GraduateAdmissions Project

Jose Aramendiz

3/1/2022

TABLE OF CONTENT

Introduction

The goal of the Graduate Admissions project is to create an algorithm to predict prospective students "admission rate" to the university, using a Graduate Admission Data set directly download from Kaggle. The final model was tested on the validation set as if the the admitted/not admitted population were unknown. More information and specifications about the data set can be found in the section "Information about the Data set". The criteria used to determine the performance on the diverse models tested was the "overall Accuracy".

To develop the prediction algorithm and avoid using the validation data set for training, an additional partition was created, dividing the data set in to different subsets: train_set & test_set. Diverse models were tested on these partitions and the accuracy recorded. The trained models with better accuracy were then retrain in the "admissions" data set and finally, evaluated against the validation data set. The validation subset is 10% of the original database and is not use for training purposes while creating the predictive model. Different predictive models such linear models, logistic regression models, K-nearest neighbors, Knn cross-validation, decision tree and random forest were evaluated.

This report goes through the different steps needed for data analysis, including a Data cleaning, Data exploration and visualization, and an analysis section evaluating each model used, and, finally a results and conclusion section with recommendation for future work.

NOTE TO THE GRADER: The code to elaborate this report is hidden. If you decide to take a look at the code, please refer to the .Rmd file or .R code.

https://github.com/josgor84/Graduate-Admissions-EDX

Thank you for your comments and feedback.

Information about the Data set

The Original Data set was downloaded from KAGGLE. Below you can find the link to the dataset

https://www.kaggle.com/mohansacharya/graduate-admissions?select=Admission_Predict_Ver1.1.csv

- Context of the data set: This data set is created for prediction of Graduate Admissions from an Indian perspective
- Citation

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

For the purpose of this project I uploaded the original data set to my github repository and in the code the data will be directly downloaded from there. I decided to do this since the graders will need to create an account in Kaggel to download directly from there. Thus, by doing this I make sure the graders will access the data set with no issues. There was no modification in the dataset.

Description of columns (parameters) found in the Data set

Serial No: A consecutive number assigned to each candidate GRE Score: GRE Scores (out of 340)
TOEFL Score: TOEFL Scores (out of 120) University Rating: University Rating (out of 5), 5 is consider a Top university SOP: Statement of Purpose (out of 5)
LOR: Letter of Recommendation (out of 5)
CGPA: Undergraduate GPA (out of 10)
Research: Research Experience (either 0 or 1)
Chance of Admit: Each candidate gave its own answer in the poll, either admitted or not

The data set has information about 500 prospective graduate students. Each student shared their GRE score, TOEFL score, SOP strength, LOR strength, undergrad GPA (CGPA), research experience (yes or no) and the rating of the University they applied. There is a final parameter "Chance of Admit", which is a personal opinion of each student. One can think about it as the chance that the student though to being admitted to the university he/she applied. Thus, is important to mention that the dataset does not include the "actual admission" result from the university. Nevertheless, the information is very valuable and useful as a guide for prospective students and their future decision to which university they can apply.

Since, there is no official admission from the universities. I applied a cutoff equal to the average of Chance of Admit to simulate admitted rate. Students above the cutoff will be admitted and students below not. Then, I removed the column "Chance of Admit" to avoid making it a predictor on the models and made admitted equal to 1 and not admitted equal to 0. The later will be included with mutate as a new column stating if the student was admitted or not when compared to the cutoff selected.

Finally, I selected this particular data set because it is very interesting to me and also because the usability rating reported in Kaggle is high.

Methods of Prediction

The method proposed is intended to maximize the overrall accuracy.

Data Exploration and Analysis

The first step of the data exploration is to determine the dimensions of the data sets. The dimension of the train_set is 455 rows and 9 columns. The test_set has 45 rows and 9 columns. Also, calculate the proportion of admitted students in each set, evaluate if there are NA values, and, study the behavior of the predictors (parameters) to check if there is a correlation between them and the results.

```
## # A tibble: 6 x 9
##   Serial GRE_Score TOEFL_Score University_Rating   SOP   LOR  CGPA Research
##    <dbl>     <dbl>       <dbl>             <dbl> <dbl> <dbl> <dbl>    <dbl>
```

```
## 1      1       337         118                  4   4.5   4.5  9.65        1
## 2      2       324         107                  4   4     4.5  8.87        1
## 3      3       316         104                  3   3     3.5  8           1
## 4      4       322         110                  3   3.5   2.5  8.67        1
## 5      5       314         103                  2   2     3    8.21        0
## 6      7       321         109                  3   3     4    8.2         1
## # ... with 1 more variable: admitted <dbl>
```

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

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

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

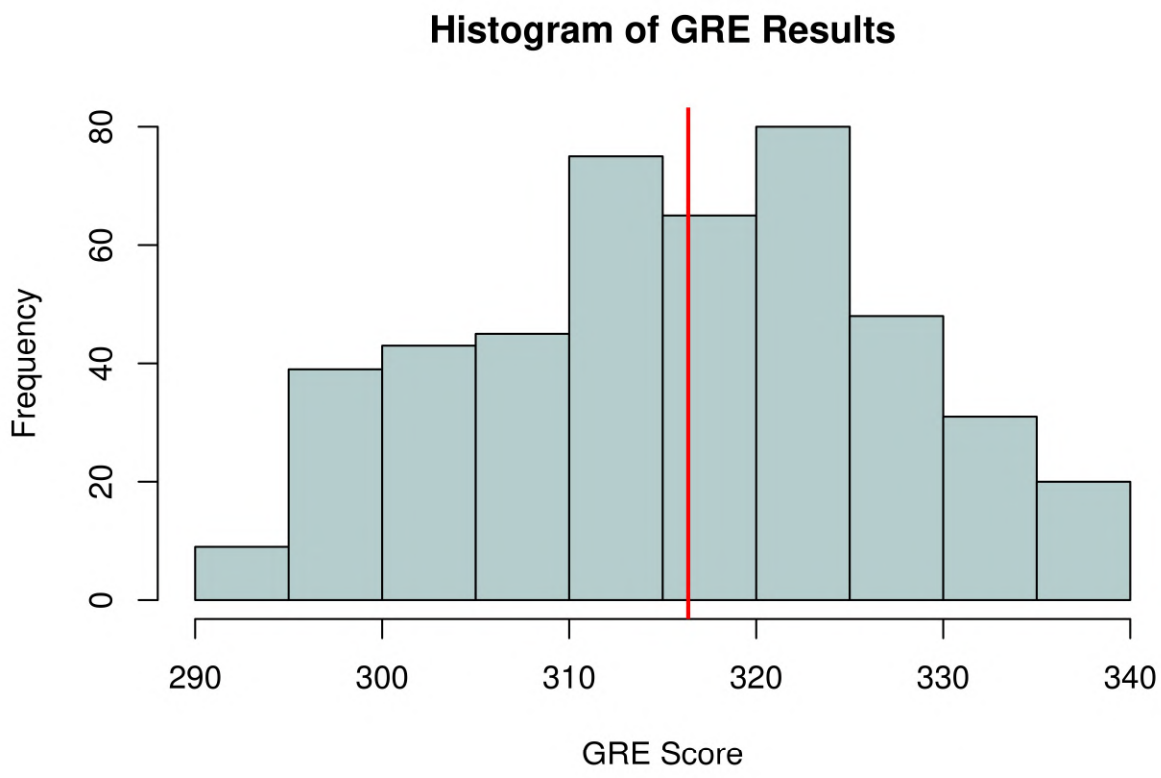
```
## [1] 0.5296703
```

```
## [1] 0
```

```
## [1] 0
```

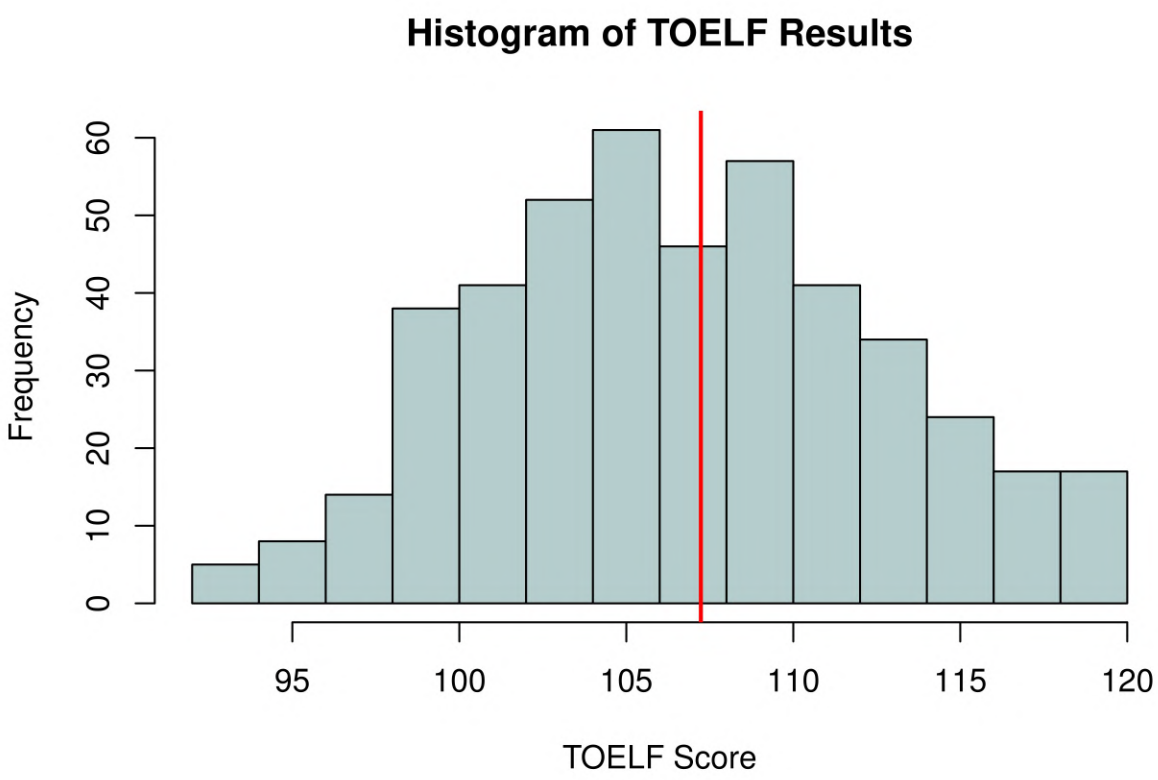
The average GRE is shown below as well as the histogram for this predictor, which seems to have bimodal dristribution

