Predicting Admission Probability in U.S Grad school using Linear Models and Machine Learning Algorithms [*]

Arumugam Thiagarajan *Professional Certificate in Data Science, Harvard University*

The project develops and compares a linear and a suite of machine learning algorithms that predicts the admission probabilty of applicants to United States Graduate Schools. The applicant characterisitics and academic standings such as, TOEFL scores, GRE scores, Cumulative Grade Point Average (CGPA), Letter of Recommendation (LOR) and Statement of Purpose are some of the attributes that are used to predict their admission probability to universities. A regression based approach was used and the dataset was explored for trends, cleansed with relevant attributes and models were built using linear regression and machine learning algorithm. Training and validation datasets were estbalished at a 50:50 proportion at random. Root Mean Square and R2 values were used as measures of performance and the results revealed that linear regression model and an ensemble of machine learning models both predicted the outcomes with the same level of accuracy. The RMSE values were at 0.066 with an r2 value of 0.88

Keywords: house rent, machine learning, linear models

Contents

[*] S.V Miller for providing the Pandoc template: github.com/svmiller

Executive Summary

This project builds a mathematical model that predicts the house rents in selected Brazilian cities.

Objective

Predict the admission rates in United States Grad School using the academic scores of the applicants, and the university rankings. Both general linear models and machine learning algorithms will be used and their performances will be compared. Root Mean Squarewill be used as the measure of performance for the model.

#Install and load up the libraries that are required for the analysis to run.

Load the data that is used for the development of the model.

The original data is available for public in the following url: https://www.kaggle.com/

Since a direct download from kaggle requires an authentication, the whole dataset is uploaded to github account. The data and codes can be downloaded from the following github repository. https://github.com/HexyCodes/

This dataset contains, 400 rows of data and 8 attributes.

#Data characteristics and Summary

```
adm=read.csv("~/Documents/Harvard/R/CapStone/Admission/US_grad_admission.csv",
             stringsAsFactors = F)
dim(adm) # find the dimensions of the data.frame
```

```
## [1] 400   9
```

Exploratory Data Analysis

Here the data is analyzed for their summarized characteristics through visualization. This step allows us to find the patterns, trends and any anamolies if exist in the data. The serial number column has been removed to cleanse the data removing unrelated column. This was an obvious choice as this will add unncessary noise to the modeling process.

```
class(adm)
```

```
## [1] "data.frame"
```

```
head(adm,5) # look at the data type of columns
```

```
##   Serial.No. GRE.Score TOEFL.Score University.Rating SOP LOR CGPA Research
## 1          1       337         118                 4 4.5 4.5 9.65        1
## 2          2       324         107                 4 4.0 4.5 8.87        1
## 3          3       316         104                 3 3.0 3.5 8.00        1
## 4          4       322         110                 3 3.5 2.5 8.67        1
## 5          5       314         103                 2 2.0 3.0 8.21        0
##   Chance.of.Admit
## 1            0.92
## 2            0.76
## 3            0.72
## 4            0.80
## 5            0.65
```

```
print(anyNA(adm))# check for missing values
```

