Cab Fare Prediction

Shrunket Sanjeev Bahadure

27th November, 2019

Contents

Chapter 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.

1.2 Data

The task of this project is to build a regression model to predict fares of the cab rides from historical data of the pilot project. Below is the sample data that we are using for the prediction.

Table 1.1 Cab Fare Sample Data (Column 1-7) (Rows 1-5)

fare_amount	pickup_datetime	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
4.5	2009-06-15 17:26:21 UTC	-73.8443	40.72132	-73.8416	40.71228	1
16.9	2010-01-05 16:52:16 UTC	-74.016	40.7113	-73.9793	40.782	1
5.7	2011-08-18 00:35:00 UTC	-73.9827	40.76127	-73.9912	40.75056	2
7.7	2012-04-21 04:30:42 UTC	-73.9871	40.73314	-73.9916	40.75809	1
5.3	2010-03-09 07:51:00	-73.9681	40.76801	-73.9567	40.78376	1

	UTC					
12.1	2011-01-06 09:50:45 UTC	-74.001	40.73163	-73.9729	40.75823	1
7.5	2012-11-20 20:35:00 UTC	-73.98	40.75166	-73.9738	40.76484	1
16.5	2012-01-04 17:22:00 UTC	-73.9513	40.77414	-73.9901	40.75105	1

Table 1.1 Cab Fare Sample Data (Column 1-7) (Rows 6-8)

From the above data it is clear that we have 6 attributes as follows:

Number of attributes:

· pickup_datetime - timestamp value indicating when the cab ride started.

· pickup_longitude - float for longitude coordinate of where the cab ride started.

· pickup_latitude - float for latitude coordinate of where the cab ride started.

· dropoff_longitude - float for longitude coordinate of where the cab ride ended.

· dropoff_latitude - float for latitude coordinate of where the cab ride ended.

· passenger_count - an integer indicating the number of passengers in the cab-ride.

We will use the above attributes to predict the target variable *fare_amount*

1.3 Hypothesis Generation

In this section we will write the possible hypothesis which will affect our target variable *fare_amount.*

1. The distance of the trip is directly proportional to the fare amount ie as the trip distance increase so will be the fare amount.

2. The fare amount may differ on the weekdays and weekends.

3. The fare amount may differ on the peak hours and non-peak hours.

Chapter 2

Methodology

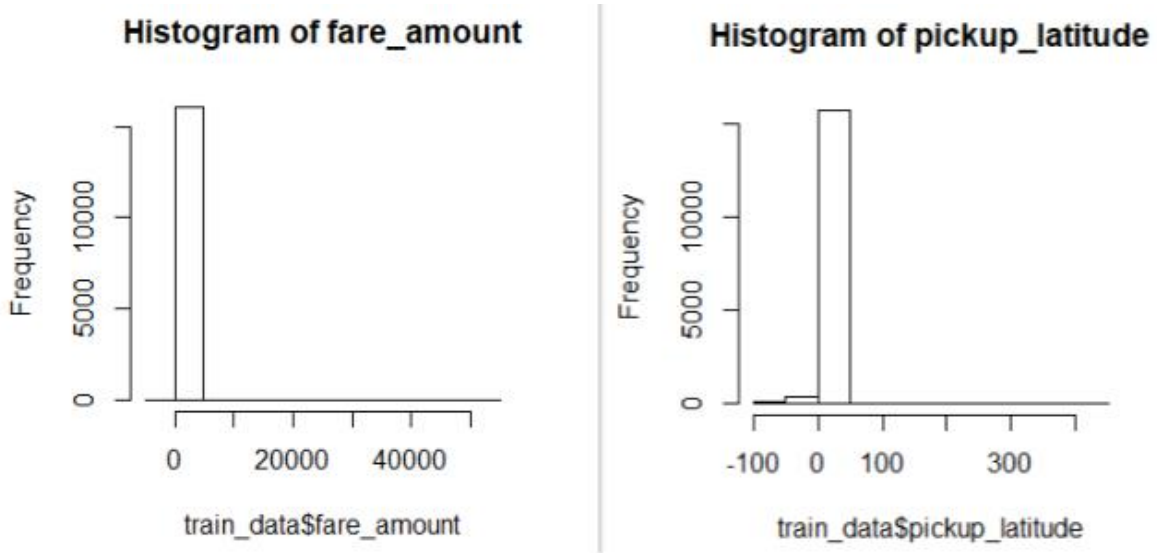
2.1 Pre Processing

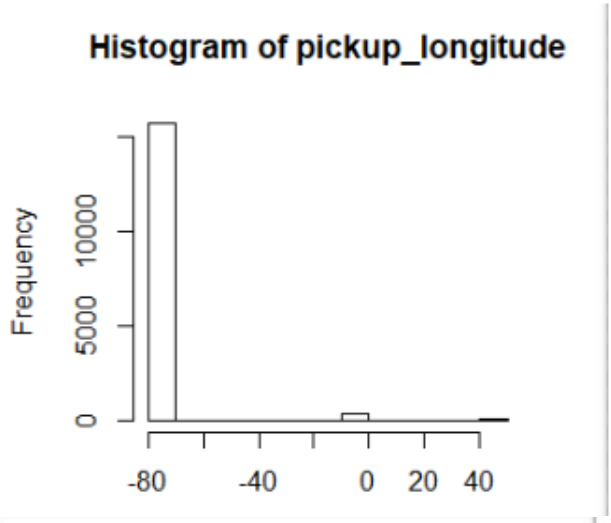
In this section we will explore and understand the available data. And we will process the raw data before designing an optimum model for the given problem.

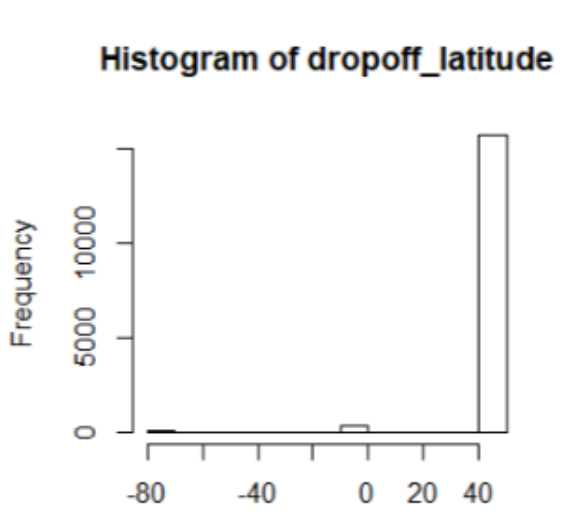
2.2.1 Exploratory Analysis

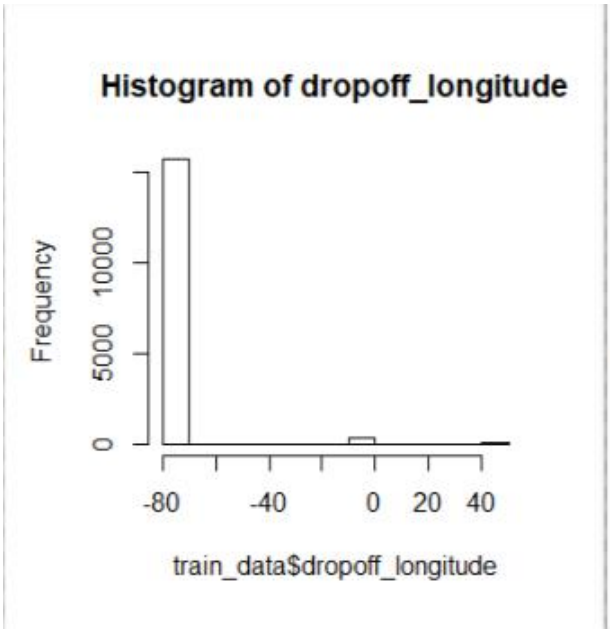
First we will explore the probability distribution of each variable from the given data. Most analysis like regression, require the data to be normally distributed.

