IST 707 Final Project

Samantha Riedener

March 15, 2019

Introduction

Graduate school is one of the biggest decisions a person can make about their education. It is a difficult undertaking to choose to pursue an even higher level of education beyond a Bachelor's Degree. Not only is graduate school an additional commitment in terms of time and effort, but it's also a large financial commitement as well. For these reasons, it's important to be ready to fully embrace the decision of attending graduate school. The first step in this process, though, is being admitted to a program.

When submitting a grad school application, there are some commonalities in what a program will expect to receive. standardized testing scores, undergraduate performance, letters of intent and recommendation, and past experience are all accounted for in the application. How these measures stack up for an individual will determine if they will be accepted into their program of choice. There are often only basic guidelines published by graduate schools to know what their standards are for accepted applications, leaving many potential students uncertain as to the strength of their application.

By using data mining techniques, it can be discovered which factors in a graduate school application are most important. Addtionally, a strong predictive method can be developed to be able to predict which students are most likely to be admitted to a graduate school program. By making these determinations, it could help reduce the stress and uncertainty that goes into this major process of continuing one's educaiton to higher levels. Knowing how an individual application measures up to so many others can help people know where to focus their efforts in order to have the best chances of reaching their goals of graduate school admission.

Analysis

The Data

The data set used for this analysis is a collection of 500 graduate school applications, measured by 8 variables: GRE Score, TOEFL Score, University Rating, Statement of Purpose, Letter of Recommendation, CGPA, Research experience, and Chance of Admission. There is also an ID number column used solely for identification purposes. These applications are from students in India, giving an international perspective into the graduate school admission experience. The variables for this data set are defined as follows:

GRE Score - A value for the score received on the Graduate Record Examinations. This is a standardized test that is required by most graduate programs for prospective students to take to give a measure of proficiency in reading, writing, and mathematics. Possible scores range from 260-340

TOEFL Score - A value for the score recevied on the Test of English as a Foreign Language. This is a standardized test that is required to be taken by students who wish to attend a United States graduate school but do not have English as their first language. The test is meant to measure proficiency in Engish. Possible scores range from 0 to 120.

University Rating - A qualitative value given to the quality of the undergraduate university that the student attended. This is a scale of whole numbers ranging from 1 to 5, with 1 being the lowest and 5 being the highest quality.

Statement of Purpose (SOP) - A qualitative value for the strength of the writing of the statement of purpose. The statement of purpose is a formally written piece that prospective students must write to describe why they wish to attend graduate school and what they hope to accomplish in their program. The possible values of this range from 1 to 5, increasing incrementally by 0.5, where one is the lowest quality and 5 is the highest.

Letter of Recommendation (LOR) - a qualitative value for the strength of the letter of recommendation for the student. A letter of recommendation is a formal letter written by someone who personally knows the student, usually a former professor or boss, describing why they believe the student to be a good candidate for the graduate school program. The possible values of this range from 1 to 5, increasing incrementally by 0.5, where 1 is the lowest quality and 5 is the highest.

CPGA - a value for the undergraduate Grade Point Average of the student. CPGA reflects the overal grade received by the student during their undergraduate career. CGPA is measured on a 10 point scale, different from the traditional 4 point scale used in the United States. It is of note that a CGPA above a 6 is considered an A, the highest possible letter grade.

Research - A binary value indicating whether or not a student has had research experience. A 0 indicates no experience, a 1 indicates the student has experence.

Chance of Admit - A percentage, expressed as a decimal, of the probability of admission based on the other seven variables.

The data set was loaded in and examined for completeness, data types, and for abnormal values within the columns.

```
library(arules)
```

```
## Warning: package 'arules' was built under R version 3.5.2
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'arules'
```

```
## The following objects are masked from 'package:base':
##
##     abbreviate, write

library(arulesViz)

## Warning: package 'arulesViz' was built under R version 3.5.2

## Loading required package: grid

library(plyr)
library(rpart)

## Warning: package 'rpart' was built under R version 3.5.2

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.5.2

library(rattle)

## Warning: package 'rattle' was built under R version 3.5.2

## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.

library(stats)
library(factoextra)

## Warning: package 'factoextra' was built under R version 3.5.2

## Loading required package: ggplot2

## Welcome! Related Books: `Practical Guide To Cluster Analysis in R` at http
s://goo.gl/13EFCZ

library(proxy)

## Warning: package 'proxy' was built under R version 3.5.2

##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist
```

```
## The following object is masked from 'package:base':
##
##     as.matrix

library(naivebayes)

## Warning: package 'naivebayes' was built under R version 3.5.2

library(e1071)

## Warning: package 'e1071' was built under R version 3.5.2

graduate <- read.csv("C:/Users/User/Desktop/Syracuse/Admission_Predict_Ver1.1
.csv")
str(graduate)

## 'data.frame':    500 obs. of  9 variables:
##  $ Serial.No.       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ GRE.Score        : int  337 324 316 322 314 330 321 308 302 323 ...
##  $ TOEFL.Score      : int  118 107 104 110 103 115 109 101 102 108 ...
##  $ University.Rating: int  4 4 3 3 2 5 3 2 1 3 ...
##  $ SOP              : num  4.5 4 3 3.5 2 4.5 3 3 2 3.5 ...
##  $ LOR              : num  4.5 4.5 3.5 2.5 3 3 4 4 1.5 3 ...
##  $ CGPA             : num  9.65 8.87 8 8.67 8.21 9.34 8.2 7.9 8 8.6 ...
##  $ Research         : int  1 1 1 1 0 1 1 0 0 0 ...
##  $ Chance.of.Admit  : num  0.92 0.76 0.72 0.8 0.65 0.9 0.75 0.68 0.5 0.45
...

length(which(is.na(graduate)))

## [1] 0

summary(graduate)

##    Serial.No.      GRE.Score      TOEFL.Score    University.Rating
##  Min.   :  1.0   Min.   :290.0   Min.   : 92.0   Min.   :1.000
##  1st Qu.:125.8   1st Qu.:308.0   1st Qu.:103.0   1st Qu.:2.000
##  Median :250.5   Median :317.0   Median :107.0   Median :3.000
##  Mean   :250.5   Mean   :316.5   Mean   :107.2   Mean   :3.114
##  3rd Qu.:375.2   3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000
##  Max.   :500.0   Max.   :340.0   Max.   :120.0   Max.   :5.000
##       SOP             LOR             CGPA         Research
##  Min.   :1.000   Min.   :1.000   Min.   :6.800   Min.   :0.00
##  1st Qu.:2.500   1st Qu.:3.000   1st Qu.:8.127   1st Qu.:0.00
##  Median :3.500   Median :3.500   Median :8.560   Median :1.00
##  Mean   :3.374   Mean   :3.484   Mean   :8.576   Mean   :0.56
##  3rd Qu.:4.000   3rd Qu.:4.000   3rd Qu.:9.040   3rd Qu.:1.00
##  Max.   :5.000   Max.   :5.000   Max.   :9.920   Max.   :1.00
##  Chance.of.Admit
##  Min.   :0.3400
##  1st Qu.:0.6300
##  Median :0.7200
```

```
##  Mean   :0.7217
##  3rd Qu.:0.8200
##  Max.   :0.9700
```

The results above showed that the data set was complete and that all the values for the columns fell within the expected ranges.

Several copies of the data set were made with different transformation of the data applied so that the different data mining techniques would be useable. Not every technique can use the information in the exact same way in order to work.

```
# ID column removed
graduate <- graduate[,-1]

# Second copy of data set made with Chance of Admit, Research as a factors
graduate2 <- graduate
graduate2$Research <- as.factor(graduate2$Research)
graduate2$Chance.of.Admit[graduate2$Chance.of.Admit >= 0.67] <- 1
graduate2$Chance.of.Admit[graduate2$Chance.of.Admit < 0.67] <- 0
graduate2$Chance.of.Admit <- as.factor(graduate2$Chance.of.Admit)
levels(graduate2$Chance.of.Admit)[levels(graduate2$Chance.of.Admit)=="1"] <-
"Admit"
levels(graduate2$Chance.of.Admit)[levels(graduate2$Chance.of.Admit)=="0"] <-
"Decline"
```

It was decided that a 67% confidence was necessary for a student to be considered "admitted". By changing this variable to a factor, this allowed for much easier predictive analyses.

A third version of the data set was created in which all variables were discretized or converted to factors and renamed.

```
graduate_ar <- graduate

graduate_ar$SOP <- as.factor(graduate_ar$SOP)
graduate_ar$SOP <- revalue(graduate_ar$SOP, c("1"="SOP.1", "1.5"="SOP.1.5", "
2"="SOP.2", "2.5"="SOP.2.5", "3"="SOP.3", "3.5"="SOP.3.5", "4"="SOP.4", "4.5"
="SOP.4.5", "5"="SOP.5"))
graduate_ar$LOR <- as.factor(graduate_ar$LOR)
graduate_ar$LOR <- revalue(graduate_ar$LOR, c("1"="LOR.1", "1.5"="LOR.1.5", "
2"="LOR.2", "2.5"="LOR.2.5", "3"="LOR.3", "3.5"="LOR.3.5", "4"="LOR.4", "4.5"
="LOR.4.5", "5"="LOR.5"))
graduate_ar$University.Rating <- as.factor(graduate_ar$University.Rating)
graduate_ar$University.Rating <- revalue(graduate_ar$University.Rating, c("1"
="UR.1", "2"="UR.2",  "3"="UR.3",  "4"="UR.4",  "5"="UR.5"))
graduate_ar$GRE.Score <- discretize(graduate_ar$GRE.Score, method = "interval
")
graduate_ar$TOEFL.Score <- discretize(graduate_ar$TOEFL.Score, method = "inte
rval")
graduate_ar$CGPA <- discretize(graduate_ar$CGPA, method = "interval")
```

```
graduate_ar$Chance.of.Admit[graduate_ar$Chance.of.Admit >= 0.67] <- 1
graduate_ar$Chance.of.Admit[graduate_ar$Chance.of.Admit < 0.67] <- 0
graduate_ar$Chance.of.Admit <- as.factor(graduate_ar$Chance.of.Admit)
levels(graduate_ar$Chance.of.Admit)[levels(graduate_ar$Chance.of.Admit)=="1"]
<- "Admit"
levels(graduate_ar$Chance.of.Admit)[levels(graduate_ar$Chance.of.Admit)=="0"]
<- "Decline"
graduate_ar$Research <- as.factor(graduate_ar$Research)
levels(graduate_ar$Research)[levels(graduate_ar$Research)=="1"] <- "Yes"
levels(graduate_ar$Research)[levels(graduate_ar$Research)=="0"] <- "No"
```

A fourth version of the data set was created as a scaled version. Scaling converts all the variables such that their average is 0 and the standard deviation is 1. This allows for easy clustering analysis.

```
graduate_scale <- graduate
graduate_scale <- scale(graduate_scale)
graduate_matrix <- data.matrix(graduate_scale)
# Data saved as matrix for addtional clustering techniques
```

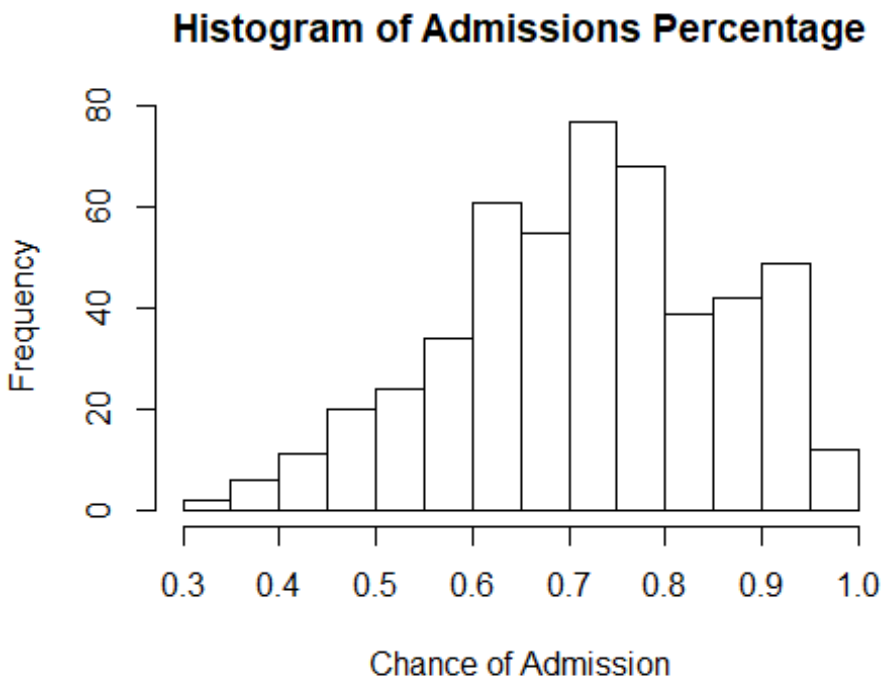
Exploratory Data Analysis

Tables for each of the non score based variables were created to get a feel for how each of these values were distributed among the students.

