# Machine Learning Analysis of College Admissions

Allison Yih and Emi Kong

June 04, 2020

## Introduction

Many high school students stress out about college admissions, often wondering what their chances are for getting into a certain university. To get a better idea of what college admissions officers view as the most important factors of a successful applicant, we developed a supervised, binary classifier that predicts an individual's admission outcome for our project.

The applicant's weighted high school GPA, standardized testing scores, and family income are the factors taken into consideration. Through building this model, we hope to help college applicants figure out which aspects of their application should be focused on in order to maximize their likelihood of acceptance.

## Data Overview

The dataset that will be used for analyzing this college's admissions is AdmissionsData.rdata, which is from a data scientist, Ernest Tavares's, blog. In this dataset, there are 8700 observations that describe each applicant's academic scores, personal background, and admissions decision. The variable "admit" is the binary response variable, which takes on the value 0 and 1 to represent getting a rejection and acception, respectively.

Our dataset contained 2196 rows with 0 values under the columns "sati.verb" and "sati.math". This could be because those individuals took the ACT, which is the alternative exam for the SAT, and those scores were not reported. By omitting the applicants with missing data, this ensured that we were only looking at the data for applicants who took the SAT exam.

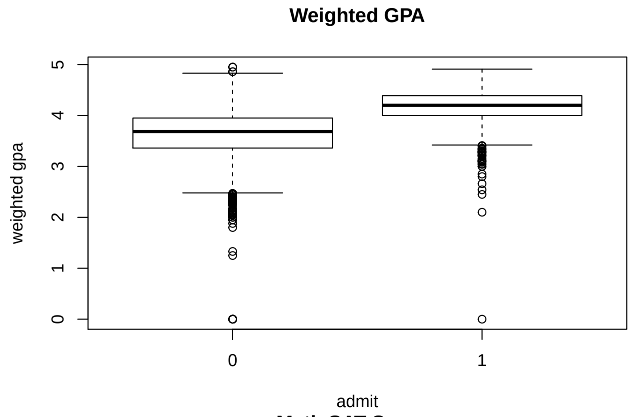
Our cleaned dataset has a dimension of 6182 rows x 5 columns. The columns include "admit" and 4 numeric predictor variables: "gpa.wtd", "sati.verb", "sati.math", and "income". The variables "anglo", "asian", and "black" were omitted in our analysis since there was an initial uneven racial distribution of all applicants, and "sex" was excluded because the amount of gender bias in college admissions differs between college types, according to an article by the Washington Post.

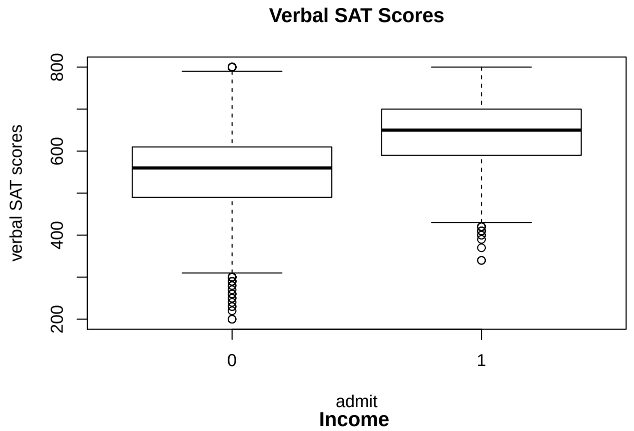
By removing missing values and these 4 predictor variables, this allowed for more consistency when comparing classification methods and test error rates, since some of the methods we used could not handle categorical predictors.

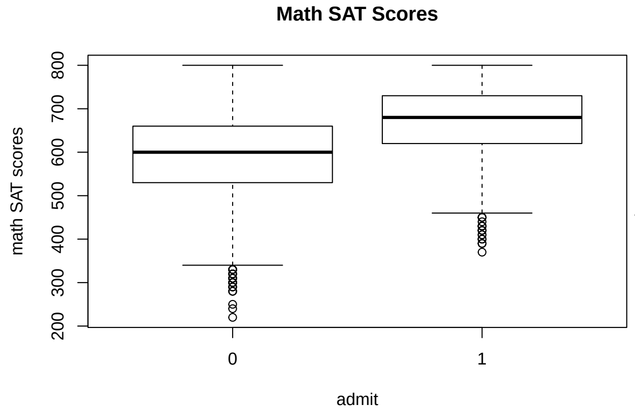
```
##  admit      gpa.wtd        sati.verb       sati.math         income
##  0:4256   Min.   :0.000   Min.   :200.0   Min.   :220.0   Min.   :  120
##  1:1926   1st Qu.:3.500   1st Qu.:510.0   1st Qu.:560.0   1st Qu.:32000
##            Median :3.860   Median :580.0   Median :630.0   Median :65000
##            Mean   :3.804   Mean   :578.1   Mean   :619.5   Mean   :62931
##            3rd Qu.:4.150   3rd Qu.:650.0   3rd Qu.:690.0   3rd Qu.:99999
##            Max.   :4.950   Max.   :800.0   Max.   :800.0   Max.   :99999
```

