***Cab Fare Prediction***

*Alfazal M*

**Table of Contents**

**Introduction**

**1.1 Problem Statement**

**1.2 Data**

**Methodology**

**2.1 Data Preprocessing**

**2.1.1 Missing Value Analysis**

**2.1.2 Outlier Analysis**

**2.1.3 Feature Selection**

**2.2 Modeling**

**2.2.1 Linear Regression**

**2.2.2 Random Forest**

**2.2.3 Decision Tree**

**Conclusion**

**3.1 Model Evaluation**

**3.1.1 Mean Absolute Percentage Error (MAPE)**

**3.1.2 RSquare Value**

**3.2 Model Selection**

**Appendix**

**1.Introduction**

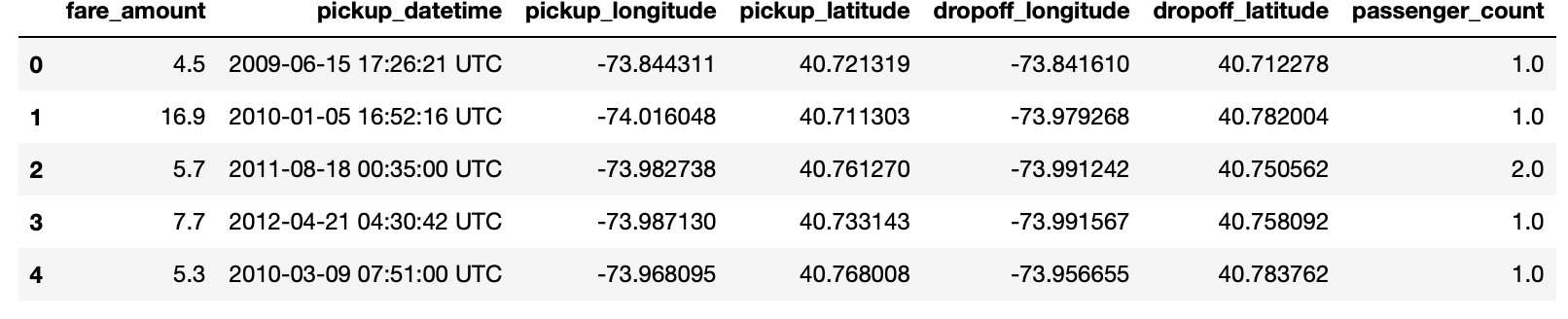
Problem Statement

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.

Objective of the project is to build a model for predicting cabfare and predict the fare amount for a test data set.

Data

Our aim is to develop a model to predict the fare amount(‘fare\_amount’). Given below is the sample of data. Using that data we will develop a model



* Given below are expansion of abbreviated column names:
* · 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

Our target variable is ‘fare\_amount’. Other variables which predict the count variable are:

1.pickup\_datetime

2.pickup\_longitude

3.pickup\_latitude

4.dropoff\_longitude

5.dropoff\_latitude

6.passenger\_count

**2.Methodology**

In methodology data processing, model development and deploying are employed

Data Preprocessing: It includes missing value analysis,outlier analysis,feature engineering,feature selection and feature scaling

Modeling: Various models are deployed on given data and the best model is chosen. And the model is tuned for better accuracy.

Data Preprocessing

We are developing various models using the sample data so the data should be without any missing value and the data should be refined(Some observations are not in the range of the variable's desired range). For that we are doing data preprocessing. Also we can drop some variables which helps to develop the model easily.

Checking Observations are in the desired Range

We know longitude is in the range of -180 to 180 we checked all observations (that are on the train dataset and test dataset)of pickup\_longtude and dropoff\_longitude are in the range of -180 to 180 it is found all longitude observations are in the desired range. Then We checked the range of latitude. While checking the range of pickup\_latitude of the train dataset some pickup\_latitude values are higher than 90 these observations are removed.

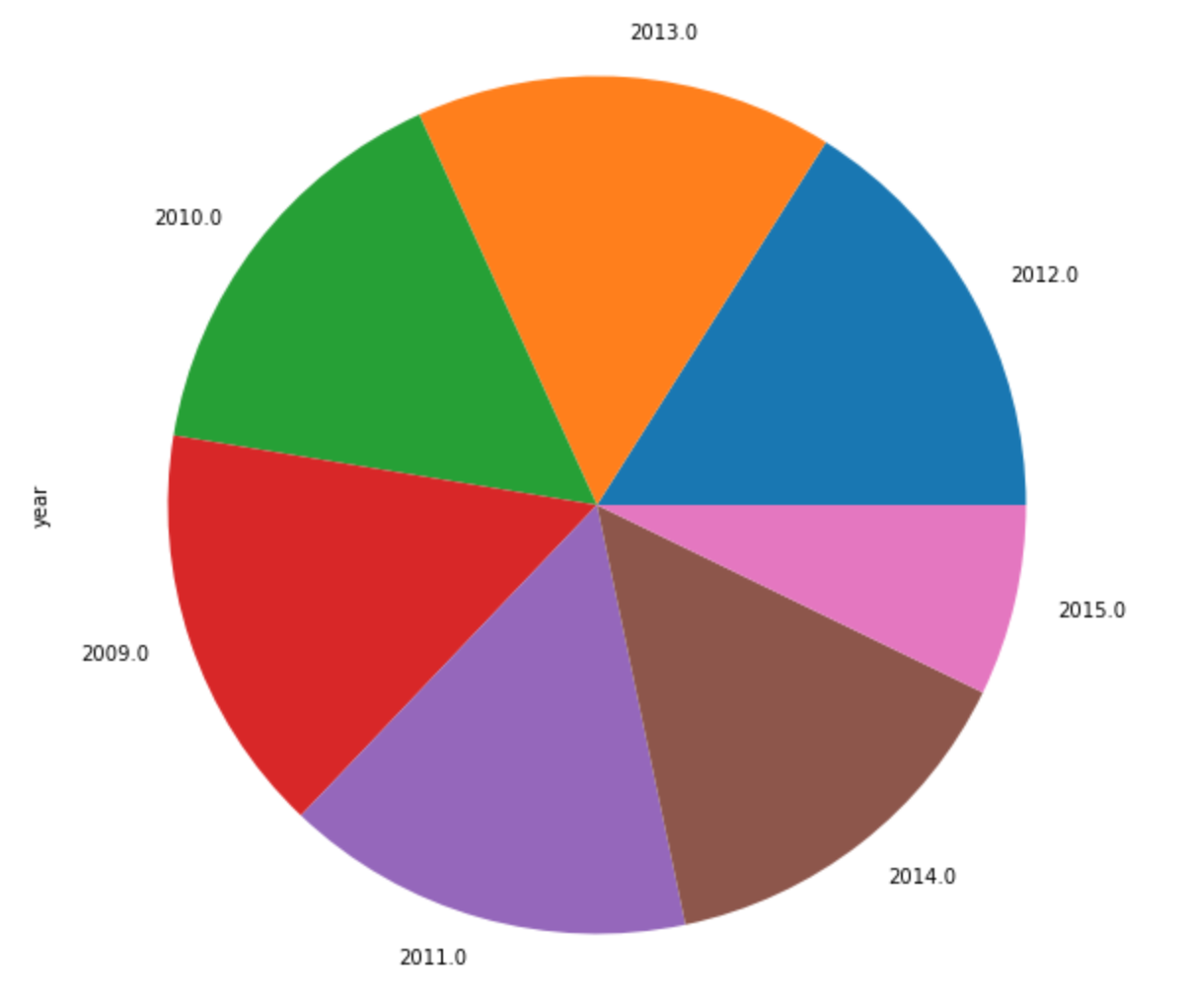
Passenger count should be in the range of 1 to 6. All the records with passenger count not in this range are removed. Fare amount should be greater than 0 those records with fare amount equal to and greater than 0 is removed.

Creating new Variables

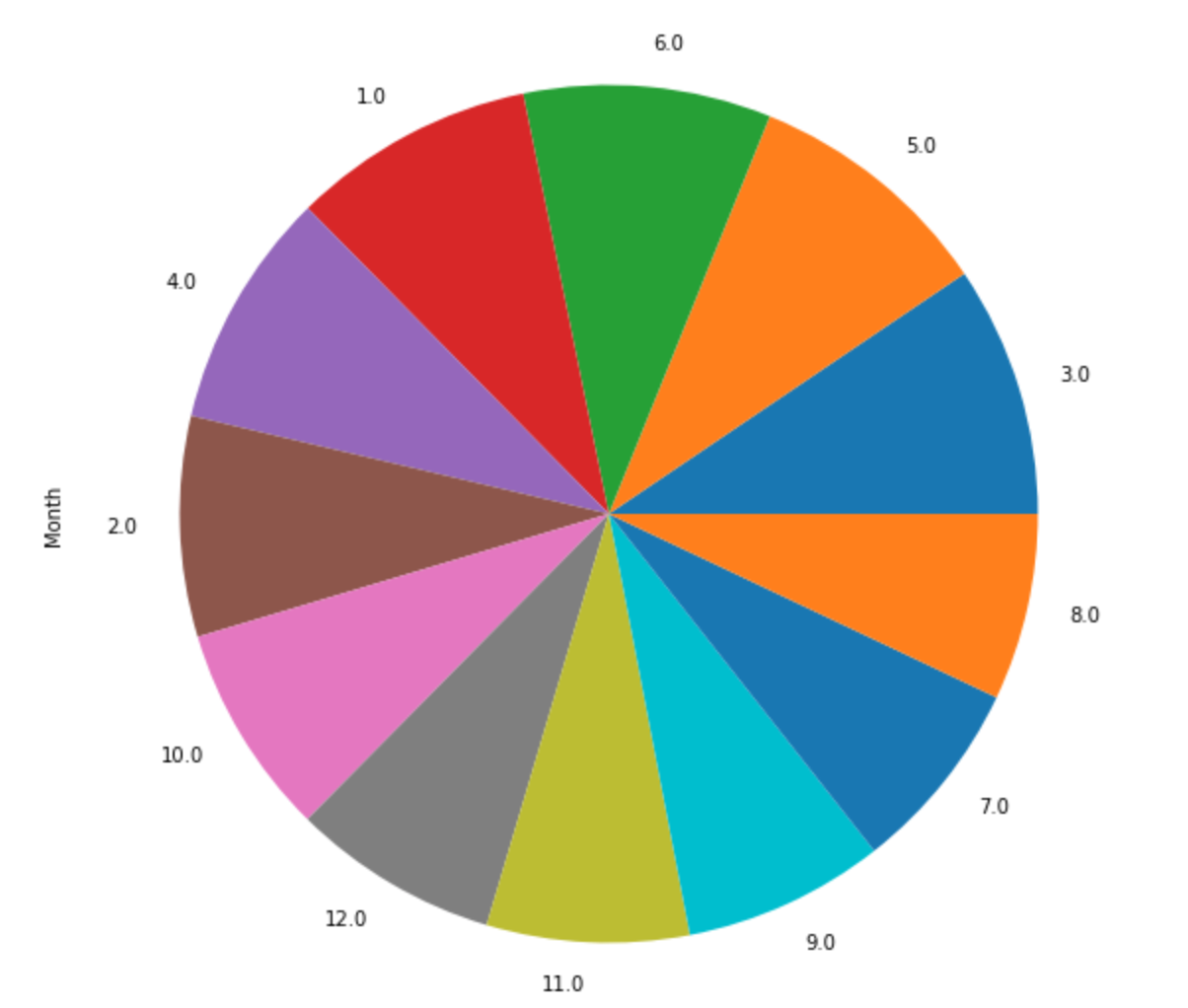
Pickup\_date, pickup\_month, pickup\_year, pickup\_weekday and pickup\_hour is created from the variable pickup\_datetime. We have data of pickup latitude and longitude and drop off latitude and longitude. These data are not related to fare amount directly so distance is calculated using Haversine formula. Another variable time is created using pickup\_hour which gives information whether the pickup is happening at Day, Evening,Night or Midnight.

