Natan Alper 5/19/2020

BICI Project

**Introduction/Description**:

The original dataset contained 398 instances (rows) of cars with 9 different characteristics (columns).

The adjusted dataset we are using contains 396 instances (so we can perform 4 cross-fold validation) and we removed the 9th column.

Description of adjusted dataset columns (3 qualitative; 5 quantitative):

#1 MPG = fuel efficiency measured in miles per gallon (mpg), Exs. 9.0, 13.0, 41.5, Type-quantitative

#2 Cylinders = number of cylinders in the engine, Exs. 3, 4, 8, Type-qualitative

#3 Displacement = engine displacement (in cubic inches), Exs. 68.0, 112.0, 455.0, Type-quantitative

#4 Horsepower = engine horsepower, Exs. 46.0, 70.0, 230.0, Type-quantitative

#5 Weight = vehicle weight (in pounds), Exs. 1613, 3615, 5140, Type-quantitative

#6 Acceleration = time to accelerate from 0 to 60 mph (in seconds), Exs. 8.00, 15.50, 24.80, Type-quantitative

#7 Model\_Year = model year, Exs. 73, 79, 82, Type-qualitative

#8 Origin = origin of car (1: American, 2: European, 3: Japanese), Exs. 1, 2, 3, Type-qualitative

\*We will adjust Origin so that 1=American and 0=non-American

The goal is to predict whether the car is of American origin or not. We will be using column #8 (Origin) as our response variable.

**Methodology**: Describe the different models/methods that were tried through the course of the analysis and how you model-performance was assessed. Do not state any results in this section.

Logistic Regression:

Using forward and backward selection for AIC and BIC, we created estimated regression models to make predictions for the response variable. We also used cv-prediciton error and created PCAs to find better models.

Decision Trees:

We ran two models of decision trees by randomly selecting control parameter (cp) values. For the third model, we used the printcp function to find the optimal cp, based on the smallest corresponding xerror. Then, we found the misclass rate of the third model.

SVM:

We created 4 SVM models, based on the quantitative data, by altering the kernel type and cost amount.

1st Model: kernel=linear, cost=1

2nd Model: kernel=linear, cost=10

3rd Model: kernel=radial, cost=1

4th Model: kernel=radial, cost=10

K-nearest Neighbors:

We first standardized the quantitative data, ran the knn model, with the number of neighbors going from 10-500 in steps of 10, for each group in the 4 fold cross validation.

We then plotted the misclass rate with the number of neighbors used to find the smallest misclass rate.

Cluster Analysis:

We ran 4 different cluster model types by adjusting the distance-method and cluster-method.

The first 3 models produced the same majorities of each group (although the predictions varied slightly), resulting in the same misclass rate.

The final model (Manhattan distance, complete linkage), produced a different majority of predictions for the second group, resulting in a better misclass rate!

Random Forests:

We ran the random forest model. We used 1000 trees and randomly selected 4 variables in each iteration.

**Analysis**: Describe what the outcome of each model/analysis was, including relevant diagrams and output. You are not responsible for using every topic from class, however most should be covered.

Logistic Regression:

*AIC backward*:

Coefficients:

(Intercept) Cylinder\_Count4 Cylinder\_Count5 Cylinder\_Count6 Cylinder\_Count8

-21.083713 13.889845 -8.834004 10.453286 18.451751

Displacement Weight Acceleration

0.133533 -0.004947 0.189718

Estimated Model: y = -21.083713 + (0.133533\*Displacement) + (-0.004947\*Weight) + (13.889845\*Cylinder\_Count4) + (-8.834004\*Cylinder\_Count5) + (10.453286\*Cylinder\_Count6) + (18.451751\*Cylinder\_Count8) + (0.189718\* Acceleration)

*AIC forward*:

Coefficients:

(Intercept) Displacement Weight Cylinder\_Count4 Cylinder\_Count5

-21.083713 0.133533 -0.004947 13.889846 -8.834004

Cylinder\_Count6 Cylinder\_Count8 Acceleration

10.453286 18.451751 0.189718

Estimated Model:

y = -21.083713 + (0.133533\*Displacement) + (-0.004947\*Weight) + (13.889846\*Cylinder\_Count4) + (-8.834004\*Cylinder\_Count5) + (10.453286\*Cylinder\_Count6) + (18.451751\*Cylinder\_Count8) + (0.189718\* Acceleration)

