Cab Fare Prediction

SHIVAM MISHRA

29 August 2019

Contents

4.3 Visualization on the count of passengers

4.4 Visualization on distribution of trip_distance

1. INTRODUCTION

1.1 Problem statement

There is a cab rental start-up company which wants to launch a cab service. They have successfully run the pilot project and now want to launch your cab service ac!ross the country. They have collected the historical data from their pilot project and now have a requirement to apply analytics for fare prediction. The aim of the project is to design a system that predicts the fare amount for a cab ride in the city.

1.2 Data

Our task is to build models that will predict the fare of the cab depending on certain conditions like location, time and weekdays, etc. Given below is a sample of the data set that we will use to predict the fare amount of cabs :

Table 1.1: Sample Data (Columns: 1-5)

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude
4.5	2009-06-15 17:26:21 UTC	-73.8443	40.72132	-73.8416
16.9	2010-01-05 16:52:16 UTC	-74.016	40.7113	-73.9793
5.7	2011-08-18 00:35:00 UTC	-73.9827	40.76127	-73.9912
7.7	2012-04-21 04:30:42 UTC	-73.9871	40.73314	-73.9916
5.3	2010-03-09 07:51:00 UTC	-73.9681	40.76801	-73.9567

Table 1.2: Sample Data (Columns: 6-7)

dropoff_latitude	passenger_count
40.71228	1
40.782	1
40.75056	2
40.75809	1
40.78376	1

Attributes present in the dataset are fare_amount, pickup_datetime, pickup_longitude, pickup_latitude, dropoff_longitude, dropoff_latitude and passenger_count.

The details of dataset attributes are as follows -

- pickup_datetime - timestamp value indicating when the cab ride started.
- pickup_longitude - float for longitude coordinate of where the cab ride started.
- pickup_latitude - float for latitude coordinate of where the cab ride started.
- dropoff_longitude - float for longitude coordinate of where the cab ride ended.
- dropoff_latitude - float for latitude coordinate of where the cab ride ended.
- passenger_count - an integer indicating the number of passengers in the cab

Now let's have a look at the data type of dataset attributes.

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

Here, the datatype of fare_amount attribute is an object which is not correct. So we converted this attribute into numeric. But, while converting it to numeric we found a problem that it contains a string value "-430" at location 1123. So we basically replaced this value with 430 and then converted it to a numeric datatype. Also passenger_count variable has datatype float so once again we will convert it to object or factor datatype.

2. Methodology

2.1 Pre Processing

Before developing any model we first need to look into the data. By, saying look into the data, I mean to explore the data. Look at the datatypes of attributes, find the minimum and maximum values of variables compare it with its mean value. Convert the required datatypes. Finding and imputing missing values using various methods like mean, median, etc. This is nothing but data preprocessing where we analyze the data, clean the data and transform the data. Data preprocessing is the probably most or one of the most important things in model development. So, it needs to be taken care of.

2.1.1 Missing Value Analysis

Missing value analysis is a method or technique to find out if there are missing values in the attributes of the dataset. When we applied missing value analysis on our dataset we found that passenger_count had most numbers of missing values followed by fare_amount. Except for these two variables, there were no missing values in any other variables. Missing values can be found by using these syntaxes