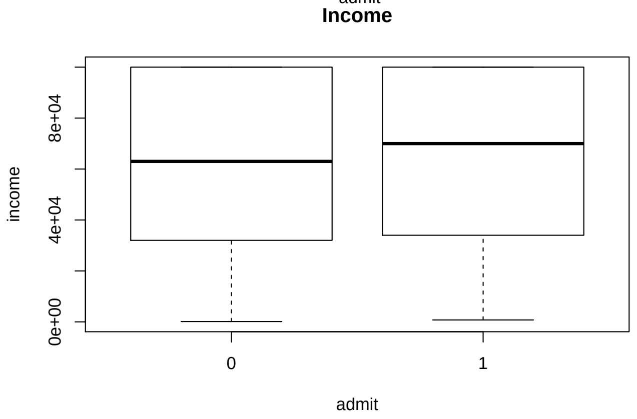
## Exploratory Data Analysis

We start by plotting box plots and histograms to investigate any significant relationships between the predictor variables and admission results.



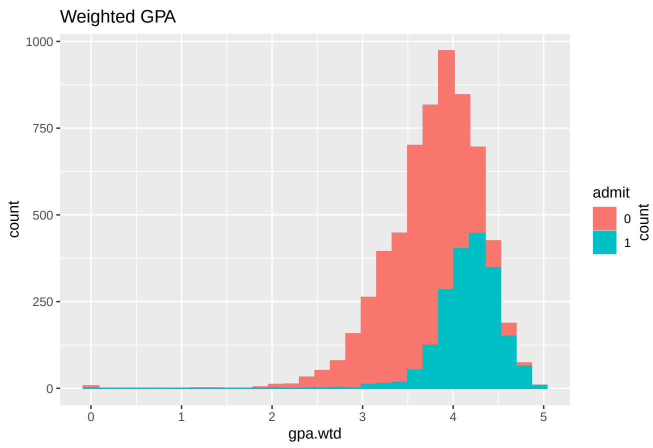


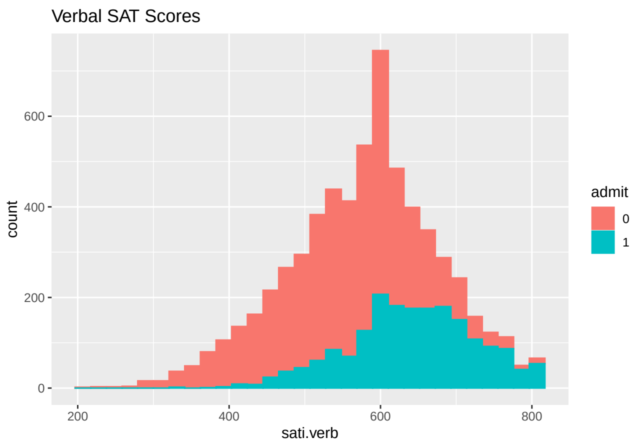


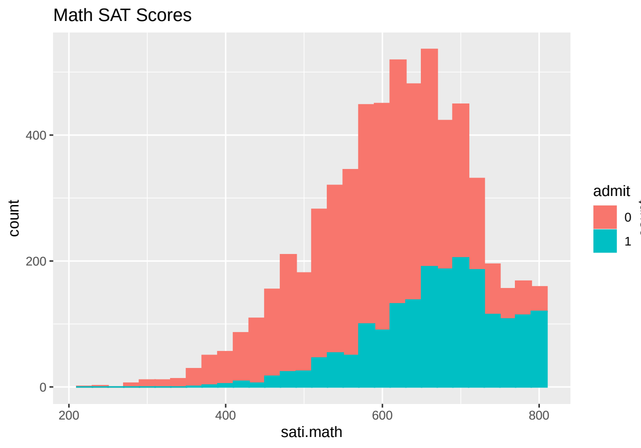


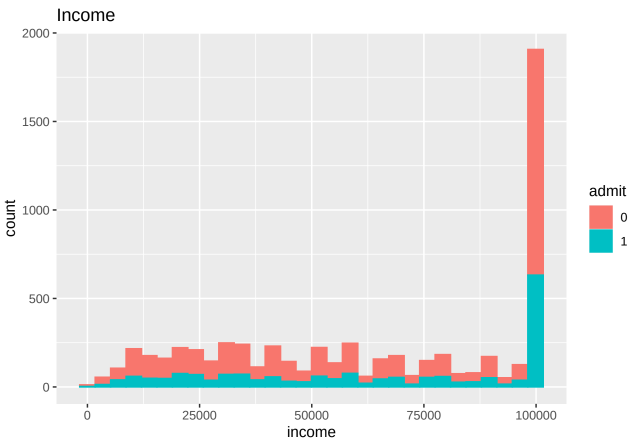
From these boxplots, we see that the people who are admitted have higher weighted GPAs, and higher verbal and math SAT scores. There is little difference in income amount for people who are admitted versus people who are not admitted.