Data Visualisation

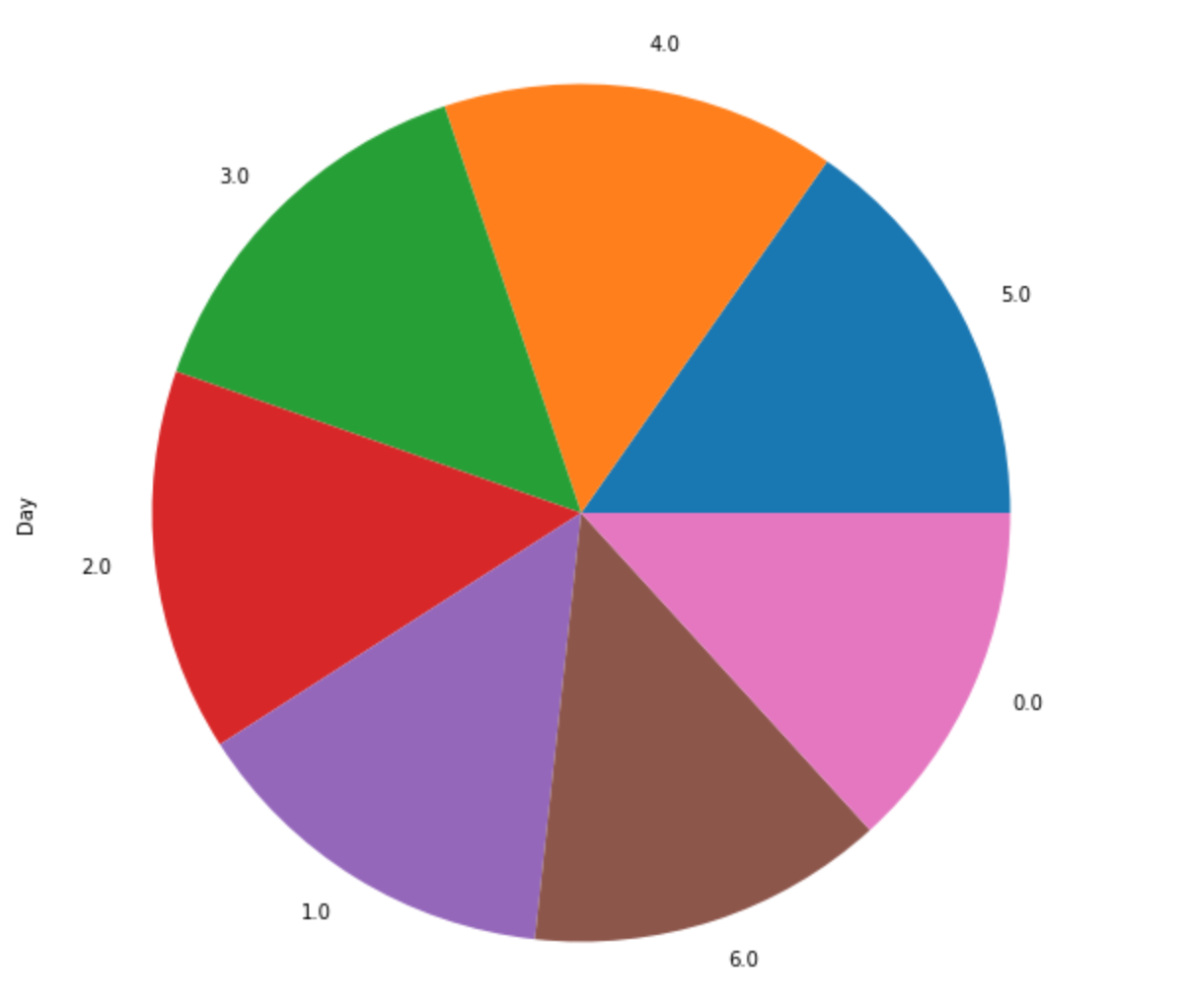
Pie Chart of year



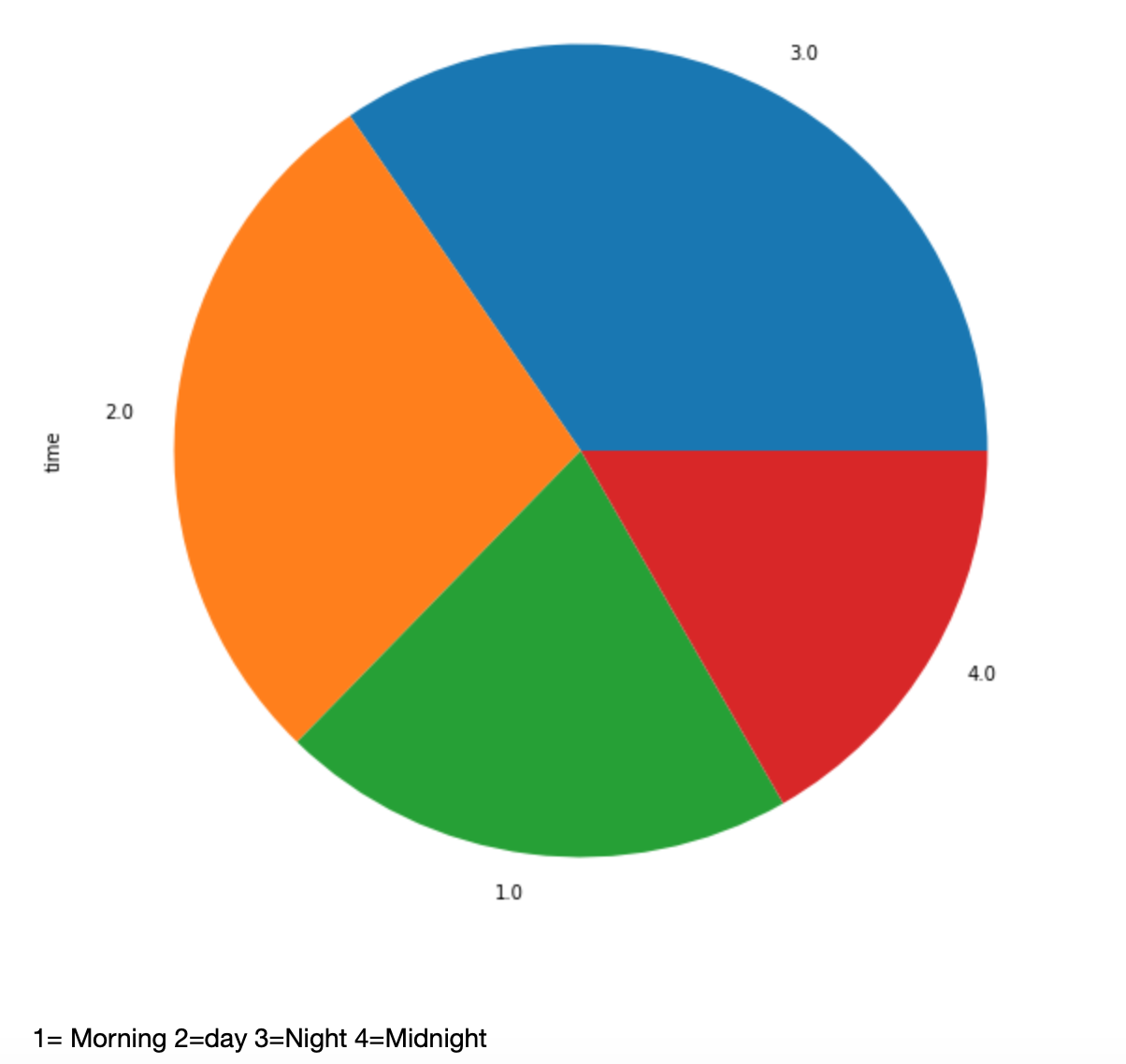
Pie Chart of Month



Pie Chart of Day



Pie Chart of Time



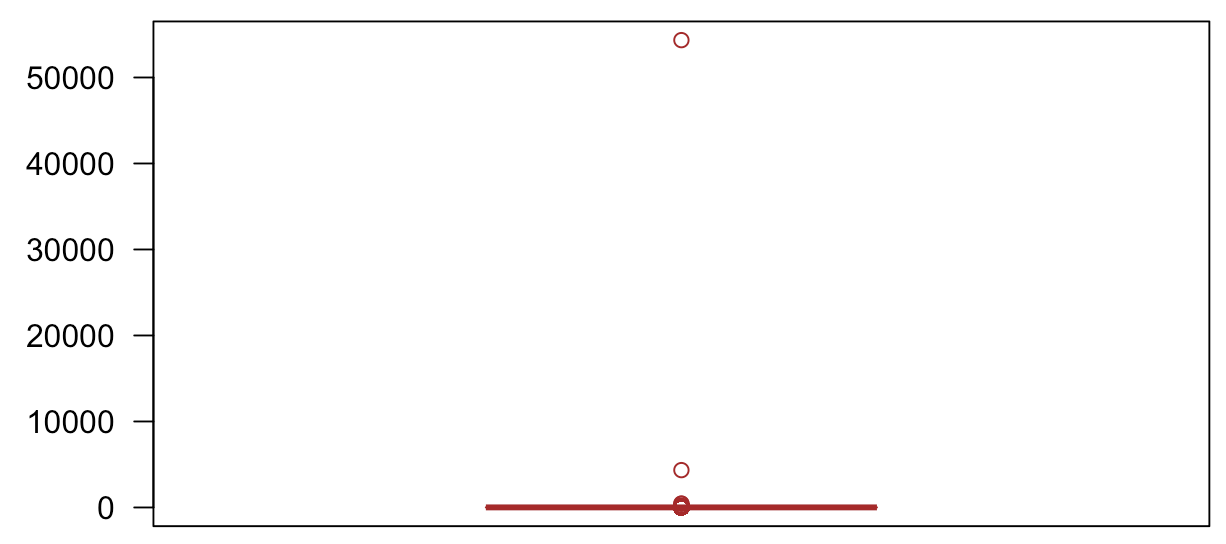
Missing Value Analysis

We are checking whether there is any missing value in the dataset. It is found there are missing values in fare\_amount and passenger\_count. We Randomly selected one observation for fare\_amount and removed the value and imputations are done with mode and knn imputation same methods are done for passenger\_count it is found knn\_impuatation gives more accuracy. So missing values are imputed using knn imputation.

