Rmarkdown1

Lucas Nobre

28 de janeiro de 2019

Predicting Graduate Admissions

About the dataset

This dataset is created for prediction of Graduate Admissions from an Indian perspective

Content

Contains several parameters which are considered important during applications for Masters programs.The parameters are: GRE Scores, TOEFL Scores, University Rating, Statement of Purpose and Letter Recommendation Strength, Undergraduate GPA, Research experience, Chance of Admit.

Inspiration

This dataset was built with the purpose of helping students in shortlisting universities with their profiles. The predicted output gives them a fair idea about their chances for a particular university.

Citation

Thanks 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 for providing the dataset.

Loading the libraries and dataset

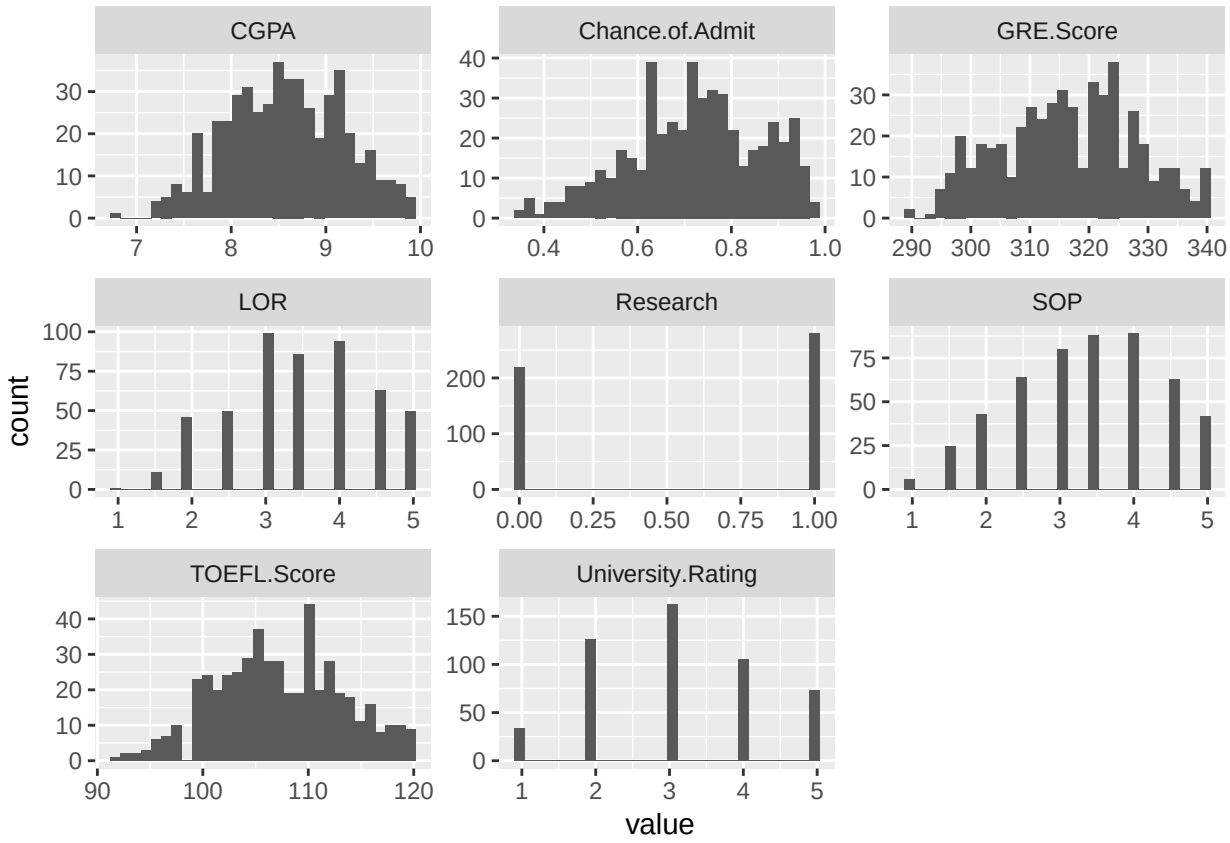
```
library(tidyverse)
library(caret)
library(e1071)
library(scales)
library(reshape2)
library(gridExtra)
data = read.csv('Admission_Predict.csv')
```

First impressions of the dataset

Let's take a look at our features and how they are distributed.