Next, we plot histogram graphs to evaluate the distibution of each predictor for all applicants separated by color to signify if they were admitted or not.









For weighted GPA, an applicants chances of getting admitted are extremely high if their GPA is around 4.00 or higher. This is also seen when applicants have a verbal SAT score of at least 600 and/or a math SAT score around 625 or greater. Since a majority of the applicants have income around $100,000 it is expected that the majority of people admitted also have that same income amount.

## Methods

To find the model that would accurately classify whether an applicant gets admitted to this college, we will apply supervised machine learning methods on a training dataset.

We split the dataset into equally-sized training and testing sets. The first 3,091 observations would be used to identify and construct our model, and the remaining half would be used to determine whether our chosen classifier gave highly accurate predictions.

The algorithms that we will use to classify our data observations are K-nearest neighbors (KNN), logistic regression, decision trees, and random forest. Leave-one-out cross validation (LOOCV) will also be used to tune parameters for KNN in order to reduce bias and test error rate. For each algorithm, we will calculate a confusion matrix for predicted vs actual admission results, and compare their test error rates. By having our model learn and identify college acceptance patterns from previous data, we hope that it will help our model more accurately predict an outcome on new, similar data.

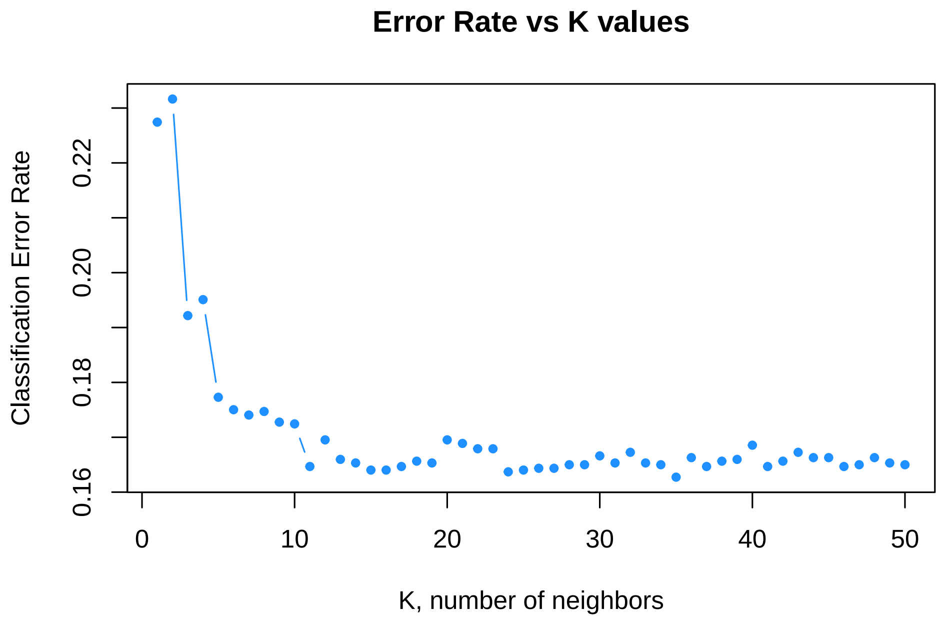
## K-Nearest Neighbors

We first use the K-nearest neighbors method to analyze the relationship between admissions and all numerical predictors. We created response vectors & design matrices to train the KNN classifier with 5, 10, and 15 neighbors initially. We chose these values because they are common K's that tend to optimize the bias-variance tradeoff. Using leave one out cross validation, we also found the best value of K was 35 neighbors and chose to train the KNN classifier with this value as well.

### Comparing KNN Test Error Rates

```
##         Test Error Rate
## K = 5         0.1879651
## K = 10        0.1708185
## K = 15        0.1695244
## K = 35        0.1633775
```

After training all KNN classifiers, we fit the test sets to each model and calculated the test error rates. The model based on 35 nearest neighbors resulted in the lowest misclassification error rate on the test set at 0.1633775. This makes sense since the purpose of LOOCV is to help us find the optimal number of neighbors, and since larger values of K produce smoother decision boundaries with lower variance. The visual representation below shows the error rates versus K values and our conclusion is further confirmed since K=35 is the lowest point on the graph.



