car price prediction project below are the points which are identified in this stage:

- Segregate the combined variables and type /Class of variable conversion.
- . As we know that some negative values in fare amount, so we must remove or impute those values.
- . Considering Passenger count would be max 6 passengers plus 1 driver if it is a XUV-500 vehicle considered for cab service. We must remove or impute values having passengers counts more than 6 and less than 1.
- There are some outlier figures in our train dataset need to identify those and make it normal.

- Latitudes range from -90 to 90. Longitudes range from -180 to 180. We need to remove the observations if any latitude and longitude lies beyond this ranges.
- Also noticed that some anomalies in data which will be addressed in later part of the analysis.

### 2.4.2 Missing Value Analysis

In real world missing value is the common occurrence of incomplete observations in an asset of data. Missing values can arise due to many reasons, error in uploading data, unable to measure an observation etc. Due to presence of missing values in the form of 0, NA or NAN. These will affect the accuracy of model which we are building. Hence it is necessary to check for any missing values in the given data.

Let's check of the dataset for missing values and identify what type are they? Listed below are some of the missing values in different forms:

- Values containing '0'- These will first need to be changed to NA and Nan in R and python respectively.
- Blank spaces that are taken as NA and NaN in R and Python respectively.
- In our dataset pickup datatime variable has one NA value which is not in correct format of date time hence removed.

| | fare_amount | pickup_datetime | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count |
|---|---|---|---|---|---|---|---|
| **1328** | 11.5 | NA | -73.98195 | 40.7282 | -73.98858 | 40.7592 | 1 |

Depending on the percentage of missing values we can decide if we need to keep a variable or drop it based on the following conditions:

- Missing value percentage < 30% - We can include the variable.

- Missing value percentage > 30 % - We need to remove those variable/s because even if we impute values, they are not the actual values, the variable will not contain actual values and hence will not contribute effectively to our model.

For the given train dataset, we can see that we have only two variables fare_amount and passenger count with missing values and the percentage of missing value is less as per the general practice mentioned above, so we are going to include those variables then in later part of the analysis will impute missing values.
Below picture shows no. of missing Values and Percentage in train & test dataset.

The missing value percentage in training data :

[20]:

| | Variables | Missing_values | Missing_Value_Percentage |
|---|---|---|---|
| 0 | passenger_count | 55 | 0.342338 |
| 1 | fare_amount | 25 | 0.155608 |
| 2 | pickup_longitude | 0 | 0.000000 |
| 3 | pickup_latitude | 0 | 0.000000 |
| 4 | dropoff_longitude | 0 | 0.000000 |
| 5 | dropoff_latitude | 0 | 0.000000 |
| 6 | Year | 0 | 0.000000 |
| 7 | Month | 0 | 0.000000 |
| 8 | Date | 0 | 0.000000 |
| 9 | Dayofweek | 0 | 0.000000 |
| 10 | Hour | 0 | 0.000000 |
| 11 | Minutes | 0 | 0.000000 |
| 12 | Seconds | 0 | 0.000000 |

[21]:

The missing value percentage in test datatest :

| | Variables | Missing_values | Missing_Value_Percentage |
|---|---|---|---|
| 0 | pickup_longitude | 0 | 0.0 |
| 1 | pickup_latitude | 0 | 0.0 |
| 2 | dropoff_longitude | 0 | 0.0 |
| 3 | dropoff_latitude | 0 | 0.0 |
| 4 | passenger_count | 0 | 0.0 |
| 5 | Year | 0 | 0.0 |
| 6 | Month | 0 | 0.0 |
| 7 | Date | 0 | 0.0 |
| 8 | Dayofweek | 0 | 0.0 |
| 9 | Hour | 0 | 0.0 |
| 10 | Minutes | 0 | 0.0 |
| 11 | Seconds | 0 | 0.0 |

Question is that how are we dealing these missing values with?

**In Python:** specifically, Pandas, NumPy and Scikit-Learn, we mark missing values as NaN. And With the help of kNN algorithm we can impute missing values.

[23]:

```
#Apply KNN imputation algorithm
train_cab = pd.DataFrame(KNN(k = 3).fit_transform(train_cab), columns = train_cab.columns)
```

```
Imputing row 1/16066 with 0 missing, elapsed time: 45.415
Imputing row 101/16066 with 0 missing, elapsed time: 49.117
Imputing row 201/16066 with 0 missing, elapsed time: 50.830
Imputing row 301/16066 with 0 missing, elapsed time: 51.125
Imputing row 401/16066 with 0 missing, elapsed time: 51.707
Imputing row 501/16066 with 0 missing, elapsed time: 52.221
Imputing row 601/16066 with 0 missing, elapsed time: 54.141
Imputing row 701/16066 with 0 missing, elapsed time: 54.619
Imputing row 801/16066 with 0 missing, elapsed time: 54.686
Imputing row 901/16066 with 0 missing, elapsed time: 54.690
Imputing row 1001/16066 with 0 missing, elapsed time: 54.690
Imputing row 1101/16066 with 0 missing, elapsed time: 54.690
Imputing row 1201/16066 with 0 missing, elapsed time: 54.691
Imputing row 1301/16066 with 0 missing, elapsed time: 54.691
Imputing row 1401/16066 with 0 missing, elapsed time: 54.692
Imputing row 1501/16066 with 0 missing, elapsed time: 54.692
Imputing row 1601/16066 with 0 missing, elapsed time: 54.693
Imputing row 1701/16066 with 0 missing, elapsed time: 54.693
Imputing row 1801/16066 with 0 missing, elapsed time: 54.694
Imputing row 1901/16066 with 0 missing, elapsed time: 54.695
Imputing row 2001/16066 with 0 missing, elapsed time: 54.696
```

[25]:

The missing value percentage in training data :

| | Variables | Missing_values | Missing_Value_Percentage |
|---|---|---|---|
| 0 | fare_amount | 0 | 0.0 |
| 1 | pickup_longitude | 0 | 0.0 |
| 2 | pickup_latitude | 0 | 0.0 |
| 3 | dropoff_longitude | 0 | 0.0 |
| 4 | dropoff_latitude | 0 | 0.0 |
| 5 | passenger_count | 0 | 0.0 |
| 6 | Year | 0 | 0.0 |
| 7 | Month | 0 | 0.0 |
| 8 | Date | 0 | 0.0 |
| 9 | Dayofweek | 0 | 0.0 |
| 10 | Hour | 0 | 0.0 |
| 11 | Minutes | 0 | 0.0 |
| 12 | Seconds | 0 | 0.0 |

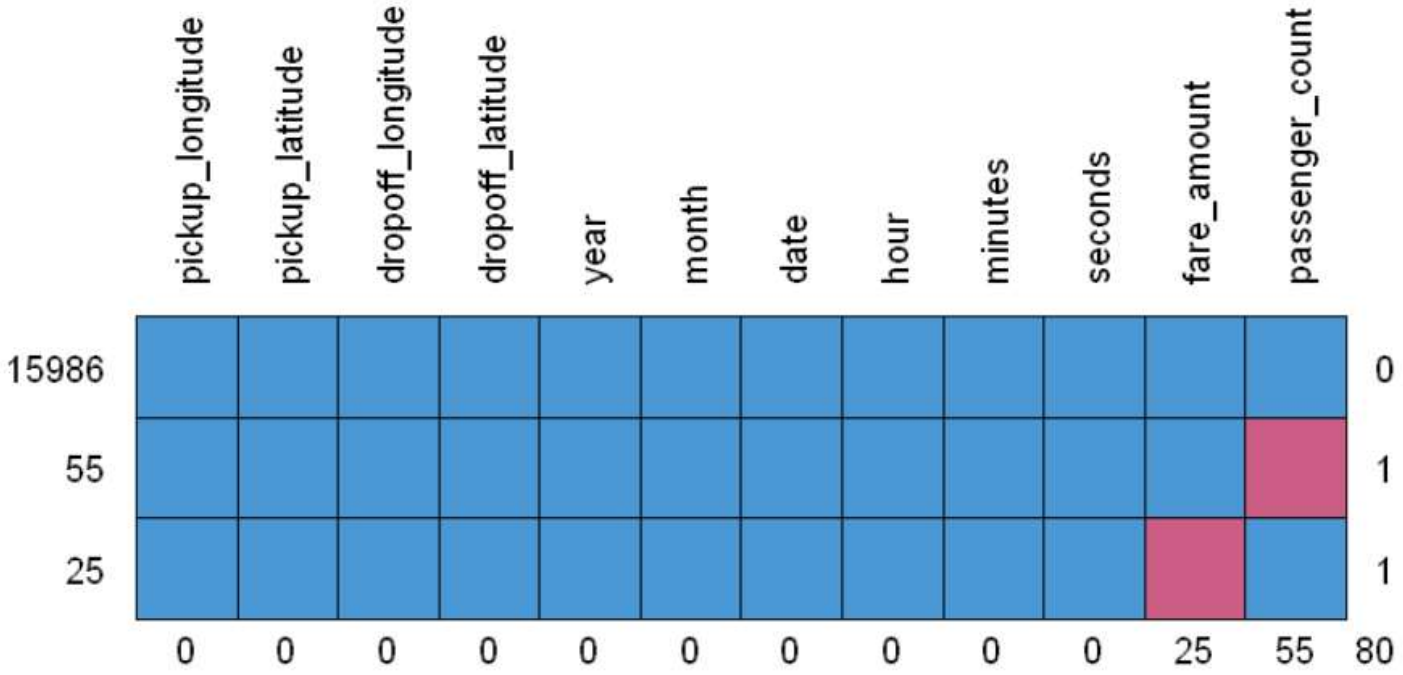
**In R :** we have several packages to deal with missing values listed below are frequently used packages, they are,

A. MICE ( Multivariate Imputation via Chained Equations)
B. Amelia ( Named after Amelia Earhart )
C. missForest ( Random Forest : non-parametric imputation method )
D. Hmisc ( multiple purpose package )
E. Mi ( (Multiple imputation with diagnostics)

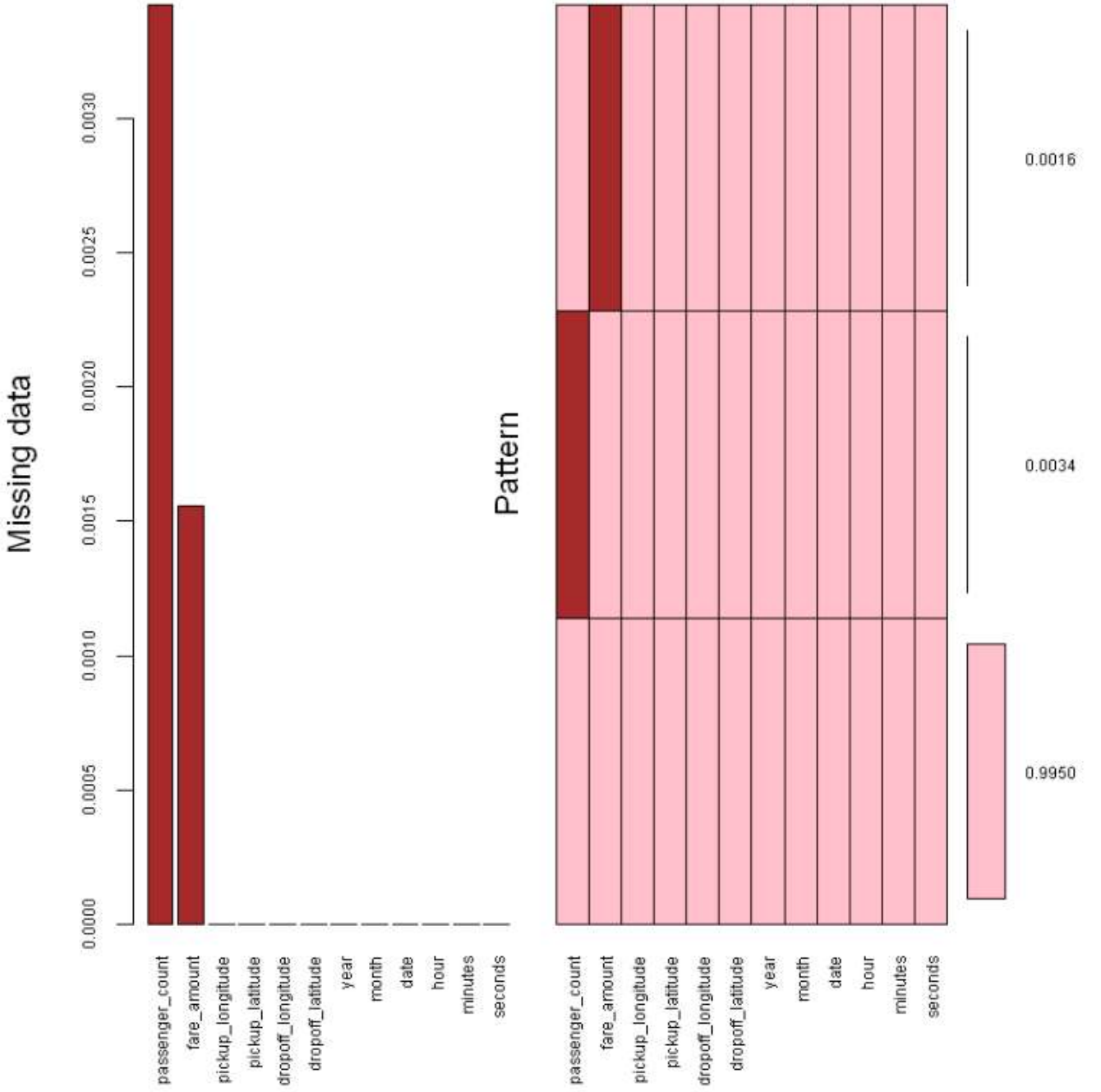
Tested top 3 packages on our train dataset. Results are as mentioned below:

### A. MICE ( Multivariate Imputation via Chained Equations )

| | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | year | month | date | hour | minutes | seconds | fare_amount | passenger_count | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 15986 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 55 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 |
| 25 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 25 | 55 | 80 |




```
Variables sorted by number of missings:
         Variable       Count
  passenger_count 0.003423379
      fare_amount 0.001556081
 pickup_longitude 0.000000000
  pickup_latitude 0.000000000
dropoff_longitude 0.000000000
 dropoff_latitude 0.000000000
             year 0.000000000
            month 0.000000000
             date 0.000000000
             hour 0.000000000
          minutes 0.000000000
          seconds 0.000000000
```


Missing data
0.0000
0.0005
0.0010
0.0015
0.0020
0.0025
0.0030
passenger_count
fare_amount
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
year
month
date
hour
minutes
seconds
Pattern
passenger_count
fare_amount
pickup_longitude
pickup_latitude
dropoff_longitude
dropoff_latitude
year
month
date
hour
minutes
seconds
0.0016
0.0034
0.9950

```
Class: mids
Number of multiple imputations:  5
Imputation methods:
 pickup_longitude   pickup_latitude dropoff_longitude  dropoff_latitude
               ""                 ""                ""                 ""
  passenger_count               year              month               date
            "pmm"                 ""                ""                 ""
        dayofweek               hour           minutes            seconds
               ""                 ""                ""                 ""
      fare_amount
            "pmm"
PredictorMatrix:
                  pickup_longitude pickup_latitude dropoff_longitude
pickup_longitude                 0                1                 1
pickup_latitude                  1                0                 1
dropoff_longitude                1                1                 0
dropoff_latitude                 1                1                 1
passenger_count                  1                1                 1
year                             1                1                 1
                  dropoff_latitude passenger_count year month date dayofweek
pickup_longitude                 1                1    1     1    1         1
pickup_latitude                  1                1    1     1    1         1
dropoff_longitude                1                1    1     1    1         1
dropoff_latitude                 0                1    1     1    1         1
passenger_count                  1                0    1     1    1         1
year                             1                1    0     1    1         1
                  hour minutes seconds fare_amount
pickup_longitude     1       1       1           1
pickup_latitude      1       1       1           1
dropoff_longitude    1       1       1           1
dropoff_latitude     1       1       1           1
passenger_count      1       1       1           1
year                 1       1       1           1
```

In [64]:
```
#check imputed values
# imputed_Data$imp$passenger_count
 imputed_Data$imp$fare_amount
```

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| **9** | 3.30 | 24.0 | 4.20 | 25.30 | 7.70 |
| **27** | 6.50 | 9.0 | 19.50 | 38.50 | 4.50 |
| **70** | 8.00 | 12.5 | 14.50 | 12.50 | 8.50 |
| **127** | 10.50 | 8.5 | 4.90 | 7.50 | 33.30 |
| **169** | 9.00 | 22.5 | 4.50 | 36.90 | 10.00 |
| **241** | 6.90 | 13.0 | 3.70 | 5.30 | 19.70 |
| **306** | 6.00 | 5.5 | 57.33 | 37.33 | 7.00 |
| **351** | 4.90 | 14.5 | 35.50 | 4.10 | 7.30 |
| **414** | 4.10 | 6.1 | 9.50 | 9.70 | 4.90 |
| **456** | 7.30 | 4.5 | 16.00 | 55.50 | 6.00 |
| **499** | 4.90 | 6.9 | 10.00 | 12.50 | 7.30 |
| **668** | 6.00 | 5.0 | 23.70 | 5.50 | 5.00 |
| **704** | 8.90 | 15.0 | 8.10 | 5.30 | 10.90 |
| **747** | 7.50 | 11.0 | 10.00 | 7.50 | 5.50 |
| **837** | 6.90 | 14.9 | 23.00 | 12.90 | 11.00 |
| **841** | 6.50 | 7.5 | 57.33 | 17.00 | 5.30 |
| **914** | 5.50 | 6.9 | 6.50 | 20.10 | 5.70 |
| **1124** | 6.90 | 20.1 | 49.57 | 7.30 | 10.00 |
| **1574** | 37.07 | 3.7 | 30.33 | 6.10 | 30.83 |
| **1628** | 21.00 | 12.0 | 6.50 | 7.30 | 19.00 |
| **1712** | 5.70 | 13.5 | 5.00 | 5.30 | 9.70 |
| **2412** | 8.50 | 8.0 | 6.00 | 5.00 | 4.00 |
| **2458** | 5.70 | 21.5 | 6.90 | 6.50 | 16.00 |
| **8178** | 6.50 | 10.5 | 12.00 | 11.70 | 18.50 |
| **8226** | 29.00 | 16.0 | 11.50 | 14.50 | 3.70 |

## B. Amelia ( Named after Amelia Earhart )

```
[89]: #specify columns and run amelia

amelia_fit <- amelia(Train_Cab, m=5, parallel = "multicore")
#noms = c("year","month","dayofweek","date","hour","minutes","seconds")
```

```
-- Imputation 1 --

  1  2

-- Imputation 2 --

  1  2

-- Imputation 3 --

  1  2

-- Imputation 4 --

  1  2

-- Imputation 5 --

  1  2
```

```
In [67]: # Access imputed outputs
#amelia_fit$imputations[[1]]
#amelia_fit$imputations[[2]]
#amelia_fit$imputations[[3]]
#amelia_fit$imputations[[4]]
head(amelia_fit$imputations[[5]])
```

| pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | year | month | date | dayofweek | hour | minutes | fare_amount |
|---|---|---|---|---|---|---|---|---|---|---|---|
| -73.84431 | 40.72132 | -73.84161 | 40.71228 | 1 | 2009 | 6 | 15 | 2 | 17 | 26 | 4.5 |
| -74.01605 | 40.71130 | -73.97927 | 40.78200 | 1 | 2010 | 1 | 5 | 3 | 16 | 52 | 16.9 |
| -73.98274 | 40.76127 | -73.99124 | 40.75056 | 2 | 2011 | 8 | 18 | 5 | 0 | 35 | 5.7 |
| -73.98713 | 40.73314 | -73.99157 | 40.75809 | 1 | 2012 | 4 | 21 | 7 | 4 | 30 | 7.7 |
| -73.96810 | 40.76801 | -73.95665 | 40.78376 | 1 | 2010 | 3 | 9 | 3 | 7 | 51 | 5.3 |
| -74.00096 | 40.73163 | -73.97289 | 40.75823 | 1 | 2011 | 1 | 6 | 5 | 9 | 50 | 12.1 |

```
In [107]: # to check values variable wise
amelia_fit$imputations[[5]]$fare_amount
```

```
4.5 16.9 5.7 7.7 5.3 12.1 7.5 16.5 -352.136058807944 8.9 5.3 5.5 4.1 7 7.7 5 12.5 5.3 5.3 4 10.5 11.5 4.5 4.9 6.1
7.3 56.5607932453155 4.5 9.3 4.5 5.5 22.54 31.9 5.7 18.1 58 4.5 5.3 9 9.8 8.1 10.9 17.5 12.1 5.3 6.9 4.5 3.3 3.3 9
9.7 5.5 8 7.5 25.5 15.3 34.67 9.7 13.7 6.5 5 4.5 4.9 7 7.3 7.5 8.5 16.5 11.5 76.7632593552747 6.9 3.7 6.9 10.1 16.5
15.07 7.7 24.9 7 8.5 5.7 5 8.5 17 8 12 12.5 7.5 6.9 5 10.5 7.7 11 5.7 9 13.5 7.5 9 14 5.7 10 5.7 9.7 14.1 7.7
52 11.3 8.5 12.9 9.5 9 10 35 9 17 6.1 14.5 3.7 32.5 5.3 11.5 4 10.5 17.7 8 5 -620.989610716389 8 7.3 6.5 8 16.5
49.57 10.5 22.5 11.5 12.5 5.5 6.5 16.5 9 8.9 10.9 8.9 8.5 17 12.5 52.5 16 8 17.3 5.7 5.5 16.5 12.9 10.1 9.5 4.5
6.5 21.5 8 4.9 5.7 15.3 14.9 7 35.5 8.5 282.810802562725 4.1 11.7 23 5.7 5.3 10.5 4.9 6.5 11.3 11.5 5.3 5.7 8.1 6.1
9.3 4.5 5.7 16.5 17 24.5 6.1 6 6.5 3.7 4.5 27 16 16.1 15.7 26.9 5.5 6 7.7 12.9 11.3 42.5 5.7 5.3 10.5 7.3 4.5
12.5 16.5 4.9 4.1 15.5 11 4.9 11.3 7.3 14.9 30.9 6.1 3.3 4.1 12 9 2.5 8.9 10.5 5.7 7 12 49.57 8.5 4.5 7.3 5.3 6.9
9.3 6.7 298.124788902297 9 13 19 5.5 8.1 9.3 18 12 7.3 4.9 16.1 12.9 4 15.3 11.7 25.3 8.9 6 6 6.5 7.5 16.5 4.9
9.7 7.7 19 16.9 10.5 8.9 7.5 5.5 10.9 8.1 5.7 18.1 10.9 8.9 13.5 5 5.3 7.7 9 16.1 4.9 5.5 57.33 8.5 9.5 7.5 6.9
7.5 8.1 6.1 5.7 14.1 5.7 7 12 11.5 12.1 26.67 4.5 10.1 7.7 -587.46491168804 19 14 5 43.5 8 8.7 3.7 20.5 34 52.83
6.1 8.5 20 10.5 26.5 8 10.6 4.5 9.7 20.5 9.3 18.5 5 34 23.5 4.5 8 26.5 5.5 6.1 4.9 5.3 8.1 20.33 4.5 7 7.7 7.3
33.07 36.5 5 23.5 6.1 11 -811.658482367925 11.3 8.1 8.9 4.5 11 8.5 8.5 8.1 6.9 6 5.3 6.1 9.5 11.5 17 5.3 10 15.5
6.9 18.1 15.5 10.1 21.5 8.5 7.3 2.5 8.5 6.1 14.1 6 8.5 8 5.3 10.5 5.5 8.1 13 8.5 6.1 12.9 10 8.5 4.9 6 5 12
13.5 33.83 25.7 13.7 10.5 5.7 9 22.5 7.5 14.1 9.3 4 5.5 9.3 7.3 4.5 -249.047517039594 5.7 8 20.5 9.3 8.5 8.1 5 5.3
5.3 6.9 8.5 6.5 29 21 14.5 32.9 6.1 13 15 4.9 8.9 8.1 6 57.33 8 8.5 8.9 10.5 10.1 49.8 5 6 13.3 7 10.5 56.8
```

## C. missForest ( Random Forest : non-parametric imputation method )

In [52]:
```
#impute missing values, using all parameters as default values ( Note: it will take 12 min to execute the code)
Train_Cab_missforest.imp <- missForest(Train_Cab_remdcat)
```

```
  missForest iteration 1 in progress...done!
  missForest iteration 2 in progress...done!
  missForest iteration 3 in progress...done!
  missForest iteration 4 in progress...done!
```

In [53]:
```
#check imputed values
head(Train_Cab_missforest.imp$ximp)
```

| pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | year | month | date | dayofweek | hour | minutes | seconds | fare_amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -73.84431 | 40.72132 | -73.84161 | 40.71228 | 1 | 2009 | 6 | 15 | 2 | 17 | 26 | 21 | 4.5 |
| -74.01605 | 40.71130 | -73.97927 | 40.78200 | 1 | 2010 | 1 | 5 | 3 | 16 | 52 | 16 | 16.9 |
| -73.98274 | 40.76127 | -73.99124 | 40.75056 | 2 | 2011 | 8 | 18 | 5 | 0 | 35 | 0 | 5.7 |
| -73.98713 | 40.73314 | -73.99157 | 40.75809 | 1 | 2012 | 4 | 21 | 7 | 4 | 30 | 42 | 7.7 |
| -73.96810 | 40.76801 | -73.95665 | 40.78376 | 1 | 2010 | 3 | 9 | 3 | 7 | 51 | 0 | 5.3 |
| -74.00096 | 40.73163 | -73.97289 | 40.75823 | 1 | 2011 | 1 | 6 | 5 | 9 | 50 | 45 | 12.1 |

As our train dataset has ordinal integers and numeric variables missForest imputation method is selected for missing value imputation.

| | Missing_values | Missing_values_Percentage |
|---|---|---|
| pickup_longitude | 0 | 0 |
| pickup_latitude | 0 | 0 |
| dropoff_longitude | 0 | 0 |
| dropoff_latitude | 0 | 0 |
| passenger_count | 0 | 0 |
| year | 0 | 0 |
| month | 0 | 0 |
| date | 0 | 0 |
| dayofweek | 0 | 0 |
| hour | 0 | 0 |
| minutes | 0 | 0 |
| seconds | 0 | 0 |
| fare_amount | 0 | 0 |

### 2.4.3 Outlier Analysis

An Outlier is any inconsistent or abnormal observation in a variable of dataset that deviates from the rest of the observations. These inconsistent observations can be due to manual error, poor quality/ malfunctioning equipment's, Experimental error or correct but exceptional data based on business use case. It can cause an error in predicting the target variable/s. Hence, we need to check for the outliers and either remove the observations containing them or replace them with NA, then impute or set upper limits/lower limits or mean/median values imputation.

