#CLEAR ENVIRONMENT AND INSTALL INITIAL PACKAGES

rm(list = ls(all = TRUE))#Clear Enviroment

#specify the packages of interest

packages=c("maps", "zipcode","mapproj","ggmap","ggplot2","gdata", "sqldf", "kernlab","e1071","gridExtra","ggplot2", "caret", "arules", "reshape2")

#use this function to check if each package is on the local machine if a package is installed, it will be loaded if any are not, the missing package(s) will be installed and loaded

package.check <- lapply(packages, FUN = function(x) {

if (!require(x, character.only = TRUE)) {

install.packages(x, dependencies = TRUE)

library(x, character.only = TRUE)

}

})

####BEGIN

#Read File Into R. DF = All Data w/ preliminary column removal

df <- read.csv(file= "C:/Users/jdine/Desktop/out-201501.csv")[,c(9,12,17,18,23,26,54,59,65,73,76,89:92,106:111,137:147,167:171,175,176,179,182,199:227,232), ]

df <- data.frame(df, stringsAsFactors = FALSE)

#Entire Southwest

df1 <- sqldf("select \* from df WHERE State\_PL like '%California%' or State\_PL like '%Arizona%' or State\_PL like '%Nevada%' or State\_PL like '%Utah%' or State\_PL like '%New Mexico%' or State\_PL like '%Colorado%' ") #Filter for California only. DF1 = All data where state = Soutwest states w/ preliminary column removal

df1$CheckInMonth\_Year <- as.Date(paste(format(as.Date(df1$CHECK\_IN\_DATE\_C), "%Y-%m"), "-01", sep=""),format="%Y-%m-%d") #add M/Y Column based on check in date

df1 <- df1[-c(3,4,6,11:14)] #Remove check in/check out date variables + other dupes

#Plot States Focused On from Sample

us<-map\_data("state")

US<- us[us$region=='california' | us$region=='arizona' | us$region=='nevada' | us$region=='utah' | us$region=='new mexico' | us$region=='colorado',]

rownames(US) <- NULL

US

cnames<-aggregate(cbind(US$long,US$lat)~US$region,data=US,FUN=function(x)mean(range(x)))

cnames$region<-cnames$`US$region`

cnames

US<-sqldf('select \* from US inner join cnames on US.region=cnames.region')

US

region<- ggplot(US, aes(map\_id=region))

region<-region + geom\_map(map=us,fill="orange",color="black")

region<-region + expand\_limits(x=US$long,y=US$lat) + coord\_map()

region<-region + geom\_label(aes(x=US$V1,y=US$V2),label=toupper(US$region),size=4)

region

#Rename Columns

colnames(df1) <- c("GuestID", "RoomDescription", "POVCode", "NightlyRate", "LengthOfStay", "GuestCountry", "Revenue\_USD", "GuestRoomFloor", "GuestState", "GuestCountry", "GuestGender", "GuestAgeRange", "POV\_H", "LanguageH", "LikelihoodToReco", "OverallSatisfaction", "GuestRoomH", "Tranquility", "HotelCondition", "CustomerService", "StaffCared", "InternetSatisfaction", "CheckInH", "FBFrequency", "FBExperience", "HotelCity", "HotelState", "HotelRegion", "HotelZipCode", "HotelCountry","HotelLatitude", "HotelLongitude", "NPSGoal", "Brand", "AllSuitesFlag", "BellStaffFlag", "BoutiqueFlag", "BusinessCenterFlag", "CasinoFlag", "ConferenceFlag", "ConventionFlag","DryCleaningFlag", "ElevatorFlag", "FitnessCenterFlag", "FitnessTrainerFlag","GolfFlag", "IndoorCorridorsFlag", "LaundryFlag", "LimoServiceFlag", "MiniBarFlag","IndoorPoolFlag", "OutdoorPoolFlag", "RegencyGrandClubFlag", "ResortFlag", "RestaurantFlag", "SelfParkingFlag", "ShuttleServiceFlag", "SkiFlag", "SpaFlag", "SpaServicesInFitnessCenterFlag", "SpaOnlineBookingFlag", "SpaFBOfferingFlag", "ValetFlag", "NPSType", "CheckInMonth\_Year")

dftest <- df1[!is.na(df1$LikelihoodToReco),] #Filter based off likelihood to reco column

