A Course Work Project Report

*on*

“Cab Price Prediction”

*by*

Pooja Dosi (T204083)

Kritika Singh (T204068)

Mrunmai Bhole (T204016)

Jayshri Sable (T204009)

*of*

**SDL: R Programming**

**T.Y. BTech**

**Academic Year: 2020-2021**

**School of Computer Engineering & Technology**



**Acknowledgement**

We want to express our gratitude towards our respected project advisor/guide Prof Sanjay Ghodke, Prof. Jayvant Devare and Prof. Shobha Mourya for his constant encouragement and valuable guidance during the completion of this project work. We also want to express our gratitude towards respected School Dean Mrs. Ranjana Badre for her continuous encouragement.

We would be failing in our duty if we do not thank all the other staff and faculty members for their experienced advice and evergreen co-operation.

1. Pooja Dosi
2. Kritika Singh
3. Mrunmai Bhole
4. Jayshri Sabale

**Contents**

**1. Introduction……………………………………………………………………………4**

1.1 Synopsis

1.2 Description

1.3 Objectives

1.4 Software Requirements

**2. Methodology…………………………………………………………….……………5**

2.1 Input Data

2.2 Processing

**3. Workflow Diagram……………………………………………………………………7**

**4. Exploratory Data Analysis……………………………………………………………8**

4.1 EDA

4.2 Visualization

**5. Code………………………………………………………………………….……….10**

5.1 Code

5.2 Diagrams

**6. Conclusion……………………………………………………………………………26**

**7. References……………………………………………………………………………27**

**Chapter 1**

**Introduction**

**1.1 SYNOPSIS:**

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 DESCRIPTION:**

This Data Science project is on Cab Price Prediction. The project is about a cab company who has done its pilot project and now they are looking to predict the fare for their future transactional cases. As, nowadays there are a number of cab companies like Uber, Ola, Meru Cabs etc. And these cab companies deliver services to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. In this case, earn most revenues. So, it becomes really important to estimate the fare prices accurately.

**1.3 OBJECTIVES:**

The Objective of this project is to predict the Cab fare price of the future test cases by analyzing the historical data. The project might help us to predict when will the company experience profit or loss. This includes data visualization and exploratory analysis of given data sets.

**1.4 SOFTWARE REQUIREMENTS:**

* R Studio
* Anaconda
* Jupyter Notebook

**Chapter 2**

**METHODOLOGY**

**2.1 INPUT DATA:** For the project to work, it is necessary to get the data correct. For accurate data we have determined a certain number of attributes.

**Attributes :**

• *fare\_amount* : fare of the given cab ride.

• *pickup\_datetime* : timestamp value explaining the time of ride start.

• *pickup\_longitude* : a float value explaining longitude location of the ride start.

• *pickup\_latitude* : a float value explaining the latitude location of the ride start.

• *dropoff\_longitude* : a float value explaining longitude location of the ride end.

• *dropoff\_latitude* : a float value explaining latitude location of the ride end

*passenger\_count* : an integer indicating the number of passengers

From the given data it is understood that, we have to predict fare amount, and other variables will help me achieve that, here pickup\_latitude/longitude, dropoff\_latitude/longitude this data are signifying the location of pick up and drop off. It explains the starting point and end point of the ride. So, these variables are crucial for us.

Passenger\_count is another variable that explains how many people or passengers boarded the ride, between the pickup and drop off locations. And pick up date time gives information about the time the passenger is picked up and the ride has started. But unlike

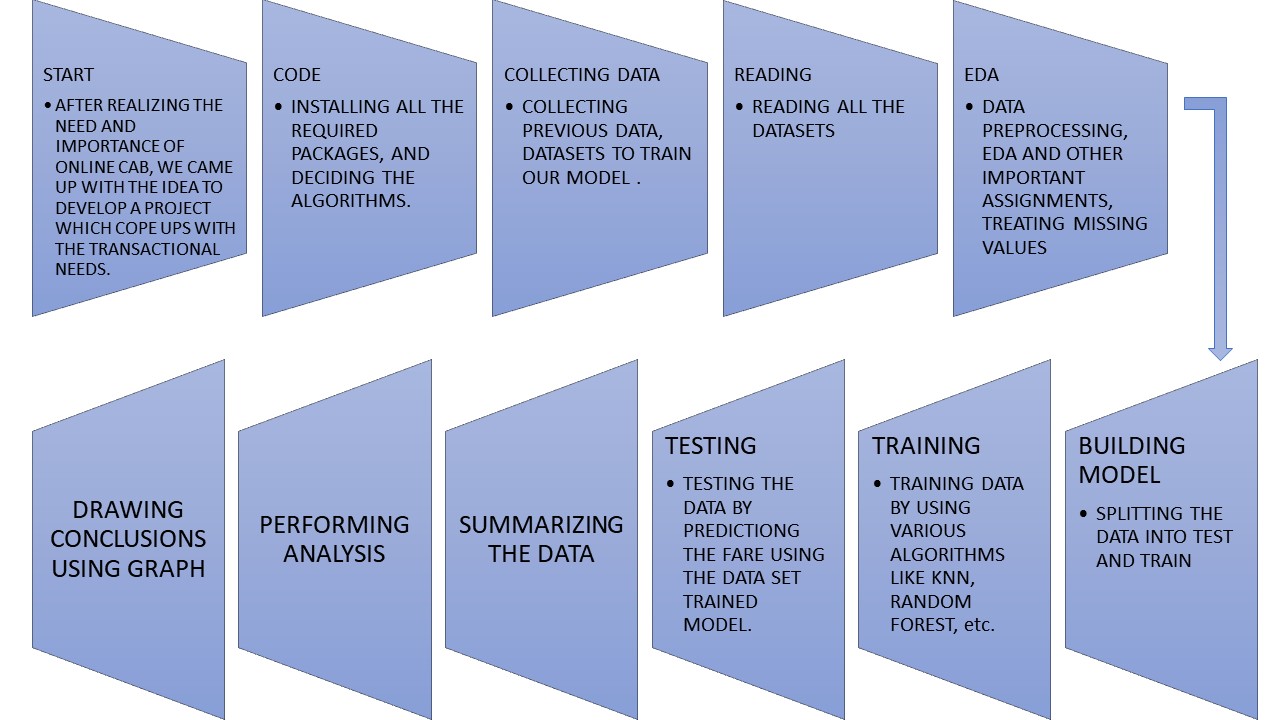
pickup and drop off locations have start and end details both in given data. The time data has only start details and no time value or time related information at the end of the ride. So, during pre-processing of data we will drop this variable. As it seems the information of time is incomplete.