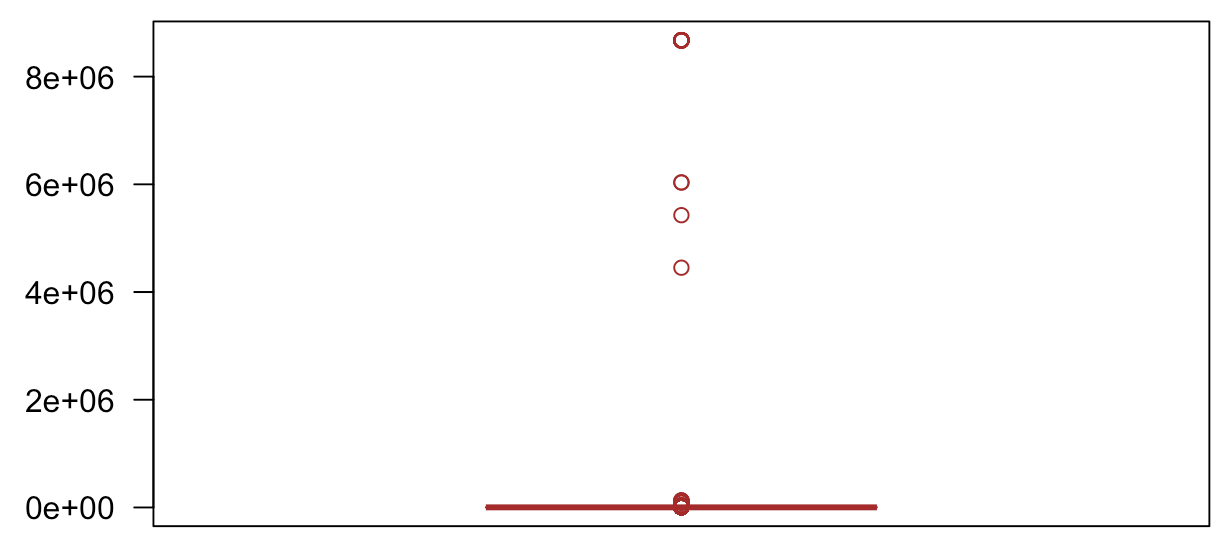
Outlier Analysis

Outliers are those data points which stand outside the overall pattern and distribution of data. Outliers are included in the data because of error of sensors we use to detect the observation, error in entering observations...etc. Outliers are detected in this project using box plots of numeric variables. The box plot of numeric variables are given below.

Box plot of fare amount.



Box Plot of Distance

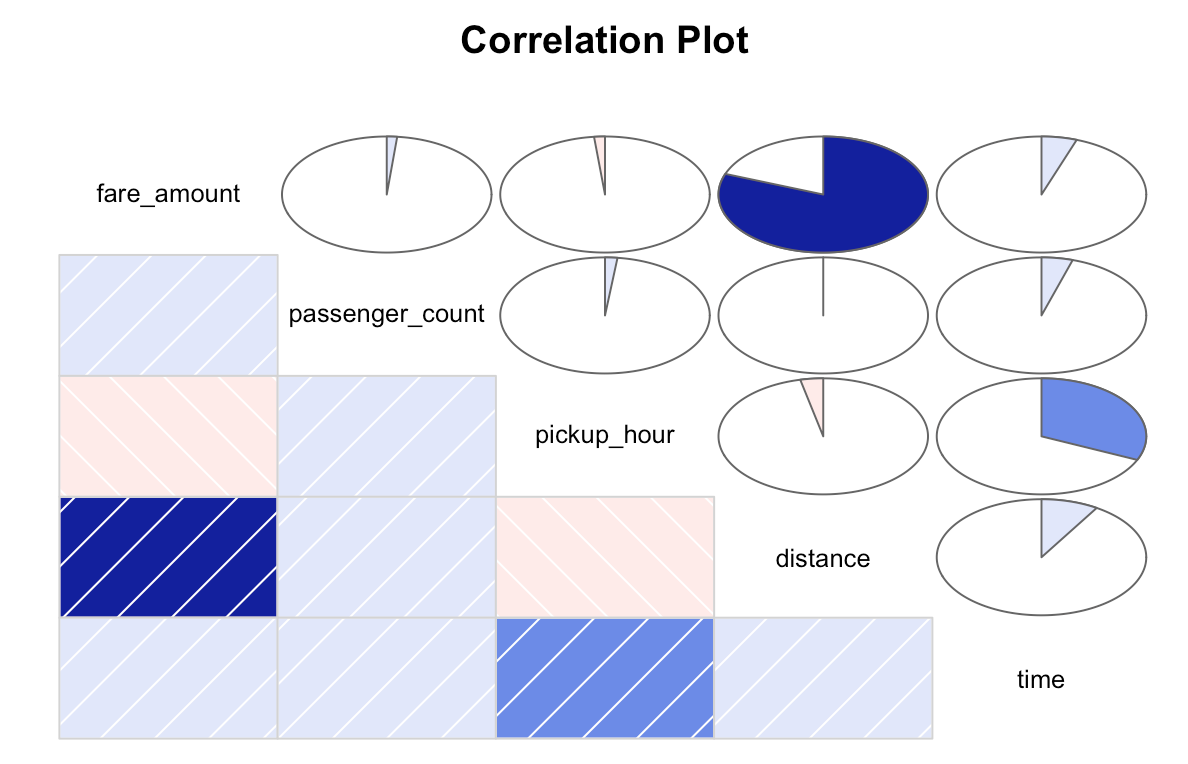


Outliers values are removed and those missing values are imputed using knn imputation.

Feature Selection

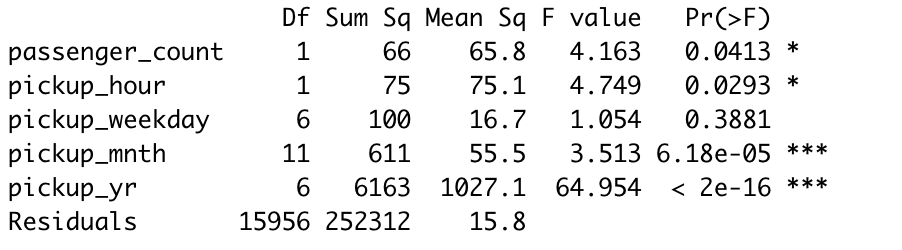
There will be some redundant variables which don't contribute significantly to predict the target variables. These variables should be removed because it will hamper the accuracy of modeling. Also some variables will be highly correlated to each other so one among them is enough need to be considered in model developing. Correlation plot is plotted for numeric variables to check collinearity and anova test is done for categorical variables to check whether target variable has dependency over each categorical variable.VIF values also checked for whether there is multicollinearity.

**Correlation Analysis**

****

There are no variables correlated much with each other.

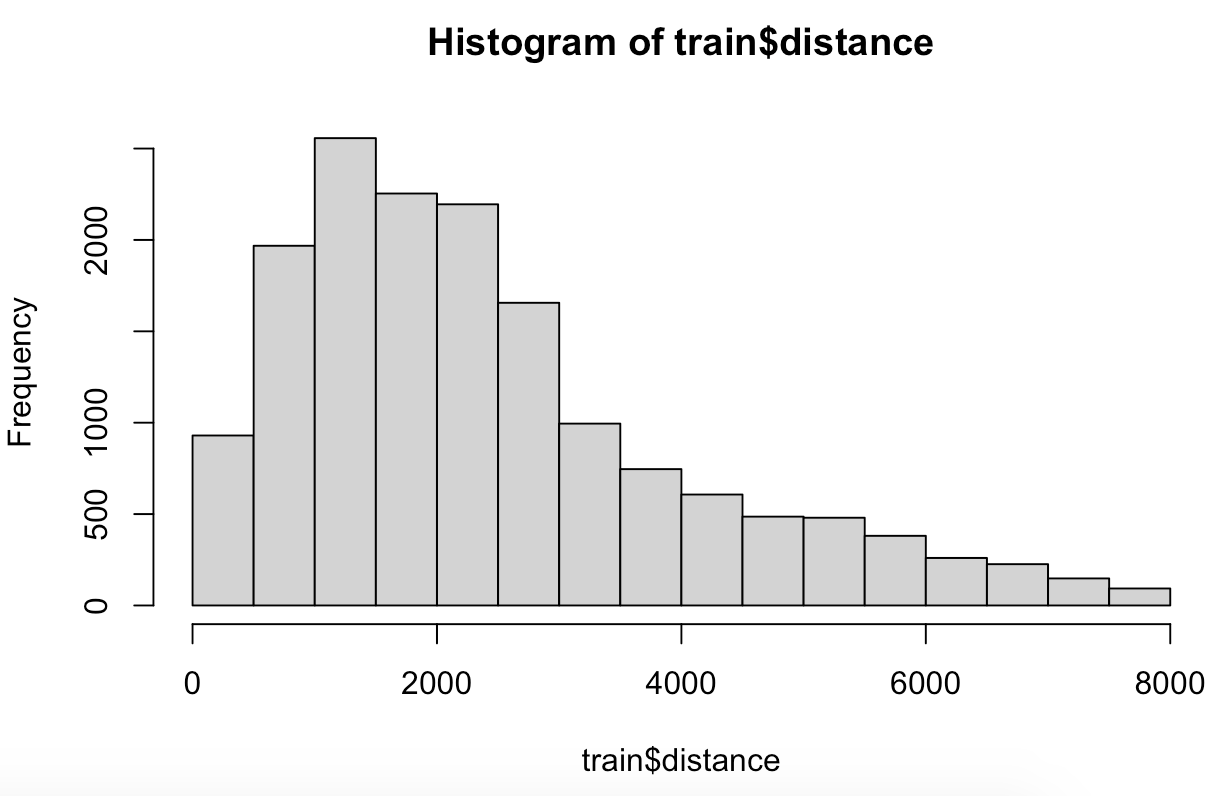
**Anova Test Summary**

****

It is found P values are higher than 0.05 for weekday So null hypothesis is accepted (Target variable doesn’t have dependancy over those variables). So that variable is dropped. It will help to reduce the dimension of data.

Feature Scaling

When we take the histogram of distance it is found it is right skewed.



Normalisation is done for distance variable.