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



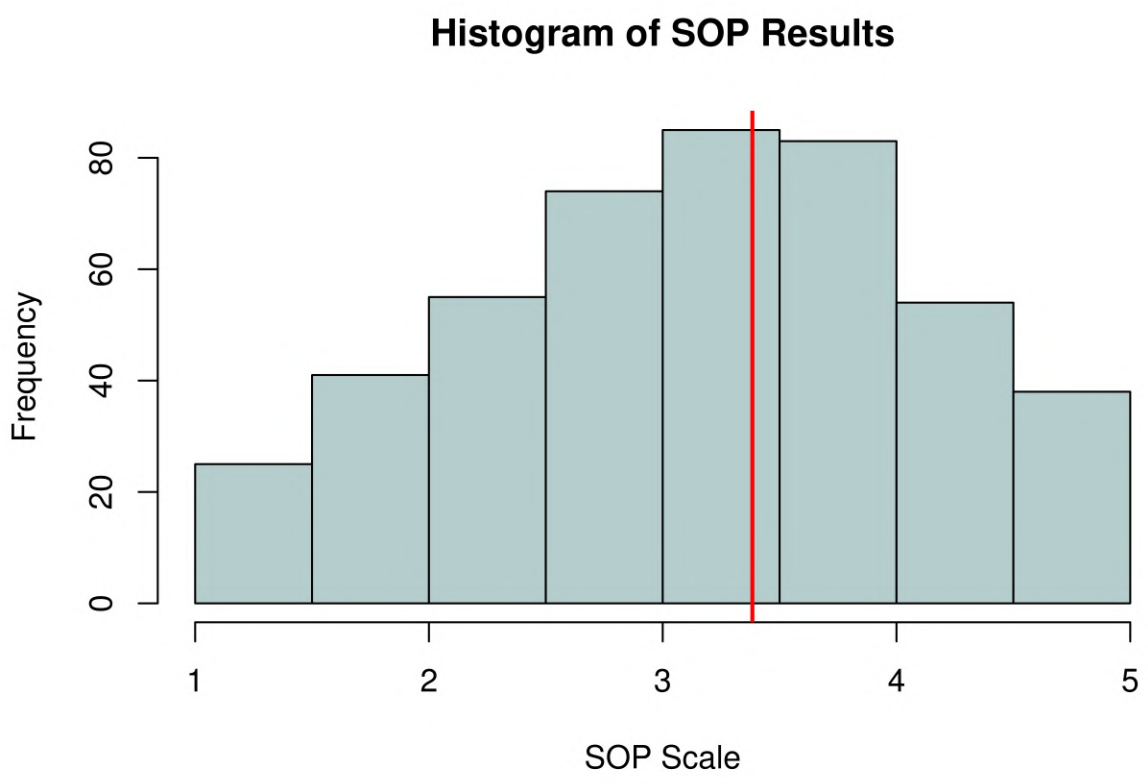
The TOEFL average is shown below as well as the histogram, which seems to have bimodal dristribution

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

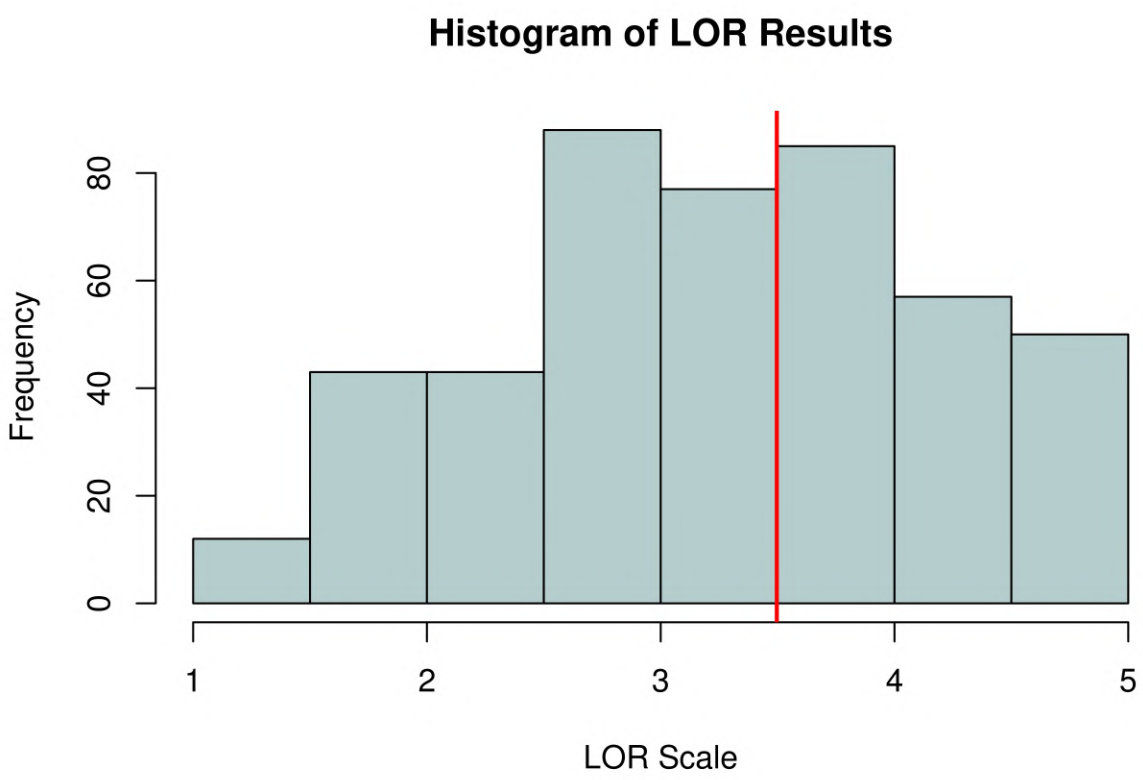


histogram of SOP, LOR, and, CGPA are shown below. SOP and CGPA seems to be skewed while LOR seems to have a bimodal distribution