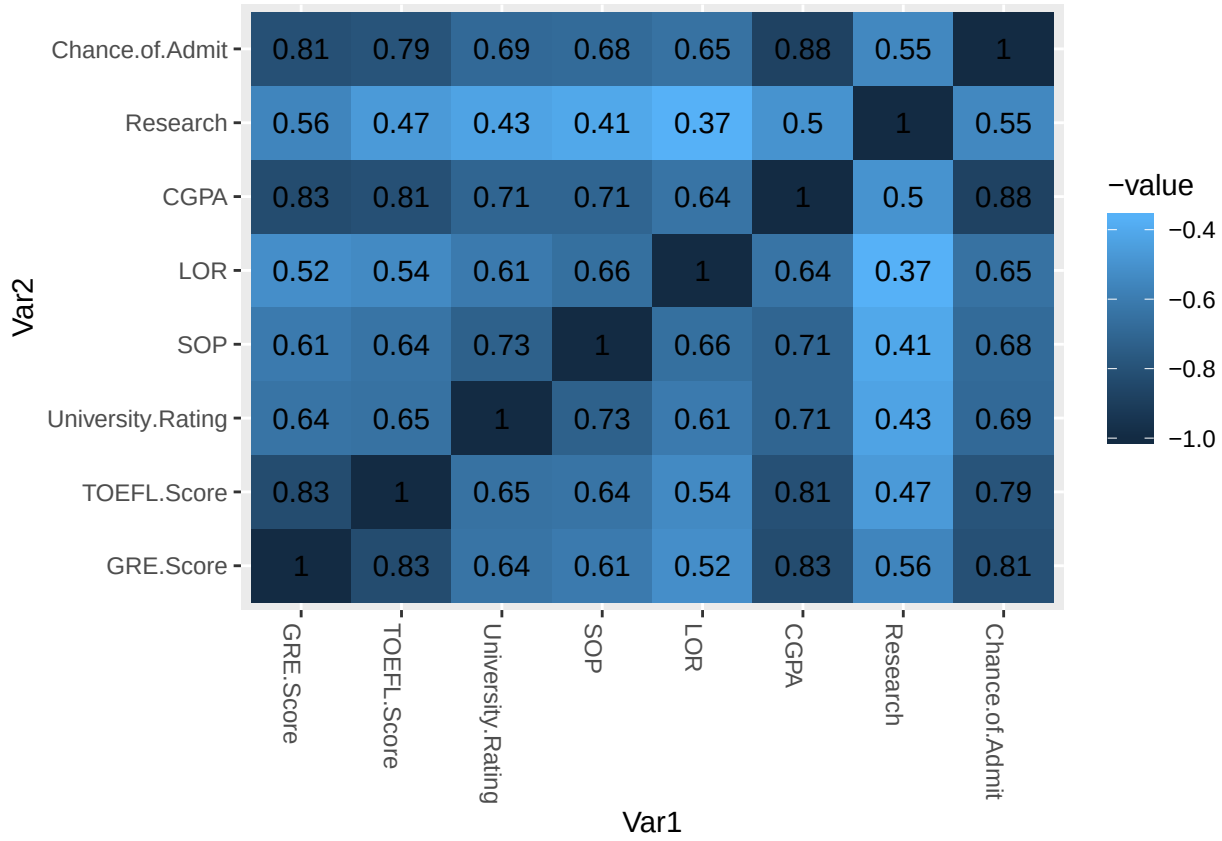
```
# Checking the histogram of each of the columns
data %>% select(-Serial.No.) %>% gather() %>% ggplot(aes(value)) + facet_wrap(~ key, scales = 'free') +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Correlation between the variables

```
correlation <- cor(data %>% select(-Serial.No.)) %>% round(2)
correlation %>% melt() %>% ggplot(aes(Var1, Var2, fill = -value)) + geom_tile() + geom_text(aes(label =
```



The most important features for the admission are CGPA, GRE Score and TOEFL Score.

Arranging the data

We can see that some of the features are discrete and others are cotinuous, but the continuous ones are not in the same scale. To address this we will scale them from 0 to 100. After that we will change the discrete features to factors, so we can start training our data.