#NPSScore column + classification (need as.int)

dftest$NPSScore <- as.character(dftest$NPSType)

dftest$NPSScore[dftest$NPSScore == "Promoter"] <- "1"

dftest$NPSScore[dftest$NPSScore == "Detractor"] <- "-1"

dftest$NPSScore[dftest$NPSScore == "Passive"] <- "0"

#Create column that sums all survey response numbers divided by total response possible, run the regression on that value compared to flag columns

#Convert surveyscores to as.numeric

for(i in 15:25)

{

dftest[,i] <- as.numeric(dftest[,i])

}

#Column that adds all surveyscores together

dftest$totalsurveyscore <- apply(dftest[,c(16:25)], 1, sum, na.rm= TRUE)

#######Step To create updated column with total survey score possible, and column with ratio of total survey score/total survey score possible

#Create new df

dftestwscores <- dftest

#Converting Y/N Flags on amenities to binary

for(i in 35:63)

{

dftestwscores[,i] <- as.character(dftestwscores[,i])

dftestwscores[,i][dftestwscores[,i] == "Y"] <- "1"

dftestwscores[,i][dftestwscores[,i] == "N"] <- "0"

dftestwscores[,i][dftestwscores[,i] == NULL] <- "Null"

dftestwscores[,i] <- as.factor(dftestwscores[,i])

}

#override original survey columns with 0/1 to calculate total survey response possible

for(i in 16:25)

{

dftestwscores[,i][dftestwscores[,i] > 0] <- 1

record <- dftestwscores[[i]]

record[sapply(record, is.na)] <- 0

dftestwscores[[i]] <- record

}

dftestwscores$totalpossible <- rowSums(dftestwscores[,c(16:25)]) \*10 #perform row addition on survey responses not equal to 0

dftestwscores$ratio <- (dftestwscores$totalsurveyscore/dftestwscores$totalpossible)\*100 #create ratio column

dftestwscores[,c(15:25)] <- dftest[,c(15:25)] #Return survey columns to original values

#Fixing the Classifications on all variables that are messed up

dftestwscores$LikelihoodToReco <- as.numeric(dftestwscores$LikelihoodToReco)

dftestwscores$Revenue\_USD <- as.numeric(dftestwscores$Revenue\_USD)

dftestwscores$LengthOfStay <- as.numeric(dftestwscores$LengthOfStay)

dftestwscores$HotelZipCode <- as.factor(dftestwscores$HotelZipCode)

dftestwscores$NPSScore <- as.factor(dftestwscores$NPSScore)

#kill additional useless columns not needed for analysis

dftestwscores <- dftestwscores[-c(1,3,4,5,8)]

str(dftestwscores)

#Linear Modeling

hotelmodel1 <- lm(formula= LikelihoodToReco~OverallSatisfaction+GuestRoomH+Tranquility+HotelCondition+StaffCared+ CustomerService+InternetSatisfaction+CheckInH+FBFrequency+FBExperience, data= dftestwscores)

summary(hotelmodel1) #OverallSat, Guestroom,HotelCondition, CustomerService were the only statistically relevant variables.

hotelmodel2 <- lm(formula= LikelihoodToReco~OverallSatisfaction+GuestRoomH+HotelCondition+StaffCared +CustomerService, data= dftestwscores)

summary(hotelmodel2)

str(hotelmodel2)

newdata1 <- data.frame(OverallSatisfaction= 8, GuestRoomH=7, HotelCondition=10, StaffCared=8, CustomerService=9 )

predict(hotelmodel2, newdata1, type="response")

library("ggplot2")