**2.2 PROCESSING:**

The first step is to determine the data requirements or how the data is grouped. Data may be separated variables. The second step in data analytics is the process of collecting it. This can be done through a variety of sources such as computers, online sources, cameras, environmental sources, or through personnel. Once the data is collected, it must be organized so it can be analyzed. Organization may take place on a spreadsheet or other form of software that can take statistical data. The data is then cleaned up before analysis. This means it is scrubbed and checked to ensure there is no duplication or error, and that it is not incomplete. This step helps correct any errors before it goes on to a data analyst to be analyzed. Data analytics techniques can reveal trends and metrics that would otherwise be lost in the mass of information. This information can then be used to optimize processes to increase the overall efficiency of a business or system.

**Chapter 3**

**WORKFLOW DIAGRAM**

****

**Chapter 4**

**EXPLORATORY DATA ANALYSIS**

**4.1 EDA:**

It is an approach where we analyze data sets to summarize and perform initial investigations on data so as to discover patterns, to spot anomalies, to test hypotheses and to check assumptions with the help of summary statistics and graphical representations. It is a good practice to understand the data first and try to gather as many insights from it. EDA is all about making sense of data in hand, before getting them dirty with it.

The data which we have is unstructured in nature so here we need to spend more time for data understanding, and data visualization to figure out new features that are better predictors of cab fare. We take few assumptions in this step.

**Assumption:**

As we are talking about how independent variables will be affected on the target variable. So there will be multiple assumptions.

1) Fare amount is highly dependent on trip distance which we can calculate from pickup and dropout latitude and longitude.

2) Fare amount is depending on how much time it will take to travel from one place to another place. Because, in the traffic it may take more time. So, indirectly it will affect the fare amount.

3) Pickup time also impacts fare charges like supposed journeys may be started in night time so night charges will impact on fare amount.

4) Suppose any location from your City available multiple cabs, so it may be possible the fare rate will be less.

* Treating missing values:

Putting mean, median, mode in place of missing values or removing that particular row.

* Checking the dependencies of all variables, i.e. if they are univariate, bivariate, etc.

**Chapter 5**

**CODE**

**5.1 CODE:**

rm(list = ls())#rm() is basically 'remove{base}', it is used to Remove all the Objects from a Specified Environment.

setwd("C:\\Users\\MRUNMAI VINOD BHOLE\\Desktop\\Cab-fare-prdiction-master")

#set working directory

#Check working directory

getwd()

install.packages("dplyr")

library("dplyr")

install.packages("corrgram")

library("corrgram")

install.packages("car")

library(car)

install.packages("class")

library(class)

install.packages("DMwR")

library(DMwR)

require(DMwR)

install.packages("ggplot2")

library(ggplot2)

install.packages("caret")

library(caret)

library(lattice)

library(rpart)

library(randomForest)

library(RRF)

library(inTrees)

# loading Train and test data

**cab\_train = read.csv("C:\\Users\\MRUNMAI VINOD BHOLE\\Desktop\\Cab-fare-prdiction-master\\train\_cab.csv")**

**test = read.csv("C:\\Users\\MRUNMAI VINOD BHOLE\\Desktop\\Cab-fare-prdiction-master\\test.csv")**

cab\_train

sapply(cab\_train,function(x)sum(is.na(x)))

#To calculate the number of NAs in the entire data.frame per column

################### Exploratory data analysis ###########################

#Getting the number of variables and obervation in the datasets

dim(cab\_train) #dimension of cab\_train

dim(test) #test

#Structure of data

str(cab\_train)

str(test)