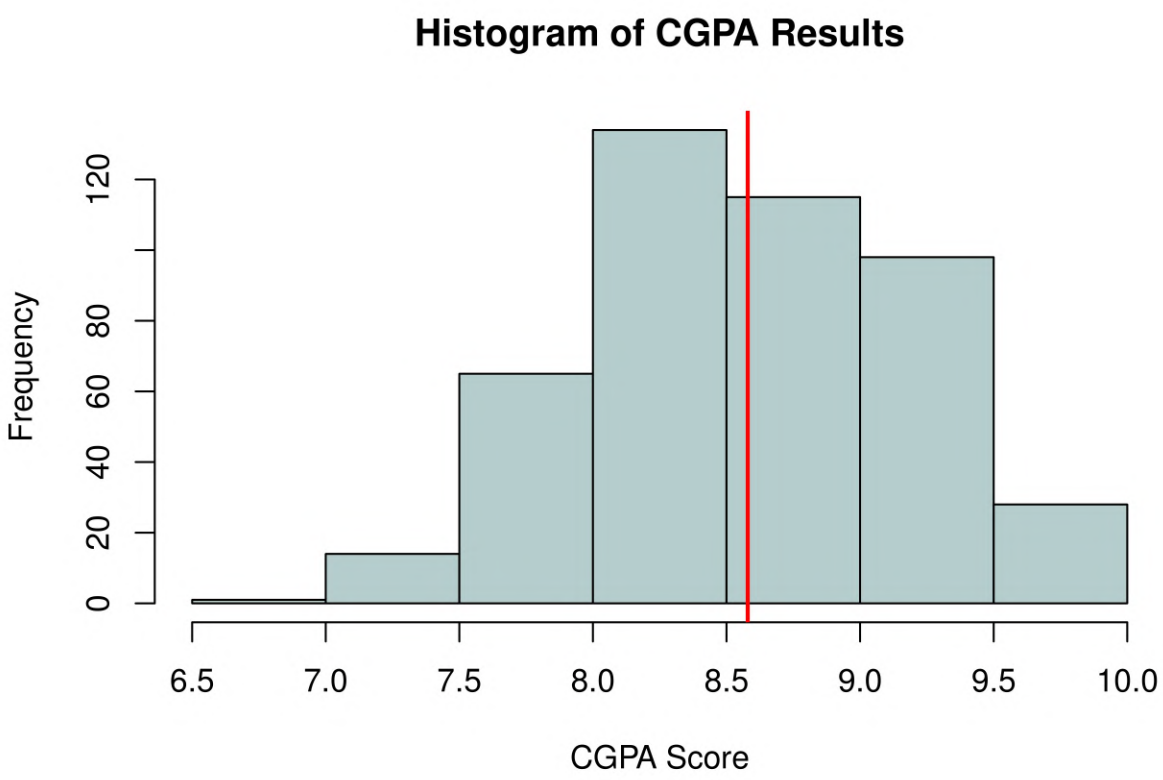
```
## [1] 3.383516
```


Histogram of SOP Results
Frequency
0
20
40
60
80
1
2
3
4
5
SOP Scale

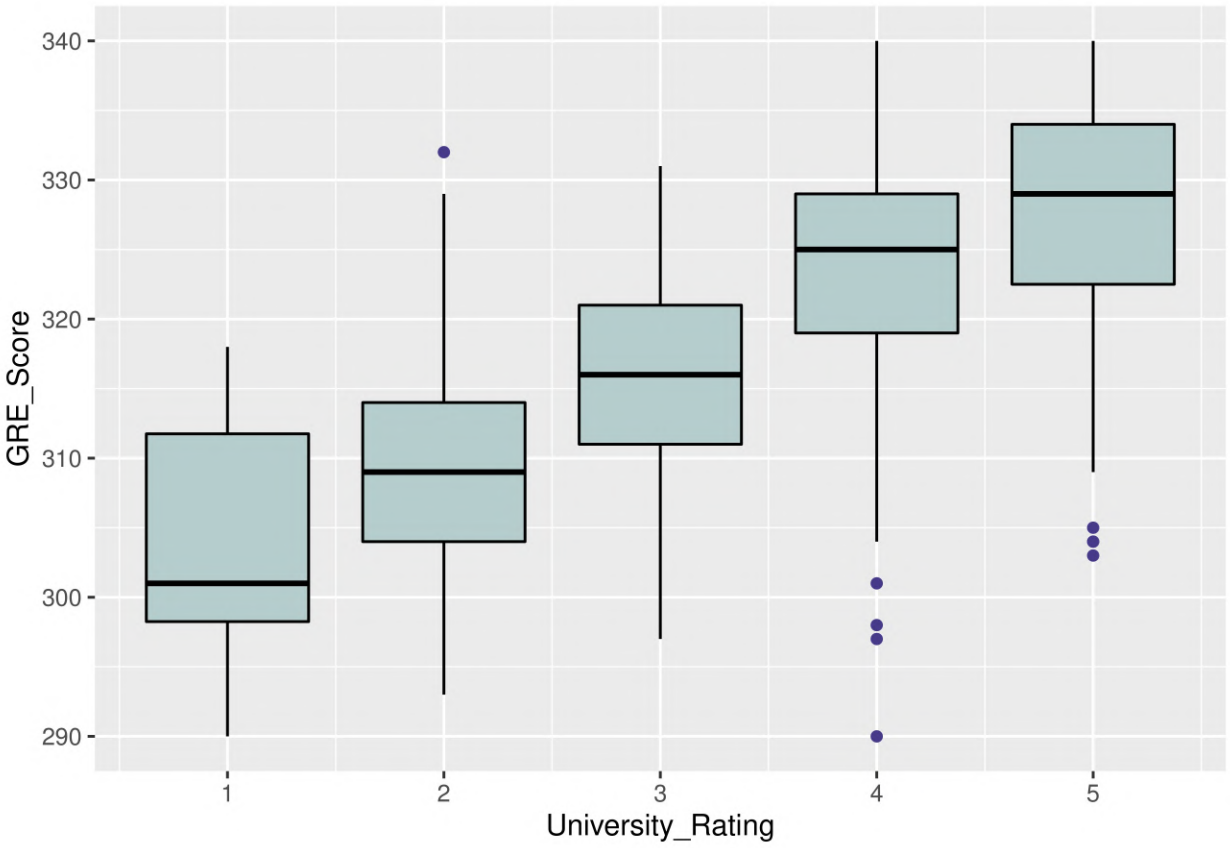
```
## [1] 3.496703
```

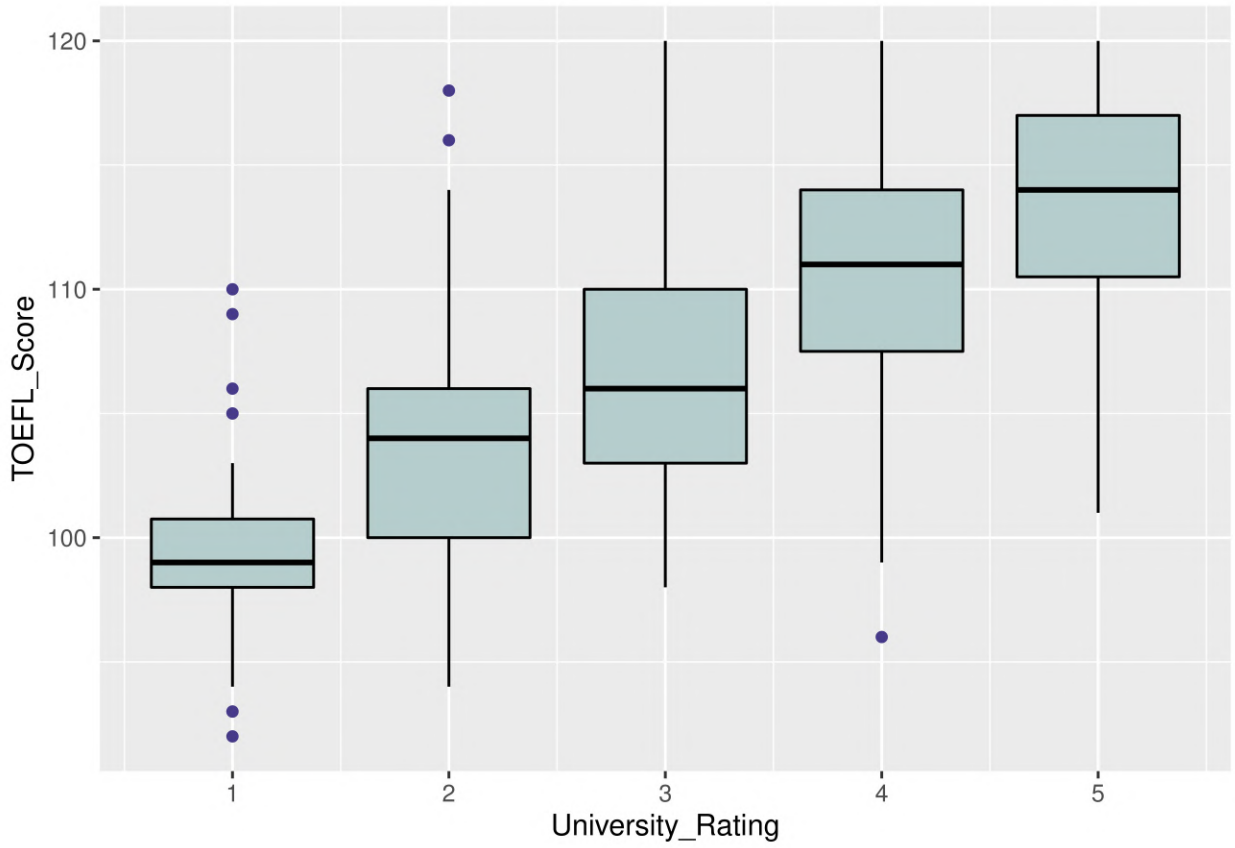

Histogram of LOR Results
Frequency
0
20
40
60
80
1
2
3
4
5
LOR Scale

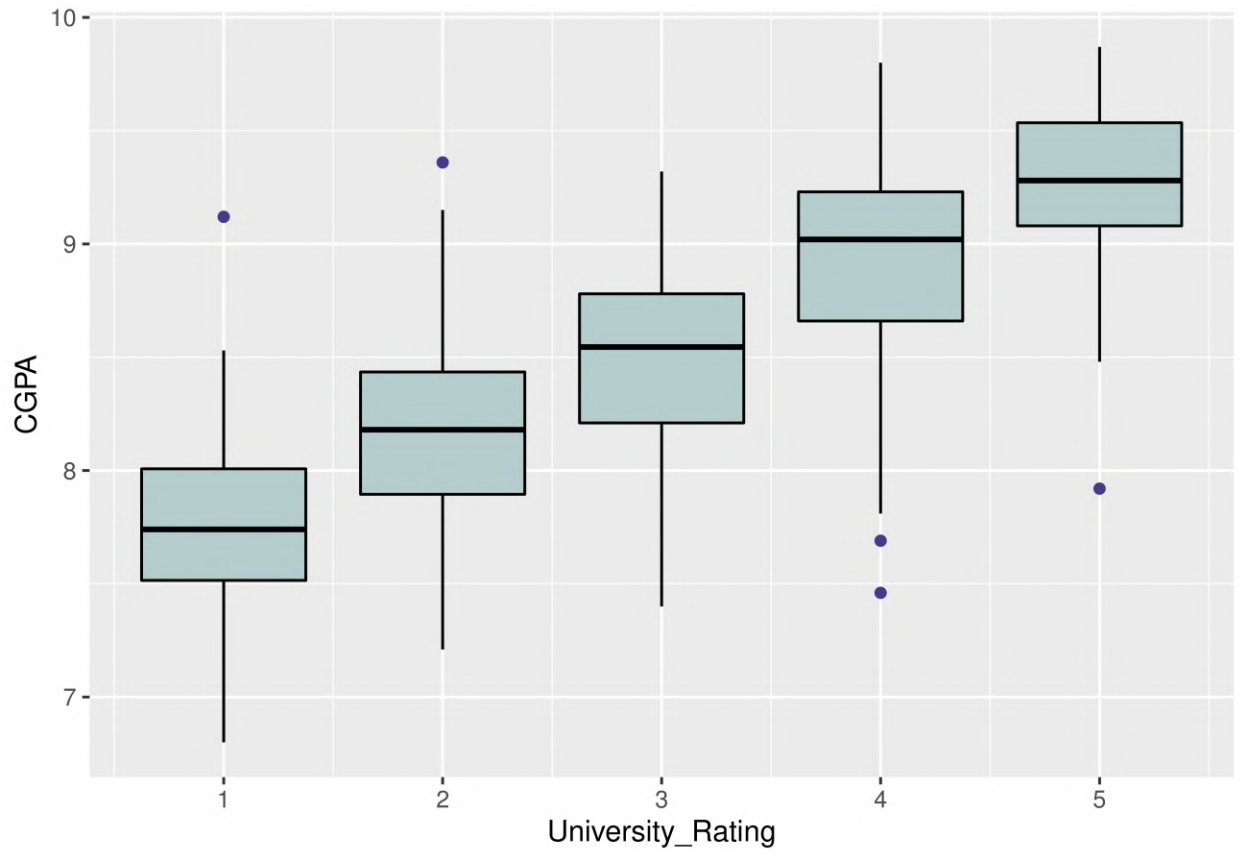
```
## [1] 8.579516
```



Also, we can observe the behavior of GRE, TOEFL and CGPA based on the rating of the university applied


340
330
320
310
300
290
GRE_Score
1
2
3
4
5
University_Rating


120
110
100
TOEFL_Score
1
2
3
4
5
University_Rating



From the charts we can see that GRE scores increases as we move up to top universities, which was expected. However, there are some outliners with low GRE Scores applied to Top universities (4,5).

In the case of TOELF, the median and interquartile increases with University rating. But, is interesting to observe some applicants (outliners) with higher scores applying to bottom ranking universities, where the interquartile is narrow compare to the others.

Finally in the case of CGPA the median and interquartile ranges are kind of uniform along the University rating. Few outliners with lower CGPA applying to top universities and two higher CGPA (outliners) applying to lower ranking universities

From the previous analysis, one can believe that GRE, TOEFL, and CGPA have a strong correlation to the population admitted (Admission rate). Thus, we can evaluate the correlation for each one of them.

The following table summarize the correlations

Correlation	Result
GRE to Admitted	0.7007328
TOELF to Admitted	0.6445673
CGPA to Admitted	0.7081798

The results support the previous statement of a strong correlation between GRE,TOELF and CGPA With Admission rate. Also, we can analize the correlation of the other parameters as shown below

Correlation	Result
SOP to Admitted	0.5763698

Correlation	Result
LOR to Admitted	0.5208935

Correlation	Result
Research to Admitted	0.5095592

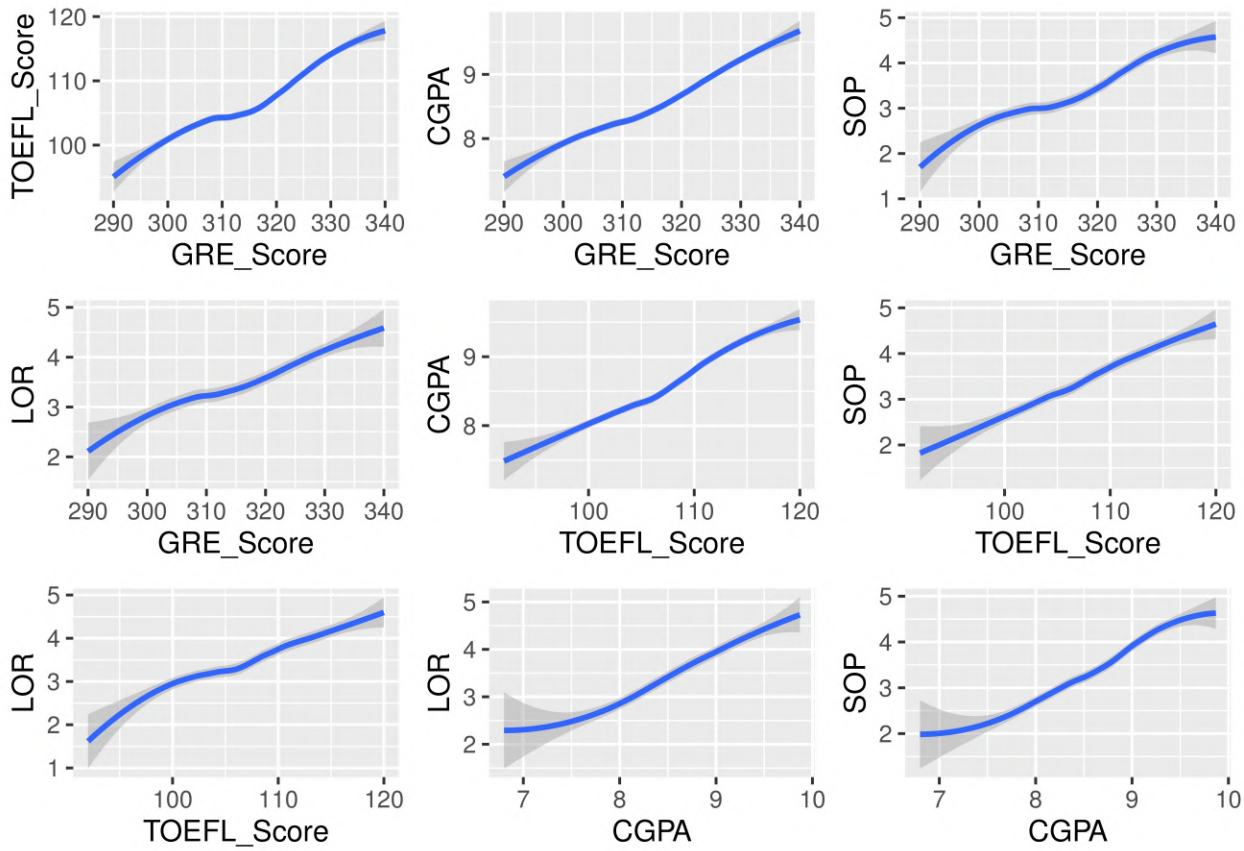
The following table summarize the correlations

Correlation	Result
GRE to Admitted	0.7007328