## Logistic Regression

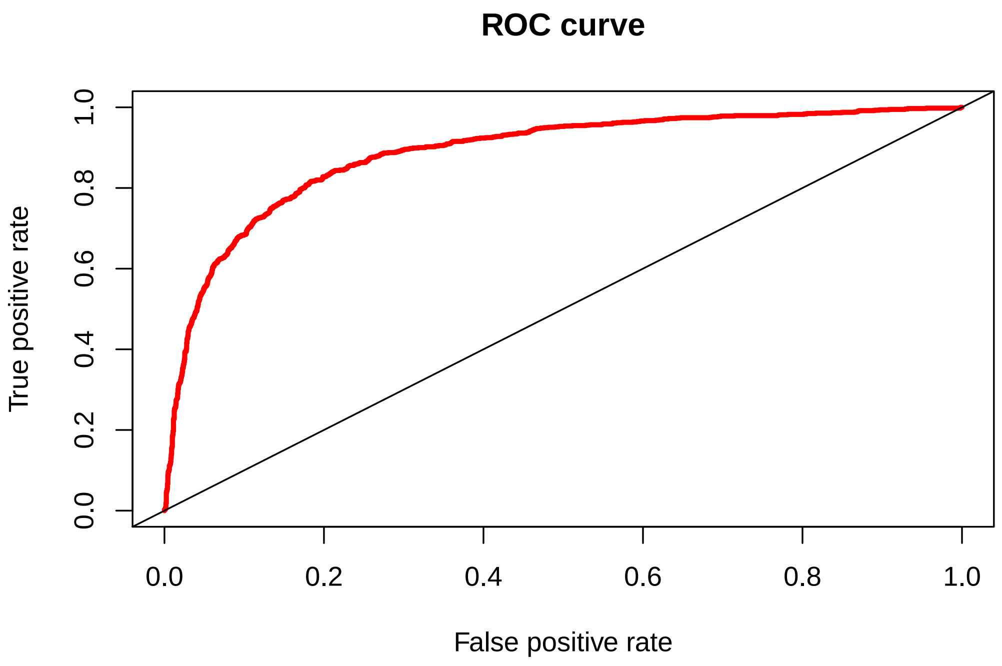
The second basic classifying method we used was logistic regression. Logistic regression is useful when determining variable importance by showing which predictors have the largest impact on our response (admission).

```
##
## Call:
## glm(formula = admit ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7951  -0.6140  -0.2558   0.5643   4.6959
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.066e+01  7.829e-01 -26.389  < 2e-16 ***
## gpa.wtd      3.340e+00  1.650e-01  20.247  < 2e-16 ***
## sati.verb    7.970e-03  6.903e-04  11.545  < 2e-16 ***
## sati.math    4.199e-03  6.792e-04   6.182 6.32e-10 ***
## income      -1.137e-05  1.669e-06  -6.813 9.53e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 3847.5  on 3090  degrees of freedom
## Residual deviance: 2451.1  on 3086  degrees of freedom
## AIC: 2461.1
##
## Number of Fisher Scoring iterations: 6
```

From the summary output, we see that all four of our predictors are significant to an applicant's admission based on their low p-values. For a one unit increase in each of these predictor values, the log-odds of getting admitted increases by the specific coefficient amount of that predictor when holding other predictors constant. For example, when there's a one unit increase in weighted GPA of an applicant, the log-odds of being admitted increases by 3.34.

## Checking How Accurate our Model is with ROC & AUC

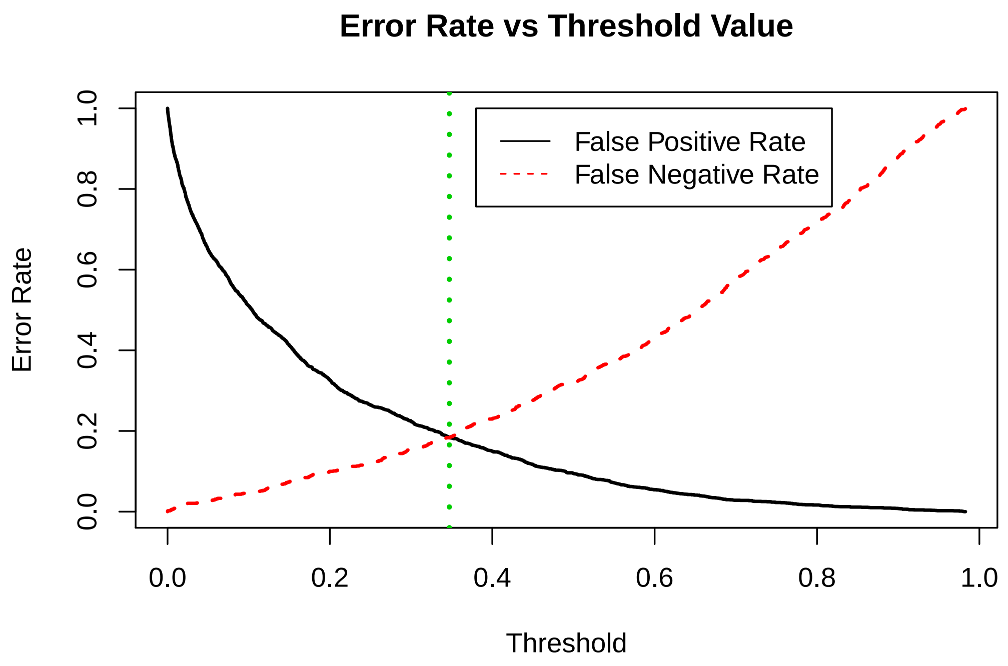
After fitting our logistic regression model on the training set, we constructed a Receiver Operating Characteristic (ROC) curve by plotting the true positive rate (TPR) against the false positive rate (FPR) of our soft classifier and calculated the area under this curve (AUC).