#Calculate NPS Score

dfNPSScoreCalc<-data.frame(table(dftest$NPSType))

dfNPSScoreCalc<-dfNPSScoreCalc[-1,]

dfNPSScoreCalc

str(dfNPSScoreCalc)

dfNPSScoreCalc$TotalSurveys<-sum(dfNPSScoreCalc$Freq)

dfNPSScoreCalc$PercentNPS<-dfNPSScoreCalc$Freq / dfNPSScoreCalc$TotalSurveys

dfNPSScoreCalc$PercentNPSScore<- dfNPSScoreCalc$PercentNPS \* 100

rownames(dfNPSScoreCalc)<- NULL

dfNPSScoreCalc

ggplot(dfNPSScoreCalc,aes(x=dfNPSScoreCalc$Var1, y=dfNPSScoreCalc$PercentNPSScore)) + geom\_col(color="black",fill="orange") + ggtitle("Percent of NPS Types") + xlab("NPS Types") + ylab("Percent") #Column Graph Based on Percentage by NPSType

dfNPSScoreCalc<-dfNPSScoreCalc[c(1,3),5]

dfNPSScoreCalc

NPSScore<-data.frame(dfNPSScoreCalc)

NPSScore<-apply(NPSScore,2,function(x)x-x[1])

NPSScore<-NPSScore[2,]

NPSScore #Run this to find total NPS Score

#Basis Descriptive Statistics

summary(dftestwscores)

str(dftestwscores)

sqldf("select GuestGender, count(GuestGender) from dftestwscores where HotelState = 'California' group by GuestGender")

sqldf("select AVG(LikelihoodToReco), GuestGender from dftestwscores group by GuestGender ")

sqldf("select AVG(LikelihoodToReco), GuestAgeRange from dftestwscores group by GuestAgeRange ")

sqldf("select AVG(LikelihoodToReco), HotelState from dftestwscores group by HotelState ")

sqldf("select AVG(LikelihoodToReco), Count(LikelihoodToReco), Brand from dftestwscores group by Brand ")

sqldf("select AVG(Revenue\_USD), GuestGender from dftestwscores group by GuestGender ")

sqldf("select Sum(Revenue\_USD), GuestGender from dftestwscores group by GuestGender ")

sqldf("select AVG(Revenue\_USD), GuestCountry from dftestwscores group by GuestCountry ")

sqldf("select Sum(Revenue\_USD), AVG(LikelihoodToReco) from dftestwscores group by LikelihoodToReco ")

#

#Build Column Chart with Counts by Brand

brandtable <- table(dftestwscores$Brand)

branddf <- data.frame(brandtable)

str(branddf)

branddf$Freq <- as.numeric(branddf$Freq)

ggplot(branddf,aes(x=branddf$Var1, y=branddf$Freq)) + geom\_col(color="black",fill="orange") + ggtitle("Count of Guests per Brand") + xlab("Brand") + ylab("Count") #Column Graph Based on Percentage by NPSType

genderage <- data.frame(sqldf("select AVG(LikelihoodToReco), GuestAgeRange from dftestwscores group by GuestAgeRange "))

genderage <- genderage[-1,]

ggplot(genderage,aes(x=genderage$GuestAgeRange, y=genderage$AVG.LikelihoodToReco.)) + geom\_point(color="black",fill="orange") + ggtitle("Average LTR Score by AgeRange") + xlab("AgeRange") + ylab("Score") #Column Graph Based on Percentage by NPSType

#Build Column Chart with percentage by Gender

agedemo <- data.frame(sqldf("select Count(GuestGender), GuestGender from dftestwscores group by GuestGender "))

agedemo <- agedemo [-1,]

str(agedemo)

agedemo$percent <- agedemo$Count.GuestGender./5713

ggplot(agedemo,aes(x=agedemo$GuestGender, y=agedemo$percent)) + geom\_col(color="black",fill="orange") + ggtitle("Gender Split") + xlab("Gender") + ylab("Percent of Whole") #Column Graph Based on Percentage by NPSType