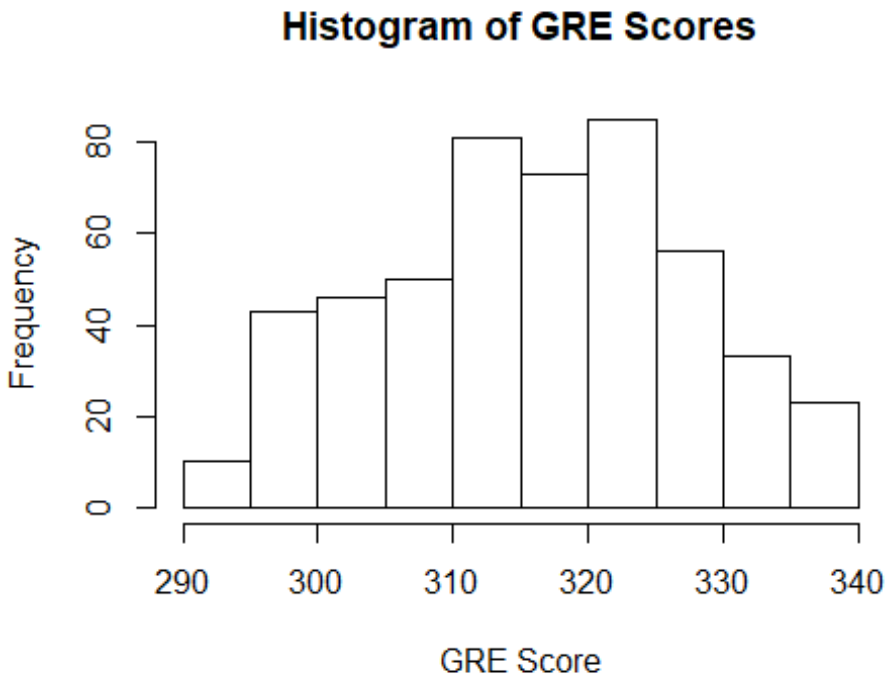
```
table(graduate$SOP)

##
##   1 1.5   2 2.5   3 3.5   4 4.5   5
##   6  25  43  64  80  88  89  63  42

table(graduate$LOR)

##
##   1 1.5   2 2.5   3 3.5   4 4.5   5
##   1  11  46  50  99  86  94  63  50

table(graduate$University.Rating)

##
##   1   2   3   4   5
##  34 126 162 105  73

table(graduate$Research)

##
##   0   1
## 220 280
```

However, it is easiest to visualize the data with histograms and barcharts to get a good feel for what the data set contains.

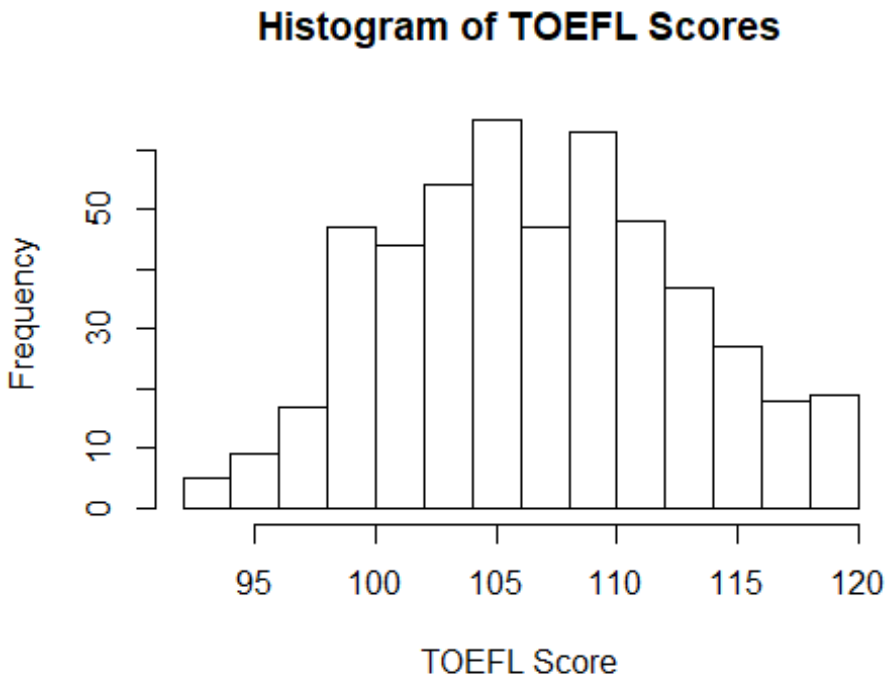
```r
hist(graduate$Chance.of.Admit, main = "Histogram of Admissions Percentage", x
lab="Chance of Admission")
```



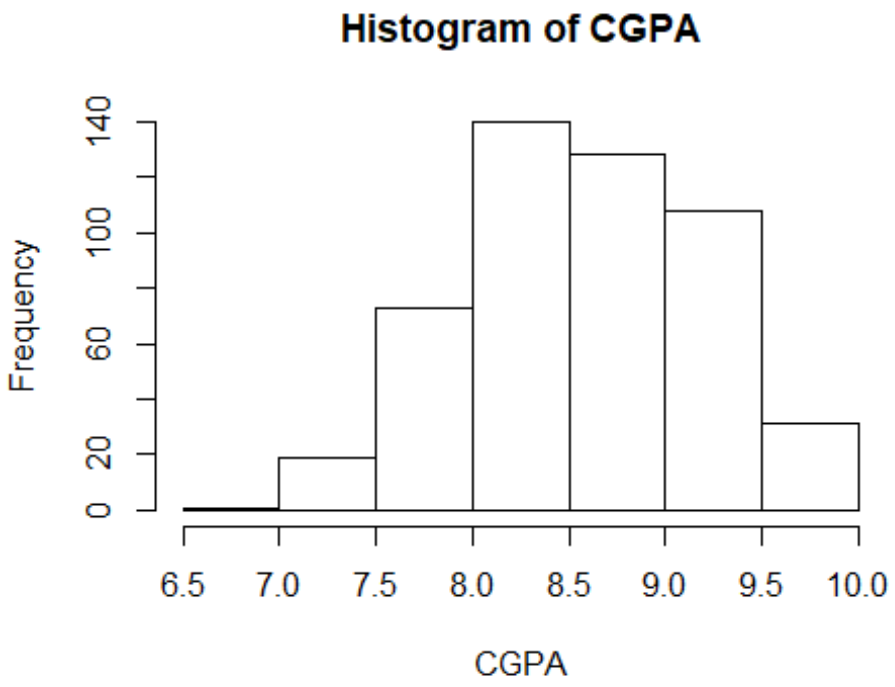

```r
hist(graduate$GRE.Score, main = 'Histogram of GRE Scores', xlab="GRE Score")
```



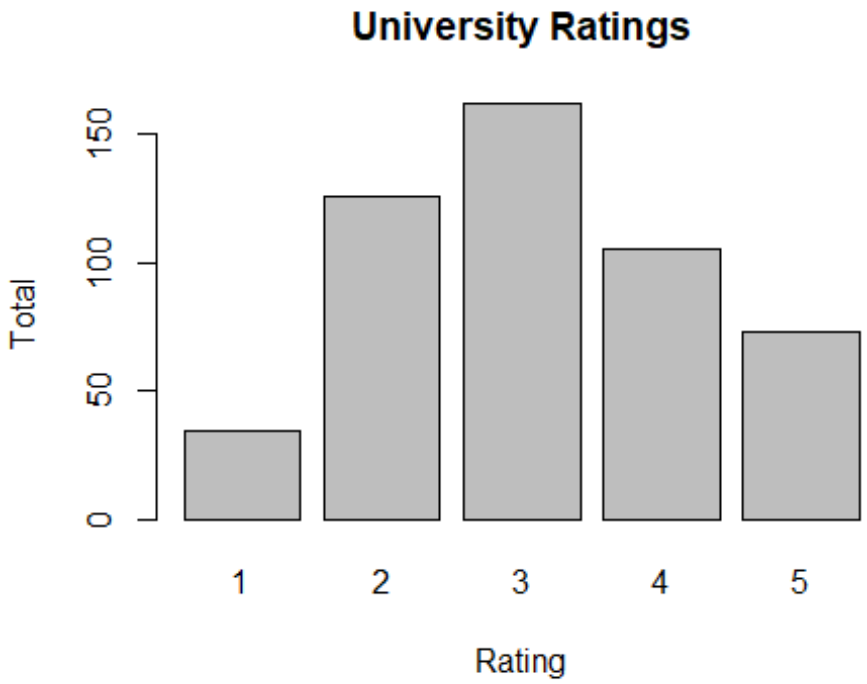

```
hist(graduate$TOEFL.Score, main="Histogram of TOEFL Scores", xlab="TOEFL Scor
e")
```



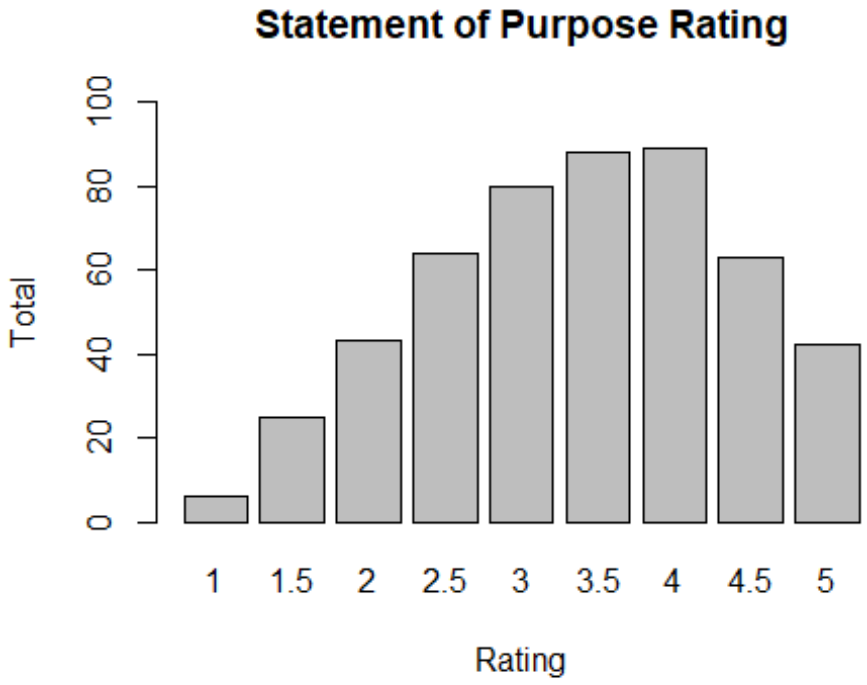
```r
hist(graduate$CGPA, main="Histogram of CGPA", xlab="CGPA")
```



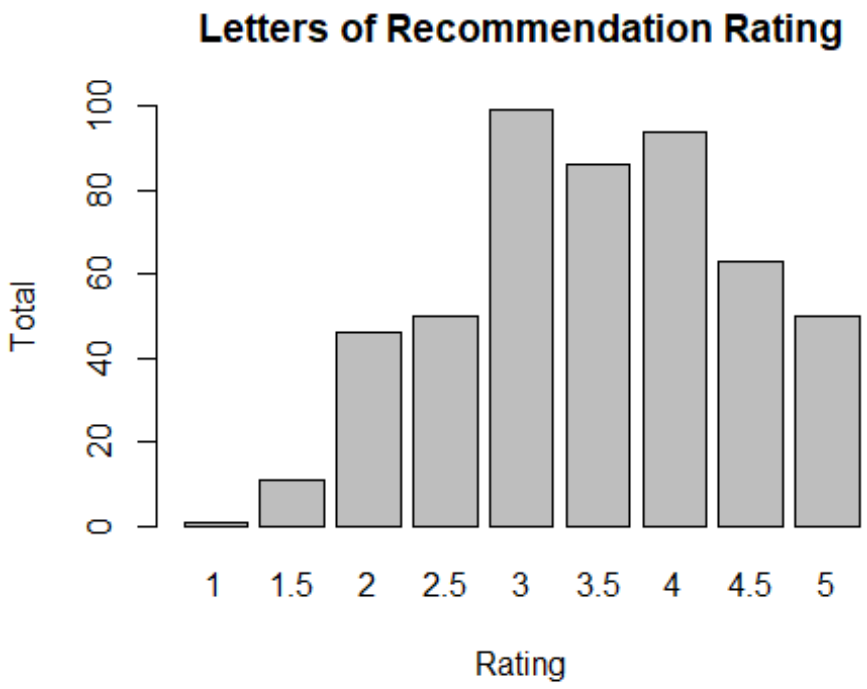

```r
barplot(table(graduate$University.Rating), main ="University Ratings", ylab="
Total", xlab="Rating")
```



