MyOwnProjectJohnEstrada- Univesity Admissions

John Estrada

1/8/2020

TABLE OF CONTENT

INTRODUCTION

The main objective of this project is to develop an algorithm using the "admissions" dataset to predict to prospective graduate students admitted to a University in the "validation" set as if the admitted/not admitted were unknown. The dataset was downloaded from Kaggle and more information is provided on the next section "About the Dataset". The "Overall Accuracy" will be used as the measuring criteria to determine how close are the results obtained from the herein algorithm to the actual validation dataset. In order to avoid training the set using the validation set, an additional partition (train_set, test_set) has been created for training purposes. Different models are evaluated on these partition in order to determine which one minimizes the Accuracy. Then, that trained model is implemented to retrain the "admissions" dataset and evaluate the final Overal Accuracy against the "validation" data set. Different predictive models such linear models, logistic regression models, K-nearest neighbours, cross-validation, decision tree and random forest are evaluated.

In this report you will navigate through the different steps taken in consideration for data analyisis. This steps include Data cleaning, Data exploration and visualization, Analysis from the data and Models Aproach. The results and analysis are presented. Finally, some conslusions and future work are listed.

NOTE TO THE GRADER: The code to elaborate this report is hidden. Only the code for determining the best fit model from the train_set and test_set, and the application of that model to the validation data is displaed on this report. If you decide to take a look to the whole code, please refer to the .Rmd file or .R code both attached to the submission or available at the following repository.

https://github.com/jeestrad/Graduate-Admissions

Thank you for your comments and feedback.

ABOUT THE DATASET

The dataset was downloaded from the database available in Kaggle https://www.kaggle.com/mohansacharya/graduate-admissions#Admission_Predict.csv This database is credited to:

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

I chose this data because the topic is appealing to me and because the usability rating in Kaggle is high. Altough the dataset is small, I wanted to experience the challenge of working with a small dataset as not always I have extensive amount of data in my field.

Since some graders may require a Kaggle account, I have included the dataset to be downloaded directly from my repository depicted on the introduction of this report.

The variables (columns) of the dataset include:

Serial No: A consecutive number assigned to the candidate on the poll GRE Score: GRE Scores (out of 340) TOEFL Score: TOEFL Scores (out of 120) Univerisity Rating: University Rating (out of 5), being 5 the Top Univeristy and 1 the ones at the bottom. SOP: Statement of Purpose Strength (out of 5) LOR: Letter of Recommendation Strength (out of 5) CGPA: Undergraduate GPA (out of 10) Research: Research Experience (either 0 or 1) Chance of Admit: This was answered by the candidate as his/her chances to be admitted

In summary, the dataset include the polls of 500 prospective graduate students who described their GRE score, TOEFL score, SOP strength, LOR strength, undergrad GPA (CGPA), research experience (yes or no), the University rating of the Univerisity each student applied, and finally the "Chance of Admit".

The Chance of Admit is the chance that each applicant thought of being accepted to the Univerisity applied. Unfortunately, the actual admitted or not is not part of the dataset but the information by the dataset is great, useful and can provide a formidable insight.

Because of that, I applied a cutoff equal to the average of Chance of Admit, arbitrary. In that sense, simulating as that student above the cutoff was admitted and the one below the cuttoff did not. After that, I deleted the column used "Chance of Admit" to avoid making it a predictor on the models. Also, I made admitted equal to 1 and not admitted equal to 0.

Now, we have a new dataset with the polls of the student and a column stating if the student was admitted or not.

The dataset is automatically downloaded in R as well as the required libraries.

A partition 9/1 has been created. 90% of the data is for training purposes and called "admit", 10% of the data is for validation purposess and called "validation". For training purposes and in order to avoid using the validation dataset for training purpuses an additional partition from the "admit" dataset was created as "train_set" and "test_set" in a proportion 9 to 1, respectively.

METHODS

The methodology followed to maximize the General Accuracy is described as follows.

Data Exploration and Analysis

Data exploration and visualization: The first step of the data exploration is to determine the dimensions of the datasets. Also, to see the behavior of the predictors and evaluate whether they are correlated among each other or to the outcomes.

The dimension of the train_set is 455 rows and 9 columns. The test_set has 45 rows and 9 columns. The proportion of students admitted on the train set and on the test set are as follows.

```
## # A tibble: 6 x 9
##   Serial   GRE TOEFL Urating   SOP   LOR  CGPA Research admitted
##    <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>    <dbl>
## 1    495   301    99       3   2.5   2    8.45        1        0
## 2    496   332   108       5   4.5   4    9.02        1        1
## 3    497   337   117       5   5     5    9.87        1        1
## 4    498   330   120       5   4.5   5    9.56        1        1
## 5    499   312   103       4   4     5    8.43        0        1
## 6    500   327   113       4   4.5   4.5  9.04        0        1
```

```
## [1] 455   9
```

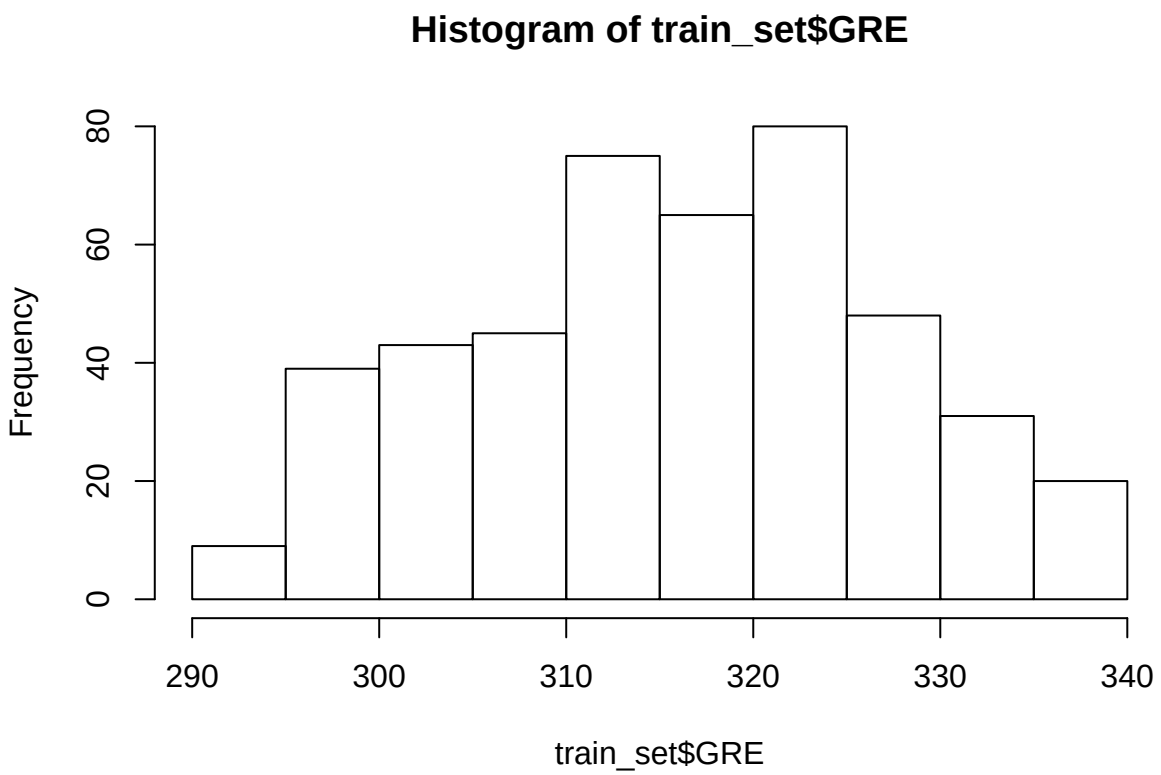
```
## [1] 45  9
```

```
## [1] 0.5111111
```

```
## [1] 0.5296703
```