*BIC forward*:

Coefficients:

(Intercept) Displacement Weight Acceleration

-4.804060 0.106482 -0.004769 0.191111

Estimated Model:

y = -4.804060 + (0.106482\*Displacement) + (-0.004769\*Weight) + (0.191111\* Acceleration)

*BIC backward*:

Coefficients:

(Intercept) Displacement Weight Acceleration

-4.804060 0.106482 -0.004769 0.191111

Estimated Model:

y = -4.804060 + (0.106482\*Displacement) + (-0.004769\*Weight) + (0.191111\* Acceleration)

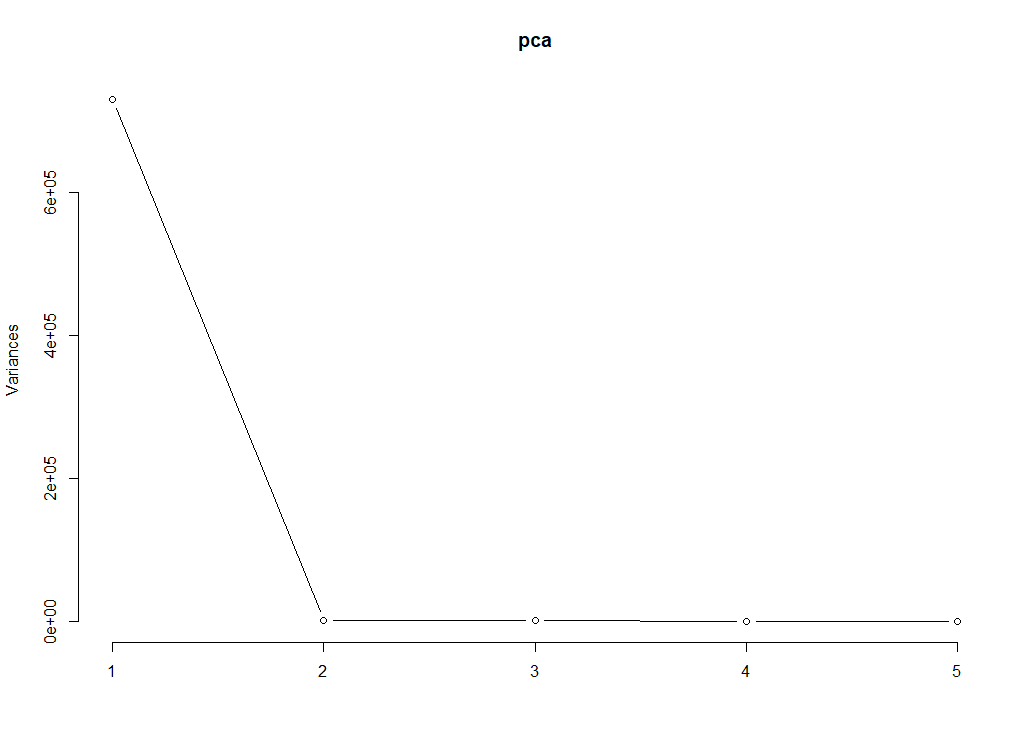
*CV-Prediction Error*:

> cv.glm(autoFactor,glm(Origin~.,family=binomial(),data=autoFactor),K=4)$delta

[1] 0.09655533 0.09282460

*PCA*:

We used a screeplot to determine the number of PCAs to use.



Since there is a large drop from 1 to 2, we conclude that using 2 PCAs should be sufficient.

Models to calculate PCAs:

#PC1=(0.007621734\*MPG) + (0.114227040\*Displacement) + (0.016867082\*Horsepower) + (-0.993281314\*Weight) + (0.001351122\*Acceleration)