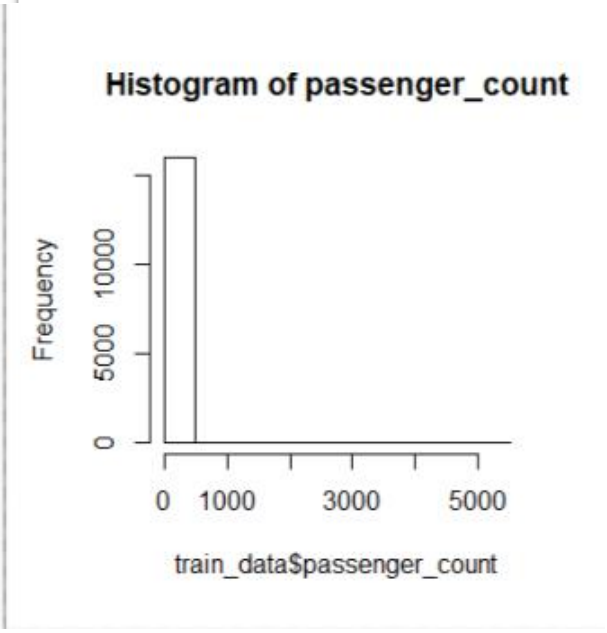
We will also analyse the summary of all the variables. This will help us to design an approach for the model development.




Histogram of pickup_longitude
Frequency
0
5000
10000
-80
-40
0
20
40


Histogram of dropoff_latitude
Frequency
0
5000
10000
-80
-40
0
20
40


Histogram of dropoff_longitude
Frequency
0
5000
10000
-80
-40
0
20
40
train_data$dropoff_longitude


Histogram of passenger_count
Frequency
0
5000
10000
0
1000
3000
5000
train_data$passenger_count

2.2.2 Missing Value Analysis

Missing values in the dataset are caused due to insufficient data or human error. Missing values in the dataset gives us misleading results. Therefore it is very important to treat these missing values.

In this section we will look for missing values in each variable. Let's check summary of each variable table below:

Table 1.2

fare_amount	
Min.	-3.00
1st Qu.	6.00
Median	8.50
Mean	15.02
3rd Qu.	12.50
Max.	54343.00
NA's	24

pickup_longitude	
Min.	-74.44
1st Qu.	-73.99
Median	-73.98
Mean	-72.46
3rd Qu.	-73.97
Max.	40.77

pickup_latitude	
Min.	-74.01
1st Qu.	40.73
Median	40.75
Mean	39.91
3rd Qu.	40.77
Max.	401.08

dropoff_longitude	
Min.	-74.43
1st Qu.	-73.99
Median	-73.98
Mean	-72.46
3rd Qu.	-73.96
Max.	40.80

dropoff_latitude	
Min.	-74.01
1st Qu.	40.73
Median	40.75
Mean	39.90
3rd Qu.	40.77
Max.	41.37

passenger_count	
Min.	0.000
1st Qu.	1.000
Median	1.000
Mean	2.625
3rd Qu.	2.000
Max.	5345.000
NA's	55

In the table above it is clear that we have missing values (shown as NA) in two variables: *fare_amount* and *passenger_count.*
Total NA's in *fare_amount: 24*
Total NA's in *passenger_count: 55*

As the total missing values in both the variables are less than 30% of the total observations respectively, therefore we will impute the values.

We will consider two methods for imputation: 1) Statistical method(mean, mode, median) 2) K-nearest neighbours. We will first randomly choose any 5 non-NA values from the variables and note down the original values for the respective columns. Then we will replace it with NA. After implementing two imputation methods, we came to the conclusion that the values imputed with K-nearest neighbors (KNN) is the closest compared to the other methods. Hence, we will use KNN imputation to impute missing values for both the variables.

