```
#clearing Envorinment
rm(list=ls())

#Setting Working Directory
setwd("E:/data science and machine learning/CAB project 1/R")

#Checking Working Directory
getwd()

#Importing CSV file
cabdf=read.csv("train_cab.csv",header=T)

View(cabdf)

#Getting Colnames
colnames(cabdf)

#checking datatype of each variable
str(cabdf)

#converting fare amount variable to numeric
cabdf$fare_amount=as.numeric(as.character(cabdf$fare_amount))

#checking datatype of fare amount
class(cabdf$fare_amount)

#Converting time stamp into POSIXct
cabdf$pickup_datetime=as.POSIXct(strptime(cabdf$pickup_datetime,"%Y-%m-%d %H:%M:%S"))

#checking Class
class(cabdf$pickup_datetime)

#Extracting years,month & weedays number from time
cabdf$year=as.numeric(format(cabdf$pickup_datetime,"%Y"))
cabdf$month=as.numeric(format(cabdf$pickup_datetime,"%m"))

#1-Monday to 7-Sunday
cabdf$wday=strftime(cabdf$pickup_datetime,"%u")

#converting character to numeric
cabdf$wday= as.numeric(as.character(cabdf$wday))

cabdf$pickup_datetime=NULL

#loading some of the libraries
#x=c("ggplot2","Corrgram","DMwR","Caret","randomForest","unbalanced","C50","dummies","e10","MASS","rpart","gbm","ROSE")
#lapply(x,require,character.only=TRUE)

#finding out number of missing value
missing_val=data.frame(apply(cabdf,2,function(x){sum(is.na(x))}))

#giving names in dataframe
missing_val$Columns=row.names(missing_val)
row.names(missing_val)=NULL
names(missing_val)[1]="Missing_Percentage"

#converting to percentage
missing_val$Missing_Percentage=(missing_val$Missing_Percentage/nrow(cabdf))*100

#Arranging in descending order
missing_val=missing_val[order(-missing_val$Missing_Percentage),]

#MISSING VALUE ANALYSIS(checking correct method)

#reference NA to check best method for passenger count
cabdf[5088,6]=NA

#Mean method
cabdf$passenger_count[is.na(cabdf$passenger_count)]=mean(cabdf$passenger_count,na.rm =T)
#Actual=2
#Analysis=2.62

#Median method
cabdf$passenger_count[is.na(cabdf$passenger_count)]=median(cabdf$passenger_count,na.rm =T)
#Actual=2
#Analysis=1

#Knn method(DMwR library)
cabdf=knnImputation(cabdf,k=5)
#Actual=2
#Analysis=2.60

#reference NA to check best method for fare
cabdf[8444,1]=NA

#Mean method
cabdf$fare_amount[is.na(cabdf$fare_amount)]=mean(cabdf$fare_amount,na.rm =T)
#Actual=15
#Analysis=15.01

#Median method
cabdf$fare_amount[is.na(cabdf$fare_amount)]=median(cabdf$fare_amount,na.rm =T)
#Actual=15
#Analysis=8.5

#Knn method(DMwR library)
cabdf=knnImputation(cabdf,k=2)
#Actual=15
#Analysis=9.268
```

```
#MISSING VALUE ANALYSIS(Locking the method)

#fare amount(Mean Method)
cabdf$fare_amount[is.na(cabdf$fare_amount)]=mean(cabdf$fare_amount,na.rm =T)

#passenger count(Median Method)
cabdf$passenger_count[is.na(cabdf$passenger_count)]=median(cabdf$passenger_count,na.rm =T)

#year(Median method)
cabdf$year[is.na(cabdf$year)]=median(cabdf$year,na.rm =T)

#month(Median method)
cabdf$month[is.na(cabdf$month)]=median(cabdf$month,na.rm =T)

#year(Median method)
cabdf$wday[is.na(cabdf$wday)]=median(cabdf$wday,na.rm =T)

#CHECKING FOR MISSING VALUE AFTER INSERING

#finding out number of missing value
missing_val1=data.frame(apply(cabdf,2,function(x){sum(is.na(x))}))

#giving names in dataframe
missing_val1$Columns=row.names(missing_val1)
row.names(missing_val1)=NULL
names(missing_val1)[1]="Missing_Percentage"

#converting to percentage
missing_val1$MissingPercentage=(missing_val1$Missing_Percentage/nrow(cabdf))*100

#OUTLIER ANALYSIS
numeric_index=sapply(cabdf,is.numeric)
numeric_data=cabdf[,numeric_index]
cnames=colnames(numeric_data)

#box plot
for(i in 1:length(cnames))
{
   assign(paste0("gn",i),ggplot(aes_string(y=(cnames[i]),x="fare_amount"),
                          data=subset(cabdf))+stat_boxplot(geom="errorbar",width=0.5)+

geom_boxplot(outlier.colour="red",fill="grey",outlier.shape=18,outlier.size=1,notch=FALSE)+theme(legend.position="bottom")+labs(y=cnames[i],