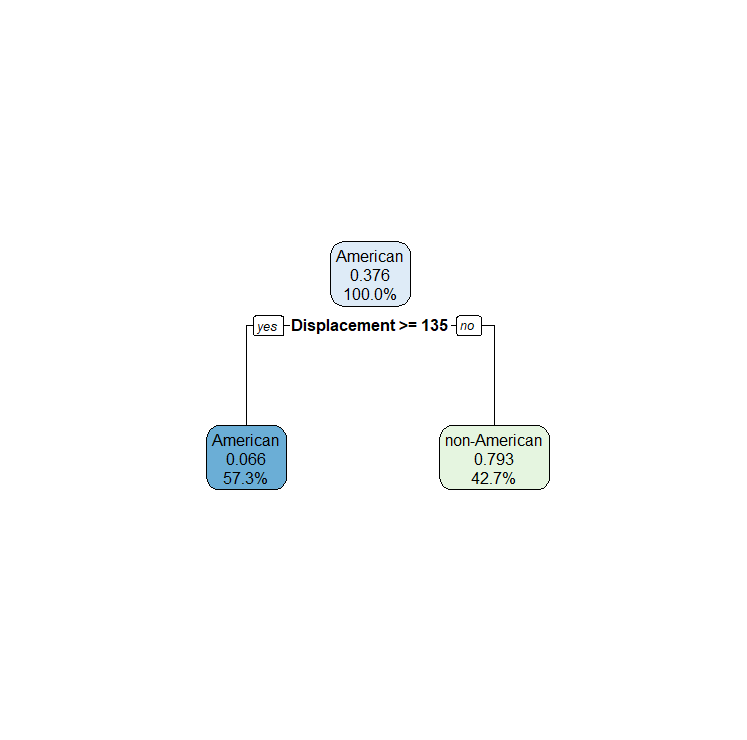
#PC2=(-0.01547889\*MPG) + (0.98530912\*Displacement) + (-0.12085733\*Horsepower) + (-0.11552377\*Weight) + (-0.03121063\*Acceleration)

> misclass\_PCA

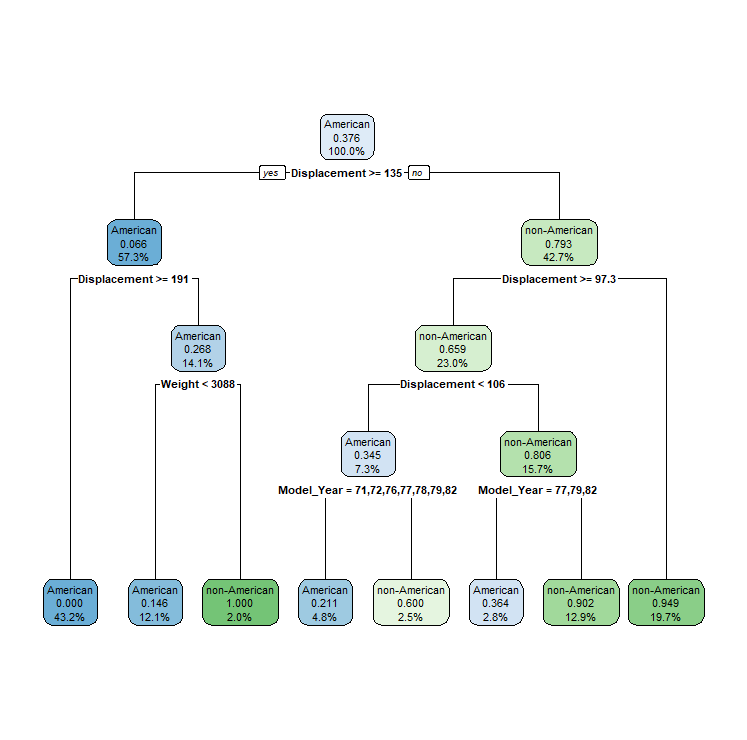
[1] 0.1590909

Decision Trees:

Decision Tree from model 1, using a CP=.1



Decision Tree from model 2, using a CP=.0001



Finding CP (find the cp value where the xerror decreases, hits a minimum, then begins to increase):

CP nsplit rel error xerror xstd

1 0.664430 0 1.00000 1.00000 0.064701

2 0.030201 1 0.33557 0.33557 0.044360

3 0.026846 3 0.27517 0.35570 0.045473

4 0.020134 5 0.22148 0.30201 0.042386

5 0.013423 6 0.20134 0.31544 0.043194

6 0.000100 7 0.18792 0.33557 0.044360

We then made a 3rd model, using the CP=.02 to find the misclass rate (turns out to be the same tree as the 2nd model).

Misclass rate = 0.5151515

SVM:

We compared the output of the misclassification rates between the 4 models to identify the best model.