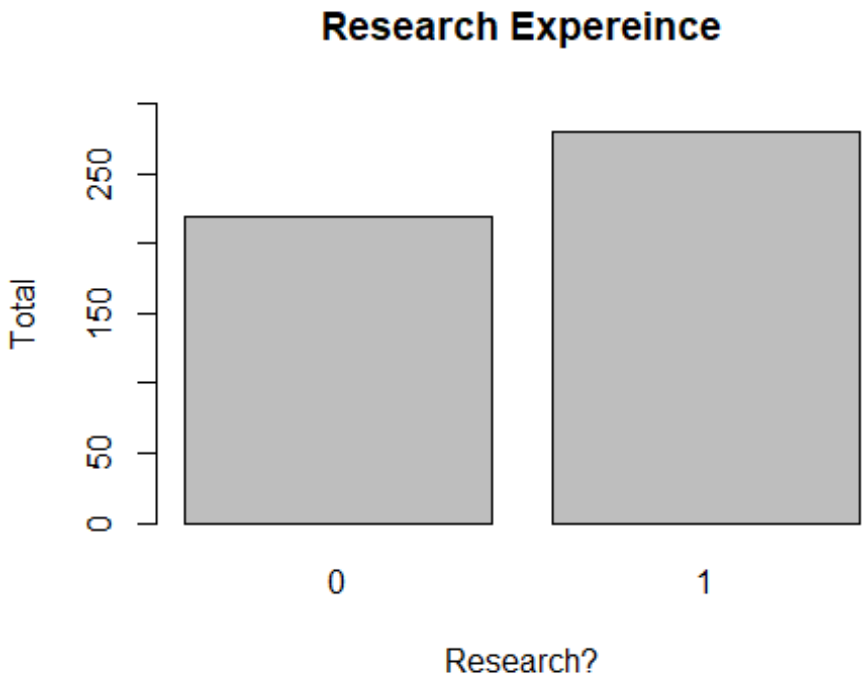

```r
barplot(table(graduate$SOP),main = "Statement of Purpose Rating", ylab='Total
', xlab="Rating", ylim=c(0,100))
```



```r
barplot(table(graduate$LOR),main = "Letters of Recommendation Rating", ylab='
Total', xlab="Rating", ylim=c(0,100))
```




```r
barplot(table(graduate$Research), main = 'Research Expereince', ylab = 'Total
', xlab = 'Research?', ylim=c(0,300))
```



Based on the graphs, each of the categories roughly follows a normal curve, with some skewness towards the higher end. Examining each values in detail gives a bit more insight into the characteristics of the students in the data set:

Chance of Admit - Most students have a higher chance to admit than be declined. Based on the 67% confidence established above, the data set is divided into 2/3 Admit and 1/3 Decline.

GRE Score - These scores range from 290 to 340. This variable has a large representation of the possible scores on this test.

TOEFL Score - These scores range from 92 to 120. All of these scores are considered to be in the highest echelon of scores.

CGPA - The scores range from 6.8 to 9.92, meaning that each of these students has earned the equivalent of an A. The data still skews much more to the higher possible CGPA's.

University Rating - A little over a third of the students are in the highest two categories of univeristy.

SOP and LOR - Both these variables skew much more towards the higher ratings than the other variables. More than half of the students are in the upper rankings in each of these variables.

Research - The data set is split close to 50% for having research or not, but there are still more students who have research experience than do not.

Overall, the group of students here represents a very well performing group. They have high grades, exceptional skills with English as a second language, and come well recommended from others. There is still enough variety in the data set in order to make determination about important variables and predictions.

Associative Rules Mining

The goal of the associative rules mining was to find if there were any attributes that occured more frequently in the admitted student's profiles. This was done by reading in the discretized version of the data set as a set of transaction. Sets of rules were then determined based on minimum support and confidence levels. In each iteration, the right hand side of the rules was always Admit, as this is the desired outcome for the applications. Ultimately, a small but strong set of rules was the desried outcome. By fine tuning the support and confidence levels and slightly raising them each time, a set of strong rules was made. This series of rules was then sorted by lift, support, and confidence.

```
write.csv(graduate_ar, "C:/Users/User/Desktop/Syracuse/graduate_ar.csv")
graduate_transaction <- read.transactions("C:/Users/User/Desktop/Syracuse/gra
duate_ar.csv", format = "basket", sep = ",")
first.rules <- apriori(graduate_transaction, parameter = list(supp = 0.01, co
nf = 0.85), appearance = list(default="lhs", rhs="Admit"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 5
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[544 item(s), 501 transaction(s)] done [0.00s].
## sorting and recoding items ... [35 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 7 8 done [0.00s].
## writing ... [1116 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
second.rules <- apriori(graduate_transaction, parameter = list(supp = 0.05, c
onf = 0.85), appearance = list(default="lhs", rhs="Admit"))
```

```
## Apriori
##
```

```
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5    0.05      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 25
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[544 item(s), 501 transaction(s)] done [0.00s].
## sorting and recoding items ... [32 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [156 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
third.rules <- apriori(graduate_transaction, parameter = list(supp = 0.05, co
nf = 0.9), appearance = list(default="lhs", rhs="Admit"))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5    0.05      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 25
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[544 item(s), 501 transaction(s)] done [0.00s].
## sorting and recoding items ... [32 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.00s].
## writing ... [151 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
fourth.rules <- apriori(graduate_transaction, parameter = list(supp = 0.1, co
nf = 0.9), appearance = list(default="lhs", rhs="Admit"))
```

```
## Apriori
##
## Parameter specification:
```

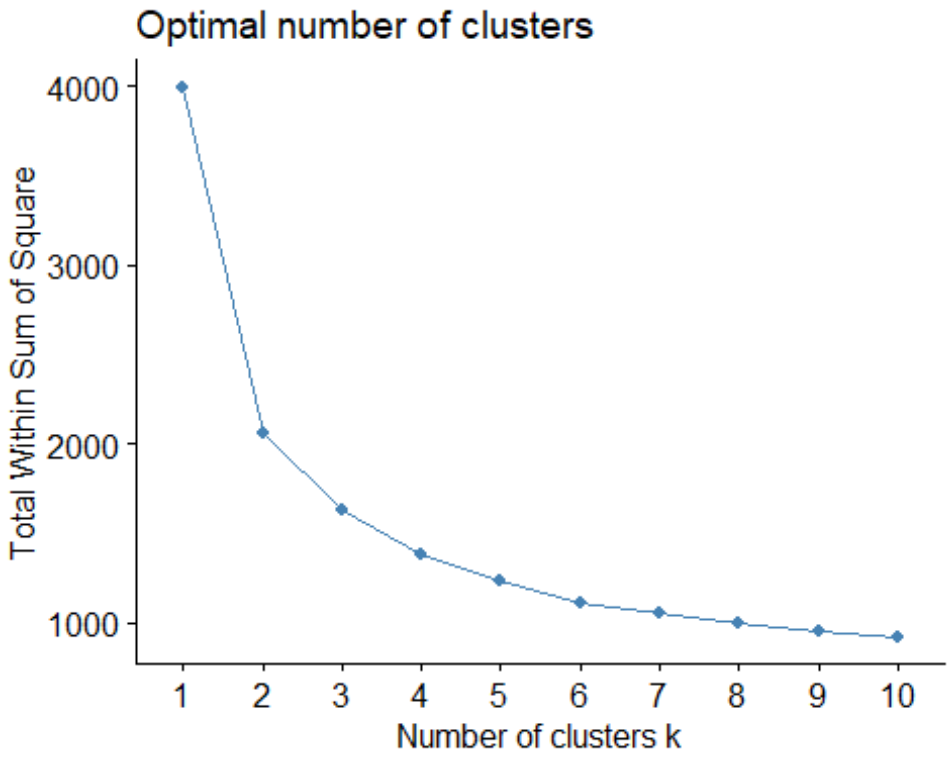
```
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5     0.1      1
##  maxlen target   ext
##      10  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 50
##
## set item appearances ...[1 item(s)] done [0.00s].
## set transactions ...[544 item(s), 501 transaction(s)] done [0.00s].
## sorting and recoding items ... [26 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [36 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
fourth.rules.sorted <- sort(fourth.rules, by="lift", decreasing=TRUE)
fourth.rules.sorted2 <- sort(fourth.rules, by="support", decreasing=TRUE)
fourth.rules.sorted3 <- sort(fourth.rules, by="confidence", decreasing=TRUE)
```

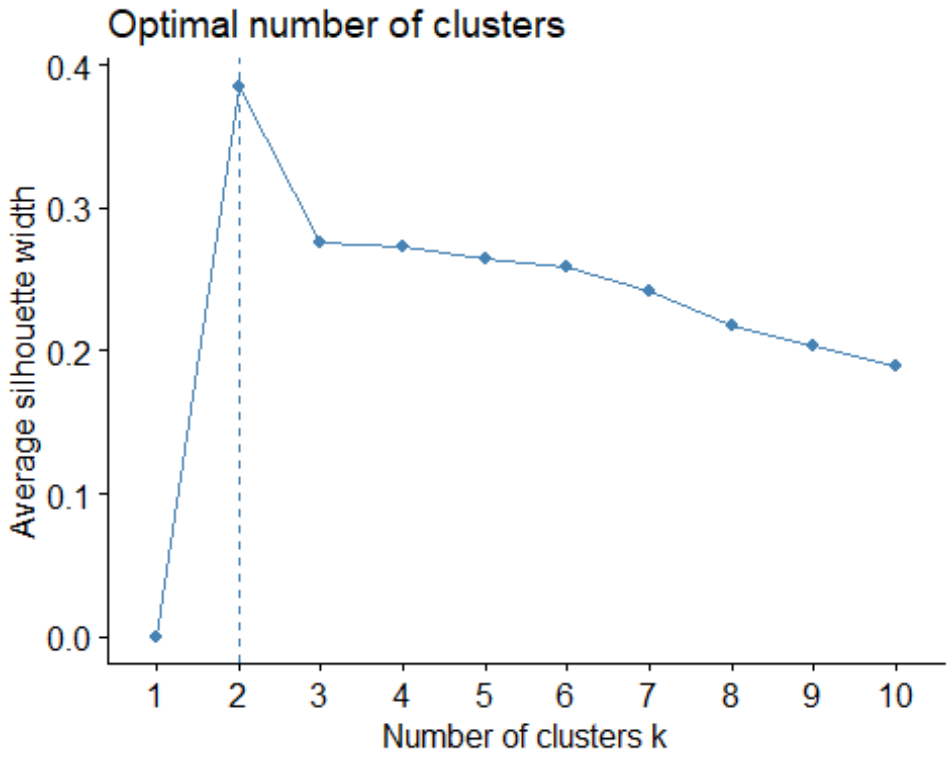
Clustering

Clustering analysis was done to see if there are any unique groupings of students that cna be made from the data set. Unique groupings could lead to discoveries about if certain variables are linked together in interesting ways. The first grouping method tried was k-means. First, the ideal number of k was estimated visually with two different methods.

```
fviz_nbclust(graduate_scale, kmeans, method = 'wss')
```

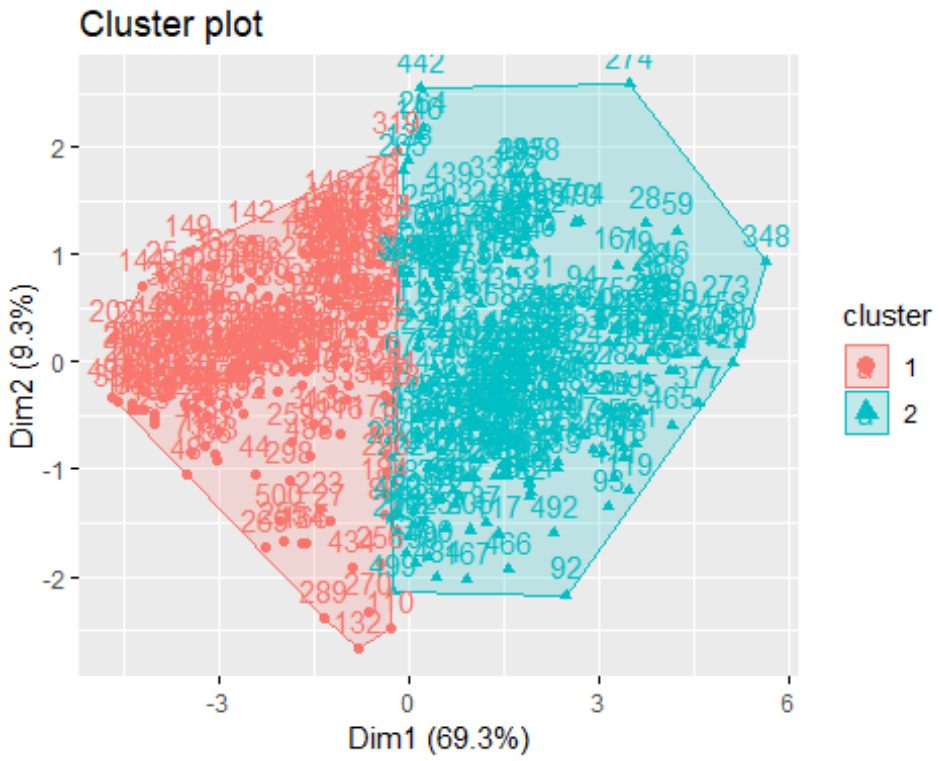



```
fviz_nbclust(graduate_scale, kmeans, method = 'silhouette')
```