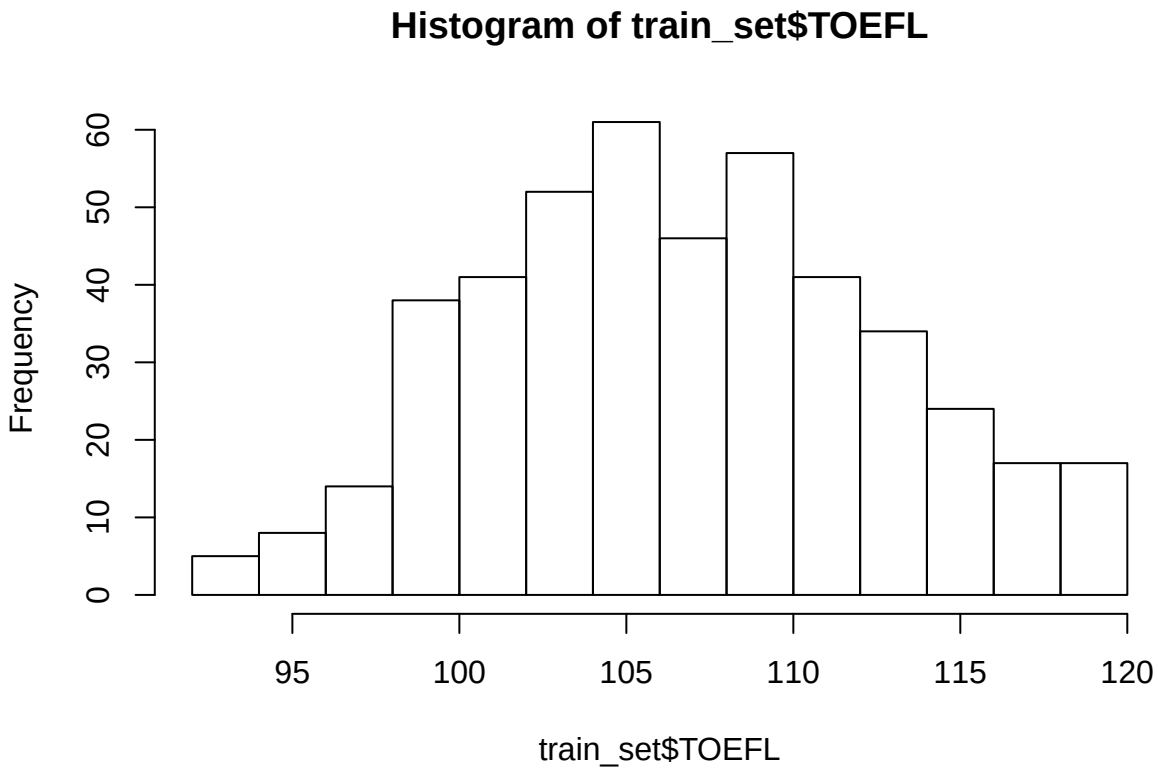
The GRE average in the train data set appears below. The histogram seems to have a bimodal distribution.

```
## [1] 316.3648
```

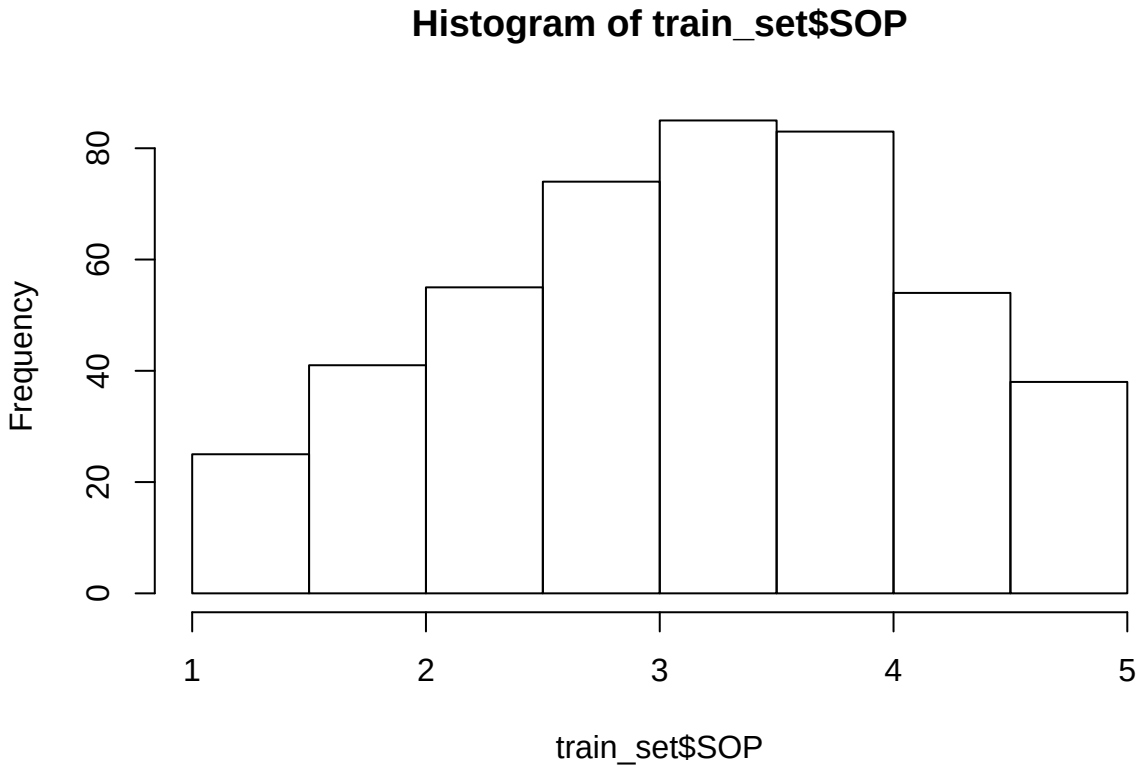


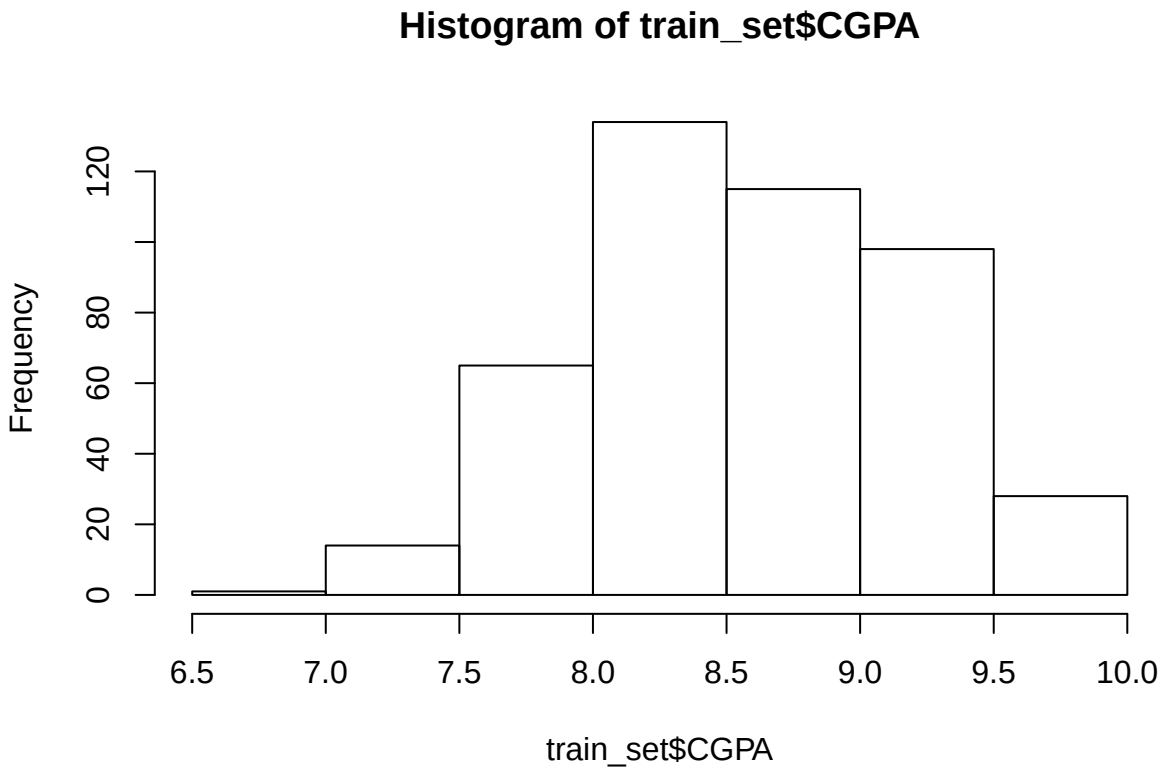
The TOEFL score average in the train data set appears below. From the historgram, the TOEFL also seems to have a bimodal distribution negatively skewed.

```
## [1] 107.233
```