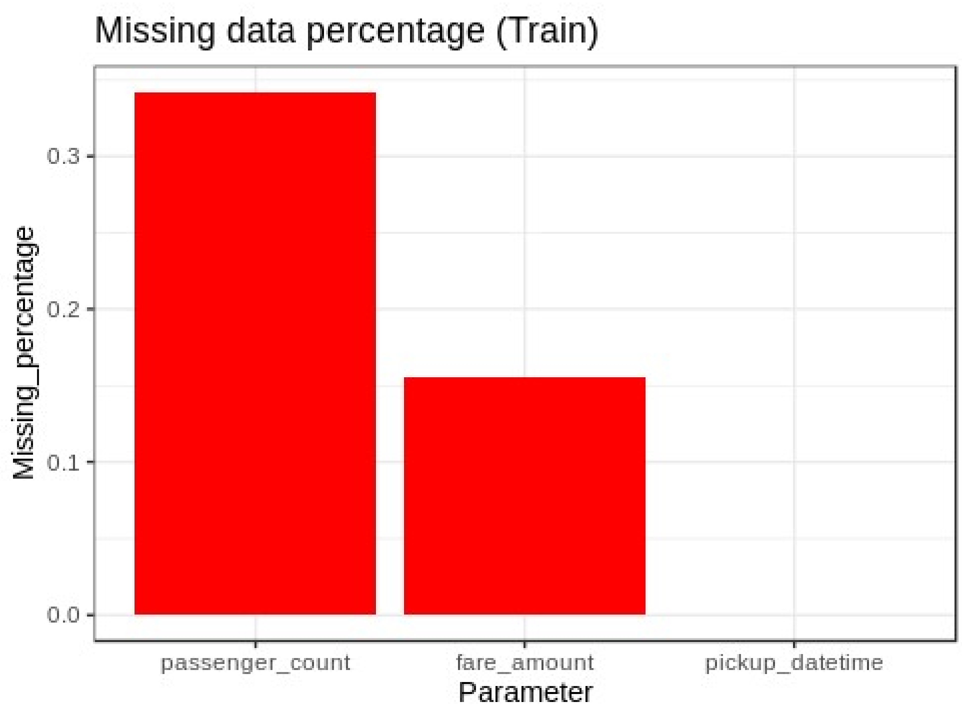
- is.null( ).sum() in python
- function(x){sum(is.na(x))} in R

We got the following result after applying missing value analysis on our dataset

	Variables	Missing_percentage
0	passenger_count	0.342317
1	fare_amount	0.149374
2	pickup_datetime	0.000000
3	pickup_longitude	0.000000
4	pickup_latitude	0.000000
5	dropoff_longitude	0.000000
6	dropoff_latitude	0.000000

Now, you can see that passenger_count has 0.342317% missing values which is obviously much small. Similarly, fare_amount has got 0.149374% missing values otherwise there were no missing values in the dataset.
As we know that passenger_count is a categorical variable we used mode method to impute missing values of this variable.
Similarly, we computed the mean and median for fare_amount as it is a numeric variable. After applying these methods we found out that mean was giving more

accurate value compared to the median. So, we decided to go with the mean method for computing the missing values of the fare_amount variable.
Let's look at the visualization of missing values.



2.1.2 Outlier Analysis

In statistics, an outlier is defined as a data point that differs significantly from other observations. Outlier analysis is a technique to find these points. Outlier analysis can only be done on a numerical variable.
Causes of Outliers

- Poor data quality/contamination
- Low-quality measurements, malfunctioning equipment, manual error
- Correct but exceptional data

In our case, we first analyzed location variables i.e latitude and longitude. As we know the fare of the cab may change from location to location hence we considered all the locations of train dataset that were outside the locations of test dataset as outlier locations. And then we removed these locations from the train dataset, for example, look at the following R code.

```
#longitude boundary
min(test$pickup_longitude, test$dropoff_longitude)
max(test$pickup_longitude, test$dropoff_longitude)
```

longitude boundary=(-74.26324,-72.98653)

```
#latitude boundary
min(test$pickup_latitude, test$dropoff_latitude)
max(test$pickup_latitude, test$dropoff_latitude)
#latitude boundary=(40.56897,41.70956)

#set boundaries
min_longitude=-74.26324
min_latitude=40.56897
max_longitude=-72.98653
max_latitude=41.70956

train=subset(train, pickup_longitude >= min_longitude)
train=subset(train,pickup_longitude <= max_longitude)
train=subset(train,pickup_latitude >= min_latitude)
train=subset(train,pickup_latitude<=max_latitude)

train=subset(train, dropoff_longitude >= min_longitude)
train=subset(train,dropoff_longitude <= max_longitude)
train=subset(train,dropoff_latitude >= min_latitude)
train=subset(train,dropoff_latitude<=max_latitude)
```

So, using these lines of code we removed outlier locations from the train dataset.