> misclass\_cross1

[1] 0.5378788

> misclass\_cross2

[1] 0.530303

> misclass\_cross3

[1] 0.5429293

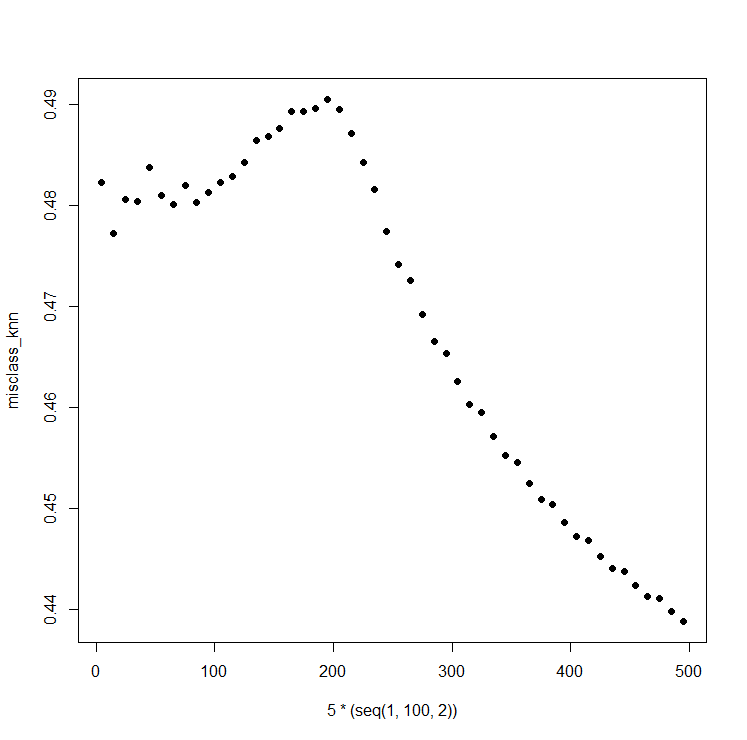
> misclass\_cross4

[1] 0.5378788

2nd model proved to be best based on having the lowest misclass rate.

K-nearest Neighbors:

Plot of misclass rates and the number of neighbors used.



We see that the lowest misclass rate occurs when the number of neighbors (“k”) equals 500.

When k = 500, the misclass rate = 0.4020202

Cluster Analysis:

Misclass rates of all 4 models:

1. ES = Euclidean, Single
2. EC = Euclidean, Complete
3. MS = Manhattan, Single
4. MC = Manhattan, Complete

misclassES

[1] 0.3762626

> misclassEC

[1] 0.3762626

> misclassMS

[1] 0.3762626

> misclassMC

[1] 0.2979798

Optimal model is ClustersMC (Manhattan, Complete) based on the misclass rate.

Random Forests:

Misclass rate after running the random forest model:

misclassRF = 0.07323232

**Conclusion**: Describe the model that we should use for predicting the class of the 𝑦-variable and assess its performance one final time.

The model which performed best at predicting the class of the y-variable was Random Forests. When ran again, we got a misclass rate of 0.07070707 (an even lower misclass rate).

**Appendix**: Include all code used in the analysis along with some description of what the code is doing.

# Natan Alper

# BICI Final Project

### 1st we will read the data in R and clean it so

### it is ready for analysis!

# Directory path to the file- autoData

path <- "C:/Users/Natan/Documents/@YeshivaUniversity/Spring2020/Business Intelligence & Consumer Insights/Final Project/"

# Alternatively, we can import the dataset manually

# Input autoMPG data from csv file into RStudio

autoFileRaw <- read.csv(file = paste0(path,"/autoData.csv"))

# Rename columns of autoFile

names(autoFileRaw) <- c("MPG", "Cylinder\_Count", "Displacement", "Horsepower", "Weight", "Acceleration", "Model\_Year", "Origin", "Car\_Name")

# Remove the Car\_Name col (since not using) and last 2 rows (for 4-fold validation)

autoFile <- data.frame(autoFileRaw[-c(nrow(autoFileRaw), (nrow(autoFileRaw)-1)),-(ncol(autoFileRaw))])

# 3 cols that should be qualitative (cols 2, 7, 8)

autoFile$Cylinder\_Count <- as.character(autoFile$Cylinder\_Count)

autoFile$Model\_Year <- as.character(autoFile$Model\_Year)

autoFile$Origin <- as.character(autoFile$Origin)

autoFile$Origin[autoFile$Origin=="1"] <- "American"

autoFile$Origin[autoFile$Origin!="American"] <- "non-American"