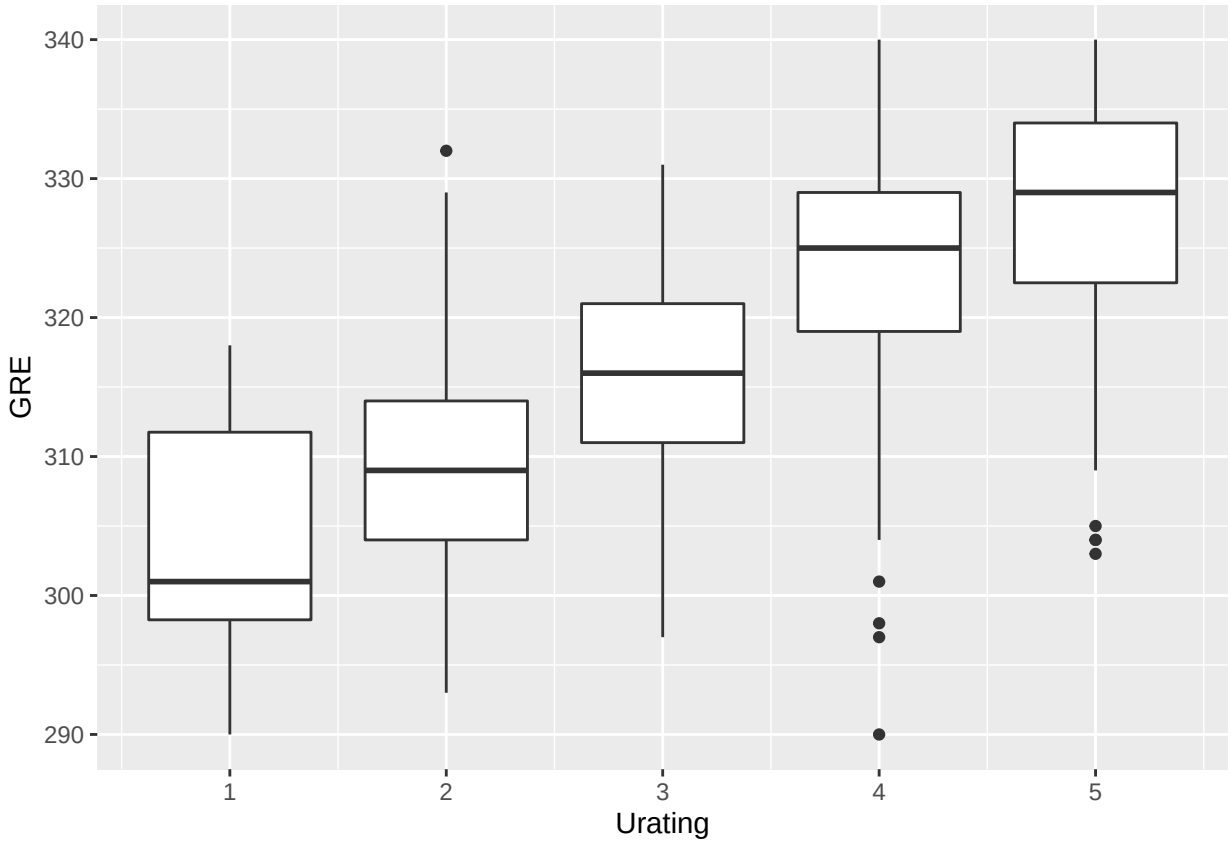


The SOP and CGPA seem to be skewed.





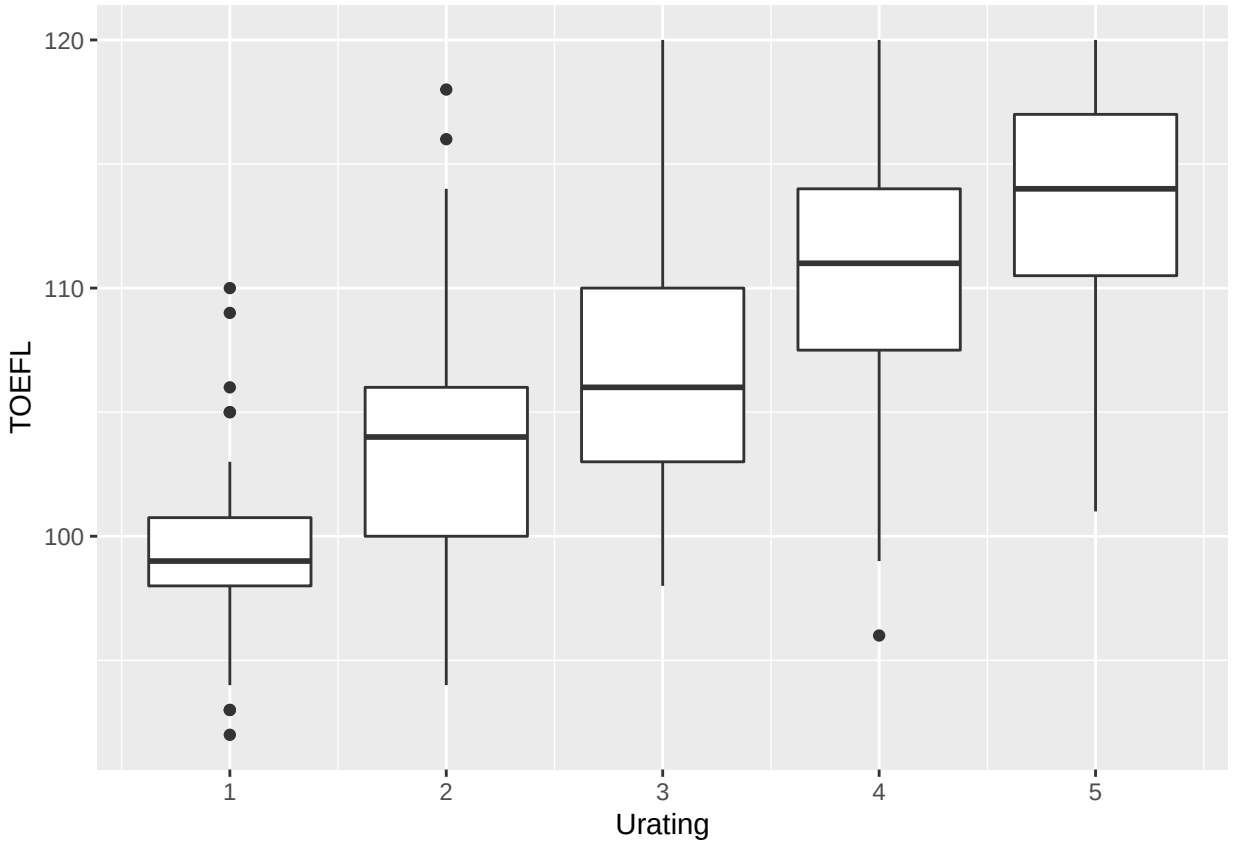
We can also see how is the behavior of the GRE with respect to the University Rating

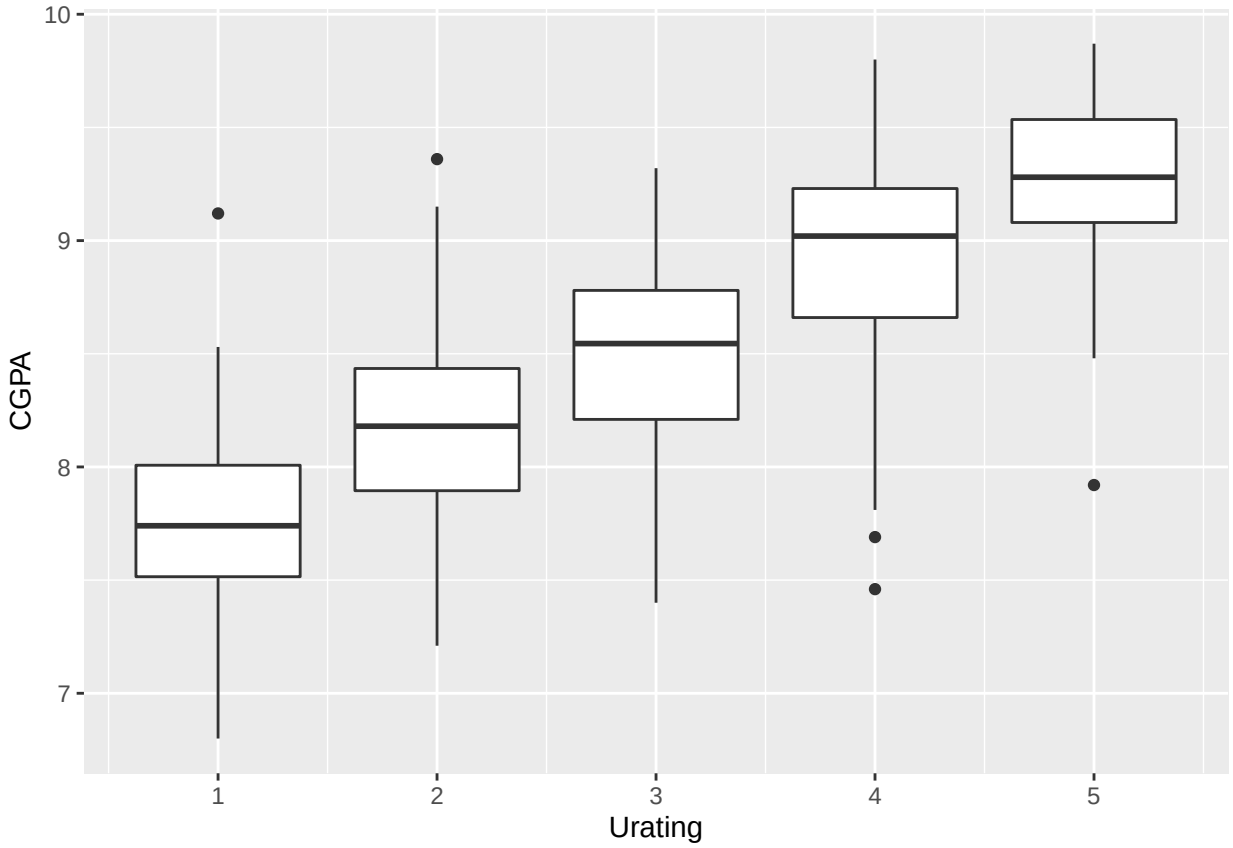


The median and intequartiles increase as the University Rating increases. There are also some outliers with

low GRE scores on Universities of rating 4 and 5.