```
new_data <- data %>%
  mutate(CGPA = rescale(CGPA, to = c(0, 100)),
         GRE.Score = rescale(GRE.Score, to = c(0,100)),
         TOEFL.Score = rescale(TOEFL.Score, to = c(0, 100))) %>%
  mutate(CGPA = CGPA - mean(CGPA), GRE.Score = GRE.Score - mean(GRE.Score),
         TOEFL.Score = TOEFL.Score - mean(TOEFL.Score),
         SOP = factor(SOP), University.Rating = factor(University.Rating),
         Research = factor(Research), LOR = factor(LOR)) %>% select(-Serial.No.)
```

Creating the training and test set

```
test_index <- createDataPartition(data$Chance.of.Admit, times = 1, p = 0.2, list = F)
test_set <- new_data[test_index,]
train_set <- new_data[-test_index,]
```

Training models

We change the chance to admit to 0 or 1, 1 if the chance to admit is highter than 0.8, else it is 0. We do this so we can test the models for accuracy

```
factor_train_set <- train_set %>%
  mutate(Chance.of.Admit = factor(ifelse(Chance.of.Admit < 0.8, 0, 1)))
factor_test_set <- test_set %>%
  mutate(Chance.of.Admit = factor(ifelse(Chance.of.Admit < 0.8, 0, 1)))
```

GLM

Our first model will be GLM

```
glm_train <- train(Chance.of.Admit ~. , method = 'glm', data = factor_train_set)
glm_pred <- predict(glm_train, factor_test_set)
glm_confusion <- confusionMatrix(glm_pred, factor_test_set$Chance.of.Admit)
glm_plot <- data.frame(glm_confusion$table) %>% ggplot(aes(Reference, Prediction)) +
  geom_tile(aes(fill = -Freq)) +
  geom_text(aes(Reference, Prediction, label = Freq)) +
  ggtitle('GLM Results')
glm_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 65  6
##          1  3 27
##
##                Accuracy : 0.9109
##                  95% CI : (0.8376, 0.9584)
##     No Information Rate : 0.6733
##     P-Value [Acc > NIR] : 1.715e-08
##
##                   Kappa : 0.7926
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9559
##             Specificity : 0.8182
##          Pos Pred Value : 0.9155
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6733
##          Detection Rate : 0.6436
##    Detection Prevalence : 0.7030
##       Balanced Accuracy : 0.8870
##
##        'Positive' Class : 0
##
```

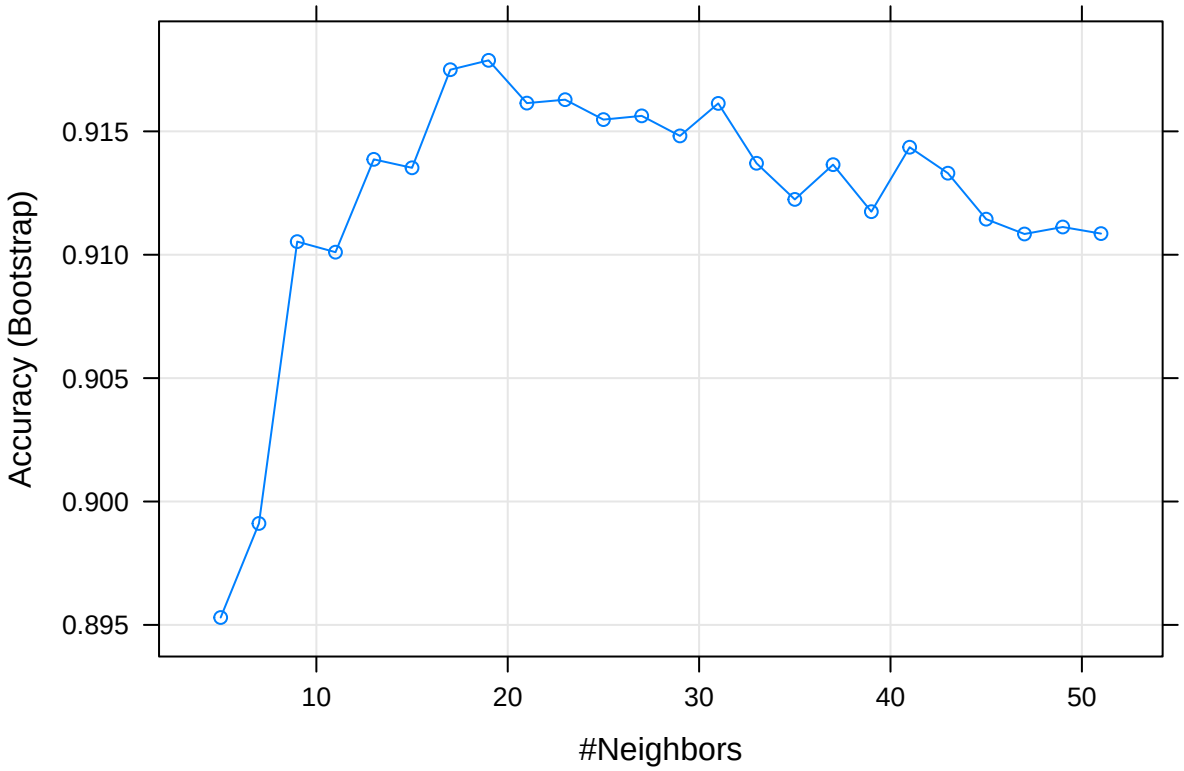
Naive Bayes