#creating muissing value dataframe

missing\_val = data.frame(apply(cab\_train, 2 , function(x){sum(is.na(x))}))

missing\_val$Columns = row.names(missing\_val)

# create missing value % column

names(missing\_val)[1] = "missing\_percentage"

#fill missing value %

missing\_val$missing\_percentage = (missing\_val$missing\_percentage/nrow(cab\_train))\*100

#fill missing value in descending order

missing\_val = missing\_val[order(-missing\_val$missing\_percentage),]

#rearrange column name

missing\_val = missing\_val[, c(2,1)]

#write missing value dataframe in disk

write.csv(missing\_val, "missing\_perc.csv", row.names = F)

cab\_train$fare\_amount = as.numeric(as.character(cab\_train$fare\_amount))

cab\_train$passenger\_count = as.numeric(as.character(cab\_train$passenger\_count))

cab\_train$pickup\_longitude = as.numeric(as.character(cab\_train$pickup\_longitude))

cab\_train$pickup\_latitude = as.numeric(as.character(cab\_train$pickup\_latitude))

cab\_train$dropoff\_longitude = as.numeric(as.character(cab\_train$dropoff\_longitude))

cab\_train$dropoff\_latitude = as.numeric(as.character(cab\_train$dropoff\_latitude))

sum(is.na(cab\_train$passenger\_count))/length(cab\_train$passenger\_count)\*100

# passenger\_count has 0.342% of missing values

sum(is.na(cab\_train$fare\_amount))/length(cab\_train$fare\_amount)\*100

# fare\_amount has 0.155% of missing values

# Now we'll check the outliers in passenger count:

boxplot(cab\_train$passenger\_count, outcol= "Red")

table(cab\_train$passenger\_count)

#freqency per value eg. 0 is 57 times in col pass count

###############Starting filling missing values of passenger\_count############

# As passenger\_count is a factor variable, we'll fill the missing values by using Mode method(most frequent value)

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

result = getmode(cab\_train$passenger\_count)

print(result)

cab\_train$passenger\_count[is.na(cab\_train$passenger\_count)] = result

table(cab\_train$passenger\_count)

###########Starting filling missing values of fare amount#######

summary(cab\_train$fare\_amount)

boxplot(cab\_train$fare\_amount, outcol= "Red")

###########below is the method to extract the outliers######

val = cab\_train$fare\_amount[cab\_train$fare\_amount %in% boxplot.stats(cab\_train$fare\_amount)$out]

length(val)

#To save other information, we shouldn't remove outliers#####

cab\_train$fare\_amount[cab\_train$fare\_amount %in% val]= NA

summary(cab\_train$fare\_amount)

#created NA in place of outliers

#now fill these NA by using mean, median and KNN

cab\_train$fare\_amount[7500] #choosing random value....at 7500 location, value is = 7

cab\_train$fare\_amount[7500] = NA

#mean = 8.902

#median = 8

#KNN = 7.857

library(class)

summary(cab\_train$fare\_amount)

install.packages("VIM")

library(VIM)

#knn method

cab\_train= knnImputation(cab\_train$fare\_amount,3)

cab\_train$fare\_amount[7500]

# As we can see that KNN imputes the nearest value, we'll freeze the KNN imputation

summary(cab\_train$fare\_amount)

#Now we'll handle longitude and latitude columns

# As we can see that KNN imputes the nearest value, we'll freeze the KNN imputation

summary(cab\_train$fare\_amount)

#Now we'll handle longitude and latitude columns

summary(cab\_train$pickup\_longitude)

summary(cab\_train$pickup\_latitude)

summary(cab\_train$dropoff\_longitude)

summary(cab\_train$dropoff\_latitude)

#After getting the summary, we found that pickup\_latitude column has a value beyond the range, hence we'll remove it.

nrow(cab\_train[which(cab\_train$pickup\_latitude > 90),])

cab\_train = cab\_train[-which(cab\_train$pickup\_latitude> 90),]

#Now, we'll calculate the distance travelled by using longitude and latitude

#There is a 'haversine' formula to calculate the great-circle distance between two points - he shortest distance over the earth's surface

#We have to create a function to calculate the distance

degree\_to\_radian = function(deg){

(deg \* pi) / 180

}

haversine\_formula = function(long1,lat1,long2,lat2){

long\_1\_radian = degree\_to\_radian(long1)

lat\_1\_radian = degree\_to\_radian(lat1)

long\_2\_radian = degree\_to\_radian(long2)

lat\_2\_radian = degree\_to\_radian(lat2)

dif\_lat = degree\_to\_radian(lat2 - lat1)

dif\_long = degree\_to\_radian(long2 - long1)

a = sin(dif\_lat/2) \* sin(dif\_lat/2) + cos(lat\_1\_radian) \* cos(lat\_2\_radian) \* sin(dif\_long/2) \* sin(dif\_long/2)

c = 2 \* asin(sqrt(a))

R = 6371e3 #radius of earth

R \* c / 1000 #1000 is used to convert to kilometers

}

cab\_train$distance\_travelled\_in\_km = haversine\_formula(cab\_train$pickup\_longitude, cab\_train$pickup\_latitude, cab\_train$dropoff\_longitude, cab\_train$dropoff\_latitude)