A similar trend (median and intequartile) increases as the university ranking increases on the CGPA (Undergrad GPA) and TOEFL. The interquartile of the TOEFL for University rating 1 is narrower in comparison to the others but has a greater number of outliers.





We can evaluate what is the correlation of each one of the predictors and the admission

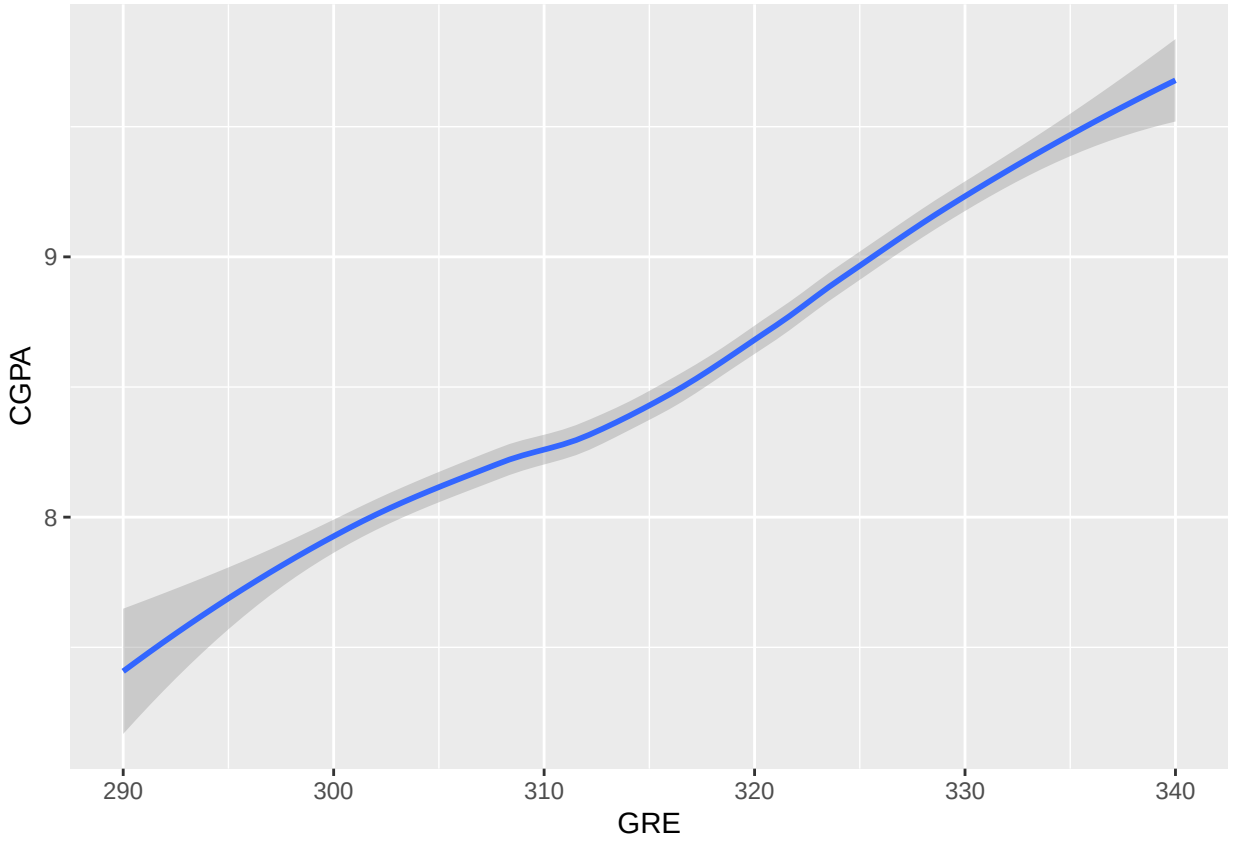
```
## [1] "correlation_GRE_admit"
## [1] 0.7007328
## [1] "correlation_TOEFL_admit"
## [1] 0.6445673
## [1] "correlation_CGPA_admit"
## [1] 0.7081798
## [1] "correlation_SOP_admit"
## [1] 0.5763698
## [1] "correlation_LOR_admit"
## [1] 0.5208935
## [1] "correlation_Research_admit"
## [1] 0.5095592
```

From these correlations, we can interpret that there is a strong correlation between the GRE, CGPA, and TOEFL scores with the students admitted. A moderate to week correlation exists with the SOP, LOR, and Research experience. However, does it make any predictor stronger than other? We will find out this later. Let's keep exploring the data.

Now, we can visualize how is the relationship among predictors and its correlation coefficient.

Let's evaluate the GRE vs CGPA

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



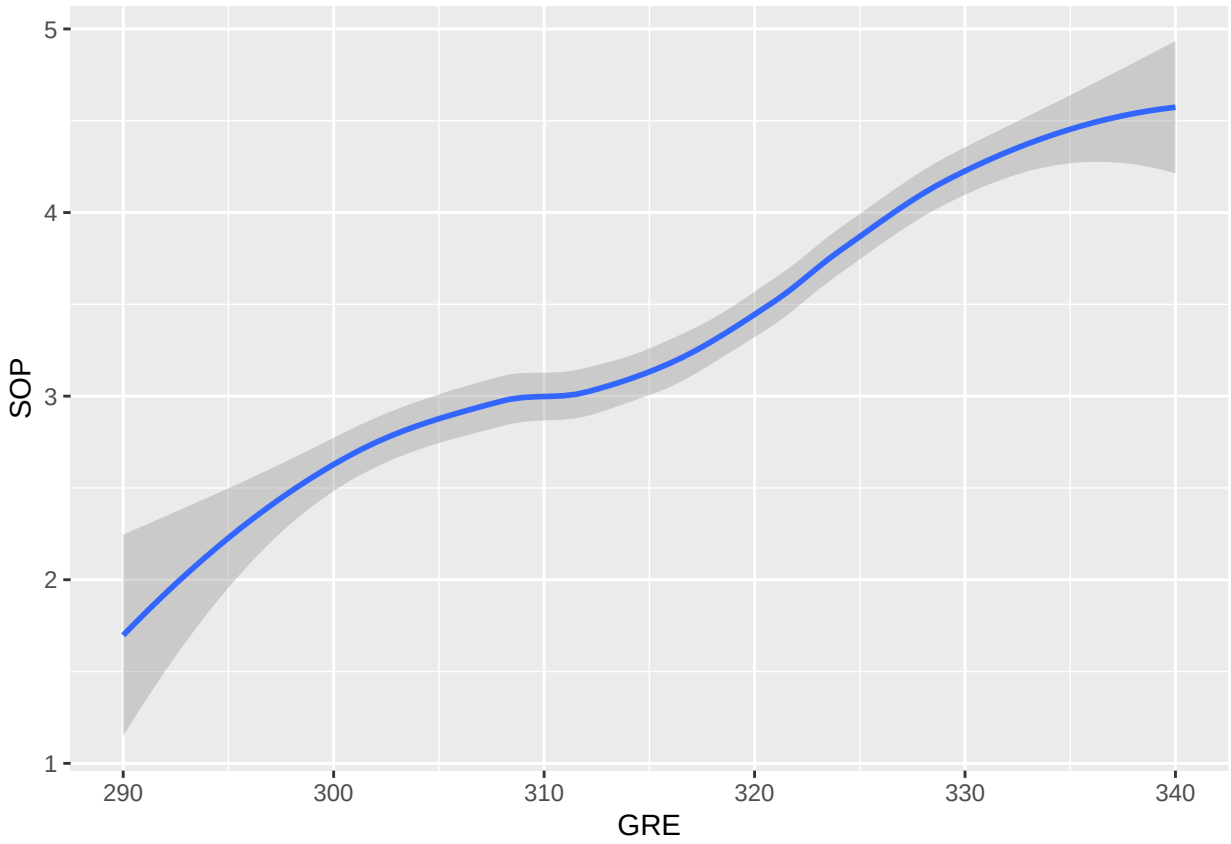

```
## [1] "correlation coefficient"
```

```
## [1] 0.8249545
```

This two variables are highly correlated, meaning that as a student obtained a high CGPA, the GRE score will be high, too.