Dropping Unwanted Variables

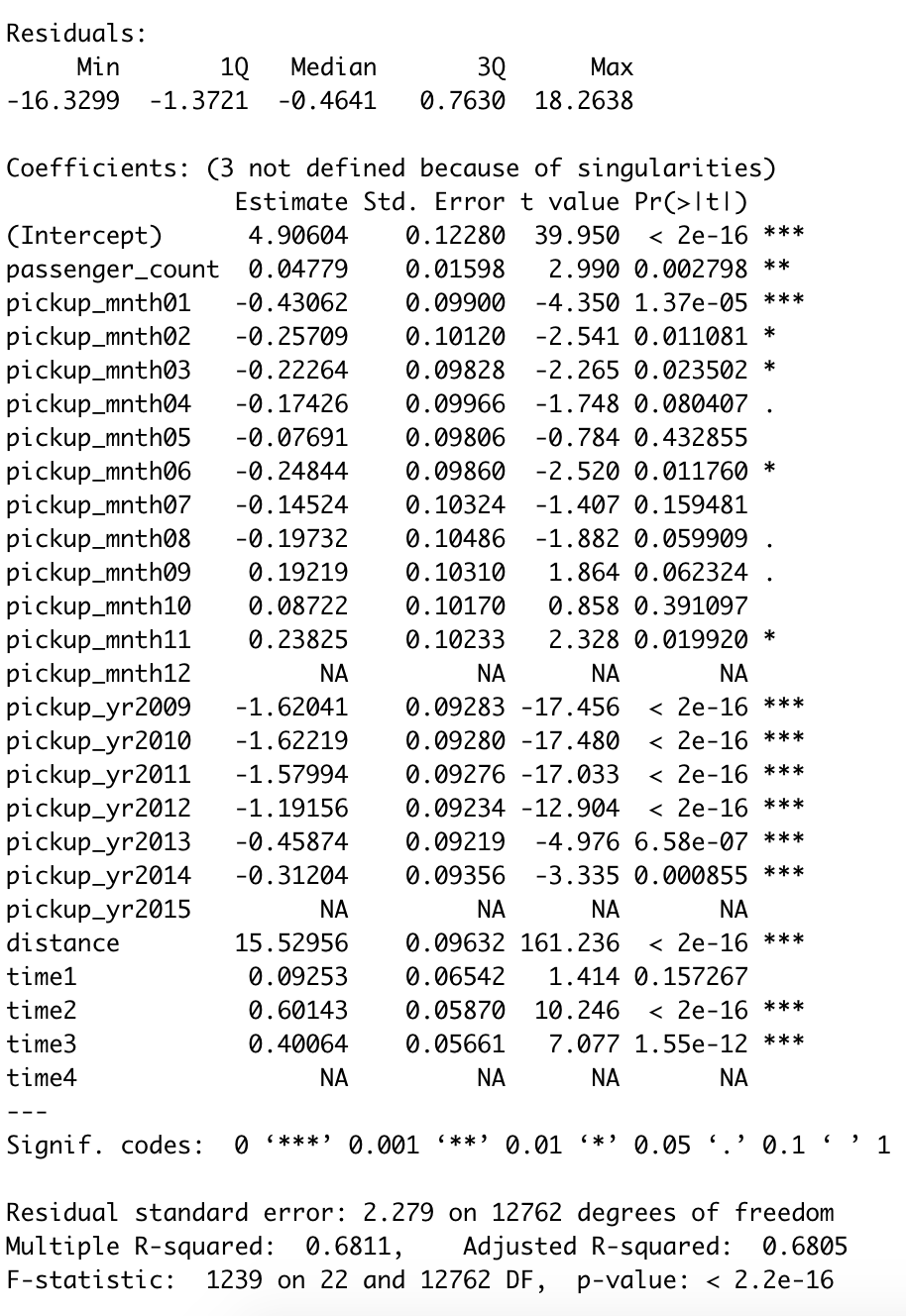
Since we created a new variable distance depending on pickup latitude and longitude and dropoff latitude and longitude. We no longer need pickup\_latitude,pickup\_longitude,dropoff\_latitude and dropoff\_longitude for modelling so we dropped these variables. Also we created variables using pickup\_datetime and pickup\_hour so these variables are removed. Pickup\_date also removed.

Modeling

There are many machine learning algorithms to predict the target variable. Since the target variable is numeric we are trying to predict the target variables using Decision Tree,Random Forest and Linear Regression models. These models are developed using 80 percent of the dataset (Train Data) and we will test it on the rest of data. And the models are compared. The best model is used to predict the fare amount of the dataset ‘test’. Since we are using some datas to model the data it is called supervised learning.

Linear Regression

Linear Regression method is used to predict target variable using one or more than one input variable. Linear regression finds out a linear relationship with input variables and target variables.

Linear Regression in R

Linear Regression in R Summary

MAPE = 19.40557

MAE = 1.6039035

RMSE= 2.2414584

Linear Regression in Python Summary

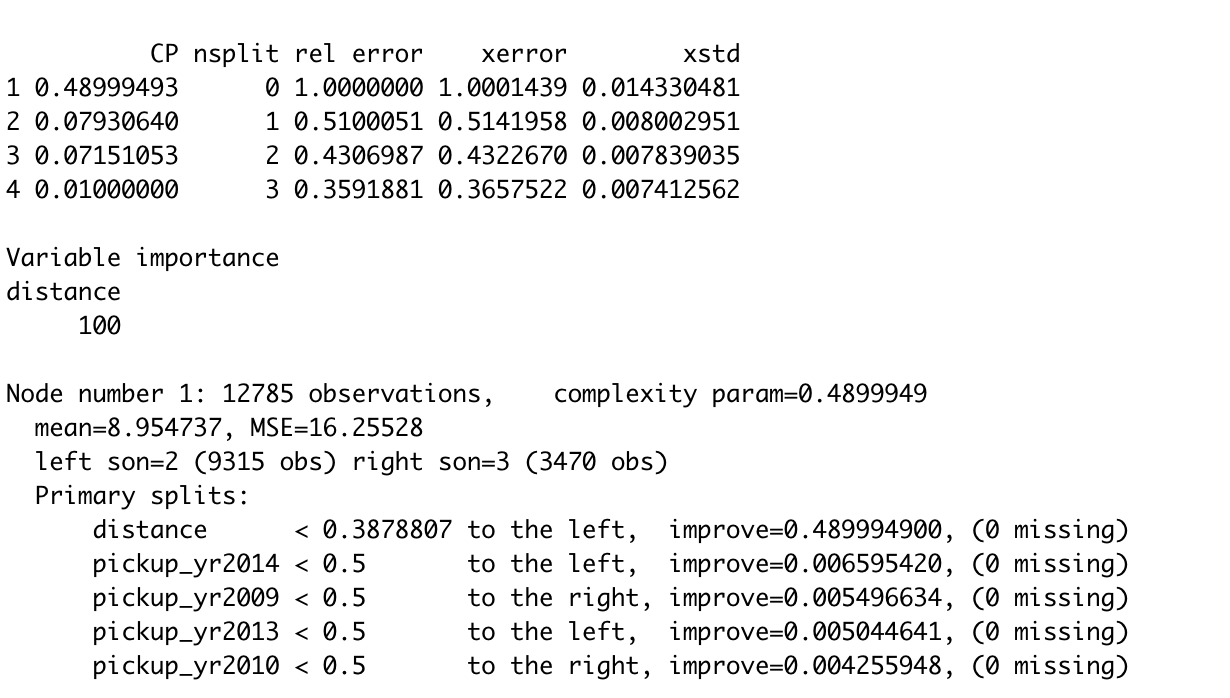
MAPE = 20.451759457170322

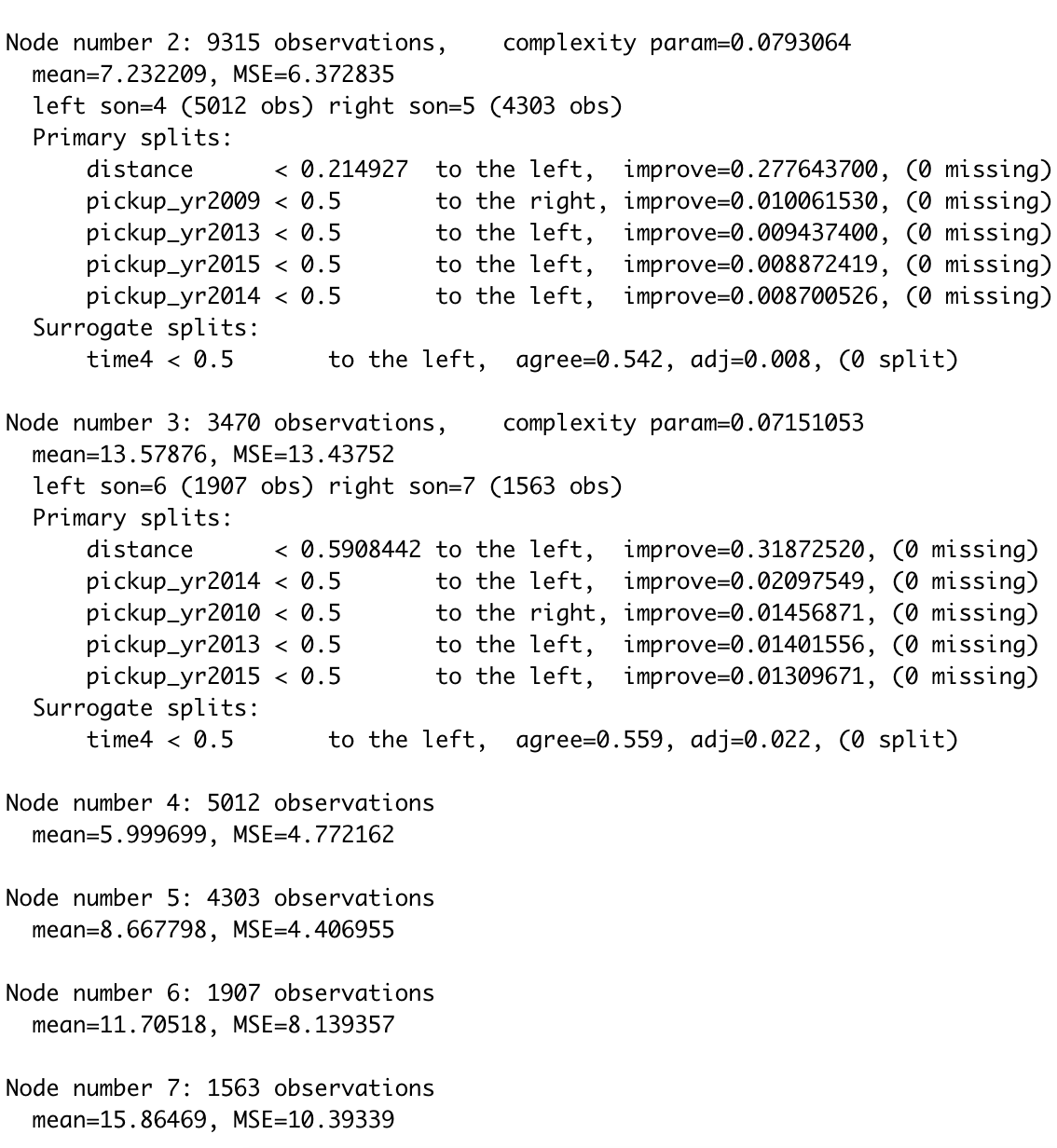
RSquare =0.6096967744976821

Decision Tree

Decision tree is a supervised machine learning algorithm which uses binary rules to predict target variable. Decision tree model is deployed in R and Python

Decision Tree Rules

****

****

The plot shows the splitting of trees. There are 7 nodes. Decision tree model is developed in R and in Python.

Decision Tree in R Summary

MAPE = 20.0223

MAE = 1.662647

RMSE = 2.335862

Decision Tree in Python Summary

MAPE= 22.2957511803962

R Square = .5940745356050848

Random Forest

Random forest is another supervised machine learning algorithm which selects random observations and uses multiple decision trees to predict the target variable. The no. of trees is set as 500 in random forest. Random forest model is deployed in R and Python as well.