# 5 cols that should be quantitative (1, 3, 4, 5, 6)

# autoFile$MPG is numeric

# autoFile$Displacment is numeric

# autoFile$Acceleration is numeric

autoFile$Horsepower <- as.numeric(autoFile$Horsepower)

autoFile$Weight <- as.numeric(autoFile$Weight)

autoQuant <- autoFile[,c(1, 3, 4, 5, 6, 8)]

### Now we will throw some models at this data and see

### what results they come up with!

# The groups we will use for 4-fold cross validation

# Only run samp code once

samp=sample((1:396), 396, replace = FALSE)

# We create 4 folds

g1=samp[1:99]

g2=samp[100:198]

g3=samp[199:297]

g4=samp[298:396]

Groups <- data.frame(g1,g2,g3,g4)

########## Logistic Regression ##########

library(boot)

# Converting character data into factors & Origin into 1s & 0s

# 1=American, 0=non-American

autoFactor <- data.frame(autoFile$MPG,as.factor(autoFile$Cylinder\_Count), autoFile$Displacement, autoFile$Horsepower, autoFile$Weight, autoFile$Acceleration, as.factor(autoFile$Model\_Year), autoFile$Origin)

names(autoFactor) <- c("MPG", "Cylinder\_Count", "Displacement", "Horsepower", "Weight", "Acceleration", "Model\_Year", "Origin")

autoFactor$Origin <- as.character(autoFactor$Origin)

autoFactor$Origin[autoFactor$Origin=="American"] <- 1

autoFactor$Origin[autoFactor$Origin!=1] <- 0

autoFactor$Origin <- as.factor(autoFactor$Origin)

### By default, step() selects models based on AIC.

# Backward Selection:

step(glm(Origin~.,family=binomial(),data=autoFactor))

# Forward Selection:

step(glm(Origin~1,family=binomial(),data=autoFactor),direction="forward",scope=list(glm(Origin~1,family=binomial(),data=autoFactor),upper=glm(Origin~.,family=binomial(),data=autoFactor)))

### To use BIC, we specify k=log(n) in the step() func (n=# of obs)

# Forward Selection:

step(glm(Origin~.,family=binomial(),data=autoFactor),k=log(396))

# Backward Selection:

step(glm(Origin~1,family=binomial(),data=autoFactor),direction="forward",scope=list(glm(Origin~1,family=binomial(),data=autoFactor),upper=glm(Origin~.,family=binomial(),data=autoFactor)),k=log(396))

### cv-prediction error

cv.glm(autoFactor,glm(Origin~.,family=binomial(),data=autoFactor),K=4)$delta

## We get 2 measures of avg pred error (only use the 1st)

# Getting PCA for quantitative data

pca=prcomp(autoQuant[-6])

summary(pca)

# Look for the large drop in screeplot

screeplot(pca,type="lines")

# New data is located in:

pca$x

##The coefficients in the linear combination are located here:

pca$rotation

#PC1=(0.007621734\*MPG)+(-0.114227040\*Displacement)+(0.016867082\*Horsepower)+(-0.993281314\*Weight)+(0.001351122\*Acceleration)

#PC2=(-0.01547889\*MPG)+(0.98530912\*Displacement)+(-0.12085733\*Horsepower)+(-0.11552377\*Weight)+(-0.03121063\*Acceleration)

PCdata=data.frame(pca$x,autoFactor$Origin)

errors=c()

for(i in 1:396){

mod=lm(autoFactor.Origin~.,data=PCdata[-i,c(1:2,6)])

predicted=predict(mod,newdata=PCdata[i,c(1:2,6)])

errors=c(errors,as.numeric(PCdata$autoFactor.Origin[i])-as.numeric(predicted))

}

errors

errors[errors<=0] <- 0

errors[errors!=0] <- 1

errors

misclass\_PCA <- sum(autoFactor$Origin!=errors)/length(errors)

misclass\_PCA

########## Decision Trees ##########

library(rpart)

library(rpart.plot)

autoTree1 <- rpart(Origin~., data=autoFile, control=rpart.control(cp=.1))

rpart.plot(autoTree1,digits=-3)

autoTree2<- rpart(Origin~., data=autoFile, control=rpart.control(cp=.0001))

rpart.plot(autoTree2,digits=-3)