Let's evaluate the GRE vs SOP

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



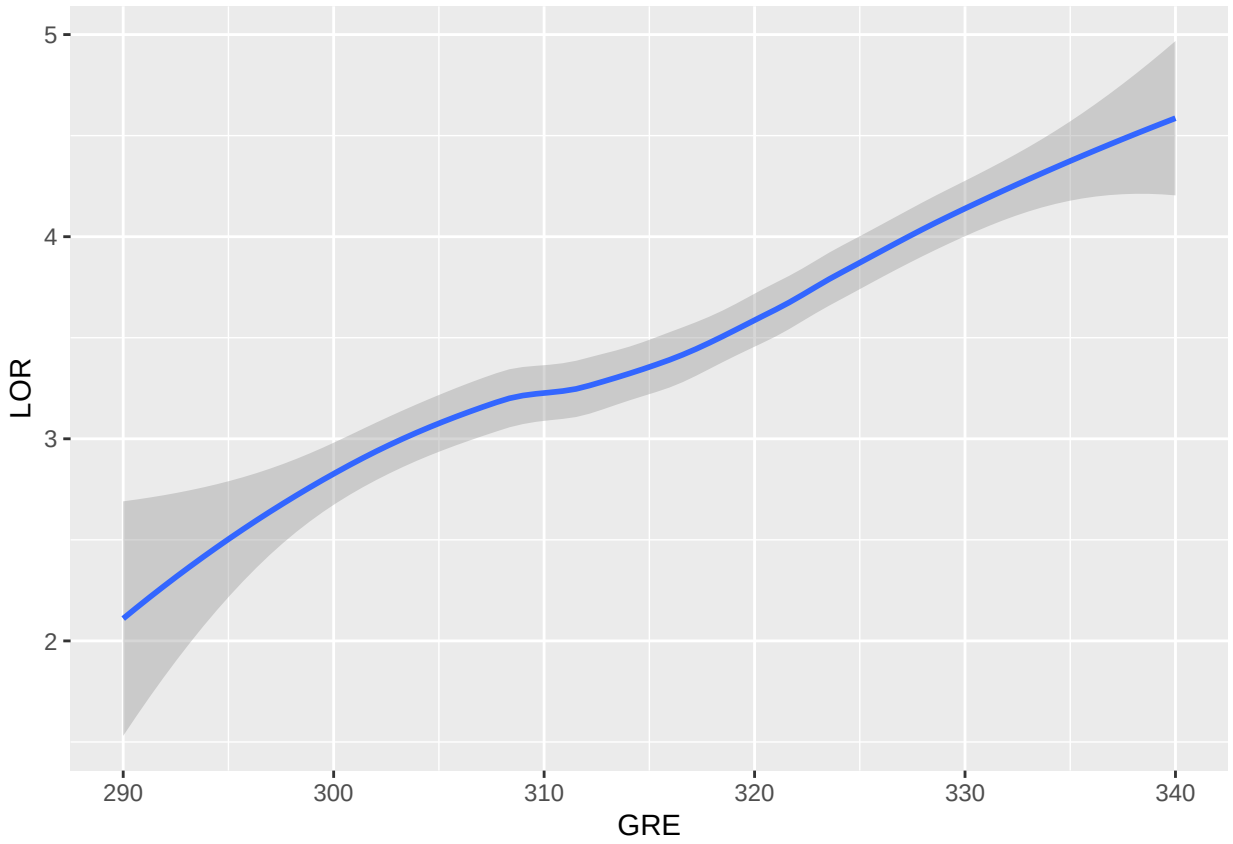

```
## [1] "correlation coefficient"
```

```
## [1] 0.6131186
```

This two variables have a moderate correlation.

Let's evaluate the GRE vs LOR

```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```




```
## [1] "correlation coefficient"
```

```
## [1] 0.5233955
```

This two variables have a moderate correlation.

To this point we can observe that some predictors are highly correlated on to each other, more than others. Also, based on the correlation coefficients and charts the GRE, CGPA, and TOEFL may be powerful predictors for admission prediction. Let's start applying some predicting methods with the training data.

Defining Overall Accuracy

The selecting parameter for determining the best model will be the Overall Accuracy. The overall accuracy is simply defined as the overall proportion of admitted that are predicted correctly. In other words, the train_set is trained to produce a model. Then, it is used to predict whether or not a student is admitted to the University on the test_set. Then, the overall accuracy is the proportion of properly predicted admitted/non admitted. The model that produces the highest accuracy will be then trained on the admission dataset and evaluated on the validation dataset.

Fit Models on train_set and test_set

After analyzing the data, there are some predictors that I want to evaluate first such as GRE, CGPA and TOEFL.

NAIVE MODEL

Test our luck here by estimating the admitted by randomly sampling

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] "accuracy_naive_model"
```

```
## [1] 0.1308642
```

The accuracy is pretty low. Not even half and half chances of picking up the right answer. Lets try other models then.

GRE MODEL

From the previous data exploration GRE seem to be highly correlated to the admission rate. Let's produce an estimation using only that predictor. However, GRE is not categorical, so which cutoffcan we use here? Let's use as cutoff the average GRE minus two standard deviations of the average obtained from those who were admitted. This parameterhas been selected by me, arbitrary. The ones above the cutoff are predicted to be admitted and those below do not.

```
## [1] "cutoff_GRE_admit"
```

```
## [1] 309.8008
```

```
## [1] "GRE_model_accuracy"
```

```
## [1] 0.7555556
```

The accuracy improved dramatically in comparison to the Naive model.

CGPA MODEL

The undergraduate GPA (CGPA) also presented a high correlation to the admission rate. The cutoff again will be the average CGPA minus two standard deviations, arbitrary, for those admitted. I anticipate an accuracy fairly good, too.

```
## [1] "cutoff_CGPA_admit"
```

```
## [1] 8.152525
```

```
## [1] "accuracy_CGPA_model"
```

```
## [1] 0.8
```

The accuracy improved slightly in comparison to the GRE model.

TOEFL MODEL

The TOEFL score alse presented a correlation with the students admitted.

```
## [1] "cutoff_TOEFL_admit"
```

```
## [1] 103.5473
```

```
## [1] "accuracy_TOEFL_model"
```

```
## [1] 0.8444444
```

Here, even though that the TOEFL did not have a high correlation coefficient as the CGPA and GRE, the accuracy here was even higher than the methods using these predictors independetly.

TOEFL_CGPA_GRE MODEL

Let's try a model that includes the predictors with highest correlation together (TOEFL, GRE, CGPA), considering again as the cutoff the mean of each predictor minus two times the standard deviation, for those admitted.

```
## [1] "accuracy_TOEFL_CGPA_GRE_mix_model"
```

```
## [1] 0.8888889
```

The use of the three predictors increased the general accuracy.

Let's summarize the accuracies of the methods evaluated so far:

```
## # A tibble: 5 x 2
##   model                     accuracy
##   <chr>                        <dbl>
## 1 naive_model                  0.131
## 2 GRE_model                    0.756
## 3 CGPA_model                   0.8
## 4 TOEFL_model                  0.844
## 5 TOEFL_CGPA_GRE_mix_model     0.889
```

CONFUSION MATRICES

Let's evaluate the sensitivity and specificity of the models depicted so far.

```
## [1] "Confusion Matrix GRE_model"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 12  1
##          1 10 22
##
##                Accuracy : 0.7556
##                  95% CI : (0.6046, 0.8712)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 0.000692
##
##                   Kappa : 0.5065
##
##  Mcnemar's Test P-Value : 0.015861
##
##             Sensitivity : 0.5455
##             Specificity : 0.9565
##          Pos Pred Value : 0.9231
##          Neg Pred Value : 0.6875
##              Prevalence : 0.4889
##          Detection Rate : 0.2667
##    Detection Prevalence : 0.2889
##       Balanced Accuracy : 0.7510
##
##        'Positive' Class : 0
##
```

```
## [1] "Confusion Matrix CGPA_model"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14  1
##          1  8 22
##
##                Accuracy : 0.8
```

```
##                  95% CI : (0.654, 0.9042)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 6.003e-05
##
##                   Kappa : 0.597
##
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.6364
##             Specificity : 0.9565
##          Pos Pred Value : 0.9333
##          Neg Pred Value : 0.7333
##              Prevalence : 0.4889
##          Detection Rate : 0.3111
##    Detection Prevalence : 0.3333
##       Balanced Accuracy : 0.7964
##
##        'Positive' Class : 0
##
```

```
## [1] "Confusion Matrix TOEFL_model"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 16  1
##          1  6 22
##
##                Accuracy : 0.8444
##                  95% CI : (0.7054, 0.9351)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 3.102e-06
##
##                   Kappa : 0.6872
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.7273
##             Specificity : 0.9565
##          Pos Pred Value : 0.9412
##          Neg Pred Value : 0.7857
##              Prevalence : 0.4889
##          Detection Rate : 0.3556
##    Detection Prevalence : 0.3778
##       Balanced Accuracy : 0.8419
##
##        'Positive' Class : 0
##
```

```
## [1] "Confusion Matrix TOEFL_CGPA_GRE_mix_model"
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 19  2
##          1  3 21
##
##                Accuracy : 0.8889
##                  95% CI : (0.7595, 0.9629)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 8.531e-08
##
##                   Kappa : 0.7774
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.8636
##             Specificity : 0.9130
##          Pos Pred Value : 0.9048
##          Neg Pred Value : 0.8750
##              Prevalence : 0.4889
##          Detection Rate : 0.4222
##    Detection Prevalence : 0.4667
##       Balanced Accuracy : 0.8883
##
##        'Positive' Class : 0
##
```

The highest balanced accuracy (0.889) and sensitivity (0.854) were obtained by mix_model, highest specificity is however higher on the other modules. In that sense, being 0 as the positive classs (non admitted) the sensitivity test the ability of a model to correctly identify those non admitted (true positive rate), whereas the specificity determines the ability of the model to correctly identify those admitted (true negative rate). In such a case, we can evaluate other models for better fit.