Since our curve (red line) follows close to the top-left corner of the graph and our AUC, being 0.8861, is close to 1, we conclude that logistic regression is an appropriate classifier for this dataset. If the curve was to closely follow the diagonal (FPR=TPR) and have an area around 0.5, this would indicate an inaccurate model.

## Using FPR & FNR to Find the Best Threshold Value

To calculate the test misclassification error rate for this model we need to change the soft classifiers (probability values) to hard classifiers (1 or 0) by setting a specific threshold value to assign the labels. We will use our false positive (FPR) and false negative (FNR) results to do this. While still using the training set logistic regression model, we calculated the euclidean distance between each point of (FPR,FNR) and (0, 0) and found thse smallest distance occurs when the threshold is 0.347139.

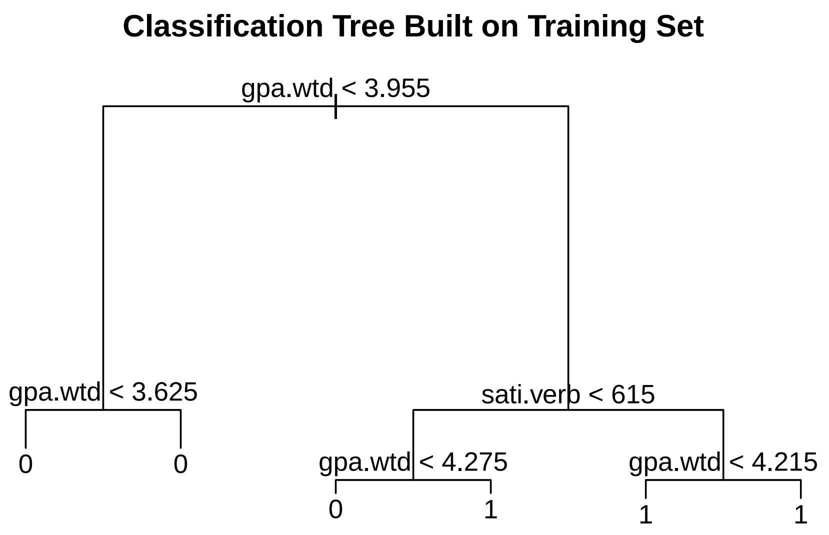


With this optimal threshold (green dashed line), we fit our testing set logistic regression model, construct a confusion matrix, and found our test misclassification error rate to be 0.1844063. This is greater than our test error rate for K-nearest neighbors when K = 35.

## Decision Trees

The decision tree algorithm was used because its output is simple to interpret. Each split represents the order in which the tree decides how to further split a set of features to eventually classify a qualitative response.

To implement the decision tree, we first fit our model with all 4 predictors on the training set.



However, based on the decision tree's plot, the data was split using only the gpa.wtd and sati.verb variables. Since decision trees involve looking at each variable's importance, this indicates that gpa.wtd, followed by sati.verb, were the most significant variables that would help our model gain the most information regarding how to make the best split at each node.

To calculate the test error rate for our classification tree, we predicted the response values for the test set and then constructed a confusion matrix.

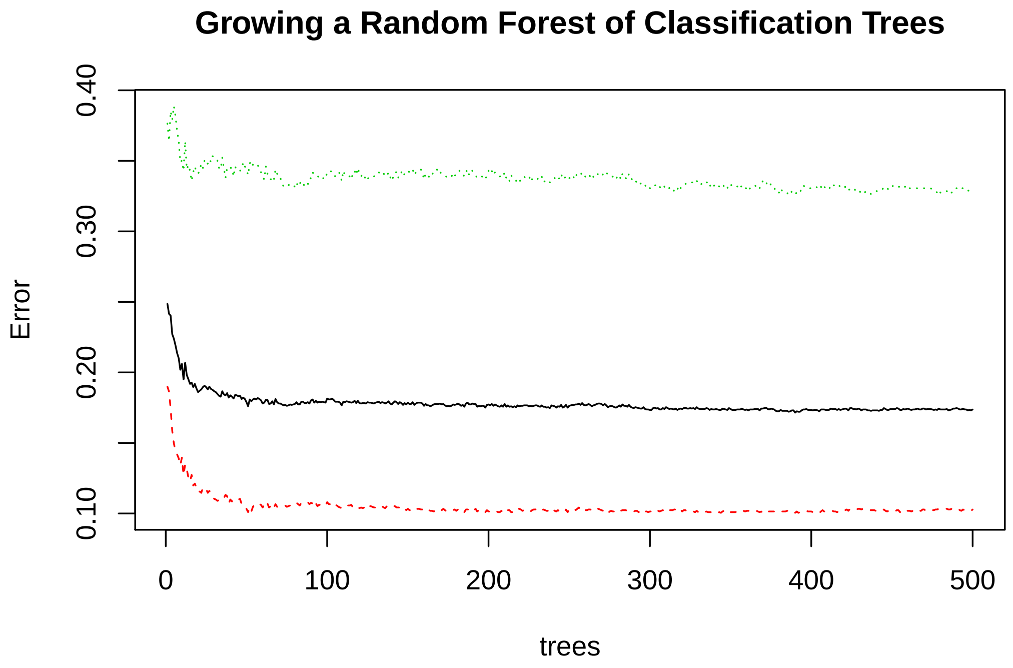
Finally, we subtracted the test accuracy rate, which is found through dividing the counter diagonal sum by the total sum, from 1 to get the test error rate.