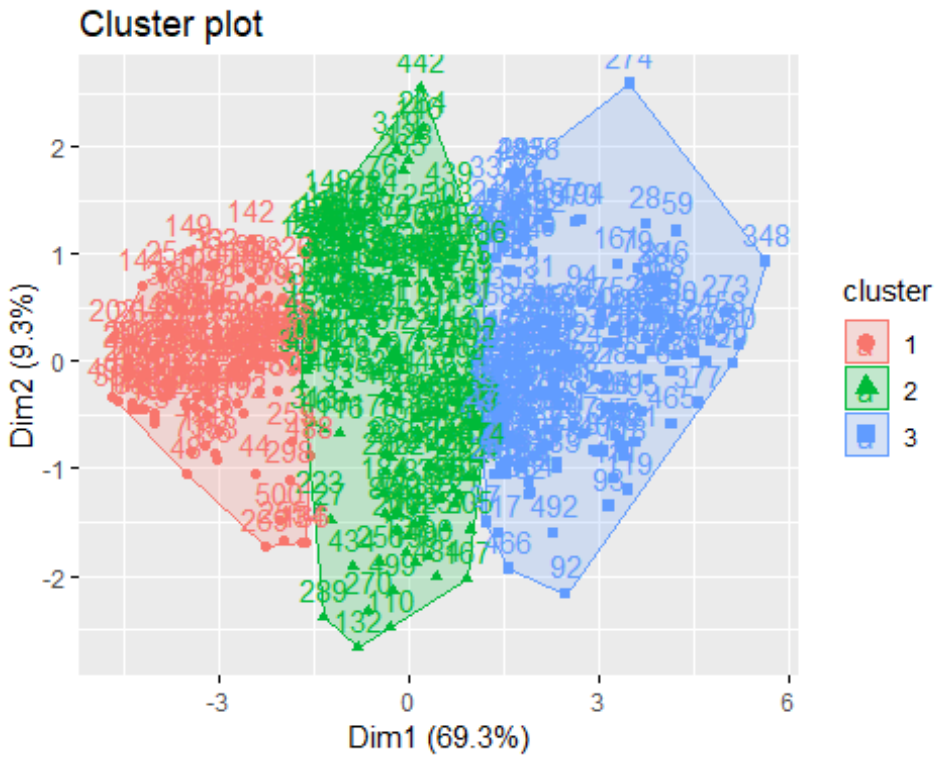


According to the graphs, the ideal number of clusters is 2. Clusters of 3 and 4 were also created for comparative purposes.

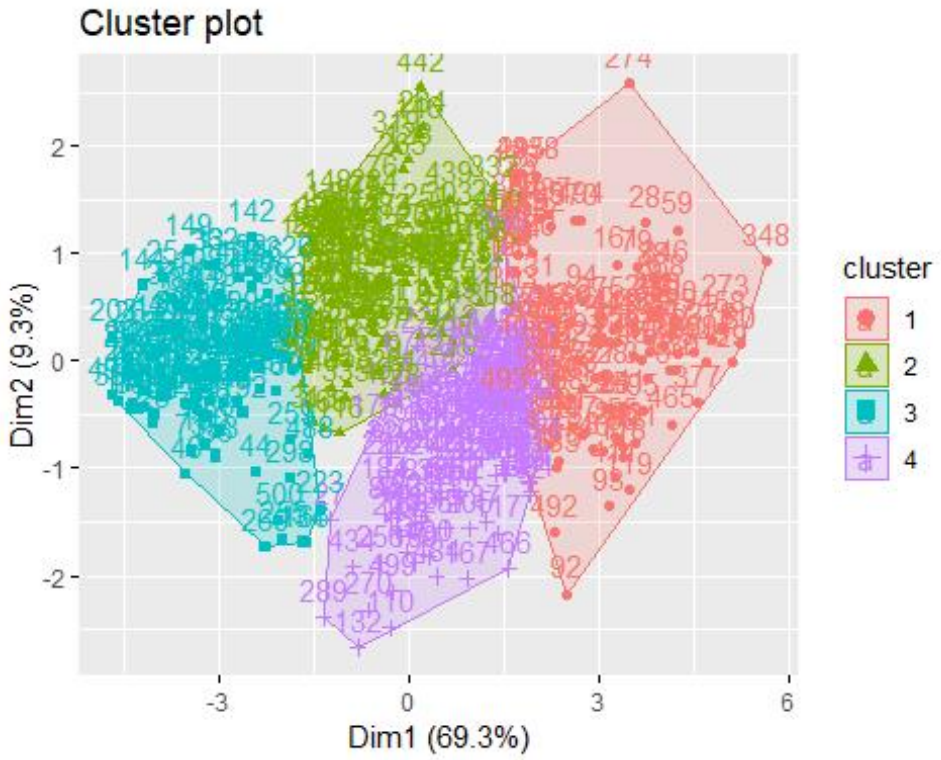
```
k2 <- kmeans(graduate_scale, centers=2, nstart= 25)
fviz_cluster(k2, data = graduate_scale)
```




```
k3 <- kmeans(graduate_scale, centers=3, nstart= 25)
fviz_cluster(k3, data = graduate_scale)
```

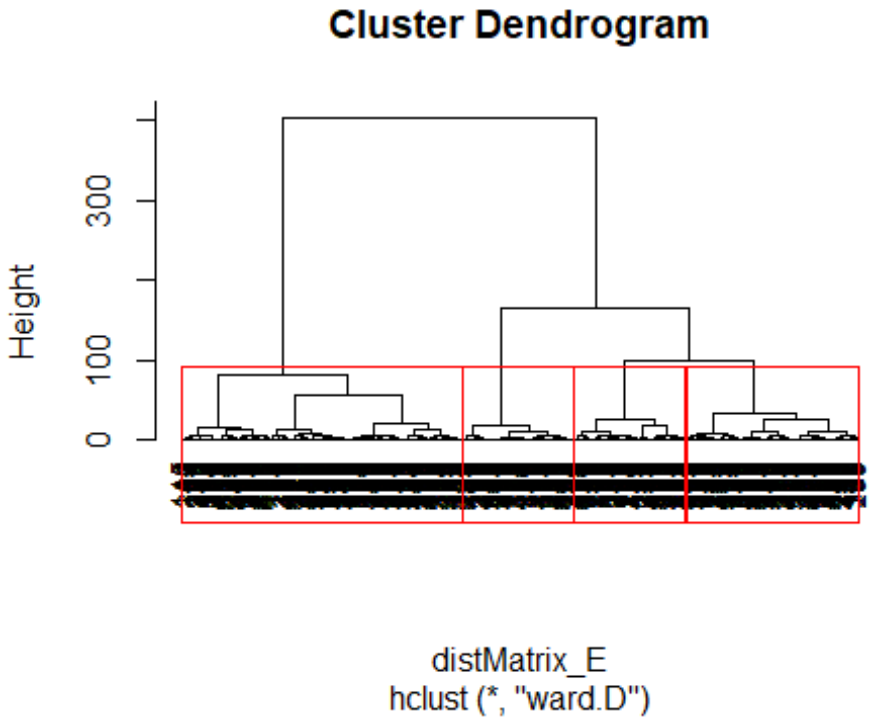



```
k4 <- kmeans(graduate_scale, centers=4, nstart= 25)
fviz_cluster(k4, data = graduate_scale)
```

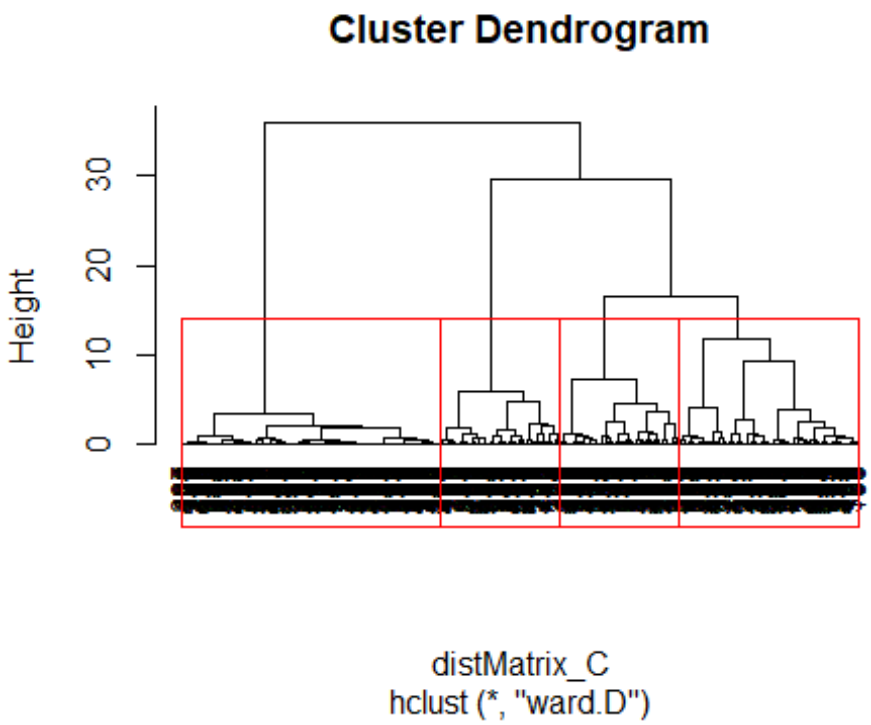


Hierarchical clustering was also attempted to see if this produced meaningful results as well. This clustering was done with two distance methods, Euclidian and cosine similarity.

```
m  <- graduate_matrix
distMatrix_E <- dist(m, method="euclidean")
distMatrix_C <- dist(m, method="cosine")

# Euclidean
groups_E <- hclust(distMatrix_E,method="ward.D")
plot(groups_E, cex=0.9, hang=-1)
rect.hclust(groups_E, k=4)
```



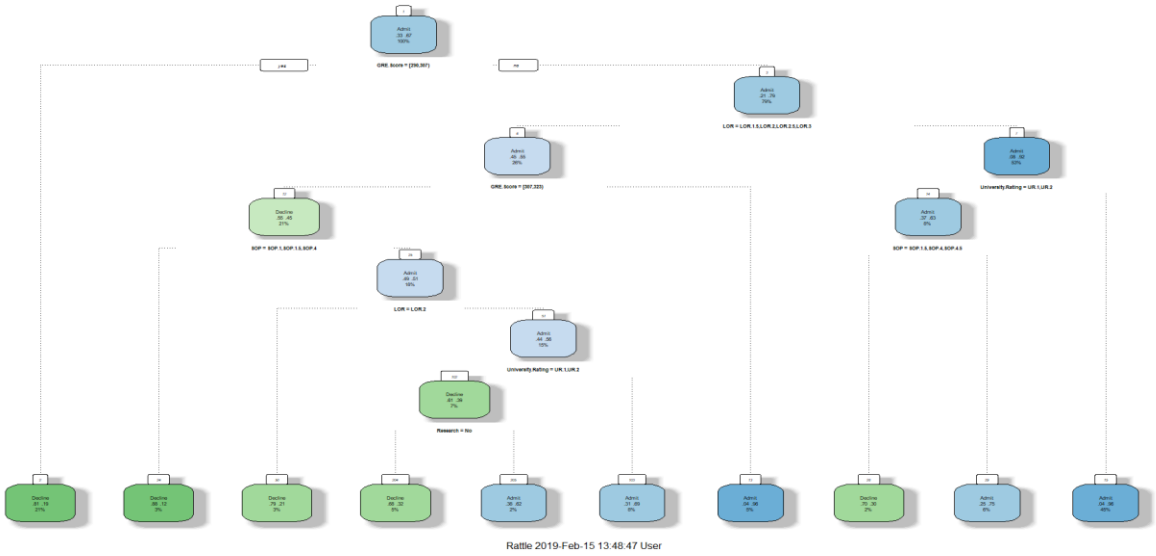

```
# Cosine Similarity
groups_C <- hclust(distMatrix_C,method="ward.D")
plot(groups_C, cex=0.9, hang=-1)
rect.hclust(groups_C, k=4)
```