#Association Rule Mining

#Attempt Using Amenities Columns

dfARM <- dftestwscores

dfARM$NPSType <- as.character(dfARM$NPSType)

str(dfARM)

dfARM$NPSType[dfARM$NPSType == "Passive"] <- NA

dfARM <- dfARM[!is.na(dfARM$NPSType),]

dfARM <- dfARM[,c(30:59)]

dfARM <- na.omit(dfARM)

str(dfARM)

dfARM$NPSType <- as.factor(dfARM$NPSType)

ARM <- apriori(dfARM, parameter = list(support=.6,confidence=.6))

inspect(ARM)

ARMdf <- data.frame(inspect(ARM))

ARMdf1 <- ARMdf[ARMdf$rhs == '{NPSType=Promoter}',] #Filtering just for good score rhs

ARMdf1

plot(ARMdf1, method="graph", control=list(type="items"))

plot(ARMdf1, method="paracoord", control=list(reorder=TRUE))

#Plotting Frequency of Amenities

dfplot <- dftestwscores[,c(6:8)]

dfplot.trans <- as(dfplot, "transactions")

itemFrequencyPlot(dfplot.trans, cex.names=.5)

itemFrequency(dfplot.trans)

#Correlation Matrix

dfcorr <- dftestwscores[,c(10,30:58)]

for(i in 1:30)

{

dfcorr[,i] <- as.numeric(dfcorr[,i])

}

dfcorr$AllSuitesFlag <- as.numeric(dfcorr$AllSuitesFlag)

str(dfcorr)

cor(dfcorr)

dfcorr1 <- data.frame(cor(dfcorr))

dfcorr1 <- round(dfcorr1, 2)

install.packages("corrplot")

library("corrplot")

corrplot(cor(dfcorr), method= "square", title="Amenities Correlation Test")

##FOR SUPPORT VECTORs

# IF we wanted to remove all of the passives from the data frame to just have promoter and detractors

dfwscoresfinal <- dftestwscores

dfwscoresfinal$NPSType <- as.character(dfwscoresfinal$NPSType)

str(dfwscoresfinal)

dfwscoresfinal$NPSType[dfwscoresfinal$NPSType == "Passive"] <- NA

dfwscoresfinal <- dfwscoresfinal[!is.na(dfwscoresfinal$NPSType),]

#Will want to remove all excess col that aren't need here

dfSVMS <- dfwscoresfinal[c(10:20,33,37,38,39,43,51,59)]

##################################################Support Vector Machines

dfSVMS <- na.omit(dfSVMS)

nrows <- nrow(dfSVMS)

random.index <- sample(1:nrows)

head(random.index)

cutPoint <- floor(nrows/3\*2)

#Training Data (2/3 of total data sampled)

hotel.trainingdata <- dfSVMS[random.index[1:cutPoint],]

dim(hotel.trainingdata)

str(hotel.trainingdata)

#Testing Data (1/3 of total data sampled)

hotel.testingdata <- dfSVMS[random.index[(cutPoint+1):nrows],]

dim(hotel.testingdata)

str(hotel.testingdata)

#root mean squared error function

rmse <- function(error)

{

sqrt(mean(error^2))

}

require(kernlab)

require(e1071)

require(ggplot2)

##KSVM MODEL

model.ksvm.train <- ksvm(LikelihoodToReco ~., data=hotel.trainingdata, kernel = "rbfdot", kpar = "automatic", C = 25, cross = 3, prob.model = TRUE) #building the model

model.ksvm.train

model.ksvm.predict <- predict(model.ksvm.train, hotel.testingdata) #testing the model on the testing data

hotel.testingdata$error <- hotel.testingdata$LikelihoodToReco - model.ksvm.predict #computing the error between the predicted vs actual

head(hotel.testingdata)

rmse(hotel.testingdata$error) #Computing RMSE. RMSE = .87

summary(model.ksvm.train)