TOELF to Admitted	0.6445673
CGPA to Admitted	0.7081798
SOP to Admitted	0.5763698
LOR to Admitted	0.5208935
Research to Admitted	0.5095592

The results suggest there is a still correlation, but weaker compared to GRE,TOELF and CGPA.

We can analyze the relationship between predictors by creating plots and calculating the correlation coefficients to have a better understanding of how they are related to each other. The charts and table with the results for this purpose are shown below.

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



Correlation	Result
GRE and TOELF	0.8220323
GRE and CGPA	0.8249545
GRE and SOP	0.6131186
GRE and LOR	0.5233955
TOELF and SOP	0.6428439
TOELF and LOR	0.5399929
TOELF and CGPA	0.8081834
CGPA and SOP	0.8081834
CGPA and LOR	0.5399929

From these charts and correlations, we can conclude that there is a strong correlation between GRE, TOELF, and, CGPA with admission rate (students admitted). Also, we can interpret that even there is a correlation between SOP and LOR with the other parameters, these correlations tend to be moderate to week. As an example one can observe that GRE and CGPA are highly correlated, meaning that as a student obtained a high CGPA, the GRE score will be high, too. On the other hand, if we evaluate GRE and LOR the correlation is weaker compared to GRE and CGPA. Thus, GRE, CGPA, and TOEFL may serve as powerful predictors to model admission predictions.

Defining Overall Accuracy

To determine the best model in this project, Overall accuracy was used as the selecting parameter. Overall accuracy is the probability that an individual will be correctly classified by a test. For the purpose of this project, the train_set is use to train and produce a model, which later will be used as a predictor model in the test_set to determine if a student is admitted or not to the university. In this project the overall accuracy

can be understood as the proportion of admitted students that are predicted correctly. As a final step the model(s) with the highest accuracy will be trained one more time using the admission data set and then evaluated on the validation set.