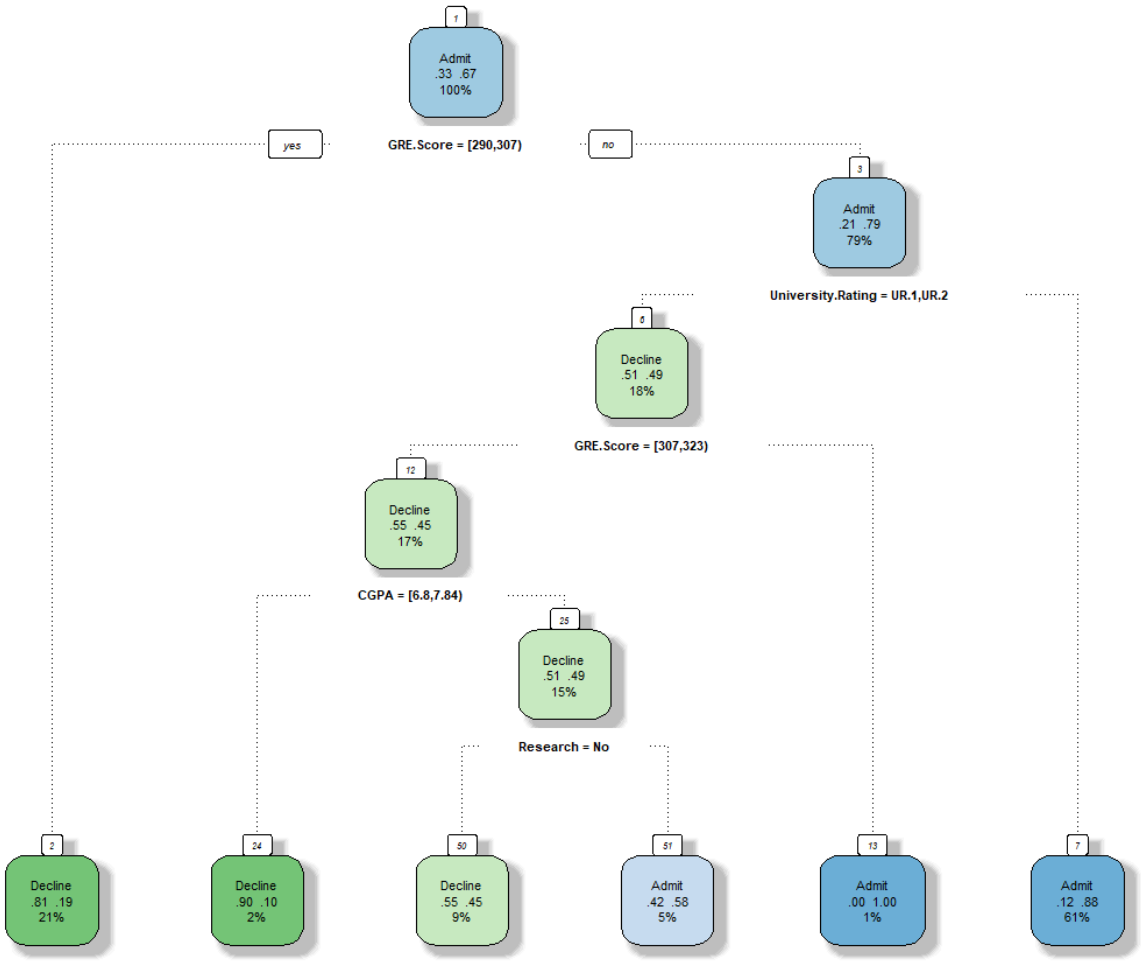
Decision Trees

Decision tree analysis was done to see which variables were chosen to be important based on this algorithm. Would certain variables be excluded naturally when attempting to divide the data in thsi manner? The first decision tree included all variables. The second included the variables that had been shown to be important based on the associative rules mining. A third tree was made using only the variables that occured in the first decision and in the strongest associating rules.

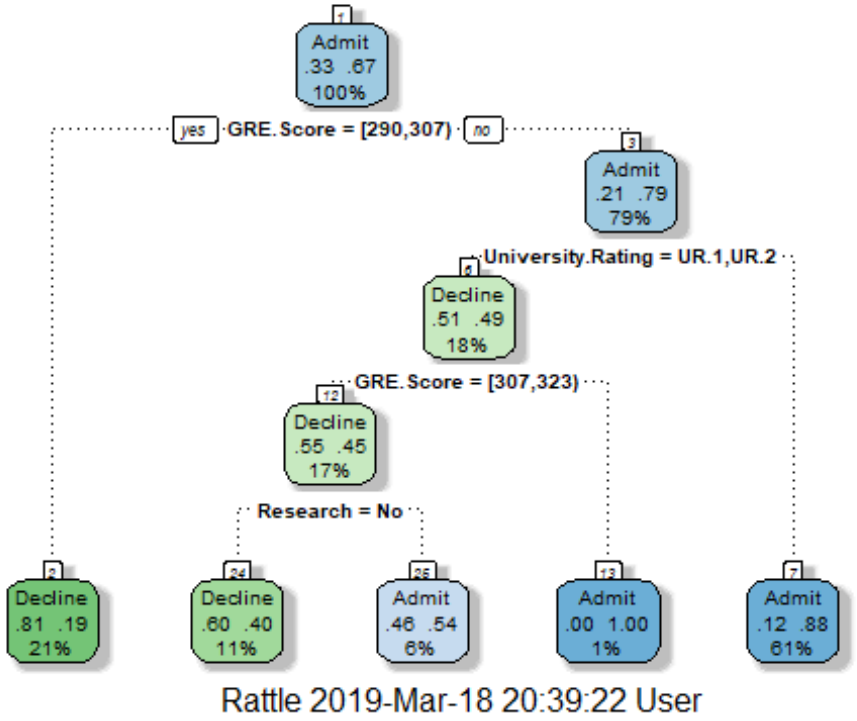
```
fit <- rpart(graduate_ar$Chance.of.Admit ~ ., data = graduate_ar, method="cla
ss")
fancyRpartPlot(fit)
```




```
fit2 <- rpart(graduate_ar$Chance.of.Admit ~ GRE.Score + TOEFL.Score + Univers
ity.Rating + CGPA + Research, data = graduate_ar, method="class")
fancyRpartPlot(fit2)
```




```
fit3 <- rpart(graduate_ar$Chance.of.Admit ~ GRE.Score + University.Rating + R
esearch, data = graduate_ar, method="class")
fancyRpartPlot(fit3)
```



Naive Bayes

The Naive Bayes algorithm was implemented to see if the data could be sucessfully predicted as Accepted or Declined. This is an important feature to be able to produce as knowing if an application falls into either the Accept or Decline category helps to determine the strength of an application. The algorithm was run three times. The first was with every variable included in the formula. The second included only the important variables based on the associative rules, and the third included the variables that were important to the decision tree. Each method was verified by using the k-folds method, with 5 folds done in each run.

```
N <- nrow(graduate2)
# Number of desired splits
kfolds <- 5
# Generate indices of holdout observations
holdout <- split(sample(1:N), 1:kfolds)

# All variables
AllResults<-list()
AllLabels<-list()
for (k in 1:kfolds){

  Kdigit_Test=graduate2[holdout[[k]], ]
  Kdigit_Train=graduate2[-holdout[[k]], ]
```

```
  ## Make sure you take the labels out of the testing data
  (head(Kdigit_Test))
  Kdigit_Test_noLabel<-Kdigit_Test[-c(8)]
  Kdigit_Test_justLabel<-Kdigit_Test$Chance.of.Admit


  # naivebayes
  ## formula is label ~ x1 + x2 + .  NOTE that label ~. is "use all to create
model"
  NB<-naive_bayes(Chance.of.Admit~., data=Kdigit_Train, laplace=0, na.action
= na.pass)
  NB_Pred <- predict(NB, Kdigit_Test_noLabel)


  ## Accumulate results from each fold
  AllResults<- c(AllResults,NB_Pred)
  AllLabels<- c(AllLabels, Kdigit_Test_justLabel)

}
table(unlist(AllResults),unlist(AllLabels))

##
##       1   2
##   1 142  62
##   2  25 271

# AR variables
AllResults2<-list()
AllLabels2<-list()
for (k in 1:kfolds){

  Kdigit_Test=graduate2[holdout[[k]], ]
  Kdigit_Train=graduate2[-holdout[[k]], ]


  ## Make sure you take the labels out of the testing data
  (head(Kdigit_Test))
  Kdigit_Test_noLabel<-Kdigit_Test[-c(8)]
  Kdigit_Test_noLabel<-Kdigit_Test_noLabel[-c(5)]
  Kdigit_Test_noLabel<-Kdigit_Test_noLabel[-c(4)]
  Kdigit_Test_justLabel<-Kdigit_Test$Chance.of.Admit


  # naivebayes
  ## formula is label ~ x1 + x2 + .  NOTE that label ~. is "use all to create
model"
  NB2<-naive_bayes(Chance.of.Admit ~ GRE.Score + TOEFL.Score + University.Rat
```

```
ing + CGPA + Research, data=Kdigit_Train, laplace=0, na.action = na.pass)
  NB_Pred2 <- predict(NB2, Kdigit_Test_noLabel)


  ## Accumulate results from each fold
  AllResults2<- c(AllResults2,NB_Pred2)
  AllLabels2<- c(AllLabels2, Kdigit_Test_justLabel)

}


# Decision Tree variables
AllResults3<-list()
AllLabels3<-list()
for (k in 1:kfolds){

  Kdigit_Test=graduate2[holdout[[k]], ]
  Kdigit_Train=graduate2[-holdout[[k]], ]


  ## Make sure you take the labels out of the testing data
  (head(Kdigit_Test))
  Kdigit_Test_noLabel<-Kdigit_Test[-c(8)]
  Kdigit_Test_noLabel<-Kdigit_Test_noLabel[-c(6)]
  Kdigit_Test_noLabel<-Kdigit_Test_noLabel[-c(2)]
  Kdigit_Test_justLabel<-Kdigit_Test$Chance.of.Admit


  # naivebayes
  ## formula is label ~ x1 + x2 + .  NOTE that label ~. is "use all to create
model"
  NB3<-naive_bayes(Chance.of.Admit ~ GRE.Score + University.Rating + SOP + LO
R + Research, data=Kdigit_Train, laplace=0, na.action = na.pass)
  NB_Pred3 <- predict(NB3, Kdigit_Test_noLabel)


  ## Accumulate results from each fold
  AllResults3<- c(AllResults3,NB_Pred3)
  AllLabels3<- c(AllLabels3, Kdigit_Test_justLabel)

}
table(unlist(AllResults),unlist(AllLabels))

##
##       1   2
##   1 142  62
##   2  25 271

table(unlist(AllResults2),unlist(AllLabels2))
```

```
##
##       1   2
##   1 147  68
##   2  20 265
```

```
table(unlist(AllResults3),unlist(AllLabels3))
```

```
##
##       1   2
##   1 137  63
##   2  30 270
```

The results of each method are summarized in the tables above. 1 indicated Decline, where 2 indicates Admit.