For Outlier analysis, Boxplot to visualize and summary descriptive statistics to check range of each numeric variable and sorted the variables to detect some strange values like zeros and negative values.

From below mentioned image for R & python descriptive summary statistics shows some anomalies in data which are nothing, but outliers let's discuss about each variable

**In R:**

```
 pickup_longitude pickup_latitude  dropoff_longitude dropoff_latitud
 Min.   :-74.44   Min.   :-74.01   Min.   :-74.43    Min.   :-74.01
 1st Qu.:-73.99   1st Qu.: 40.73   1st Qu.:-73.99    1st Qu.: 40.73
 Median :-73.98   Median : 40.75   Median :-73.98    Median : 40.75
 Mean   :-72.46   Mean   : 39.91   Mean   :-72.46    Mean   : 39.90
 3rd Qu.:-73.97   3rd Qu.: 40.77   3rd Qu.:-73.96    3rd Qu.: 40.77
 Max.   : 40.77   Max.   :401.08   Max.   : 40.80    Max.   : 41.37
 passenger_count         year          month             date
 Min.   :   0.000   Min.   :2009   Min.   : 1.000   Min.   : 1.00
 1st Qu.:   1.000   1st Qu.:2010   1st Qu.: 3.000   1st Qu.: 8.00
 Median :   1.000   Median :2012   Median : 6.000   Median :16.00
 Mean   :   2.623   Mean   :2012   Mean   : 6.261   Mean   :15.67
 3rd Qu.:   2.000   3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00
 Max.   :5345.000   Max.   :2015   Max.   :12.000   Max.   :31.00
   dayofweek         hour         minutes       fare_amount
 Min.   :1.00   Min.   : 0.0   Min.   : 0.00   Min.   :   -3.00
 1st Qu.:2.00   1st Qu.: 9.0   1st Qu.:15.00   1st Qu.:    6.00
 Median :4.00   Median :14.0   Median :30.00   Median :    8.50
 Mean   :4.09   Mean   :13.5   Mean   :29.64   Mean   :   15.01
 3rd Qu.:6.00   3rd Qu.:19.0   3rd Qu.:45.00   3rd Qu.:   12.50
 Max.   :7.00   Max.   :23.0   Max.   :59.00   Max.   :54343.00
```

```
    seconds
 Min.   : 0.00
 1st Qu.: 0.00
 Median : 5.00
 Mean   :16.16
 3rd Qu.:32.00
 Max.   :59.00
```

## In Python:

| | count | mean | std | min | 25% | 50% | 75% | m |
|---|---|---|---|---|---|---|---|---|
| fare_amount | 16066.0 | 15.010283 | 430.139334 | -3.000000 | 6.000000 | 8.500000 | 12.500000 | 54343.000000 |
| pickup_longitude | 16066.0 | -72.462693 | 10.578707 | -74.438233 | -73.992156 | -73.981697 | -73.966837 | 40.766125 |
| pickup_latitude | 16066.0 | 39.914675 | 6.826797 | -74.006893 | 40.734935 | 40.752605 | 40.767381 | 401.083332 |
| dropoff_longitude | 16066.0 | -72.462233 | 10.575384 | -74.429332 | -73.991182 | -73.980170 | -73.963642 | 40.802437 |
| dropoff_latitude | 16066.0 | 39.897852 | 6.187276 | -74.006377 | 40.734647 | 40.753566 | 40.768015 | 41.366138 |
| passenger_count | 16066.0 | 2.621614 | 60.741821 | 0.000000 | 1.000000 | 1.000000 | 2.000000 | 5345.000000 |
| Year | 16066.0 | 2011.730860 | 1.864275 | 2009.000000 | 2010.000000 | 2012.000000 | 2013.000000 | 2015.000000 |
| Month | 16066.0 | 6.260612 | 3.447727 | 1.000000 | 3.000000 | 6.000000 | 9.000000 | 12.000000 |
| Date | 16066.0 | 15.669862 | 8.683210 | 1.000000 | 8.000000 | 16.000000 | 23.000000 | 31.000000 |
| Dayofweek | 16066.0 | 3.032615 | 1.968929 | 0.000000 | 1.000000 | 3.000000 | 5.000000 | 6.000000 |
| Hour | 16066.0 | 13.497821 | 6.519985 | 0.000000 | 9.000000 | 14.000000 | 19.000000 | 23.000000 |
| Minutes | 16066.0 | 29.639549 | 17.295289 | 0.000000 | 15.000000 | 30.000000 | 45.000000 | 59.000000 |
| Seconds | 16066.0 | 16.162330 | 19.489786 | 0.000000 | 0.000000 | 5.000000 | 32.000000 | 59.000000 |

- The valid range of latitude in degrees is -90 and +90 for the southern and northern hemisphere respectively. Longitude is in the range -180 and +180 specifying co-ordinates west and east of the Prime Meridian, respectively. For reference, the Equator has a latitude of 0°, the North pole has a latitude of 90° north (written 90° N or +90°), and the South pole has a latitude of -90°. The Prime Meridian has a longitude of 0° that goes through Greenwich, England. The International Date Line (IDL) roughly follows the 180° longitude. A longitude with a positive value falls in the eastern hemisphere and negative value falls in the western hemisphere. From this knowledge we come to know that longitudes and latitudes in our data are out of range.

- passenger_count maximum value is 5345 which will never possible practically maximum number of passengers can be 6

- In fare_amount maximum fare is 54343, In actual case. which is not possible.

## **In R:** Share of dataset

```
[77]: print(dim(Train_Cab))
      print(dim(Test_Cab))
```

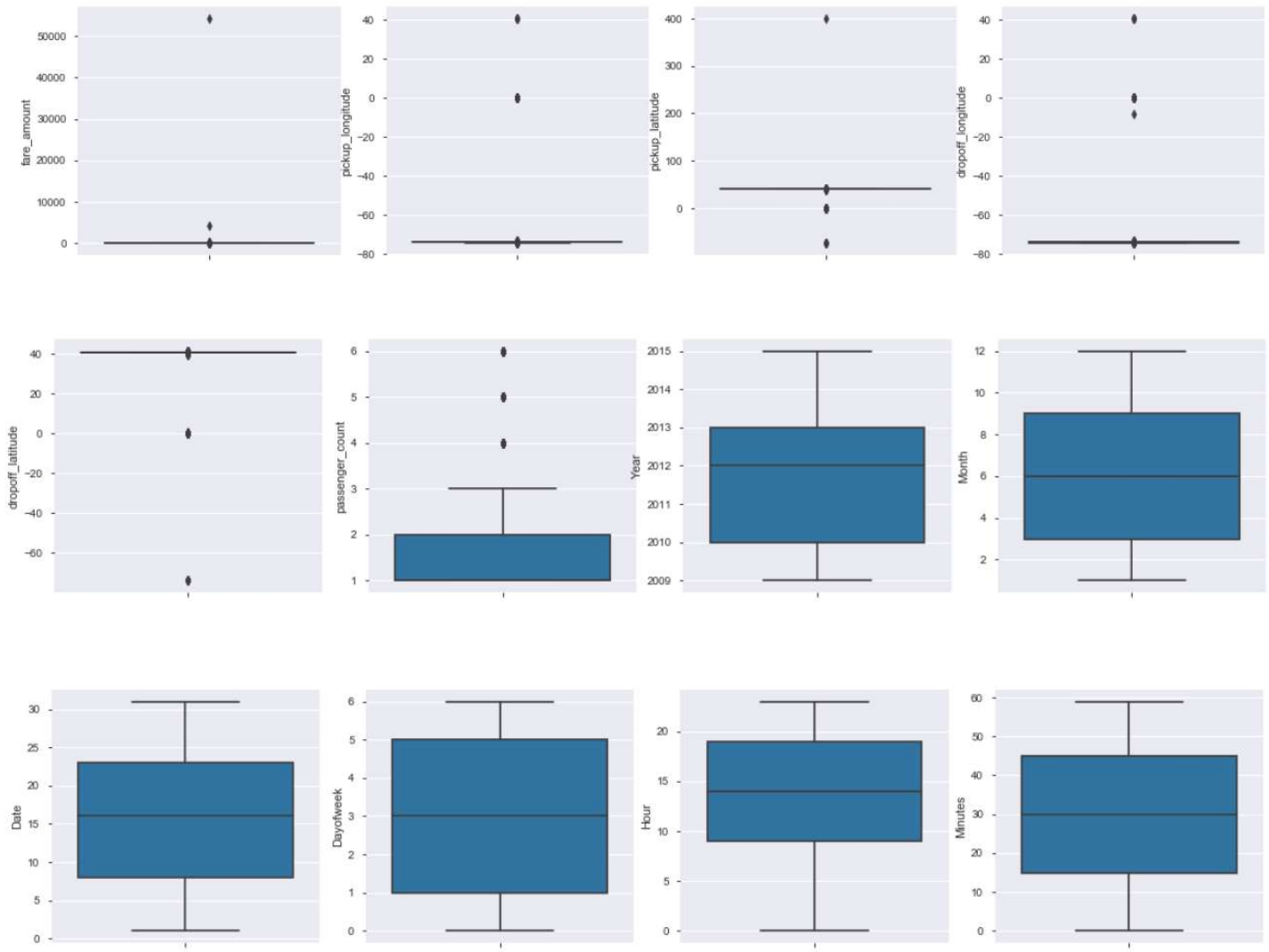
```
[1] 15980    13
[1] 9914   12
```

## **In Python**: Shape of dataset:

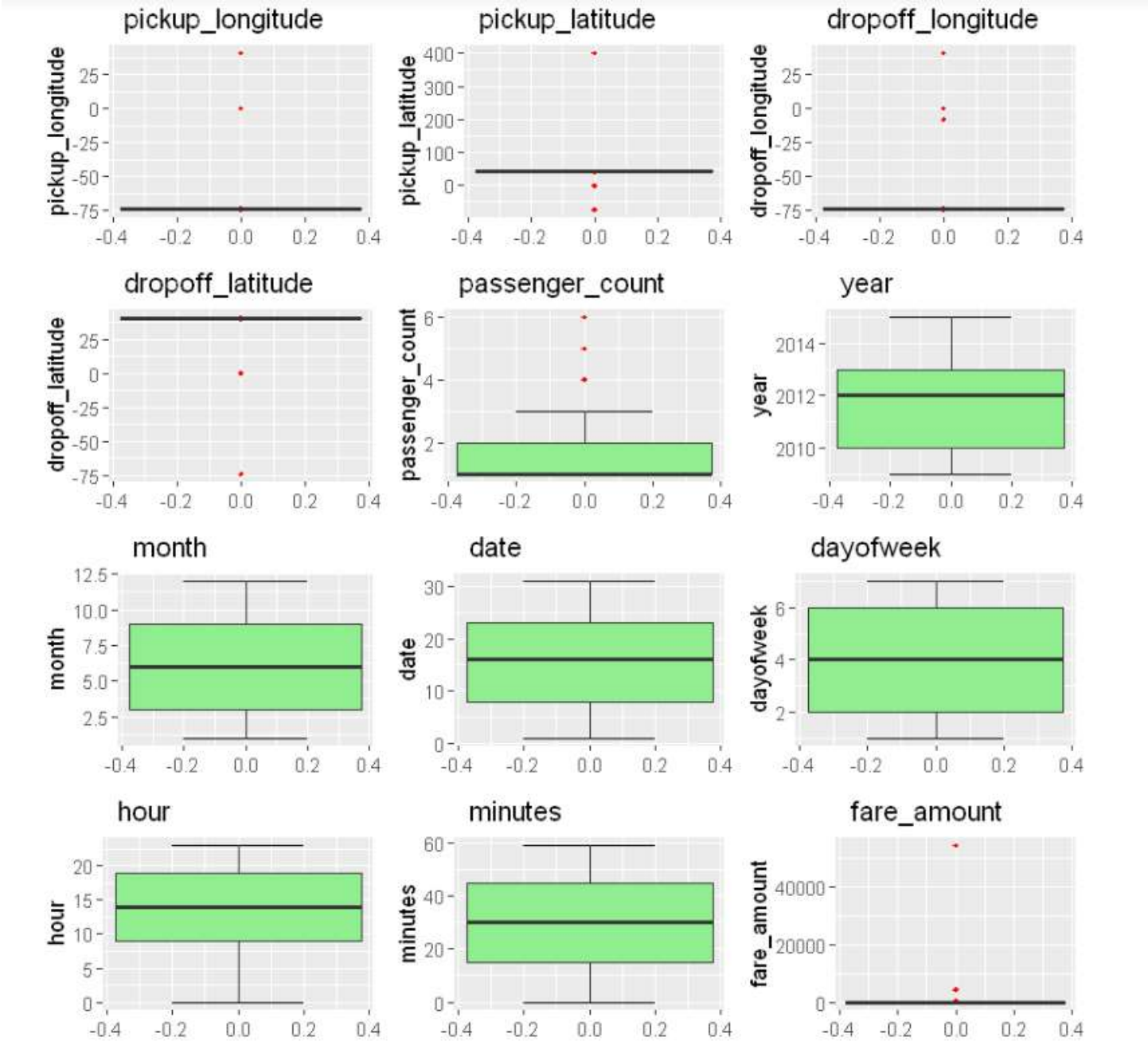
```
In [41]: print("Shape of training data initial exploration is: ",train_cab.shape)
         print("Shape of test data initial exploration is: ",test_cab.shape)
```

```
Shape of training data initial exploration is:  (15976, 13)
Shape of test data initial exploration is:  (9914, 12)
```

## **In python:** Box plots for both numeric and categorical variables:




## **In R:** Box plots for both numeric and categorical variables:




For our cab price prediction project opted for capping method in which we are going to impute outlier with upper fence and lower fence value reason behind to opt this method is we don't want to delete the observations with outliers as data collection is also a crucial step in data analytics for which client has to spend more money if don't have any past data specially for startup companies .by keeping this point into consideration we tried to retain the data wherever possible in the preprocessing.

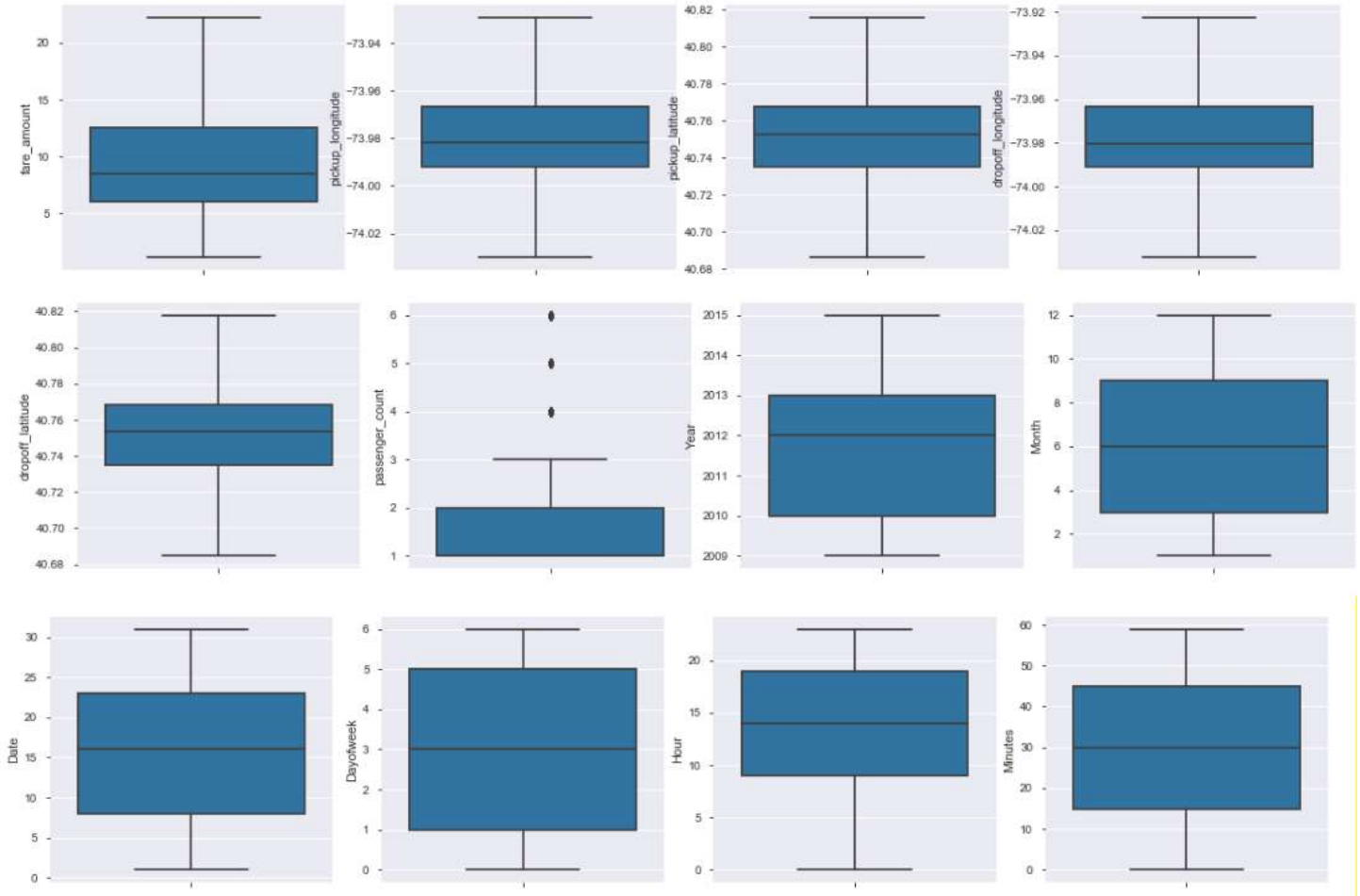
**In R & Python:** below observation was found out of range hence removed.

[51]:

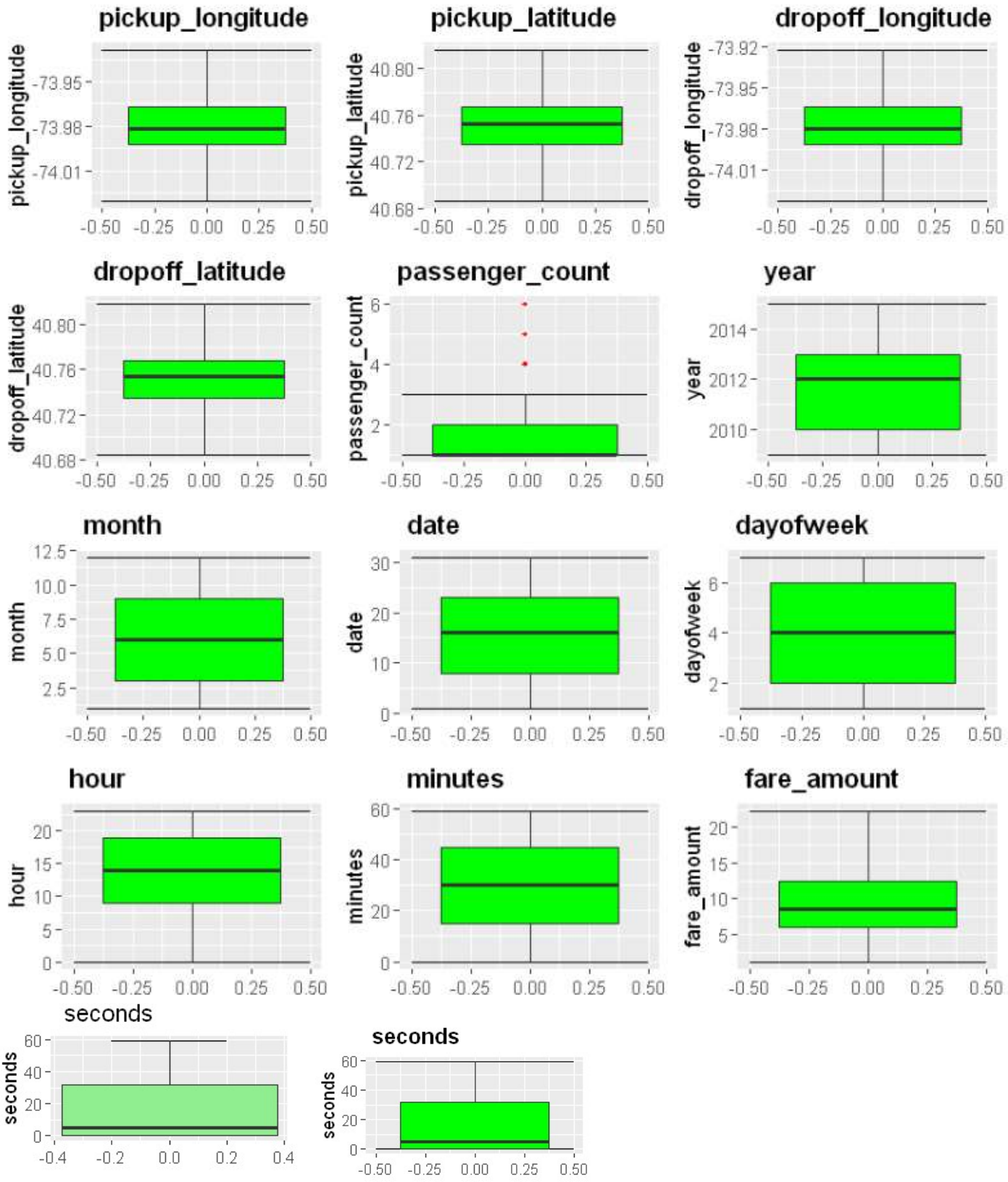
| | fare_amount | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | Year | Month | Date | Dayofweek | Hour | Minutes |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **5685** | 3.3 | -73.947235 | 401.083332 | -73.951392 | 40.778927 | 1.0 | 2011.0 | 7.0 | 30.0 | 5.0 | 11.0 | 15.0 |

Boxplots are one of the methods to remove outlier and after removing outliers by capping method. Now our data is free from outliers.