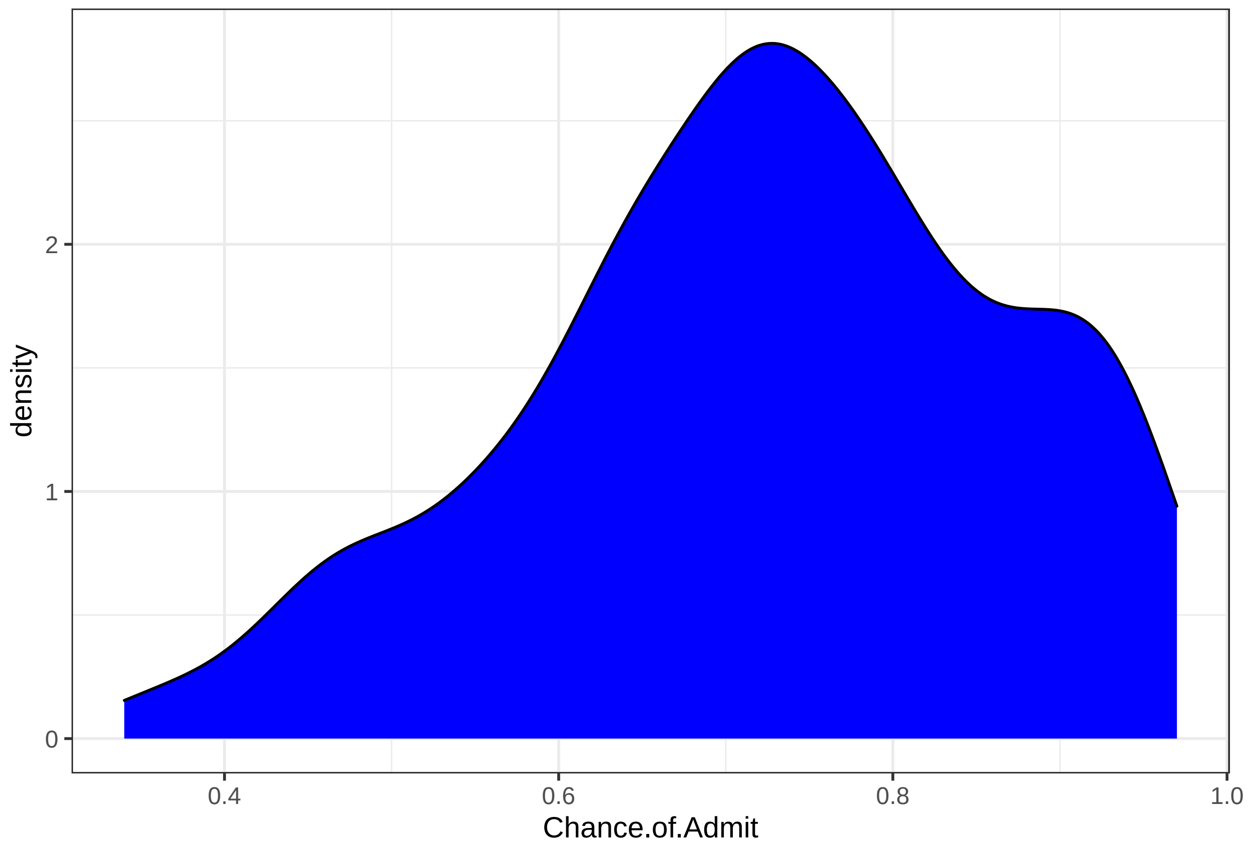
```
## [1] FALSE
```

```
summary(adm) # quantile distribution of the predicted values
```

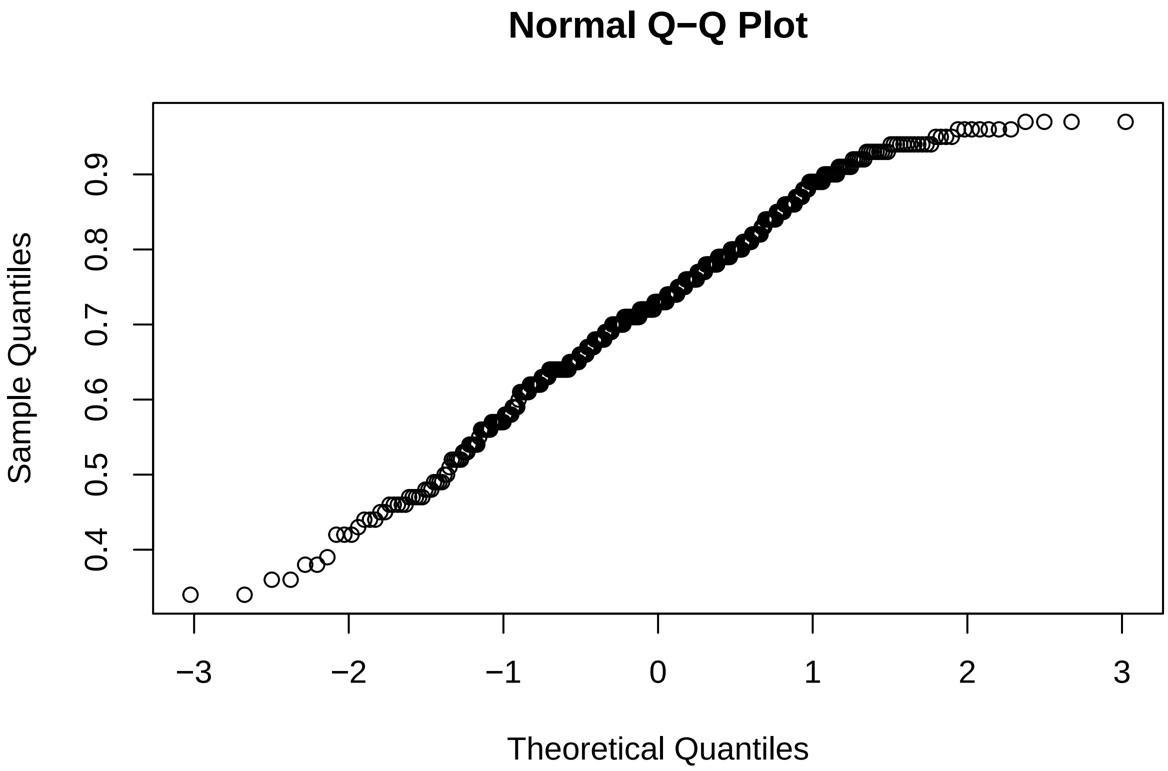
```
##    Serial.No.      GRE.Score      TOEFL.Score    University.Rating
##  Min.   :  1.0   Min.   :290.0   Min.   : 92.0   Min.   :1.000    
##  1st Qu.:100.8   1st Qu.:308.0   1st Qu.:103.0   1st Qu.:2.000    
##  Median :200.5   Median :317.0   Median :107.0   Median :3.000    
##  Mean   :200.5   Mean   :316.8   Mean   :107.4   Mean   :3.087    
##  3rd Qu.:300.2   3rd Qu.:325.0   3rd Qu.:112.0   3rd Qu.:4.000    
##  Max.   :400.0   Max.   :340.0   Max.   :120.0   Max.   :5.000    
##       SOP           LOR             CGPA            Research     
##  Min.   :1.0   Min.   :1.000   Min.   :6.800   Min.   :0.0000  
##  1st Qu.:2.5   1st Qu.:3.000   1st Qu.:8.170   1st Qu.:0.0000  
##  Median :3.5   Median :3.500   Median :8.610   Median :1.0000  
##  Mean   :3.4   Mean   :3.453   Mean   :8.599   Mean   :0.5475  
##  3rd Qu.:4.0   3rd Qu.:4.000   3rd Qu.:9.062   3rd Qu.:1.0000  
##  Max.   :5.0   Max.   :5.000   Max.   :9.920   Max.   :1.0000  
##  Chance.of.Admit 
##  Min.   :0.3400  
##  1st Qu.:0.6400  
##  Median :0.7300  
##  Mean   :0.7244  
##  3rd Qu.:0.8300  
##  Max.   :0.9700  
```

```
adm%>%ggplot(aes(x=Chance.of.Admit))+geom_density(bins = 20, fill="blue") + theme_bw()
```