                         x="fare_amount")+ggtitle(paste("Box Plot of fare_amount for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)


cabdf1=cabdf
#deleting variable which has outliers
for(i in cnames)
{
  print(i)
  outliers=cabdf[,i][cabdf[,i] %in% boxplot.stats(cabdf[,i])$out]
  cabdf=cabdf[which(!cabdf[,i] %in% outliers),]
}

#FEATURE SELECTION
install.packages("corrgram",dependencies = TRUE)
library(knitr)
library(fpca)
library(corrgram)

corrgram(cabdf[,numeric_index],order=TRUE,upper.panel=panel.pie,text.panel=panel.txt,main="Correlation Plot")

library(ggplot2)
library(scales)
library(gplots)
library(psych)
cabdf3=cabdf

#BAR GRAPH
ggplot(data=cabdf3,aes(x=cabdf3$year,y=cabdf$fare_amount))+
  geom_col(position=position_dodge())



#SAMPLING(DIVIDING data into training & test data)
cabdf2=cabdf
train_index=sample(1:nrow(cabdf2),0.8*nrow(cabdf2))
train = cabdf2[train_index,]
test = cabdf2[-train_index,]

#LINEAR REGRESSION
library(rpart)
library(MASS)
library(DMwR)

#checking multicollinearity
library(usdm)
vif(cabdf[ ,-1])

#checking correlation with thershold 90%
vifcor(cabdf[,-1], th = 0.9)

#Linear regression model
lm_model = lm(fare_amount ~., data = train)
```

```
#Summary of the model
summary(lm_model)

#Predict
predictions_LR = predict(lm_model, test[,2:9])
rmse_LR=sqrt(mean((test$fare_amount-predictions_LR)^2))
#RMSE=3.67

#DECISION TREE

#rpart for regression
fit = rpart(fare_amount ~ ., data = train, method = "anova")

#Predict for new test cases
predictions_DT = predict(fit, test[,-1])
rmse_DT=sqrt(mean((test$fare_amount-predictions_DT)^2))


#RMSE=3.21


#KNN PREDICTION

library(class)
##predict test data
KNN_predictions=knn(train[,2:9],test[,2:9],train$fare_amount,k=4)
str(KNN_predictions)
KNN_predictions=as.numeric(as.character(KNN_predictions))
rmse_KNN=sqrt(mean((test$fare_amount-KNN_predictions)^2))

#RMSE=4.57


#RANDOM FOREST
library(randomForest)
library(RRF)
library(inTrees)

RF_Model=randomForest(fare_amount~.,train,importance=TRUE,ntree=100)

#extract rules
treelist=RF2List(RF_Model)

exec=extractRules(treelist,train[,-1])
#visualize some rules
exec[1:2,]

#Make rules more readable
readableRules=presentRules(exec,colnames(train))
readableRules[1:4,]

#get rule metrics
ruleMetric=getRuleMetric(exec,train[,-1],train$fare_amount)

ruleMetric[1:2,]


#prediction of test data using RF Model
RF_Prediction=predict(RF_Model,test[,-1])
rmse_RF=sqrt(mean((test$fare_amount-RF_Prediction)^2))

#RMSE=2.33


#PUTTING

cabdf_final=read.csv("test.csv",header=T)
#checking datatype of each variable
str(cabdf_final)

#converting fare amount variable to numeric
#cabdf$fare_amount=as.numeric(as.character(cabdf$fare_amount))

#checking datatype of fare amount
#class(cabdf$fare_amount)

#Converting time stamp into POSIXct
cabdf_final$pickup_datetime=as.POSIXct(strptime(cabdf_final$pickup_datetime,"%Y-%m-%d %H:%M:%S"))

#checking Class
class(cabdf_final$pickup_datetime)

#Extracting years,month & weedays number from time
cabdf_final$year=as.numeric(format(cabdf_final$pickup_datetime,"%Y"))
cabdf_final$month=as.numeric(format(cabdf_final$pickup_datetime,"%m"))

#1-Monday to 7-Sunday
cabdf_final$wday=strftime(cabdf_final$pickup_datetime,"%u")

#converting character to numeric
cabdf_final$wday= as.numeric(as.character(cabdf_final$wday))

cabdf_final$pickup_datetime=NULL

#loading some of the libraries
#x=c("ggplot2","Corrgram","DMwR","Caret","randomForest","unbalanced","C50","dummies","e10","MASS","rpart","gbm","ROSE")
#lapply(x,require,character.only=TRUE)

#finding out number of missing value
missing_value=data.frame(apply(cabdf_final,2,function(x){sum(is.na(x))}))
```

```
#DEPLOYMENT OF MODEL INTO FINAL DATA SET
RF_Prediction1=predict(RF_Model,cabdf_final[,])

#ADDING THE PREDICT_AMOUNT TO THE DATASET
cabdf_final1=read.csv("test.csv",header=T)
cabdf_final1$predicted_fare_amount=with(cabdf_final1,RF_Prediction1)

#WRITING THE FINAL DATA INTI HDD
write.csv(cabdf_final1,"cab_fare_predict.csv",row.names = T)
```