Predicting Graduate School Admissions Given Application Materials and Data

Julius Kintop, Luke Langan, Reece Miron (Dataset: admissions9.txt)

Introduction:

The goal of this analysis is to study the importance of different aspects of graduate school applications, determining what factors are most impactful when admissions offices are determining the result of your application. In addition, we are also looking to create a model that will predict whether an applicant is admitted or not. The predictors being studied are: GRE score, TOEFL score, university rating, quality of the statement of purpose (SOP), average score of the letters of recommendation (LOR), average cumulative GPA (CGPA), and if research was conducted or not prior to applying. These variables are used to predict the response, whether or not a student is likely to be admitted. The variables most relevant in predicting the chance of admission are cumulative GPA, GRE, Research, and TOEFL.

Methods:

In the analyses performed in this study, we started by importing the dataset and changing the Research variable to a binomial factor. Next, we imputed the missing numerical data in the dataset using mean imputation, changing the missing data to the means of their variables across all of the non-missing data. If there was a missing piece of data for a categorical observation, we just excluded the observation.

On this imputed dataset, we then created a variable, Admit, which changed the variable Chance_of_Admit to a categorical variable taking the value 1 if Chance_of_Admit was greater than 0.8 and taking the value 0 otherwise.

After all of this initial data setup, we started with the analyses. To begin, we split the data into a training set with 80% of the observations and a validation set with the remaining 20% of the observations.

The first classification model we fit to predict whether a student would get admitted or not is Logistic Regression. We fit a logistic regression model on the training set and calculated the Test Error Rate, constructed an ROC curve, and calculated the AUC to determine how well the model predicted the data.

Next, we fit a KNN model to predict the same thing as before. We used cross validation to select the number of nearest neighbors, K, in which we wanted to use. Then, we calculated

the Test Error Rate, constructed an ROC curve, and calculated the AUC to assess the accuracy of the model. We used a KNN model because it is useful for large datasets with decision boundaries we don't expect to be linear. We used it over QDA and LDA because we can't say for sure that the covariates are normally distributed.

The next classification model we fit was a Random Forest model for the same purpose as the other two steps. We constructed the same accuracy methods as the previous two models. We also constructed an importance plot to help illustrate which variables are most important in predicting whether a student will get admitted or not. We chose a Random Forest model because the dataset is large, but it will also not lead to overfitting.

We then imputed the original data a second time using iterative regression. With this newly imputed data, we split the dataset into the training and validation sets, just as we did before, and fit the same three models on this dataset, using the same techniques as outlined above.

Results:

Through these analyses explained above, we obtained results on what classification methods are best suited for this data, and what models will do so. We also obtained information on which predictors are the most relevant in determining whether or not an applicant will be accepted based on their application. The following paragraphs summarize the results of this study.

Logistic regression (Mean Imputation)

When we fit a logistic regression model on the mean imputation data set, we obtain the following summary:

```
Call:
glm(formula = formula, family = binomial, data = train)
Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.45348  -0.17898  -0.02936   0.09425   3.11893
Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -78.893813  16.227207  -4.862 1.16e-06 ***
GRE                 0.021270   0.057391   0.371    0.711
TOEFL               0.139364   0.089123   1.564    0.118
University_Rating   0.613345   0.377115   1.626    0.104
SOP                -0.227907   0.562663  -0.405    0.685
LOR                 0.009557   0.444806   0.021    0.983
CGPA                6.103420   1.448826   4.213 2.52e-05 ***
Research            1.476248   0.606714   2.433    0.015 *
(Dispersion parameter for binomial family taken to be 1)
```

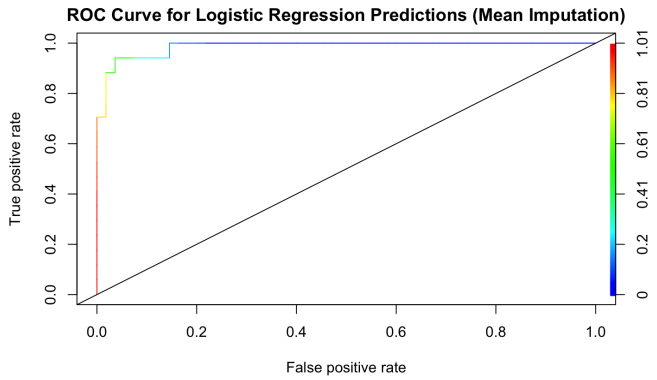
```
Residual deviance: 100.91  on 278  degrees of freedom
AIC: 116.91

Confusion Matrix:
   0  1
0 53  1
1  2 16

Error rate: 0.04166667
AUC: 0.9860963
```

Our model fitted by our training set is:

$$\hat{y} = -78.893813 + 0.021270x_1 + 0.139364x_2 + 0.613345x_3 - 0.227907x_4 + 0.009557x_5 + 6.103420x_6 + 1.476248x_7.$$

In the formula, we define x_1: GRE, x_2: $TOEFL$, x_3: $University\ Rating$, x_4: SOP, x_5: LOR x_6: $Cumulative\ GPA$, x_7: $Whether\ an\ applicant\ did\ research.$ Looking at the summary, we see CGPA and Research are the most significant in our model. Interpreting this, the estimates of the slope of CGPA: 6.103420 and Research: 1.476248 can be explained by the following sentences. With every 1 unit increase in cumulative GPA, an applicant becomes exp(6.103420)=447.385 times more likely to get admitted into graduate school, while adjusting for the other variables. In other words, for every 0.1 unit increase in Cumulative GPA an applicant becomes exp(0.6103420)=1.841 times more likely to get admitted into graduate school, while adjusting for the other variables. If the applicant did research it is exp(1.476248)=4.3764942309 times more likely to get admitted into graduate school than an applicant who did not do research, while adjusting for the other variables.

We get an error rate of 0.04166667 for the logistic regression model on this dataset. This means our model fit on our training set incorrectly predicts our test set data 4.166667% of the time. Based on our calculated error rate, we can conclude that the model is a good fit for the data. This is supported by our ROC curve as shown below, and its corresponding



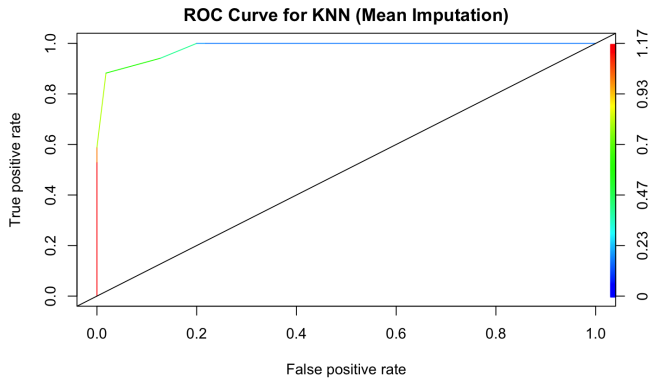
AUC=0.9860963. An AUC close to one means a better performance and implies a high sensitivity and high specificity. This is shown on the ROC curve as it is close to the point (0,1) which means we have a low false positive rate and a high true positive rate.