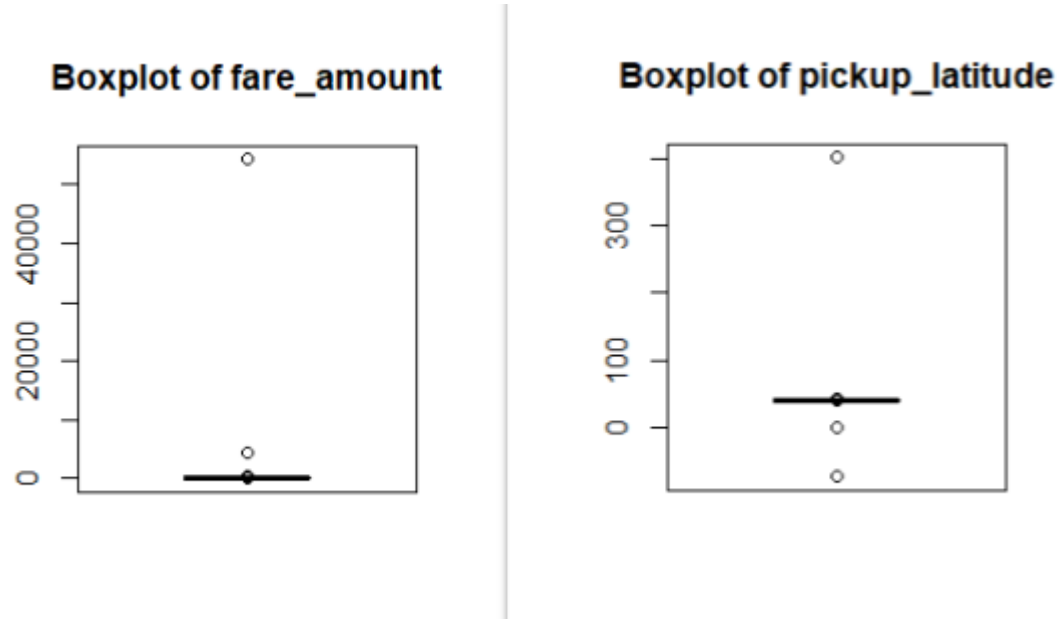
2.2.3 Outlier Analysis

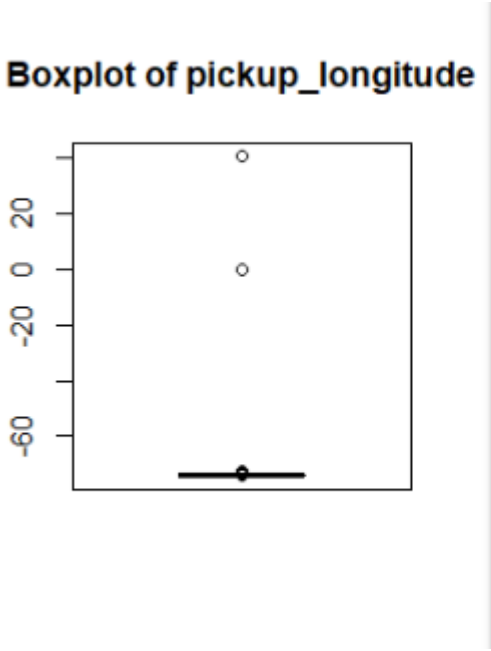
In the fare_amount column in Table 1.2, minimum value is -3.00 and maximum value is 54343. As the fare cannot be zero or negative number , we will consider this as a human error and drop the observations with fare less than or equal to zero. As we have seen the distribution of fare_amount variable all falls below 100. Hence we will also drop the observations with the fare values greater than 100.

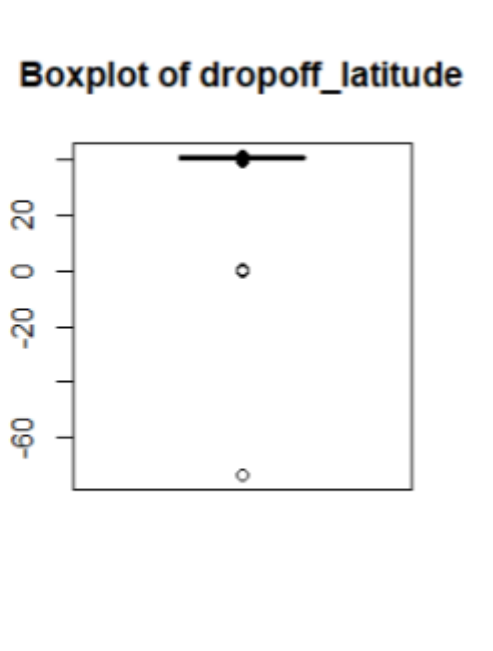
In the passenger_count column in Table 1.2, minimum value is 0 and maximum value is 5345. As the number of passenger cannot be 0 and more than 8, hence we drop the observations with the above conditions.

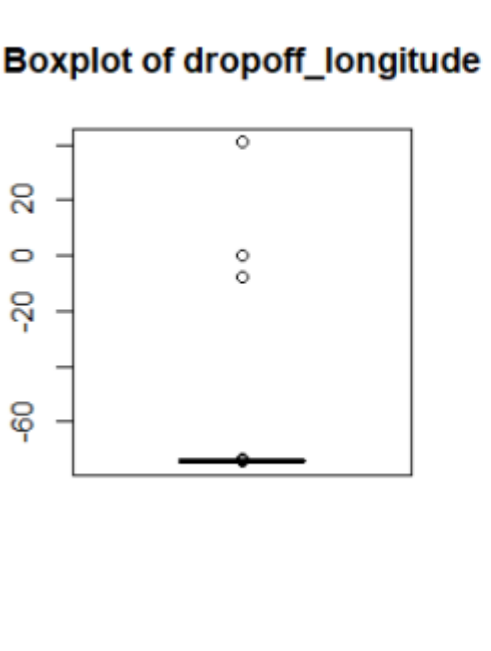
The outlier in an observation is an observation point that is distant from other observations. Outliers can either be a mistake or just variance. This can significantly impact our model performance. Therefore it is important to treat the outliers either by removing it or using imputation.

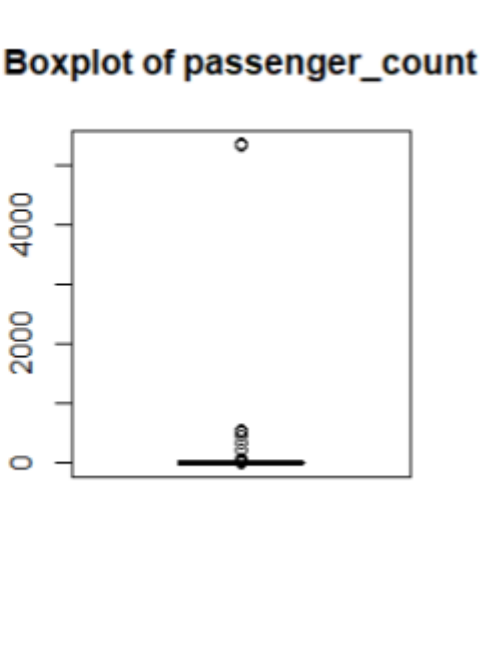
In our case outliers are caused due to human error. Therefore we have decided to drop the outlier observations from the dataset. To identify outliers we will use boxplot method.




Boxplot of pickup_longitude
20
0
-20
-60


Boxplot of dropoff_latitude
20
0
-20
-60


Boxplot of dropoff_longitude
20
0
-20
-60


Boxplot of passenger_count
4000
2000
0

2.2.4 Feature Engineering

The purpose of feature engineering is to derive more variables from the existing variables which will be more helpful in describing the target variable and eventually improve the performance of the machine learning model.

As per our hypothesis, we can derive trip distance with the help of location variables. This derived variable adds significant value in describing our target variable. We can also derive separate date and time columns from the pickup_datetime variable.

Transformed dataset is as shown below:

Table 1.2 Cab Fare Sample Data (Column 1-6) (Rows 1-5)

fare_amount	pickup_longitude	pickup_latitude	dropoff_longitude	dropoff_latitude	passenger_count
16.9	-74.016	40.7113	-73.9793	40.782	1
5.7	-73.9827	40.76127	-73.9912	40.75056	2
7.7	-73.9871	40.73314	-73.9916	40.75809	1
5.3	-73.9681	40.76801	-73.9567	40.78376	1
12.1	-74.001	40.73163	-73.9729	40.75823	1