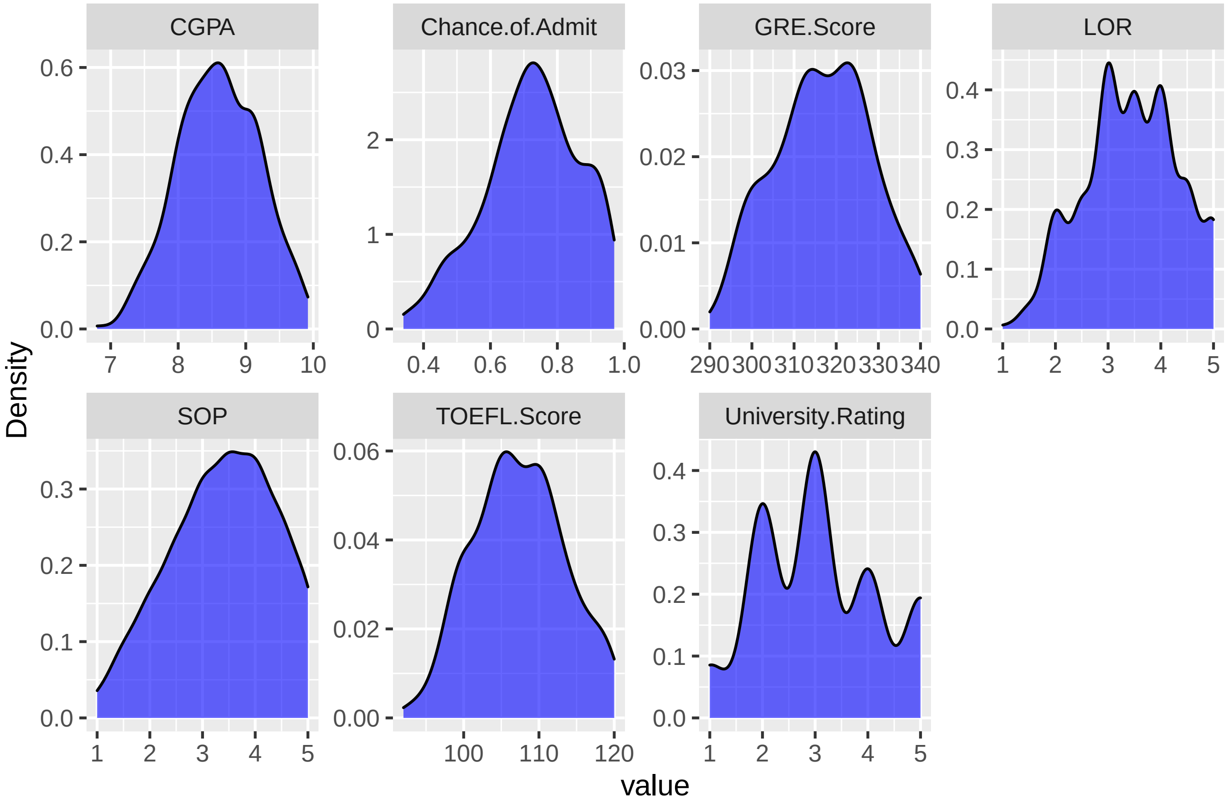
```
## Warning: Ignoring unknown parameters: bins
```



```
  # distribution
#pattern of the predicted value
qqnorm(adm$Chance.of.Admit)
```



```
adm%>%dplyr::select(-Serial.No.)->adm # Remove Serial number from the daa.
plot_density(adm,
             geom_density_args = list("fill"="blue",
                                      "alpha"=0.6)) # distribution all predictors in the data fr
```



