```
rm(list=ls(all=T)) #removes all objects from the current workspace (R memory)

setwd("C:/Users/swat/Desktop/Edwiser/Final_project") #Seting up the working directory.
getwd()


###################################LOAD LIBRARIES
#################################

x=c("ggplot2", "DMwR", "corrgram", "Hmisc", "rpart", "randomForest", "geosphere")

install.packages(x) #install packages
lapply(x, require, character.only = TRUE)
#load a bunch of packages at once to working environment
# library() or require() only load one package at a time --library(xlsx)

rm(x) #removing the x object as packages installed and loaded to working space.


################ LOAD THE GIVEN TRAIN DATA ##############################

train= read.csv("./train_cab/train_cab.csv", header = T)[,-2]

#####checking the column names
colnames(train)
#Getting the number of variables and obervation in the datasets
dim(train)

#Getting first 10 rows of the dataset
head(train, 5)

########Check data shape ########
str(train)

##################convert fare to Numeric type and Passenger to Integer
type###################

train$fare_amount=as.numeric(as.character(train$fare_amount))
train$passenger_count=as.integer(train$passenger_count)

############## Eliminate cells with same pickup and dropoff location
```

```
train=subset(train, !(train$pickup_longitude==train$dropoff_longitude &
train$pickup_latitude==train$dropoff_latitude))

########replace "0's" with NA

train[train==0]= NA

########################Missing Value Analysis ##############################


#########function to calculate missing values ###################

missingvalue= function(data){
  missing_value = data.frame(apply(data, 2 , function(x){sum(is.na(x))}))
  colnames(missing_value)="Missing_Value_count"
  missing_value$percentage=apply(missing_value , 1 , function(x){x/nrow(train)*100})
  missing_value = cbind(row.names(missing_value), missing_value)
  row.names(missing_value)=NULL
  colnames(missing_value)[1]="Variables"
  print(missing_value)


  ############plot Missing Values#######################
  library(ggplot2)
  ggplot(data = missing_value, aes(x=reorder(Variables , -percentage),y = percentage))+
  geom_bar(stat = "identity",fill = "blue")+xlab("Variables")+
  ggtitle("Missing Values") + theme_bw()
}


###################Calculate Missing Values########################

missingvalue(train)



############As PAssenger_count is a categorical Variable , so we will use mode for
Imputation#########

##########calculate mode - create function ############
mode= function(data){
  uniq=unique(data)
  as.numeric(as.character(uniq[which.max(tabulate(match(data,uniq)))]))
```

```
  #print(mode_d)
}

mode(train$passenger_count)


###################################IMPUTATION############################

#impute with the mode
train$passenger_count[is.na(train$passenger_count)] = mode(train$passenger_count)




#################Choose for suitable method for imputation of missing values for other
variables ############

# ####Taking a subset of data
# #train[40,1]= 17.5  #######Data noted to compare ####Actual value
#
# ###Mean method
# train$fare_amount[is.na(train$fare_amount)] = mean(train$fare_amount, na.rm = T)
#
# #Mean= 15.12488
#
# ####Median Method
#
# train$fare_amount[is.na(train$fare_amount)] = median(train$fare_amount, na.rm = T)
#
# #Median= 8.5
#
# ######KNN Method
#
# train = knnImputation(train, k = 5)
# #KNN= 15.90051


########Saving the data in df set ############

df=train
train=train[complete.cases(train[,1]),]
```

```
#As KNN is giving the value closest to Actual Value, We choose KNN for missing value
imputation

library(DMwR)
train=knnImputation(train, k=5)

missingvalue(train)




######################OUTLIER
ANALYSIS#############################################

df=train
############outliers in fare_amount
#Remove negative values from 'fare_amount'
train$fare_amount=ifelse(train$fare_amount<0, NA, train$fare_amount)
train$fare_amount=ifelse(train$fare_amount>30,NA, train$fare_amount)

###################outliers in passenger_count
###################all values greater than 8 are converted to NA
unique(train$passenger_count)

###################Convert more then 8 to NA ##########

for (i in 1:nrow(train)){
  if (as.integer(train$passenger_count[i]) > 8){
    train$passenger_count[i]=NA
  }
}


#######################Outliers in location points ##########

#range of the locations
range(train$pickup_longitude)
range(train$pickup_latitude)
range(train$dropoff_longitude)
range(train$dropoff_latitude)

cnames=colnames(train[,c(2:5)])
```

```
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "fare_amount"), data = train)+
           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="y")+
           ggtitle(paste("Box plot of fare amount",cnames[i])))
}



############## Plotting plots together
gridExtra::grid.arrange(gn1, gn2, ncol=2)
gridExtra::grid.arrange(gn3, gn4, ncol=2)



#Replace all outliers with NA and impute
#create NA on outliers
for(i in cnames){
  val = train[,i][train[,i] %in% boxplot.stats(train[,i])$out]
  print(length(val))
  train[,i][train[,i] %in% val] = NA
}