**In Python:-** Box Plot after removing outliers



**In R:-** Box Plot after removing outliers



## Summary statistics both in R & Python after outliers removal:

```
pickup_longitude pickup_latitude dropoff_longitude dropoff_latitude
Min.   :-74.03   Min.   :40.69   Min.   :-74.03    Min.   :40.68
1st Qu.:-73.99   1st Qu.:40.73   1st Qu.:-73.99    1st Qu.:40.73
Median :-73.98   Median :40.75   Median :-73.98    Median :40.75
Mean   :-73.98   Mean   :40.75   Mean   :-73.98    Mean   :40.75
3rd Qu.:-73.97   3rd Qu.:40.77   3rd Qu.:-73.96    3rd Qu.:40.77
Max.   :-73.93   Max.   :40.82   Max.   :-73.92    Max.   :40.82
passenger_count      year          month             date          dayofweek
Min.   :1.000   Min.   :2009   Min.   : 1.000   Min.   : 1.00   Min.   :1.00
1st Qu.:1.000   1st Qu.:2010   1st Qu.: 3.000   1st Qu.: 8.00   1st Qu.:2.00
Median :1.000   Median :2012   Median : 6.000   Median :16.00   Median :4.00
Mean   :1.649   Mean   :2012   Mean   : 6.262   Mean   :15.67   Mean   :4.09
3rd Qu.:2.000   3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:6.00
Max.   :6.000   Max.   :2015   Max.   :12.000   Max.   :31.00   Max.   :7.00
     hour          minutes         seconds       fare_amount
Min.   : 0.00   Min.   : 0.00   Min.   : 0.00   Min.   : 1.14
1st Qu.: 9.00   1st Qu.:15.00   1st Qu.: 0.00   1st Qu.: 6.00
Median :14.00   Median :30.00   Median : 5.00   Median : 8.50
Mean   :13.49   Mean   :29.65   Mean   :16.12   Mean   :10.07
3rd Qu.:19.00   3rd Qu.:45.00   3rd Qu.:32.00   3rd Qu.:12.50
Max.   :23.00   Max.   :59.00   Max.   :59.00   Max.   :22.25
```

[72]:

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| fare_amount | 15975.0 | 10.074651 | 5.459685 | 1.140000 | 6.000000 | 8.500000 | 12.500000 | 22.250000 |
| pickup_longitude | 15975.0 | -73.978121 | 0.020731 | -74.030129 | -73.992157 | -73.981698 | -73.966842 | -73.928870 |
| pickup_latitude | 15975.0 | 40.750318 | 0.025823 | 40.686269 | 40.734946 | 40.752632 | 40.767398 | 40.816075 |
| dropoff_longitude | 15975.0 | -73.975960 | 0.022854 | -74.032488 | -73.991182 | -73.980168 | -73.963644 | -73.922338 |
| dropoff_latitude | 15975.0 | 40.750590 | 0.028225 | 40.684738 | 40.734708 | 40.753577 | 40.768021 | 40.817990 |
| passenger_count | 15975.0 | 1.649703 | 1.264895 | 1.000000 | 1.000000 | 1.000000 | 2.000000 | 6.000000 |
| Year | 15975.0 | 2011.732770 | 1.867097 | 2009.000000 | 2010.000000 | 2012.000000 | 2013.000000 | 2015.000000 |
| Month | 15975.0 | 6.261659 | 3.447493 | 1.000000 | 3.000000 | 6.000000 | 9.000000 | 12.000000 |
| Date | 15975.0 | 15.667167 | 8.683720 | 1.000000 | 8.000000 | 16.000000 | 23.000000 | 31.000000 |
| Dayofweek | 15975.0 | 3.033427 | 1.968802 | 0.000000 | 1.000000 | 3.000000 | 5.000000 | 6.000000 |
| Hour | 15975.0 | 13.494460 | 6.521594 | 0.000000 | 9.000000 | 14.000000 | 19.000000 | 23.000000 |
| Minutes | 15975.0 | 29.654523 | 17.299442 | 0.000000 | 15.000000 | 30.000000 | 45.000000 | 59.000000 |
| Seconds | 15975.0 | 16.125196 | 19.489079 | 0.000000 | 0.000000 | 5.000000 | 32.000000 | 59.000000 |

### 2.4.4 Feature Engineering :

Feature engineering is the science (and art) of extracting more information from existing data, not adding any new data to it, but making the data more meaningful and usable.

In our Project we derived a new variable distance from given pickup and drop off latitudes and longitudes using haversine formula.

The **haversine formula** determines the great-circle distance between two points on a sphere given their longitudes and latitudes. Important in navigation, it is a special case of a more general formula in spherical trigonometry, the law of haversines, that relates the sides and angles of spherical triangles.

## Haversine formula to find distance between two points on a sphere

The **Haversine** formula calculates the shortest distance between two points on a sphere using their latitudes and longitudes measured along the surface. It is important for use in navigation. The haversine can be expressed in trignometric function as:

$$haversine(\theta) = sin^2\left(\frac{\theta}{2}\right)$$

The haversine of the central angle (which is d/r) is calculated by the following formula:

$$\left(\frac{d}{r}\right) = haversine(\Phi_2 - \Phi_1) + cos(\Phi_1)cos(\Phi_2)haversine(\lambda_2 - \lambda_1)$$

where r is the radius of earth(6371 km), d is the distance between two points, $\phi_1, \phi_2$ is latitude of the two points and $\lambda_1, \lambda_2$ is longitude of the two points respectively.

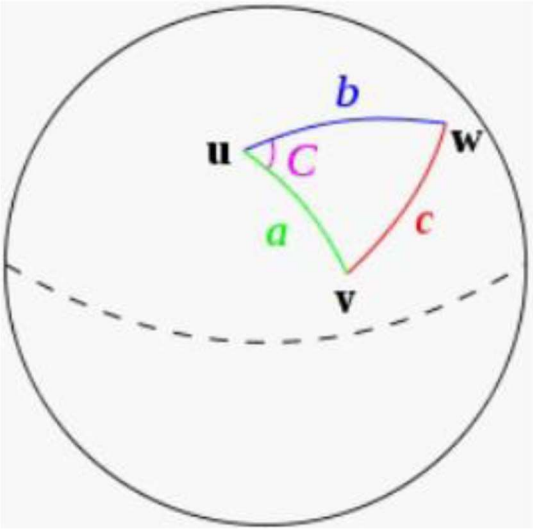
Solving d by applying the inverse haversine or by using the inverse sine function, we get:

$$d = rhav^{-1}(h) = 2rsin^{-1}(\sqrt{h})$$

or

$$d = 2rsin^{-1}\left(\sqrt{sin^2\left(\frac{\Phi_2-\Phi_1}{2}\right) + cos(\Phi_1)cos(\Phi_2)sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right)$$

The distance between Big Ben in London (51.5007° N, 0.1246° W) and The Statue of Liberty in New York (40.6892° N, 74.0445° W) is 5574.8 km. This is not the exact measurement because the formula assumes that the Earth is a perfect sphere when in fact it is an oblate spheroid.




After extracting distance, our updated train dataset becomes;

**In R :**

| pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | year | month | date | dayofweek | hour | minutes | distance | fare_amount |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| -73.92886 | 40.72132 | -73.92234 | 40.71228 | 1 | 2009 | 6 | 15 | 2 | 17 | 26 | 1.147313 | 4.5 |
| -74.01605 | 40.71130 | -73.97927 | 40.78200 | 1 | 2010 | 1 | 5 | 3 | 16 | 52 | 8.459600 | 16.9 |
| -73.98274 | 40.76127 | -73.99124 | 40.75056 | 2 | 2011 | 8 | 18 | 5 | 0 | 35 | 1.391082 | 5.7 |
| -73.98713 | 40.73314 | -73.99157 | 40.75809 | 1 | 2012 | 4 | 21 | 7 | 4 | 30 | 2.802406 | 7.7 |
| -73.96810 | 40.76801 | -73.95665 | 40.78376 | 1 | 2010 | 3 | 9 | 3 | 7 | 51 | 2.001396 | 5.3 |
| -74.00096 | 40.73163 | -73.97289 | 40.75823 | 1 | 2011 | 1 | 6 | 5 | 9 | 50 | 3.791482 | 12.1 |

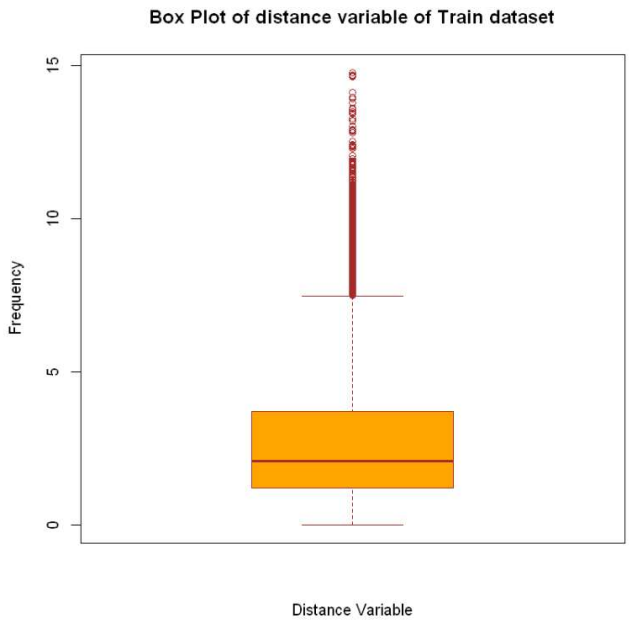
**In Python:**

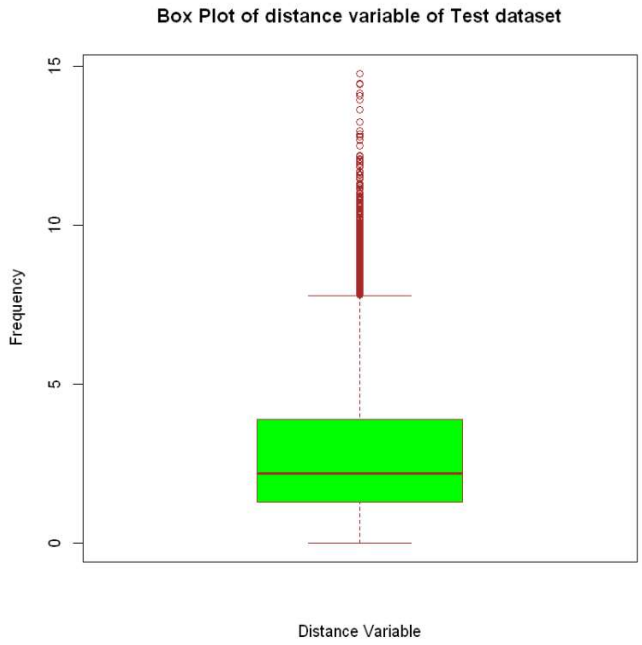
[78]:

| | fare_amount | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | Year | Month | Date | Dayofweek | Hour | Minutes | distance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4.5 | -73.928852 | 40.721319 | -73.922334 | 40.712278 | 1.0 | 2009.0 | 6.0 | 15.0 | 0.0 | 17.0 | 26.0 | 1.145610 |
| 1 | 16.9 | -74.016048 | 40.711303 | -73.979268 | 40.782004 | 1.0 | 2010.0 | 1.0 | 5.0 | 1.0 | 16.0 | 52.0 | 8.450134 |
| 2 | 5.7 | -73.982738 | 40.761270 | -73.991242 | 40.750562 | 2.0 | 2011.0 | 8.0 | 18.0 | 3.0 | 0.0 | 35.0 | 1.389525 |
| 3 | 7.7 | -73.987130 | 40.733143 | -73.991567 | 40.758092 | 1.0 | 2012.0 | 4.0 | 21.0 | 5.0 | 4.0 | 30.0 | 2.799270 |
| 4 | 5.3 | -73.968095 | 40.768008 | -73.956655 | 40.783762 | 1.0 | 2010.0 | 3.0 | 9.0 | 1.0 | 7.0 | 51.0 | 1.999157 |

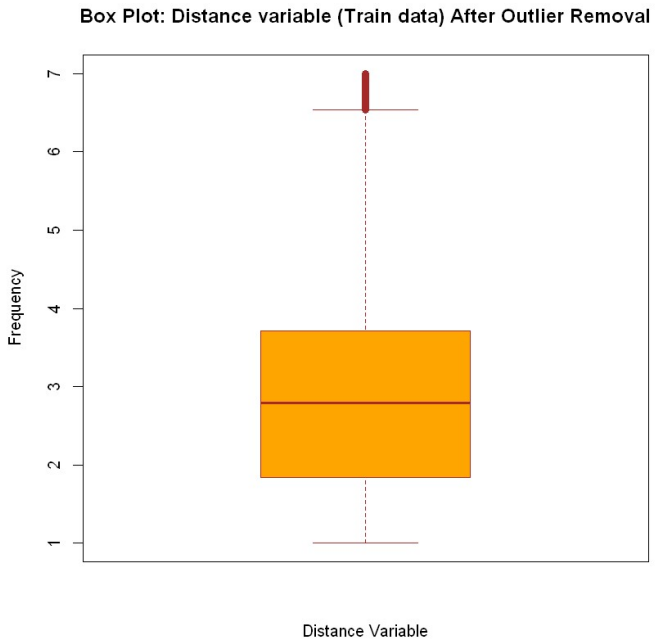
Let's repeat the process of outlier analysis for newly extracted distance variable. And apply capping method to address outliers if any.

## In R: -

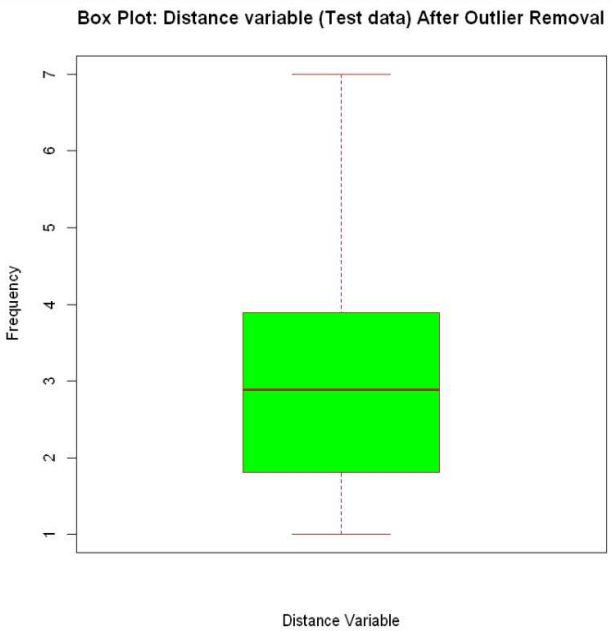



Box plot shows minimum value is zero which is incorrect for any cab ride. Hence 2987 and 1549 outliers in train and test dataset respectively are imputed with mean values. for upper limit will use capping method to limit maximum values.

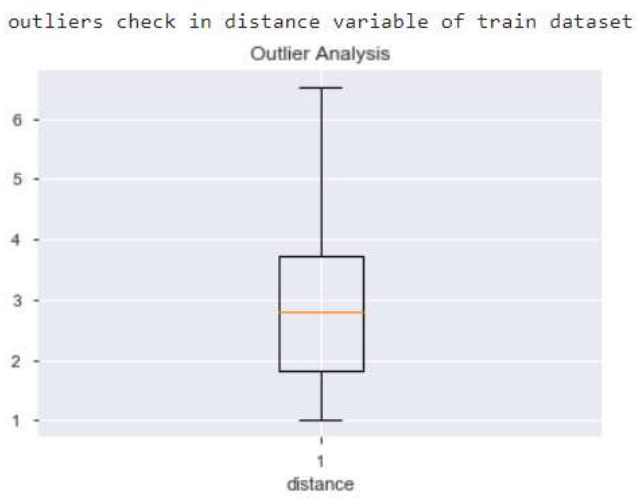


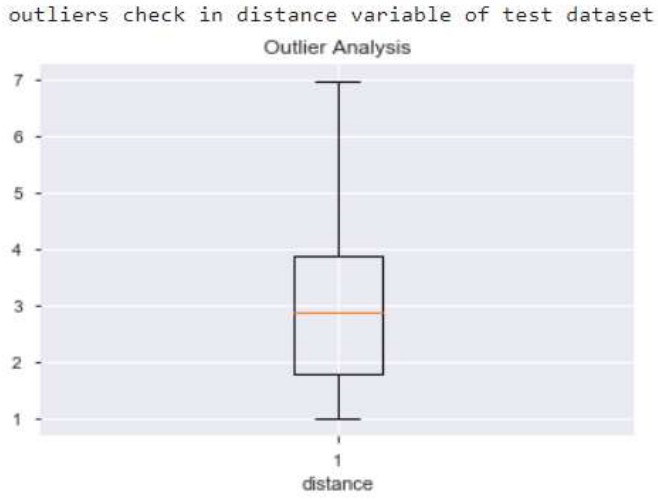

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.837   2.792   3.073   3.716   7.000
```



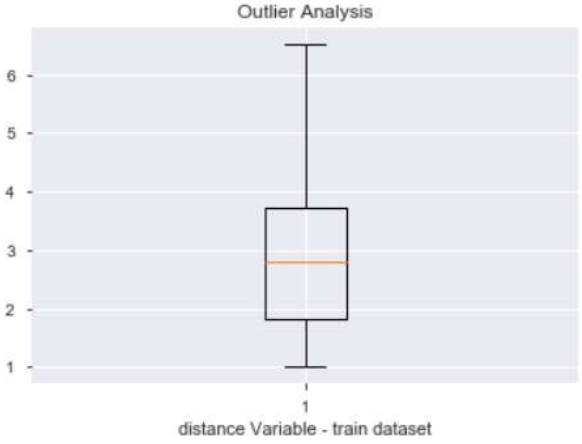

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   1.810   2.888   3.124   3.887   7.000
```

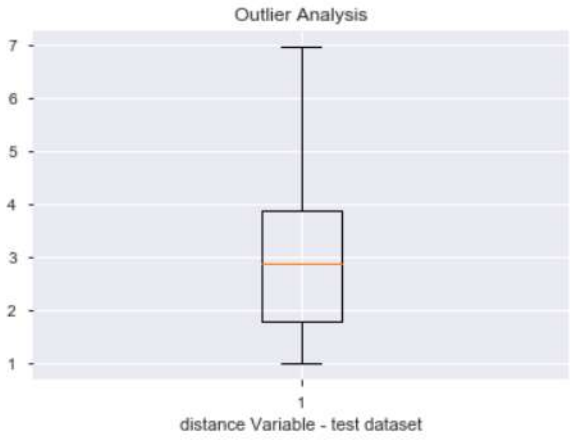
**In python:-**




Box plot shows minimum value is zero which is incorrect for any cab ride. Hence 2990 and 1553 outliers in train and test dataset are imputed with mean values. for upper limit will use capping method to limit maximum values.




### 2.4.5 Frequency Distribution Visualization:

To check distribution of each continuous variable we plotted histogram for each variable Both in R and Python, we can also check distribution using summary or describe function. In our project it can be observed that fare amount and distance lightly skewed, whereas rest of the variables normally distributed. The skewness is likely because of the presence of more information or huge data in those variables.

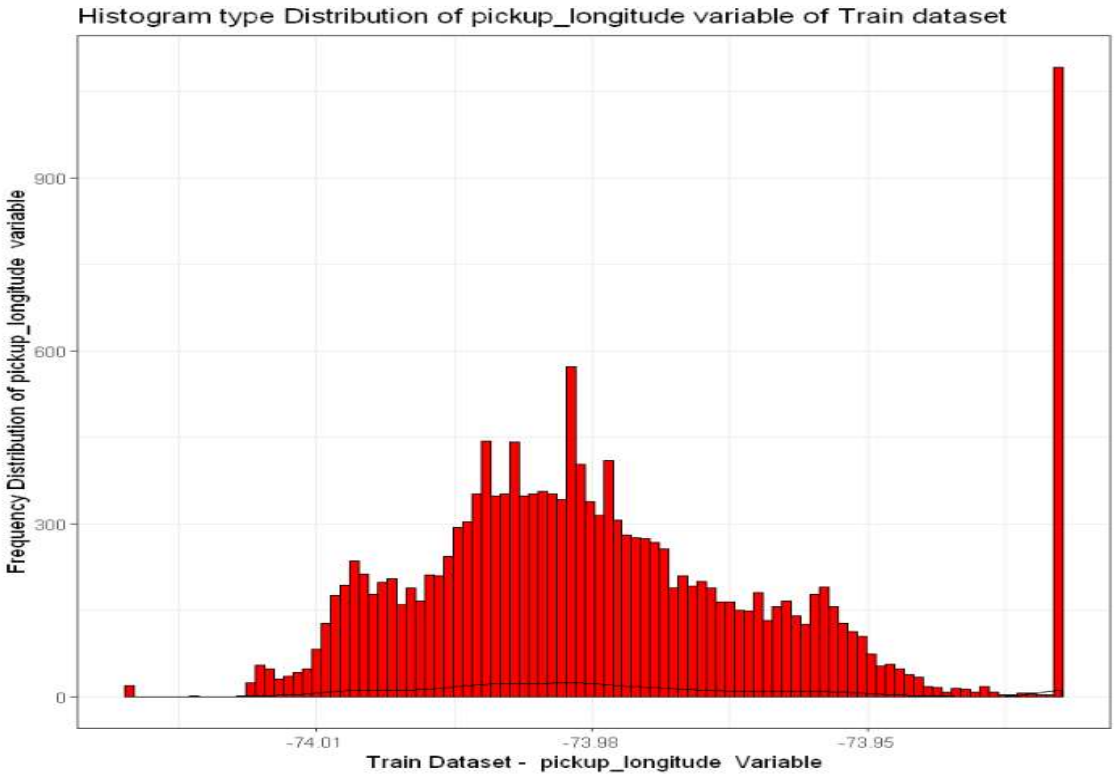



Figure 1.1

Pickup_Longitude variable fairly follows normally distributed.

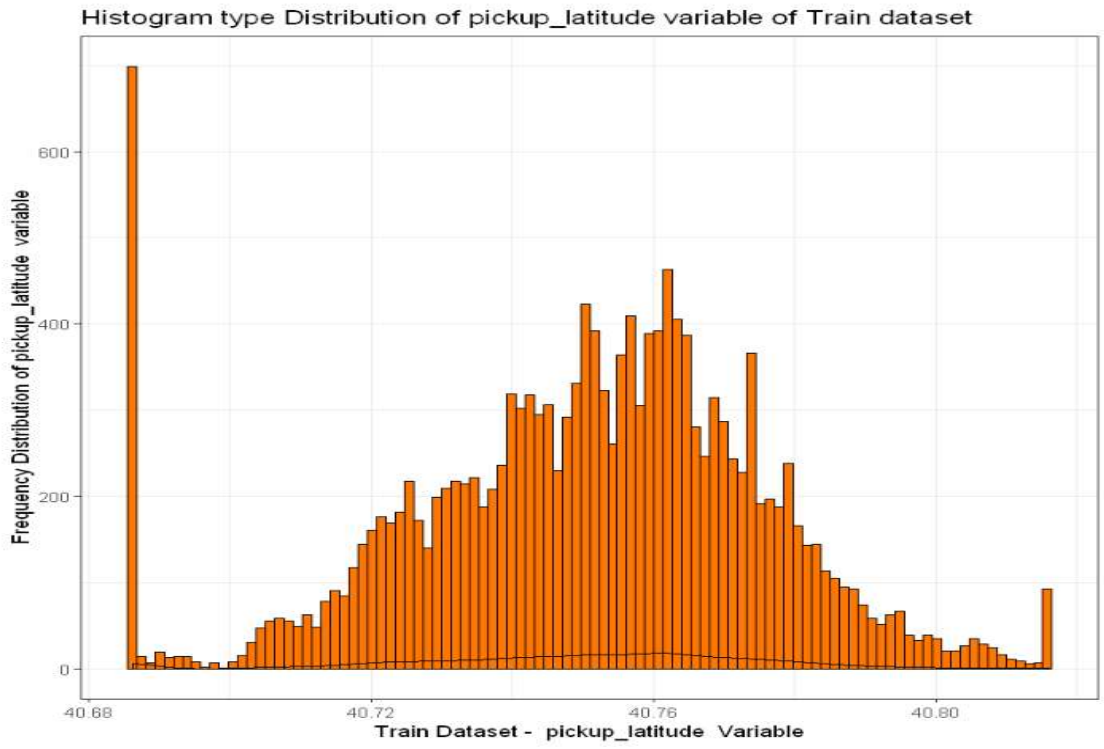



Figure 1.2

Pickup_Latitude variable fairly follows normally distributed.