Evaluate models on train and test sets

From the exploration and visualization section we observed that certain predictors(GRE,TOELF,CGPA) are highly correlated to the admission rate. Thus, they are evaluated first individually and then together.

Baseline Model

Simply guessing the admitted by randomly sampling and ignoring predictors

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

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

The result is not good. Actually, it is very low. just about 13% of chances guessing the right answer

MODEL_GRE: GRE As predictor

From the exploration of the original data we could observe that GRE seems to have a high correlation with the admitted population (Admission rate). This Model will use only GRE as predictor. Now, since GRE is not categorical, we need a cutoff.For this project I suggested as a cutoff the average of the GRE minus two standard deviations using only the GRE scores of those who were admitted. Thus, I need to filter a table with just GRE scores of the admitted population. This criteria was selected by me, arbitrary.

```
## [1] "cutoff_GRE_admitted"
```

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

```
## [1] "Model_GRE_accuracy"
```

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

The results shows a significant improvement compare to the baseline model. Nevertheless, there is still room for improvement, evaluating other predictors with high correlation and a mixture of them.

MODEL_TOELF: Toelf As predictor

When exploring the data TOEFL correlation was not at high as GRE, or CPGA. Still, it was the third higher among all parameters, so Let's see if the accuracy is lower as we may expect compare to GRE or not. here I used the same procedure used to establish the cutoff in the GRE model

```
## [1] "cutoff_TOEFL_admitted"
```

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

```
## [1] "Model_TOELF_accuracy"
```

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

The overall accuracy of the model was a surprise (about 84%), considering that TOELF had a lower correlation compare to GRE. Let´s evaluate the predictor with the highest correlation to the admitted population , CGPA

MODEL_CGPA: CGPA As predictor

```
## [1] "cutoff_CGPA_admitted"
```

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

```
## [1] "Model_CGPA_accuracy"
```

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

This is a slightly improvement compare to the GRE model. But, surprising it was just below the TOELF model. So far, we evaluate the three predictors individually. Now let´s evaluate all of them together (TOEFL, GRE, CGPA). One can expect better results since all of them showed a good correlation with the admitted population. The Cutoff will continue to be the same proposed in the previous models.

MODEL_GRE_TOEFL_CGPA three predictors

```
##########################################
## MODEL_GRE_TOEFL_CGPA three predictors  ##
##########################################

Model_GRE_TOEFL_CGPA <- ifelse(test_set$GRE_Score >= cutoff_GRE_admitted &
                                 test_set$TOEFL_Score  >= cutoff_TOEFL_admitted &
                                 test_set$CGPA>= cutoff_CGPA_admitted
                                   , 1, 0)
Model_GRE_TOEFL_CGPA_accuracy <- mean(Model_GRE_TOEFL_CGPA == test_set$admitted)
Model_GRE_TOEFL_CGPA_accuracy
```

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

The mix model improved the overall accuracy compare to models using individual predictor.

Below, the summary of the accuracies of the methods evaluated so far.

MODEL	Accuracy
Baseline	0.1308642
MODEL_GRE	0.7555556
MODEL_TOELF	0.8444444
MODEL_CGPA	0.8000000
MODEL_GRE_TOEFL_CGPA	0.8888889

Confusion Matrices and F1 results

Now, let's analyze the result of confusion matrix for each model proposed to study sensitivity and specificity

Definitions:

Sensitivity: Ability of an algorithm to predict a positive outcome when the actual outcome is positive.
Specificity: Ability of an algorithm to not predict a positive outcome when the actual outcome is not positive.

```
# Confusion Matrix GRE

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

```
## 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
##
```

```
# Confusion Matrix TOELF

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

```
## 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
##
```

```
# Confusion Matrix CGPA

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

```
## 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
##
```

```
# Confusion Matrix Mix Model

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

```
## 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
##
```

From the Results, it is possible to observe that the highest accuracy (0.8889) comes from the mix model as well as the highest sensitivity (0.8636). However, highest specificity is higher in the models with single predictors.

Let´s evaluate F1 scores

```
## [1] "F1_Model_GRE"
```

```
## [1] 0.6857143
```

```
## [1] "F1_Model_TOELF"
```

```
## [1] 0.8205128
```

```
## [1] "F1_Model_CGPA"
```

```
## [1] 0.7567568
```

```
## [1] "F1_Model_GRE_TOEFL_CGPA"
```

```
## [1] 0.8837209
```

The F1 score is maximum for the mix model. This model (Mix) performed very good. Still, we can evaluate other models to study if we can increase the overall accuracy.

METHOD_QDA QUADRATIC DISCRIMINANT ANALYSIS

From this point I will use the caret package to have a cleaner code while evaluating the models. Also, I will used the predictors with highest correlation (GRE,TOELF, and CGPA).

GRE Score as the unique predictor

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 17  2
##          1  5 21
##
##                Accuracy : 0.8444
##                  95% CI : (0.7054, 0.9351)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 3.102e-06
##
##                   Kappa : 0.6878
##
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.7727
##             Specificity : 0.9130
##          Pos Pred Value : 0.8947
##          Neg Pred Value : 0.8077
##              Prevalence : 0.4889
##          Detection Rate : 0.3778
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.8429
```

```
##
##        'Positive' Class : 0
##
```

TOELF Score as the unique predictor

```
## 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
##
```

CGPA Score as the unique predictor

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  4 22
##
##                Accuracy : 0.8889
##                  95% CI : (0.7595, 0.9629)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 8.531e-08
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.8182
##             Specificity : 0.9565
##          Pos Pred Value : 0.9474
```

```
##          Neg Pred Value : 0.8462
##              Prevalence : 0.4889
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.8874
##
##        'Positive' Class : 0
##
```

Surprisingly, TOELF continue to show the highest accuracy among the three best predictors. Let´s evaluate it with the validation set to check how it performs and if there is an over fitting or not.

```
set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Model_QDA_TOELF <- train(factor(admitted) ~ TOEFL_Score , method = "qda", data = admit)
QDA_TOELF_Admitted <- predict(Model_QDA_TOELF, validation)
mean(QDA_TOELF_Admitted == factor(validation$admitted))
```

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

```
confusionMatrix(data = (QDA_TOELF_Admitted), reference = factor(validation$admitted))
```

```
## 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 result suggest it may be an over fitting, since the overall accuracy drop from 0.956 to 0.84. Now, Let´s see what happen if we use the three predictors together (GRE, TOELF, CGPA)

GRE, TOELF, CGPA as predictors

```
## 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 overall accuracy is fairly good, but, QDA_TOELF still better. Let´s see what happen if we consider all the predictors in the QDA model

```
################################################
##  METHOD_QDA QUADRATIC DISCRIMINANT ANALYSIS ##
################################################

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Model_QDA_ALL <- train(factor(admitted) ~ GRE_Score + CGPA + TOEFL_Score + SOP + LOR + Research + Unive
QDA_ALL_Admitted<- predict(Model_QDA_ALL, test_set)
QDA_ALL_Accuracy <- mean(QDA_ALL_Admitted == factor(test_set$admitted))
confusionMatrix(data = (QDA_ALL_Admitted), reference = factor(test_set$admitted))
```

QDA model with all predictors

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 20  5
##          1  2 18
##
##                Accuracy : 0.8444
##                  95% CI : (0.7054, 0.9351)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 3.102e-06
##
##                   Kappa : 0.6897
##
```

```
##  Mcnemar's Test P-Value : 0.4497
##
##             Sensitivity : 0.9091
##             Specificity : 0.7826
##          Pos Pred Value : 0.8000
##          Neg Pred Value : 0.9000
##              Prevalence : 0.4889
##          Detection Rate : 0.4444
##    Detection Prevalence : 0.5556
##       Balanced Accuracy : 0.8458
##
##        'Positive' Class : 0
##
```

The overall accuracy drop when we consider all the predictors in the model, which suggest that some parameters may affect the model´s capability to predict. Thus, for future models i will consider only the three parameters whit highest correlation (GRE, TOELF, CGPA)

METHOD_LDA LINEAR DISCRIMINANT ANALYSIS

```
################################################
##  METHOD_LDA LINEAR DISCRIMINANT ANALYSIS    ##
################################################

## LDA with TOEFL, GRE, CGPA as predictors.

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Model_LDA_GRE_TOELF_CGPA <- train(factor(admitted) ~ GRE_Score + CGPA + TOEFL_Score, method = "lda", da
LDA_GRE_TOELF_CGPA_Admitted <- predict(Model_LDA_GRE_TOELF_CGPA, test_set)
LDA_GRE_TOELF_CGPA_Accuracy <- mean(LDA_GRE_TOELF_CGPA_Admitted == factor(test_set$admitted))
confusionMatrix(data = (LDA_GRE_TOELF_CGPA_Admitted), reference = factor(test_set$admitted))
```

LDA with TOEFL, GRE, CGPA as predictors.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  4 22
##
##                Accuracy : 0.8889
##                  95% CI : (0.7595, 0.9629)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 8.531e-08
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.8182
##             Specificity : 0.9565
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8462
##              Prevalence : 0.4889
```

```
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.8874
##
##        'Positive' Class : 0
##
```

There was no improvement compare to QDA. Actually, both models yields to the same overall accuracy (0.8889)

METHOD_GLM GENERALIZED LINEAR MODEL

```
########################################
##  METHOD_GLM GENERALIZED LINEAR MODEL ##
########################################

## GLM with TOEFL, GRE, CGPA as predictors.

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Model_GLM_GRE_TOELF_CGPA <- train(factor(admitted) ~ GRE_Score + CGPA + TOEFL_Score, method = "glm", da
GLM_GRE_TOELF_CGPA_Admitted <- predict(Model_GLM_GRE_TOELF_CGPA, test_set)
GLM_GRE_TOELF_CGPA_Accuracy <- mean(GLM_GRE_TOELF_CGPA_Admitted == factor(test_set$admitted))
confusionMatrix(data = (GLM_GRE_TOELF_CGPA_Admitted), reference = factor(test_set$admitted))
```

GLM with TOEFL, GRE, CGPA as predictors.