KNN (Mean Imputation)

```
Confusion Matrix:
     0  1
  0 49  6
  1  2 15
Test Error Rate: 0.1111111
AUC: 0.9834225
```

Based on the 5-folds cross validation we performed to choose our preferred value of K, we chose a K-value of 5 for this KNN classification. When we fit this KNN classification model with K=5 on the training set, we predicted the outcomes of the validation set based on the majority of the 5 nearest observations of the training set. This predicted 51 observations to be "Not Admitted", and 21 observations to be "Admitted". Out of the predicted "Not Admitted" outcomes, 2 were incorrectly predicted, and were actually "Admitted". Out of the predicted "Admitted" outcomes, 6 were incorrectly predicted, and were actually "Not Admitted".

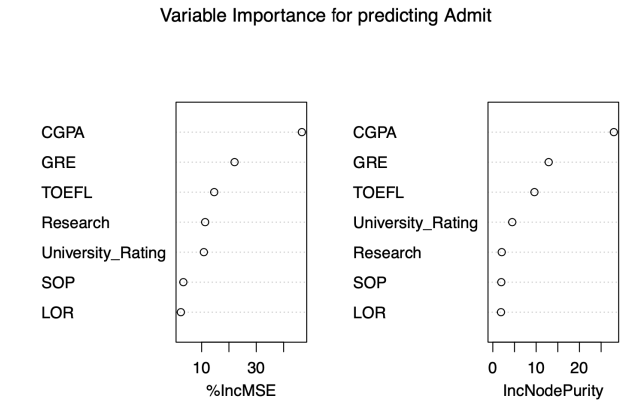
We get an error rate of 0.1111111 for the KNN model on this dataset. This means our model fit on our training set incorrectly predicts our test set data 11.11111% of the time. Based on our calculated error rate, we can conclude that the model is a good fit for the data. This is supported by our ROC curve as shown below, and its corresponding AUC=0.9834225. The graph of the ROC curve is close to the point (0,1).



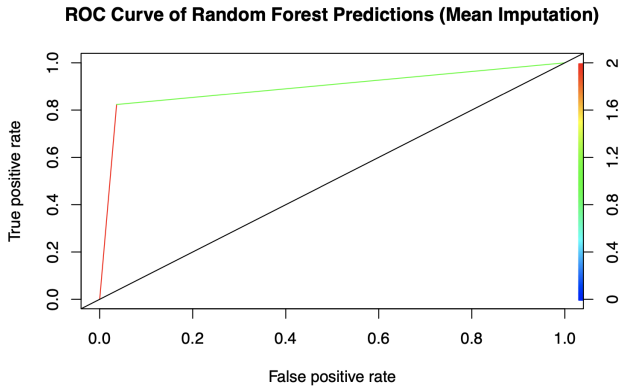
Random Forests (Mean Imputation)

```
Call:
 randomForest(formula = formula, data = train, mtry = 3, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3
```

```
          Mean of squared residuals: 0.07004414
                    % Var explained: 68.9
AUC: 0.8935829
Confusion Matrix:
       0  1
    0 53  3
    1  2 14

Test Error Rate: 0.06944444
```



This random forest model considers 3 variables at each split, which we decided to do because of the p/3 rule, which in this case is 7/3 = 2.33 which we round up to 3. We considered 500 trees to predict admissions. The random forest model predicted 55 applicants to be denied and 17 to be accepted. It incorrectly predicted 2 of the 55 and 3 of the 17. We get an error rate of 0.06944444 for the random forest on this dataset. This is supported by our ROC curve as shown



above and its corresponding AUC=0.8935829. The ROC curve is not close to the point (0,1), when compared to the other models.

Summary (Mean Imputation)

Logistic Regression: ER=0.04166667, AUC=0.9860963; KNN: ER=0.1111111, AUC=0.9834225 ; Random Forests: ER=0.06944444, AUC=0.8935829.

Comparing the three methods we used on the mean imputed data, if we solely looked at the error rate we would say the logistic regression has the lowest error rate of 0.04166667. This indicates that the model is making fewer mistakes and is therefore performing better. Looking at the AUC value, which determines performance of the model, this confirms our selection as the AUC is closest to 1 of the three models we made. A value close to 1 indicates the model's ability to make accurate predictions about the class of new data points.

Logistic Regression (Iterative Regression)

```
Call:
glm(formula = Admit ~ GRE + TOEFL + University_Rating + SOP +
    LOR + CGPA + Research, family = binomial, data = train2)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.31447  -0.10795  -0.01628   0.03955   2.18432

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)       -68.58640   17.34020  -3.955 7.64e-05 ***
GRE                -0.07371    0.06899  -1.068 0.285364
TOEFL               0.28541    0.11116   2.568 0.010239 *
University_Rating   0.97449    0.47093   2.069 0.038519 *
SOP                -0.34650    0.61113  -0.567 0.570727
LOR                 0.61145    0.45217   1.352 0.176288
CGPA                6.09914    1.66201   3.670 0.000243 ***
Research           2.63802    0.80727   3.268 0.001084 **
(Dispersion parameter for binomial family taken to be 1)
Residual deviance:  88.706  on 312  degrees of freedom
AIC: 104.71
Confusion matrix:

   0  1
0 46  6
1  2 26
Test Error Rate: 0.1
AUC: 0.953125
```

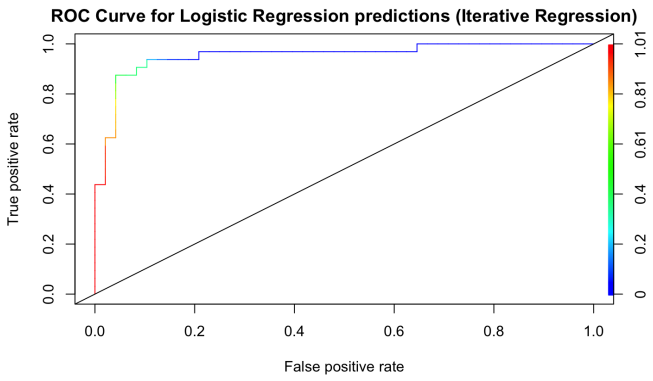
Our model fitted by our training set is:

$$\hat{y} = -68.58640 - 0.07371x_1 + 0.28541x_2 + 0.97449x_3 - 0.34650x_4 + 0.61145x_5 + 6.09914x_6 + 2.63802x_7.$$

In the formula, we define x_1: GRE, x_2: $TOEFL$, x_3: $University\ Rating$, x_4: SOP, x_5: LOR x_6: $Cumulative\ GPA$, x_7: $Whether\ an\ applicant\ did\ research.$ Looking at the summary, we

see TOEFL, University_Rating, CGPA and Research are the most significant in our model. Interpreting this, the estimates of the slope of TOEFL: 0.28541, University_Rating: 0.97449, CGPA: 6.09914 and Research: 2.63802 can be explained as by the following sentences. For every point increase in TOEFL score, an applicant becomes exp(0.28541)=1.33030734233 times more likely to get admitted into graduate school, while adjusting for the other variables. Every point increase in University Rating, an applicant becomes exp(0.97449)=2.64981546043 times more likely to get admitted into graduate school, while adjusting for the other variables. With every 1 unit increase in cumulative GPA, an applicant becomes exp(6.09914)=445.474497231 times more likely to get admitted into graduate school, while adjusting for the other variables. In other words, for every 0.1 unit increase in Cumulative GPA an applicant becomes exp(0.609914)=1.84027312849 times more likely to get admitted into graduate school, while adjusting for the other variables. If the applicant did research it is exp(2.63802)=13.9854849152 times more likely to get admitted into graduate school than an applicant who did not do research, while adjusting for the other variables.