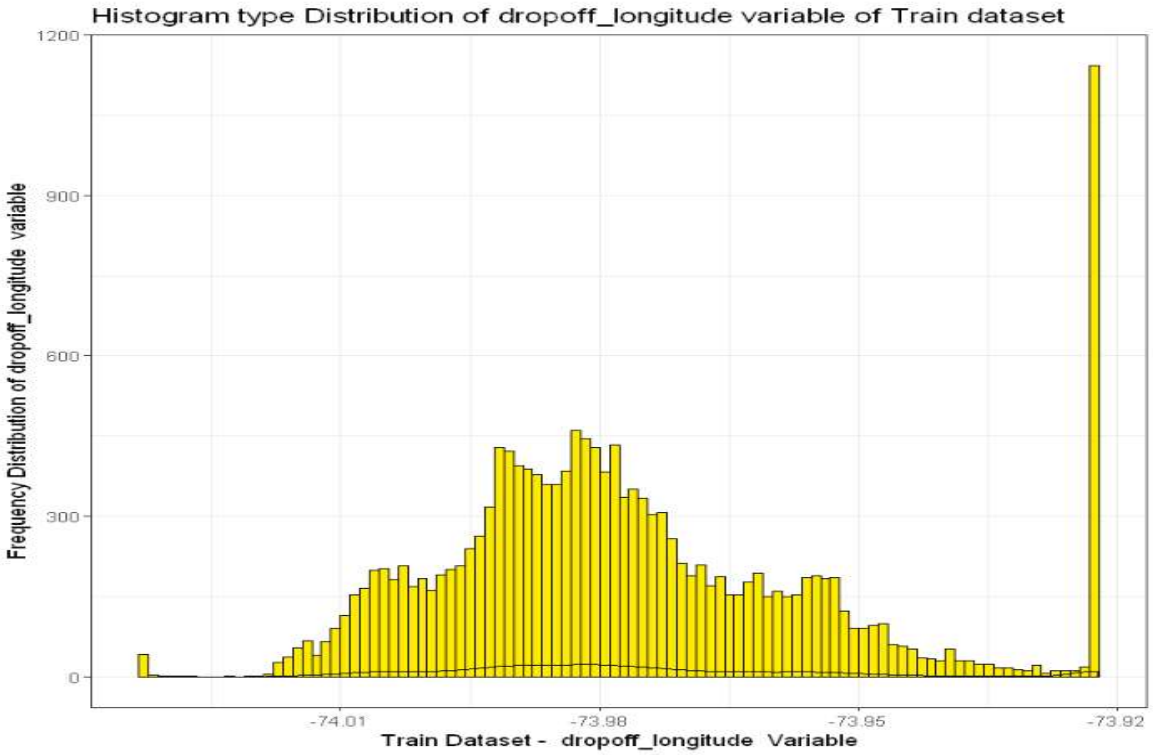



Figure 1.3

Dropoff_Longitude variable fairly follows normally distributed.

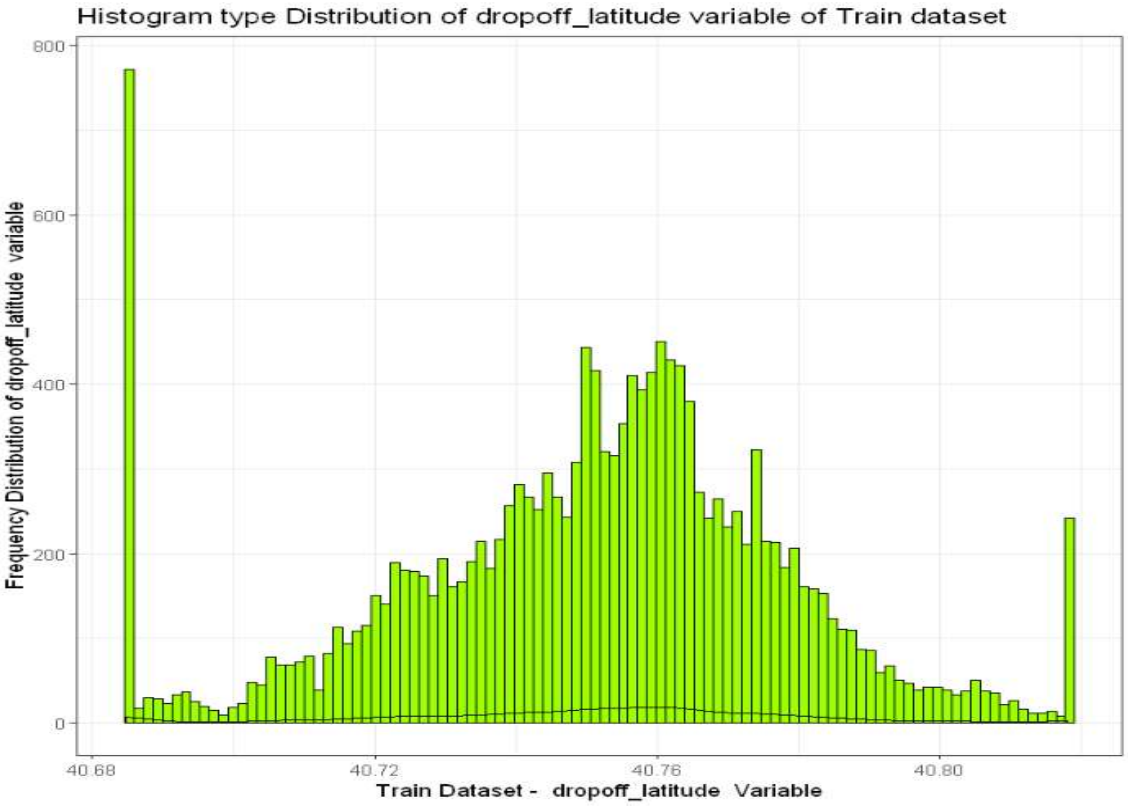



Figure 1.4

Dropoff_Latitude variable fairly follows normally distributed.

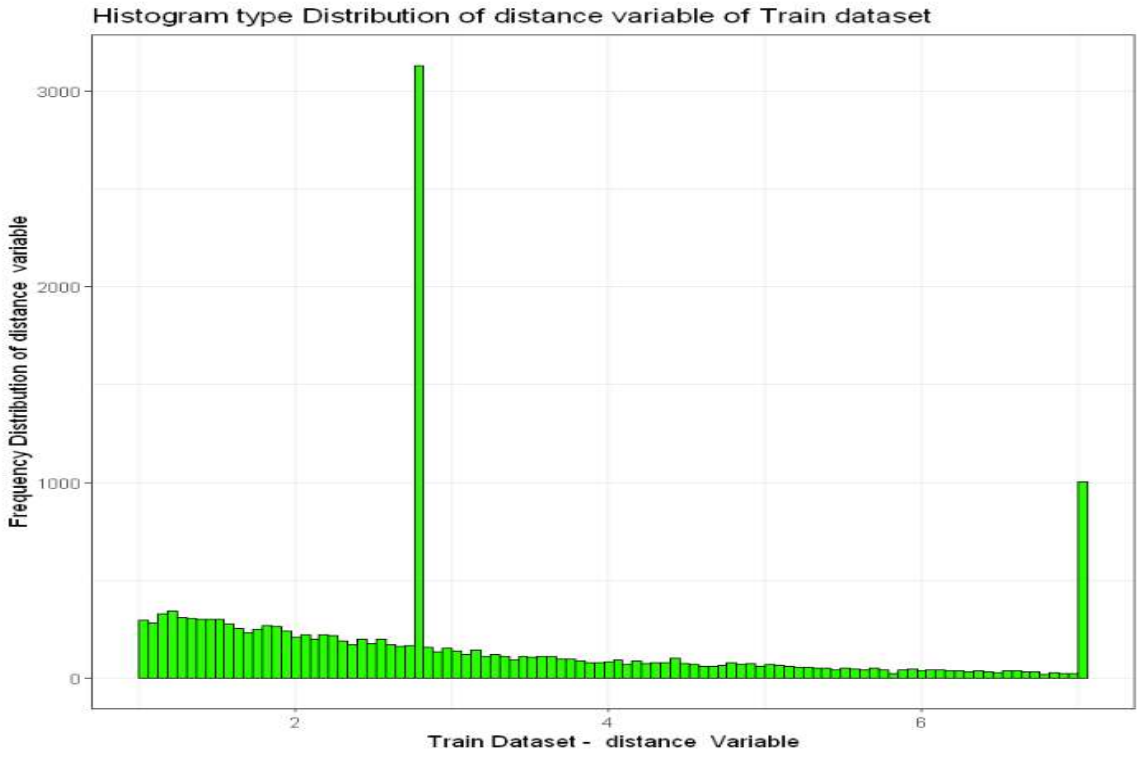



Figure 1.5

Distance variable is right skewed needs to be addressed for normalization.

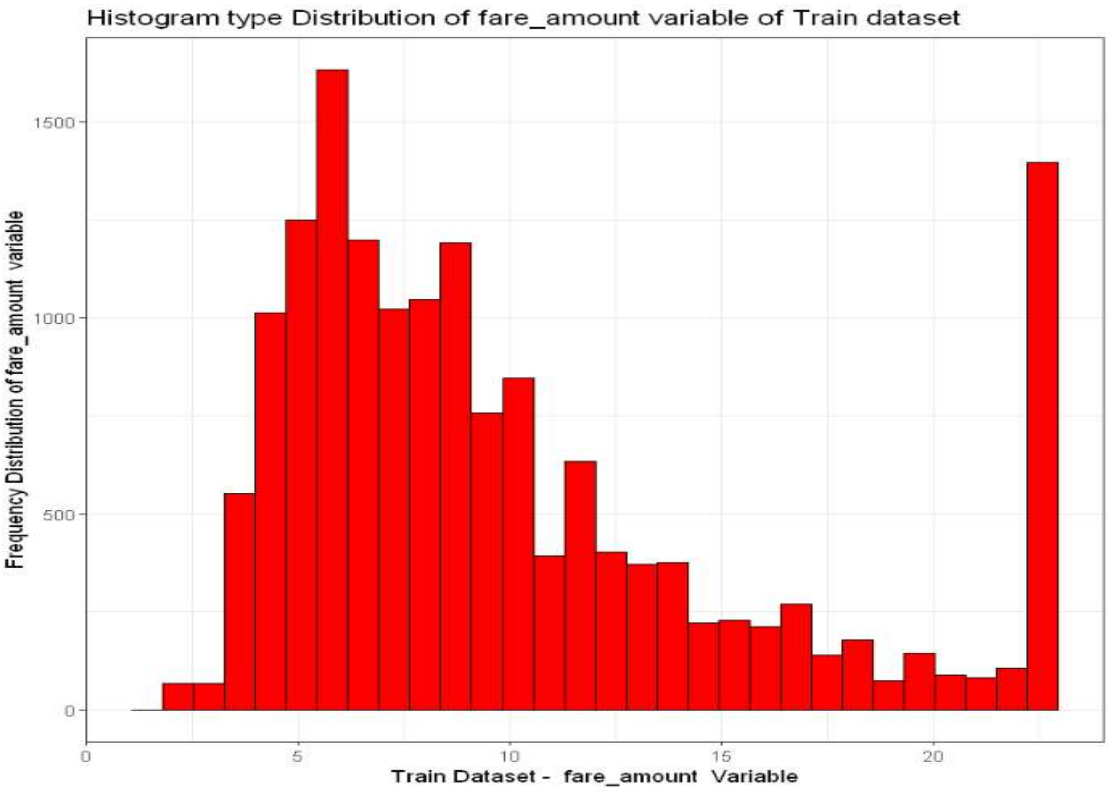



Figure 1.6

Fare amount variable is right skewed needs to be addressed for normalization.

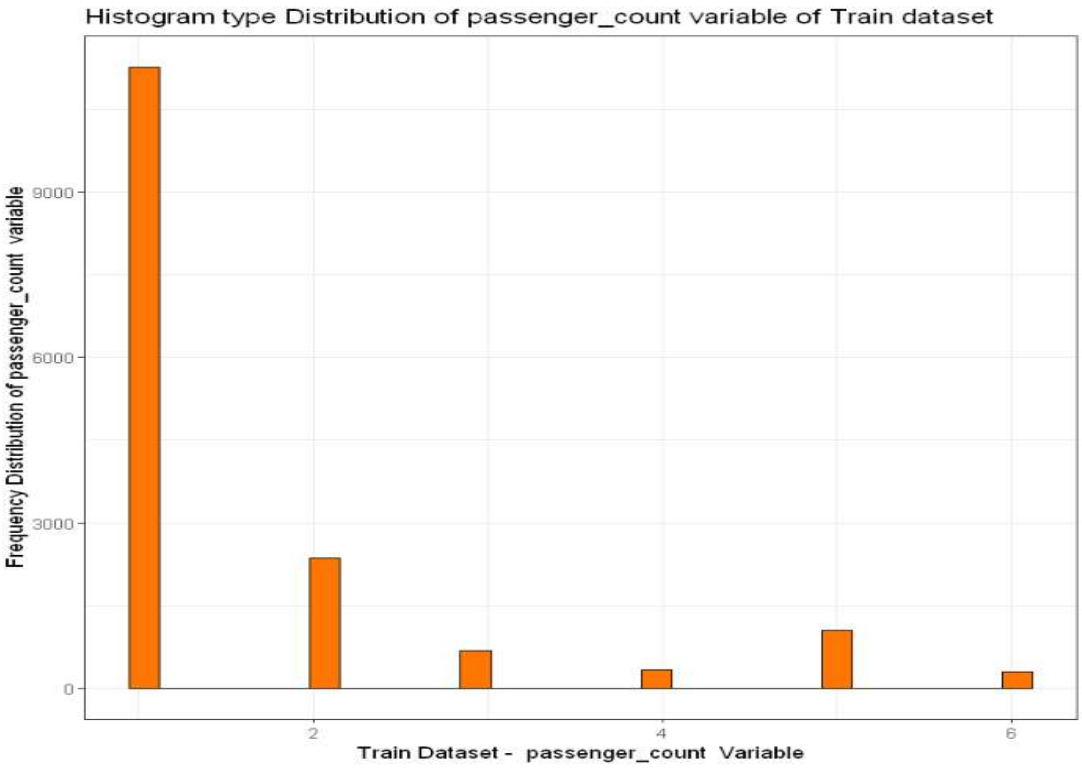



Figure 1.7

Passenger count ranges from 1-6. But majority rides for single person preferred.

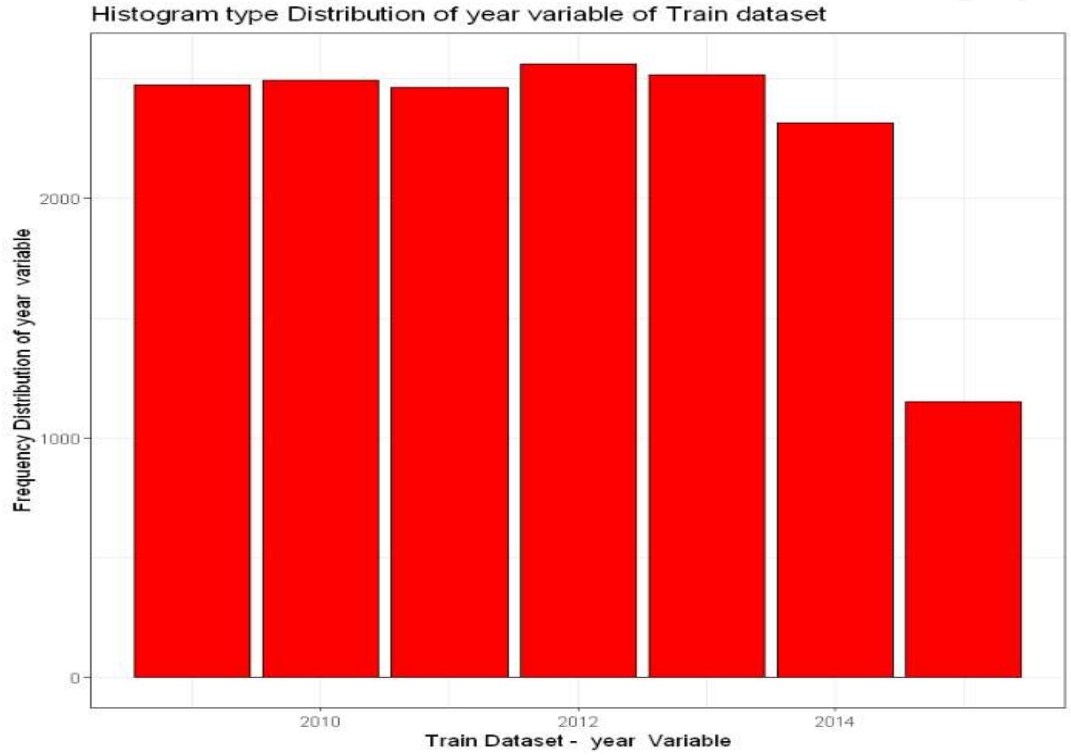



Figure 1.8

In 2012 fare amount had highest fare amount. Considerably 2015 has low fare amount.

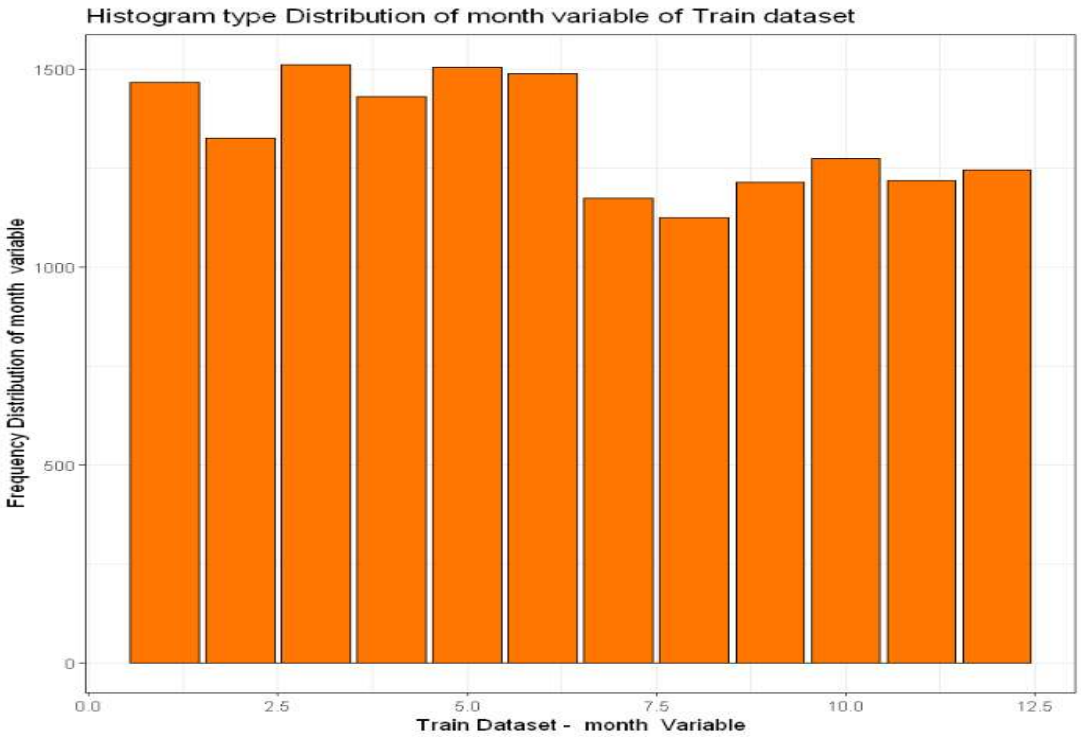



Figure 1.9

March to June had highest fare amount due to more no. of bookings.

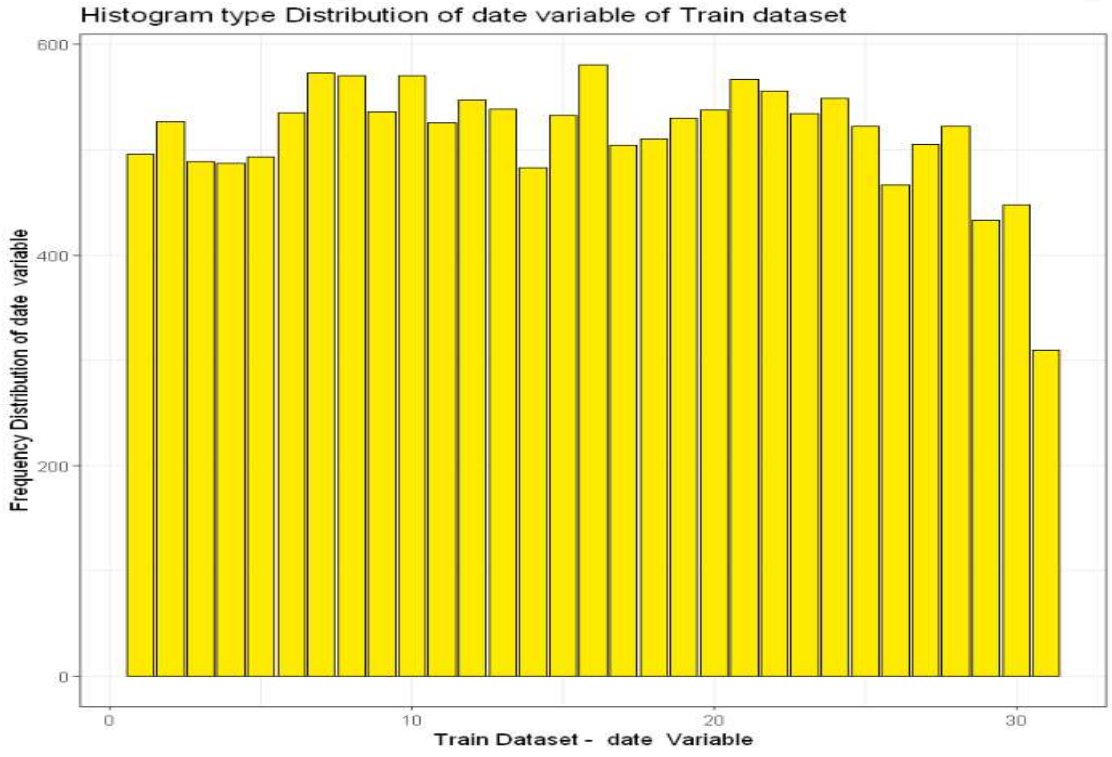



Figure 1.10

During mid weeks of the month had more fare amount. And less during month ends.