Now, we analyzed fare_amount and saw significantly larger values than mean. For better understanding let's look at the summary of fare_amount.

```
count    16043.000000
mean        15.040871
std        430.459997
min         -3.000000
25%          6.000000
50%          8.500000
75%         12.500000
max      54343.000000
Name: fare_amount, dtype: float64
```

So, we can see from the above summary that the maximum value of fare_amount i.e 54343 is way larger than the mean value which is 15.044. Similarly, if you look at the minimum value in fare_amount is negative which is not possible. So, we considered all the value below 1 and above 150 in the fare_amoubt variable as an outlier and removed it from the dataset.

Similarly, if you will look at the summary of passenger_count you will find that the maximum value in passenger count is huge. Let's have a look at the summary of the passenger_count.

```
count    16012.000000
mean         2.625070
std         60.844122
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max       5345.000000
Name: passenger_count, dtype: float64
```

As you can see maximum passenger_count is 5345 which is not possible as probably no cab in this world can take these many passengers at once. Similarly, if you look at the minimum value of passenger_count you will find that the minimum value is 0 which is of no use. So after analyzing passenger_count we decided to drop all the observations having count below 1 and above 6. Now when we look at the unique values in passenger_count we found an odd value.

```
1.0    11051
2.0     2281
5.0     1023
3.0      662
4.0      320
6.0      295
1.3        1
Name: passenger_count, dtype: int64
```

As you can see there is one unique value as 1.3 which is not possible as there can not be 1.3 passengers.

Now, we analyzed the summary of the trip_distance which we extracted from longitude and latitude data and will discuss more it later in the Feature Engineering part. Now we found minimum trip_distance value as zero. So, we decided to drop all the observations having trip_distance value below 0.2, as most of the people do not prefer a cab for distance below 200 meters.

2.1.3 Feature Engineering

Feature Engineering is used to extract important or valuable features from the data. In our case, we had a timestamp attribute pickup_datetime which will be of no use if we don't extract important features from it. We extracted many important features like day, year, month, weekday_names, hour from this timestamp variable. To do so we used "pd.to_datetime" in python and "as.Date" in R.

Similarly, we calculated trip_distance from pickup and dropoff latitudes and longitudes. To calculate this distance we used Haversine distance formula. For example, look at the below python code

```
def trip_distance(lon1, lat1, lon2, lat2):
    lon1, lat1, lon2, lat2 = map(np.radians, [lon1, lat1, lon2, lat2])
    dlon = lon2 - lon1
    dlat = lat2 – lat1
    a = np.sin(dlat/2.0)**2 + np.cos(lat1) * np.cos(lat2) * np.sin(dlon/2.0)**2
    c = 2 * np.arcsin(np.sqrt(a))
    km = 6371 * c
    return km
```

Here, we have used np.radians to convert latitude and longitudes into radian. And from 2nd last line to 4th last line is the Haversine formula. Where 6371 is nothing but the radius of the earth in kilometers.

2.1.4 Feature Selection

Feature selection is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. In machine learning and statistics feature selection is also known as variable selection, attribute selection or variable subset selection. In our dataset pickup_datetime which is a timestamp variable is of no use hence dropped this column from our train and test dataset. And selected the rest of the features.

2.2 Modeling

2.2.1 Model Selection

As we know that we need to predict the fare of the cab we can understand the problem category that this is a forecasting problem. So, we knew that we have to use regression models to predict the target variable. Hence we used the following regression models to predict the result.