We get an error rate of 0.1 for the logistic regression model on this dataset. This means our model fit on our training set incorrectly predicts our test set data 10% of the time. Based on our calculated error rate, we can conclude that the model is a good fit for the data. This is supported by our ROC curve as shown below, and its corresponding AUC=0.953125. The ROC curve is close to the point (0,1).



KNN (Iterative Regression)

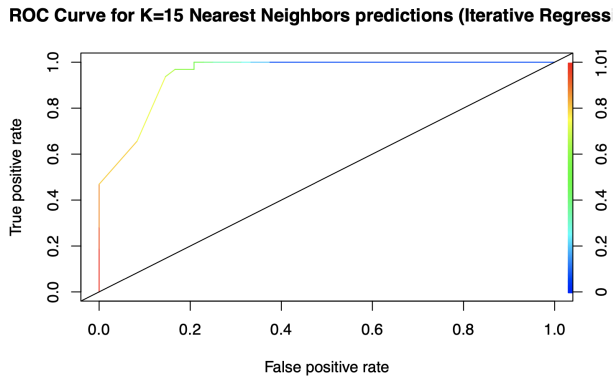
Error Rate: `0.1625`

```
Confusion Matrix:
   0  1
0 42  7
1  6 25
```

AUC: 0.9485677

Based on the 5-folds cross validation we performed to choose our preferred value of K, we chose a K-value of 15 for this KNN classification. When we fit this KNN classification model with K=15 on the training set, we predicted the outcomes of the validation set based on the majority of the 15 nearest neighbors of the training set. This predicted 48 observations to be "Not Admitted", and 32 observations to be "Admitted". Out of the predicted "Not Admitted" outcomes, 6 were incorrectly predicted, and were actually "Admitted". Out of the predicted "Admitted" outcomes, 7 were incorrectly predicted, and were actually "Not Admitted".

We get an error rate of 0.1625 for the KNN model on this dataset. This means our model fit on our training set incorrectly predicts our test set data 16.25% of the time. Based on our calculated error rate, we can conclude that the model is a good fit for the data. This is supported by our ROC curve as shown below, and its corresponding AUC=0.9485677. This is The ROC curve is close to the point (0,1).



Random Forests (Iterative Regression)

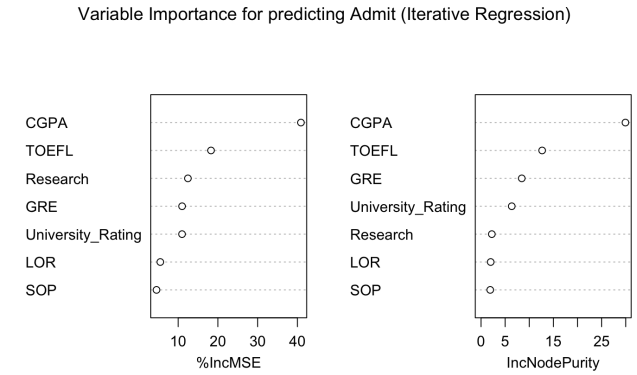
```
Call:
 randomForest(formula = formula, data = train2, mtry = 3, importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 3
       Mean of squared residuals: 0.06464668
                     % Var explained: 69.22
```

Confusion Matrix:

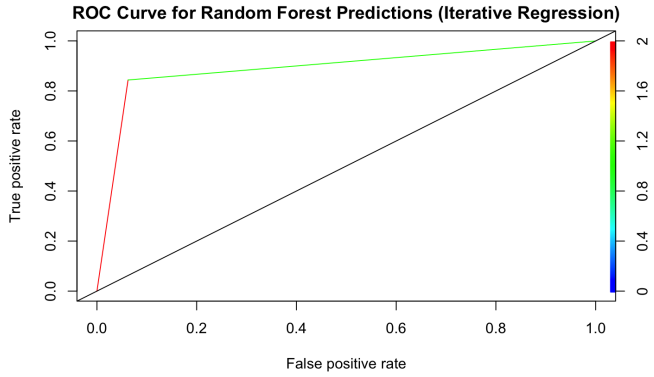
```
   0  1
0 45  5
1  3 27
```

Test Error Rate: 0.1

AUC: 0.890625



Just as we did in the last random forest, we consider 3 variables at each split and build 500 trees. The random forest model predicted 48 applicants to be denied and 32 to be accepted. It incorrectly predicted 3 of the 48 and 5 of the 32. We get an error rate of 0.1 for the random forest model on this dataset.. Based on our calculated error rate, we can conclude that the model is a good fit for the data. This is supported by our ROC curve as shown below, and its corresponding AUC=0.890625. This is a relatively low AUC compared to the other models, so this is not our best predictive model.



<u>Summary (Iterative Regression)</u>

Logistic Regression: ER=0.1 , AUC =0.953125 ; KNN: ER =0.1625 , AUC =0.9485677 ; Random Forests: ER=0.1 , AUC=0.890625 .

Comparing the three methods we used on the iterative regression data, if we solely looked at the error rate we would say the logistic regression and random forests models performed best as they both have the lowest error rate of 0.1. This indicates that the model is

making fewer mistakes and is therefore performing better. If we needed to pick just one model we would look at their AUC values, which determines performance of the model and a value close to 1 indicates the model's ability to make accurate predictions about the class of new data points. With this, we would say the logistic regression model performed the best using the data with iterative regression. Observing our two different data sets, mean imputation and iterative regression, we can see that the random forest models in both performed the worst(lowest AUC, highest ER). This can be explained by the model's flexibility. Random forests are the most flexible model and in turn has a higher variance. We see the best model is the logistic regression model for both data sets, this is logically correct as the logistic regression model is the least flexible and goes without losing prediction performance.

Discussion:

After performing two different types of missing data imputation and three different predictive models on each of the datasets, we find that in general, the models we made were good at predicting whether or not students would be admitted to graduate school based on their cumulative GPA, TOEFL scores, GRE scores, University rating, SOP, scores of letters of recommendation, and quality of their statements of purpose. The variables that were the most important for predicting admission status were CGPA, TOEFL score, and GRE score, and Research, with a large emphasis on CGPA. The model with the best predictive performance in the mean imputed dataset was logistic regression, with an error rate less than 0.05 and an AUC of 0.986, which is very good. The best model at predicting admissions in the iterative regression dataset was also logistic regression with an error rate of 0.1 and AUC of 0.953. The error rates were higher in general for the iterative regression imputation which is especially apparent with KNN, which produced an error rate of 0.1625, which is relatively inaccurate in terms of predictive power. We could also fit all potential models we considered that we didn't actually fit including: LDA, QDA, Bagging, and Boosting, and picking the most accurate of those. We could also choose every single tuning parameter involved using cross-validated error. However, these methods would be computationally expensive.