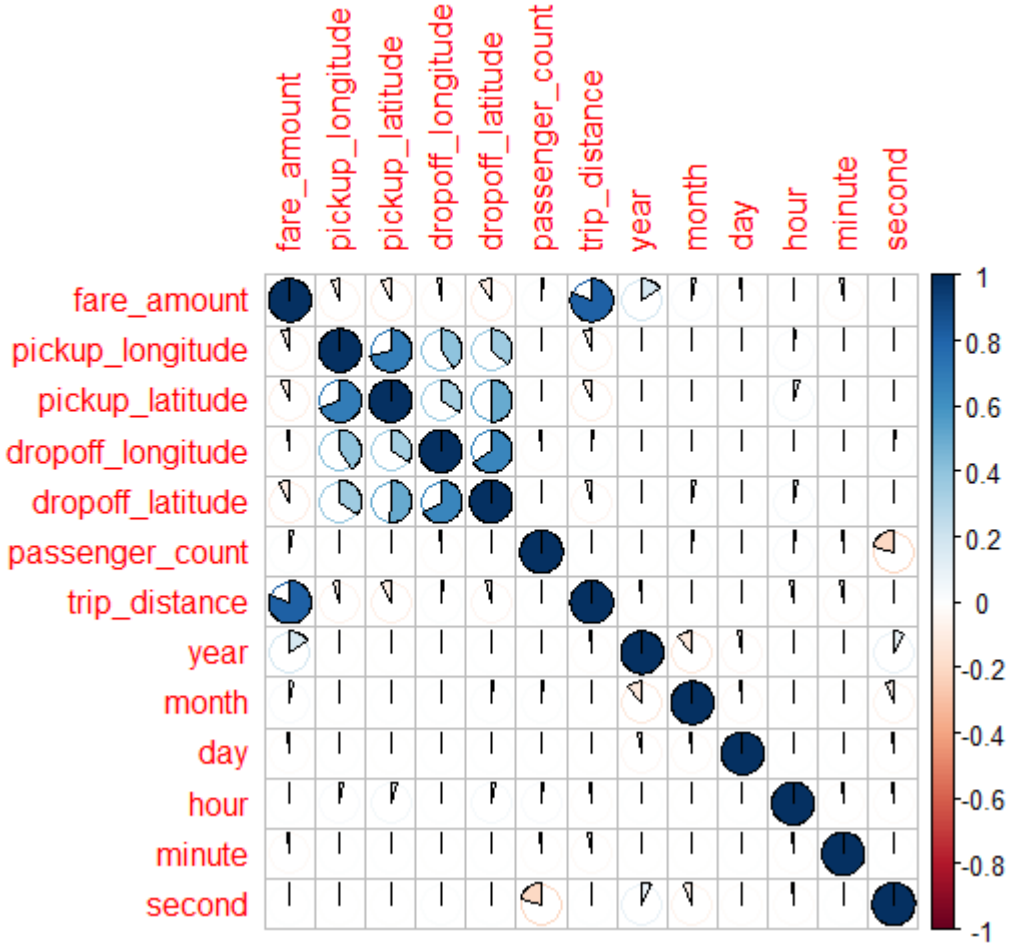
Table 1.2 Cab Fare Sample Data (Column 7-13) (Rows 1-5)

trip_distance	year	month	day	hour	minute	second
5.256552	2010	1	5	16	52	16
0.864378	2011	8	18	0	35	0
1.741334	2012	4	21	4	30	42
1.24361	2010	3	9	7	51	0
2.355917	2011	1	6	9	50	45

2.2.5 Feature Selection

Collinear variables don't add any significant information in describing the target variable. In return it increases the complexity of the model. Hence it is important to remove highly collinear variables from the dataset before training the model.

To check co-linearity between the variables we will plot correlation heat-map between the variables.



Extreme red colour indicates High Negative correlation and extreme blue colour indicates High Positive correlation.

From the above figure , we can say that pickup_latitude and pickup_longitude are positively correlated. dropoff_latitude and dropoff_longitude is also positively correlated. Therefore we will remove any one from the two variables. So we will discard pickup_longitude and dropoff_longitude from our dataset.

2.2.6 Feature Scaling

To remove bias from the model and speed up the calculations in an algorithm, feature scaling is used. Feature scaling helps to normalise the data within a particular range between 0 to 1. There are two methods: Normalisation and Standardisation. If the variable follows normality curve, standardisation is used or else normalisation is used.

Standardised score also known as Z score is calculated as:

$$z = \frac{x - \mu}{\sigma}$$

where:

μ is the mean of the population.

σ is the standard deviation of the population.

Normalised score is calculated as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

As pickup_latitude and dropoff_latitude follow normality function, we will use standardisation. For the rest of the variables we will use normalisation.

2.2 Model Development

2.2.1 K-Fold Cross Validation

Before selecting an algorithm for development, we must test the performance of each of the algorithm used for regression. We will start testing the performance starting from simple algorithm to the complex one.

To do this we will use K-Fold Cross Validation technique. So we will randomly split the training data into K folds with each observation has equal chance of getting selected. Then fit the model using the K — 1 (K minus 1) folds and validate the model using the remaining Kth fold. Repeat the process K times. We will apply all this steps for each algorithm and get the error score. Based on better error score we will choose the algorithm for further development.

Linear Regression

We are selecting K=10 for each algorithm and record there error scores.

```
library(tidyverse)
library(caret)
library(party)
set.seed(123)

train_con= trainControl(method = 'cv',number = 10)

LR_model= train(fare_amount~.,data =
train_data,method='lm',trControl= train_con)
LR_model
```

Result

```
Linear Regression:  RMSE: 2.0925 | Rsquared: 0.7015 | MAE:
1.4996
```

Logistic Regression

```
train_con= trainControl(method = 'cv',number = 10)
LR_model= train(fare_amount~.,data =
train_data,method='glm',trControl= train_con)
LR_model
```

Result

Logistic Regression: RMSE: 2.0925 Rsquared:0.7015 MAE: 1.4996

Decision tree

```
LR_model = train(fare_amount~., data =
train_data,method ='ctree' , trControl = train_con)
LR_model
```

Result

Decision Tree: RMSE: 2.0788 Rsquared:0.70539 MAE: 1.4822

KNN

```
LR_model = train(fare_amount~., data =
train_data,method ='knn' , trControl = train_con)
LR_model
```

Result

KNN Algorithm : RMSE: 3.6384 Rsquared:0.1022 MAE: 2.8172

Random Forest

```
train_con= trainControl(method = 'oob',number = 10)

LR_model= train(fare_amount~.,data =
train_data,method='rf',trControl= train_con)
LR_model
```

Result

Random Forest: RMSE: 1.9454 Rsquared: 0.7419

2.2.2 Model Selection

Evaluation Summary:

Linear Regression: RMSE: 2.0925 Rsquared:0.7015 MAE: 1.4996

Logistic Regression: RMSE: 2.0925 Rsquared:0.7015 MAE: 1.4996

Decision Tree: RMSE: 2.0788 Rsquared:0.70539 MAE: 1.4822

KNN Algorithm : RMSE: 3.6384 Rsquared:0.1022 MAE: 2.8172

Random Forest: RMSE: 1.9454 Rsquared:0.7419

We have implemented K-Fold Cross Validation for each algorithm. From the evaluation summary it is clear that Random Forest is better than other algorithms for our dataset with lowest RMSE of 1.9454 and highest Rsquared value of 0.7419.
Therefore we will use Random Forest for model development.

2.3 Modelling

2.3.1 Hyper-parameter tuning

To improve the model performance we will perform parameter tuning for *mtry* and *ntree* parameters in Random forest algorithm.

After trying value of mtry=(1,2,3,4,5,6,7,8,9,10), we got a better performance for mtry=4.