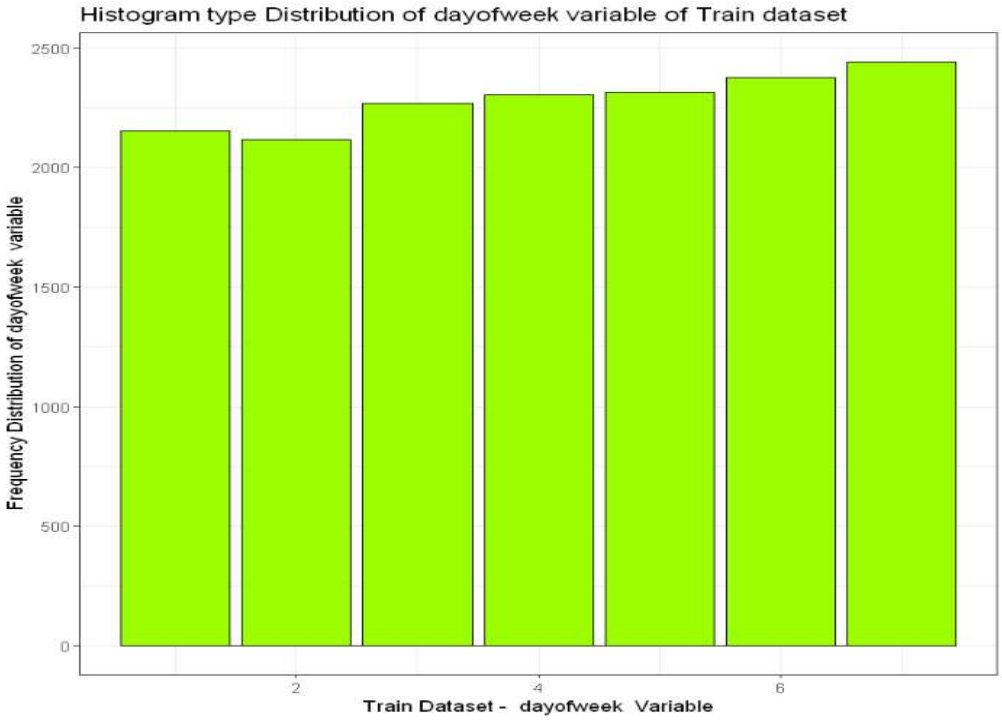



Figure 1.11

During weekends fare amount shows highest due to more no. of bookings.

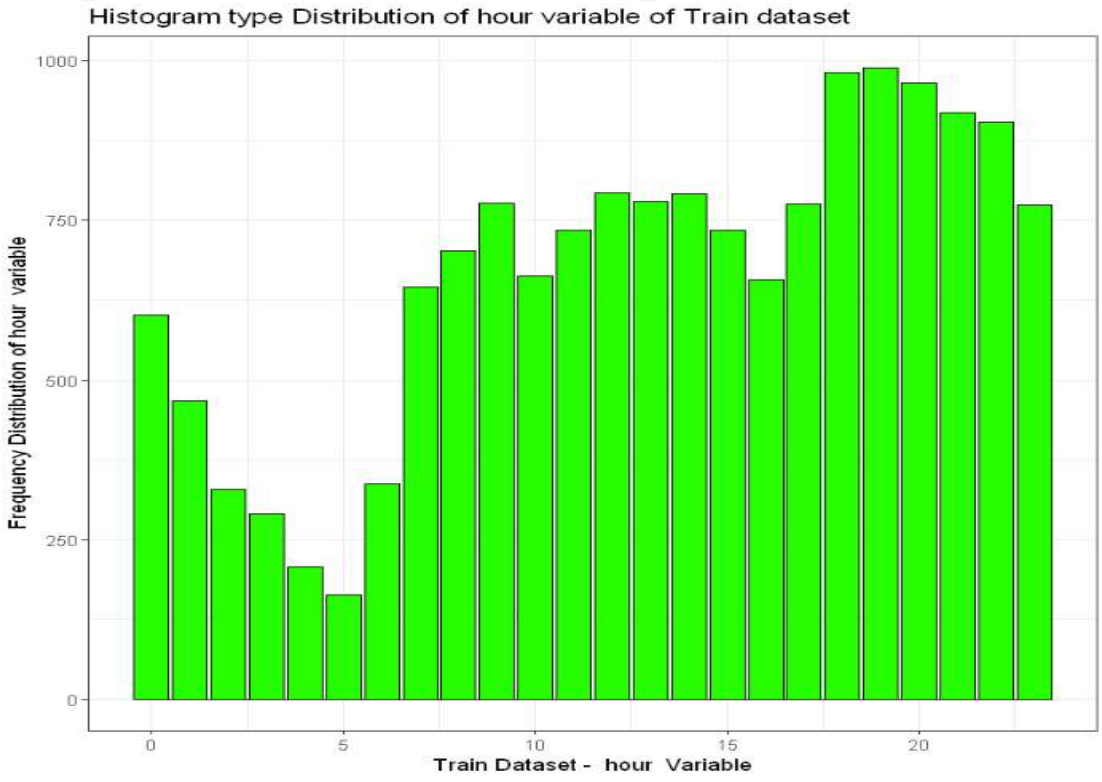



Figure 1.12

During 6pm to 11pm became peak hours to fetch highest.

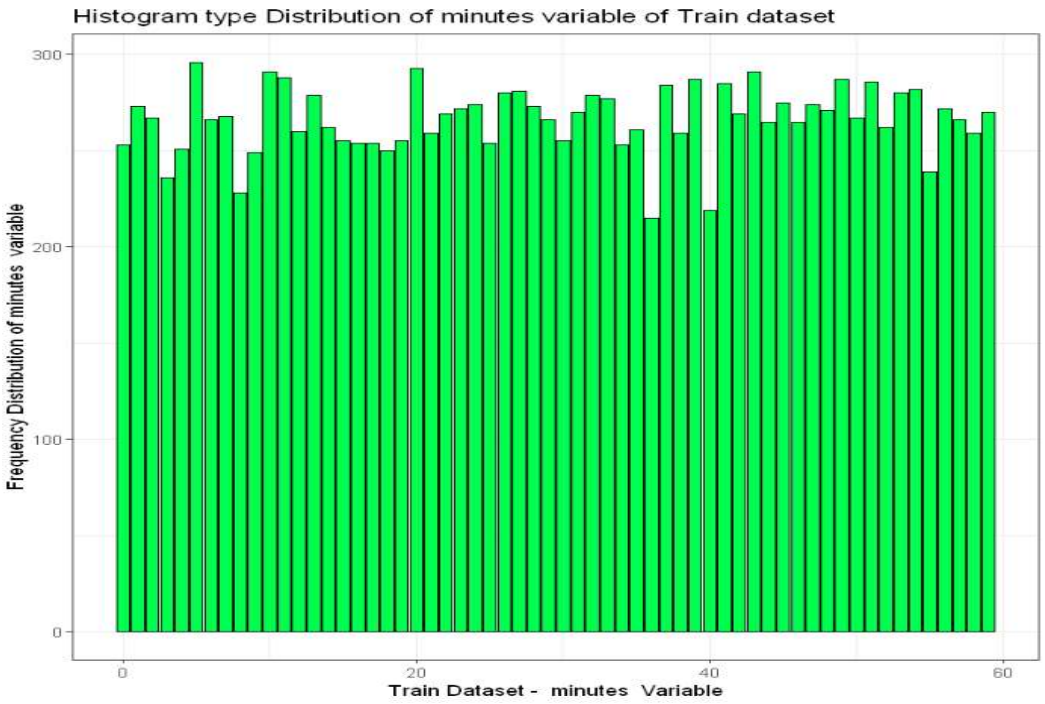



Figure 1.13

Fare amounts are higher during 10-20-50 minutes.

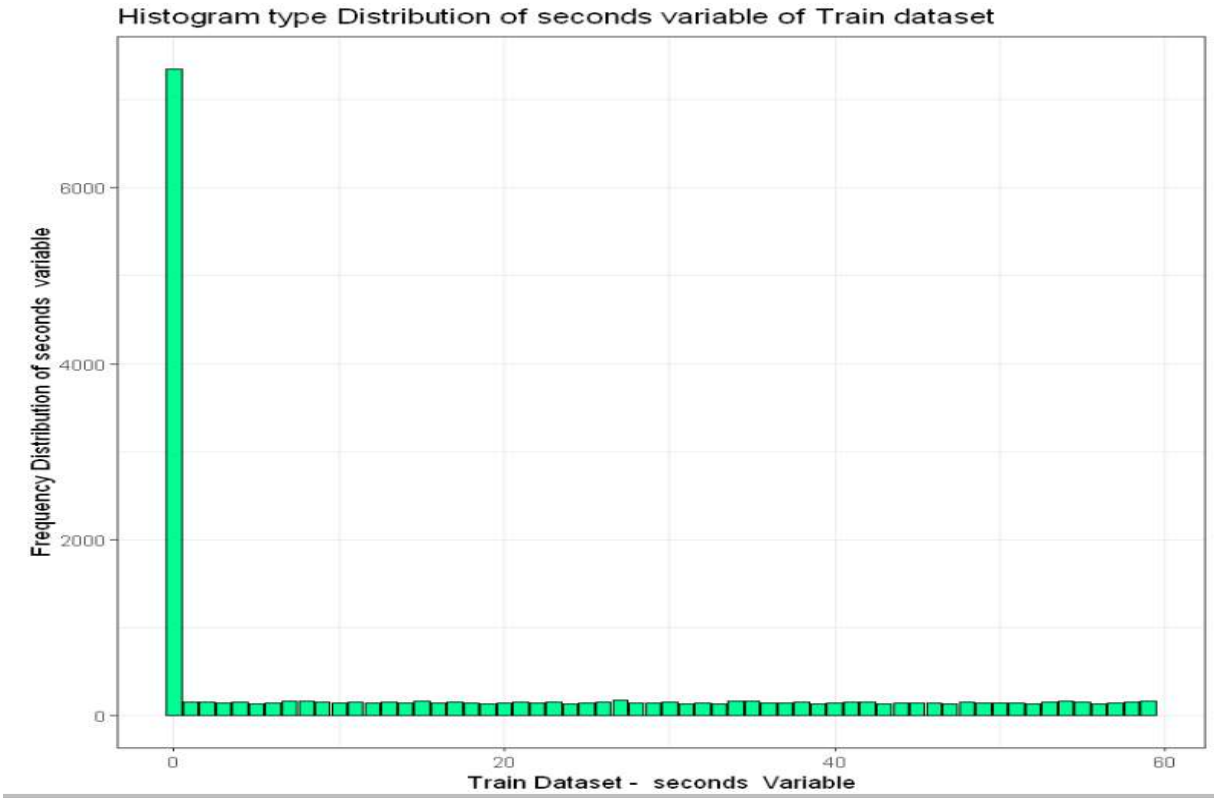



Figure 1.14

Fare amounts doesn't much affected with respect to Seconds.

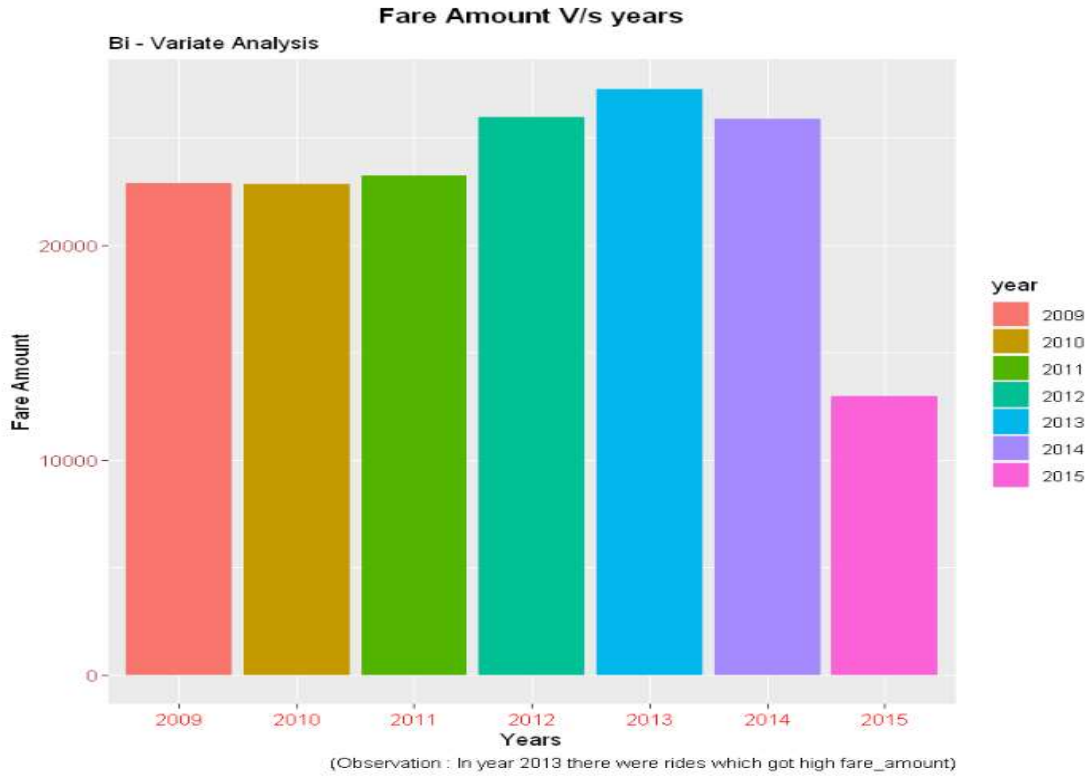



Figure 2.1

From above bar plot, in year 2013 rides had high fare_amount and lowest ride fare amounts in 2015. From 2009 after 2-3 years due to popularity fare amount went high. But In 2015 due to high competition might have lowered the fare amount.

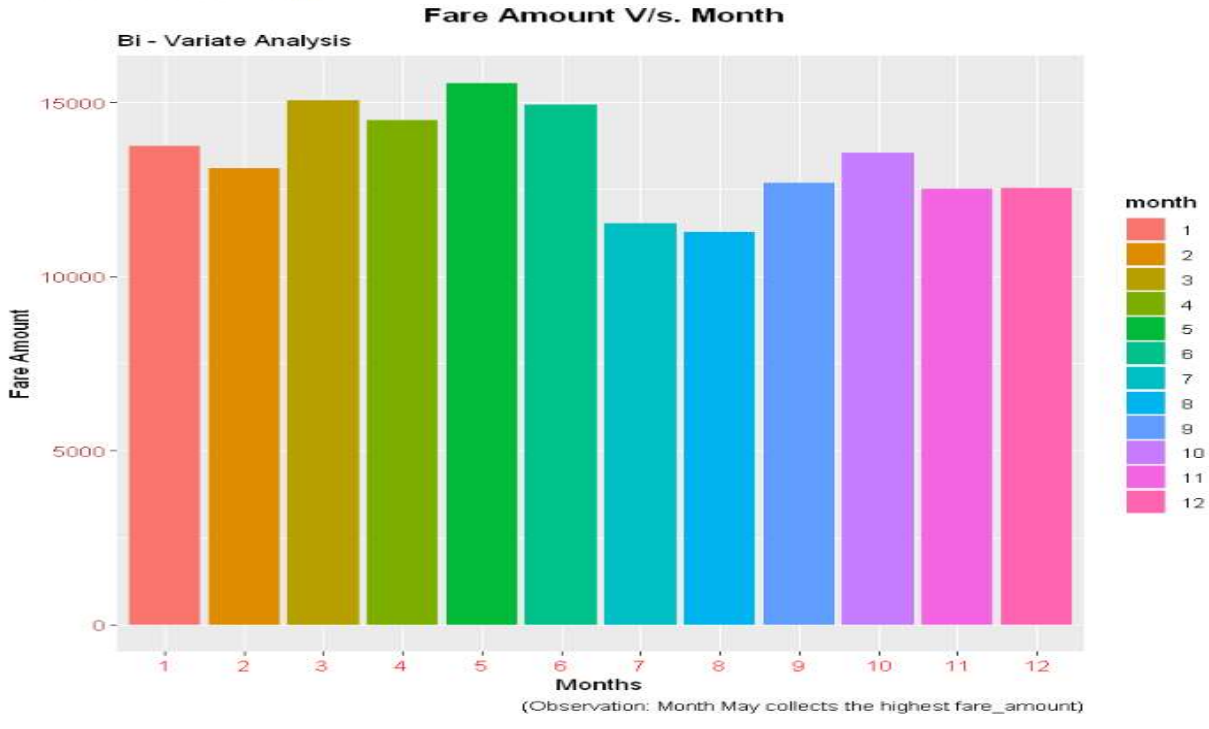



Figure 2.2

March to June month had very good fare amounts compared to other months. Lowest fare amount found in august month.

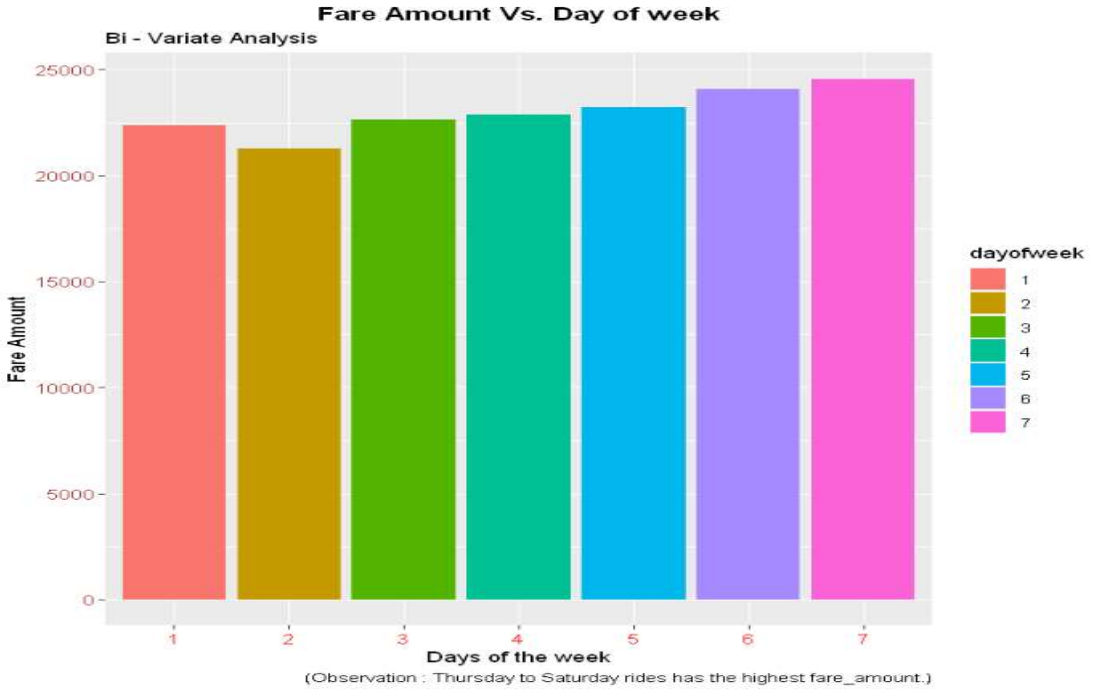



Figure 2.3

Friday and Saturday rides fare amount are higher than other weekdays due to weekend.

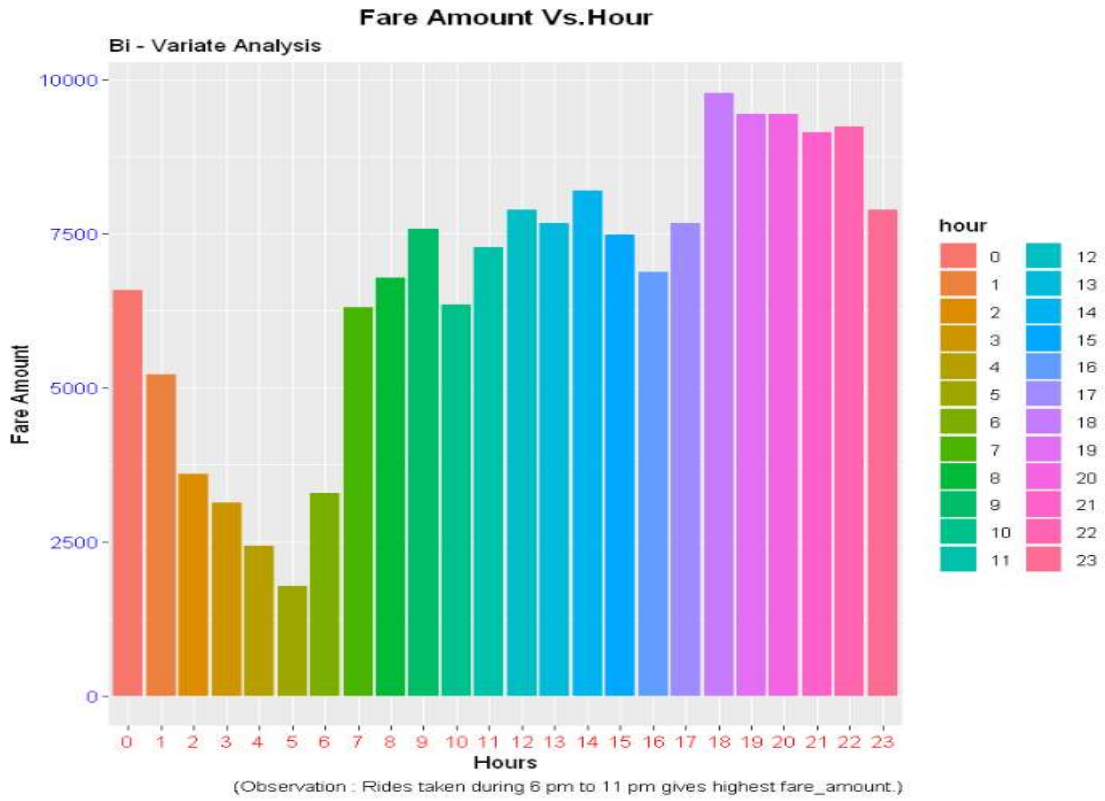



Figure 2.4

Rides during 6pm to 11pm has higher fare amounts these might be due to late office working hours. During 5am low fare amounts due to non-availability of bookings.

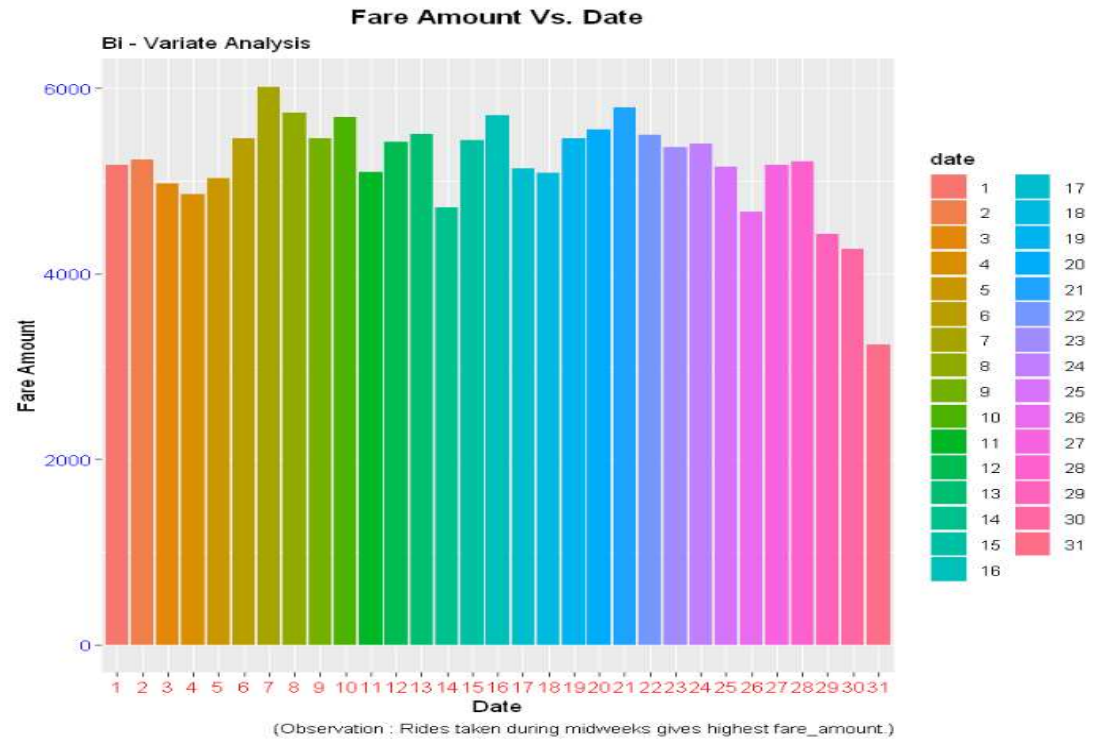



Figure 2.5

Rides taken during midweeks of the month has moderately has high ride fare amount.