Our decision tree's misclassification error rate is 0.1779. This means that our classification tree was able to correctly predict the admissions decision for 82.21% of the observations in the test set. When compared to KNN, our classification tree algorithm had a slightly higher test error rate, but had a smaller test error rate than logistic regression.

## Random Forest

Another method that was used for our classification was random forest. While containing multiple decision trees makes them harder to interpret, random forests prevent data overfitting and give highly accurate predictions, due to their robustness as an ensemble learning method.

To implement the classification tree version of a random forest, we fit the model and set the number of parameters as the square root of the total number of predictors.



Then, we constructed a confusion matrix and calculated the misclassification rate to be 0.1715.

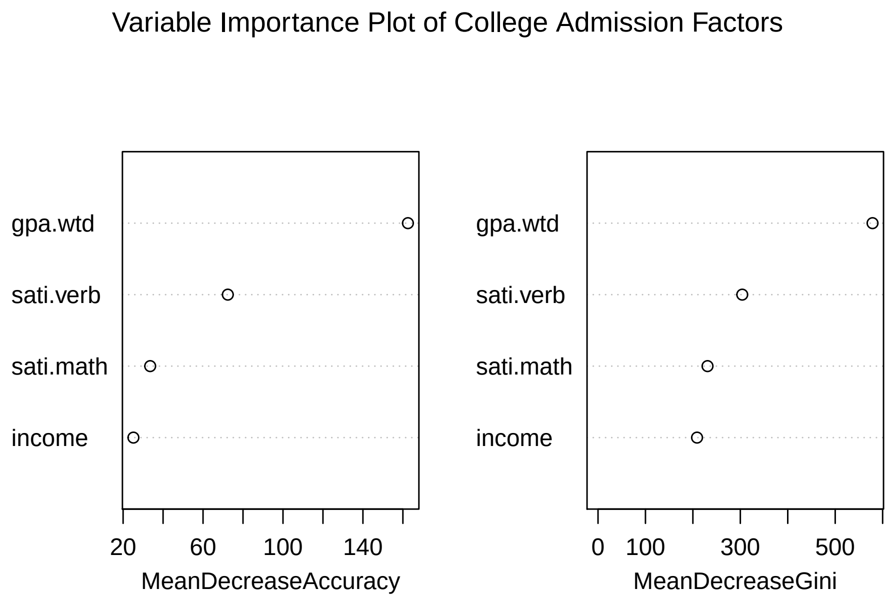
Another advantage of using random forests is that they provide information about not only the strength of the relationship between each predictor and response, but also how influential each predictor is on the classifications.

Each feature's importance is based off of their mean decrease in accuracy and Gini index. To view the degree of influence that each predictor has on the result, we use the importance() function, which displays these 2 factors in decreasing order.

```
##                   0         1 MeanDecreaseAccuracy MeanDecreaseGini
## gpa.wtd   79.438433 174.41098            162.51179         578.7278
## sati.verb 27.389149  79.60557             72.39928         304.0293
## sati.math  9.349955  36.67799             33.55685         230.8480
## income    16.192781  20.22250             25.16201         208.7158
```

From this output, it tells us that most important variable for getting admitted into the college was weighted GPA, as it had the highest decrease in mean accuracy and Gini index. Verbal SAT score was the 2nd-most important factor that determined the individual's admission decision.

Additionally, we plotted each variable's mean decreases in accuracy and Gini index.



From those graphs, it reemphasizes that the "gpa.wtd" variable had the most influence over the final admissions decision across all trees in the random forest the most, in terms of model accuracy and Gini index.

Overall, random forest had the 2nd-lowest misclassification rate, but did not perform as well as the KNN LOOCV method.

## Conclusion

When deciding our final model for this project, we chose to compare all the error rates (for our testing set) that each model produced. A model with a low test error rate signifies that the model classified very accurately and fit well to new data.

**Test Error Rates for All Models**

```
##                     Test Error Rate
## KNN (K = 35)              0.1633775
## Logistic Regression       0.1844063
## Decision Tree             0.1827887
## Random Forest             0.1724361
```

For our project, we conclude that K-nearest neighbors, with 35 neighbors, is our final model for predicting college admission results because it had the smallest test error rate.

