## IST687

## Student name: Kartheek Sunkara

## Project

## Date submitted: 21 Feb 2019 (3:03AM)

## Date due: 21 Feb 2019 (9:00AM)

##

## Attribution statement: (choose the one statement that is true)

## 1. I did this homework by myself and some websites: tutorialspoint, r bloggers, rdocumentation, sthda, stackoverflow and rfunction

rm(list=ls()) # is used to remove all the objects from the workspace when you use list=ls() as base

dev.off() #shuts down all open graphics devices

cat('\014') #clears the console space

##Setting working directory

##setwd("C:\\Users\\KARTHEEK SP\\Desktop\\IST 687")

#install.packages("psych")

library(psych)

#install.packages("ggplot2")

library(ggplot2)

#install.packages("lm")

#library(lm)

##importing the dataset Spring19Survey.csv

df <- read.csv('C:/Users/KARTHEEK SP/Desktop/IST 687/spring19survey.csv')

##Viewing the dataframe to have an overview

#View(df)

##Summary of the columns

#summary(df)

##Know the column names

colnames(df)

# [1] "Satisfaction" "Airline.Status" "Age"

# [4] "Gender" "Price.Sensitivity" "Year.of.First.Flight"

# [7] "Flights.Per.Year" "Loyalty" "Type.of.Travel"

# [10] "Total.Freq.Flyer.Accts" "Shopping.Amount.at.Airport" "Eating.and.Drinking.at.Airport"

# [13] "Class" "Day.of.Month" "Flight.date"

# [16] "Partner.Code" "Partner.Name" "Orgin.City"

# [19] "Origin.State" "Destination.City" "Destination.State"

# [22] "Scheduled.Departure.Hour" "Departure.Delay.in.Minutes" "Arrival.Delay.in.Minutes"

# [25] "Flight.cancelled" "Flight.time.in.minutes" "Flight.Distance"

# [28] "Arrival.Delay.greater.5.Mins" "Long.Duration.Trip"

# [A] Data Cleaning (Finding rows and columns with all NA's and cleaning them)

##Since we can perform analysis having NA's in the dataframe it is not sugegstible to delete the columns or rows that have NA's

##it would lead to the data loss

##But we can delete the rows/coolumns with all NA values

## (a.1) Checking for the columns that have all NA values

onlyNAcolumns\_idx <- colSums( !is.na(df) ) == 0 # avoid apply loop

## (a.2) Checking for the rows that have all NA values

onlyNArows\_idx <- rowSums( !is.na(df) ) == 0 # avoid apply loop

#Counting the masked values of df[onlyNArows\_idx] so that the rows with all NA values are filtered out

nrow(df[onlyNArows\_idx])==nrow(df)

#To find the mean of Satisfaction and replace the NA values with it

summary(df['Satisfaction'])

df$Age <- as.numeric(df$Age)

df$Flight.time.in.minutes <- as.numeric(df$Flight.time.in.minutes)

df$Day.of.Month <- as.numeric(df$Day.of.Month)

df$Flight.Distance <- as.numeric(df$Flight.Distance)

df[5:8] <- lapply(df[5:8],as.numeric)

df[10:12] <- lapply(df[10:12],as.numeric)

df[22:24] <- lapply(df[10:12],as.numeric)

#checking for any rows which are not complete

sum(!complete.cases(df)) #4113

ncol(df) #29

nrow(df) #194833

#converting all integer columns to numeric

df$Age <- as.numeric(df$Age)

df$Flight.time.in.minutes <- as.numeric(df$Flight.time.in.minutes)

df$Day.of.Month <- as.numeric(df$Day.of.Month)

df$Flight.Distance <- as.numeric(df$Flight.Distance)

df[5:8] <- lapply(df[5:8],as.numeric)

df[10:12] <- lapply(df[10:12],as.numeric)

df[22:24] <- lapply(df[10:12],as.numeric)

#View(df)

#replacing na values with mean

for(i in 1:ncol(df)){

df[is.na(df[,i]), i] <- mean(df[,i], na.rm = TRUE)

}

#Validating that the data is cleaned

# complete.cases: Return a logical vector indicating which cases are complete, i.e., have no missing values.

sum(!complete.cases(df)) # reduced from 510 to zero

is.null(df)

#View(Airdata)

nrow(df)

nrow(df)

str(df)#replacing NA's in satisfaction with mean

#df[["Satisfaction"]][is.na(df[["Satisfaction"]])] <- 3.382

# eco\_mean <- 0

# eco\_no <- 0

# plus\_mean <- 0

# plus\_no <- 0

# bus\_mean <- 0

# bus\_no <- 0

#

# for(i in 1:nrow(df)){

# if (df[i,13] == 'Eco') {

# eco\_mean <- eco\_mean+df[i,1]

# eco\_no <- eco\_no+1}

# else if (df[i,13] == 'Eco Plus'){

# plus\_mean <- plus\_mean+df[i,1]

# plus\_no <- plus\_no+1}

# else if (df[i,13] == 'Business'){

# bus\_mean <- bus\_mean+df[i,1]

# bus\_no <- bus\_no+1}}

#

# eco\_mean <- eco\_mean/eco\_no

# plus\_mean <- plus\_mean/plus\_no

# bus\_mean <- bus\_mean/bus\_no

#

# for (i in 1:nrow(df)){

# if (df[i,13] == 'Eco') {

# df[i,1] <- eco\_mean}

# else if (df[i,13] == 'Eco Plus'){

# df[i,1] <- plus\_mean}

# else if (df[i,13] == 'Business'){

# df[i,1] <- bus\_mean}}

df <- df[df$Class == 'Eco',]

a1 <- mean(df$Satisfaction)