- Linear Regression
- Decision Tree
- Random Forest

2.2.2 Multiple Linear Regression

model1 = sm.OLS(y_train, X_train).fit( )
model1.summary( )

OLS Regression Results

Dep. Variable:	fare_amount	**R-squared:**	0.865
Model:	OLS	**Adj. R-squared:**	0.865
Method:	Least Squares	**F-statistic:**	7130.
Date:	Tue, 27 Aug 2019	**Prob (F-statistic):**	0.00
Time:	22:11:31	**Log-Likelihood:**	-38071.
No. Observations:	12272	**AIC:**	7.616e+04
Df Residuals:	12261	**BIC:**	7.625e+04
Df Model:	11		
Covariance Type:	nonrobust		

	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
pickup_longitude	-13.2237	1.227	-10.773	0.000	-15.630	-10.818
pickup_latitude	-43.4709	1.719	-25.287	0.000	-46.841	-40.101
dropoff_longitude	3.7004	1.223	3.025	0.002	1.303	6.098
dropoff_latitude	-6.2047	1.620	-3.830	0.000	-9.380	-3.029
passenger_count	0.0510	0.039	1.321	0.186	-0.025	0.127
day	-0.0002	0.006	-0.044	0.965	-0.011	0.011
year	0.6583	0.025	26.013	0.000	0.609	0.708
month	0.1071	0.014	7.557	0.000	0.079	0.135
hour	0.0059	0.008	0.780	0.436	-0.009	0.021
weekday	-0.0897	0.025	-3.610	0.000	-0.138	-0.041
trip_distance	1.8394	0.013	145.320	0.000	1.815	1.864

Omnibus:	15533.389	**Durbin-Watson:**	1.993
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	32449063.719
Skew:	-6.008	**Prob(JB):**	0.00
Kurtosis:	254.625	**Cond. No.**	8.69e+04

As you can see from the adjusted R-squared value, we can explain nearly 86.5 % of our data using our linear model. Which is quite good. Now if you look at the p-value then we can say that the null hypothesis for passenger_count, day and hour is true.

```
###
X_train1,X_test1,y_train1,y_test1=modeling(train,'fare_amount',drop_cols=['pickup_datetime'],is_train=True,split=0.3)

### model2 = sm.OLS(y_train1, X_train1).fit( )

### model2.summary( )
```

OLS Regression Results

Dep. Variable:	fare_amount	**R-squared:**	0.862
Model:	OLS	**Adj. R-squared:**	0.862
Method:	Least Squares	**F-statistic:**	6077.
Date:	Tue, 27 Aug 2019	**Prob (F-statistic):**	0.00
Time:	22:15:39	**Log-Likelihood:**	-33384.
No. Observations:	10738	**AIC:**	6.679e+04
Df Residuals:	10727	**BIC:**	6.687e+04
Df Model:	11		
Covariance Type:	nonrobust		

