A. Column Chart by Brand

***#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*

B. Focused States

***#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*

C. Revenue by ZipCode

*#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))*

D. Descriptive Statistics

***#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 ")*

tapply(merged$Revenue, merged$state, sum)

E. Column Chart by Gender Percentage

***#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")*

F. Histogram by Guest Age Range

histogram(dftestwscores$GuestAgeRange)

G. Line Chart Likelihood to Recommend Score by Age Range

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("LTR Score by AgeRange") + xlab("AgeRange") + ylab("Score")

H. NPS Score Calculation

***#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")*

I. Column Chart by Percent NPS Type

***#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*

J. Numberline with NPS score Plotted

#Numberline showing how well the NPS score rates

xlim <-c(-100,100)

ylim<-c(0,100)

px<-c(0,50,round(NPSScore),70)

txtpx<-c('Good','Excellent',paste(round(NPSScore), ' - Dataset NPS Score'),'World Class')

py<-c(0,0,0,0)

lx.buf<-25

lx<-seq(xlim[1]+lx.buf,xlim[2]-lx.buf,len=length(px))

lx

ly<-20

par(xaxs='i',yaxs='i',mar=c(5,1,1,1))

plot(NA,xlim=xlim,ylim=ylim,axes=F,ann=F)

axis(1)

segments(px,py,lx,ly)

points(px,py,pch=16,xpd=NA)

text(lx,ly,txtpx,pos=3)

text(0,50,'NPS Score Levels') ## Levels based on website. "https://www.promoter.io/blog/good-net-promoter-score/"

K. NPS Score by State

##NPS Score by State

dfNPSState<-data.frame(table(dftest$NPSType, dftest$HotelState))

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

dfNPSState

rownames(dfNPSState) <- NULL

totalSurveyed<-tapply(dfNPSState$Freq,dfNPSState$Var2, sum)

Var2<-rownames(totalSurveyed)

dfTotals<-data.frame(Var2,totalSurveyed)

rownames(dfTotals) <- NULL

dfTotals

dfNPSState<-merge(x=dfNPSState,y=dfTotals,by="Var2",all=TRUE)

dfNPSState

dfNPSState<-dfNPSState[dfNPSState$Freq!=0,]

rownames(dfNPSState)<-NULL

dfNPSState$PercentNPS<-dfNPSState$Freq / dfNPSState$totalSurveyed

dfNPSState$PercentNPSScore<- dfNPSState$PercentNPS \* 100

dfNPSState<-dfNPSState[dfNPSState$Var1!='Passive',]

dfNPSState$PercentNPSScore<- ifelse(dfNPSState$Var1=='Detractor',dfNPSState$PercentNPSScore\*-1,dfNPSState$PercentNPSScore\*1)

dfNPSState

NPSByState<-tapply(dfNPSState$PercentNPSScore,dfNPSState$Var2,sum)

NPSByState

dfNPSByState<-data.frame(rownames(NPSByState),NPSByState)

rownames(dfNPSByState)<-NULL

dfNPSByState<-na.omit(dfNPSByState)

rownames(dfNPSByState)<-NULL

dfNPSByState<-dfNPSByState[order(dfNPSByState$NPSByState,decreasing = TRUE),]

colnames(dfNPSByState)<-c('State','NPSScore')

rownames(dfNPSByState)<-NULL

dfNPSByState

gNPSbyState<-ggplot(dfNPSByState,aes(x=dfNPSByState$State, y=dfNPSByState$NPSScore,fill=dfNPSByState$State)) + geom\_col(stat="identity") + theme\_minimal() + ggtitle("NPS Score by State") + xlab("State") + ylab("Score") +scale\_fill\_discrete(name="State")#Column Graph Based on NPS Score by State

gNPSbyState

L. Column Chart by Count of Customers per Likelihood to Recommend Score

##Likelihood to Recommend Score Breakdown

dfLikelihoodToRe<-data.frame(table(dftest$LikelihoodToReco))

dfLikelihoodToRe

gLikelihoodToRe<-ggplot(dfLikelihoodToRe,aes(x=dfLikelihoodToRe$Var1, y=dfLikelihoodToRe$Freq,fill=dfLikelihoodToRe$Var1)) + geom\_col() + theme\_minimal() + ggtitle("Score Breakdown") + xlab("Score") + ylab("Number of Scores") +scale\_fill\_discrete(name="Score")#Column Graph Based on NPS Score by State

gLikelihoodToRe

M. Correlation Matrix

#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")

N. Linear Modeling

**#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)

O. Linear Modeling Prediction

newdata1 <- data.frame(OverallSatisfaction= 8, GuestRoomH=7, HotelCondition=6, StaffCared=8, CustomerService=9 )

P. Support Vector Models + Plots

#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

##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)

Q. Association Rules Mining

#Association Rule Mining

#Using Audience Demographic Data

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(1:9,59)]

dfARM <- na.omit(dfARM)

str(dfARM)

dfARM$Revenue\_USD <- as.factor(dfARM$Revenue\_USD)

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

ARM <- apriori(dfARM, parameter = list(support=.5,confidence=.3))

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

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

ARMdf1