Removing Extreme values

Histogram and boxplots of the admission vector indicates a normal distribution for all the features. It is a recommended practice to examine the dataset for outliers. Therefore, a Inter quantile range (IQR) methodology was used to identify the "proposed outliers". First, the Q1 and Q3 quantile are identified, then the IRQ was calculated as teh difference between the Q3 and Q1. The range of values that exist below the IQR*1.5 or above IQR*1.5 were eliminated for this project. From the results only two rows were identified as potential outliers. Considering the low occurrence of these values, the dataset is being used as such with no removal of outliers.

```
any(is.na(adm)) # Checking for any missing values
```

```
## [1] FALSE
```

```
IQR.outliers <- function(x) {
  if(any(is.na(x)))
    stop("x is missing values")
  if(!is.numeric(x))
    stop("x is not numeric")
  Q3<-quantile(x,0.75)
  Q1<-quantile(x,0.25)
  IQR<-(Q3-Q1)
  left<- (Q1-(1.5*IQR))
  print(left)

  right<- (Q3+(1.5*IQR))
    print(right)
  c(x[x <left],x[x>right])
}



outliers=IQR.outliers(adm$Chance.of.Admit)
```

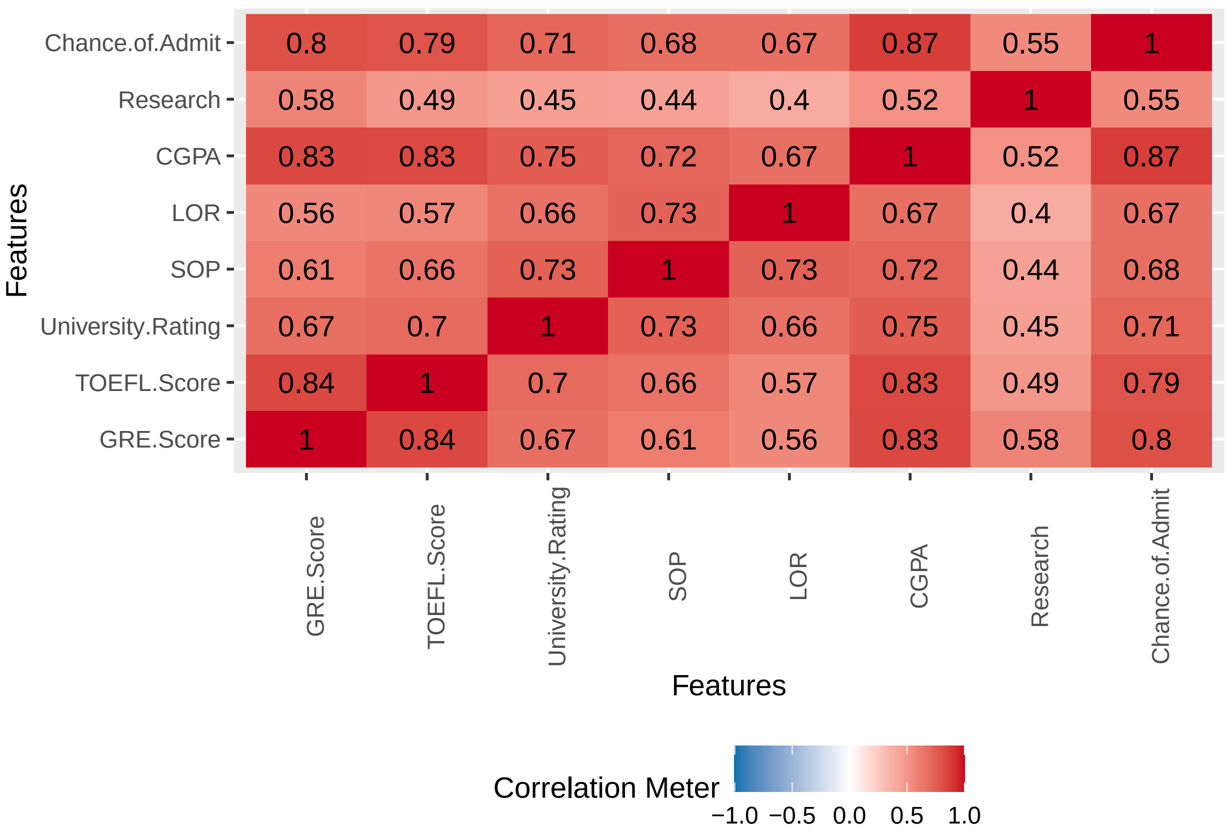
```
##   25%
## 0.355
##   75%
## 1.115
```