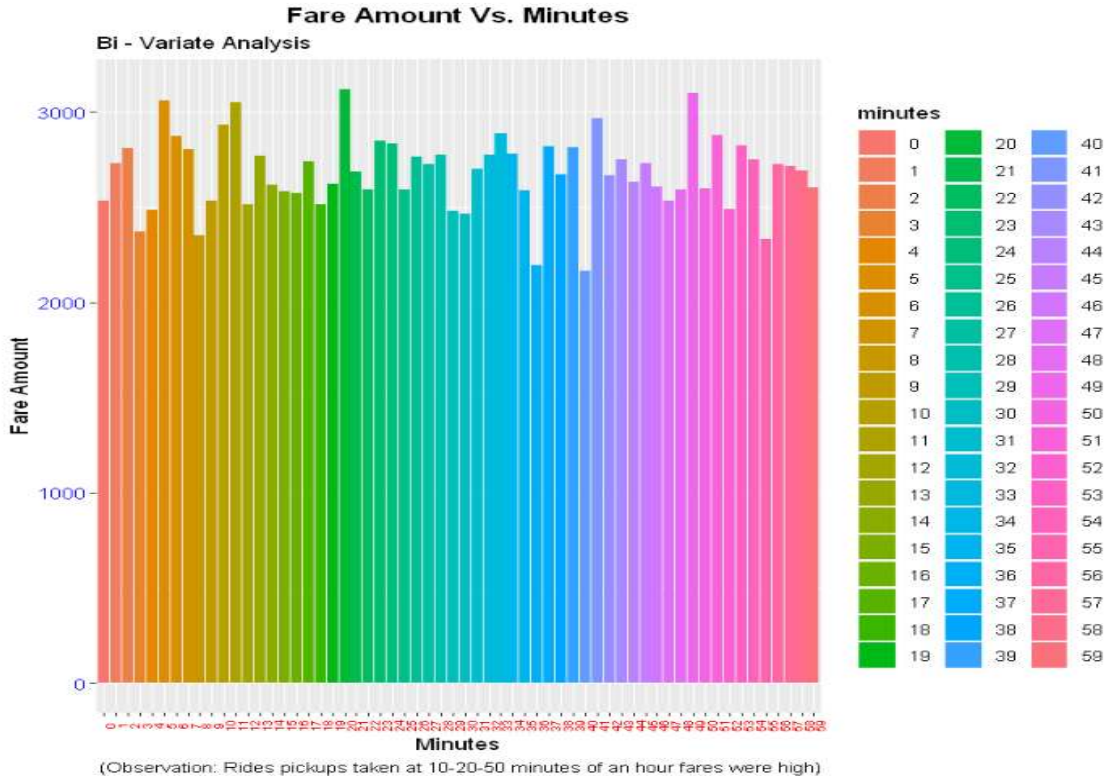



Figure 2.5

No. of bookings taken place at 10-20-50 minutes are having higher fare amounts.

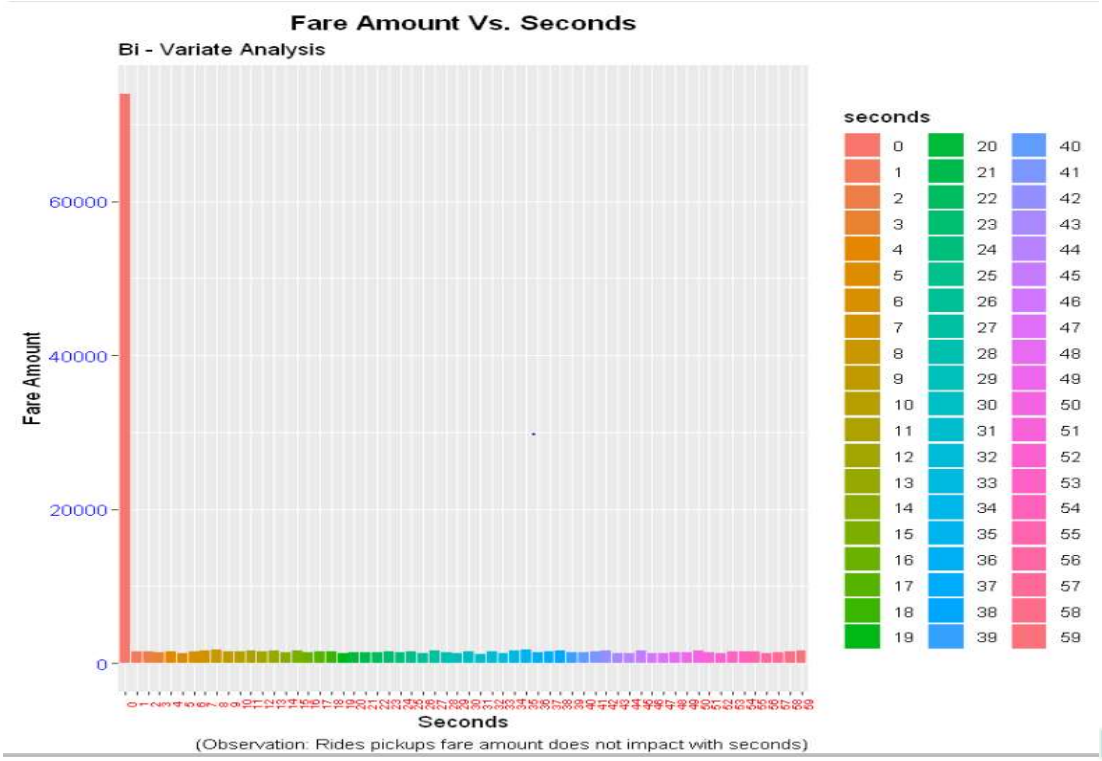
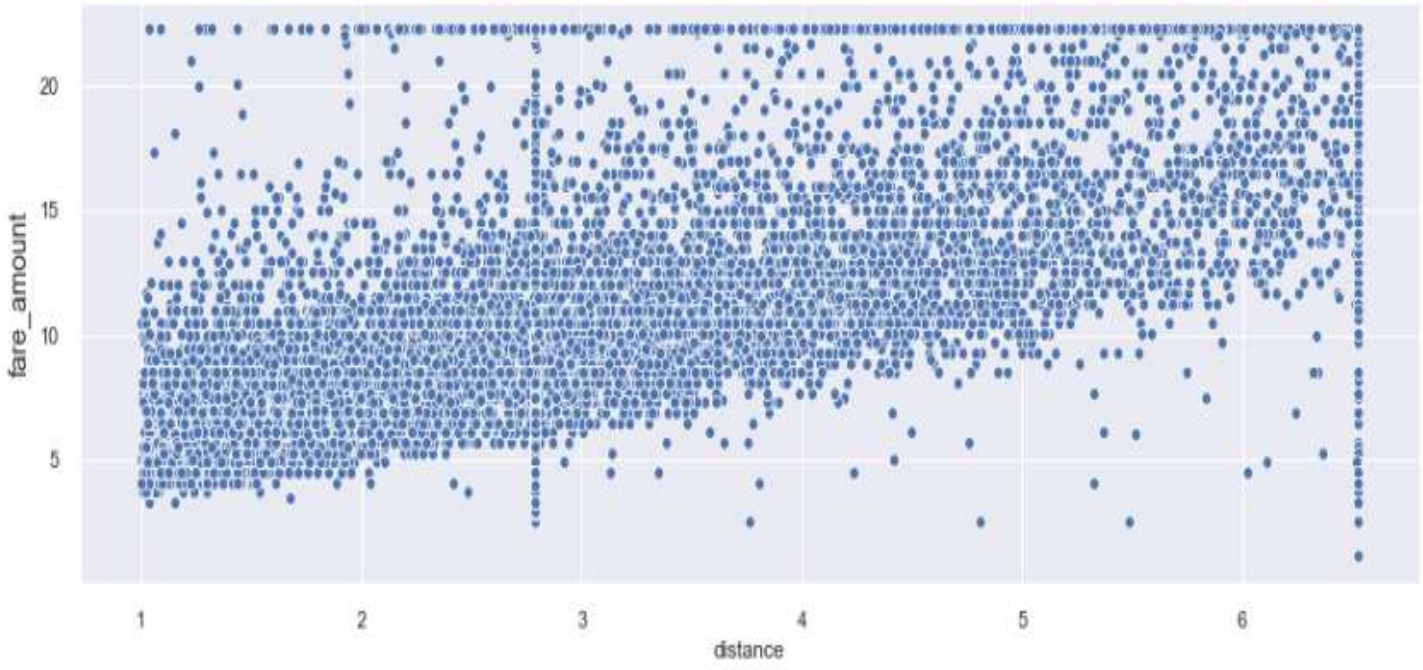
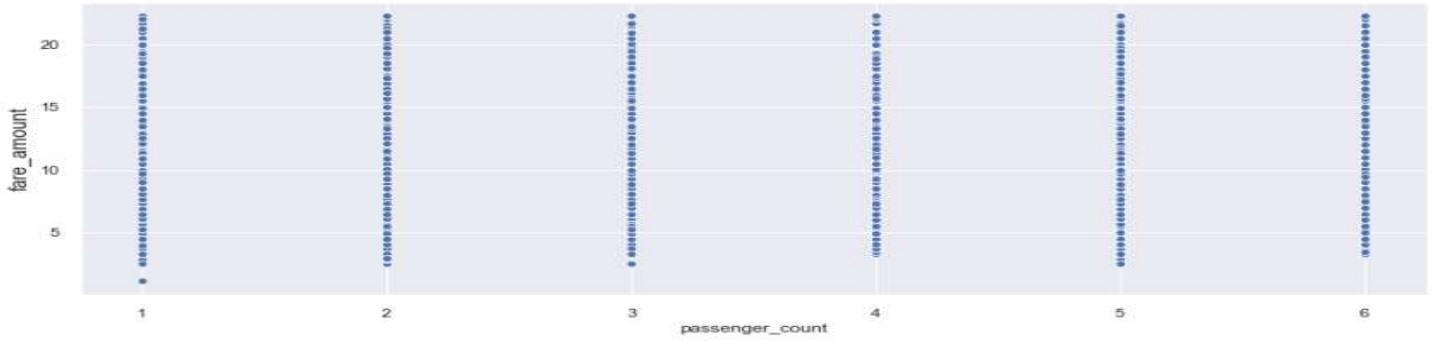


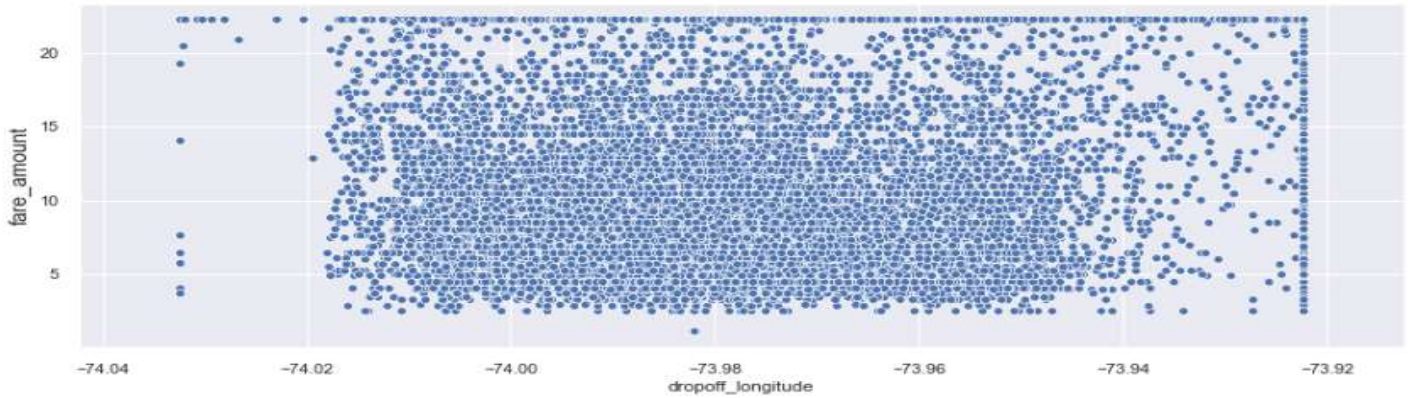

Figure 2.6

Rides pickups fare amount doesn't impact much with no. of seconds.

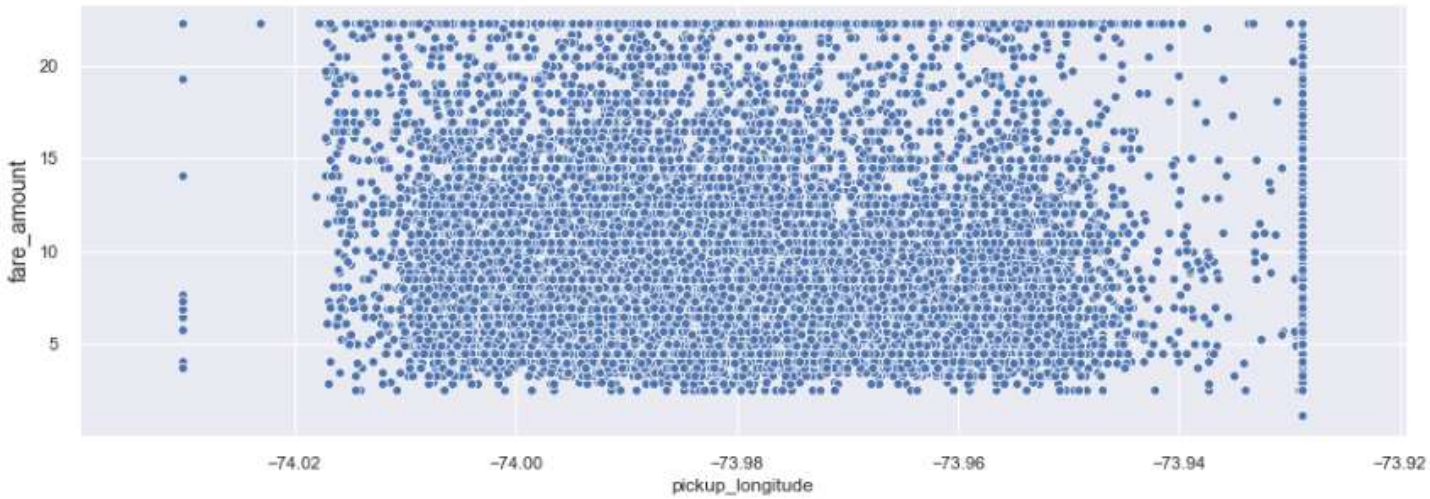
Scatter plot to understand overall distribution with respect to fare amount.




fare_amount
20
15
10
5
1
2
3
4
5
6
passenger_count


fare_amount
20
15
10
5
−74.04
−74.02
−74.00
−73.98
−73.96
−73.94
−73.92
dropoff_longitude


fare_amount
20
15
10
5
40.68
40.70
40.72
40.74
40.76
40.78
40.80
40.82
pickup_latitude


fare_amount
20
15
10
5
−74.02
−74.00
−73.98
−73.96
−73.94
−73.92
pickup_longitude

### 2.4.6 Feature Selection

Before creating a model, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. Hence, feature selection can help in reducing the time for computation of model as well as the complexity of the model. Also, few models require the independent variables to be free from multicollinear effect, hence it is needed to perform various procedures to ensure that the independent variables are not collinear.

Correlation analysis includes correlation matrix and correlation plot to find out significant continuous variables and we found that there is no multi-collinearity among the predictor variables.

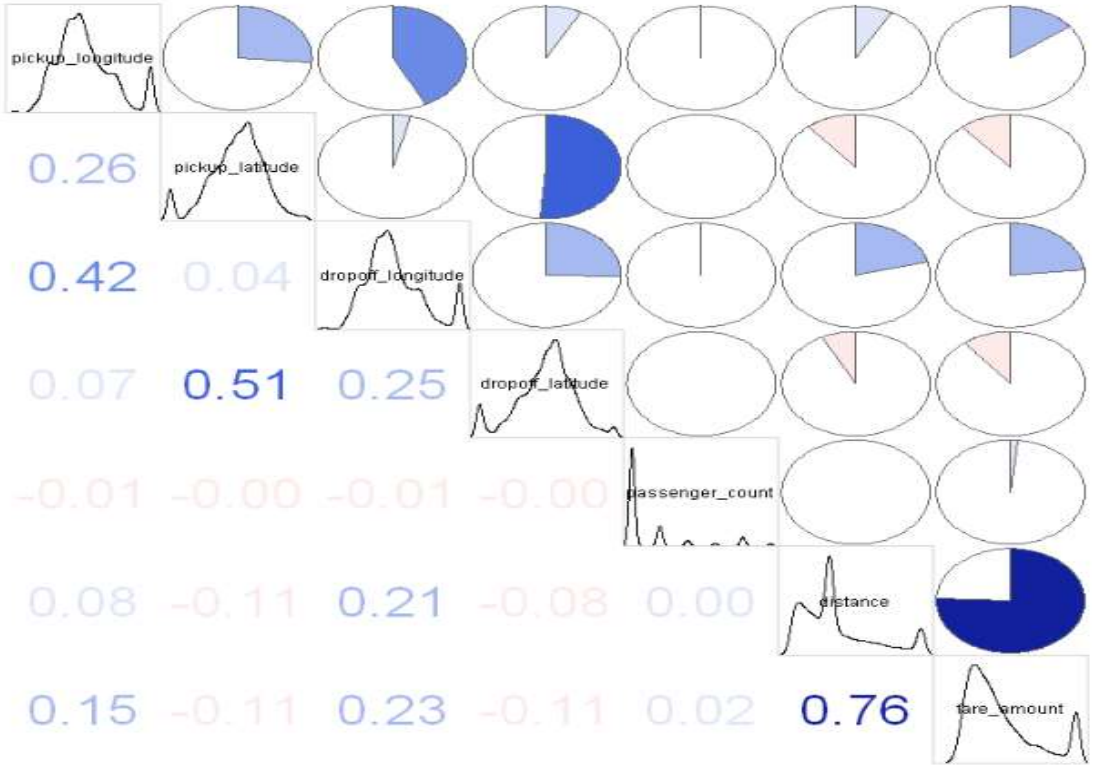
Using ANOVA test to find out significant categorical variable.

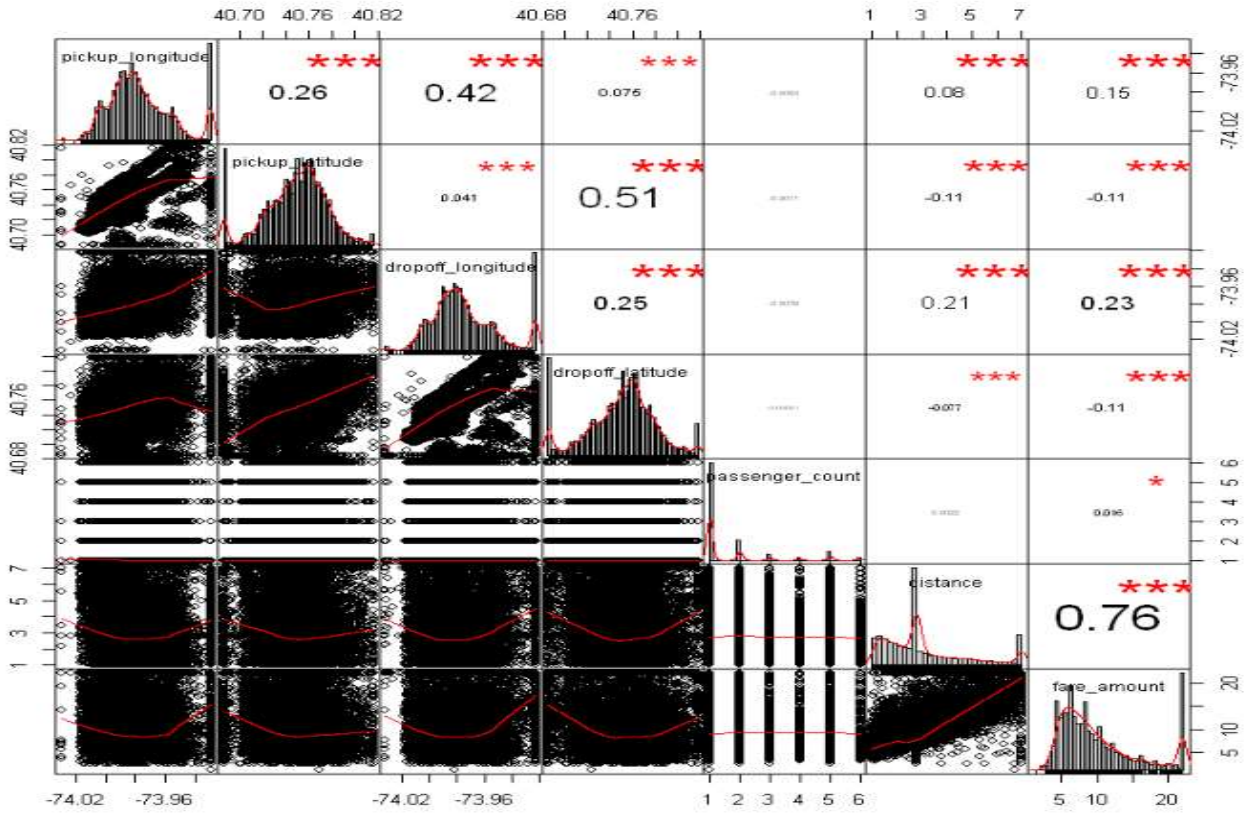
**In R:** Correlation matrix, Correlation Plot and ANOVA Test Results:

| | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | distance | fare_amount |
|---|---|---|---|---|---|---|---|
| pickup_longitude | 1.00 | 0.26 | 0.42 | 0.07 | -0.01 | 0.08 | 0.15 |
| pickup_latitude | 0.26 | 1.00 | 0.04 | 0.51 | 0.00 | -0.11 | -0.11 |
| dropoff_longitude | 0.42 | 0.04 | 1.00 | 0.25 | -0.01 | 0.21 | 0.23 |
| dropoff_latitude | 0.07 | 0.51 | 0.25 | 1.00 | 0.00 | -0.08 | -0.11 |
| passenger_count | -0.01 | 0.00 | -0.01 | 0.00 | 1.00 | 0.00 | 0.02 |
| distance | 0.08 | -0.11 | 0.21 | -0.08 | 0.00 | 1.00 | 0.76 |
| fare_amount | 0.15 | -0.11 | 0.23 | -0.11 | 0.02 | 0.76 | 1.00 |

From correlation matrix and correlation plot we can say distance variable is positively correlated with target variable which means this variable carries more information to predict the target variable. rest other variables also weekly correlated