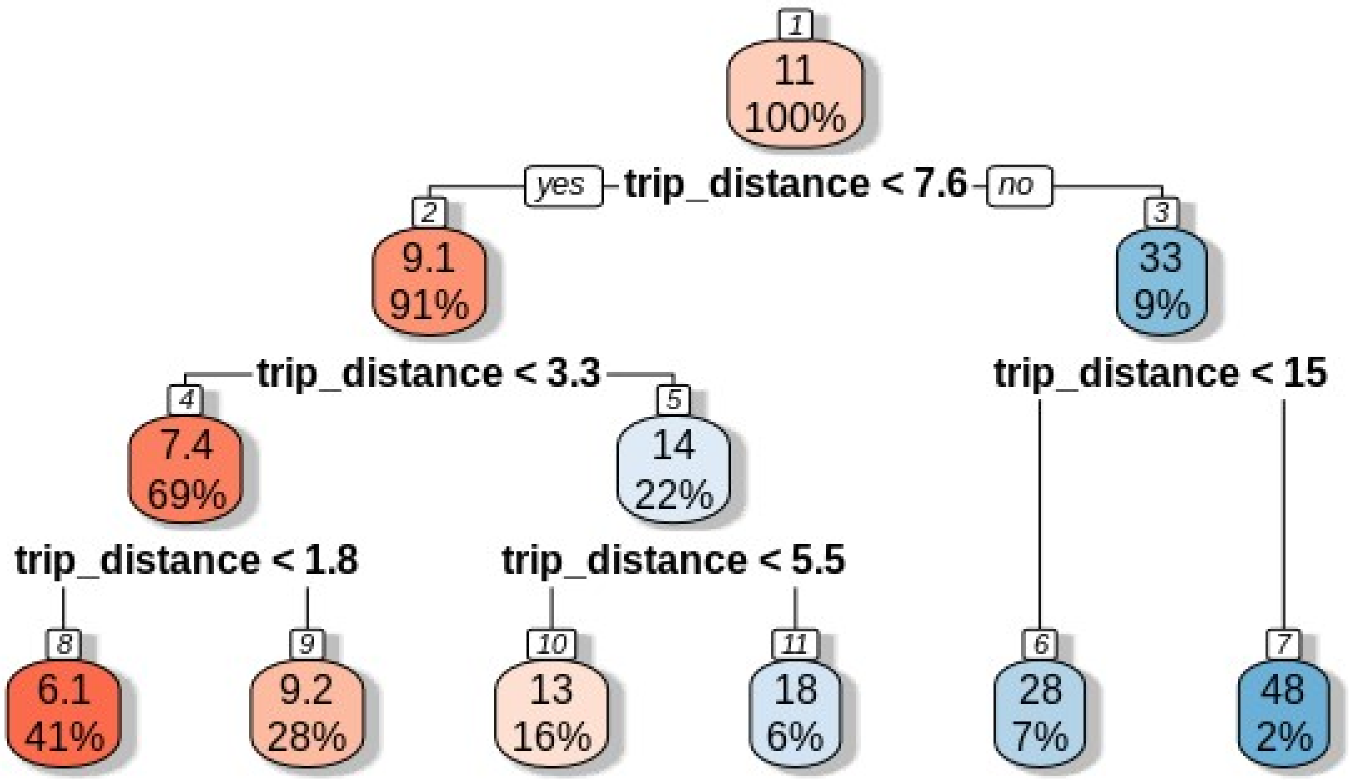
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
pickup_longitude	-13.3092	1.315	-10.125	0.000	-15.886	-10.732
pickup_latitude	-49.1592	1.832	-26.833	0.000	-52.750	-45.568
dropoff_longitude	2.1519	1.315	1.636	0.102	-0.426	4.730
dropoff_latitude	-3.5173	1.739	-2.022	0.043	-6.927	-0.108
passenger_count	0.0438	0.041	1.063	0.288	-0.037	0.125
day	-0.0020	0.006	-0.336	0.737	-0.014	0.010
year	0.6591	0.027	24.188	0.000	0.606	0.713
month	0.1033	0.015	6.765	0.000	0.073	0.133
hour	0.0042	0.008	0.516	0.606	-0.012	0.020
weekday	-0.0970	0.027	-3.626	0.000	-0.149	-0.045
trip_distance	1.7927	0.014	132.620	0.000	1.766	1.819

Omnibus:	13667.474	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	23448967.651
Skew:	-6.136	**Prob(JB):**	0.00
Kurtosis:	231.602	**Cond. No.**	8.67e+04

Even after splitting the dataset into a 70% train and 30% test we didn't see much improvement.

2.2.3 Regression Trees

Now, we will use other regression models to predict the fare of a cab ride.
Below is a visualization of the decision tree used in our model.



2.2.4 Decision Tree

We have divided train data into 80% train and 20% test datasets for the decision tree model. Let’s look at the decision tree model development code in python.

```
### fit_DT=DecisionTreeRegressor(max_depth=6,random_state=42).fit(X_train,
y_train)
###  predictions_DT = fit_DT.predict(X_test)
```

Here, X_train is subset data from the train dataset for training and has all independent variables. Similarly, y_train is a training dataset with only the target variable.
X_test is test data that is a subset of the train dataset and has all the independent variables.