Data Exploration

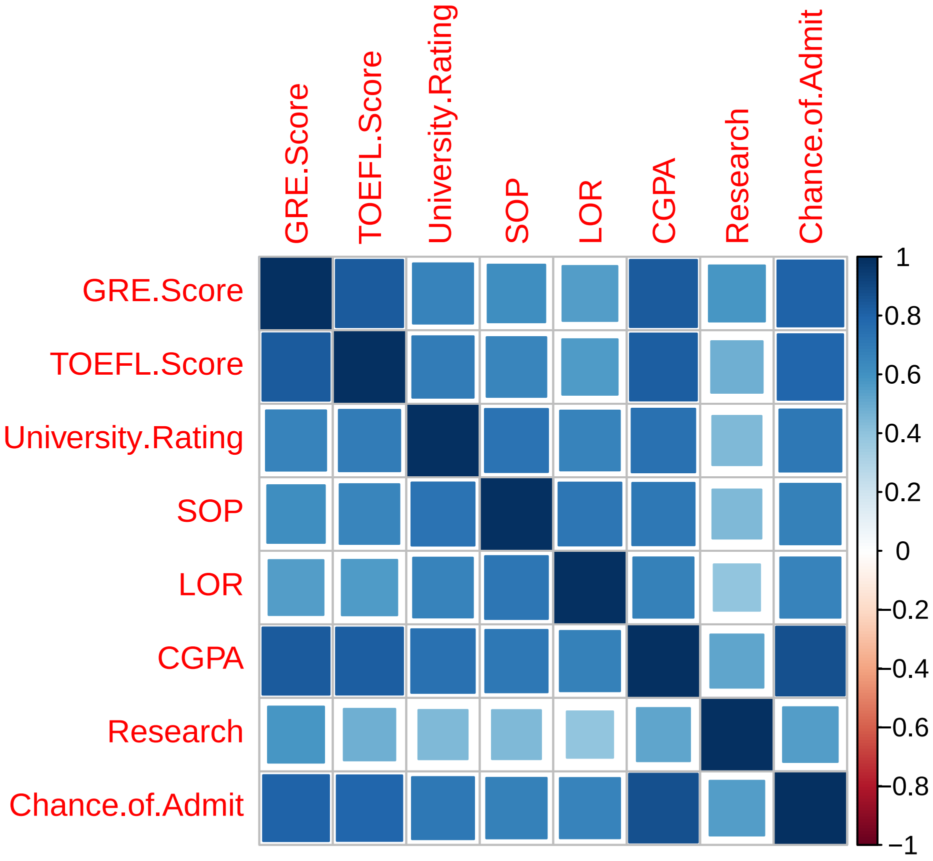
Check for correlations

The dataset is examined for correlations among the different attributes. There seems to a be strong correlations (>50%) between all of the features, such as GRE. Score, TOEFL. Score, University. Rating, SOP, LOR, CGPA and Research. The boxplot on the important feature reveal a linear relationship with these features. This is an interesting trend, because many of these characteristics have confounding effects or colinearity that exist with them. For instance, a person scoring high in GRE has high probability of scoring high in TOEFL and potentially wrote a worthy statemnent of purpose. Therefore, it is essential to examinte partial correlation coefficients of these attributes on the admisssion chances.

```
# Converting the data into matrix format for conduction correlation analysis
data.matrix(adm)->adm_mat
# plotting the hrent matrix results
plot_correlation(adm_mat)
```