Support Vector Machine

Much like the Naive Bayes algorithm, the support vector machines algorithm was done to find the most accurate prediction process possible for the data set. Three separate kernels were tested (polynomial, linear, and radial) with three different costs for a missort (1, 10, and 100), giving 9 algorithms run in total. The entirety of the variables were used in the formula for prediction for the sake of simplicity. Given the size of the data set, this is not an issue.

```
svm_graduate_fit_p <- svm(Chance.of.Admit~., data=graduate2, kernel="polynomi
al", cost=1, scale=FALSE)
pred_graduate_p <- predict(svm_graduate_fit_p, graduate2[-8], type="class")
table(pred_graduate_p, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p Decline Admit
##         Decline     136    42
##         Admit        31   291
```

```
svm_graduate_fit_p2 <- svm(Chance.of.Admit~., data=graduate2, kernel="polynom
ial", cost=10, scale=FALSE)
pred_graduate_p2 <- predict(svm_graduate_fit_p2, graduate2[-8], type="class")
table(pred_graduate_p2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p2 Decline Admit
##          Decline     135    38
##          Admit        32   295
```

```
svm_graduate_fit_p3 <- svm(Chance.of.Admit~., data=graduate2, kernel="polynom
ial", cost=100, scale=FALSE)
pred_graduate_p3 <- predict(svm_graduate_fit_p3, graduate2[-8], type="class")
table(pred_graduate_p3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p3 Decline Admit
##          Decline     136    47
##          Admit        31   286
```

```
svm_graduate_fit_l <- svm(Chance.of.Admit~., data=graduate2, kernel="linear",
cost=1, scale=FALSE)
pred_graduate_l <- predict(svm_graduate_fit_l, graduate2[-8], type="class")
table(pred_graduate_l, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l Decline Admit
##         Decline     138    36
##         Admit        29   297
```

```
svm_graduate_fit_l2 <- svm(Chance.of.Admit~., data=graduate2, kernel="linear"
, cost=10, scale=FALSE)
pred_graduate_l2 <- predict(svm_graduate_fit_l2, graduate2[-8], type="class")
table(pred_graduate_l2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l2 Decline Admit
##          Decline     133    32
##          Admit        34   301
```

```
svm_graduate_fit_l3 <- svm(Chance.of.Admit~., data=graduate2, kernel="linear"
, cost=100, scale=FALSE)
pred_graduate_l3 <- predict(svm_graduate_fit_l3, graduate2[-8], type="class")
table(pred_graduate_l3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l3 Decline Admit
##          Decline     131    37
##          Admit        36   296
```

```
svm_graduate_fit_r <- svm(Chance.of.Admit~., data=graduate2, kernel="radial",
cost=1, scale=FALSE)
pred_graduate_r <- predict(svm_graduate_fit_r, graduate2[-8], type="class")
table(pred_graduate_r, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r Decline Admit
##         Decline     147    22
##         Admit        20   311
```

```
svm_graduate_fit_r2 <- svm(Chance.of.Admit~., data=graduate2, kernel="radial"
, cost=10, scale=FALSE)
pred_graduate_r2 <- predict(svm_graduate_fit_r2, graduate2[-8], type="class")
table(pred_graduate_r2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r2 Decline Admit
```

```
##          Decline     163     4
##          Admit         4   329
```

```
svm_graduate_fit_r3 <- svm(Chance.of.Admit~., data=graduate2, kernel="radial"
, cost=100, scale=FALSE)
pred_graduate_r3 <- predict(svm_graduate_fit_r3, graduate2[-8], type="class")
table(pred_graduate_r3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r3 Decline Admit
##          Decline     167     1
##          Admit         0   332
```

Text Mining

For this data set, text mining would not be an appropriate tool to use. As this data set contains no text values, there is no reasonable way to implement this. Were it possibe, text mining could be used very effectively on the Statement of Purpose for each application. The common words and tokens could be discovered for each of the quality levels for the SOP to find out which phrases should be used to make a higher quality SOP or even which words and phrases to avoid to not make a lower quality SOP. This is outside the scope of this dataset, however, and unfortunately cannot be carried out or included.

Results

Associative Rules Mining

When looking at the top 10 rules when sorted by lift, support, and confidence, the same types of variables appear repeatedly in each set.

```
inspect(fourth.rules.sorted[1:10])
```

```
##      lhs                                rhs     support   confidence lift
## [1]  {[111,120],UR.5}                => {Admit} 0.1097804 1          1.504505
## [2]  {[323,340],UR.5}                => {Admit} 0.1037924 1          1.504505
## [3]  {[8.88,9.92],UR.5}              => {Admit} 0.1237525 1          1.504505
## [4]  {UR.5,Yes}                      => {Admit} 0.1277445 1          1.504505
## [5]  {[111,120],Yes}                 => {Admit} 0.2634731 1          1.504505
## [6]  {[111,120],[8.88,9.92],UR.5}    => {Admit} 0.1077844 1          1.504505
## [7]  {[111,120],UR.5,Yes}            => {Admit} 0.1057884 1          1.504505
## [8]  {[323,340],UR.5,Yes}            => {Admit} 0.1017964 1          1.504505
## [9]  {[8.88,9.92],UR.5,Yes}          => {Admit} 0.1197605 1          1.504505
## [10] {[111,120],UR.4,Yes}            => {Admit} 0.1057884 1          1.504505
##      count
## [1]   55
## [2]   52
## [3]   62
## [4]   64
```

```
## [5]  132
## [6]   54
## [7]   53
## [8]   51
## [9]   60
## [10]  53
```

```
inspect(fourth.rules.sorted2[1:10])
```

```
##      lhs                              rhs     support   confidence lift
## [1]  {[8.88,9.92]}                 => {Admit} 0.3193613 0.9815951  1.476814
## [2]  {[323,340]}                   => {Admit} 0.2914172 0.9733333  1.464384
## [3]  {[111,120]}                   => {Admit} 0.2894212 0.9731544  1.464115
## [4]  {[8.88,9.92],Yes}             => {Admit} 0.2854291 0.9930556  1.494057
## [5]  {[323,340],Yes}               => {Admit} 0.2734531 0.9927536  1.493602
## [6]  {[111,120],Yes}               => {Admit} 0.2634731 1.0000000  1.504505
## [7]  {[111,120],[8.88,9.92]}       => {Admit} 0.2375250 0.9834711  1.479637
## [8]  {[111,120],[323,340]}         => {Admit} 0.2355289 0.9752066  1.467203
## [9]  {[323,340],[8.88,9.92]}       => {Admit} 0.2355289 0.9833333  1.479429
## [10] {[111,120],[8.88,9.92],Yes} => {Admit} 0.2235529 1.0000000  1.504505
##      count
## [1]  160
## [2]  146
## [3]  145
## [4]  143
## [5]  137
## [6]  132
## [7]  119
## [8]  118
## [9]  118
## [10] 112
```

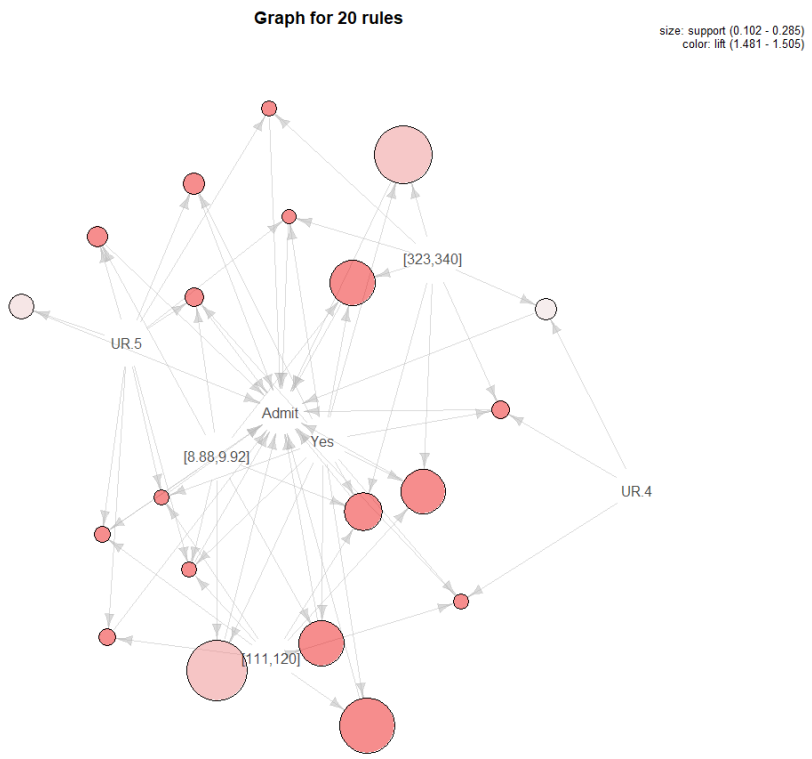
```
inspect(fourth.rules.sorted3[1:10])
```

```
##      lhs                                rhs     support   confidence lift
## [1]  {[111,120],UR.5}                => {Admit} 0.1097804 1          1.504505
## [2]  {[323,340],UR.5}                => {Admit} 0.1037924 1          1.504505
## [3]  {[8.88,9.92],UR.5}              => {Admit} 0.1237525 1          1.504505
## [4]  {UR.5,Yes}                      => {Admit} 0.1277445 1          1.504505
## [5]  {[111,120],Yes}                 => {Admit} 0.2634731 1          1.504505
## [6]  {[111,120],[8.88,9.92],UR.5} => {Admit} 0.1077844 1          1.504505
## [7]  {[111,120],UR.5,Yes}            => {Admit} 0.1057884 1          1.504505
## [8]  {[323,340],UR.5,Yes}            => {Admit} 0.1017964 1          1.504505
## [9]  {[8.88,9.92],UR.5,Yes}          => {Admit} 0.1197605 1          1.504505
## [10] {[111,120],UR.4,Yes}            => {Admit} 0.1057884 1          1.504505
##      count
## [1]   55
## [2]   52
## [3]   62
## [4]   64
## [5]  132
```

```
## [6]   54
## [7]   53
## [8]   51
## [9]   60
## [10]  53
```

When visualizing the top 20 rules when sorted by lift, it is easy to see which variables are repeated most frequently.

```
fourth.plot <- fourth.rules.sorted[1:20]
plot(fourth.plot, method="graph")
```



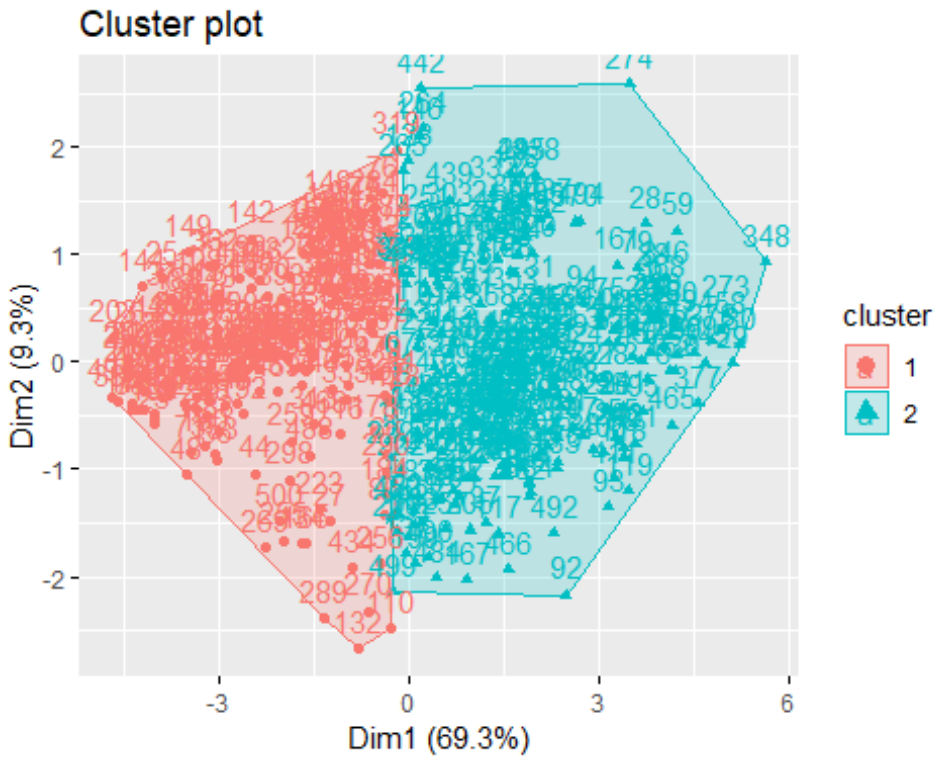
A high University Ranking, the highest GRE Score, highest TOEFL Score, highest CGPA, and Research experience are the features that are found across each of the 20 rules. This means that the SOP and LOR ranking is not important for the strongest associative rules.