### Study Limitations:

In addition to test scores and GPA, an applicant's accomplishments in extracurriculars, strength of essay responses and reccommendation letters are also important criteria in deciding one's admission decision. However, these are factors that a machine learning model cannot fully consider. For instance, it would be difficult to develop a model that could read in an applicant's essay and decide whether it is well-written, since that is a subjective opinion.

Additionally, we do not know the details about this type of school and population that our dataset describes. Different kinds of schools have varying acceptance rates. For example, liberal arts schools tend to accept the same amount of genders, while at engineering schools, women have a 10-15% greater chance of getting admitted, according to a Washington Post article. Since our data is based off of one college, the context about the admissions data is limited to that school's applicants, so we would not be able to generalize how much our model's predictors influence college admittance at all colleges.

### Potential Research:

In the future, we hope to consider how college admissions factors and acceptance affect the yield rate at a school. Many people apply to "safety" schools, where previously admitted students have lower academic scores than that applicant However, they usually do not end up attending those schools because they believe that their academic accomplishments make them overqualified. We would also be interested in utilizing unsupervised learning methods on datasets without a response variable to classify admission decisions.

## References

- Anderson, Nick. "The Gender Factor in College Admissions: Do Men or Women Have an Edge?" The Washington Post, WP Company, 26 Mar. 2014, www.washingtonpost.com/local/education/the-gender-factor-in-college-admissions/2014/03/26/4996e988-b4e6-11e3-8020-b2d790b3c9e1_story.html.
- Tavares, Ernest. "College Admissions Exploratory Analysis in R." Ernest Tavares III Full RSS, etav.github.io/projects/random_forest_college_admissions.html.