Appendix:

```r
#read in data
data = read.table("admissions9-2.txt")
```

Data Preparation

```r
# impute the mean for missing continuous data
mean_imp_data = data
mean_imp_data$TOEFL[is.na(data$TOEFL)] = mean(data$TOEFL, na.rm=TRUE)
summary(mean_imp_data)
```

```
##        ID              GRE             TOEFL       University_Rating
##  Min.   :  1.0   Min.   :290.0   Min.   : 92.0   Min.   :1.000
##  1st Qu.:100.8   1st Qu.:308.0   1st Qu.:104.0   1st Qu.:2.000
##  Median :200.5   Median :317.0   Median :107.6   Median :3.000
##  Mean   :200.5   Mean   :316.8   Mean   :107.6   Mean   :3.087
##  3rd Qu.:300.2   3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000
##  Max.   :400.0   Max.   :340.0   Max.   :120.0   Max.   :5.000
##
##       SOP             LOR             CGPA          Research      Chance_of_Admit
##  Min.   :1.0   Min.   :1.000   Min.   :6.800   Min.   :0.0000   Min.   :0.340
##  1st Qu.:2.5   1st Qu.:3.000   1st Qu.:8.170   1st Qu.:0.0000   1st Qu.:0.640
##  Median :3.5   Median :3.500   Median :8.610   Median :1.0000   Median :0.740
##  Mean   :3.4   Mean   :3.453   Mean   :8.599   Mean   :0.5279   Mean   :0.732
##  3rd Qu.:4.0   3rd Qu.:4.000   3rd Qu.:9.062   3rd Qu.:1.0000   3rd Qu.:0.840
##  Max.   :5.0   Max.   :5.000   Max.   :9.920   Max.   :1.0000   Max.   :0.970
##                                                NA's   :42       NA's   :26
```

```r
mean_imp_data$Chance_of_Admit[is.na(data$Chance_of_Admit)] =
  mean(data$Chance_of_Admit, na.rm=TRUE)

mean_imp_data = mean_imp_data[!is.na(mean_imp_data$Research), ]

# create variable Admit (0 or 1)
mean_imp_data["Admit"] = ifelse(mean_imp_data$Chance_of_Admit >= 0.8, 1, 0)
```

Suitable Logistic Regression Model

```r
set.seed(1234)

# Split into training and testing sets
sample = sample(nrow(mean_imp_data), 0.8*nrow(mean_imp_data))
```

```r
train = mean_imp_data[sample,]
test = mean_imp_data[-sample,]

# predict Admit using logistic regression
formula = Admit ~ GRE + TOEFL + University_Rating + SOP + LOR + CGPA + Research
logreg1 = glm(formula, data=train, family=binomial)
summary(logreg1)
```

```
##
## Call:
## glm(formula = formula, family = binomial, data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.45348  -0.17898  -0.02936   0.09425   3.11893
##
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -78.893813  16.227207  -4.862 1.16e-06 ***
## GRE                 0.021270   0.057391   0.371    0.711
## TOEFL               0.139364   0.089123   1.564    0.118
## University_Rating   0.613345   0.377115   1.626    0.104
## SOP                -0.227907   0.562663  -0.405    0.685
## LOR                 0.009557   0.444806   0.021    0.983
## CGPA                6.103420   1.448826   4.213 2.52e-05 ***
## Research            1.476248   0.606714   2.433    0.015 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 367.67  on 285  degrees of freedom
## Residual deviance: 100.91  on 278  degrees of freedom
## AIC: 116.91
##
## Number of Fisher Scoring iterations: 7
```

```r
# make predictions using logistic regression model
pred_log1 = predict(logreg1, newdata=test, type = "response")
pred_log1 = ifelse(pred_log1 >= 0.5, 1, 0)
y_log1 = ifelse(pred_log1 > 0.5, 1, 0)
ER_log1 = mean((test$Admit - y_log1)^2)
ER_log1
```

```
## [1] 0.04166667
```

```
table(pred_log1, test$Admit)
```

```
##
## pred_log1  0  1
##         0 53  1
##         1  2 16
```

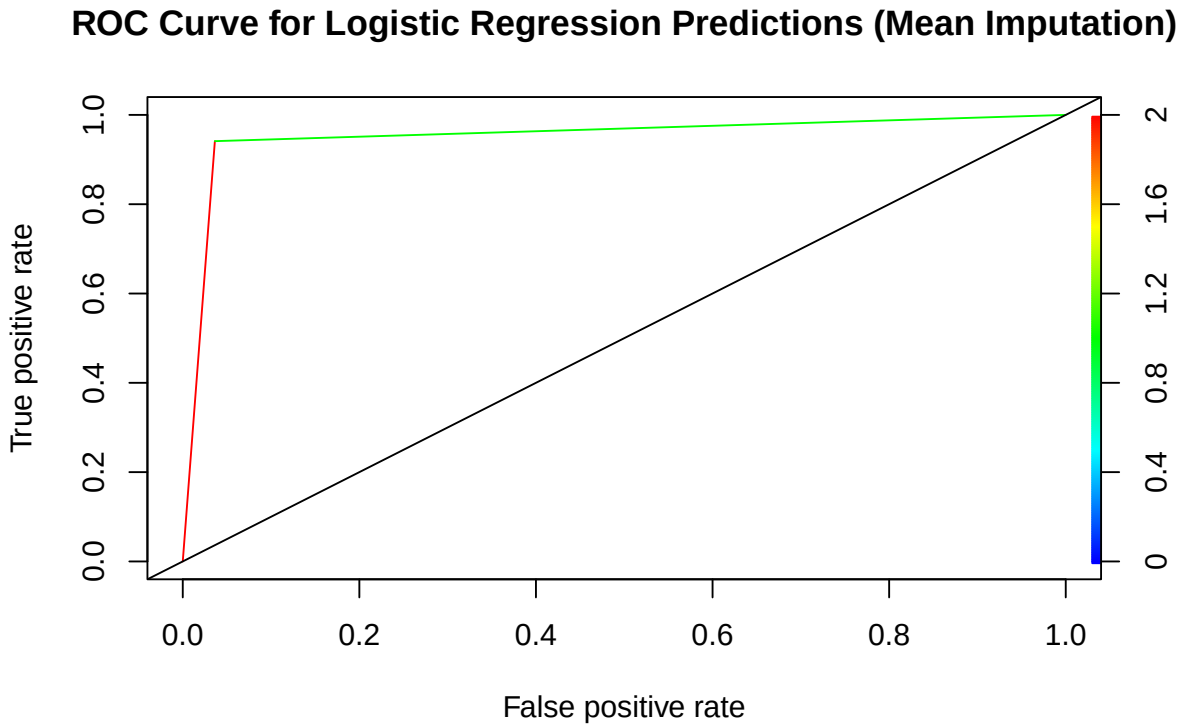
```
summary(pred_log1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00    0.25    0.25    1.00
```

```
library(ROCR)
pred_logit_test = prediction(pred_log1, test$Admit)
perf_logit_test = performance(pred_logit_test, "tpr", "fpr")
roc_logreg1 = plot(perf_logit_test, colorize=TRUE,
main = "ROC Curve for Logistic Regression Predictions (Mean Imputation)")
performance(pred_logit_test, "auc")@y.values[[1]]
```

```
## [1] 0.9524064
```

```
abline(0, 1)
```