After trying value of ntree=(250, 300, 350, 400, 450, 500, 550, 600, 800, 1000, 2000), we got a better performance for ntree=500.

So the final performance of the Random Forest: Rsquared= 0.7466 MSE= 3.7168 RMSE= 1.9279

2.3.2 Random Forest

```
library(randomForest)
RF_model= randomForest(fare_amount~.,train_data,mtry=4,importance=T)
RF_predict= predict(RF_model,test_data)
RF_model
summary(RF_predict)
```

Chapter 3

Conclusion

3.1 Model Evaluation

In this section we will take a look at the error metrics of the model that we developed before and we will understand why it has been used.

3.1.1 R-squared (R2)

R-squared value represents how the independent variables describe the target variable. The perfect R-squared value is 1. Closer to the ideal better is the model. We can calculate using formula:

$$R^2 = 1 - \frac{\sum(y_i - \hat{y})^2}{\sum(y_i - \bar{y})^2}$$

Where,

$\hat{y}$ $-$ *predicted value of* y

$\bar{y}$ $-$ *mean value of* y

For our Random Forest model we achieved R2= 0.7419

To know how close the data are to the fitted regression line we choose R-squared as a statistical measure.

3.1.2 Root Mean Squared Error (RMSE)

When we do a square root of summation of the square of distances from the points to the line, we get root mean squared error. The drawback of this error metric is that it is sensitive to outliers.
We can calculate using formula:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

For our Random Forest model we achieved RMSE= 1.9279

As we have removed outliers from our data we choose RMSE as an error metric for model evaluation.

Predicted values of the test data using R is in ***final_result_R.csv*** file.

Predicted values of the test data using Python is in ***final_result_Python.csv*** file.

Appendix A R Code

```
rm(list=ls())
getwd()
setwd('C:/Users/Samruddhi/Desktop/Edwisor Project 1')

train_data= read.csv('train_cab.csv')
test_data= read.csv('test.csv')
head(train_data)
head(test_data)
str(train_data)
str(test_data)
train_data$fare_amount=
as.numeric(as.character(train_data$fare_amount))

num_col=
c('fare_amount','pickup_longitude','pickup_latitude','dropoff_longitude','dr
opoff_latitude','passenger_count')
for (i in num_col){
 print(summary(train_data[i]))
}
test_num_col=
c('pickup_longitude','pickup_latitude','dropoff_longitude','dropoff_latitude'
,'passenger_count')
for (i in test_num_col){
  print(summary(test_data[i]))
}
hist(train_data$passenger_count, main='Histogram of passenger_count')
```

Missing Value Analysis

```
train_data[111,1] #9
train_data[569,1] #6.5
train_data[11000,1] #9.7
train_data[3,1] #5.7
train_data[6038,1] #6
```

```
#train_data$fare_amount[is.na(train_data$fare_amount)]=
mean(train_data$fare_amount,na.rm = TRUE) # 15.02
#train_data$fare_amount[is.na(train_data$fare_amount)]=
median(train_data$fare_amount,na.rm = TRUE) # 8.50
library(dplyr)
library(DMwR)

train_data_1= knnImputation(train_data,k=5) #Value of KNN with k=5 is
more near to the actual values as than to median
train_data_2= knnImputation(train_data,k=5,meth='median')
train_data_1$passenger_count=train_data_2$passenger_count
str(train_data_1)
train_data= train_data_1

summary(train_data$fare_amount)
summary(train_data$passenger_count)
summary(train_data)
```

Outlier Analysis

```
boxplot(train_data$passenger_count, main='Boxplot of passenger_count')

boxplot.stats(train_data$fare_amount)
barplot(table(train_data$fare_amount),ylim = c(0,100))
train_data= train_data[train_data$fare_amount > 0 &
train_data$fare_amount < 100, ]
train_data= train_data[train_data$passenger_count < 8 &
train_data$passenger_count >=1,]

summary(train_data)
boxplot(test_data$passenger_count)

num_col=
c('fare_amount','pickup_longitude','pickup_latitude','dropoff_longitude','dr
opoff_latitude')
for(i in num_col){
  print(i)
  val = train_data[,i][train_data[,i] %in%
boxplot.stats(train_data[,i])$out]
  print(length(val))
```

```
  train_data = train_data[which(!train_data[,i] %in% val),]
}
```

Feature engineering

```
train_data=train_data[!train_data$pickup_datetime==43,]
library(geosphere)
distance= function(pickup_long,pickup_lat,dropoff_long,dropoff_lat){

trip=distHaversine(c(pickup_long,pickup_lat),c(dropoff_long,dropoff_lat))
  return(trip)
}
for (i in 1:nrow(train_data)){

attempt=distance(train_data$pickup_longitude[i],train_data$pickup_latit
ude[i],train_data$dropoff_longitude[i],train_data$dropoff_latitude[i])
  train_data$trip_distance[i]= attempt/1609.344
}
for (i in 1:nrow(test_data)){

attempt=distance(test_data$pickup_longitude[i],test_data$pickup_latitud
e[i],test_data$dropoff_longitude[i],test_data$dropoff_latitude[i])
  test_data$trip_distance[i]= attempt/1609.344
}

summary(train_data$trip_distance)


library(lubridate)
for (i in 1:nrow(train_data)){
  train_data$year[i]= year(train_data$pickup_datetime[i])
  train_data$month[i]= month(train_data$pickup_datetime[i])
  train_data$day[i]= day(train_data$pickup_datetime[i])
  train_data$hour[i]= hour(train_data$pickup_datetime[i])
  train_data$minute[i]= minute(train_data$pickup_datetime[i])
  train_data$second[i]= second(train_data$pickup_datetime[i])
}
for (i in 1:nrow(test_data)){
  test_data$year[i]= year(test_data$pickup_datetime[i])
  test_data$month[i]= month(test_data$pickup_datetime[i])
```

```
  test_data$day[i]= day(test_data$pickup_datetime[i])
  test_data$hour[i]= hour(test_data$pickup_datetime[i])
  test_data$minute[i]= minute(test_data$pickup_datetime[i])
  test_data$second[i]= second(test_data$pickup_datetime[i])
}

train_data=train_data[,-c(2)]
test_data= test_data[,-c(1)]
train_data= train_data[!train_data$trip_distance==0,]
#write.csv(train_data,'final_train.csv',row.names = F)
```

Feature Selection

```
library(corrplot)
con_var=
c('fare_amount','pickup_longitude','pickup_latitude',"dropoff_longitude","
dropoff_latitude","trip_distance","year","month","day","hour","minute","s
econd")
corrplot(cor(train_data),method='pie')
cor(train_data)
train_data= train_data[,-c(2,4)]
```

Feature Scaling

```
hist(train_data$pickup_latitude)
std_var=c('pickup_longitude','pickup_latitude',"dropoff_longitude","dropof
f_latitude")
norm_col=c('trip_distance','year','month','day','hour','minute','second')
# Normalisaztion on norm_col
for(i in norm_col){
  print(i)
  train_data[,i] = (train_data[,i] - min(train_data[,i]))/
    (max(train_data[,i] - min(train_data[,i])))
}
#Standardisation on std_var
for(i in std_var){
  print(i)
  train_data[,i] = (train_data[,i] - mean(train_data[,i]))/
    (sd(train_data[,i] ))
}
```

```
corrgram(train_data)
```