QUADRATIC DISCRIMINANT ANALYSIS (QDA)

We will use the caret package from this point on for simplicity and homogeneity in the code.

QDA_TOEFL

Let's evaluate only one variable, the TOEFL

```
# Let's try the TOEFL Score as the unique predictor
#set.seed(1)
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
fit_QDA_TOEFL <- train(factor(admitted) ~ TOEFL, method = "qda", data = train_set)
Admitted_QDA_TOEFL <- predict(fit_QDA_TOEFL, test_set)
accuracy_QDA_TOEFL <- mean(Admitted_QDA_TOEFL == factor(test_set$admitted))
print("Confusion Matrix TOEFL_CGPA_GRE_mix_model")
```

```
## [1] "Confusion Matrix TOEFL_CGPA_GRE_mix_model"
```

```
confusionMatrix(data = (Admitted_QDA_TOEFL), reference = factor(test_set$admitted))
```

```
## Confusion Matrix and Statistics
##
##           Reference
```

```
## Prediction  0  1
##          0 21  1
##          1  1 22
##
##                Accuracy : 0.9556
##                  95% CI : (0.8485, 0.9946)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 7.258e-11
##
##                   Kappa : 0.9111
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9545
##             Specificity : 0.9565
##          Pos Pred Value : 0.9545
##          Neg Pred Value : 0.9565
##              Prevalence : 0.4889
##          Detection Rate : 0.4667
##    Detection Prevalence : 0.4889
##       Balanced Accuracy : 0.9555
##
##        'Positive' Class : 0
##
```

The general accuracy, sensitivity and specificity are all very high and looks like a promising model based on the testing parameter. However, let's produce a quick inspection on the validation dataset to evaluate how well it performs.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.84
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 16  5
##          1  3 26
##
##                Accuracy : 0.84
##                  95% CI : (0.7089, 0.9283)
##     No Information Rate : 0.62
##     P-Value [Acc > NIR] : 0.0006247
##
##                   Kappa : 0.6672
##
##  Mcnemar's Test P-Value : 0.7236736
##
##             Sensitivity : 0.8421
##             Specificity : 0.8387
##          Pos Pred Value : 0.7619
##          Neg Pred Value : 0.8966
##              Prevalence : 0.3800
##          Detection Rate : 0.3200
```

```
##    Detection Prevalence : 0.4200
##       Balanced Accuracy : 0.8404
##
##        'Positive' Class : 0
##
```

The accuracy, sensitivity and specificity dropped considerable, suggesting some overfitting of the model. Let's keep evaluating other models in such a case.

QDA_MIX

Let's evaluate QDA with the predictors that presented higher correlation (TOEFL, GRE, CGPA)

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.8888889
```

A fairly good accuracy but not higher than QDA_TOEFL

QDA_ALL

Let's evaluate QDA with all the predictors from the dataset.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.8444444
```

Here the general accuracy decreased suggesting that other predictors may have affecting the model.

LINEAR DISCRIMINANT ANALYSIS

LDA_MIX

Since using all the predictors seems to be afecting the model, let's apply LDA only with the predictors with highest correlation.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.8888889
```

To this point, both LDA and QDA models yield a similar accuracy around 0.889. Not bad, but let's see if we can improve a bit.

GENERALIZED LINEAR MODEL

GLM_MIX

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.8888889
```

Very similar to the previous models.

GLM_ALL

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.9333333
```

In this particular case all the predictors seems to strenghten the model. Let's quickly evaluate it on the validation dataset.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
## [1] 0.86
```

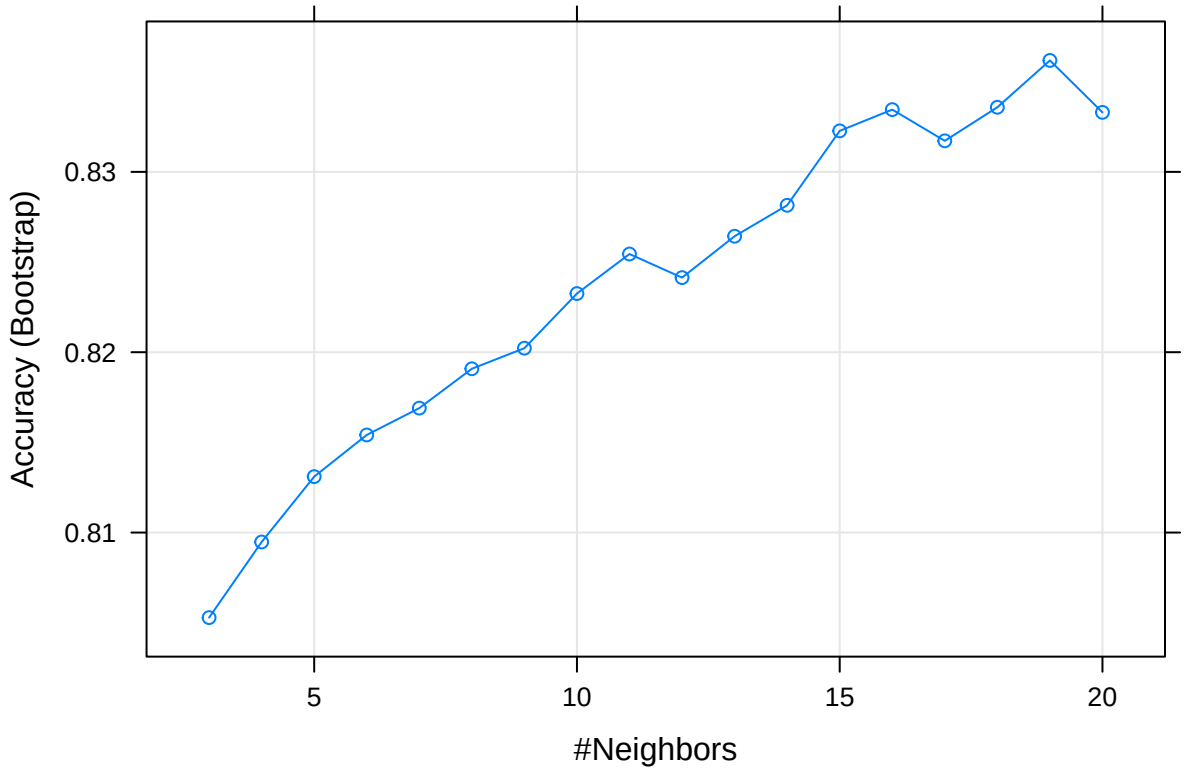
Here again, the accuracy on the validation decreased, perhaps due to some overfitting. Let's keep trying other models.

K-NEAREST NEIGHBOR

Let's apply KNN to a model that includes all the predictors.

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
##  Accuracy
## 0.8888889
```




```
##     k
## 17 19
```

KNN_CV

Finally, let's evaluate the crossvalidated KNN to see if we can improve the KNN model.

```
# kNN and Cross-Validation
#################################
#set.seed(1)
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

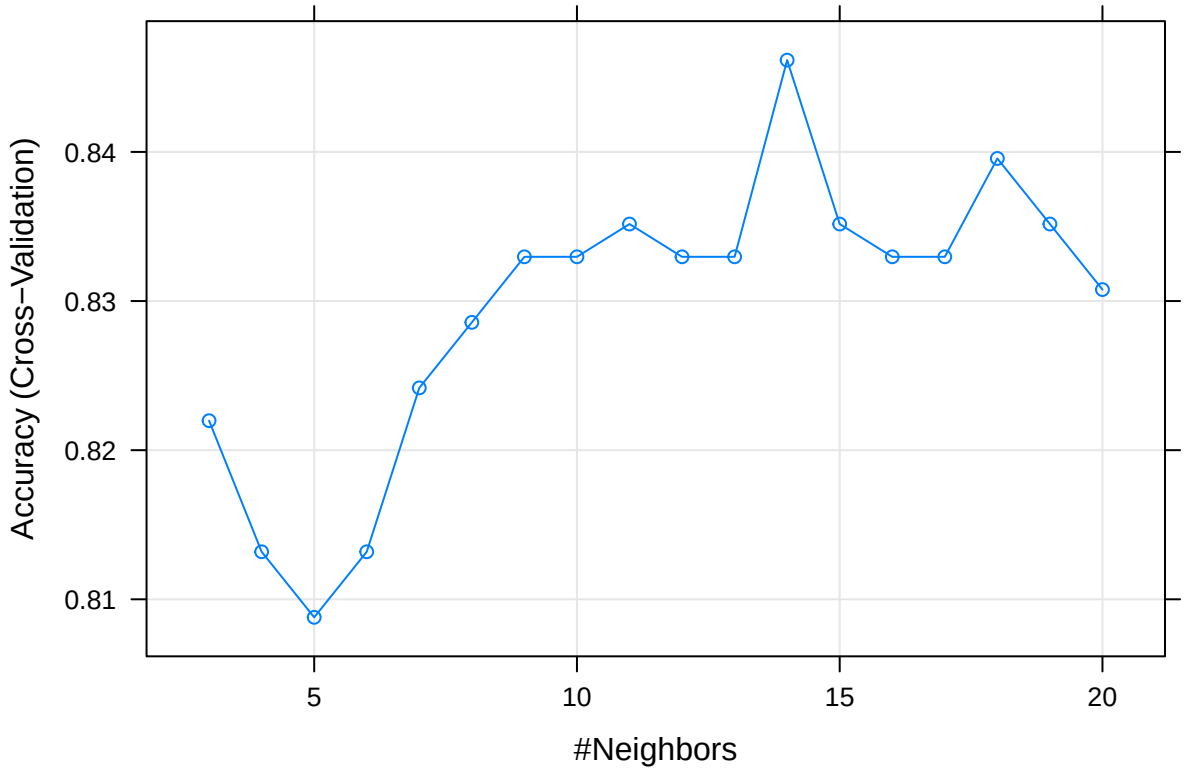
```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
```