K-Nearest Neighbors Classification

```
set.seed(1234)
# Implement KNN to predict Admit
library(class)
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
# Use 5-fold CV to pick a good value for K, between 5, 10, and 15

X = mean_imp_data[, c("GRE", "TOEFL", "University_Rating",
                      "SOP", "LOR", "CGPA", "Research")]
y = mean_imp_data[, "Admit"]

nfolds = 5
kCV_error = rep(0,3)
fold = createFolds(1:nrow(mean_imp_data), k=nfolds, list=FALSE)
for(i in 1:nfolds){
  k5_cv = knn(as.matrix(X[fold != i,]), as.matrix(X[fold == i,]), cl = y[fold != i], k=5)
  k10_cv = knn(as.matrix(X[fold != i,]), as.matrix(X[fold == i,]), cl = y[fold != i], k=10)
  k15_cv = knn(as.matrix(X[fold != i,]), as.matrix(X[fold == i,]), cl = y[fold != i], k=15)

  kCV_error[1] = kCV_error[1]+mean(k5_cv != y[fold==i])/nfolds
  kCV_error[2] = kCV_error[2]+mean(k10_cv != y[fold==i])/nfolds
  kCV_error[3] = kCV_error[3]+mean(k15_cv != y[fold==i])/nfolds
}
data.frame(k=c(5,10,15), CV_error=kCV_error)
```

```
##    k  CV_error
## 1  5 0.1172926
## 2 10 0.1201878
## 3 15 0.1229264
```

```
# Based on this, I will use k=15 to classify students using KNN

pred_cols = c("GRE", "TOEFL", "University_Rating", "SOP", "LOR", "CGPA", "Research")
pred5 = knn(as.matrix(train[, pred_cols]),
            as.matrix(test[, pred_cols]), cl = train$Admit, k=5)
table(test$Admit, pred5)
```

```
##    pred5
##      0  1
##   0 49  6
##   1  2 15
```

```
test_ER = mean(test$Admit != pred5)
test_ER
```

```
## [1] 0.1111111
```

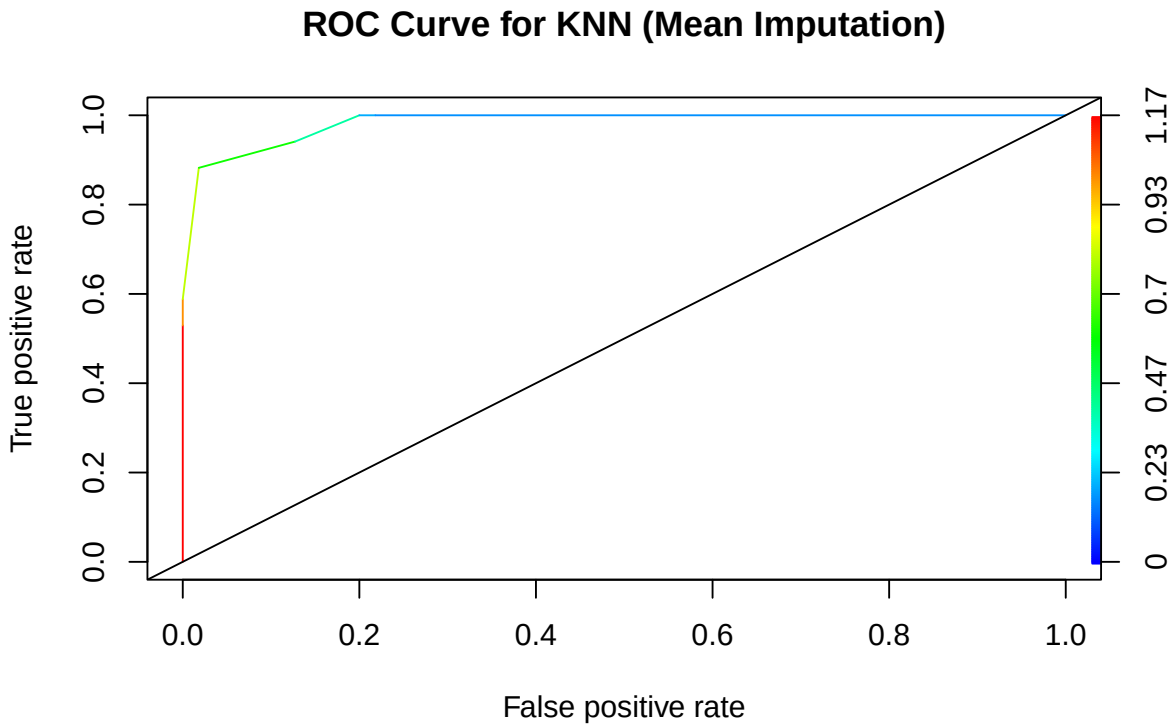
```
summary(pred5)
```

```
##  0  1
## 51 21
```

```
# ROC curve for KNN pt1

xval1 = as.matrix(X[-sample, pred_cols])
winning_pi_knn_train1=attr(knn(xval1,xval1,test$Admit,k=5,prob=TRUE),"prob")
winning_class1=knn(xval1,xval1,test$Admit,k=5)
pi_knn_train1=ifelse(winning_class1=="1",winning_pi_knn_train1,1-winning_pi_knn_train1)
pred_knn1=prediction(pi_knn_train1,test$Admit)
perf_knn1=performance(pred_knn1,"tpr","fpr")
plot(perf_knn1,colorize=TRUE,
     main="ROC Curve for KNN (Mean Imputation)")

(AUC_knn1=performance(pred_knn1,"auc")@y.values[[1]])
```

```
## [1] 0.9834225
```

```
abline(0, 1)
```



Random Forest

```
set.seed(1234)
library(randomForest)
```

```
## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
#Build a random forest, set mtry = 2 (Find justification for this parameter)

rf1 = randomForest(formula, data=train, mtry=3, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
pred1 = predict(rf1, test)
pred1 = ifelse(pred1 >= 0.5, 1, 0)
table(pred1, test$Admit)
```

```
##
## pred1  0  1
##     0 53  3
##     1  2 14
```

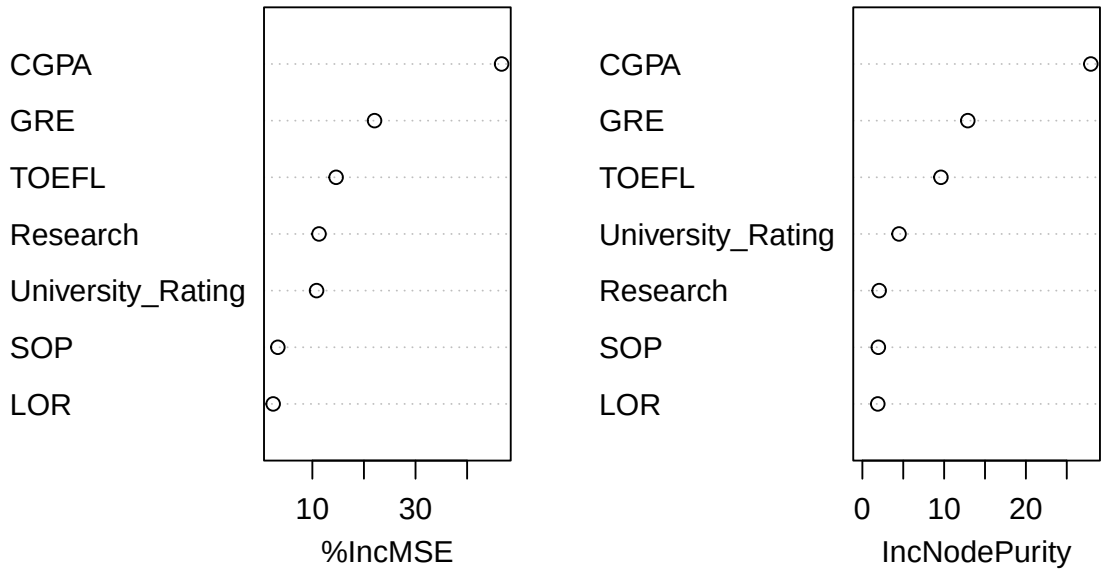
```
test_ER_rf = mean(pred1 != test$Admit)
test_ER_rf
```

```
## [1] 0.06944444
```

```
importance(rf1)
```

```
##                     %IncMSE IncNodePurity
## GRE                22.054837     12.893699
## TOEFL              14.605852      9.607285
## University_Rating  10.818270      4.481345
## SOP                 3.307282      1.951872
## LOR                 2.400875      1.882927
## CGPA               46.683856     27.940872
## Research           11.289973      2.055653
```