K-Fold Cross Validation

```
library(tidyverse)
library(caret)
library(party)
set.seed(123)

train_con= trainControl(method = 'oob',number = 10)

# Multiple Algorithms
LR_model= train(fare_amount~.,data =
train_data,method='rf',trControl= train_con)
LR_model
```

Hyper-parameter tuning

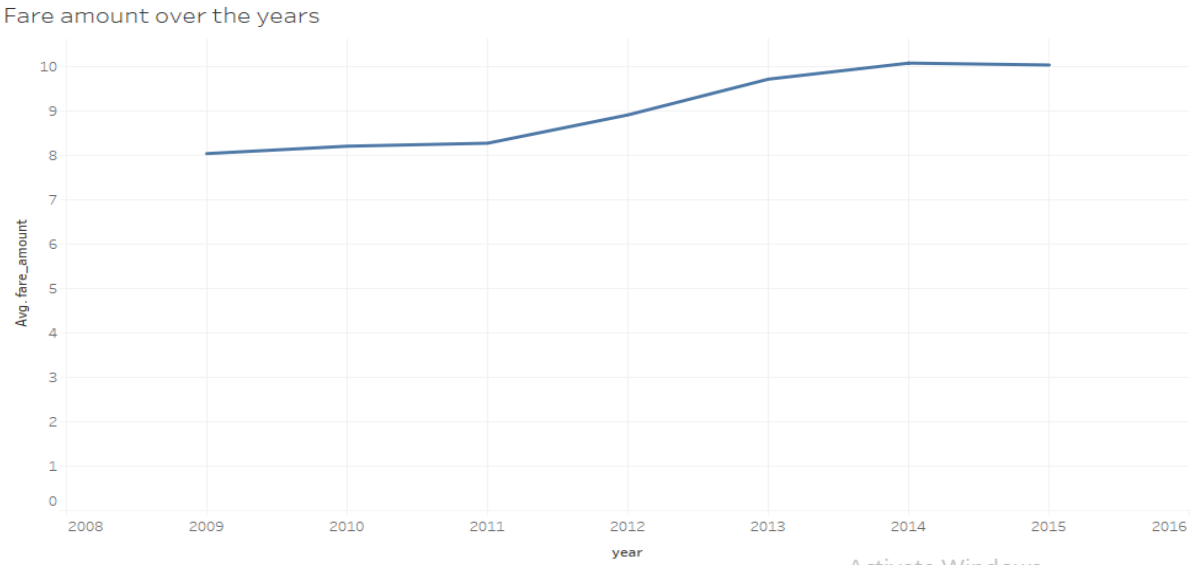
```
set.seed(1234)
tune_Grid = expand.grid(.mtry = c(1: 10))
rf_mtry = train(fare_amount~.,
                data = train_data,
                method = "rf",
                tuneGrid = tune_Grid,
                trControl = train_con,
                importance = TRUE,
                ntree=num_tree
)
print(rf_mtry)
```

Model Building

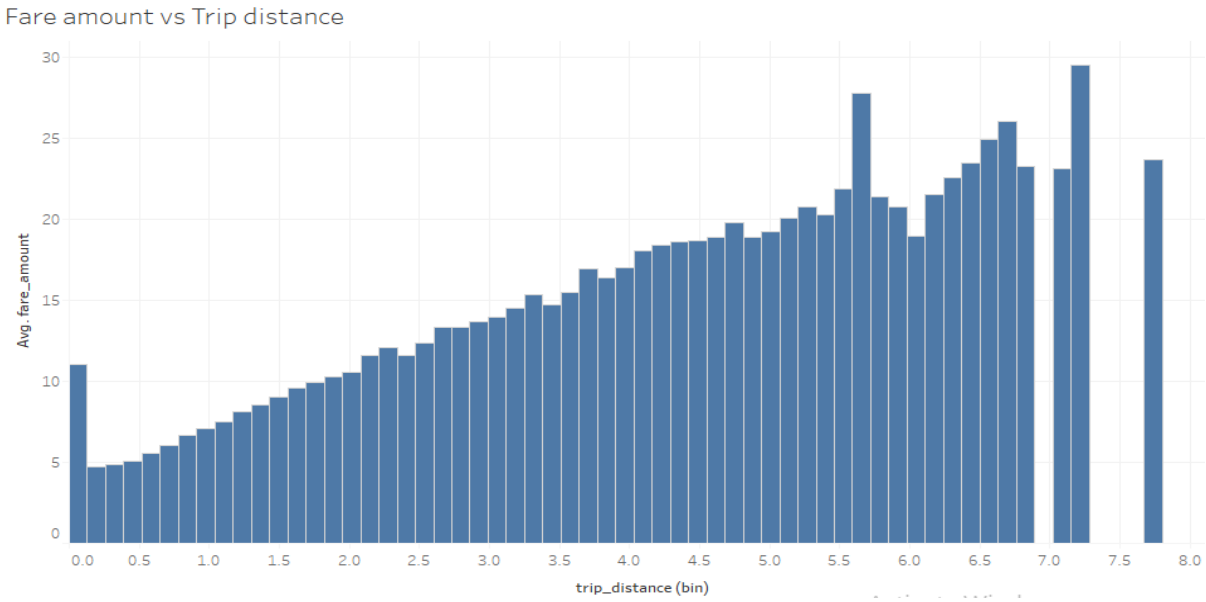
```
library(randomForest)
RF_model= randomForest(fare_amount~.,train_data,mtry=4,importance=T)
RF_predict= predict(RF_model,test_data)
RF_model
summary(RF_predict)

fare_amount_pred= cbind(test_data,RF_predict)
test_data= read.csv('test.csv')
test_data= cbind(test_data,RF_predict)
colnames(test_data)[colnames(test_data)=='RF_predict']= 'fare_amount'
write.csv(test_data,'final_result_R.csv',row.names = F)
```