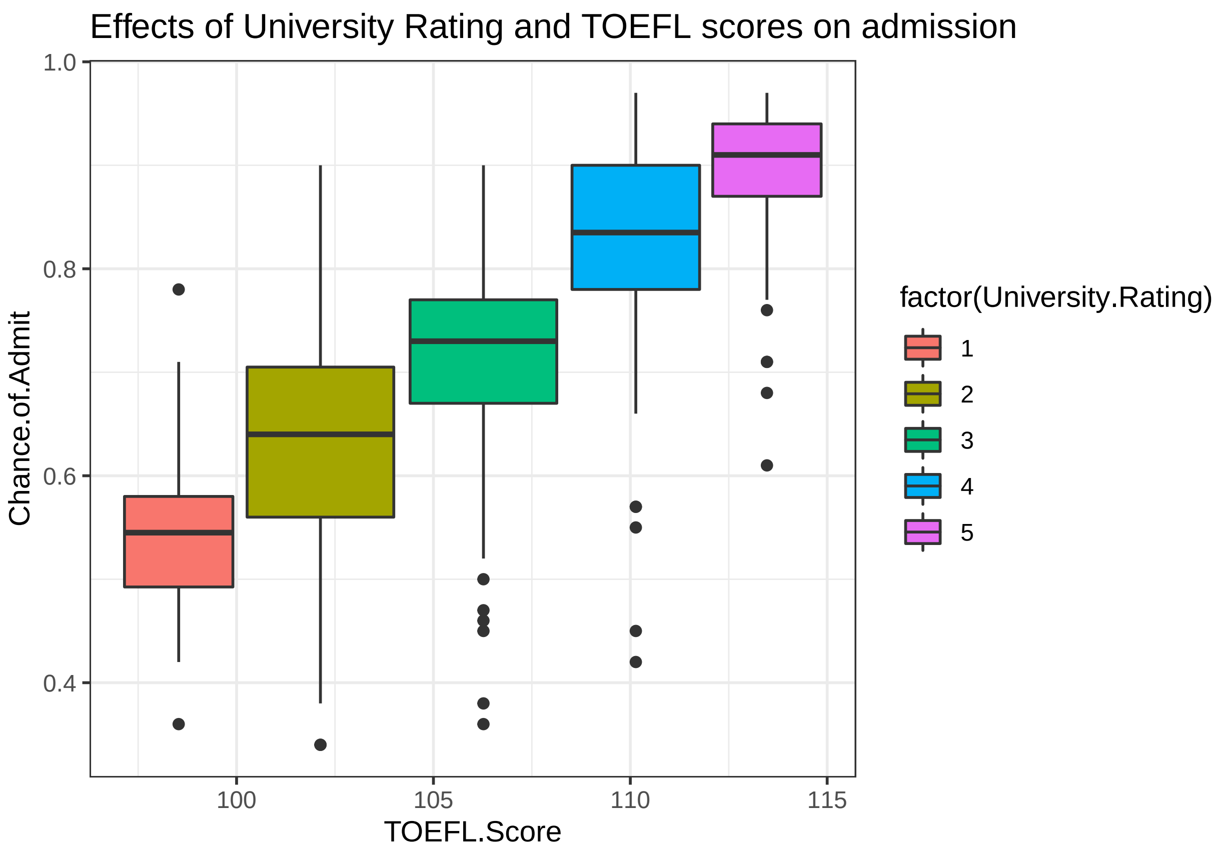
```r
# plotting the correlation strength through size of squares.
corrplot(cor(adm_mat), method = "square")
```