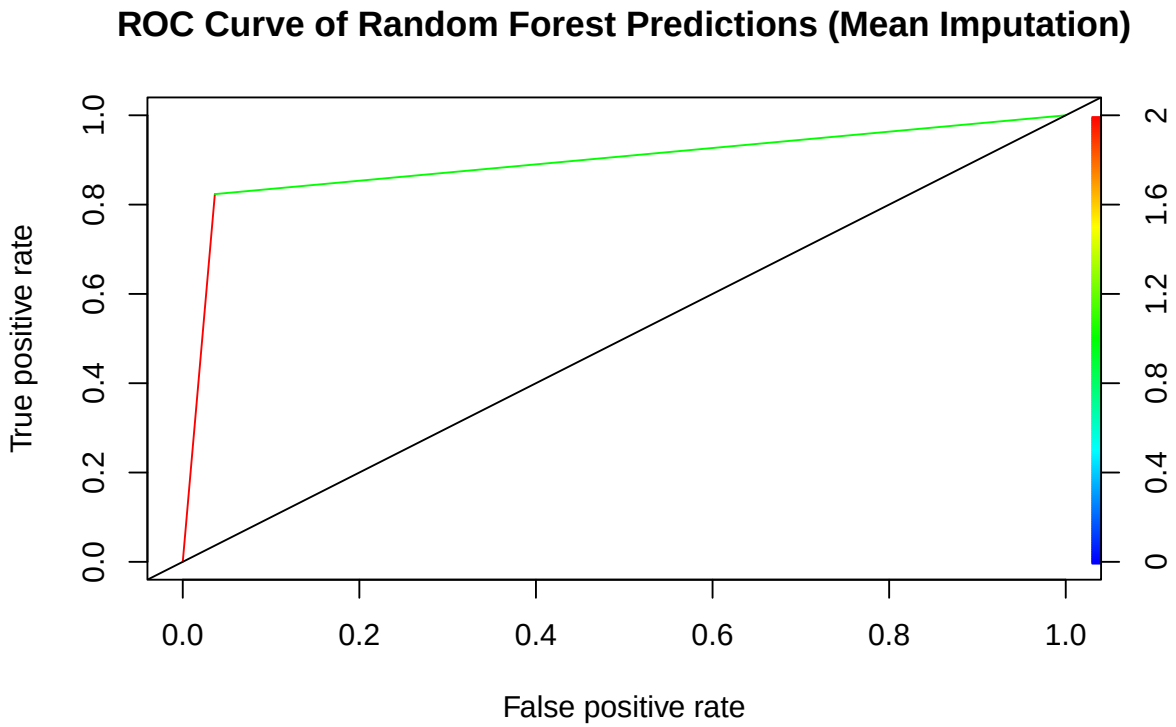
```
varImpPlot(rf1, main = "Variable Importance for predicting Admit")
```




```
# Random Forest ROC
pi_rf1 = predict(rf1, newdata=test)
pi_rf1 = ifelse(pi_rf1 > 0.5, 1, 0)
pred_rf1 = prediction(pi_rf1, test$Admit)
perf_rf1 = performance(pred_rf1, "tpr", "fpr")
plot(perf_rf1, colorize=TRUE,
     main="ROC Curve of Random Forest Predictions (Mean Imputation)")
(auc_rf = performance(pred_rf1, "auc")@y.values[[1]])
```

```
## [1] 0.8935829
```

```
abline(0, 1)
```




```
set.seed(1234)
data$Research = as.factor(data$Research)
iter_reg_data=data
iter_reg_data$Research[is.na(iter_reg_data$Research)]="1"

iter_reg_data$TOEFL[is.na(iter_reg_data$TOEFL)]=mean(data$TOEFL,na.rm=TRUE)


#Set number of iterations
n_iter=10
for(i in 1:n_iter)
{
#impute Chance_of_Admit give rest
m_Chance_of_Admit=lm(Chance_of_Admit~.-ID,iter_reg_data,
                     subset=!is.na(data$Chance_of_Admit))
pred_Chance_of_Admit=predict(m_Chance_of_Admit,
                             iter_reg_data[is.na(data$Chance_of_Admit),])
iter_reg_data$Chance_of_Admit[is.na(data$Chance_of_Admit)]=pred_Chance_of_Admit
#impute TOEFL given rest
m_TOEFL=lm(TOEFL~.-ID,data=iter_reg_data,subset =! is.na(data$TOEFL))
pred_TOEFL=predict(m_TOEFL,iter_reg_data[is.na(data$TOEFL),])
iter_reg_data$TOEFL[is.na(data$TOEFL)]=pred_TOEFL
```

```
#impute Research given rest
library(nnet)
m_Research=multinom(Research~.-ID,iter_reg_data,
                    subset =! is.na(data$Research), trace = FALSE)
pred_Research=predict(m_Research,iter_reg_data[is.na(data$Research),])
iter_reg_data$Research[is.na(data$Research)]=pred_Research
}
##Change Admit
iter_reg_data$Admit <- ifelse(iter_reg_data$Chance_of_Admit >= 0.8, 1, 0)
```

Logistic Regression

```
set.seed(1234)
# Split into training and testing sets
sample2 = sample(nrow(iter_reg_data), 0.8*nrow(iter_reg_data))
train2 = iter_reg_data[sample2,]
test2 = iter_reg_data[-sample2,]

logreg2 = glm(Admit ~ GRE + TOEFL + University_Rating
              + SOP + LOR + CGPA + Research, data=train2, family=binomial)

# make predictions using logistic regression model
pred_log2 = predict(logreg2, newdata=test2, type = "response")
pred_log2 = ifelse(pred_log2 >= 0.5, 1, 0)
y_log2 = ifelse(pred_log2 > 0.5, 1, 0)
ER_log2 = mean((test2$Admit - y_log2)^2)
ER_log2
```

```
## [1] 0.1
```

```
summary(logreg2)
```

```
##
## Call:
## glm(formula = Admit ~ GRE + TOEFL + University_Rating + SOP +
##     LOR + CGPA + Research, family = binomial, data = train2)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.31447  -0.10795  -0.01628   0.03955   2.18432
##
## Coefficients:
```

```
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -68.58640   17.34020  -3.955 7.64e-05 ***
## GRE                -0.07371    0.06899  -1.068 0.285364
## TOEFL               0.28541    0.11116   2.568 0.010239 *
## University_Rating   0.97449    0.47093   2.069 0.038519 *
## SOP                -0.34650    0.61113  -0.567 0.570727
## LOR                 0.61145    0.45217   1.352 0.176288
## CGPA                6.09914    1.66201   3.670 0.000243 ***
## Research1           2.63802    0.80727   3.268 0.001084 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 390.953  on 319  degrees of freedom
## Residual deviance:  88.706  on 312  degrees of freedom
## AIC: 104.71
##
## Number of Fisher Scoring iterations: 8
```