#3.373627 so 1.626373 less than 5 (fully satisfied)

hist(airline\_status$Satisfaction)

#

#Columns that are not included in the linear modelling analysis

# 14 Day.of.Month (The graph is normally distributed)

#19 Origin.State (coz we have origin city and destination)

#21 Destination.State

#28 Arrival.Delay.greater.5.Mins

#Long.Duration.Trip

linear\_model\_eco <- lm(formula = Satisfaction ~ Age + Price.Sensitivity +

Flights.Per.Year + Loyalty + Total.Freq.Flyer.Accts +

Shopping.Amount.at.Airport + Eating.and.Drinking.at.Airport

+ Scheduled.Departure.Hour + Departure.Delay.in.Minutes +

Flight.time.in.minutes+ Arrival.Delay.in.Minutes+ Airline.Status+Price.Sensitivity+ Type.of.Travel+Flight.date+Orgin.City + Destination.City+ Long.Duration.Trip +Gender,

data = df)

summary(linear\_model\_eco) #Adjusted R-squared: 0.4168

####Significant variables are::: Age, Price.Sensitivity, Flights.Per.Year,

#Shopping.Amount.at.Airport,Departure.Delay.in.Minutes,

#Flight.time.in.minutes, Arrival.Delay.in.Minutes,

#Airline.Status (Gold), Airline.Status (Platinum)

#Airline.Status (Silver),Type.of.Travel (Mileage tickets),

#Type.of.Travel (Personal Travel)

#Flight.date03/11/2014, Flight.date2/22/2014,

#Orgin.CityAllentown/Bethlehem/Easton (PA), Orgin.CityAlpena, MI

#p-value: < 2.2e-16

df\_1 <- df[,c(1,2,3,5,7,11,23,24,26)]

######################SVM Starts########################################

#install.packages("kernlab") #The package used here to create a model

library(kernlab)

data(df\_1) #load the data

dim(df\_1) #158606x8

table(df\_1$Satisfaction) #table function is a numeric variable output

#Randomizing the dataset befor dividing into train and test datasets

randIndex <- sample(1:dim(df\_1)[1]) #So all the 10000 entries are getting randomized

summary(randIndex)

#Min. 1st Qu. Median Mean 3rd Qu. Max.

#1 39675 79349 79349 119022 158696

length(randIndex) #is 158696 (it is as expected)

head(randIndex) #will give u first 6 values of randIndex

#the indices get changed everytime u sample

cutPoint2\_3 <- floor(2 \* dim(df\_1)[1]/3) # calculate the cut point that would divide the spam data set into a two-thirds training set and a one-third test set

cutPoint2\_3 #floor() function in the above command chops off any decimal part of the calculation

trainData <- df\_1[randIndex[1:cutPoint2\_3],] # build our training set from the first 105797 rows

testData <- df\_1[randIndex[(cutPoint2\_3+1):dim(df\_1)[1]],] #creating the test set similar to the above

#Building a Model using ksvm

#5. Build a support vector model using the ksvm( ) function using two or three of the variables to predict a happy customer.

#Once you have specified the model statement and the name of the training data set, you can use the same parameters as shown on page 237:

#kernel= "rbfdot", kpar = "automatic", C = 5, cross = 3, prob.model = TRUE

svmOutput <- ksvm(Satisfaction~Age+Price.Sensitivity+ Flights.Per.Year+

Shopping.Amount.at.Airport+Departure.Delay.in.Minutes+

Flight.time.in.minutes+Arrival.Delay.in.Minutes+

Airline.Status,

data=trainData, kernel=

'rbfdot',kpar='automatic', C=50,cross=5,

prob.model = TRUE)

#Storing the output of kvsm( ) in a variable and then echo that variable to the console.

svmOutput #output structure

hist(alpha(svmOutput)[[1]])

svmresult <- predict(svmOutput,testdata,type="votes")

View(svmresult)

str(svmresult) #Observation -- num [1:17659, 1]

head(svmresult)

SatPred <- svmresult[,1]

View(SatPred)

happyPred[SatPred>3] <- 1

happyPred[happyPred<] <- 0

View(testdata)

str(testdata)

ctable <- data.frame(testdata$Satisfaction, happyPred)

table(ctable)

#11. Calculate an error rate based on what you see in the confusion matrix. See pages 243-244 for more information.

errorrate <- ((table(ctable)[1,2] + table(ctable)[2,1])/(table(ctable)[1,1]+ table(ctable)[1,2] + table(ctable)[2,1] +table(ctable)[2,2]))\*100

errorrate #We find that error rate is around 11.72% which is good since it means more than only 10% of our predictions are wrong