```
## used
```

```
control <- trainControl(method = "cv", number = 5, p = .1)
fit_knn_cv <- train(factor(admitted) ~ GRE + TOEFL + CGPA + Research + SOP + LOR + Urating, method = "kn
                    data = train_set,
                    tuneGrid = data.frame(k = seq(3, 20, 1)),
                    trControl = control)
accuracy_knn_cv <- confusionMatrix(predict(fit_knn_cv, test_set), as_factor(test_set$admitted))$overall
accuracy_knn_cv
```

```
##  Accuracy
## 0.9333333
```

```
plot(fit_knn_cv)
```




```
best_k <- fit_knn_cv$bestTune
best_k
```

```
##     k
## 12 14
```

```
# the accuracy here seems to have improved.
```

The accuracy here seems to have improved.

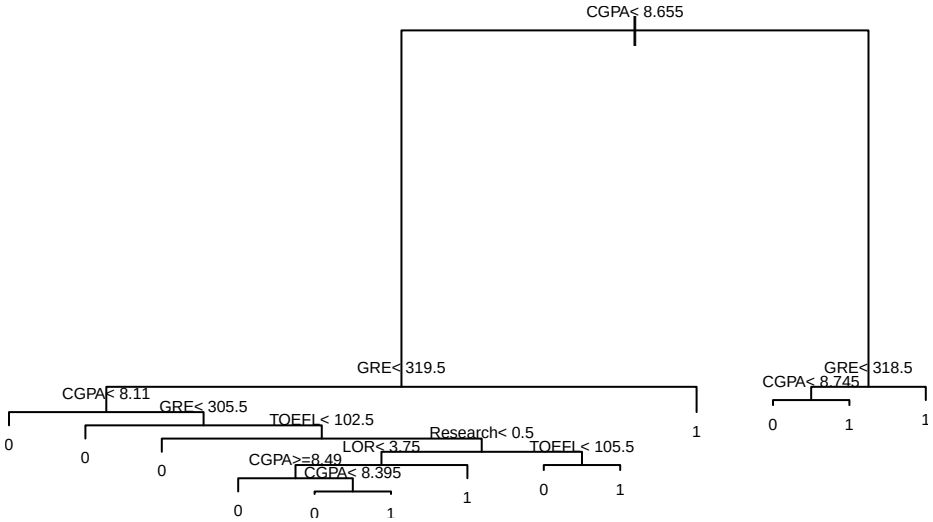
TREE CLASSIFICATION MODEL

Perhaps this tree may provide aN insight for potential graduate students to determine their possibilities of getting accepted to a particular university. Let's see:

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
##      cp
## 1 0.001
```

```
##  Accuracy
## 0.8888889
```



Although the accuracy of the model is not higher than kNN, the tree provides an insightful data for those applicants willing to apply and get accepted. Unfortunatelly, the tree does not show the rating of the university. One workaround here (not included) would be grouping by Urating and obtaining trees for each one of them. Thus, one student would have to pick the tree corresponding to the University Rating of their choice.

RANDOM FOREST

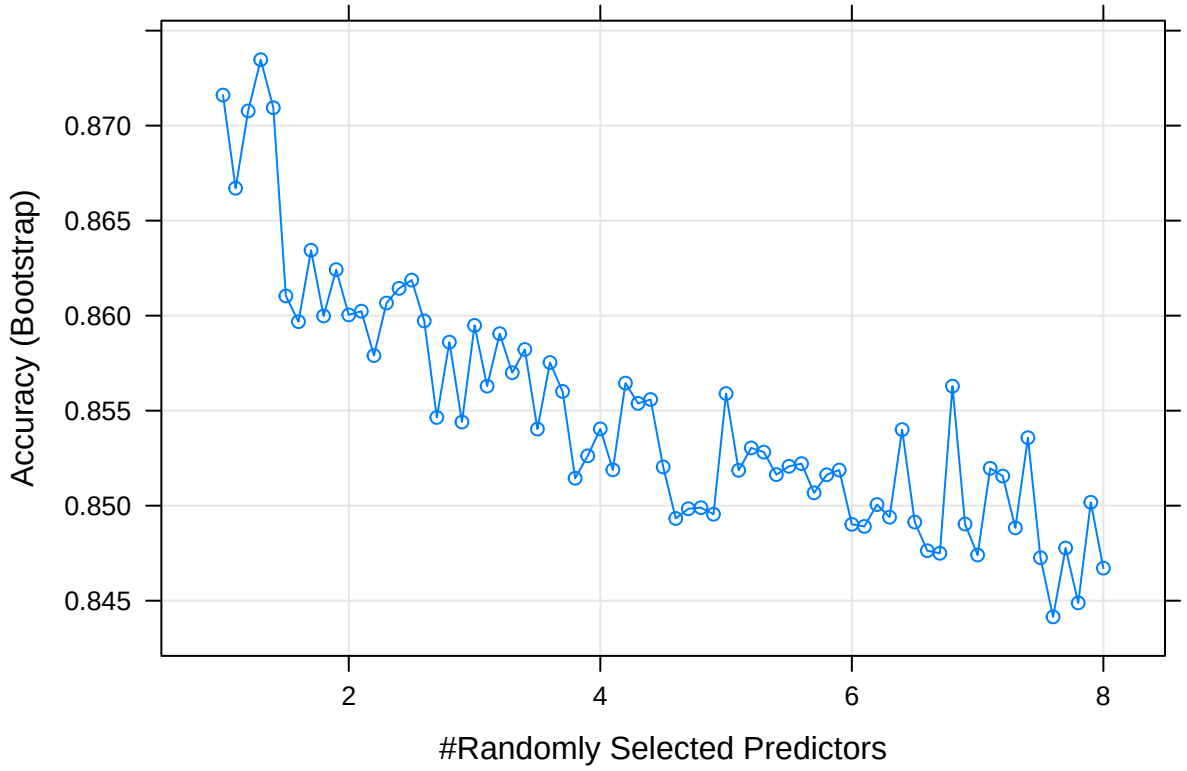
We apply Random Forest here to all the variables. We determine first the best tuning parameter (mtry). Then, the optimum number of trees that maximizes the accuracy. With these two parameters, the model is trained on the train_set and evaluated on the test_set.

```
# RANDOM FOREST
################################

# Select the best parameter of mtry
#set.seed(1)
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

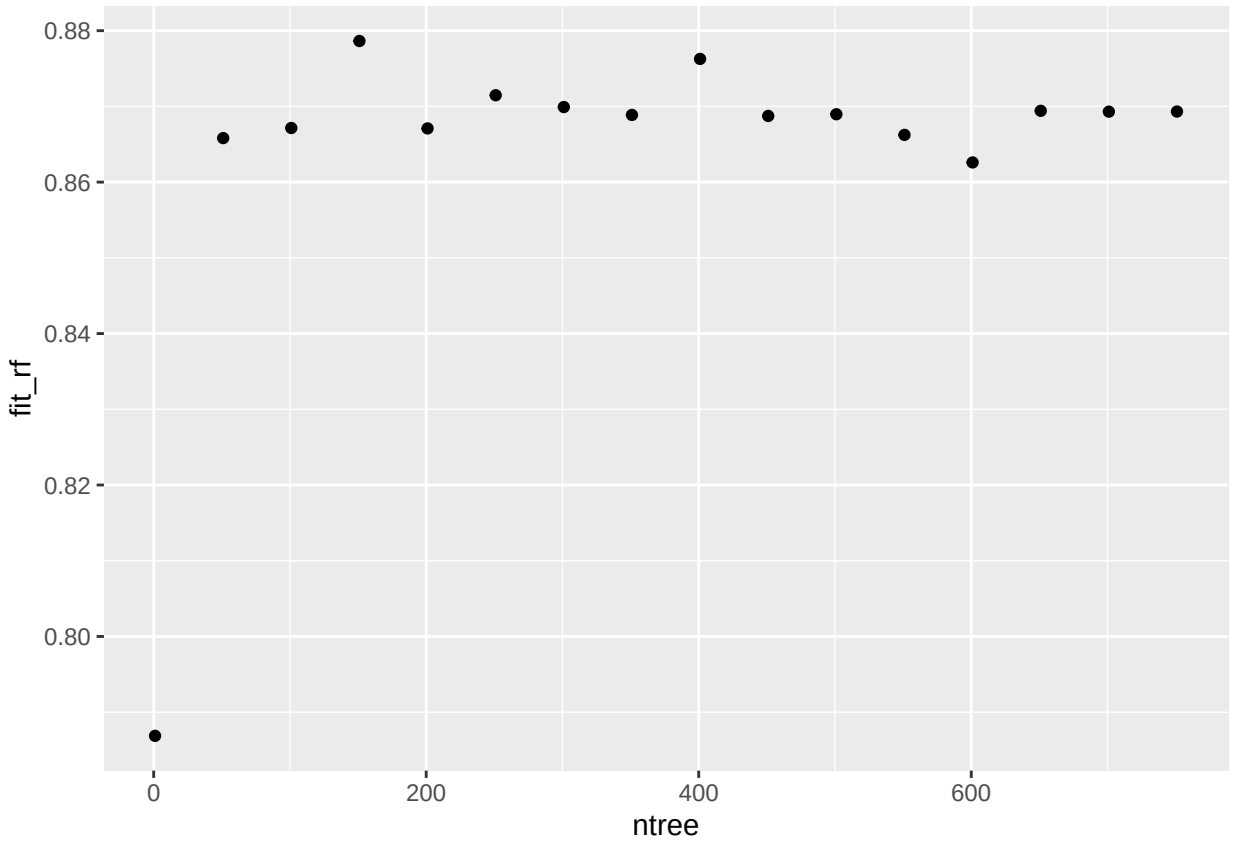
```
fit_rf <- train(as_factor(admitted) ~ ., method = "rf", data = train_set, tuneGrid = data.frame(mtry = s
#confusionMatrix(predict(fit_rf, test_set), as_factor(test_set$admitted))$overall["Accuracy"]
plot(fit_rf)
```