## CORRELATION PLOT



## ANOVA Test Results in R:

```
[1] "month"
                  Df Sum Sq Mean Sq F value   Pr(>F)
Train_Cab[, i]     1    646   646.2   21.72 3.18e-06 ***
Residuals      15977 475345    29.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "year"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1   9932    9932   340.5 <2e-16 ***
Residuals      15977 466059      29
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "date"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1     30   29.94   1.005  0.316
Residuals      15977 475961   29.79
[1] "dayofweek"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1     41   40.74   1.368  0.242
Residuals      15977 475950   29.79
[1] "hour"
                  Df Sum Sq Mean Sq F value   Pr(>F)
Train_Cab[, i]     1    592   591.6   19.88 8.29e-06 ***
Residuals      15977 475400    29.8
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "minutes"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1    226  225.78   7.582 0.0059 **
Residuals      15977 475765   29.78
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "seconds"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1      6   6.028   0.202  0.653
Residuals      15977 475985  29.792
[1] "passenger_count"
                  Df Sum Sq Mean Sq F value Pr(>F)
Train_Cab[, i]     1    124  123.94   4.161 0.0414 *
Residuals      15977 475867   29.78
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

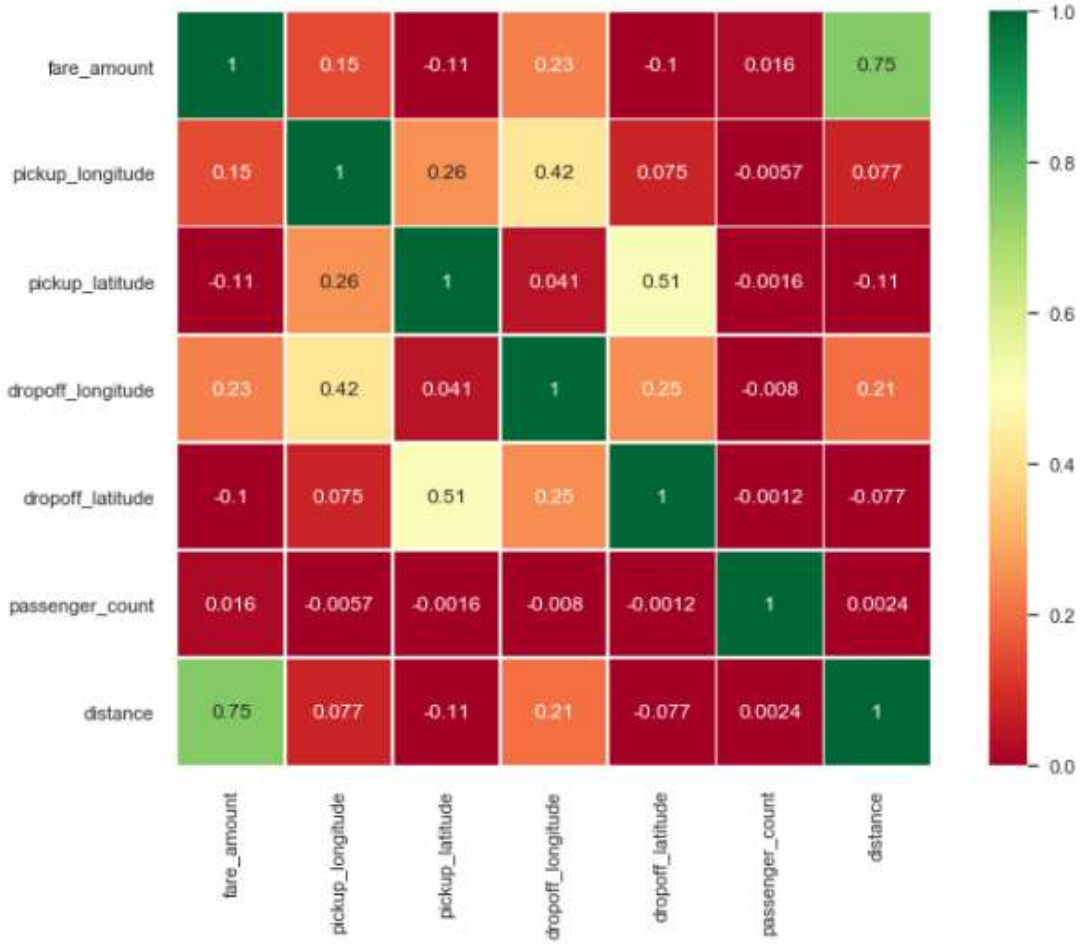
Removed date, dayofweek, minutes and seconds variables because whose p values>0.05 these variables are not important from prediction machine learning point of view.

**In Python:** Correlation matrix, Correlation Plot and ANOVA Test Results:

[127]:

| | fare_amount | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | distance |
|---|---|---|---|---|---|---|---|
| **fare_amount** | 1.000000 | 0.146506 | -0.112826 | 0.234120 | -0.104944 | 0.015670 | 0.751620 |
| **pickup_longitude** | 0.146506 | 1.000000 | 0.261399 | 0.422492 | 0.074568 | -0.005693 | 0.077231 |
| **pickup_latitude** | -0.112826 | 0.261399 | 1.000000 | 0.040601 | 0.513220 | -0.001619 | -0.107742 |
| **dropoff_longitude** | 0.234120 | 0.422492 | 0.040601 | 1.000000 | 0.252226 | -0.008022 | 0.205613 |
| **dropoff_latitude** | -0.104944 | 0.074568 | 0.513220 | 0.252226 | 1.000000 | -0.001190 | -0.077005 |
| **passenger_count** | 0.015670 | -0.005693 | -0.001619 | -0.008022 | -0.001190 | 1.000000 | 0.002419 |
| **distance** | 0.751620 | 0.077231 | -0.107742 | 0.205613 | -0.077005 | 0.002419 | 1.000000 |

From correlation matrix and correlation plot we can say distance variable is positively correlated with target variable which means this variable carries more information to predict the target variable. rest other variables also weekly correlated.




## **In Python: -**ANOVA Test Results:

```
                sum_sq       df           F        PR(>F)
Year       9877.628585      1.0  338.371895  8.539539e-75
Residual  466277.973805  15973.0         NaN           NaN
                sum_sq       df           F    PR(>F)
Month       646.459449      1.0  21.715454  0.000003
Residual  475509.142942  15973.0         NaN       NaN
                sum_sq       df         F    PR(>F)
Date         30.694737      1.0  1.029745  0.310234
Residual  476124.907653  15973.0       NaN       NaN
                 sum_sq       df         F    PR(>F)
Dayofweek    142.927862      1.0  4.796063  0.028539
Residual  476012.674528  15973.0       NaN       NaN
                sum_sq       df          F    PR(>F)
Hour        568.673059      1.0  19.099378  0.000012
Residual  475586.929332  15973.0         NaN       NaN
                sum_sq       df         F   PR(>F)
Minutes     230.488893      1.0  7.735669  0.00542
Residual  475925.113498  15973.0       NaN      NaN
                sum_sq       df         F    PR(>F)
Seconds       6.798192      1.0  0.228054  0.632977
Residual  476148.804198  15973.0       NaN       NaN
```

Removed date, dayofweek , minutes and seconds variables because whose p values>0.05 these variables are not important from prediction machine learning point of view.

### 2.4.6 Feature Scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. Widely used feature scaling methods are min max scaling and Standardization.

In car price prediction Project, the predictor variable distance is right skewed so by using log transformation technique we tried to reduce the skewness of this variable where Log of a variable is a common transformation method used to change the shape of distribution of the variable on a distribution plot. It is generally used for reducing right skewness of variables. Though, it can't be applied to zero or negative values as well. After log transformation of distance variable these now it appears to be almost normally distributed and rest all predictors are normally distributed, so we are not applying feature scaling techniques like normalization and standardization on our dataset. We can observe variables distribution as below plots

**In R :** Let's check the skewness of numeric variables

```
In [265]: # We can check using the function skewness from the e1071 library.
          skewness(Train_Cab$pickup_longitude)
          skewness(Train_Cab$dropoff_longitude)
          skewness(Train_Cab$pickup_latitude)
          skewness(Train_Cab$dropoff_latitude)
          skewness(Train_Cab$distance) # for distance Variable
          skewness(Train_Cab$fare_amount) # for fare_amount Variable
          skewness(Train_Cab$passenger_count) # for passenger_count
```

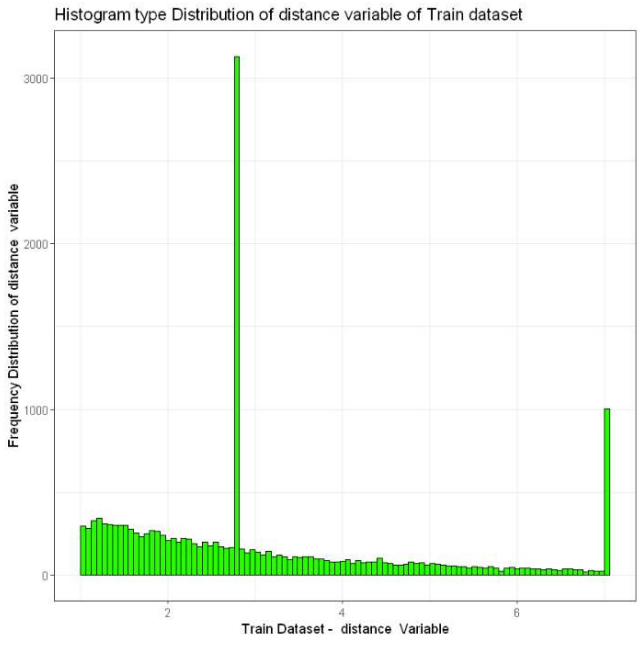
```
0.725905831326842
0.755593486813635
-0.430548506462288
-0.345116204898479
1.06432721119447
1.03438538386457
2.07747902715918
```

- If the skewness of the predictor variable is 0, the data is perfectly symmetrical,
- If the skewness of the predictor variable is less than -1 or greater than +1, the data is highly skewed,
- If the skewness of the predictor variable is between -1 and -0.5 or between +1 and +0.5 then the data is moderately skewed,
- If the skewness of the predictor variable is -0.5 and +0.5, the data is approximately symmetric.

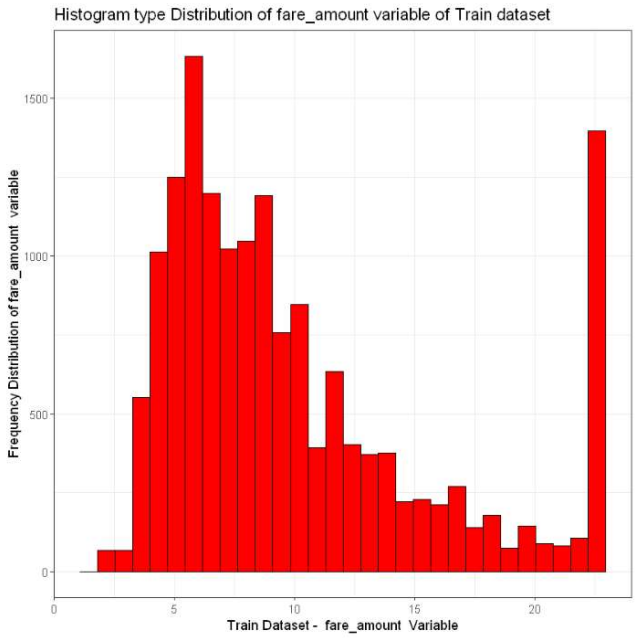
Skewed data have a negative impact on linear regression. For example,
if we take the scatter-plot of fare amount and distance variable we will see something very odd. Both are right skewed and correlated.

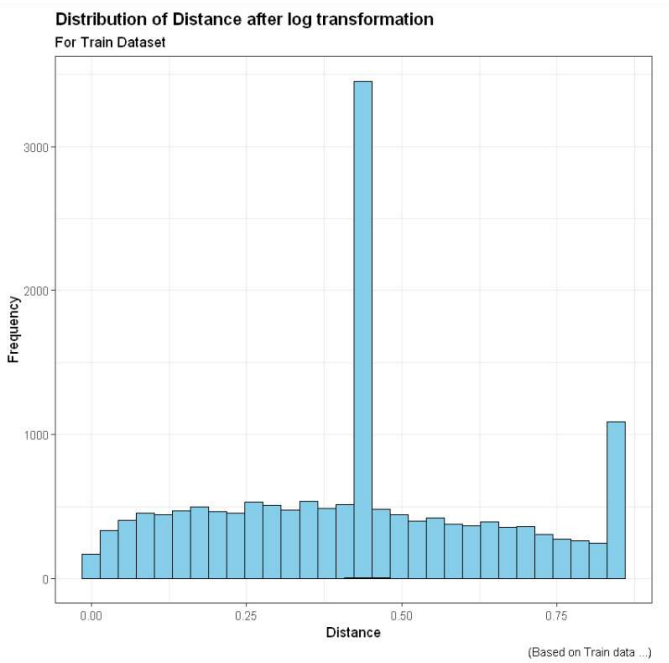
From Skewness values we found that:
distance (1.06), fare_amount (1.03) and passenger_count (2.076) are highly skewed.
pickup_latitude (-0.43), hour (-0.42), minutes (-0.011) and dropoff_latitude (-0.34) are moderately skewed.
pickup_longitude (0.72) and dropoff_longitude (0.75) are moderately skewed.
Let's apply log to fare amount and distance to reduce skewness in the data.

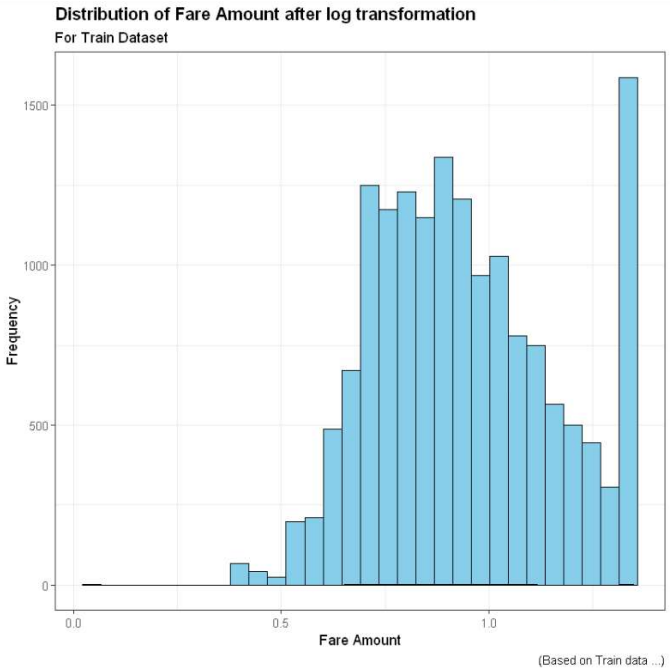
**In R : -**



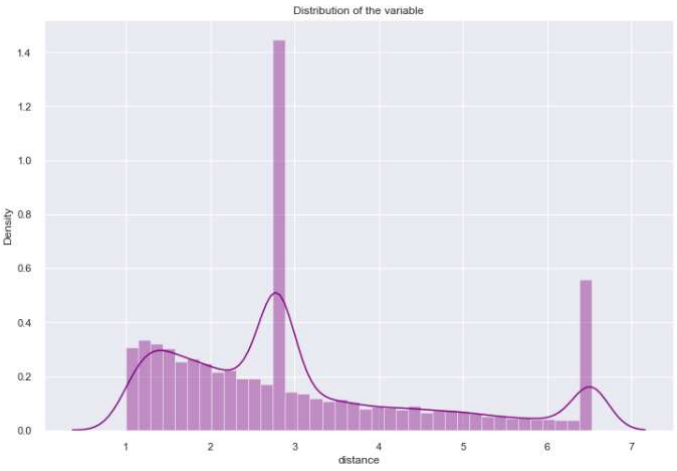

....

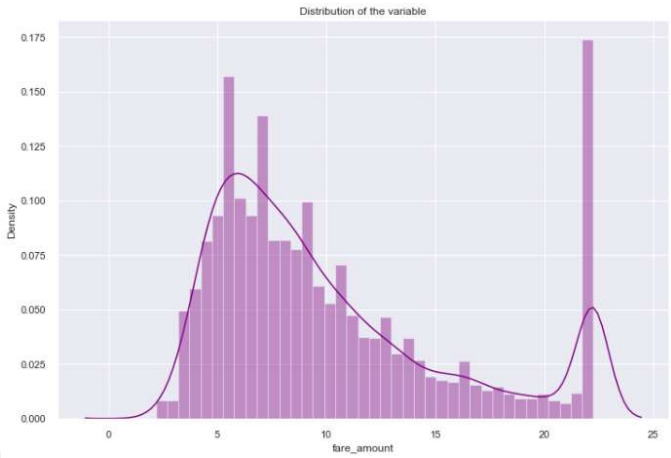


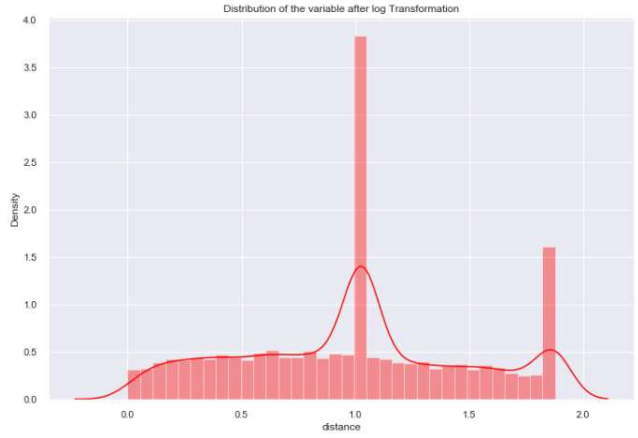


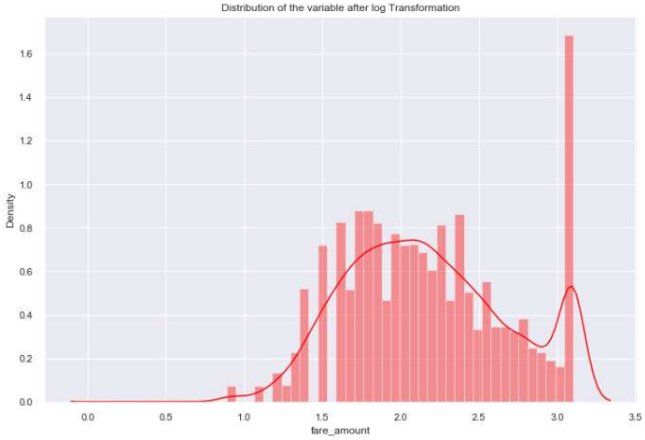




**In python:** fare amount and distance variable are highly skewed. We apply log to reduce skewness from the data.



## 2.5 Predictive Modeling:

**Predictive modeling** is a commonly used statistical technique to predict future behavior. Predictive modeling solutions are a form of data-mining technology that works by analyzing historical and current data and generating a model to help predict future outcomes. It's a technique that uses mathematical and computational methods to predict an event or outcome. A mathematical approach uses an equation-based model that describes the phenomenon under consideration. The model is used to forecast an outcome at some future state or time based upon changes to the model inputs. The model parameters help explain how model inputs influence the outcome.

Predictive modeling is a process that uses data mining and probability to forecast outcomes. Each model is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical model is formulated. The model may employ a simple linear equation, or it may be a complex neural network, mapped out by sophisticated software. As additional data becomes available, the statistical analysis model is validated or revised.

### 2.5.1 Model Selection

It is the task of selecting a statistical model from a set of candidate models, given data. In the simplest cases, a pre-existing set of data is considered. However, the task can also involve the design of experiments such that the data collected is well-suited to the problem of model selection.

Once completing data cleaned next process is model selection it is based on problem statement. In car fare prediction problem statement understood that it comes under supervised machine learning because it has both input and output variables and its regression problem as out target variable is fare amount which is of numeric /

continuous type. So, we can consider linear regression, Decision Tree, Random Forest etc.,

In our project used three models viz., linear regression, Decision Tree, Random Forest. Error matrix chosen for the given problem statement is Root Mean Squared Error (RMSE) and R2(R-Squared). Before building an any model we divided the preprocessed train_cab data set in to train and test set. Data was divided into 80:20 ratio, 80% of data was used as 'train' set and rest of the 20% was used as 'test' set. The training set is used to fit the model and the test set is used to estimate the model prediction accuracy.

### 2.5.2 Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.

The Formula for Multiple Linear Regression Is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

**where, for $i = n$ observations:**

$y_i$ = dependent variable

$x_i$ = expanatory variables

$\beta_0$ = y-intercept (constant term)

$\beta_p$ = slope coefficients for each explanatory variable

$\epsilon$ = the model's error term (also known as the residuals)

## Explaining Multiple Linear Regression

A simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables—an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables.
- The independent variables are not too highly correlated with each other.
- yi observations are selected independently and randomly from the population.
- Residuals should be normally distributed with a mean of 0 and variance $\sigma$.

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the independent variables. R2 always increases as more predictors are added to the MLR model even though the predictors may not be related to the outcome variable.

R2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

# Linear Regression model:

[164]:
```
# fit linear regression model we will use the lm() function from stats package:

linear_Reg_model <- lm(fare_amount ~.,data = train)
summary(linear_Reg_model)
```

```
Call:
lm(formula = fare_amount ~ ., data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-1.19259 -0.06071  0.02388  0.09841  0.68766

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)       59.3437823  7.4703760   7.944 2.12e-15 ***
pickup_longitude   0.3105142  0.0834412   3.721 0.000199 ***
pickup_latitude   -0.1509960  0.0710372  -2.126 0.033557 *
dropoff_longitude  0.6184102  0.0769474   8.037 1.00e-15 ***
dropoff_latitude  -0.4495789  0.0644667  -6.974 3.24e-12 ***
passenger_count    0.0024650  0.0011545   2.135 0.032778 *
year               0.0171488  0.0007897  21.717  < 2e-16 ***
month              0.0028240  0.0004276   6.605 4.14e-11 ***
hour               0.0002950  0.0002256   1.308 0.191005
distance           0.6260924  0.0067556  92.677  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1653 on 12776 degrees of freedom
Multiple R-squared:  0.4536,    Adjusted R-squared:  0.4532
F-statistic:  1178 on 9 and 12776 DF,  p-value: < 2.2e-16
```

Let's interpret the Outputs or coefficient of regression model:

Intercept: (b0) We can interpret intercept as when there is no impact of predictor variables on target variable then there is minimum expected target value (fare amount) here we got intercept around 59 which cannot be interpretable as the value is bit high.

Slope: (b1) One unit(1km) increase in the distance variable fare amount will increase by 0.626 units (Rs.) We can interpret hour, month, year, dropoff_lattitude and longitude as well but they don't make any sense so we are only interpreting distance variable.

**Adjusted R-squared: The** definition of adjusted R square same as R-square which says the proportion of total variability/variation in the target variable that is explained by its regression on the predictor variables when there are only significant variables in the

model. for our project total variation in the fare amount that is explained by its regression on the predictor variables is about 45%.

Let's check the assumptions of linear regression:

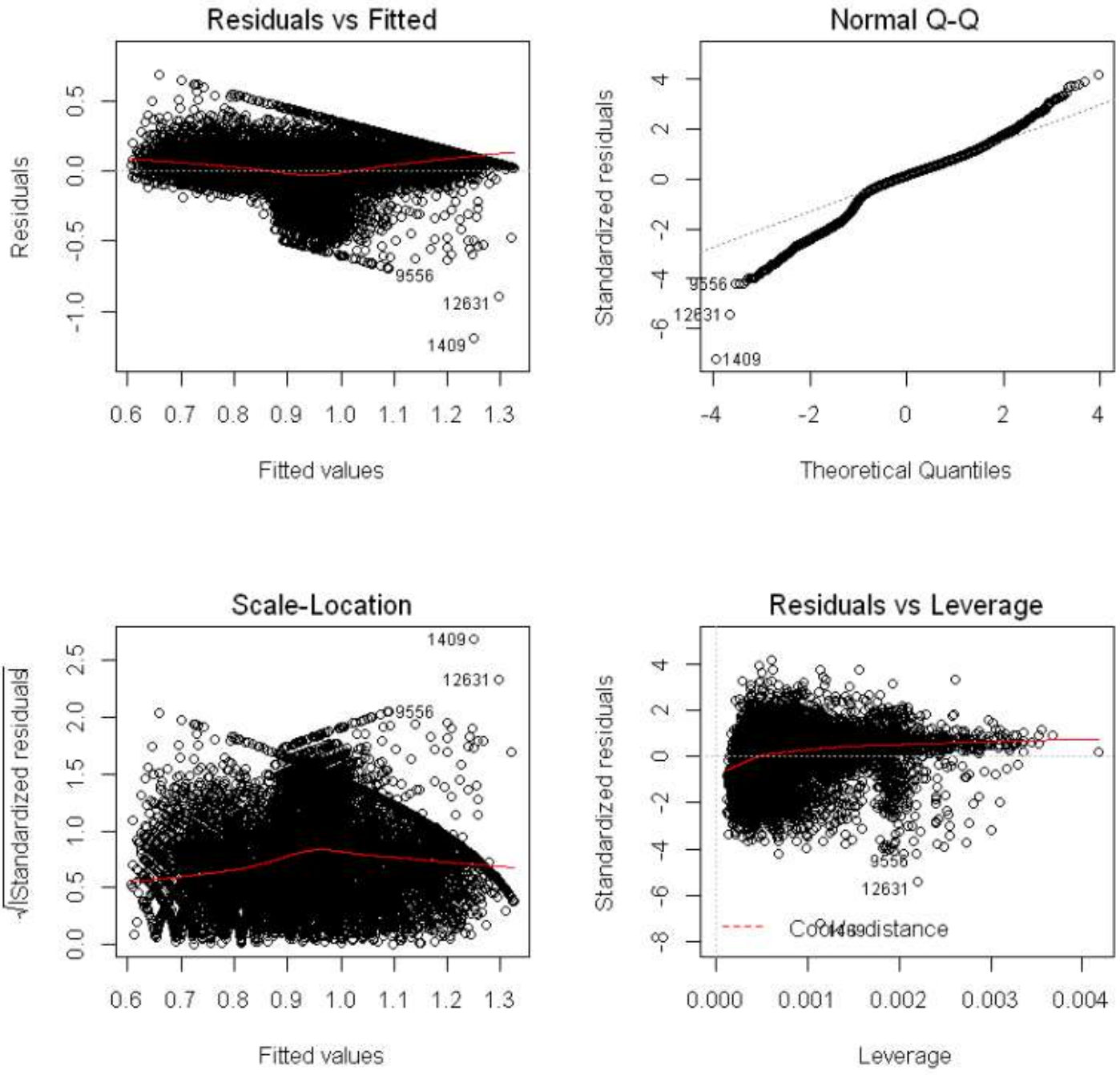
a) Error should follow normal distribution
b) Error should follow Homoscadacity or No Heteroscadacity
c) No multicollinearity among the independent variables
d) No serial / autocorrelation in error

**Error should follow normal distribution:**

To check this assumption, we plot normal qq plot. The normal qq plot helps us to determine if our target variable is normally distributed by plotting quantiles (i.e. percentiles) from our distribution against a theoretical distribution. If our data is normally distributed, it will be plotted in a generally straight line on the qq plot. We can also plot histogram where error should show a curvy bell shape for our project this assumption satisfied.

**Error should follow Homoscadacity or No Heteroscadacity :**

To check this assumption, we can use residual plot Residual plots help us evaluate and improve our regression model. A residual is the difference between the observed value of the dependent variable (y) and the predicted value (ŷ). A "good" residual vs. fitted plot should be relatively shapeless without clear patterns in the data, no obvious outliers, and be generally symmetrically distributed. For our Project this assumption is violated (residuals following a pattern not scattered) we can see it in Residuals vs fitted plot.




## No multicollinearity among the independent variables

To check this assumption, we are going to use Variance Inflation Factor (VIF) Variance inflation factor is a measure of the amount of multicollinearity in a set of multiple regression variables The Variance Inflation Factor (VIF) is 1/Tolerance, it is always greater than or equal to 1. There is no formal VIF value for determining presence of

multicollinearity. Values of VIF that exceeds 10 are often regarded as indicating multicollinearity, but in weaker models values above 2.5 may be a cause for concern for our project VIF values are within the range and this assumption is satisfied.