# Now, we'll remove the variable which have distance\_travelled less than 25m (taking practical scenario)

nrow(cab\_train[which(cab\_train$distance\_travelled\_in\_km < 0.025),])

cab\_train = cab\_train[-which(cab\_train$distance\_travelled\_in\_km < 0.025),]

##########outliers analysis############

boxplot(cab\_train$distance\_travelled\_in\_km, outcol= "Red")

summary(cab\_train$distance\_travelled\_in\_km)

val2 = cab\_train$distance\_travelled\_in\_km[cab\_train$distance\_travelled\_in\_km %in% boxplot.stats(cab\_train$distance\_travelled\_in\_km)$out]

length(val2)

nrow(cab\_train[which(cab\_train$distance\_travelled\_in\_km > 3000),])

######we found that there are 1339 outliers in distance\_travelled\_in\_km column

#########To save other information, we shouldn't remove outliers

cab\_train$distance\_travelled\_in\_km[cab\_train$distance\_travelled\_in\_km %in% val2]= NA

summary(cab\_train$distance\_travelled\_in\_km)

#created NA in place of outliers

#now fill these NA by using mean, median and KNN

train$distance\_travelled\_in\_km[780] #choosing random value....at 780 location, value is = 4.439835

train$distance\_travelled\_in\_km[780] = NA

#mean = 2.4772

#median = 2.0057

#KNN = 3.9620

summary(cab\_train$distance\_travelled\_in\_km)

train = knnImputation(cab\_train, k=7)

cab\_train$distance\_travelled\_in\_km[780]

# As we can see that KNN imputes the nearest value, we'll freeze the KNN imputation

summary(cab\_train$distance\_travelled\_in\_km)

class(cab\_train$pickup\_datetime)

as.character(cab\_train$pickup\_datetime)

cab\_train$pickup\_datetime = as.factor(as.character(cab\_train$pickup\_datetime))

class(cab\_train$pickup\_datetime)

install.packages("data.table")

library(data.table)

install.packages("datetime")

library(datetime)

install.packages("date")

library(date)

x <- c("1jan1960", "2jan1960", "31mar1960", "30jul1960")

z <- as.Date(x, "%d%b%Y")

## Sys.setlocale("LC\_TIME", lct)

z

x = strptime(cab\_train$pickup\_datetime,"%Y-%m-%d %H:%M:%S")

cab\_train$Date = as.Date(cab\_train$pickup\_datetime,"%Y-%m-%d")

cab\_train$Time = as.factor(format(x,"%H"))

cab\_train$Year = as.factor(format(x, "%Y"))

cab\_train$month = as.factor(format(x, "%m"))

cab\_train$day = as.numeric(format(x,"%d"))

###############visualization fare amount vs hour graph#########################

ggplot(data = df\_train, aes(x = pickup\_hour,y = fare\_amount))+

geom\_bar(stat = "identity",color ="DarkSlateBlue")+

labs(title = "Fare Amount Vs. hour", x = "hour", y = "Fare")+

theme(plot.title = element\_text(hjust = 0.5, face = "bold"))+

theme(axis.text.x = element\_text( color="black", size=6, angle=45))

#########plot histogram to check the normality of data###########

par("mar")

par(mar=c(4,4,4,4))

hist(df\_train$distance\_travelled\_in\_km)

hist(df\_train$pickup\_longitude)

hist(df\_train$pickup\_latitude)

###Above histogramme results shows those variable are not normally distributed then we go for normalization of data step.

###########Normalization of data#####

c\_names = c("pickup\_latitude", "pickup\_longitude", "distance\_travelled\_in\_km")

for (i in c\_names) {

print(i)

df\_train[,i] = (df\_train[,i]- min(df\_train[i]))/(max(df\_train[,i]-min(df\_train[i])))

}

head(df\_train)

str(df\_train)

names(df\_train)

cat\_var = c("passenger\_count", "Year", "month", "pickup\_sessions", "pickup\_days")

##############Performing ANOVA test#######

aov(df\_train$fare\_amount~ df\_train$Year)

# for all categorical variables

for(i in cat\_var){

print(i)

Anova\_test\_result = summary(aov(formula = fare\_amount~df\_train[,i],df\_train))

print(Anova\_test\_result)

}

df\_train$pickup\_days = NULL

head(df\_train)

df = cab\_train

###############################Linear Regression model#############################################3

# fit linear regression model

# we will use the lm() function in the stats package

lm\_model = lm(fare\_amount ~., data =train)

#predictions

summary(lm\_model)

# Lets check the assumptions of ols regression

#Error should follow normal distribution and no hetroscadacity

# assumptions are checked usig residual plot and normal qq plot

# Change the panel layout to 2 x 2

par(mfrow = c(2, 2))

plot(lm\_model)

# No multicolinearity between Independent variables

vif(df\_train[,-1])

vif(lm\_model)

###########predicting for splitted test data\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

predictions\_lm = predict(lm\_model, test[,-1])

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))

}

print(postResample(pred = predictions\_lm , obs =test$fare\_amount))