missingvalue(train)

##########replace missing value with mode
mode(train$passenger_count)
train$passenger_count[is.na(train$passenger_count)] = mode(train$passenger_count)
train=train[complete.cases(train[,1]), ]

#replace all other missing value with mean
train$fare_amount[is.na(train$fare_amount)] = mean(train$fare_amount, na.rm=T)
train$pickup_longitude[is.na(train$pickup_longitude)] = mean(train$pickup_longitude, na.rm=T)
train$pickup_latitude[is.na(train$pickup_latitude)] = mean(train$pickup_latitude, na.rm=T)
train$dropoff_longitude[is.na(train$dropoff_longitude)] = mean(train$dropoff_longitude,
na.rm=T)
train$dropoff_latitude[is.na(train$dropoff_latitude)] = mean(train$dropoff_latitude, na.rm=T)
```

```
missingvalue(train)

#now convert Passenger_count into factor
train$passenger_count=as.factor(train$passenger_count)



#########################FEATURE
SCALING/ENGINEERING#####################
df=train

#create new variable
library(geosphere)
train$dist= distHaversine(cbind(train$pickup_longitude, train$pickup_latitude),
cbind(train$dropoff_longitude,train$dropoff_latitude))
#the output is in metres, Change it to kms
train$dist=as.numeric(train$dist)/1000
df=train
train=df

#############################################CORRELATION AMALYSIS
#####################################

library(corrgram)
corrgram(train[,-6], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

#####correlation between the numeric variables

num_cor=round(cor(train[,-6]), 3)


#Eliminate the pickup and dropoff locations if same (if any)

train=subset(train, !(train$pickup_longitude==train$dropoff_longitude &
train$pickup_latitude==train$dropoff_latitude))

#######remove unnecessary variables

rm(abc,df,gn1,gn2,gn3,gn4,cnames,i,val)
```

```
########################## MODEL DEVELOPMENT
#####################################################3

#create sampling and divide data into train and test

set.seed(123)
train_index = sample(1:nrow(train), 0.8 * nrow(train))

train1 = train[train_index,]#do not add column if already removed
test1 = train[-train_index,]#do not add column if already removed

########### Define Mape - The error matrix to calculate the error and accuracy
################

MAPE = function(y, yhat){
  mean(abs((y - yhat)/y*100))
}

##############################################Decision
Tree#####################################

library(rpart)
fit = rpart(fare_amount ~. , data = train1, method = "anova", minsplit=5)

summary(fit)
predictions_DT = predict(fit, test1[,-1])

MAPE(test1[,1], predictions_DT)

write.csv(predictions_DT, "DT_R_PRed5.csv", row.names = F)

#Error 27.75005
#Accuracy 73.25


########################################Random
Forest################################################

library(randomForest)
RF_model = randomForest(fare_amount ~.  , train1, importance = TRUE, ntree=100)
RF_Predictions = predict(RF_model, test1[,-1])

MAPE(test1[,1], RF_Predictions)
```

```
importance(RF_model, type = 1)

#error 22.50844 for n=100
#accuracy = 77.50



#####################################Linear
Regression#########################################################################

lm_model = lm(fare_amount ~. , data = train1)
summary(lm_model)

predictions_LR = predict(lm_model, test1[,-1])
MAPE(test1[,1], predictions_LR)

#error 26.12016
#Accuracy 73.88

#####################################KNN
Implementation#####################################################################

library(class)

KNN_Predictions = knn(train1[, 2:7], test1[, 2:7], train1$fare_amount, k = 1)

#convert the values into numeric
KNN_Predictions=as.numeric(as.character((KNN_Predictions)))

#Calculate MAPE
MAPE(test1[,1], KNN_Predictions)

#error 33.7978
#Accuracy = 66.21



##############Model Selection and Final Tuning#########################

#Random Forest with using mtry = 2 that is fixing only two variables to split at each tree node

RF_model = randomForest(fare_amount ~.  , train1, importance = TRUE, ntree=200, mtry=2)
RF_Predictions = predict(RF_model, test1[,-1])
```

```
MAPE(test1[,1], RF_Predictions)
importance(RF_model, type = 1)

#error 22.38 for n=100
#Accuracy 77.7

rm(a, num_cor,pre, i)


####################################Predict VAlues in Test Data##################

pred_data=read.csv("./test/test.csv", header= T)[,-1]

#########create distance variable
pred_data=subset(pred_data, !(pred_data$pickup_longitude==pred_data$dropoff_longitude &
pred_data$pickup_latitude==pred_data$dropoff_latitude))
pred_data[pred_data==0]= NA

# COnnvert Data into proper data types

str(pred_data)
pred_data$passenger_count=as.factor(pred_data$passenger_count)

#calculate distance

pred_data$dist= distHaversine(cbind(pred_data$pickup_longitude, pred_data$pickup_latitude),
cbind(pred_data$dropoff_longitude,pred_data$dropoff_latitude))

#the output is in metres, Change it to kms

pred_data$dist=as.numeric(pred_data$dist)/1000

# Create the target variable
pred_data$fare_amount=0
pred_data=pred_data[,c(1,2,3,4,5,6,7)]

#Random Forest
RF_model = randomForest(fare_amount ~.  , train, importance = TRUE, ntree=200, mtry=2)

write.csv(pred_data, "Predicted_Data.csv", row.names = F)
```