Random Forest in R Summary

MAPE = 18.60025

MAE = 1.5376444

RMSE = 2.1471000

Random Forest in Python Summary

MAPE = 19.68234888027742

RSquare =0.6323447146032193

**3.Conclusion**

We have deployed various models (Decision Tree, Random Forest and Linear Regression) on our data. We applied those models on data and compared data prediction with actual data values. From these we measured average error values. By looking at error values we can select a model for bike rent prediction.

MAPE (Mean Absolute Percentage Error)

Mean absolute percentage error is calculated using taking the mean of absolute value of predicted value and actual value and multiplied with 100.

Let’s see the MAPE values for various models.

It is found MAPE value is least in Random Forest method in R and Python.

R Square

R square shows the strength of relation between predicted value and actual value. The higher the R Square value the higher the accuracy of the model. Basically we are taking the square of the correlation of predicted value and actual value to compute R Square value.

R square is highest for Random Forest in Python

Model selection

MAPE is least in Random Forest in and R. So Random Forest is the best suitable method.

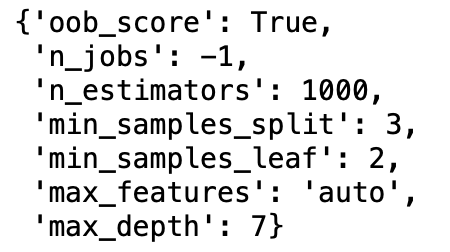
Tuning the Selected Model

It is found that the Random Forest algorithm is the best method to predict target variable. There are many hyperparameters which can be tuned for better accuracy.

In R we tried to find the best ‘mtry’ parameter and used that ‘mtry’ value for prediction but it is observed accuracy decreasing. In python we tried to find all the best parameter value using command ‘RandomizedSearchCV’ and used that parameter value for model making and prediction it is observed accuracy is increased.