MAPE(test[,1],predictions\_lm)

MAPE

#evaluation

regr.eval(test[,1],predictions\_lm, stats = c('mape','rmse'))

#MAPE=18.22

#RMSE=0.224

#Accuracy=81.78

#################################Decision tree###########################################

library(rpart)

library(dplyr)

DT = rpart(fare\_amount ~ . , data =train , method = "anova")

predictions\_DT = predict(DT, test[,-1])

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))

}

print(postResample(pred = predictions\_DT , obs =test$fare\_amount))

MAPE(test[,1],predictions\_DT)

MAPE

#evaluation

regr.eval(test[,1], predictions\_DT, stats = c('mape','rmse'))

# MAPE=21.5

# RMSE=0.249

# Accuracy=78.5

############################ Random forest model#############

RF\_model = randomForest(fare\_amount ~.,data = train, importance = TRUE, ntree = 200)

RF\_Predictions = predict(RF\_model, test[,-1])

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))

}

print(postResample(pred = RF\_Predictions , obs =test$fare\_amount))

MAPE(test[,1],RF\_Predictions)

MAPE

#MAPE=18.20

#RMSE=0.221

#Accuracy=81.8

##########manipulating test data for prediction##########

head(test)

df\_test = test

df\_test$distance\_travelled\_in\_km = haversine\_formula(df\_test$pickup\_longitude, df\_test$pickup\_latitude, df\_test$dropoff\_longitude, df\_test$dropoff\_latitude)

head(df\_test)

as.character(df\_test$pickup\_datetime)

df\_test$pickup\_datetime = as.factor(as.character(df\_test$pickup\_datetime))

class(df\_test$pickup\_datetime)

df\_test$Date = as.Date(df\_test$pickup\_datetime)

y = strptime(df\_test$pickup\_datetime,"%Y-%m-%d %H:%M:%S")

df\_test$pickup\_hour = as.factor(format(y,"%H"))

df\_test$Year = as.factor(format(y, "%Y"))

df\_test$month = as.factor(format(y, "%m"))

df\_test$day = as.numeric(format(y,"%d"))

df\_test$pickup\_hour = as.numeric(format(y,"%H"))

df\_test$pickup\_datetime = NULL

df\_test$pickup\_date = NULL

df\_test = rename(df\_test, "pickup\_date" = "Date")

sum(is.na(df\_test$pickup\_hour))

class(train$pickup\_hour)

head(df\_test)

df\_test$pickup\_sessions[df\_test$pickup\_hour >= 0 & df\_test$pickup\_hour < 7] = "1"

df\_test$pickup\_sessions[df\_test$pickup\_hour >= 7 & df\_test$pickup\_hour < 18] = "2"

df\_test$pickup\_sessions[df\_test$pickup\_hour >= 18 & df\_test$pickup\_hour <= 23] = "3"

head(df\_test)

df\_test$pickup\_date = NULL

df\_test$dropoff\_longitude = NULL

df\_test$dropoff\_latitude = NULL

df\_test$pickup\_hour = NULL

df\_test$day = NULL

head(df\_test)

df\_test$passenger\_count %<>% factor

df\_test$month %<>% factor

df\_test$pickup\_sessions %<>% factor

col\_names = c("pickup\_latitude", "pickup\_longitude", "distance\_travelled\_in\_km")

for (j in col\_names) { print(j) df\_test[,j] = (df\_test[,j]- min(df\_test[j]))/(max(df\_test[,j]-min(df\_test[j])))}

#So we can freeze model Random forest because give high accuracy and also less value of MAPE among all model.¶

#############selecting Random Forest for predict test data##########

#predictions

predictions\_test = predict(RF\_model, df\_test)

#save predictions as dataframe

predictions\_test = as.data.frame(df\_test)

#importing original test dataset

cab\_results = read.csv("test.csv", header = T)

#columnbind target results with test data

cab\_results = cbind(predictions\_test, cab\_results)

#renaming column

names(cab\_results)[1] = "fare\_amount\_predicted"

#saving output in csv format

write.csv(cab\_results, "fare amount results r.csv", row.names = F)