##SVM MODEL

Model.svm.train <- svm(LikelihoodToReco ~., data=hotel.trainingdata) #building the model

Model.svm.train

model.svm.predict <- predict(Model.svm.train, hotel.testingdata)

hotel.testingdata$error <- hotel.testingdata$LikelihoodToReco - model.svm.predict #computing the error between the predicted vs actual

head(hotel.testingdata)

rmse(hotel.testingdata$error) #Computing RMSE. RMSE = .72

############################### Step 4 : Create a Variable

hotel.trainingdata$goodScore <- ifelse(hotel.trainingdata$NPSType == 'Detractor', 0, 1)

hotel.testingdata$goodScore <- ifelse(hotel.testingdata$NPSType == 'Detractor', 0, 1)

hotel.trainingdata$goodScore <- as.factor(hotel.trainingdata$goodScore)

hotel.testingdata$goodScore <- as.factor(hotel.testingdata$goodScore)

# remove "likelihood" from train data

hotel.trainingdata <- hotel.trainingdata[,-1]

# remove "likelihood" from test data

hotel.testingdata <- hotel.testingdata[,-1]

#Predicting Promoters V Detractors

#KSVM

model.ksvm.train <-ksvm(goodScore~., data=hotel.trainingdata, kernel = "rbfdot", kpar = "automatic", C = 50, cross = 3, prob.model = TRUE)

hotel.testingdata$predictedgoodScore <- predict(model.ksvm.train, hotel.testingdata, type = "response")

head(hotel.testingdata)

str(hotel.testingdata)

results <- table(hotel.testingdata$predictedgoodScore, hotel.testingdata$goodScore)

print(results)

percentCorrect <- (results[1,1]+results[2,2])/(results[1,1]+results[1,2]+results[2,1]+results[2,2])\*100

print(round(percentCorrect) )

#Plot KSVM Model

compgood1 <- data.frame(hotel.testingdata$goodScore, hotel.testingdata$predictedgoodScore)

colnames(compgood1) <- c("test", "pred")

compgood1$correct <- ifelse(compgood1$test==compgood1$pred,"correct","wrong")

Plot\_ksvm <- data.frame(compgood1$correct,hotel.testingdata$OverallSatisfaction,hotel.testingdata$GuestRoomH,hotel.testingdata$goodScore,compgood1$pred)

colnames(Plot\_ksvm) <- c("correct","OverallSatisfaction","GuestRoomH","goodScore","Predict")

ksvmgoodbadplot <- ggplot(Plot\_ksvm, aes(x=OverallSatisfaction,y=GuestRoomH)) +

geom\_point(aes(size=correct,color=goodScore,shape = Predict))+

ggtitle("ksvm - good/bad score")

ksvmgoodbadplot

#SVM

model.svm.train <-svm(goodScore~., data=hotel.trainingdata)

hotel.testingdata$predictedgoodScore <- predict(model.svm.train, hotel.testingdata)

head(hotel.testingdata)

str(hotel.testingdata)

results <- table(hotel.testingdata$predictedgoodScore, hotel.testingdata$goodScore)

print(results)

percentCorrect <- (results[1,1]+results[2,2])/(results[1,1]+results[1,2]+results[2,1]+results[2,2])\*100

print(round(percentCorrect) )

#Plot SVM Model

compgood2 <- data.frame(hotel.testingdata$goodScore, hotel.testingdata$predictedgoodScore)

colnames(compgood2) <- c("test", "pred")

compgood2$correct <- ifelse(compgood2$test==compgood2$pred,"correct","wrong")