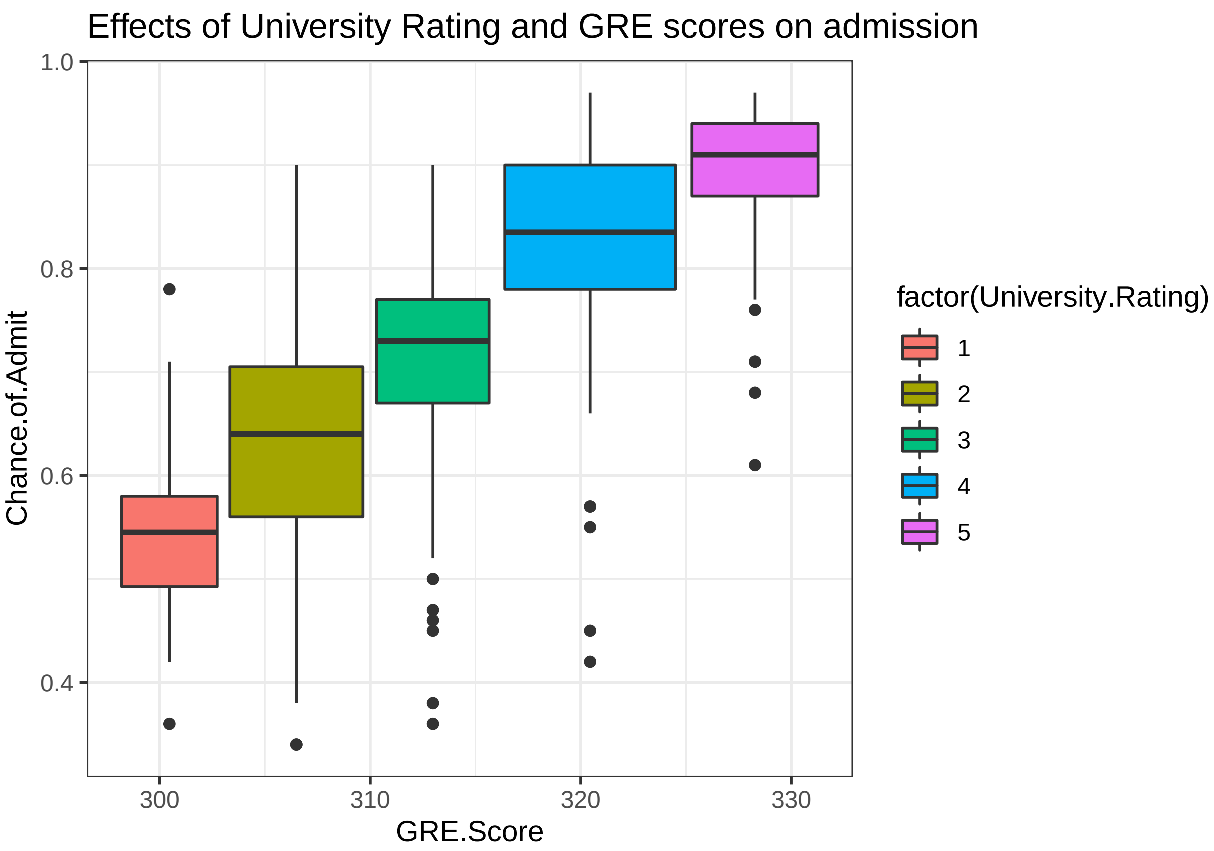
Visualizing the relationship

This step further explores the relationship by visualizing the spread of the attributes and presenting the relationship between combination of attributes in influencing the admission rates. I explore the impacts of TOEFL. Score, GRE.Score and CGPA on Admission grouped by University Rating. The trend is linear and there is strong evidence that these is positively related to the admission rates and the requirements vary with Universities.