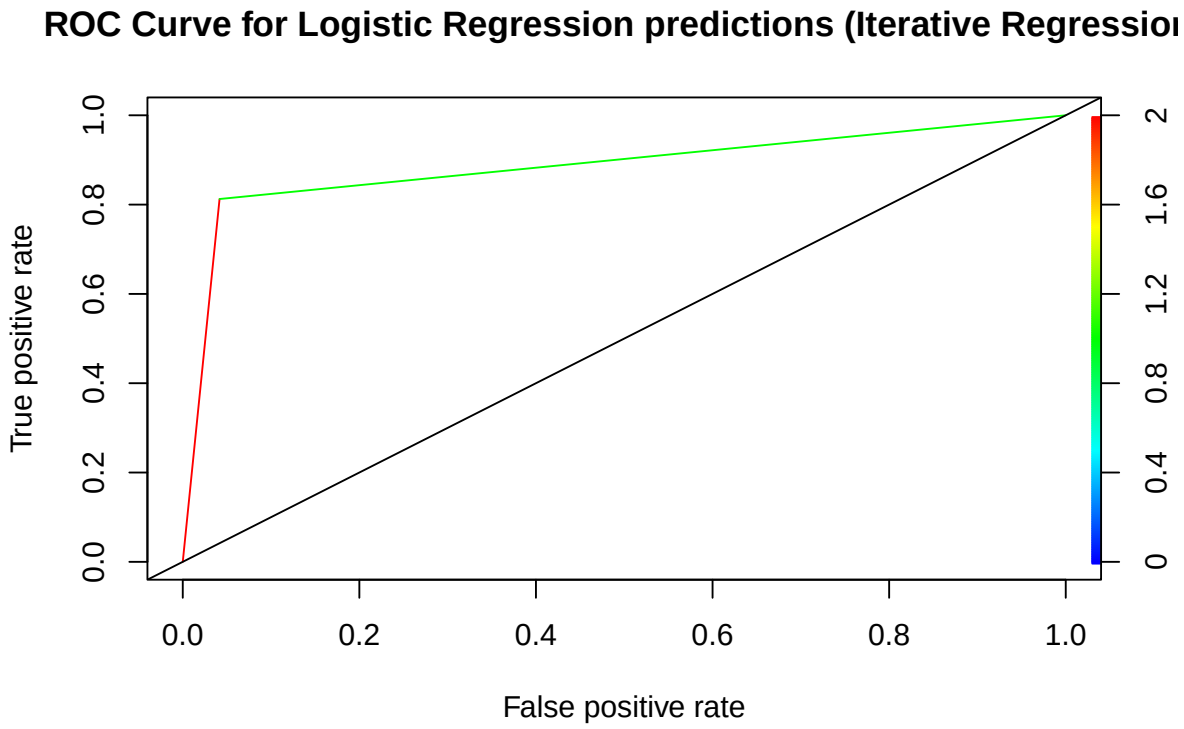
```
table(pred_log2, test2$Admit)
```

```
##
## pred_log2  0  1
##         0 46  6
##         1  2 26
```

```
# ROC curve for Logistic Regression 2
pred_logit_test2 = prediction(pred_log2, test2$Admit)
perf_logit_test2 = performance(pred_logit_test2, "tpr", "fpr")
plot(perf_logit_test2, colorize=TRUE, main =
       "ROC Curve for Logistic Regression predictions (Iterative Regression)")
performance(pred_logit_test2, "auc")@y.values[[1]]
```

```
## [1] 0.8854167
```

```
abline(0, 1)
```



KNN

```
set.seed(1234)
# Implement KNN to predict Admit

# Use 5-fold CV to pick a good value for K, between 5, 10, and 15

X2 = iter_reg_data[, c("GRE", "TOEFL",
                       "University_Rating", "SOP", "LOR", "CGPA", "Research")]
y2 = iter_reg_data[, "Admit"]

nfolds2 = 5
kCV_error2 = rep(0,3)
fold2 = createFolds(1:nrow(iter_reg_data), k=nfolds, list=FALSE)
for(i in 1:nfolds2){
  k5_cv2 = knn(as.matrix(X2[fold2 != i,]),
               as.matrix(X2[fold2 == i,]), cl = y2[fold2 != i], k=5)
  k10_cv2 = knn(as.matrix(X2[fold2 != i,]),
                as.matrix(X2[fold2 == i,]), cl = y2[fold2 != i], k=10)
  k15_cv2 = knn(as.matrix(X2[fold2 != i,]),
```

```
                as.matrix(X2[fold2 == i,]), cl = y2[fold2 != i], k=15)

  kCV_error2[1] = kCV_error2[1]+mean(k5_cv2 != y2[fold2==i])/nfolds2
  kCV_error2[2] = kCV_error2[2]+mean(k10_cv2 != y2[fold2==i])/nfolds2
  kCV_error2[3] = kCV_error2[3]+mean(k15_cv2 != y2[fold2==i])/nfolds2
}
data.frame(k=c(5,10,15), CV_error=kCV_error2)
```

```
##    k CV_error
## 1  5   0.1325
## 2 10   0.1225
## 3 15   0.1200
```

```
# Based on this, I will use k=15 to classify students using KNN

pred_cols = c("GRE", "TOEFL", "University_Rating", "SOP", "LOR", "CGPA", "Research")
pred15_2 = knn(as.matrix(train2[, pred_cols]),
               as.matrix(test2[, pred_cols]), cl = train2$Admit, k=15)
table(test2$Admit, pred15_2)
```

```
##    pred15_2
##      0  1
##   0 42  6
##   1  7 25
```

```
test_ER2 = mean(test2$Admit != pred15_2)
test_ER2
```

```
## [1] 0.1625
```

```
summary(pred15_2)
```

```
##  0  1
## 49 31
```