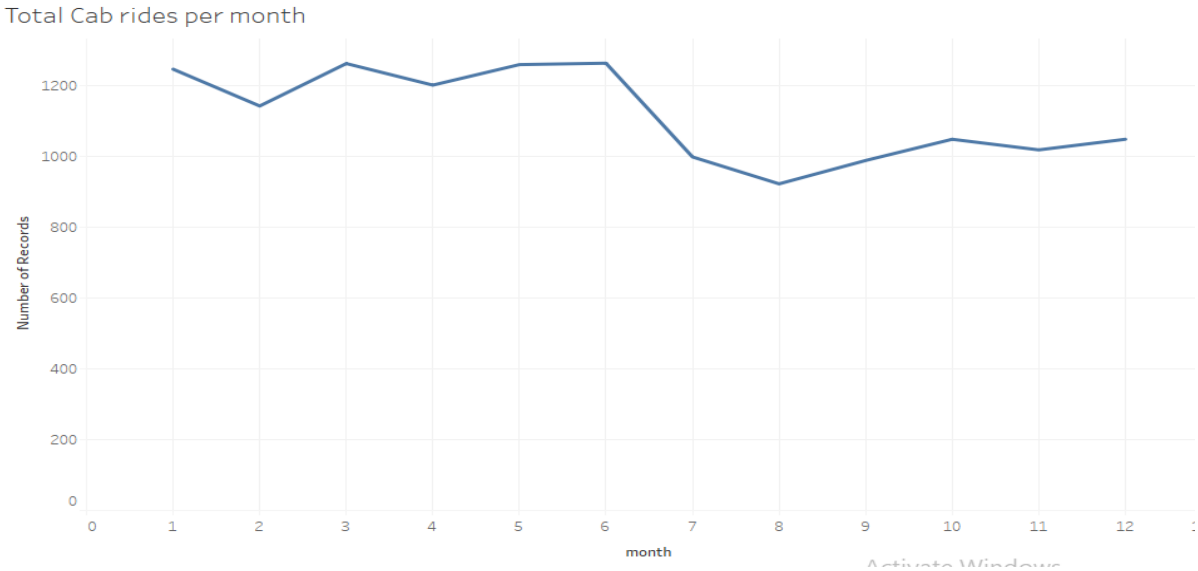
Appendix B Additional Learning



Average fare_amount increased over the years.



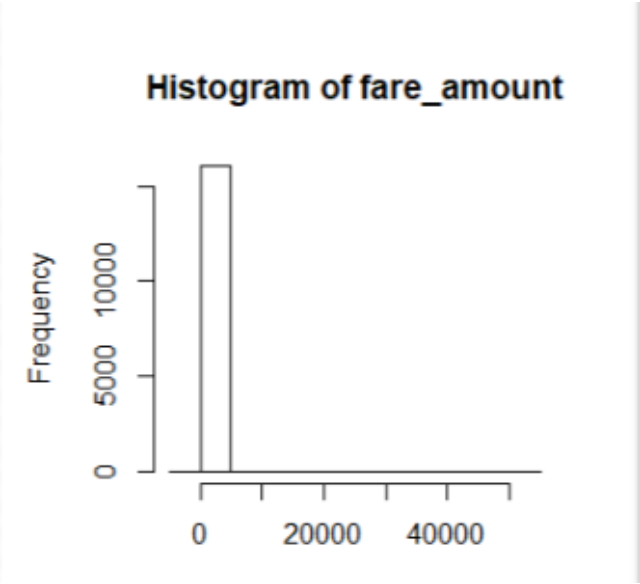
As per our hypothesis, average fare amount increases as the trip distance increase.

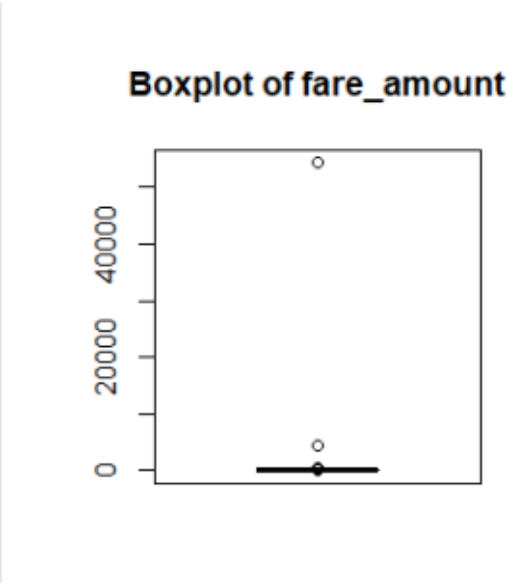


Number of cab rides in the month of June is the highest while cab rides in the month of August is the lowest.