# Find optimal cp based on XERROR

printcp(autoTree2,digits=5)

autoTree3 <- rpart(Origin~., data=autoFile, control=rpart.control(cp=.02))

rpart.plot(autoTree3,digits=-3)

predictions <- c()

for(i in 1:4){

autoTree3 <- rpart(Origin~., data=autoFile[-Groups[,i],],control=rpart.control(cp=.02))

predictions\_per\_fold <- predict(autoTree3,type="class",newdata=autoFile[Groups[,i],])

predictions <- c(predictions,as.character(predictions\_per\_fold))

}

misclass\_cross <- sum(autoFile$Origin!=predictions)/length(predictions)

misclass\_cross

########## SVM Model ##########

library(e1071)

# Only use quant variables plus Origin = autoQuant

# 1st Model: kernel=linear, cost=1

preds <- c()

for(i in 1:4){

modelSVM <- svm(Origin~.,data=autoQuant[-Groups[,i],],kernel="linear",type="C-classification",cost=1)

predictions\_per\_fold <- predict(modelSVM,type="class",newdata=autoQuant[Groups[,i],])

preds <- c(preds,as.character(predictions\_per\_fold))

}

misclass\_cross1 <- sum(autoQuant$Origin!=preds)/length(preds)

# 2nd Model: kernel=linear, cost=10

preds <- c()

for(i in 1:4){

modelSVM <- svm(Origin~.,data=autoQuant[-Groups[,i],],kernel="linear",type="C-classification",cost=10)

predictions\_per\_fold <- predict(modelSVM,type="class",newdata=autoQuant[Groups[,i],])

preds <- c(preds,as.character(predictions\_per\_fold))

}

misclass\_cross2 <- sum(autoQuant$Origin!=preds)/length(preds)

# 3rd Model: kernel=radial, cost=1

preds <- c()

for(i in 1:4){

modelSVM <- svm(Origin~.,data=autoQuant[-Groups[,i],],kernel="radial",type="C-classification",cost=1)

predictions\_per\_fold <- predict(modelSVM,type="class",newdata=autoQuant[Groups[,i],])

preds <- c(preds,as.character(predictions\_per\_fold))

}

misclass\_cross3 <- sum(autoQuant$Origin!=preds)/length(preds)

# 4th Model: kernel=radial, cost=10

preds <- c()

for(i in 1:4){

modelSVM <- svm(Origin~.,data=autoQuant[-Groups[,i],],kernel="radial",type="C-classification",cost=10)

predictions\_per\_fold <- predict(modelSVM,type="class",newdata=autoQuant[Groups[,i],])

preds <- c(preds,as.character(predictions\_per\_fold))

}

misclass\_cross4 <- sum(autoQuant$Origin!=preds)/length(preds)

# Find lowest misclass rate to identify the optimal model

misclass\_cross1

misclass\_cross2

misclass\_cross3

misclass\_cross4

# Optimal Model= 2nd Model: kernel=linear, cost=10

########## K-nearest Neighbors ##########

library(class)

# Use autoQuant

# Standardize data

for(i in 1:(ncol(autoQuant)-1)){ ## Note that response variable is assumed to be the last column here

autoQuant[,i] <- (autoQuant[,i]-mean(autoQuant[,i]))/sd(autoQuant[,i])

}

preds <- c()

misclass\_knn <- c()

for(k in 10\*(1:50)){

for(i in 1:4){

predictions\_per\_fold <- knn(train=autoQuant[-Groups[,i],-6],test=autoQuant[Groups[,i],-6],cl=autoQuant[-Groups[,i],6],k=k)

preds <- c(preds,as.character(predictions\_per\_fold))

}

misclass\_knn <- c(misclass\_knn, sum(autoQuant[-Groups[,i],6]!=preds)/length(preds))

}

plot(10\*(1:50), misclass\_knn,pch=16) # finding a choice of "k" which would lead to the lowest misclass rate.

misclass\_knn

########## Cluster Analysis ##########

autoQuant <- autoFile[,c(1, 3, 4, 5, 6)]

# Find distances between observations using "euclidean" and "manhattan"

Eucl.Distances= dist(autoQuant,method="euclidean")

Manh.Distances= dist(autoQuant,method="manhattan")