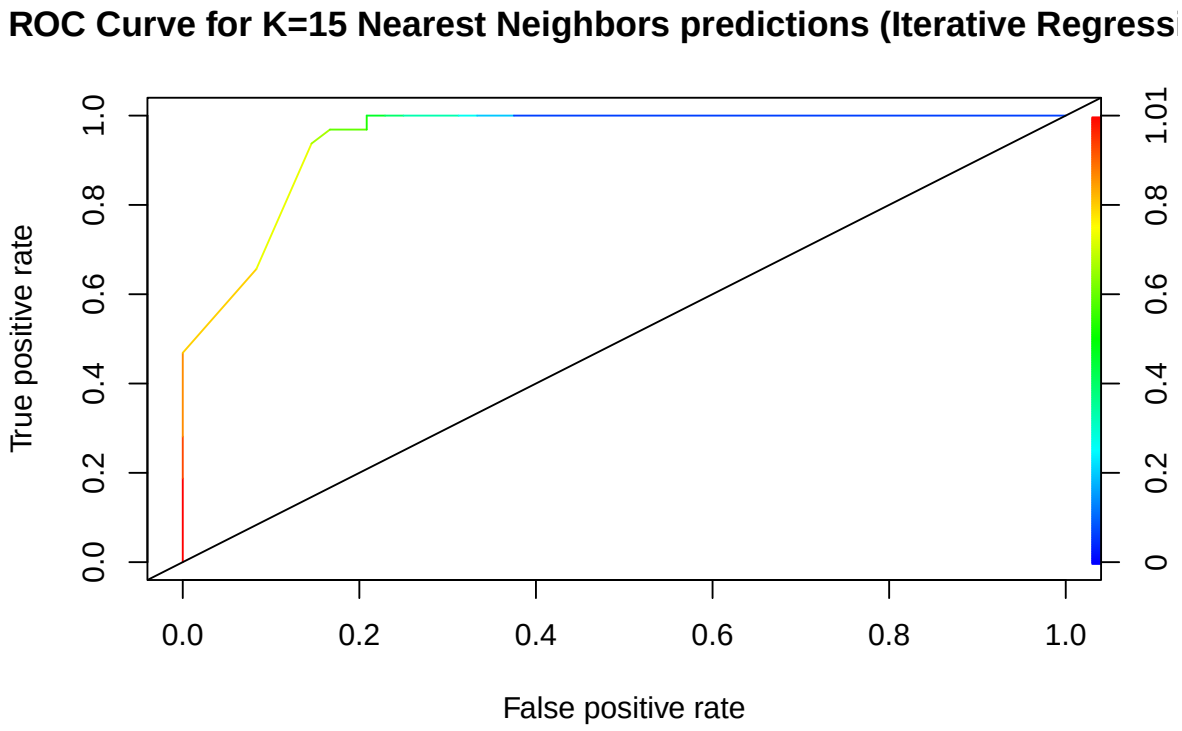
```
table(pred15_2, test2$Admit)
```

```
##
## pred15_2  0  1
##        0 42  7
##        1  6 25
```

```
set.seed(1234)
# ROC curve for KNN 2
xval2 = as.matrix(X2[-sample2, pred_cols])
winning_pi_knn_train2=attr(knn(xval2,xval2,test2$Admit,
                               k=15,prob=TRUE),"prob")
winning_class2=knn(xval2,xval2,test2$Admit,k=15)
pi_knn_train2=ifelse(winning_class2=="1",winning_pi_knn_train2,
                     1-winning_pi_knn_train2)
pred_knn2=prediction(pi_knn_train2,test2$Admit)
perf_knn2=performance(pred_knn2,"tpr","fpr")
plot(perf_knn2,colorize=TRUE,
main="ROC Curve for K=15 Nearest Neighbors predictions (Iterative Regression)")

(AUC_knn2=performance(pred_knn2,"auc")@y.values[[1]])
```

```
## [1] 0.9485677
```

```
abline(0, 1)
```



Random Forest

```
set.seed(1234)

#Build a random forest, set mtry =3

rf2 = randomForest(formula, data=train2, mtry=3, importance = TRUE)
```

```
## Warning in randomForest.default(m, y, ...): The response has five or fewer
## unique values. Are you sure you want to do regression?
```

```
pred2 = predict(rf2, test2)
pred2 = ifelse(pred2 >= 0.5, 1, 0)
table(pred2, test2$Admit)
```

```
##
## pred2  0  1
##     0 45  5
##     1  3 27
```

```
test_ER_rf2 = mean(pred2 != test2$Admit)
test_ER_rf2
```

```
## [1] 0.1
```

```
rf2
```

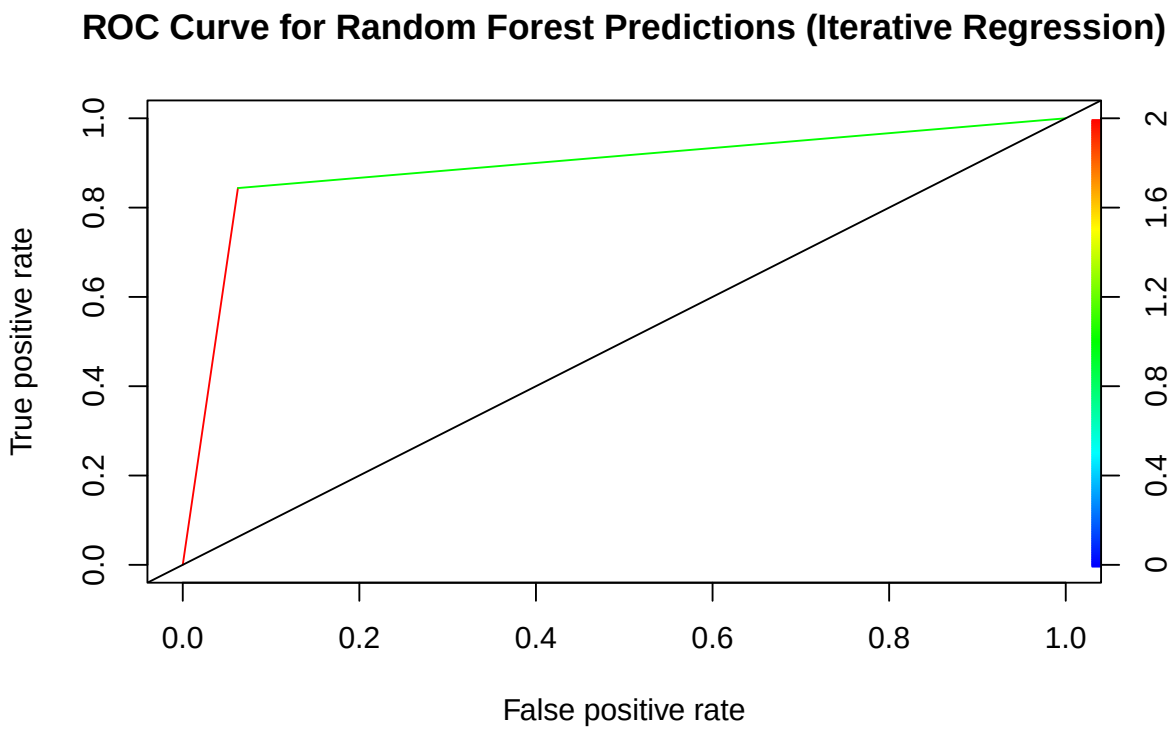
```
##
## Call:
##  randomForest(formula = formula, data = train2, mtry = 3, importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
##
##           Mean of squared residuals: 0.06464668
##                     % Var explained: 69.22
```

```
# ROC Curve for Random Forest 2
pi_rf2 = predict(rf2, newdata=test2)
pi_rf2 = ifelse(pi_rf2 > 0.5, 1, 0)
pred_rf2 = prediction(pi_rf2, test2$Admit)
```

```
perf_rf2 = performance(pred_rf2, "tpr", "fpr")
plot(perf_rf2, colorize=TRUE,
     main="ROC Curve for Random Forest Predictions (Iterative Regression)")
(auc_rf2 = performance(pred_rf2, "auc")@y.values[[1]])
```

```
## [1] 0.890625
```

```
abline(0, 1)
```




```
varImpPlot(rf2,
      main = "Variable Importance for predicting Admit (Iterative Regression)")
```

Variable Importance for predicting Admit (Iterative Regression)