```
best_mtry <- fit_rf$bestTune
best_mtry
```

```
##   mtry
## 4  1.3
```

```
# Select the appropiate number of trees
ntree <- seq(1, 800, 50)
fit_rf <- sapply(ntree, function(n){
  train(as_factor(admitted) ~ ., method = "rf", data = train_set, tuneGrid = data.frame(mtry = best_mtry
})
qplot(ntree,fit_rf)
```




```
best_tree <- ntree[which.max(fit_rf)]
best_tree
```

```
## [1] 151
```

```
# Then, the final model will be with the best MTRY and best NTREE
fit_rf <- train(as_factor(admitted) ~ ., method = "rf", data = train_set, tuneGrid = data.frame(mtry = 1
accuracy_rf <- confusionMatrix(predict(fit_rf, test_set), as_factor(test_set$admitted))$overall["Accura
accuracy_rf
```

```
##  Accuracy
## 0.9111111
```

```
varImp(fit_rf)
```

```
## rf variable importance
##
##          Overall
## GRE      100.000
## CGPA      92.608
## TOEFL     62.858
## LOR       13.267
## SOP       12.024
## Research   8.652
## Urating    3.350
## Serial     0.000
```

Here the best mtry is 1.3, the optimum number of trees is 151. From the previous charts we can see that the ntree reach a point of stabilization and therefore we can infere that the number of trees selected is the

appropriate. With these parameters, the accuracy on the test_set es roughly 91%, a very good estimate. This is a potential candidate for final model.

RESULTS AND ANALYSIS

Let's summarize the accuracy of the methods depicted above.

```
## # A tibble: 15 x 2
##    model                     accuracy
##    <chr>                        <dbl>
##  1 naive_model                  0.131
##  2 GRE_model                    0.756
##  3 CGPA_model                   0.8
##  4 TOEFL_model                  0.844
##  5 TOEFL_CGPA_GRE_mix_model     0.889
##  6 QDA_TOEFL                    0.956
##  7 QDA_mix                      0.889
##  8 QDA_all                      0.844
##  9 LDA_all                      0.889
## 10 GLM_mix                      0.889
## 11 GLM_all                      0.86
## 12 kNN_all                      0.889
## 13 kNN_CV                       0.933
## 14 REGRESSION TREES             0.889
## 15 RANDOM FOREST                0.911
```

```
## [1] "***being all: using all predictors, being mix: using only GRE+CGPA+TOEFL"
```

The highest accuracies were found in QDA_TOEFL, kNN_CV and RANDOM FOREST.

Now, let's evaluate the model with the tuning parameters obtain on the training and test into the "Validation Dataset".

```
#################
#ANALYSIS
#################

# QDA_TOEFL ON VALIDATION
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
fit_QDA_TOEFL <- train(factor(admitted) ~ TOEFL, method = "qda", data = admit)
Admitted_QDA_TOEFL <- predict(fit_QDA_TOEFL, validation)
accuracy_QDA_TOEFL_validation <- mean(Admitted_QDA_TOEFL == factor(validation$admitted))
#The accuracy dropped notably.

# KNN_CV ON VALIDATION
#set.seed(1)
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
control <- trainControl(method = "cv", number = 5, p = .1)
fit_knn_cv <- train(factor(admitted) ~ GRE + TOEFL + CGPA + Research + SOP + LOR + Urating, method = "k
                    data = admit,
```

```
                    tuneGrid = data.frame(k = best_k),
                    trControl = control)
accuracy_kNN_CV_validation <- confusionMatrix(predict(fit_knn_cv, validation), as_factor(validation$admi
# The accuracy decreased slightly, probably due to some minor overfitting.

# RANDOM FOREST ON VALIDATION
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
```

```
## Warning in set.seed(1, sample.kind = "Rounding"): non-uniform 'Rounding' sampler
## used
```

```
fit_rf_final <- train(as_factor(admitted) ~ ., method = "rf", data = admit, tuneGrid = data.frame(mtry =
accuracy_RF_validation <- confusionMatrix(predict(fit_rf_final, validation), as_factor(validation$admit
varImp(fit_rf_final)
```

```
## rf variable importance
##
##          Overall
## CGPA     100.000
## GRE       87.531
## TOEFL     62.761
## SOP       33.604
## LOR       22.112
## Serial    13.635
## Urating    6.052
## Research   0.000
```

```
summary <- tibble(model = "QDA_TOEFL", accuracy = accuracy_QDA_TOEFL_validation)
summary <- bind_rows(summary, tibble(model = "kNN_CV", accuracy = accuracy_kNN_CV_validation))
summary <- bind_rows(summary, tibble(model = "RANDOM FOREST", accuracy = accuracy_RF_validation))
summary
```

```
## # A tibble: 3 x 2
##   model         accuracy
##   <chr>            <dbl>
## 1 QDA_TOEFL         0.84
## 2 kNN_CV            0.88
## 3 RANDOM FOREST     0.82
```

The accuracy is substancially lower with the validation set than that obtained with the train and test sets among the three methods.The validation accuracy is lower because perhaps I've made it artificially harder for the train_set to give the right asnwer that it ended up overfitting the training data.

Another possible reason might be that perhaps the validation set was not very representative in regard to the training set. In such case, a larger dataset would be useful to tune the models even more.

Finally, based on the overall accuracy obtained here, the best model is the cross-validated K-nearest neighbors with an accuracy on the validation of 0.88.

CONCLUSIONS, LIMITATION AND FUTURE WORK

Perhaps one the main flaws of the raw data is that the "Chance of Admission" column is a subjective value that were given by the prospective student in the poll. A way more interesting data would have been the actual prospective student that were admitted to the University. Indeed, a very powerful tool could be constructed from this data to guide prospective students to evaluate their chances to get accepted to their University of preference. In a similar fashion, the University Rating is rather subjective but is a data that could be collected in this regard as, for example, University rankings from well-known sources as "Best

National Universities – US News Rankings" including as a variable the University's acceptance ratio to the model proposed in this project.

The report included different methods tested to predict wheter a student is accepted to the Graduate School to a particular University based on different parameters required for admission such as GRE score, undergraduate GPA, TOEFL score, Statement of Purpose Strength, and Letter or Recomendation Strengths. The models tested including Linear Regression, Logistic Regression, Generative Models, Classification and Regression Trees and Random Forest.

The Overall accuracy defined as as the overall proportion of admitted that are predicted correctly was selected as the definying criteria for selecting the best model. From the results, cross-validated K-Nearest Neighbors yielded the highest accuracy of 0.88 tested on the validation dataset.

The accuracy was substancially lower with the validation set than that obtained with the train and test sets among the three methods.I can think of two possible reasons for this. One, the models were artificially harder on the training to give the right asnwer that it ended up overfitting the training data and not fitting in the same way the validation data. Alternatively, perhaps the validation set was not very representative in regard to the training set. In such case, a larger dataset would have been useful to tune the models even more.