**5.2 DIAGRAMS:**

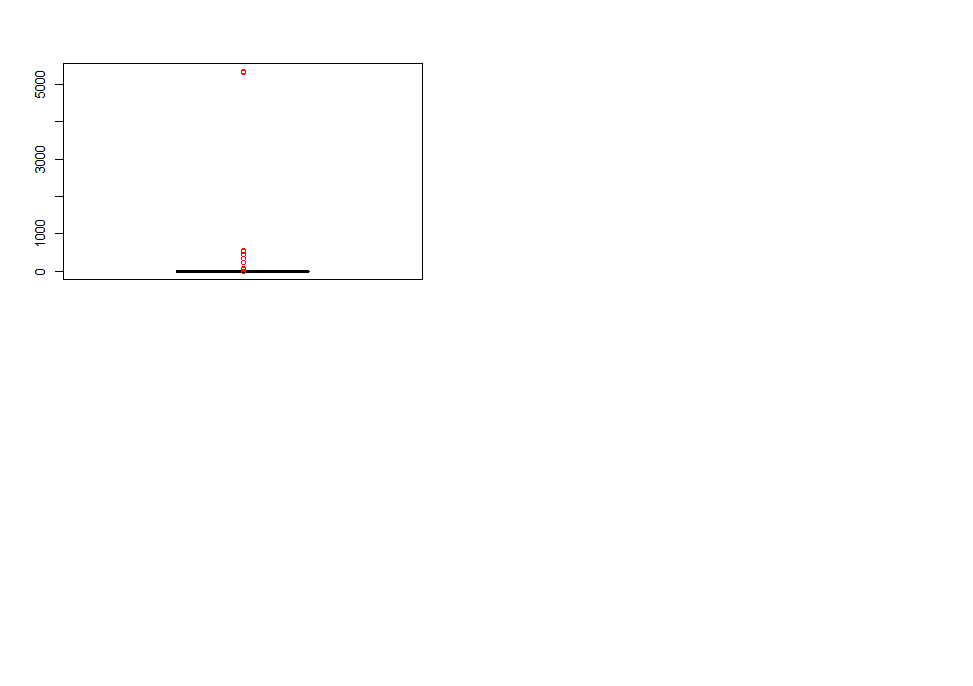


DIAGRAM 5.1 **–** Finding Outliers

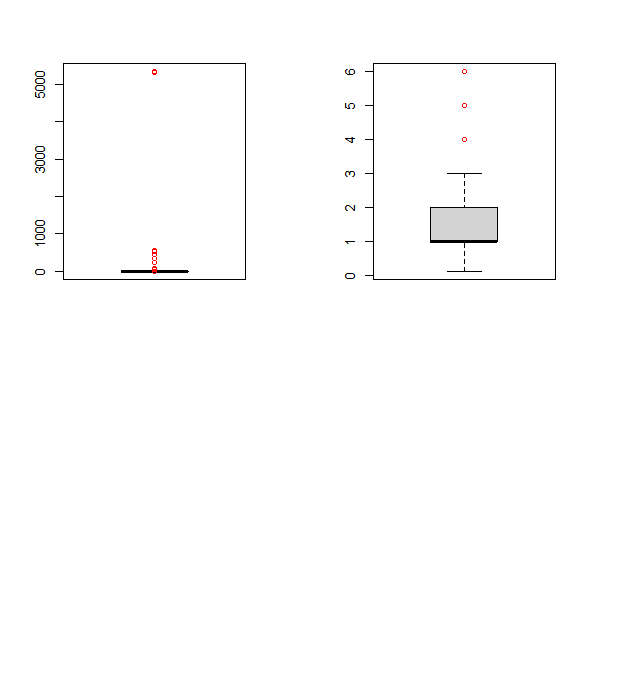


DIAGRAM 5.2 **–** Boxplot Analysis for outliers

**Rplot 287:**

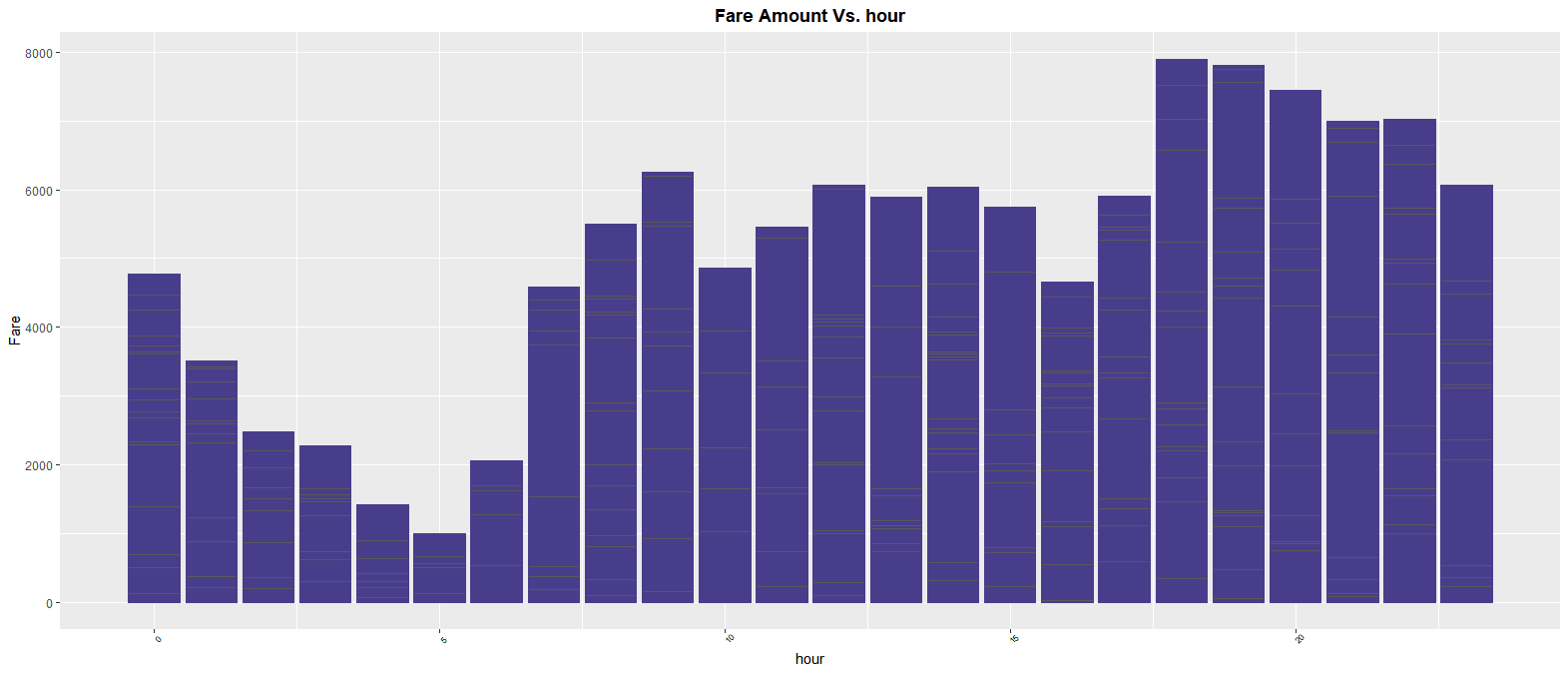


DIAGRAM 5.3 **–** Fare Amount Vs Hour Bar graph

**Rplot 308:**

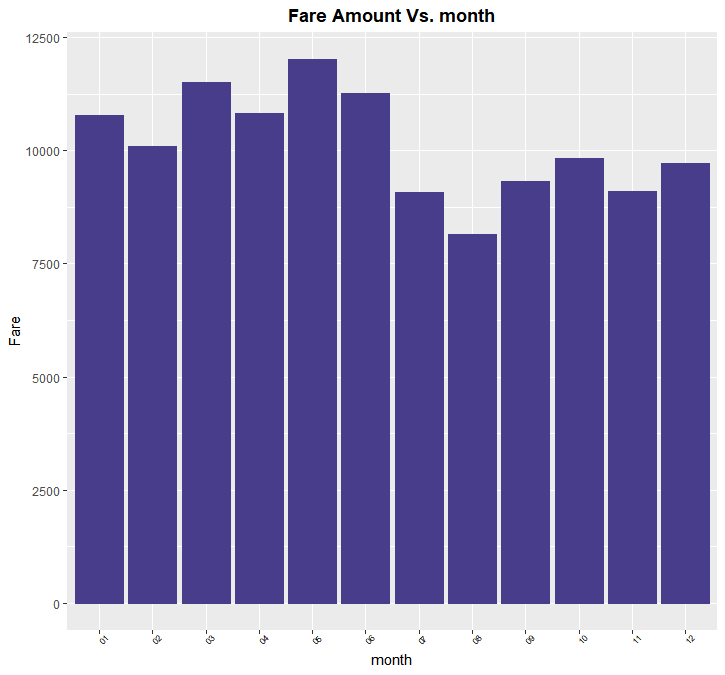


DIAGRAM 5.4 **–** Fare Amount Vs Month Bar graph

**Rplot 316:**

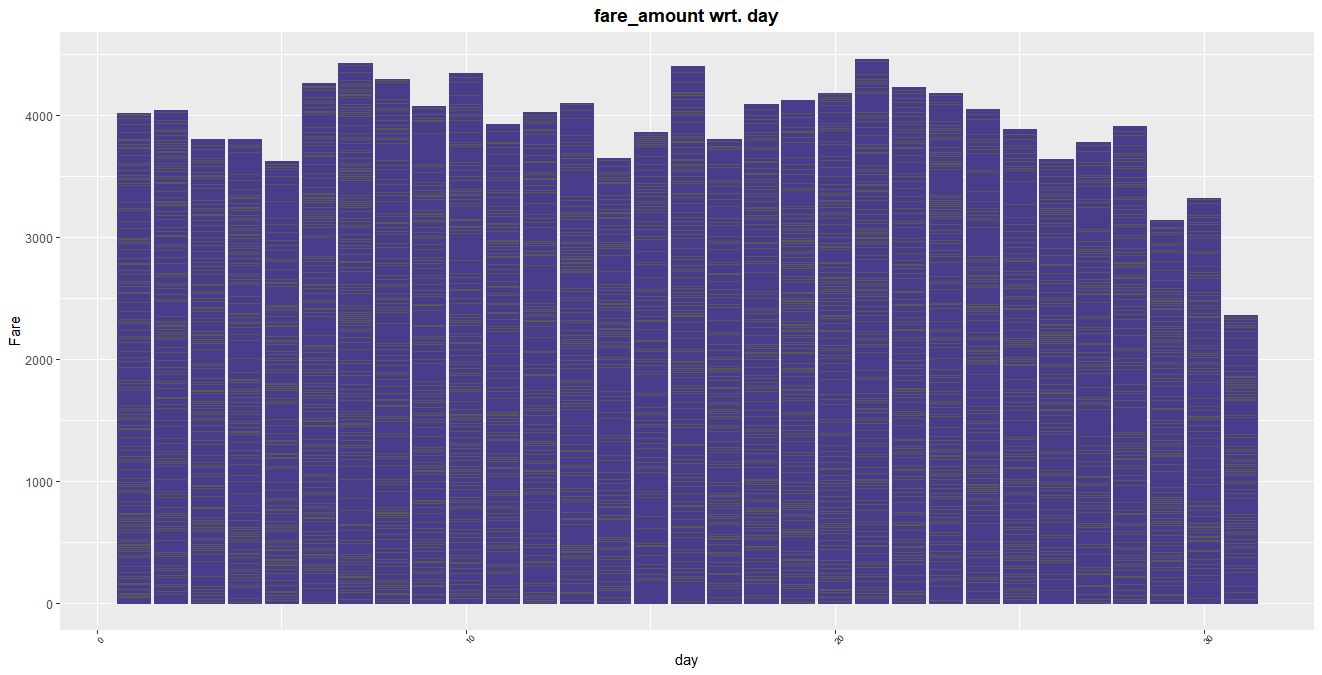


DIAGRAM 5.5 **–** Fare Amount Vs Day Bar graph

**Rplot 330:**

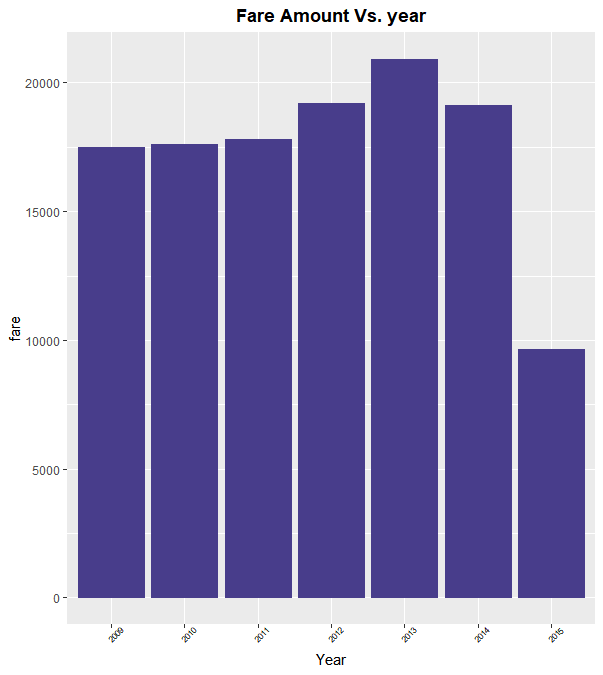