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 18  1
##          1  4 22
##
##                Accuracy : 0.8889
##                  95% CI : (0.7595, 0.9629)
##     No Information Rate : 0.5111
##     P-Value [Acc > NIR] : 8.531e-08
##
##                   Kappa : 0.777
##
##  Mcnemar's Test P-Value : 0.3711
##
##             Sensitivity : 0.8182
##             Specificity : 0.9565
##          Pos Pred Value : 0.9474
##          Neg Pred Value : 0.8462
##              Prevalence : 0.4889
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4222
##       Balanced Accuracy : 0.8874
##
##        'Positive' Class : 0
##
```

The results was very similar compare to the two previous models. there was no improvement. Let´s evaluate

other model such as kNN, kNN and Cross Validation, Classification (Decision) tree, and, random forests to check for potential improvements.

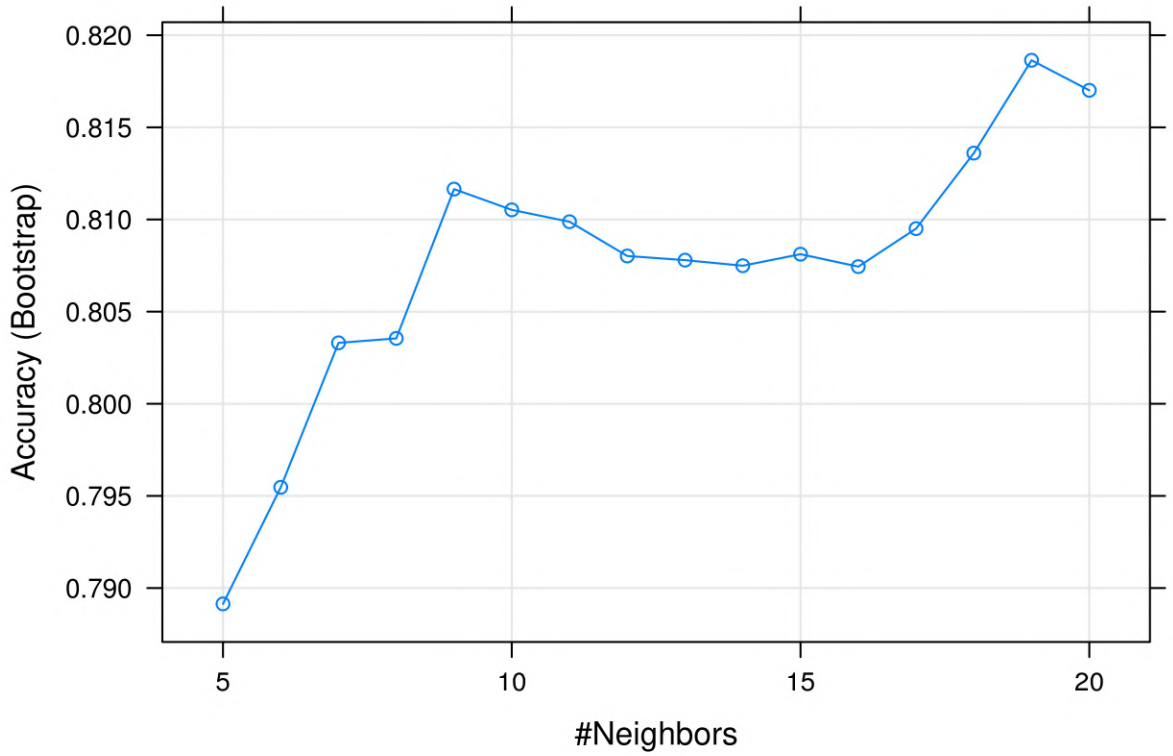
METHOD_KNN K- nearest neighbor

```
#####################################
##  METHOD_KNN K-nearest neighbor  ##
#####################################

## KNN with TOEFL, GRE, CGPA as predictors.

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Model_KNN_GRE_TOELF_CGPA <- train(factor(admitted) ~ GRE_Score + CGPA + TOEFL_Score, method = "knn", da
KNN_GRE_TOELF_CGPA_Accuracy <- confusionMatrix(predict(Model_KNN_GRE_TOELF_CGPA , test_set), as_factor(
KNN_GRE_TOELF_CGPA_Accuracy
```

KNN with TOEFL, GRE, CGPA as predictors.

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

```
plot(Model_KNN_GRE_TOELF_CGPA)
```




```
best_k <- Model_KNN_GRE_TOELF_CGPA$bestTune
best_k
```

```
##     k
```

```
## 15 19
```

The result indicates a good accuracy, but not better than the models previously proposed.

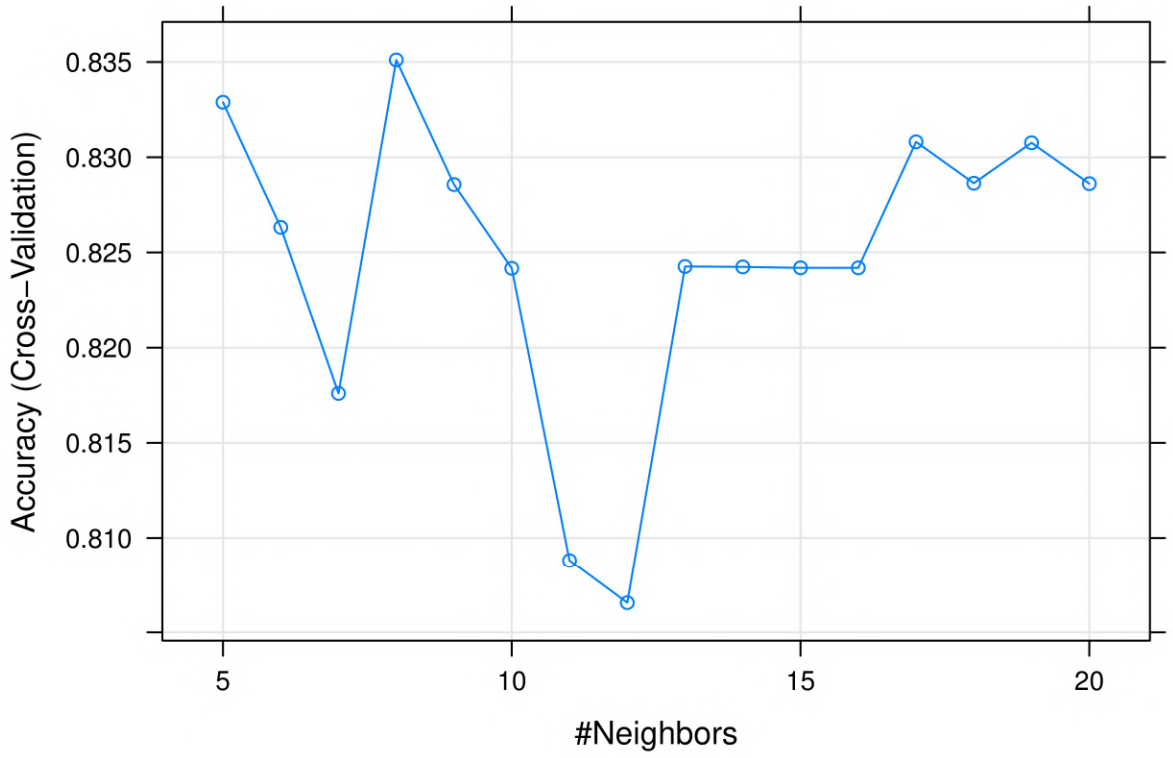
METHOD_KNN_CV Knn + Cross-Validation

```
##########################################
##  METHOD_KNN_CV Knn + Cross-Validation  ##
##########################################


set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
control <- trainControl(method = "cv", number = 5, p = .1)
Model_KNN_CV_GRE_TOELF_CGPA <- train(factor(admitted) ~ GRE_Score + TOEFL_Score + CGPA , method = "knn"
                    data = train_set,
                    tuneGrid = data.frame(k = seq(5, 20, 1)),
                    trControl = control)
KNN_CV_GRE_TOELF_CGPA_Accuracy <- confusionMatrix(predict(Model_KNN_CV_GRE_TOELF_CGPA, test_set), as_fa
KNN_CV_GRE_TOELF_CGPA_Accuracy
```

KNN_CV with TOEFL, GRE, CGPA as predictors

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

```
plot(Model_KNN_CV_GRE_TOELF_CGPA)
```



```r
best_k <- Model_KNN_CV_GRE_TOELF_CGPA$bestTune
best_k
```

```
##   k
## 4 8
```

The accuracy with Knn - cross-validation shows an improvement compared to Knn

METHOD_TREE Classification(Decision) Tree

Perhaps this tree may provide a insight for potential graduate students to determine their possibilities of getting accepted to a particular university. In this model I used all the predictors.