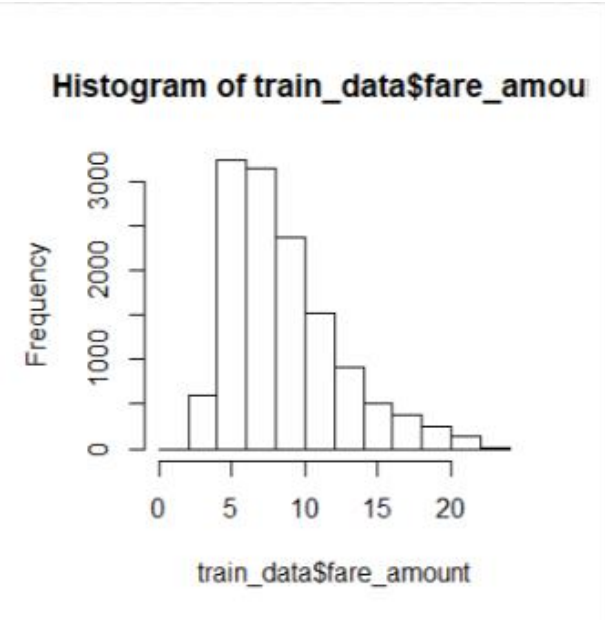
Extra Figures

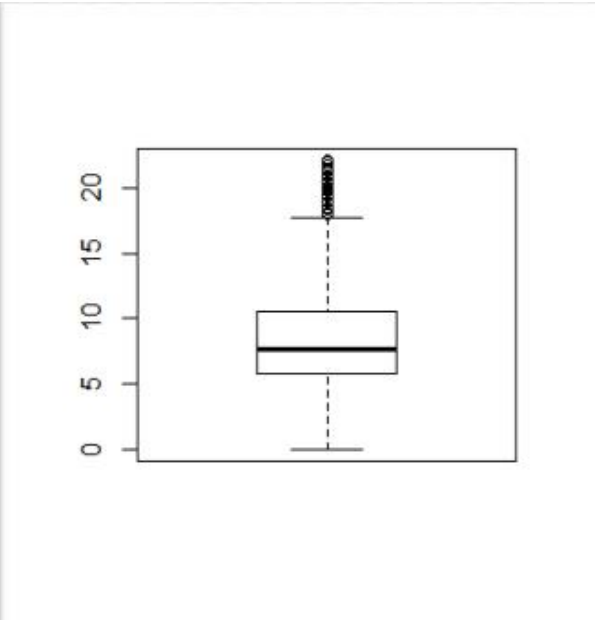
With Outliers



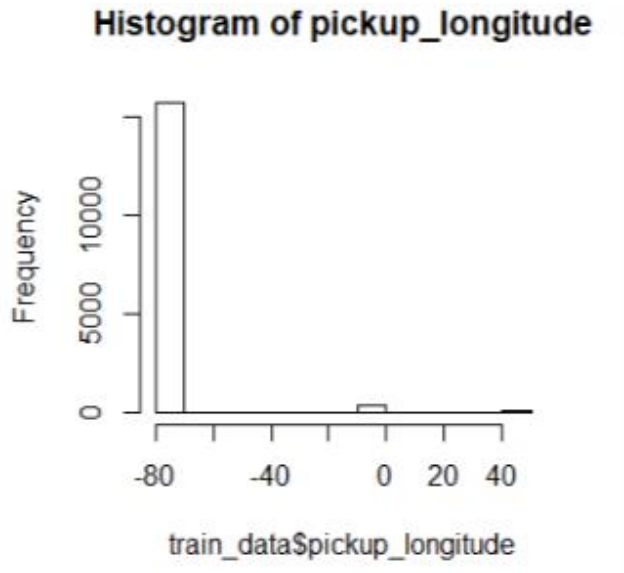


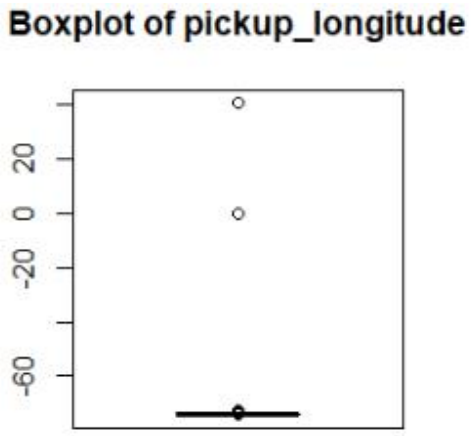
Without Outliers



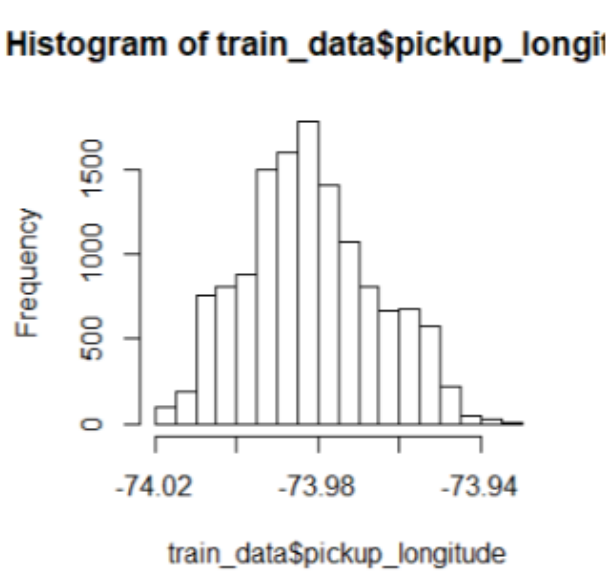


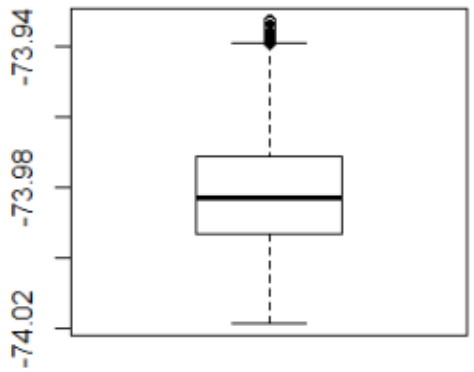
With Outliers



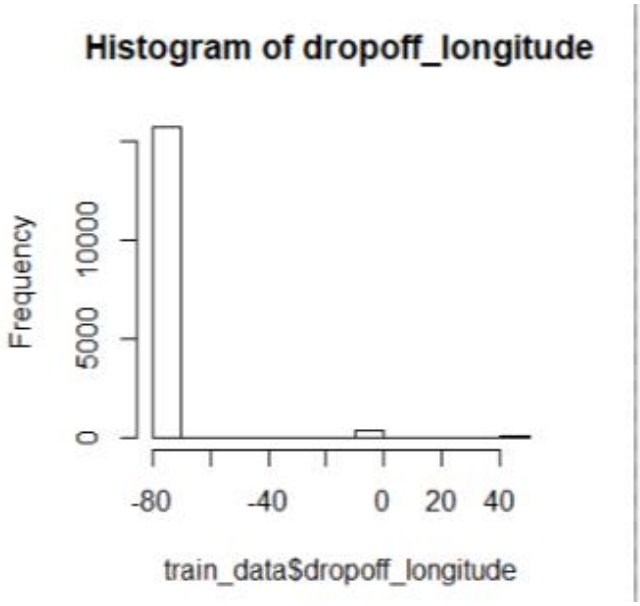


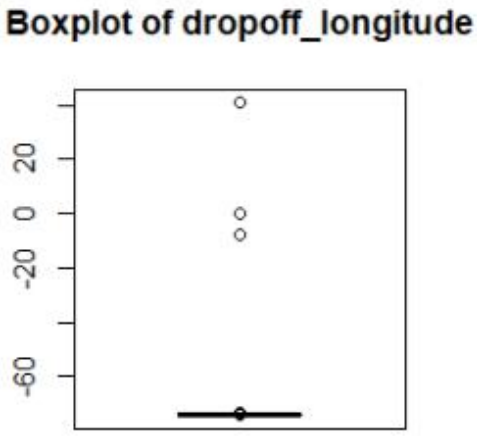
Without Outliers



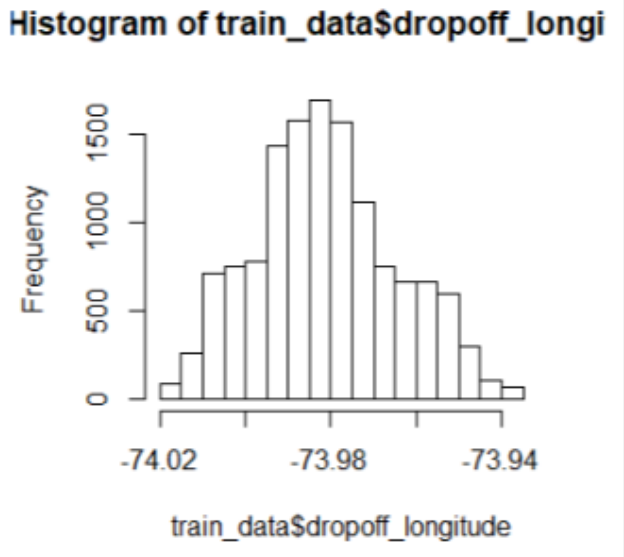


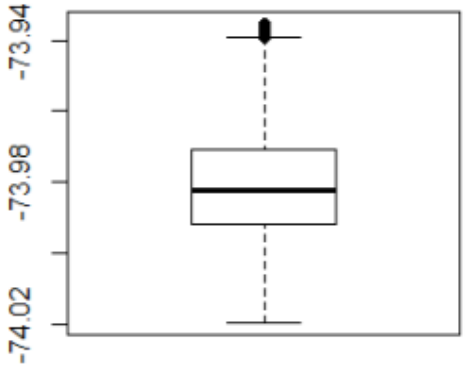
With Outliers





Without Outliers





References

1. https://edwisor.com/
2. https://www.analyticsvidhya.com/
3. https://medium.com/
4. https://towardsdatascience.com/
5. https://en.wikipedia.org/