Clustering

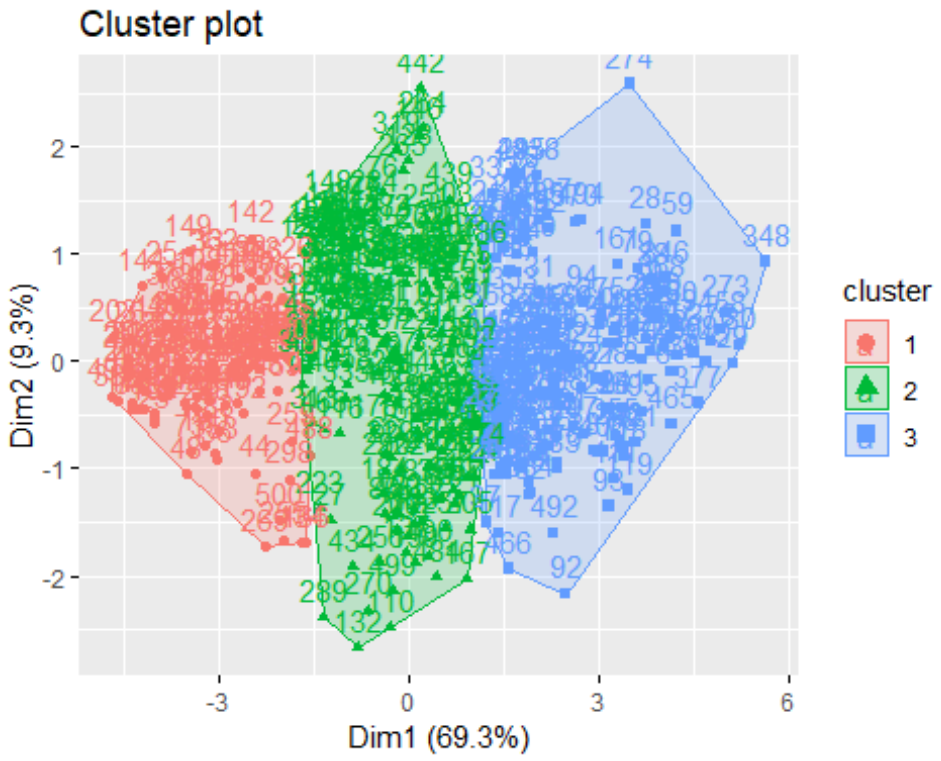
Clustering proved not to generate any insightful information. Each of the clusters barely keep from overlapping each other in the ideal case of 2 clusters and they only get less well defined in increasing numbers of clusters. Additionally, when the 2 clusters are inspected, it seems to have just grouped the students together by the highest variables in one cluster and the lower ones in the other. The hierarchical clustering produced much the same uninsightful results. The clusters are only somewhat well defined in either distance

measurement. It is interesting to note that the clearest clusters are made in 4 groups as opposed to 2 in the k means method.

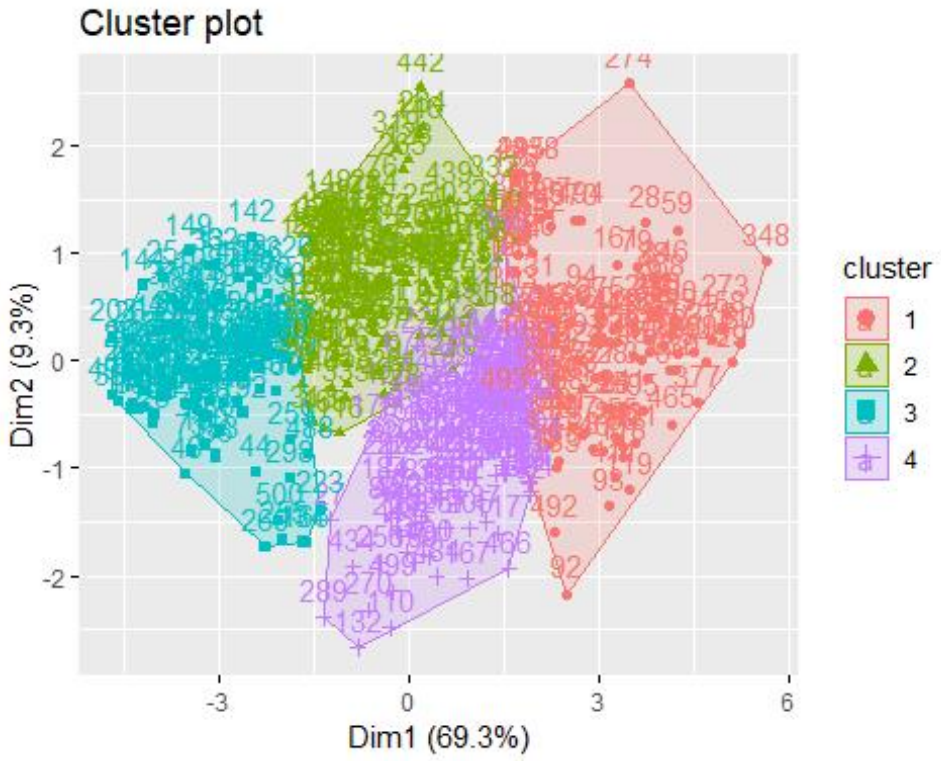
```
# k means results
fviz_cluster(k2, data = graduate_scale)
```




```
fviz_cluster(k3, data = graduate_scale)
```



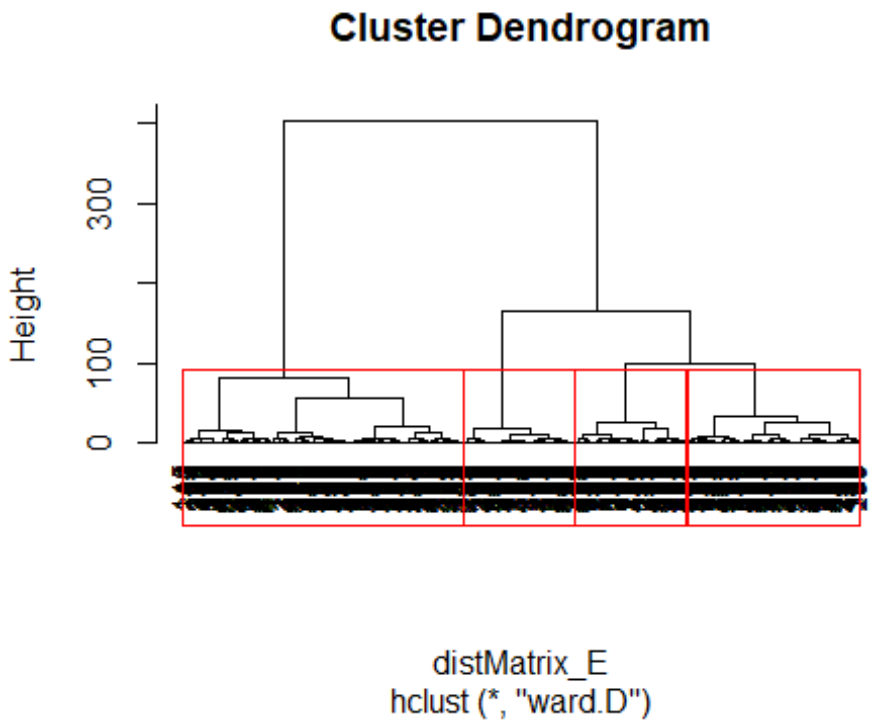

```
fviz_cluster(k4, data = graduate_scale)
```



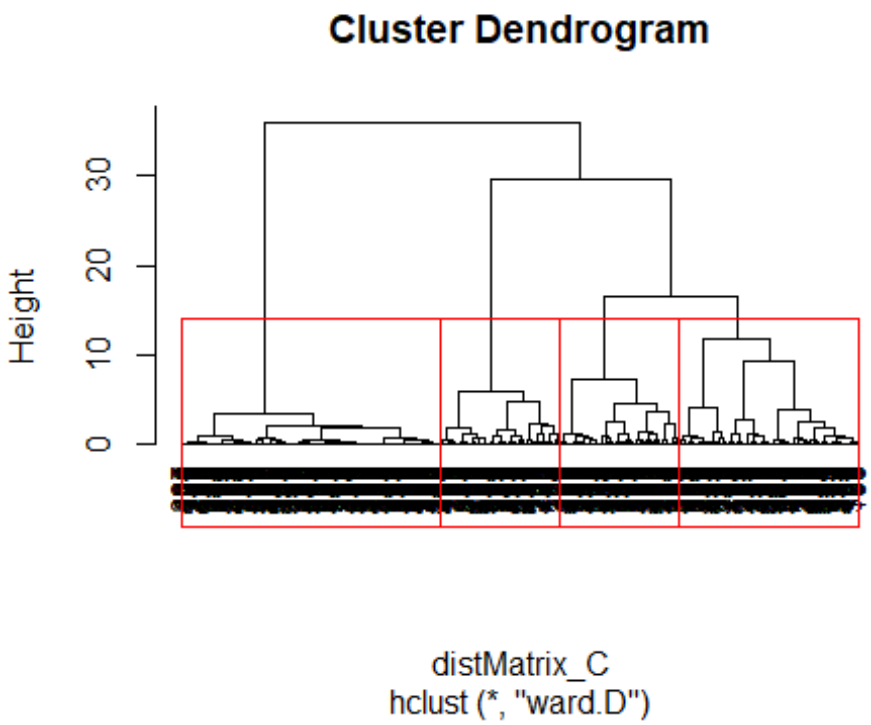

```
print(k2)
```

```
## K-means clustering with 2 clusters of sizes 224, 276
##
## Cluster means:
##    GRE.Score TOEFL.Score University.Rating        SOP        LOR
## 1  0.8218099   0.7773341         0.7748062  0.7488080  0.6661048
## 2 -0.6669761  -0.6308798        -0.6288282 -0.6077282 -0.5406068
##         CGPA   Research Chance.of.Admit
## 1  0.8418148  0.6788735       0.8404195
## 2 -0.6832120 -0.5509698      -0.6820796
##
## Clustering vector:
##   [1] 1 1 2 1 2 1 1 2 2 2 2 1 1 2 2 2 2 2 2 2 2 1 1 1 1 2 2 2 2 1 1 1
1
##  [36] 1 2 2 2 2 2 2 2 1 1 1 1 1 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 2 2 1
1
##  [71] 1 1 1 1 2 1 1 2 2 2 2 1 1 1 1 1 2 2 2 1 2 2 2 2 2 2 2 1 1 1 1 2 2 2
1
## [106] 1 1 1 1 1 2 1 2 2 2 1 2 2 2 1 1 1 2 2 2 2 1 2 1 1 1 1 2 1 1 1 2 2 1
2
## [141] 1 1 1 1 1 2 2 1 1 2 1 1 1 2 1 2 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2 1 1 1
1
## [176] 1 1 1 2 2 2 2 2 1 2 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 1 1 2 2 2 2 2
2
## [211] 1 1 1 1 1 1 1 1 1 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 1 1 1 1 2 2 2 2 1 1
2
## [246] 1 2 2 1 1 2 2 2 1 1 1 2 1 1 1 1 2 2 2 2 2 2 2 1 1 2 2 2 2 2 1 1 2 2
2
## [281] 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 1 1 2 2 2 2 1 2 1 1 1 2 2 1 1 1 2
2
## [316] 2 2 2 1 1 1 1 2 2 2 1 2 2 1 2 1 2 2 2 2 1 1 2 1 1 1 2 1 2 2 2 2 2 2 2
2
## [351] 2 1 2 2 2 2 1 2 2 2 1 1 1 2 1 1 1 2 2 2 2 1 1 1 2 2 2 2 2 2 1 2 1 2
1
## [386] 1 2 2 2 2 1 2 2 1 2 1 1 1 1 2 1 2 2 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 2
2
## [421] 2 1 1 1 1 1 2 1 2 1 2 1 1 1 2 2 2 2 2 2 2 2 1 1 1 1 1 2 2 1 1 1 2
2
## [456] 2 2 2 2 2 1 1 2 2 2 2 2 1 1 1 1 2 1 2 2 2 2 2 1 1 1 1 1 2 2 2 2 1 1
2
## [491] 2 2 2 2 2 2 1 1 1 2 1
##
## Within cluster sum of squares by cluster:
## [1]  783.3647 1276.9682
##  (between_SS / total_SS =  48.4 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

```r
# Hierarchical results
plot(groups_E, cex=0.9, hang=-1)
rect.hclust(groups_E, k=4)
```



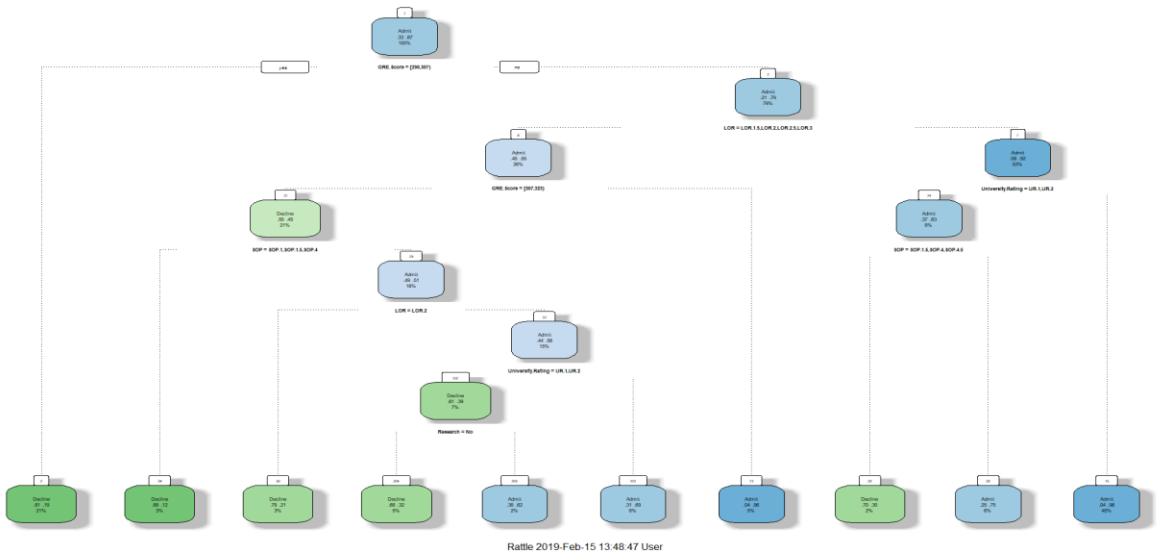