Optimised Random Forest Model in Python

Tuned Random Forest’s Parameter values

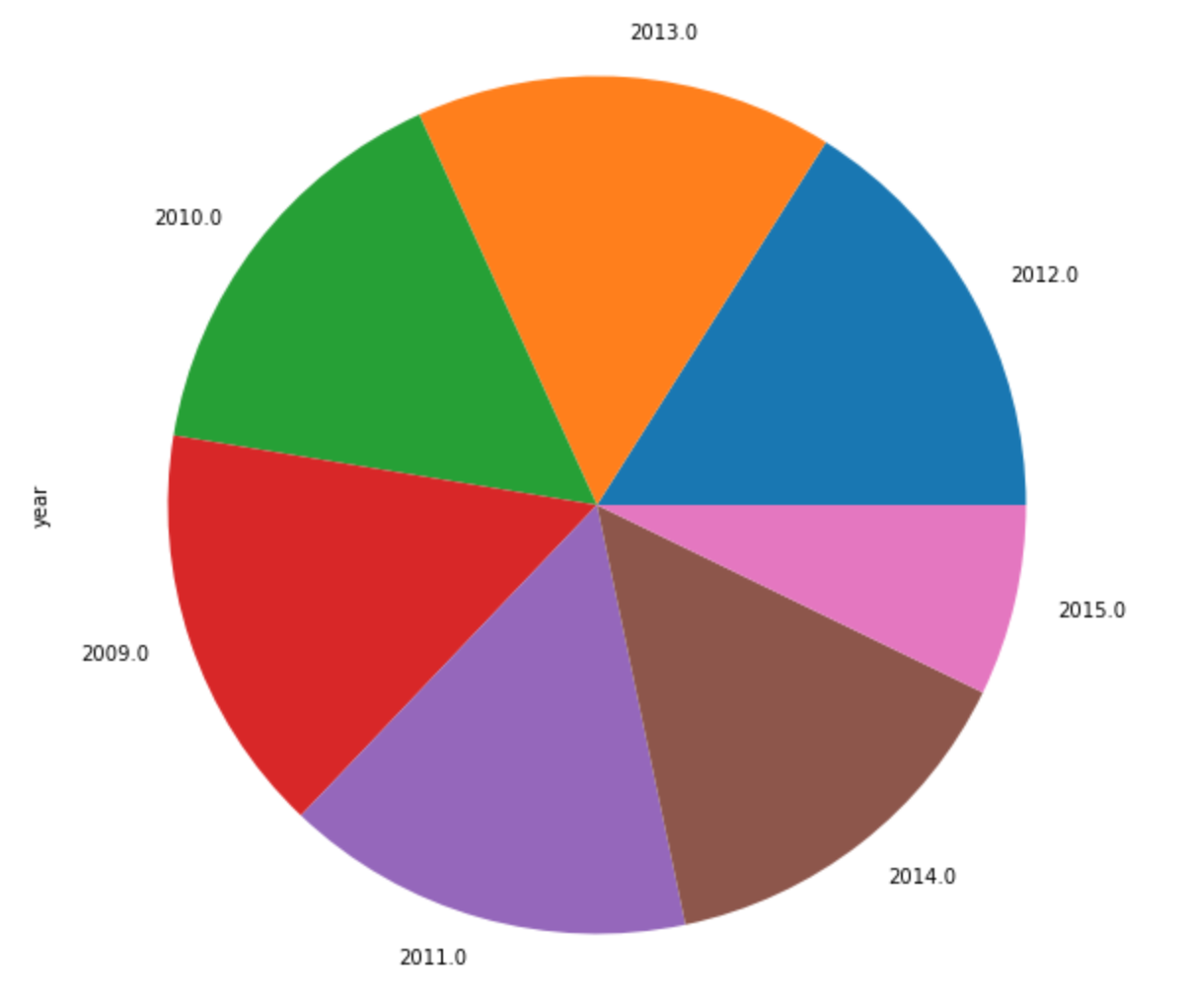


MAPE = 18.766614151460843

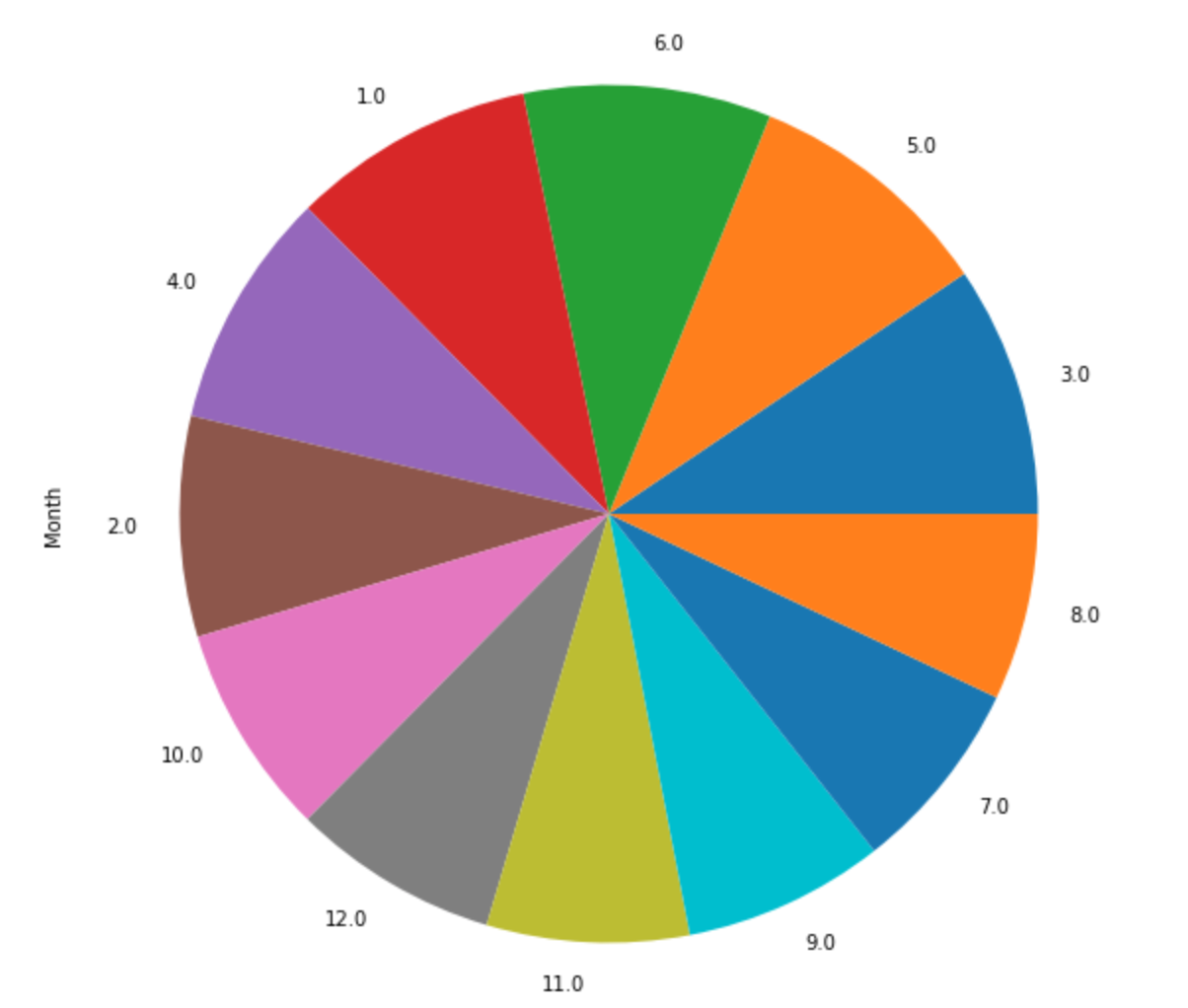
RSquare = 0.6096967744976821

**Appendix**

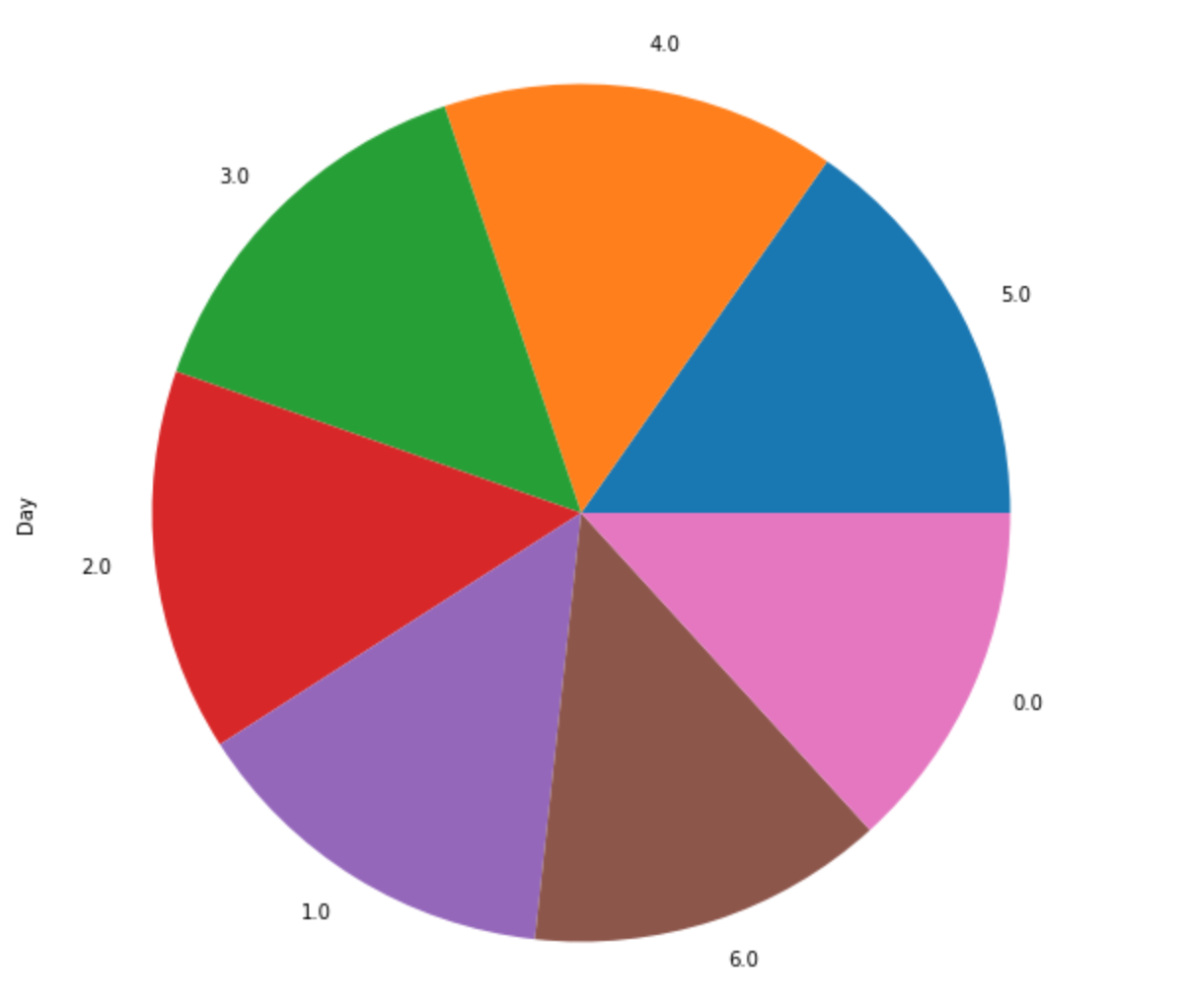
**Pie Chart of year**

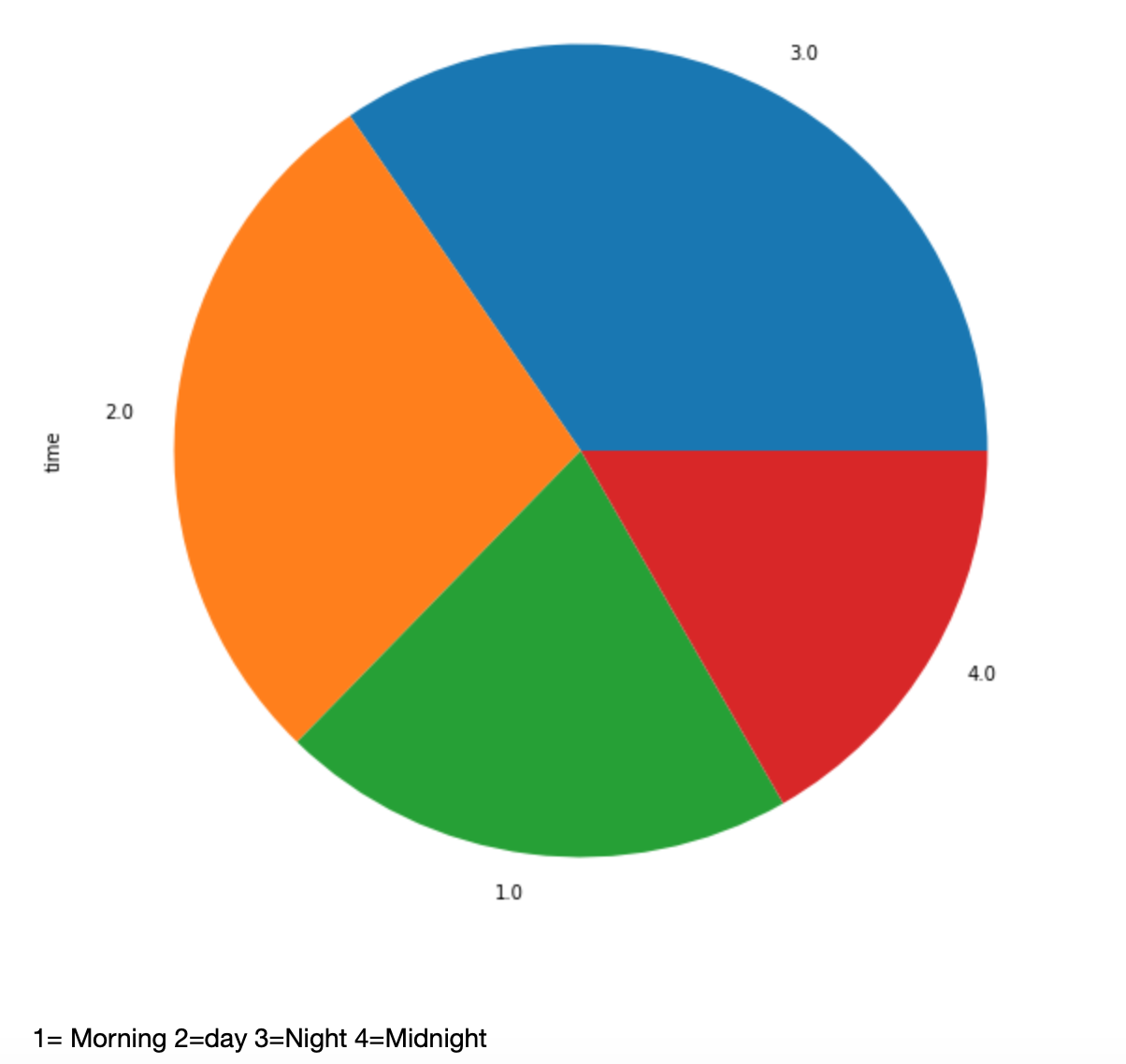


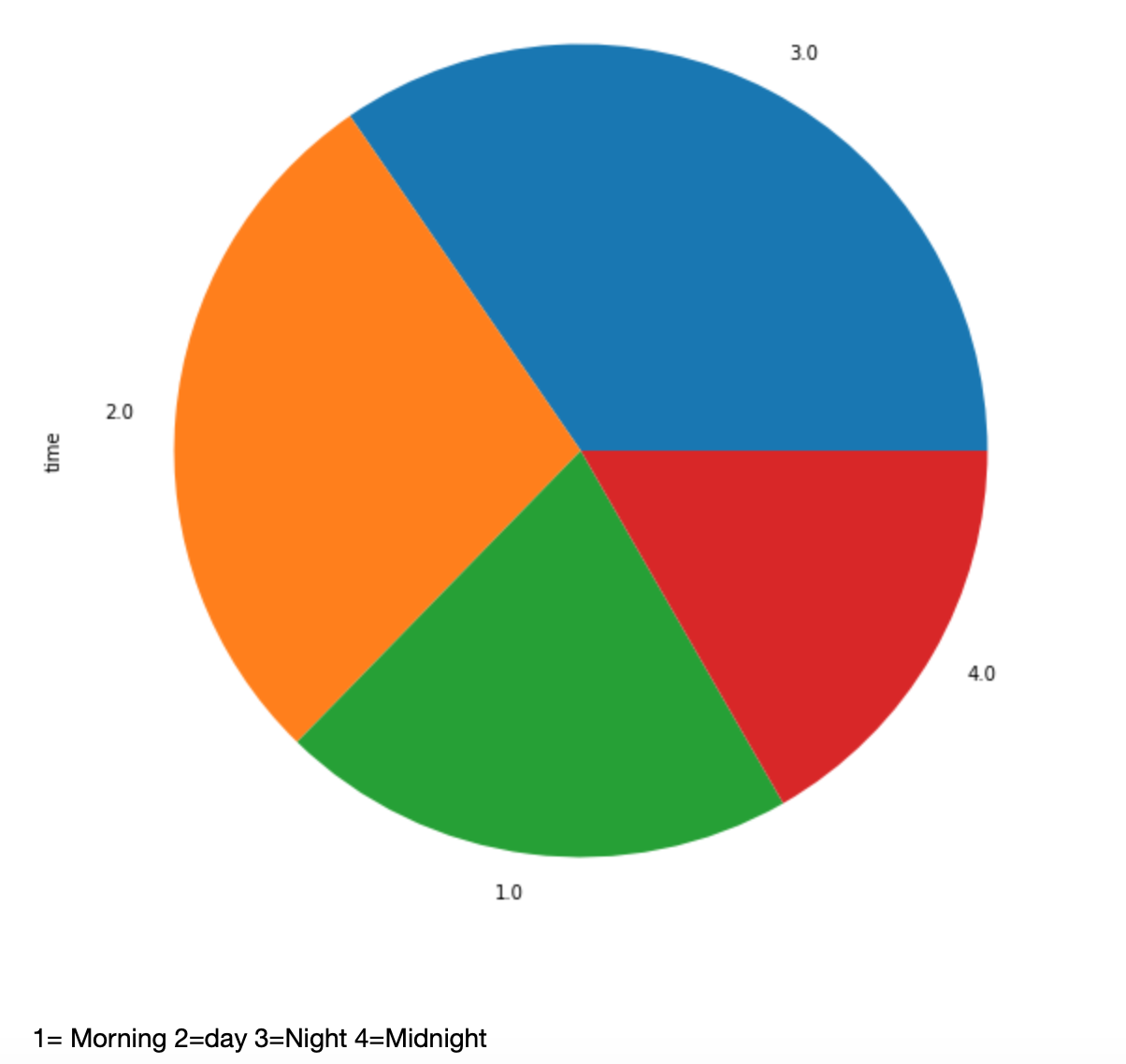
Pie Chart of Month



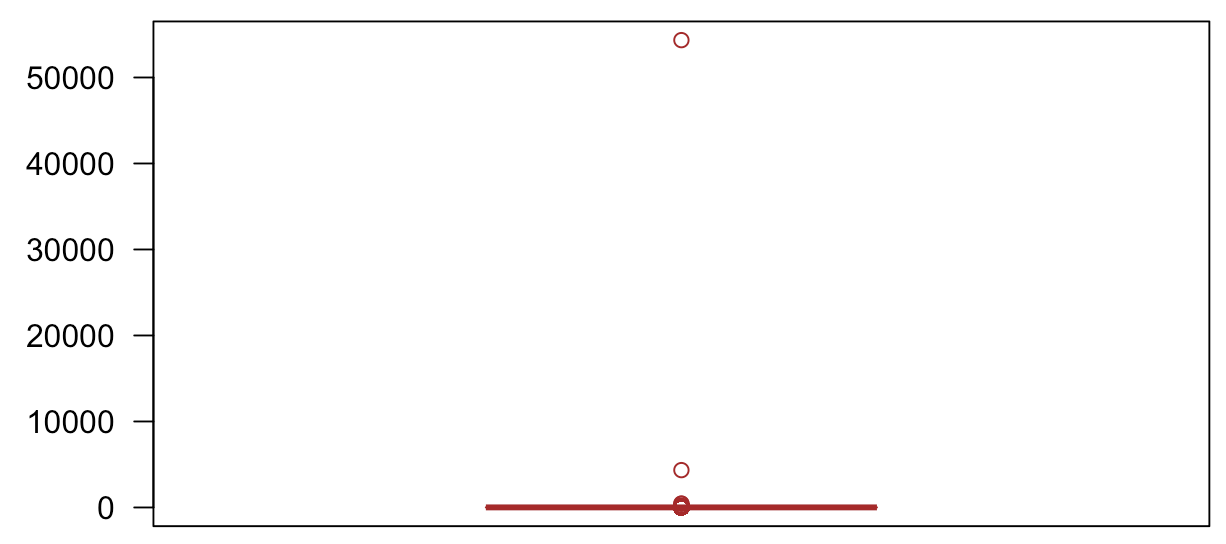
Pie Chart of Day



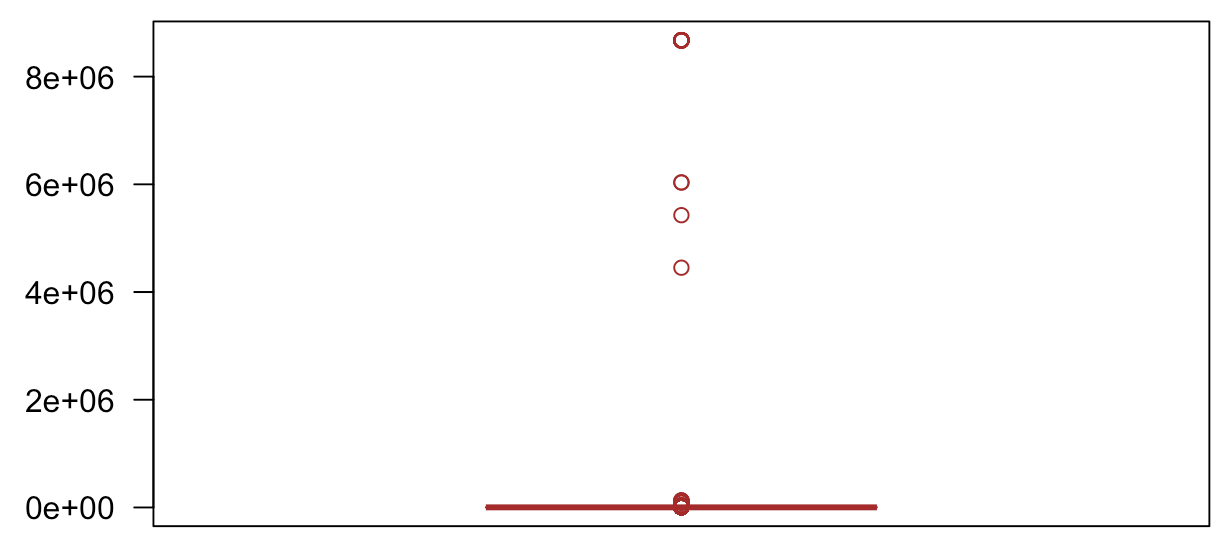
Pie Chart of Time

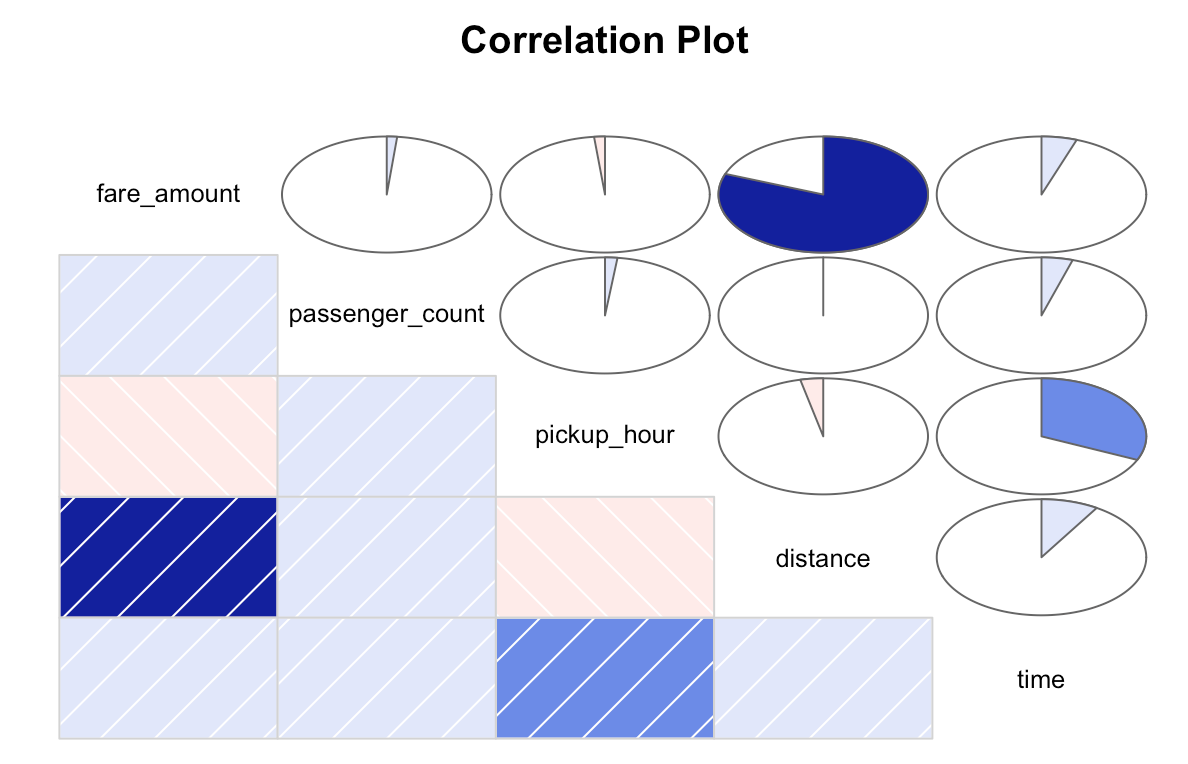


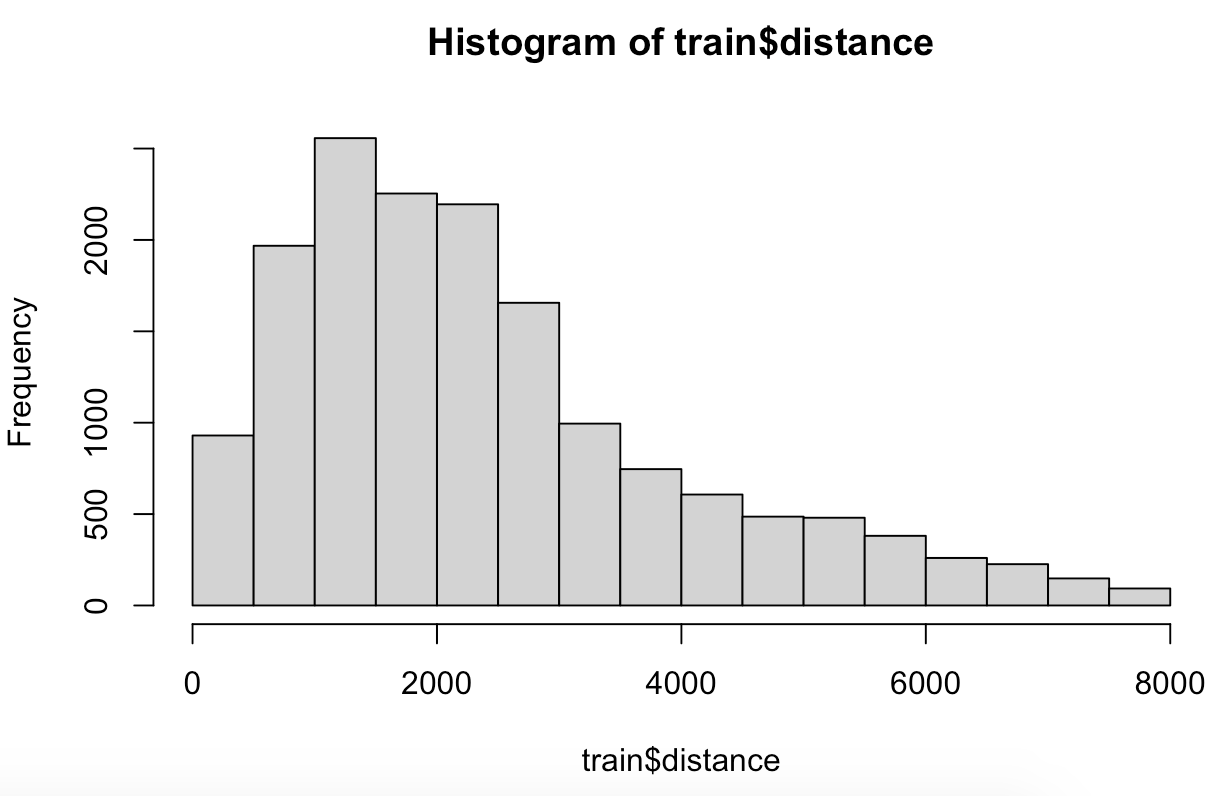
Box Plot of Fare Amount



Box Plot of Distance



****



**R Code**

#clean R environment

rm(list=ls())

#loadinng libraries

x = c("ggplot2", "corrgram", "DMwR", "usdm", "caret", "randomForest", "e1071",

"DataCombine", "doSNOW", "inTrees", "rpart",'MASS','xgboost','stats')

#load Packages

lapply(x, require, character.only = TRUE)

str(train)

library(dummies)

library(tidyverse)

library(caret)

library(usdm)

library(mlbench)

#setting working directory

setwd("/Users/alfazalm/Documents/carfare")

#loading csv fiile

train=read.csv("train\_cab.csv",header = T)