## Appendix

```r
library(dplyr)
library(class) #for knn classifiers
library(ROCR) #for log regression (ROC)
library(ggplot2)
library(tree)
library(randomForest)
library(gbm)

admissions_data <- Admissions
admissions_new <- na.omit(admissions_data)
admissions_new <- filter(admissions_new, sati.math !=0, sati.verb != 0)
#colSums(admissions_new)
admissions_new <- subset(admissions_new, select = c(admit, gpa.wtd, sati.verb, sati.math, income))
head(admissions_new)
summary(admissions_new)
dim(admissions_new)

#convert admit variable into factors
admissions_new$admit<-as.factor(admissions_new$admit)
str(admissions_new)

boxplot(gpa.wtd ~ admit, data = admissions_new, main = "Weighted GPA",
        ylab = "weighted gpa")
boxplot(sati.verb ~ admit, data = admissions_new, main = "Verbal SAT Scores",
        ylab = "verbal SAT scores")
boxplot(sati.math ~ admit, data = admissions_new, main = "Math SAT Scores",
        ylab = "math SAT scores")
boxplot(income ~ admit, data = admissions_new, main = "Income",
        ylab = "income")

ggplot(admissions_new, aes(x=gpa.wtd, fill = admit, color=admit)) + geom_histogram(bins=30) +
  ggtitle("Weighted GPA")
ggplot(admissions_new, aes(x=sati.verb, fill = admit, color=admit)) + geom_histogram(bins=30) +
  ggtitle("Verbal SAT Scores")
ggplot(admissions_new, aes(x=sati.math, fill = admit, color=admit)) + geom_histogram(bins=30) +
  ggtitle("Math SAT Scores")
ggplot(admissions_new, aes(x=income, fill = admit, color=admit)) + geom_histogram(bins=30) +
  ggtitle("Income")

set.seed(50)
#50% train, 50% test
cutoff <- sort(sample(nrow(admissions_new), nrow(admissions_new)*.5))
train <- admissions_new[cutoff,]
test <- admissions_new[-cutoff,]

#YTrain is the observed results for admit (0 = not admitted, 1 = admitted)
YTrain <- train$admit
#XTrain is the design matrix after removing admit variable and standardizing
XTrain <- train[ ,-which(names(train) %in% c("admit"))] %>%
  scale(center = TRUE, scale = TRUE)
```

```r
#mean and standard deviation of training set
meanvec <- attr(XTrain,'scaled:center')
sdvec <- attr(XTrain,'scaled:scale')

#X and Y for testing set
YTest <- test$admit
XTest <- test[ ,-which(names(train) %in% c("admit"))] %>%
  scale(center = meanvec, scale = sdvec)

#k=5
pred5 = knn(train=XTrain, test=XTest, cl=YTrain, k=5)
conf.matrix5 <- table(pred=pred5, true=YTest)
err5 = 1 - sum(diag(conf.matrix5)/sum(conf.matrix5))
#k=10
pred10 = knn(train=XTrain, test=XTest, cl=YTrain, k=10)
conf.matrix10 <- table(pred=pred10, true=YTest)
err10 = 1 - sum(diag(conf.matrix10)/sum(conf.matrix10))
#k=15
pred15 = knn(train=XTrain, test=XTest, cl=YTrain, k=15)
conf.matrix15 <- table(pred=pred15, true=YTest)
err15 = 1 - sum(diag(conf.matrix15)/sum(conf.matrix15))

#Using LOOCV to Find the Best Value of K:
set.seed(150)
#define an empty vector to save validation errors in future
validation.error = NULL
#give possible number of nearest neighbours to be considered
allK = 1:50
#for each number in allK, use LOOCV to find a validation error
for (i in allK){
  pred.Yval <- knn.cv(train=XTrain, cl=YTrain, k=i)
  validation.error <- c(validation.error, mean(pred.Yval!=YTrain))
}
#best number of neighbors
bestK <- max(allK[validation.error == min(validation.error)])
bestK

#make predictions on the test set with bestK
pred.YTest = knn(train=XTrain, test=XTest, cl=YTrain, k=bestK)
#confusion matrix
conf.matrix.knn <- table(pred=pred.YTest, true=YTest)
conf.matrix.knn
#test error rate
err_bestk = 1 - sum(diag(conf.matrix.knn)/sum(conf.matrix.knn))
err_bestk

knn_results <- matrix(c(err5, err10, err15, err_bestk), ncol=1, byrow=TRUE)
colnames(knn_results) <- c("Test Error Rate")
rownames(knn_results) <- c("K = 5","K = 10","K = 15", "K = 35")
knn_results.table <- as.table(knn_results)
knn_results.table

plot(validation.error, type = "b", col = "dodgerblue", cex = 1, pch = 20, xlab = "K, number of neighbor
#log reg fit
train.glm <- glm(admit ~ ., data=train, family=binomial)
summary(train.glm)

#Checking How Accurate our Model is with ROC & AUC
train.glm.pred <- predict(train.glm, type="response")
pred = prediction(train.glm.pred, train$admit)
perf = performance(pred, measure="tpr", x.measure="fpr")
#ROC
plot(perf, col=2, lwd=3, main="ROC curve")
abline(0,1)
#AUC
performance(pred, "auc")@y.values

#Using FPR & FNR to Find the Best Threshold Value
fpr = performance(pred, "fpr")@y.values[[1]]
threshold = performance(pred, "fpr")@x.values[[1]]
fnr = performance(pred,"fnr")@y.values[[1]]
#calculating euclidean distance
rate = as.data.frame(cbind(Cutoff=threshold, FPR=fpr, FNR=fnr))
rate$distance = sqrt((rate[,2])^2+(rate[,3])^2)
#select threshold with the smallest euclidean distance
index = which.min(rate$distance)
best = rate$Cutoff[index]
best

#plot
matplot(threshold, cbind(fpr,fnr), type="l",lwd=2, xlab="Threshold",ylab="Error Rate")
legend(0.38, 1, legend=c("False Positive Rate","False Negative Rate"),col=c(1,2), lty=c(1,2))
abline(v=best, col=3, lty=3, lwd=3)

test.glm <- glm(admit ~ ., data=test, family=binomial)
#soft classifier
test.glm.pred <- predict(test.glm, type="response")
#hard classifier w/ "best" threshold
test_new = test %>% mutate(admit_pred=as.factor(ifelse(test.glm.pred<=best, 0, 1)))
#confusion matrix
conf.matrix.log <- table(pred=test_new$admit_pred, true=test$admit)
#test error rate
log_error <- 1 - sum(diag(conf.matrix.log)/sum(conf.matrix.log))
log_error

tree_fit <- tree(admit ~ ., data = train) #fitting classification tree
plot(tree_fit) #plotting tree
text(tree_fit, pretty = 0)
title("Classification Tree Built on Training Set")

admit.testset <- test$admit
#predict each observation's class using the test set
yhat.testset <- predict(tree_fit, test, type="class")
#confusion matrix
test_tree_matrix <- table(yhat.testset, admit.testset)
test_tree_matrix
```

```r
tree_error_rate <- 1-sum(diag(test_tree_matrix))/sum(test_tree_matrix)
tree_error_rate

rf_admits = randomForest(admit ~ ., data=train, ntree=500, importance = TRUE)
plot(rf_admits)
print(rf_admits)

yhat.rf = predict(rf_admits, newdata = test)
rf_error = table(pred = yhat.rf, truth = test$admit) #confusion matrix
test_rf_error = 1 - sum(diag(rf_error))/sum(rf_error) #test error rate
test_rf_error

importance(rf_admits)
varImpPlot(rf_admits, sort = T, main = "Variable Importance Plot of College Admission Factors")

error_results <- matrix(c(err_bestk, log_error, tree_error_rate, test_rf_error), ncol=1, byrow=TRUE)
colnames(error_results) <- c("Test Error Rate")
rownames(error_results) <- c("KNN (K = 35)","Logistic Regression","Decision Tree", "Random Forest")
error_results.table <- as.table(error_results)
error_results.table
```