DIAGRAM 5.5 **–** Fare Amount Vs Year Bar graph

**Rplot 365:**

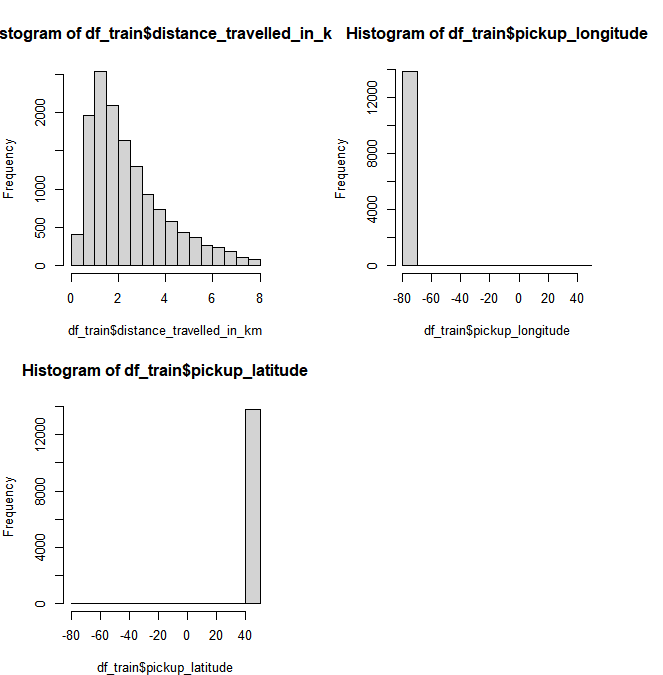


DIAGRAM 5.7 **–** Column-wiseHistogram

**CHAPTER - 6**

**CONCLUSION**

After the selection of the best possible model, it was fit to the large test dataset for which the ‘fare\_amount’ was to be predicted. Data pre- processing was also done on the test data for maximum accuracy. No missing observation was found in missing value analysis. Feature scaling was also done because the original dataset was trained on the scaled data, thus the predicted results would be accurate only if the model fitting is done on the scaled test data. After fitting the model, the results were in the range from 0 to 1 as the whole data was normalized.

**CHAPTER - 7**

**REFERENCES**

1. https://edwisor.com/career-data-scientist
2. <https://www.geeksforgeeks.org/plotting-graph-using-seaborn-python/>

3. <https://santorinidave.com/best-time-to-visit-nyc>

4. <https://www.movable-type.co.uk/scripts/latlong.html>

5. <https://github.com/mukeshsablani3126/Cab-fare-prdiction>