test=read.csv("test.csv",header = T)

test2 = read.csv("test.csv",header = T)

#creating columns with date,day,month,year and hour of pickup from the column "pickup\_date\_time".

train$pickup\_date = as.Date(as.character(train$pickup\_datetime))

train$pickup\_weekday = as.factor(format(train$pickup\_date,"%u"))

train$pickup\_mnth = as.factor(format(train$pickup\_date,"%m"))

train$pickup\_yr = as.factor(format(train$pickup\_date,"%Y"))

train$pickup\_hour = as.factor(format(strptime(train$pickup\_datetime,"%Y-%m-%d %H:%M:%S"),"%H"))

#same thing does for test data

test$pickup\_date = as.Date(as.character(test$pickup\_datetime))

test$pickup\_weekday = as.factor(format(test$pickup\_date,"%u"))

test$pickup\_mnth = as.factor(format(test$pickup\_date,"%m"))

test$pickup\_yr = as.factor(format(test$pickup\_date,"%Y"))

test$pickup\_hour = as.factor(format(strptime(test$pickup\_datetime,"%Y-%m-%d %H:%M:%S"),"%H"))

str(train)

#changing data type of fare amount to numeric

train$fare\_amount=as.numeric(train$fare\_amount)

#fare\_amount should not be zero and less than zero

train=train[-which(train$fare\_amount == 0 ),]

train=train[-which(train$fare\_amount < 0 ),]