2.2.5 Random Forest

For Random Forest also we have divided train data into 80% train and 20% test datasets. Let’s look at the random forest model development code in python.

```
###fit_RF=RandomForestRegressor(n_estimators=50,random_state=42).fit(X_train,y_
train)
### prediction_RF=fit_RF.predict(X_test)
```

Here, X_train is a subset data from the train dataset for training and has all independent variables. Similarly, y_train is a training dataset with only the target variable.
X_test is test data that is a subset of the train dataset and has all the independent variables.
n_estimators is nothing but no. Of trees to be used in the random forest.

3. Conclusion

3.1 Model Evaluation

Now that we have three models for predicting the cab fare, we need to decide which one to choose. There are several criteria that are used for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case, we have used predictive performance criteria to select the best model. That means model which gives the best accuracy we will select that model.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables and calculating some error metrics.

3.1.1 Root Mean Squared Error (RMSE)

RMSE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have developed. This is also called as Root Mean Squared Deviation (RMSD)
– it Squares the errors, find their average and takes the square root
–Time-based measure

Lets use RMSE to test accuracy of the Model

```
In [102]: rf_rmse=np.sqrt(mean_squared_error(prediction_RF,y_test))
          print("RMSE = ",rf_rmse)

RMSE = [illegible]
```

In the above code, prediction_RF is the predicted value and y_test is the actual value.
It will provide the error percentage of the model.
“Reason, why we selected RMSE for measuring error, is that we wanted to punish the larger errors. For example, if we predict very high fare for a small distance then it will lead to the negative impact of the company on the customers. Hence our focus was to remove such errors.”

RMSE value that we got in python is as follows:-

Model Name	Error Rate	Accuracy
Linear Regression	5.38%	94.62%
Deciosion Tree	3.70%	96.30%
Random Forest	3.62%	96.38%

RMSE value that we got in python is as follows:-

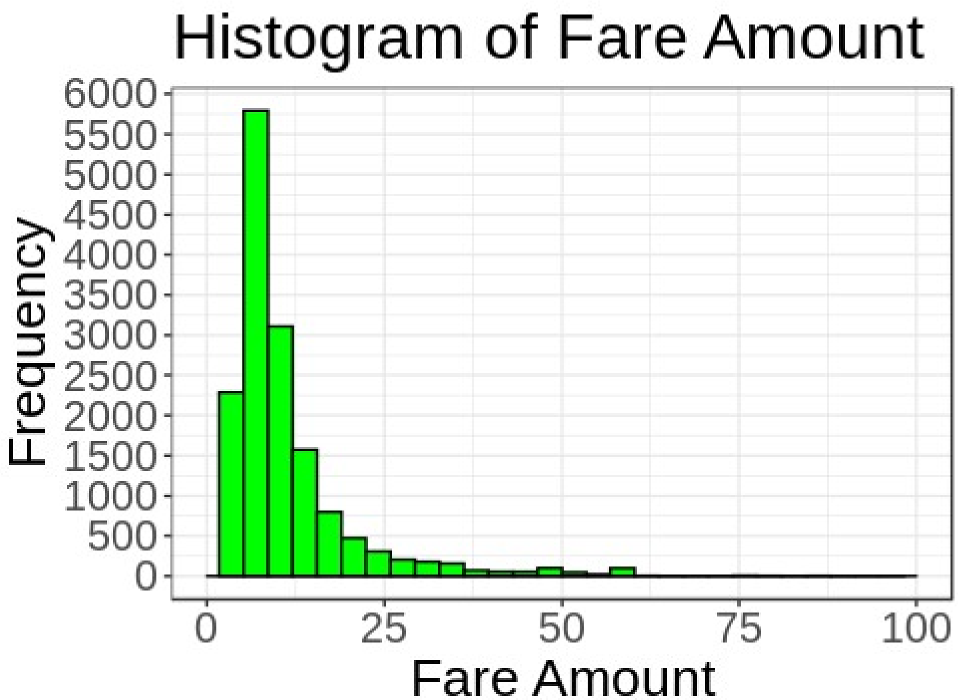
Model Name	Error Rate	Accuracy
Linear Regression	5.35%	94.65%
Deciosion Tree	4.44%	95.56%
Random Forest	3.67%	96.33%