```
nb_train <- train(Chance.of.Admit ~. , method = 'nb', data = factor_train_set)
nb_pred <- predict(nb_train, factor_test_set)
nb_confusion <- confusionMatrix(nb_pred, factor_test_set$Chance.of.Admit)
nb_plot <- data.frame(nb_confusion$table) %>% ggplot(aes(Reference, Prediction)) +
  geom_tile(aes(fill = -Freq)) +
  geom_text(aes(Reference, Prediction, label = Freq)) +
  ggtitle('NB Results')
nb_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 67  8
##          1  1 25
##
##                Accuracy : 0.9109
##                  95% CI : (0.8376, 0.9584)
##     No Information Rate : 0.6733
##     P-Value [Acc > NIR] : 1.715e-08
##
##                   Kappa : 0.7858
##  Mcnemar's Test P-Value : 0.0455
##
##             Sensitivity : 0.9853
##             Specificity : 0.7576
##          Pos Pred Value : 0.8933
##          Neg Pred Value : 0.9615
##              Prevalence : 0.6733
##          Detection Rate : 0.6634
##    Detection Prevalence : 0.7426
##       Balanced Accuracy : 0.8714
##
##        'Positive' Class : 0
##
```

KNN

```
knn_train <- train(Chance.of.Admit ~., method = 'knn', data = factor_train_set,
                   tuneGrid = data.frame(k = seq(5, 51, 2)))
knn_pred <- predict(knn_train, factor_test_set)
knn_confusion <- confusionMatrix(knn_pred, factor_test_set$Chance.of.Admit)
# Plot showing the tuning parameters
plot(knn_train)
```




```
knn_plot <- data.frame(knn_confusion$table) %>% ggplot(aes(Reference, Prediction)) +
  geom_tile(aes(fill = -Freq)) +
  geom_text(aes(Reference, Prediction, label = Freq)) +
  ggtitle('KNN Results')
knn_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 65  6
##          1  3 27
##
##                Accuracy : 0.9109
##                  95% CI : (0.8376, 0.9584)
##     No Information Rate : 0.6733
##     P-Value [Acc > NIR] : 1.715e-08
##
##                   Kappa : 0.7926
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9559
##             Specificity : 0.8182
##          Pos Pred Value : 0.9155
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6733
```

```
##          Detection Rate : 0.6436
##    Detection Prevalence : 0.7030
##       Balanced Accuracy : 0.8870
##
##        'Positive' Class : 0
##
```

Random Forest