#Records with No.of passengers less than 1 shuold be removed

train=train[-which(train$passenger\_count < 1 ),]

#Records with no. of passengers greater than 6 should be removed

train=train[-which(train$passenger\_count > 6 ),]

#Passenger count shoul be integer checking all values are integer

unique(train$passenger\_count)

unique(test$passenger\_count)

#There is a value 1.3 it should be removed

train = train[-which(train$passenger\_count == 1.3),]

unique(train$passenger\_count)

#Latitude should be in range of -90 to 90

#checking the range

range(train$pickup\_latitude)

#There are values grearter than 90 So it need to be removed

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

range(train$dropoff\_latitude)

#The values are in range of -90 to 90

range(test$pickup\_latitude)

#The values are in range of -90 to 90

range(test$dropoff\_latitude)

#The values are in range of -90 to 90

#longitude should be in range of -180 to 180

range(train$pickup\_longitude)

#The values are in range of -180to 180

range(train$dropoff\_latitude)

#The values are in range of -180to 180

range(test$pickup\_longitude)

#The values are in range of -180to 180

range(test$dropoff\_longitude)

#The values are in range of -180to 180

var = c("fare\_amount", "pickup\_datetime","pickup\_longitude","pickup\_latitude","dropoff\_longitude","dropoff\_latitude",

"passenger\_count","pickup\_date","pickup\_weekday","pickup\_mnth","pickup\_yr","pickup\_hour")

####Missing Value Analysis####

missing\_value = data.frame(apply(train,2,function(x){sum(is.na(x))}))

missing\_value$percentage = 0

missing\_value$percentage = (missing\_value[1]/nrow(train)) \* 100

names(missing\_value)[1] = "Missing\_values"

names(missing\_value)[2] = "Percentage"

# There is no missing value in "Pickup\_datetime" but there are missing values in the column we created from the same column

#let's checkc it

mi = which (is.na(train$pickup\_date))

#let's check the pickup\_datetime of that particular observation with pickup\_date null value

train$pickup\_datetime[mi]

#it's found pickup\_datetime is 43 for an observation that should be removed

train=train[-(mi),]

#Now check the missing values and missing percentage once again

missing\_value = data.frame(apply(train,2,function(x){sum(is.na(x))}))

missing\_value$percentage = 0

missing\_value$percentage = (missing\_value[1]/nrow(train)) \* 100

names(missing\_value)[1] = "Missing\_values"

names(missing\_value)[2] = "Percentage"

####Missing value imputation for passenger count

# Mode method and knn imputation will be checked for a sample value and the best method will be used to impute the missing value

#taking random index of 500 and value and both method will be checked

value\_p = train$passenger\_count[500]

getmode <- function(v) {

uniqv <- unique(v)

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

}

mode\_p = getmode(train$passenger\_count)

train2 = train

str(train2)

#removing some character and factor columns for knnImputation

train2$passenger\_count[500] = NA

train2 = subset(train2, select = -c(pickup\_datetime,pickup\_date,pickup\_weekday,pickup\_mnth,pickup\_yr))

train2 = knnImputation(train2, k = 181)

knn\_p = train2$passenger\_count[500]

# Actual value =3

# mode = 1

#knn imputation = 2.39

# mode gives value of 1 and knn imputation gives value 2.39 and actual value is 3 so knn\_imputation is used for imputing missing value

train$passenger\_count = train2$passenger\_count

#passenger count should be integers so we will round the value

train$passenger\_count=round(train$passenger\_count)

#####Missing value imputation for fare amount

value\_f = train$fare\_amount[500]

train2 = train

mode\_f = getmode(train$fare\_amount)

train2$passenger\_count[500] = NA

train2 = subset(train2, select = -c(pickup\_datetime,pickup\_date,pickup\_weekday,pickup\_mnth,pickup\_yr))

train2 = knnImputation(train2, k = 181)

knn\_f = train2$fare\_amount[500]

# Actual value =6

# mode = 6.5

#knn imputation = 6

# mode gives value of 6.5 and knn imputation gives value 6 and actual value is 6 so knn\_imputation is used for imputing missing value

train$fare\_amount = train2$fare\_amount

#outlier analysis is done after converting latitude,longitude variables

#to distance because distance is used for modelling

###########Converting latitude and longitude variable to distance

my\_dist <- function(long1, lat1, long2, lat2) {

rad = pi/180

a1 = lat1\*rad

a2 = long1\*rad

b1 = lat2\*rad

b2 = long2\*rad

dlon = b2 - a2

dlat = b1 - a1

a = (sin(dlat/2))^2 + cos(a1)\*cos(b1)\*(sin(dlon/2))^2

c= 2\*atan2(sqrt(a), sqrt(1 - a))

R =6378137

d = R\*c

return(d)

}

train$distance = my\_dist(train$pickup\_longitude,train$pickup\_latitude,train$dropoff\_longitude,train$dropoff\_latitude)

test$distance = my\_dist(test$pickup\_longitude,test$pickup\_latitude,test$dropoff\_longitude,test$dropoff\_latitude)

###outlier Analysis

boxplot(train$fare\_amount,

names = c("Fare Amount"),

las = 2,

col = c("blue"),

border = "brown",

horizontal = FALSE,notch = FALSE

)

boxplot(train$distance,

names = c("Distance"),

las = 2,

col = c("blue"),

border = "brown",

horizontal = FALSE,notch = FALSE

)

#There are outliers in fare amount and distance

outliers\_f=boxplot(train$fare\_amount, plot=FALSE)$out

outliers\_d=boxplot(train$distance, plot=FALSE)$out

#put NA values in outliers

train[,'fare\_amount'][train[,'fare\_amount'] %in% outliers\_f] = NA

train[,'distance'][train[,'distance'] %in% outliers\_d] = NA

#imputing in outlier values

train3 = subset(train, select = -c(pickup\_datetime,pickup\_date,pickup\_weekday,pickup\_mnth,pickup\_yr))

train3 = knnImputation(train3, k = 181)

train$fare\_amount=train3$fare\_amount

train$distance=train3$distance

#checking whether there is missing values

sum(is.na(train))

##Adding new feature time depending on pickup hour column

train$pickup\_hour=as.integer(train$pickup\_hour)

test$pickup\_hour=as.integer(test$pickup\_hour)

train$time[train$pickup\_hour>=4 & train$pickup\_hour<=10]=1 #'Morning'

train$time[train$pickup\_hour>10 & train$pickup\_hour<=16]=2 #'Day'

train$time[train$pickup\_hour>16 & train$pickup\_hour<=22]=3 #'Night'

train$time[train$pickup\_hour>22 | train$pickup\_hour<5]=4 #'Midnight'

test$time[test$pickup\_hour>=4 & test$pickup\_hour<=10]=1 #'Morning'

test$time[test$pickup\_hour>10 & test$pickup\_hour<=16]=2 #'Day'

test$time[test$pickup\_hour>16 & test$pickup\_hour<=22]=3 #'Night'

test$time[test$pickup\_hour>22 | test$pickup\_hour<5]=4 #'Midnight'

##Removing variables that we used for feature engineering

train = subset(train,select = -c(pickup\_longitude,pickup\_latitude,dropoff\_longitude,dropoff\_latitude))

test = subset(test,select = -c(pickup\_longitude,pickup\_latitude,dropoff\_longitude,dropoff\_latitude))

train = subset(train,select = -c(pickup\_datetime))

test = subset(test,select = -c(pickup\_datetime))

###Feature selection

##Correlation plot for numeric variables

n = sapply(train,is.numeric)

numeric = train[,n]

cnames = colnames(numeric)

corrgram(train[,cnames], order = F,

upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

##There is no enough correlation to remove variables

###Anova test for categorical variables

aov\_results = aov(fare\_amount ~ passenger\_count + pickup\_hour + pickup\_weekday + pickup\_mnth + pickup\_yr,data = train)

summary(aov\_results)

#Pickup\_weekday has a p value greater than 0.05, so null hypotheis true and the variable will be omitted

train = subset(train,select=-pickup\_weekday)

test = subset(test,select=-pickup\_weekday)

#####Feature Scaling

hist(train$fare\_amount)

hist(train$distance)

#Distance is right skewed Normalisation required

train[,'distance'] = (train[,'distance'] - min(train[,'distance']))/

(max(train[,'distance'] - min(train[,'distance'])))

#Drop unwanted variables

train = subset(train,select=-c(pickup\_hour,pickup\_date))

test = subset(test,select=-c(pickup\_hour,pickup\_date))

#####Modeling####

#Adding Dummy variables for categorical and factor variables for better performance of model

categ2 = c('time','pickup\_yr','pickup\_mnth')

train = dummy.data.frame(train, categ2)

test = dummy.data.frame(test, categ2)

##Splitting train data to 80:20 ratio for model performance checking

train\_index=sample(1:nrow(train),0.8\*nrow(train))

train\_80= train[train\_index,]

train\_20= train[-train\_index,]

#Linear Regression

vif(train[,-1])

vifcor(train[,-1],th=0.8)

# train\_20 values without target variable

train\_20\_var= subset(train\_20, select=-fare\_amount)

##It shows there is no collinearity problem

model\_lr = lm(fare\_amount~., train\_80)

predictions\_lr = predict(model\_lr,train\_20\_var)

regr.eval(train\_20$fare\_amount,predictions\_lr)

#mae mse rmse mape

#1.5840254 5.1925158 2.2787092 0.1918588

###Decision Tree

model\_dt = rpart(fare\_amount ~ .,data=train\_80,method="anova")

predictions\_dt = predict(model\_dt,train\_20\_var)

regr.eval(train\_20$fare\_amount,predictions\_dt)

#mae mse rmse mape

#1.7557661 5.9000076 2.4289931 0.2201762

#Random Forest

model\_rf = randomForest(fare\_amount~., train\_80, ntree = 500, importance = TRUE)

predictions\_rf = predict(model\_rf,train\_20\_var)

regr.eval(train\_20$fare\_amount,predictions\_rf)

#mae mse rmse mape

#1.5664307 4.9213228 2.2184055 0.1915134

##Best method is random forest where accuracy is better

##We are tuning the random forest algorithm to find out the best mtry value using algorithm tools

seed = 7

metric = "Accuracy"

x1=train\_80[,-1]

x2=train\_80[,1]

# Algorithm Tune (tuneRF)

set.seed(seed)

bestmtry= tuneRF(x1, x2, stepFactor=1.5, improve=1e-5, ntree=500)

print(bestmtry)

### error is less with mtry value 8

model\_rf2 = randomForest(fare\_amount~., train\_80, ntree = 500,mtry = 8, importance = TRUE)

predictions\_rf2 = predict(model\_rf2,train\_20\_var)

regr.eval(train\_20$fare\_amount,predictions\_rf2)

##it is found with mtry value 8 accuracy decreasing so the default random forest method is more accurate

###########Finalising model and saving model and applying on test dataset######

test\_pickup\_datetime = test2$pickup\_datetime

predictions\_rf3 = predict(model\_rf,test)

predictions = data.frame(test\_pickup\_datetime,"predictions" = predictions\_rf3)

# save the predicted fare\_amount in disk as .csv format

write.csv(predictions,"predictions\_R1.csv",row.names = FALSE)