```
[120]: car::vif(model1)
```

| | |
|---|---|
| **pickup_longitude** | 1.36653617783186 |
| **pickup_latitude** | 1.54389474180515 |
| **dropoff_longitude** | 1.43410237576456 |
| **dropoff_latitude** | 1.53429473345079 |
| **passenger_count** | 1.04508038350928 |
| **year** | 1.02292957043643 |
| **month** | 1.01839624549393 |
| **date** | 1.00194699449427 |
| **dayofweek** | 1.00239516533831 |
| **hour** | 1.00929143969593 |
| **minutes** | 1.00094096785077 |
| **seconds** | 1.056070005135 |
| **distance** | 1.0790496800399 |

## No serial / autocorrelation in error:

Errors of all observation independent each other we can check this assumption using dwt test or Durbin Watson test The Durbin Watson (DW) statistic is a test for autocorrelation in the residuals from a statistical regression analysis. The Durbin - Watson statistic will always have a value between 0 and 4. A value of 2.0 means that there is no autocorrelation detected in the sample. Values from 0 to less than 2 indicate positive autocorrelation and values from from 2 to 4 indicate negative autocorrelation. So, our model is fine with this assumption.

```
[167]: # No auto-correlation between errors
       dwt(linear_Reg_model) # dwt < 2 so there is no autocorrelation in error
```

```
 lag Autocorrelation D-W Statistic p-value
   1      0.01031306      1.979325   0.224
Alternative hypothesis: rho != 0
```

Using linear regression for our project we have got MAPE value about 14.94% which says our model is only 85% accurate it means linear regression is not performing well on our dataset.

## In R :

```
      RMSE  Rsquared       MAE
0.1662967 0.4457562 0.1215490
```

```
[171]: MAPE = function(y, yhat){
         mean(abs((y - yhat)/y))
       }

       MAPE(test[,10],predictions_LR) # Fare amount variable at 10th position in train dataset.
```

0.149411374122453

We can say LM model is around 85.06 % accurate

## In Python:

| | MAPE | Accuracy |
|---|---|---|
| Linear Regression | 0.150202 | 84.979806 |
| Decision Tree | 0.123915 | 87.608502 |
| Random Forest | 0.093802 | 90.619772 |

## Reason of Accepting or Rejecting The model :

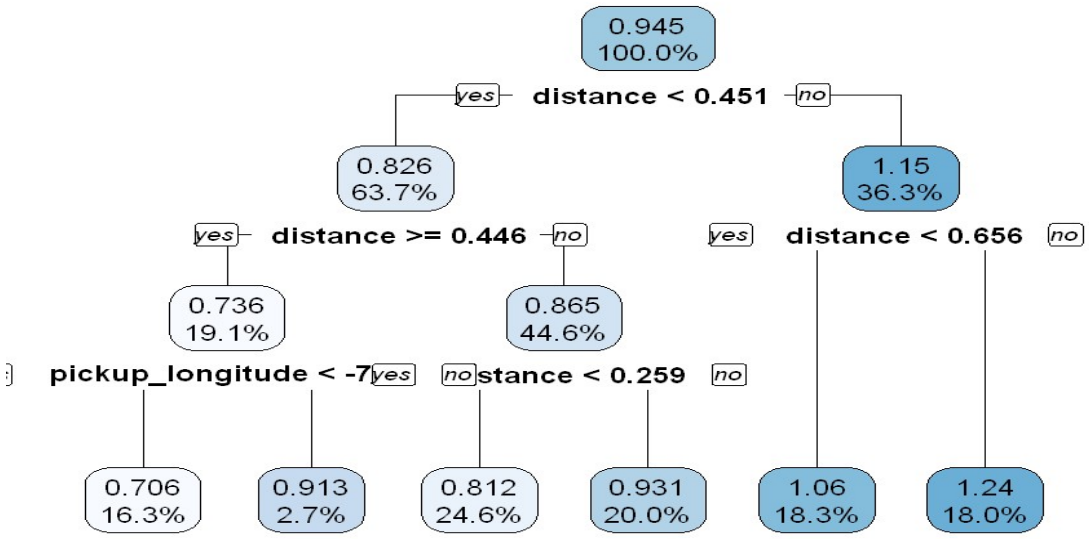
We are rejecting linear regression model because assumption of linear regression 2 is violated, RMSE is low and high adjusted R Square in train data around 45% and MAPE value is also less when applied trained model on test data which is around 14.9 % it means our linear regression model is 85% accurate.

### 2.5.3 Decision Tree

Decision Tree is a supervised machine learning algorithm, which is used to predict the data for classification and regression. It accepts both continuous and categorical variables. 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”. Extremely easy to understand by the business users. It provides its output in the form of rule, which can easily understand by a non -technical person also.

Output of Decision tree regression model is as below

```
n= 12786

node), split, n, deviance, yval
      * denotes terminal node

 1) root 12786 638.90640 0.9446758
   2) distance< 0.4506228 8147 213.45640 0.8262914
     4) distance>=0.4455466 2441  87.84938 0.7358079
       8) pickup_longitude< -73.93355 2090  51.95351 0.7060710 *
       9) pickup_longitude>=-73.93355 351  23.04306 0.9128737 *
     5) distance< 0.4455466 5706  97.07233 0.8649998
      10) distance< 0.2594092 3147  42.25801 0.8115122 *
      11) distance>=0.2594092 2559  34.73891 0.9307777 *
   3) distance>=0.4506228 4639 110.74980 1.1525820
     6) distance< 0.6559484 2334  37.03609 1.0628880 *
     7) distance>=0.6559484 2305  35.92278 1.2434050 *
```




Using Decision tree to predict fare amount we got MAPE value is around 11.39% which means our model is 88.61% accurate

```
      RMSE  Rsquared       MAE
0.1375506 0.6211007 0.1007961
```

```
[174]: MAPE(test[,10],predictions_DTR)
```

```
0.113916300263086
```

We can say our decision tree model around 88.61% accurate

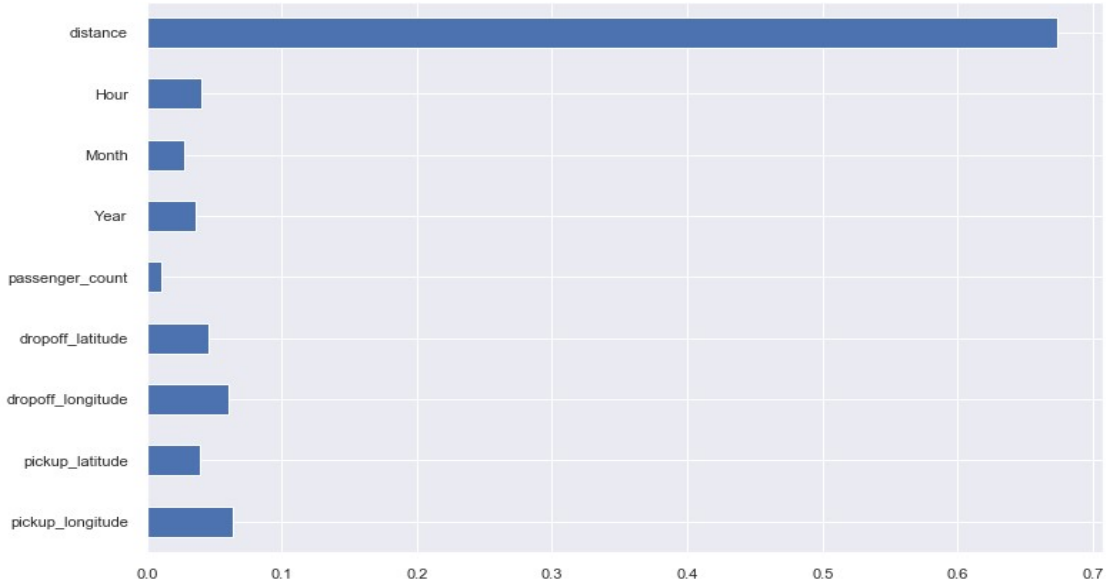
## Reason of Accepting or Rejecting The model :

We are rejecting Decision tree model because here also we can see low RMSE and High R-square value when applied train model on test data set

### 2.2.4 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 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. The Number of trees used in R and Python for random forest model are 200 no's

**Variable Importance of random forest model**




The above plot explains the which variable carries highest information to predict the target variable, the highest information is called feature importance, from the above plot distance, pickup_longitude, dropoff_lattitude, Hour and year have high feature importance

Using Random Forest model, MAPE value is around 9.20% which means our model is 90.797% accurate in R and 90.62%in Python.

**In R :**

```
      RMSE   Rsquared        MAE
0.11680419 0.72668167 0.07836754

[177]: MAPE(test[,10],RF_Predictions)

0.092036990087736
```

Random forest model score is 90.797% accurate

**In Python:**

| | MAPE | Accuracy |
|---|---|---|
| Linear Regression | 0.150202 | 84.979806 |
| Decision Tree | 0.123915 | 87.608502 |
| Random Forest | 0.093802 | 90.619772 |

**Reason of Accepting or Rejecting The model :**

We are accepting Random forest for our cab fare amount prediction because we have got optimum values of RMSE, R-square values compared to linear regression and decision tree. Our prediction accuracy has been increased from 85% to 90% using random Forest

### 3.1 Conclusion:

### 3.2 MAE, MSE, RMSE, R-squared, MAPE

We have calculated RMSE, R-Square, MAE (Mean Absolute Error) for all the three models.

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors) Residuals are a measure of how far from the regression line data points are, RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Whereas R squared is a relative measure of fit, 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 and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit.

R- squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-Squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. We also said about MAPE value for all the three models which is a measure of prediction accuracy of a forecasting method. It measures accuracy in terms of percentage

### 3.3 Model Selection :

**In R :**

| | RMSE | Rsquared | MAE | MAPE | Accuracy |
|---|---|---|---|---|---|
| **LinearRegression** | 0.1663 | 0.4457 | 0.1215 | 0.1494 | 85.06 |
| **DecisionTree** | 0.1375 | 0.6211 | 0.1000 | 0.1139 | 88.61 |
| **RandomForest** | 0.1170 | 0.7255 | 0.0780 | 0.0920 | 90.80 |

**In Python:**

| | RMSE | R2 | MAE | MAPE | Accuracy |
|---|---|---|---|---|---|
| Linear Regression | 0.385706 | 0.430023 | 0.280576 | 0.150202 | 84.979806 |
| Decision Tree | 0.332337 | 0.576842 | 0.250435 | 0.123915 | 87.608502 |
| Random Forest | 0.269567 | 0.721593 | 0.184424 | 0.093802 | 90.619772 |

Based on the above results, Random Forest is the better model for our analysis. Hence Random Forest is chosen as the model for Cab Fare Prediction.

## 3.4 Hyper parameter tuning:

Hyper parameter tuning in R for Random forest model by random search CV and grid search CV but accuracy was marginally improved much.

In random search CV method accuracy was improved from 90.797% to 90.828%

Whereas,

In grid search CV method accuracy was improved from 90.797% to 90.838%.

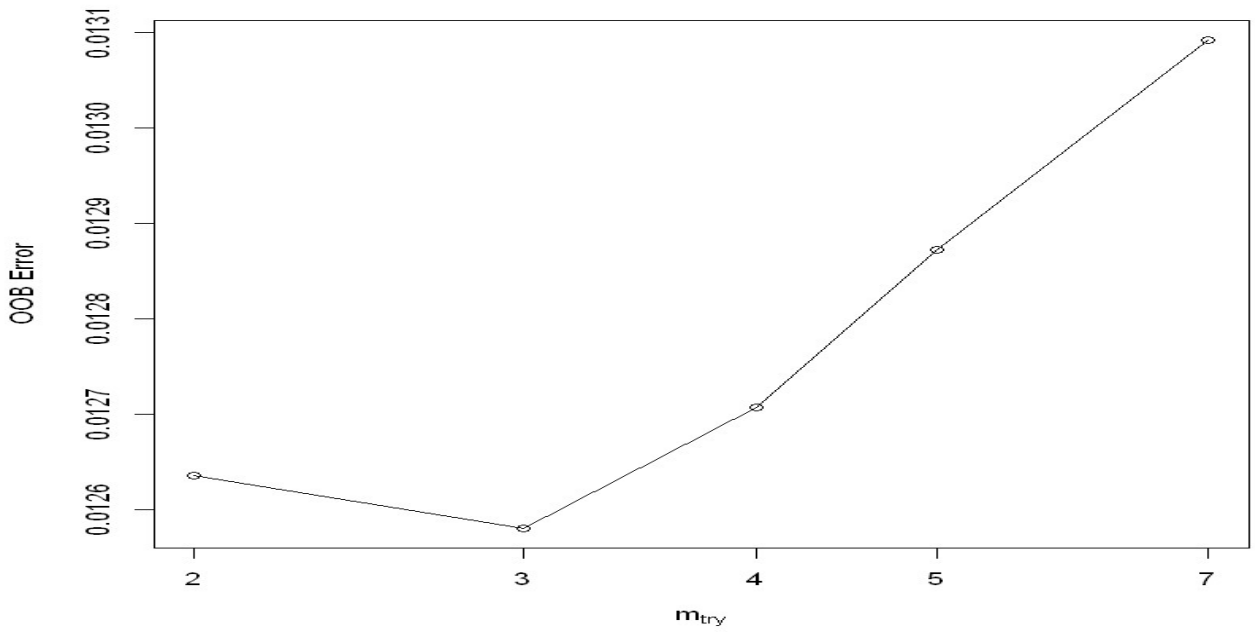
Below are the results in python:

## In R :

## Random Search CV on Random Forest Model:

```
Random Forest

12786 samples
    9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 1 times)
Summary of sample sizes: 10228, 10230, 10229, 10228, 10229
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  1     0.1272094  0.7088315  0.09651742
  2     0.1129118  0.7464565  0.07884066
  3     0.1124103  0.7474289  0.07758618
  4     0.1128979  0.7450069  0.07777481
  6     0.1138469  0.7406502  0.07843967
  7     0.1142506  0.7388352  0.07872723
  9     0.1150487  0.7352127  0.07918018

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
  mtry
3    3
```

```
0.01284446 0.01
0.01001024 0.01
-0.004440104 0.01
-0.04073264 0.01
```




```
[1] " Random Search CV Random Forest Regressor Model Performance:"
[1] " RMSE : 0.116427923735064"
[1] " R-Squared : 0.728434848415654"
[1] " MAPE : 0.0917121687205551"
[1] " Accuracy : 90.8287831279445"
```

**Accuracy: 1-0.0917 =90.828%**

## Grid Search CV on Random Forest Model:

```
Random Forest

12786 samples
    9 predictor

No pre-processing
Resampling: Cross-Validated (5 fold, repeated 2 times)
Summary of sample sizes: 10228, 10229, 10229, 10229, 10229, 10229, ...
Resampling results across tuning parameters:

  mtry  RMSE       Rsquared   MAE
  1     0.1271497  0.7097970  0.09641003
  2     0.1129073  0.7465367  0.07882519
  3     0.1125075  0.7470130  0.07759605
  4     0.1129432  0.7448438  0.07770906
  5     0.1134693  0.7423958  0.07806360
  6     0.1139298  0.7403022  0.07837103
  7     0.1143549  0.7383569  0.07868762
  8     0.1148173  0.7362486  0.07899499

RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 3.
  mtry
3    3
[1] " Grid Search CV Random Forest Regressor Model Performance:"
[1] " RMSE : 0.116633652806088"
[1] " R-Squared : 0.727450272622753"
[1] " MAPE : 0.0916149788499397"
[1] " Accuracy : 90.838502115006"
```

**Accuracy: 1-0.09161 =90.838%**

In Python, random search CV method accuracy was improved from 90.62% to 90.67%

Whereas,

In grid search CV method accuracy was improved from 90.62% to 90.66%.

Below are the results in python:

## In Python:

## Random Search CV on Random Forest Model:

```
Random Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'n_estimators': 15, 'max_depth': 9}
RMSE =  0.2674251221867559
R-squared = 0.73.
MAE =  0.18461174868779545
MAPE =  0.09337670767878548
```

**Accuracy: 1-0.0933 =90.67%**

## Grid Search CV on Random Forest Model:

```
Grid Search CV Random Forest Regressor Model Performance:
Best Parameters =  {'max_depth': 9, 'n_estimators': 19}
RMSE =  0.2673829013395307
R-squared = 0.73.
MAE =  0.18470311659828748
MAPE =  0.0934523607130628
```

**Accuracy: 1-0.0933 =90.66%**

## 3.2 Selected Model Results:

Predicted fare amount using Random Forest Model both R and Python.

### In R:

| | RMSE | Rsquared | MAE | MAPE | Accuracy |
|---|---|---|---|---|---|
| **LinearRegression** | 0.1663 | 0.4457 | 0.1215 | 0.1494 | 85.06 |
| **DecisionTree** | 0.1375 | 0.6211 | 0.1000 | 0.1139 | 88.61 |
| **RandomForest** | 0.1170 | 0.7255 | 0.0780 | 0.0920 | 90.80 |

Grid-search = 90.835%

```
head(Test_Cab)
```

| pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | year | month | hour | distance | Predicted_fare_amount |
|---|---|---|---|---|---|---|---|---|---|
| -73.97332 | 40.76381 | -73.98143 | 40.74384 | 1 | 2015 | 1 | 13 | 2.325862 | 9.60 |
| -73.98686 | 40.71938 | -73.99889 | 40.73920 | 1 | 2015 | 1 | 13 | 2.428070 | 9.37 |
| -73.98252 | 40.75126 | -73.97965 | 40.74614 | 1 | 2011 | 10 | 11 | 2.887776 | 9.19 |
| -73.98116 | 40.76781 | -73.99045 | 40.75164 | 1 | 2012 | 12 | 21 | 1.963229 | 9.23 |
| -73.96605 | 40.78977 | -73.98856 | 40.74443 | 1 | 2012 | 12 | 21 | 5.393336 | 15.12 |
| -73.96098 | 40.76555 | -73.97918 | 40.74005 | 1 | 2012 | 12 | 21 | 3.226159 | 9.99 |

### In Python:

| | RMSE | R2 | MAE | MAPE | Accuracy |
|---|---|---|---|---|---|
| Linear Regression | 0.385706 | 0.430023 | 0.280576 | 0.150202 | 84.979806 |
| Decision Tree | 0.332337 | 0.576842 | 0.250435 | 0.123915 | 87.608502 |
| Random Forest | 0.269567 | 0.721593 | 0.184424 | 0.093802 | 90.619772 |

Out[198]:

| | pickup_longitude | pickup_latitude | dropoff_longitude | dropoff_latitude | passenger_count | Year | Month | Hour | distance | Predicted_Fare_Amount |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -73.973320 | 40.763805 | -73.981430 | 40.743835 | 1 | 2015.0 | 1.0 | 13.0 | 2.323259 | 9.639143 |
| 1 | -73.986862 | 40.719383 | -73.998886 | 40.739201 | 1 | 2015.0 | 1.0 | 13.0 | 2.425353 | 9.935656 |
| 2 | -73.982524 | 40.751260 | -73.979654 | 40.746139 | 1 | 2011.0 | 10.0 | 11.0 | 2.884545 | 8.904876 |
| 3 | -73.981160 | 40.767807 | -73.990448 | 40.751635 | 1 | 2012.0 | 12.0 | 21.0 | 1.961033 | 9.326688 |
| 4 | -73.966046 | 40.789775 | -73.988565 | 40.744427 | 1 | 2012.0 | 12.0 | 21.0 | 5.387301 | 15.332295 |

## 4.1 Brief Insights about the Cab Fare prediction Project

Cabs make it easy to drive in and around the city, especially if we need a vehicle for short timelines, or if we are travelling to a different city. It brings in a lot of freedom and convenience. From this project we analyzed theee are key important points where cab fare is influenced.

Distance variable has more influence on fare amount longer the distance higher will be the fare amount, these two are directly proportional to each other

Year variable tells us it's based on business cab fare increase or decreasing over a period of time with respect to competition in the market like in the year 2013 fare amount was very high and in the year 2015 it was low

Hour variable an import key variable, in the peak hours 6 pm to 10 pm we can expect more demand more fare as well and less demand hence lower fare at 3 am to 5am

Month variable affects fare due to occasions make fare more for example festivals Holidays summer vacations etc. might leads more travelling more demand more cab. Requirement. Observed that in march to may month fare amount has more demand and more fare amount for a cab ride.

Day doesn't much affect the fare but it definitely influences to some extent but through visualization we can say in weekends like Friday and Saturday fare for cab ride will be more. Also midweek of the month has more no. of rides were taken.

Coordinates like pickup and dropoff laongitudes and latitudes doesn't make any influence on fare

Passenger count not influence on fare amount but we can say single passengers are frequent travelers by looking at our visualization /distribution plot

## 4.2 Appendix A – R and Python Scripts and Notebooks


Cab_Fare_Prediction_Python.py


Cab_Fare_Prediction_R.r


Cab_Fare_Prediction_Python.ipynb


Cab_Fare_Prediction_R.ipynb

## 5 Appendix B – References:


1. https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/
2. https://trainingdatascience.com/workshops/histograms-and-skewed-data/
3. Date-time conversion Reference Blog: https://www.displayr.com/r-date-conversion/
4. For Missing values Referred blog: https://www.analyticsvidhya.com/blog/2016/03/tutorial-powerful-packages-imputing-missing-values/
5. https://medium.com/coinmonks/dealing-with-missing-data-using-r-3ae428da2d17
6. Reference Blog: https://ggplot2.tidyverse.org/reference/labs.html
7. For more information on plots Referred Blog: https://cran.r-project.org/web/packages/corrgram/vignettes/corrgram_examples.html
8. Reference Blog link: http://rpubs.com/marvinlemos/log-transformation
9. Referred Blog: https://www.r-bloggers.com/log-transformations-for-skewed-and-wide-distributions-from-practical-data-science-with-r/
10. https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/
11. For parameter Tuning https://uc-r.github.io/random_forests