```
rf_train <- train(Chance.of.Admit ~., method = 'rf', data = factor_train_set)
rf_pred <- predict(rf_train, factor_test_set)
rf_confusion <- confusionMatrix(rf_pred, factor_test_set$Chance.of.Admit)
rf_plot <- data.frame(rf_confusion$table) %>% ggplot(aes(Reference, Prediction)) +
  geom_tile(aes(fill = -Freq)) +
  geom_text(aes(Reference, Prediction, label = Freq)) +
  ggtitle('RF Results')
rf_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 65  6
##          1  3 27
##
##                Accuracy : 0.9109
##                  95% CI : (0.8376, 0.9584)
##     No Information Rate : 0.6733
##     P-Value [Acc > NIR] : 1.715e-08
##
##                   Kappa : 0.7926
##  Mcnemar's Test P-Value : 0.505
##
##             Sensitivity : 0.9559
##             Specificity : 0.8182
##          Pos Pred Value : 0.9155
##          Neg Pred Value : 0.9000
##              Prevalence : 0.6733
##          Detection Rate : 0.6436
##    Detection Prevalence : 0.7030
##       Balanced Accuracy : 0.8870
##
##        'Positive' Class : 0
##
```

Ensemble

With all the predictions from the previous models, we can build an ensemble. The ensemble will take the majority of votes of each model and create its prediction.

```
ensemble_df <- data.frame(rf_pred, knn_pred, nb_pred, glm_pred)
votes <- rowMeans(ensemble_df == '1')
ensemble_pred <- ifelse(votes > 0.5, '1', '0')
```

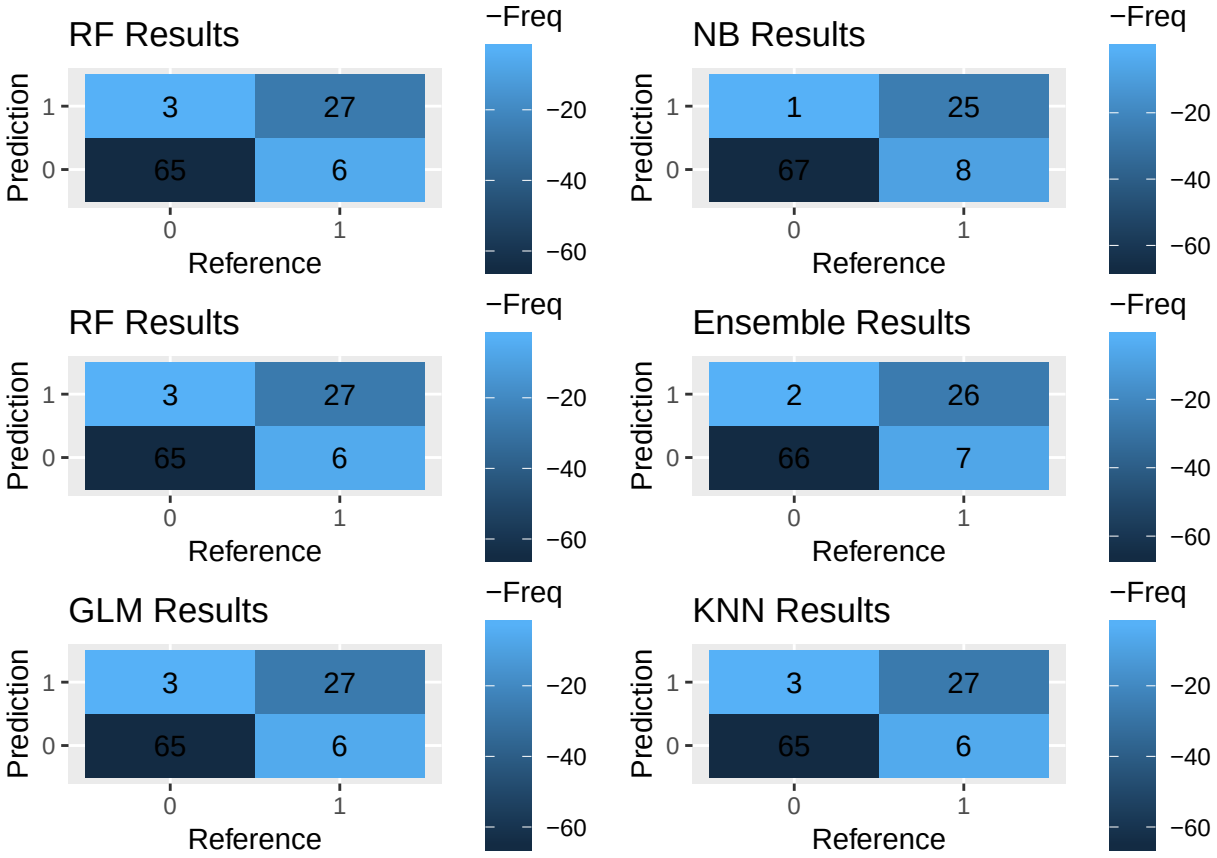
```
ensemble_confusion <- confusionMatrix(factor(ensemble_pred), factor_test_set$Chance.of.Admit)
ensemble_plot <- data.frame(ensemble_confusion$table) %>% ggplot(aes(Reference, Prediction)) +
  geom_tile(aes(fill = -Freq)) +
  geom_text(aes(Reference, Prediction, label = Freq)) +
  ggtitle('Ensemble Results')
ensemble_confusion
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 66  7
##          1  2 26
##
##                Accuracy : 0.9109
##                  95% CI : (0.8376, 0.9584)
##     No Information Rate : 0.6733
##     P-Value [Acc > NIR] : 1.715e-08
##
##                   Kappa : 0.7892
##  Mcnemar's Test P-Value : 0.1824
##
##             Sensitivity : 0.9706
##             Specificity : 0.7879
##          Pos Pred Value : 0.9041
##          Neg Pred Value : 0.9286
##              Prevalence : 0.6733
##          Detection Rate : 0.6535
##    Detection Prevalence : 0.7228
##       Balanced Accuracy : 0.8792
##
##        'Positive' Class : 0
##
```

Results and choosing the best model

Let's look at our results until now. First let's plot all the confusion matrices heatmaps.