```r
plot(groups_C, cex=0.9, hang=-1)
rect.hclust(groups_C, k=4)
```



Decision Trees

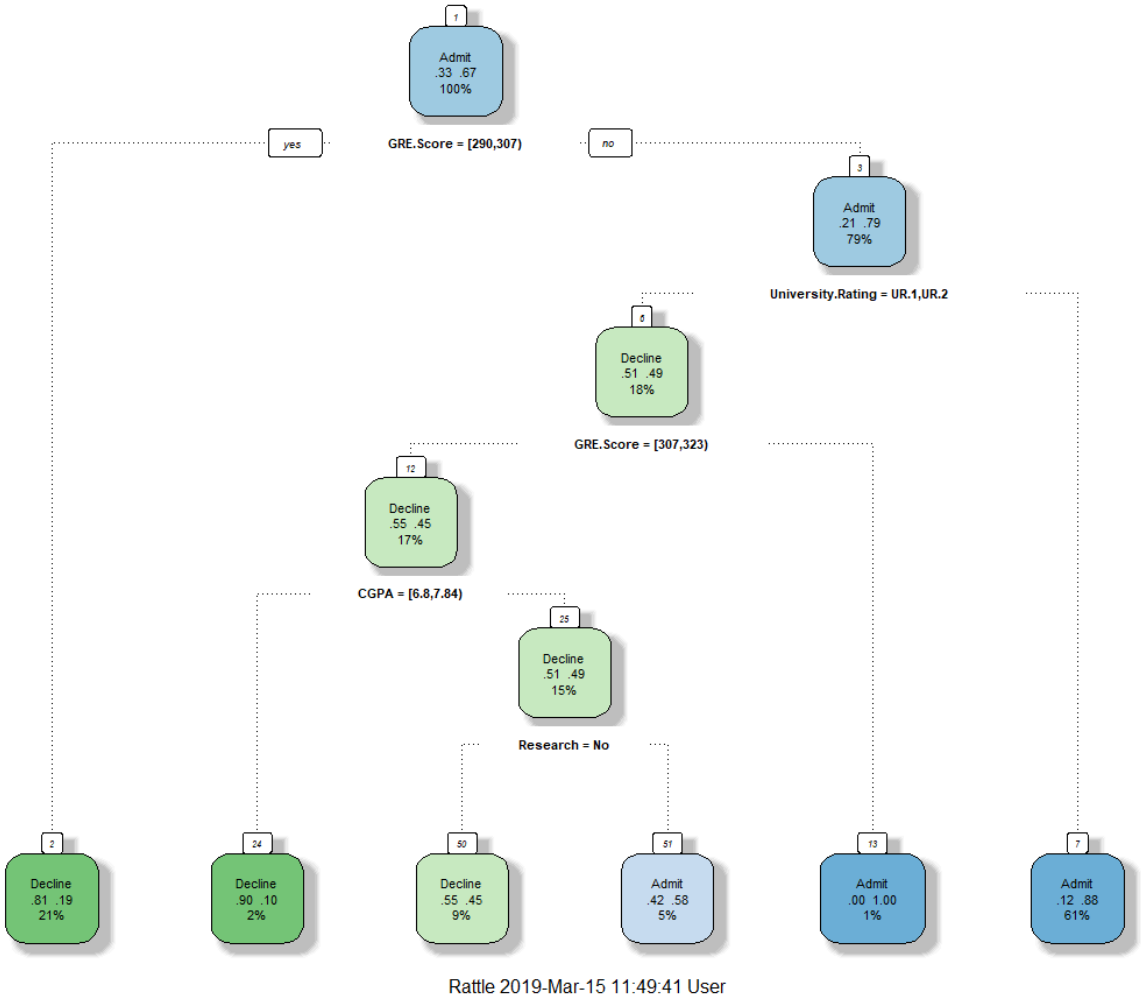
When giving the decision tree all of the variable to work with, it only used five of the variables to split up the data: GRE Score, LOR, SOP, University Rating, and Research.

```
fancyRpartPlot(fit)
```



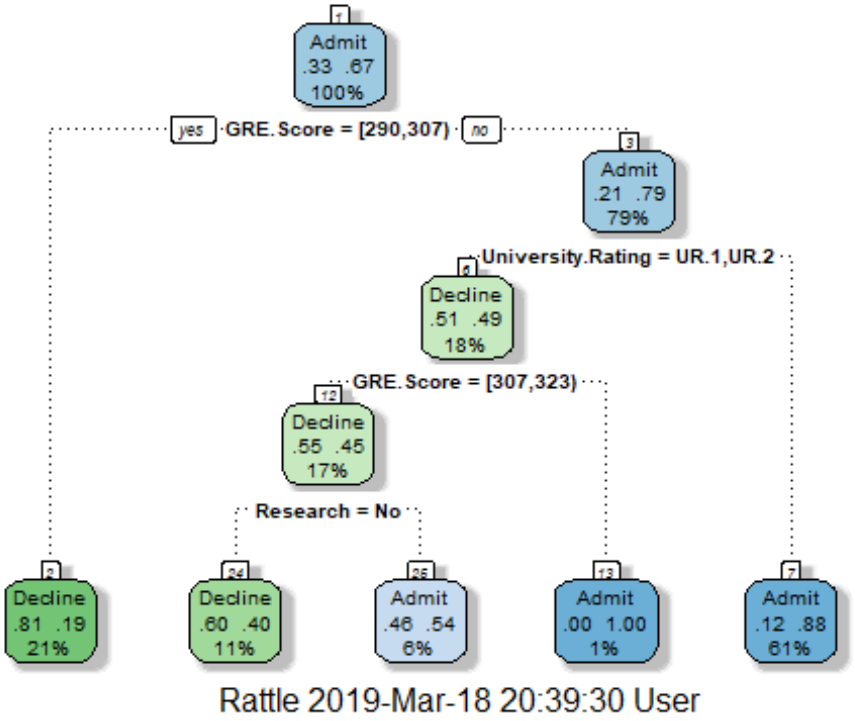
When following the tree, this shows that even with a high GRE score, a lower University Rating and SOP can lead to a Decline in application. A lower quality LOR can be made up for by a high GRE Score. This first tree is a bit difficult to read, though, and so a pruned tree was made using the variables discovered in the associative rule mining.

```
fancyRpartPlot(fit2)
```



This tree follows some of the similar patterns as the above tree. For example, the GRE score is still at the top and the University Rating follows to the right hand side. These two variables still remain most important for determining the difference between Admit and Decline. While this tree does not account for as many of the possible circumstances, it is still much easier to read. A third pruned tree made from the overlap of variables in associative rules and the first tree was also produced.

```
fancyRpartPlot(fit3)
```



Intuitively, this tree is the simplest of three and the easiest to read. As with the other two trees, GRE Score and University rating appear at the top of the tree. This tree uses each of the variables it is given to build the output, unlike the other two trees.

It is most likely that the associative rules and decision tree found different variables to be important due to how each method works. Associative rules is looking at the counts of events together, whereas the decision tree is looking for the most efficient way to split the data to the desired outcome. Just because a variable or combination of variables occurs frequnetly does not mean it will necessarily be good at dividing the data into distinct groups.

Naive Bayes

The results for each of the Naive Bayes models built is summarized again below. Additionally, a table summarizing the overall accuracy, precision, and recall for each model is shown below.

```
table(unlist(AllResults),unlist(AllLabels))

##
##       1   2
##   1 142  62
##   2  25 271

table(unlist(AllResults2),unlist(AllLabels2))
```

```
##
##       1   2
##   1 147  68
##   2  20 265
```

```
table(unlist(AllResults3),unlist(AllLabels3))
```

```
##
##       1   2
##   1 137  63
##   2  30 270
```

Parameter	Accuracy	Precision Decline	Recall Decline	Precision Admit	Recall Admit
All Variables	82.8%	85.6%	69.8%	81.4%	91.9%
AR Variables	82.8%	88.6%	68.8%	79.9%	93.3%
DT Variables	81.4%	81.4%	68.7%	81.4%	89.7%

While the Naive Bayes algorithm has good success at predicting an Accept decision, it does poorly with Decline, achieving only about 70% success in its Recall when all the variables were used for sorting.

Support Vector Machine

With the mild success of the Naive Bayes, support vector machine prediction was attempted. Each of the 9 algorithms is again summarized in tabular format and in a summary table showing the accuracy, precision and recall for each.

```
table(pred_graduate_p, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p Decline Admit
##         Decline     136    42
##         Admit        31   291
```

```
table(pred_graduate_p2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p2 Decline Admit
```

```
##          Decline     135    38
##          Admit        32   295
```

```
table(pred_graduate_p3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_p3 Decline Admit
##          Decline     136    47
##          Admit        31   286
```

```
table(pred_graduate_l, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l Decline Admit
##         Decline     138    36
##         Admit        29   297
```

```
table(pred_graduate_l2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l2 Decline Admit
##          Decline     133    32
##          Admit        34   301
```

```
table(pred_graduate_l3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_l3 Decline Admit
##          Decline     131    37
##          Admit        36   296
```

```
table(pred_graduate_r, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r Decline Admit
##         Decline     147    22
##         Admit        20   311
```

```
table(pred_graduate_r2, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r2 Decline Admit
##          Decline     163     4
##          Admit         4   329
```

```
table(pred_graduate_r3, graduate2$Chance.of.Admit)
```

```
##
## pred_graduate_r3 Decline Admit
##          Decline     167     1
##          Admit         0   332
```

Parameter	Accuracy	Precision Decline	Recall Decline	Precision Admit	Recall Admit
Polynomial, Cost = 1	85.4%	81.4%	76.4%	87.4%	90.4%
Polynomial, Cost = 10	86.0%	80.8%	79.8%	88.6%	90.2%
Polynomial, Cost = 100	84.4%	81.4%	74.3%	85.9%	90.2%
Linear, Cost = 1	87.0%	82.6%	79.3%	89.2%	91.1%
Linear, Cost = 10	86.8%	79.6%	80.6%	90.7%	89.9%
Linear, Cost = 100	85.4%	78.4%	78.0%	88.9%	89.2%
Radial, Cost = 1	91.6%	88.0%	87.0%	93.4%	94.0%
Radial, Cost = 10	98.4%	97.6%	97.6%	98.8%	98.8%
Radial, Cost = 100	99.8%	100.0%	99.4%	99.7%	100.0%

The support vector machine methods all performed better than the Naive Bayes algorithm. The radial kernel was the most successful, being able to predict the data set almost perfectly with a high enough cost. Even at the low cost of 1, the algorithm performed exceptionally well.

Conclusions

Based on the data ming techniques performed for this data set, the most important variables for determining the success of a graduate school application are GRE Score, Univeristy Rating, and Research experience. It is interesting that a variable that represents standardized testing, undergraduate experience, and practical skills application are all represented in the three reoccuring important variables. The most successful method for predicting the outcome of the data was the support vector machine with the radial kernel. It could be that as the data set grows and becomes more varied, the kernel that produces the best results changes.

These conclusions are all based on data that comes from an Indian perspective. This means that these important values might not have universal application. Not only does this

data set focus on student from India, but they are specifically students looking to apply to universities in the United States since they all have TOEFL scores. These conclusions are still valuable, but they are most useful to a specific subset of potential graduate students.

Moving forward, it would be interesting to obtain a testing data set. By having a set where the results are unknown, the true strength of the algorithms could be tested. Additionally, if data could be gathered from students from other countries, a comparative analysis could be performed. It is possible that the variables that are important to the Indian students is not as important to students from the United States, for example. By increasing the size and the robustness of the data set, the ability to measure the strength of an individual's graduate school application would increase and become that much more useful a tool to prospective students.

Citation

Mohan S Acharya, Asfia Armaan, Aneeta S Antony : A Comparison of Regression Models for Prediction of Graduate Admissions, IEEE International Conference on Computational Intelligence in Data Science 2019