```r
################################################
##  METHOD_TREE Classification(Decision) Tree  ##
################################################

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Method_Tree_all <- train(factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + C
Method_Tree_all
```

```
## CART
##
## 455 samples
##   7 predictor
##   2 classes: '0', '1'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 455, 455, 455, 455, 455, 455, ...
## Resampling results across tuning parameters:
##
##   cp            Accuracy   Kappa
##   0.000000e+00  0.8137222  0.6252997
##   4.166667e-05  0.8137222  0.6252997
##   8.333333e-05  0.8137222  0.6252997
##   1.250000e-04  0.8137222  0.6252997
##   1.666667e-04  0.8137222  0.6252997
##   2.083333e-04  0.8137222  0.6252997
##   2.500000e-04  0.8137222  0.6252997
##   2.916667e-04  0.8137222  0.6252997
##   3.333333e-04  0.8137222  0.6252997
##   3.750000e-04  0.8137222  0.6252997
##   4.166667e-04  0.8137222  0.6252997
##   4.583333e-04  0.8137222  0.6252997
##   5.000000e-04  0.8137222  0.6252997
##   5.416667e-04  0.8137222  0.6252997
##   5.833333e-04  0.8137222  0.6252997
##   6.250000e-04  0.8137222  0.6252997
##   6.666667e-04  0.8137222  0.6252997
##   7.083333e-04  0.8137222  0.6252997
##   7.500000e-04  0.8137222  0.6252997
##   7.916667e-04  0.8137222  0.6252997
##   8.333333e-04  0.8137222  0.6252997
##   8.750000e-04  0.8137222  0.6252997
```

```
##   9.166667e-04  0.8137222  0.6252997
##   9.583333e-04  0.8137222  0.6252997
##   1.000000e-03  0.8137222  0.6252997
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.001.
```

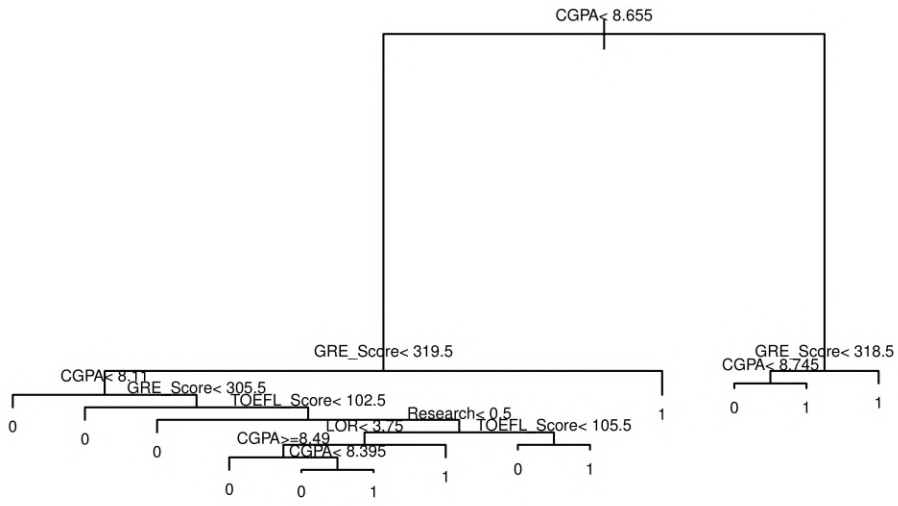
```
best_cp <- Method_Tree_all$bestTune
best_cp
```

```
##       cp
## 25 0.001
```

```
Method_tree_all_accuracy <- confusionMatrix(predict(Method_Tree_all, test_set), as_factor(test_set$admi
Method_tree_all_accuracy
```

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

```
plot(Method_Tree_all$finalModel, margin = 0.04)
text(Method_Tree_all$finalModel, cex = 0.5)
```



Although this model does not represent and improvement compared to previous models. it gives very insightful data to the applicants about where to apply and get accepted. Still, I did not group by university rating, which should give a good idea to potential applicants if the trees for each one can be explore. this step could be for future work.

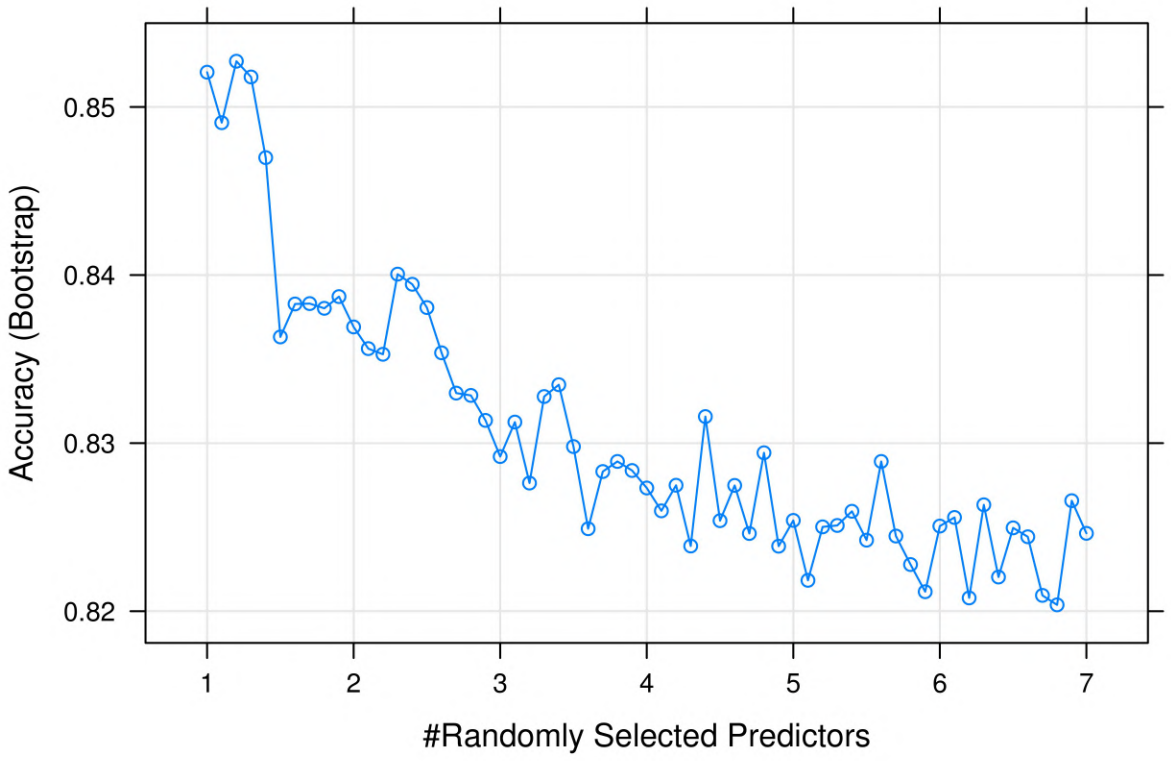
METHOD_RF Random Forest

As in KNN_CV I use all the parameters in this model.As a first step we need to determine the best tuning parameter (mtry) and, 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.

```
##############################
##  METHOD_RF Random Forest ##
##############################

# For this model I will use all the parameters

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Method_rf_all <- train(as_factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + C
confusionMatrix(predict(Method_rf_all, test_set), as_factor(test_set$admitted))$overall["Accuracy"]
```

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