```
grid.arrange(rf_plot, nb_plot, rf_plot, ensemble_plot, glm_plot, knn_plot)
```



And now a table with the parameters we will be looking at.

```
enb_df <- data.frame(naive_bayes = nb_confusion$byClass[1:3], ensemble = ensemble_confusion$byClass[1:3]
                     knn = knn_confusion$byClass[1:3], glm = glm_confusion$byClass[1:3],
                     random_forest = rf_confusion$byClass[1:3])
enb_df['Accuracy',] <- c(nb_confusion$overall['Accuracy'], ensemble_confusion$overall['Accuracy'],
                         knn_confusion$overall['Accuracy'], glm_confusion$overall['Accuracy'],
                         rf_confusion$overall['Accuracy'])
enb_df
```

```
##                naive_bayes  ensemble       knn       glm random_forest
## Sensitivity      0.9852941 0.9705882 0.9558824 0.9558824     0.9558824
## Specificity      0.7575758 0.7878788 0.8181818 0.8181818     0.8181818
## Pos Pred Value   0.8933333 0.9041096 0.9154930 0.9154930     0.9154930
## Accuracy         0.9108911 0.9108911 0.9108911 0.9108911     0.9108911
```

Now we should choose which model to use. Even though the Naive Bayes model has a better accuracy, its positive predictive value is lower than the ensemble. This means that the ensemble model will have a better probability to predict if the student is admitted or not. So for this problem, choosing only the highest accuracy will not be a good idea, because we don't want students to choose a university with a admitted(1) prediction and then finding out they have failed.

Conclusion

This dataset is big enough to create a valid prediction model. We've trained a Naive Bayes model, a KNN, GLM, a random_forest and an ensemble, using the votes from every model. The ensemble, even though is not the highest accuracy, have the best sensitivity and specificity of all of the models, because of this it

has the best positive predictive value, which is why we choose it to give our predictions based on students features.