3.2 Model Selection

As we can see from the above tables the random forest gave the best accuracy both in R and python. That's why we selected the Random Forest model for predicting the fare.

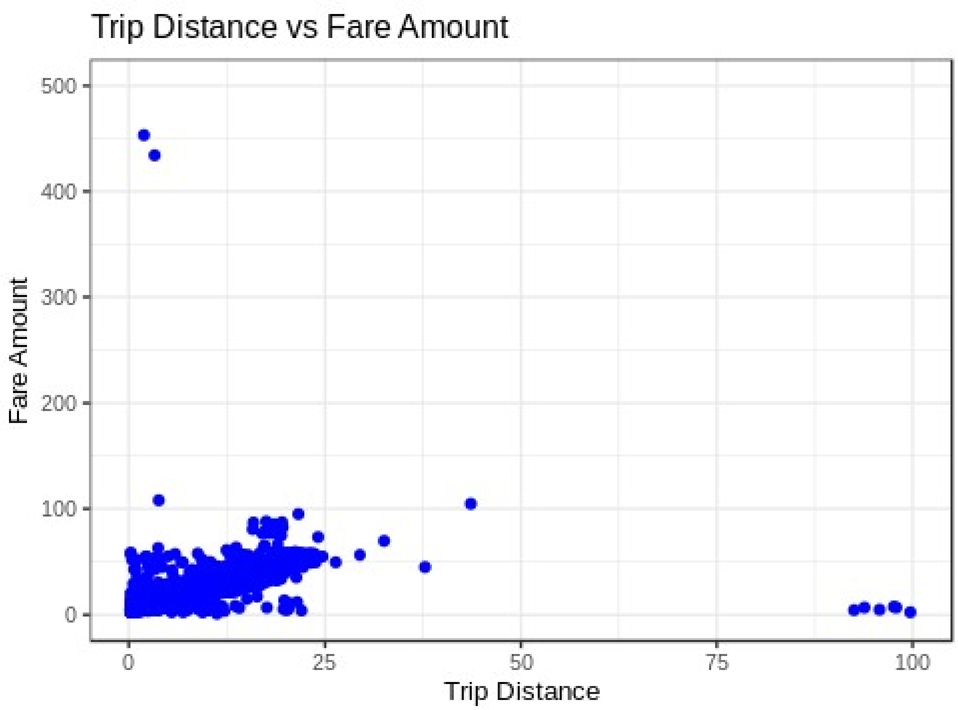
4. Visualizations

4.1 Visualization of the distribution of fare_amount



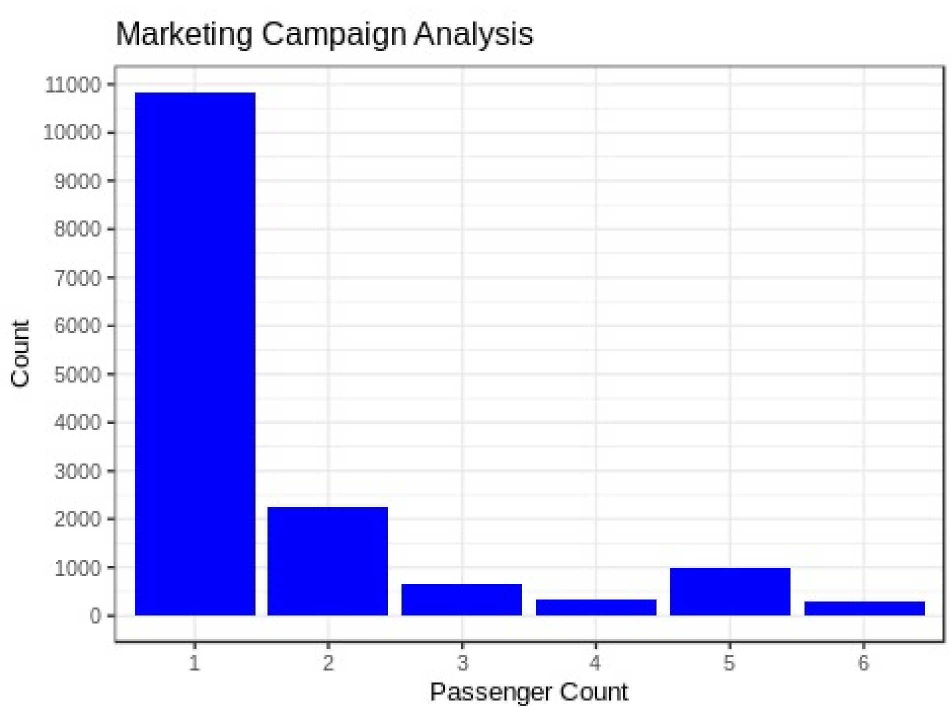
As you can see in the above Histogram that most of the fare_amounts are somewhere between 5 to 15 dollars.

4.2 Visualization on distribution of fare_amount over trip_distance



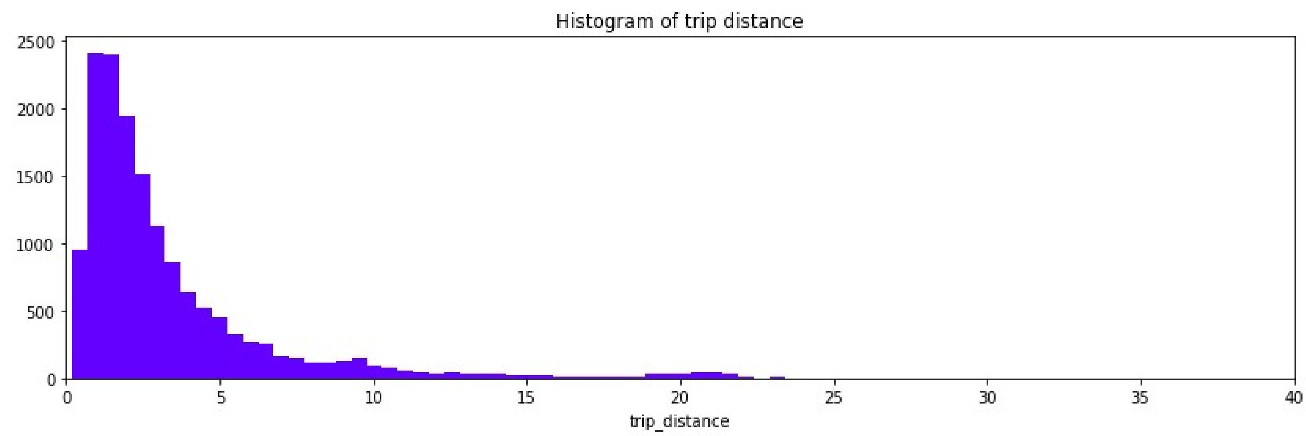
As we can see that in the above scatter plot that fare_amount is almost fixed for trips over 80 KM. Also, fare_amount is few times very high for small distances which we considered as an outlier and removed during outlier analysis.

4.3 Visualization on the count of passengers



As we can see in the above bar graph that single passengers booked a cab for most numbers of the time whereas family booking was least.

4.4 Visualization on distribution of trip_distance



As we can see in the above histogram that most of the trip distance was between