# 4 different clusters, adjusting distance-method and cluster-method

# Abbreviations: (E=eucl, M=manh) (S=single, C=complete); ES, EC, MS, MC

ClustersES=hclust(Eucl.Distances,method="single")

ClustersEC=hclust(Eucl.Distances,method="complete")

ClustersMS=hclust(Manh.Distances,method="single")

ClustersMC=hclust(Manh.Distances,method="complete")

### Misclass for ClustersES

GroupsES=cutree(ClustersES,k=2)

GroupsES

# Find the # of positions in Group 1 that = non-American in autoFile

sum(autoFile$Origin[GroupsES==1]=="non-American")#149

# ... = American

sum(autoFile$Origin[GroupsES==1]=="American")#246

## Majority of Group 1="American"-set all Group 1 positions in pred to "American"

PredES=rep(NA,396)

PredES[GroupsES==1]="American"

# Find the # of positions in Group 2 that = non-American in autoFile

sum(autoFile$Origin[GroupsES==2]=="non-American")#0

# ... = American

sum(autoFile$Origin[GroupsES==2]=="American")#1

## Majority of Group 2="American"-set all Group 1 positions in pred to "American"

PredES[GroupsES==2]="American"

misclassES=sum(autoFile$Origin[1:396]!=PredES)/396

### Misclass for ClustersEC

GroupsEC=cutree(ClustersEC,k=2)

GroupsEC

# Group 1 pos = non-American

sum(autoFile$Origin[GroupsEC==1]=="non-American")#148

# Group 1 pos = American

sum(autoFile$Origin[GroupsEC==1]=="American")#159

## Maj Group 1 = American, set all of Group 1 to American

PredEC=rep(NA,396)

PredEC[GroupsEC==1]="American"

# Group 2 pos = non-American

sum(autoFile$Origin[GroupsEC==2]=="non-American")#1

# Group 2 pos = American

sum(autoFile$Origin[GroupsEC==2]=="American")#88

## Maj Group 2 = American, set all of Group 2 to American

PredEC[GroupsEC==2]="American"

misclassEC=sum(autoFile$Origin[1:396]!=PredEC)/396

### Misclass for ClustersMS

GroupsMS=cutree(ClustersMS,k=2)

GroupsMS

# Group 1 pos = non-American

sum(autoFile$Origin[GroupsMS==1]=="non-American")#149

# Group 1 pos = American

sum(autoFile$Origin[GroupsMS==1]=="American")#246

## Maj Group 1 = American, set all of Group 1 to American

PredMS=rep(NA,396)

PredMS[GroupsMS==1]="American"

# Group 2 pos = non-American

sum(autoFile$Origin[GroupsMS==2]=="non-American")#0

# Group 2 pos = American

sum(autoFile$Origin[GroupsMS==2]=="American")#1

## Maj Group 2 = American, set all of Group 2 to American

PredMS[GroupsMS==2]="American"

misclassMS=sum(autoFile$Origin[1:396]!=PredMS)/396

### Misclass for ClustersMC

GroupsMC=cutree(ClustersMC,k=2)

GroupsMC

# Group 1 pos = non-American

sum(autoFile$Origin[GroupsMC==1]=="non-American")#3

# Group 2 pos = American

sum(autoFile$Origin[GroupsMC==1]=="American")#132

## Maj Group 1 = American, set all of Group 1 to American

PredMC=rep(NA,396)

PredMC[GroupsMC==1]="American"

# Group 2 pos = non-American

sum(autoFile$Origin[GroupsMC==2]=="non-American")#146

# Group 2 pos = American

sum(autoFile$Origin[GroupsMC==2]=="American")#115

## Maj Group 2 = American, set all of Group 2 to American

PredMC[GroupsMC==2]="non-American"

misclassMC=sum(autoFile$Origin[1:396]!=PredMC)/396

# Find lowest misclass rate to identify the optimal model

misclassES

misclassEC

misclassMS

misclassMC

# Optimal Model= ClustersMC

########## Random Forests ##########

library(randomForest)

RF=randomForest(as.factor(Origin)~.,data=autoFile, ntree=1000, mtry=4, type="class")

# Getting predictions

predictionsRF=predict(RF)

# Getting misclass rate

misclassRF=sum(autoFile$Origin!=predictionsRF)/396

misclassRF