Plot\_svm <- data.frame(compgood2$correct,hotel.testingdata$OverallSatisfaction,hotel.testingdata$GuestRoomH,hotel.testingdata$goodScore,compgood2$pred)

colnames(Plot\_svm) <- c("correct","OverallSatisfaction","GuestRoomH","goodScore","Predict")

svmgoodbadplot <- ggplot(Plot\_svm, aes(x=OverallSatisfaction,y=GuestRoomH)) +

geom\_point(aes(size=correct,color=goodScore,shape = Predict))+

ggtitle("svm - good/bad score")

svmgoodbadplot

#NAIVE BAYES

model.naivebayes.train <-naiveBayes(goodScore~., data=hotel.trainingdata)

hotel.testingdata$predictedgoodScore <- predict(model.naivebayes.train, hotel.testingdata)

head(hotel.testingdata)

str(hotel.testingdata)

results <- table(hotel.testingdata$predictedgoodScore, hotel.testingdata$goodScore)

print(results)

percentCorrect <- (results[1,1]+results[2,2])/(results[1,1]+results[1,2]+results[2,1]+results[2,2])\*100

print(round(percentCorrect) )

#Plot NB Model

compgood3 <- data.frame(hotel.testingdata$goodScore, hotel.testingdata$predictedgoodScore)

colnames(compgood3) <- c("test", "pred")

compgood3$correct <- ifelse(compgood3$test==compgood3$pred,"correct","wrong")

Plot\_NB <- data.frame(compgood3$correct,hotel.testingdata$OverallSatisfaction,hotel.testingdata$GuestRoomH,hotel.testingdata$goodScore,compgood3$pred)

colnames(Plot\_NB) <- c("correct","OverallSatisfaction","GuestRoomH","goodScore","Predict")

NBgoodbadplot <- ggplot(Plot\_NB, aes(x=OverallSatisfaction,y=GuestRoomH)) +

geom\_point(aes(size=correct,color=goodScore,shape = Predict))+

ggtitle("Naive Bayes - good/bad score")

NBgoodbadplot

grid.arrange(ksvmgoodbadplot,svmgoodbadplot,NBgoodbadplot, nrow=2)

tapply(dftestwscores$LikelihoodToReco, dftestwscores$HotelState, mean)

#Plotting revenue by zip code

meanlikelihood<- data.frame(dftestwscores$HotelZipCode, dftestwscores$Revenue\_USD)

meanlikelihood <- na.omit(meanlikelihood)

colnames(meanlikelihood) <- c("zip", "Revenue")

meanlikelihood$dftestwscores.HotelZipCode <- clean.zipcodes(meanlikelihood$zip)

str(meanlikelihood)

zipcodes <- data(zipcode) #saved as zipcode

merged <- merge(meanlikelihood, zipcode, by="zip")

str(merged)

score <- tapply(merged$Revenue, merged$state, sum) # calc mean of median by state

head(score)

head(merged)

merged$stateName <- state.name[match(merged$state,state.abb)]

merged$stateName <- tolower(merged$stateName)

head(merged)

us <- map\_data("state") # performed above, not adding anything new

minx <- min(merged$longitude)

maxx <- max(merged$longitude)

miny <- min(merged$latitude)

maxy <- max(merged$latitude)

mapZip <- ggplot(merged, aes(map\_id = stateName))

mapZip <- mapZip + geom\_map(map=us, fill="black", color="white")

mapZip <- mapZip + expand\_limits(x =maxx, y = maxy )

mapZip <- mapZip + geom\_point(data = merged,aes(x = merged$longitude, y = merged$latitude, color=merged$Revenue))

mapZip <- mapZip + coord\_map() + ggtitle("Revenue per zip code")

mapD <- mapZip + geom\_density\_2d(data = merged, aes(x = merged$longitude, y = merged$latitude))

mapD

theme\_update(plot.title = element\_text(hjust = 0.5))

minx <- min(merged$longitude)

maxx <- max(merged$longitude)

miny <- min(merged$latitude)

maxy <- max(merged$latitude)

mapD <- mapD + xlim(minx, maxy)

mapZipZoomed <- mapZip + geom\_point(aes(x = merged$longitude, y = merged$latitude), color="darkred", size = 3)

mapZipZoomed <- mapD + xlim(merged$longitude0) + ylim(merged$latitude-100) + coord\_map()