```
plot(Method_rf_all)
```

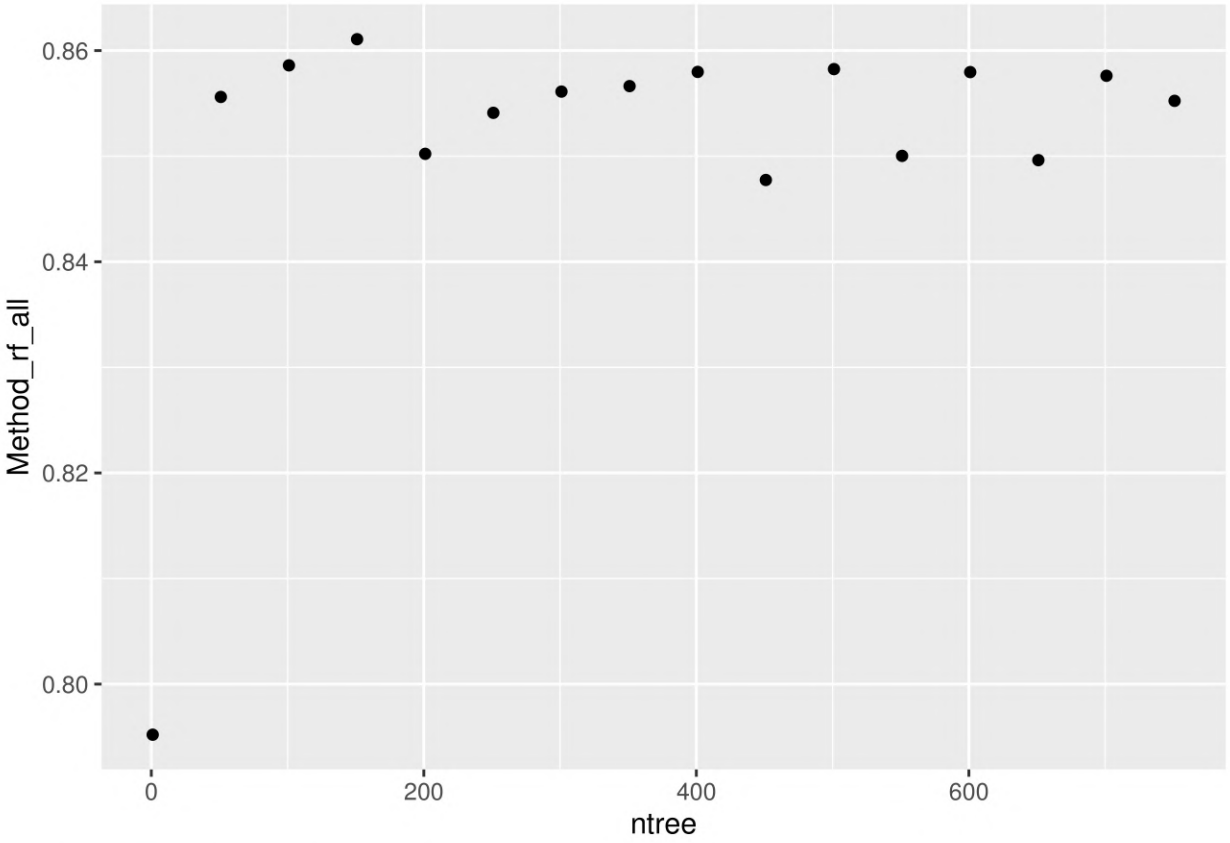



```
best_mtry <- Method_rf_all$bestTune
best_mtry
```

```
##   mtry
## 3  1.2
```

```
# Select the appropriate number of trees
ntree <- seq(1, 800, 50)
```

```
Method_rf_all <- sapply(ntree, function(n){
  train(as_factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + CGPA + Research
})
qplot(ntree,Method_rf_all)
```




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

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

```
# Then, the final model will be with the best MTRY and best NTREE
Method_rf_all <- train(as_factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + C
Method_rf_all_accuracy <- confusionMatrix(predict(Method_rf_all, test_set), as_factor(test_set$admitted)
varImp(Method_rf_all)
```

```
## rf variable importance
##
##                   Overall
## GRE_Score         100.000
## CGPA               74.471
## TOEFL_Score        57.360
## LOR                 8.803
## University_Rating   8.467
## SOP                 7.534
## Research            0.000
```

The results indicated that the best mtry is 1.2 and the optimum number of trees 151. the stabilization on the previous chart support this idea and the overall accuracy of the model makes it a potential candidate for

final model.

Results and Analysis Section

RESULTS - SUMMARY OF ACCURACIES ON TRAINED MODELS

MODEL	Accuracy
Baseline	0.1308642
MODEL_GRE	0.7555556
MODEL_TOELF	0.8444444
MODEL_CGPA	0.8000000
MODEL_GRE_TOEFL_CGPA	0.8888889
MODEL_QDA_GRE	0.8444444
MODEL_QDA_TOELF	0.9555556
MODEL_QDA_CGPA	0.8888889
MODEL_QDA_GRE+TOELF+CGPA	0.8888889
MODEL_QDA_ALL	0.8444444
MODEL_LDA_GRE+TOELF+CGPA	0.8888889
MODEL_GLM_GRE+TOELF+CGPA	0.8888889
MODEL_KNN_GRE+TOELF+CGPA	0.8888889
MODEL_KNN_CV_GRE+TOELF+CGPA	0.9111111
MODEL_TREE_ALL	0.8888889
MODEL_RF_ALL	0.9333333

- being All: using all predictors (GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + CGPA + Research)

The best results were obtained with the models: MODEL_QDA_TOELF,MODEL_KNN_CV_GRE+TOELF+CGPA and MODEL_RF_ALL. Also, it was very interesting to notice that most of the other models yielded to an accuracy of 0.889

The next step is to evaluate the best models generated on the training and test set into the Validation Data set as a final step.

Analysis on Validation Dataset

```
# MODEL_QDA_TOELF ON VALIDATION

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
VAL_QDA_TOEFL <- train(factor(admitted) ~ TOEFL_Score, method = "qda", data = admit)
Admitted_QDA_TOEFL <- predict(VAL_QDA_TOEFL, validation)
accuracy_QDA_TOEFL_validation <- mean(Admitted_QDA_TOEFL == factor(validation$admitted))
accuracy_QDA_TOEFL_validation
```

MODEL_QDA_TOELF ON VALIDATION

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

The accuracy dropped notably.

```
# MODEL_KNN_CV_GRE+TOELF+CGPA ON VALIDATION

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

```
control <- trainControl(method = "cv", number = 5, p = .1)
VAl_KNN_CV <- train(factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + CGPA +
                    data = admit,
                    tuneGrid = data.frame(k = best_k),
                    trControl = control)
accuracy_kNN_CV_validation <- confusionMatrix(predict(VAl_KNN_CV, validation), as_factor(validation$admi
accuracy_kNN_CV_validation
```

MODEL_KNN_CV_GRE+TOELF+CGPA ON VALIDATION

```
## Accuracy
##     0.86
```

The accuracy decreased slightly, probably due to some minor overfitting.

```
# MODEL_RF_ALL ON VALIDATION

set.seed(1, sample.kind="Rounding") # if using a later version than R 3.5
Val_rf_final <- train(as_factor(admitted) ~ GRE_Score + TOEFL_Score + University_Rating + SOP + LOR + C
accuracy_RF_validation <- confusionMatrix(predict(Val_rf_final, validation), as_factor(validation$admit
accuracy_RF_validation
```

MODEL_RF_ALL ON VALIDATION

```
## Accuracy
##     0.84
```

```
varImp(Val_rf_final)
```

```
## rf variable importance
##
##                   Overall
## GRE_Score         100.000
## CGPA               74.558
## TOEFL_Score        70.535
## University_Rating  41.075
## SOP                19.186
## LOR                 6.245
## Research            0.000
```

The accuracy dropped notably. Below the summary table of accuracies on the validation set

MODEL_VALIDATION	Accuracy
QDA_TOEFL	0.84
MODEL_KNN_CV	0.86
MODEL_RF_ALL	0.84

Final Analysis All three models experienced a reduction in their accuracy when evaluated in the validation set compared to the values obtained with the train and test set. The later, could be due to the conditions I did assume for the models when testing on the train and test set, that made it harder for them to yield to the correct answer and probably ended in an over fitting of the training data. On the other hand, another reason for the final results, could be that the validation set by its own was not representative when compared to the set used during training because of the population size. A larger population may lead to better results and tuning possibilities.

Among the three best models, K-nearest neighbors was the most stable when comparing accuracies result during training and validation. It only experience a slightly reduction (2.27%), from 0.88 (training) to 0.86 (Validation). The other two Models QDA_TOEFL and Random forest experienced a decrease of 11.57% and 9.67% respectively, and their final accuracy was 0.84 in both cases, which indicates that the K-nearest neighbors was the best model in this project.

Conclusion Section and Future Work

The project gives a good idea for prospective students when applying to universities. However, the data contains some flaws, such as the raw data in the column "Chance of Admission". The values in the later were given by the students themselves based on their thoughts and not by the actual student population that was indeed admitted to the university.

I believe that having access to the actual admitted students as well as let´s say University acceptance ratio from valid sources, could help create a powerful tool that can guide prospective students in their selection of universities and chances of admission. All this ideas could serve for future work.

This report included various methods studied during the course and were used to predict if a graduate student is admitted or not to the University. The parameters (predictors) used were GRE score, undergraduate GPA (CGPA), TOEFL score, Statement of Purpose Score (SOP), Letter or Recommendation Score (LOR), and research experience. Among the models tested were Linear Regression, Logistic Regression, Generative Models, Classification and Regression Trees and Random Forest.

The overall accuracy was the parameter used to select the best model. From the results, cross-validated K-Nearest Neighbors yielded the highest accuracy on the validation dataset, followed by QDA_TOEFL and Random forest.

The overall accuracy of the three models on the validation set experienced a reduction compared to the values obtained in the train_set and test_set. the later, could be because the population of the validation data was small and may not be a good representation in regard to the training set. Probably a larger population could help improve the tuning process and the final results. On the other hand, the conditions impose to the models on the training data made it harder to the training set to yield to the right answer, which may have caused and overfitting, and thus, affected the fitting on the validation set.

Note to the grader:

When running the code you will notice warnings, which corresponds to the setting the seeds() (random sampler generator) probably due to my version, i am not sure. But it does not affect the results. I did not show the warnings in the RDM file to make it easier to follow and interpret.