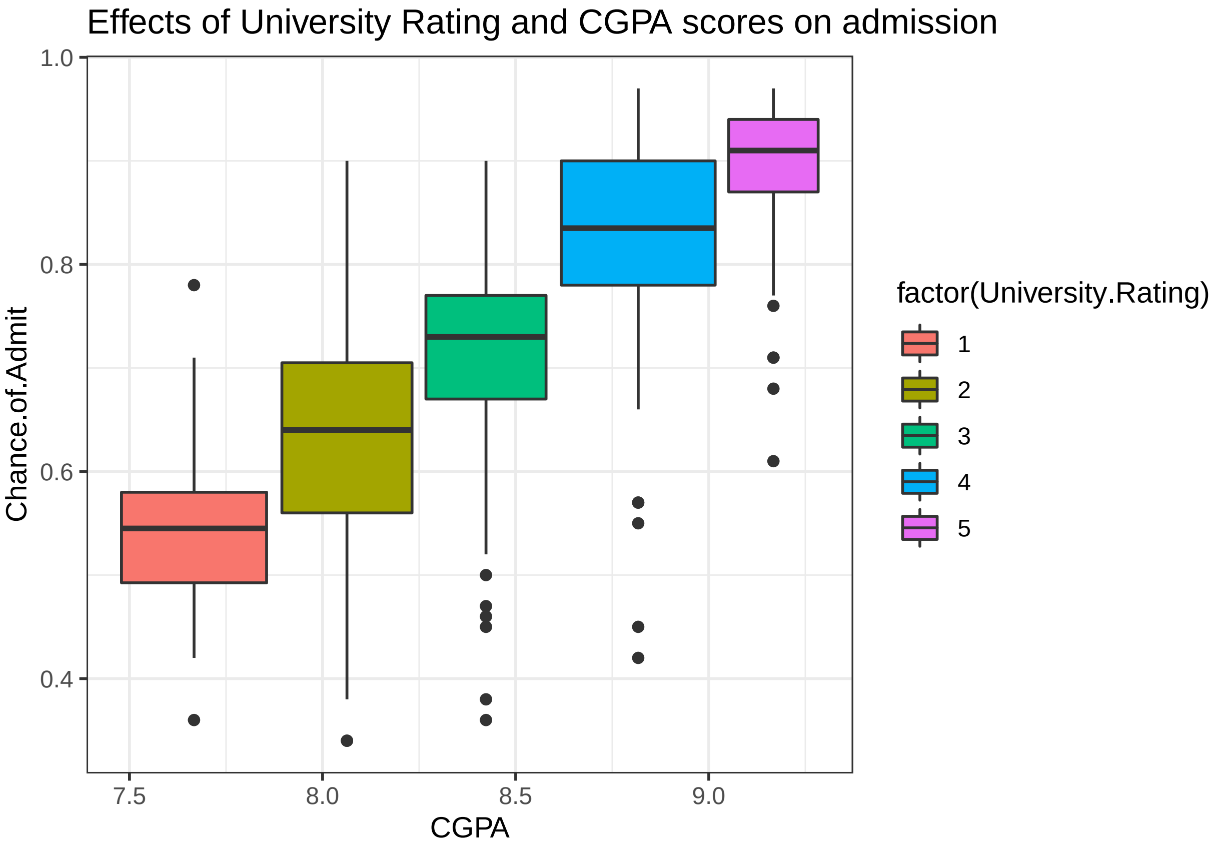
```
adm%>%
  ggplot(aes(x=TOEFL.Score,y=Chance.of.Admit, fill=factor(University.Rating))) +
  geom_boxplot() +theme_bw()+
  ggtitle("Effects of University Rating and TOEFL scores on admission")
```



```
adm%>%
  ggplot(aes(x=GRE.Score,
             y=Chance.of.Admit, fill=factor(University.Rating))) +
  geom_boxplot() +theme_bw()+
  ggtitle("Effects of University Rating and GRE scores on admission")
```



```
adm%>%
  ggplot(aes(x=CGPA,y=Chance.of.Admit, fill=factor(University.Rating))) +
  geom_boxplot() +theme_bw()+
  ggtitle("Effects of University Rating and CGPA scores on admission")
```



Partial correlation coefficient

Beyond the correlation coefficients, the partial correlations reveal the influence of invidual attribute to our dependent varaible of interest. Furthermore, partial correlation ensures that the confounding effects of variables are eliminated. This analysis show that the SOP and University.Rating had a minor influence when partial correlation values are considered. Based on these findings SOP, University.Rating will be cleansed from our datasets before our modeling efforts.

```
partials=pcor(adm_mat) # Conducting partial correlation analysis
print("Partial Correlations for the Dependent Variable: Rent")
```

```
## [1] "Partial Correlations for the Dependent Variable: Rent"
```

```
Estimates=data.frame(partials$estimate[,7:8])
P.values=data.frame(partials$p.value[, 7:8])
# printing the results
kable((Estimates), format = "markdown", digits=2,
      caption="Partial correlations  of the input dataset attributes")
```

	Research	Chance.of.Admit
GRE.Score	0.26	0.15
TOEFL.Score	-0.07	0.13
University.Rating	0.01	0.06
SOP	0.08	-0.03
LOR	-0.01	0.20
CGPA	-0.05	0.44
Research	1.00	0.15
Chance.of.Admit	0.15	1.00

```
kable((P.values), format = "markdown", digits=2,
      caption="Probability values  of the input dataset attributes")
```

	Research	Chance.of.Admit
GRE.Score	0.00	0.00
TOEFL.Score	0.17	0.01
University.Rating	0.78	0.23
SOP	0.10	0.55
LOR	0.87	0.00
CGPA	0.36	0.00
Research	0.00	0.00
Chance.of.Admit	0.00	0.00

Data Preparation for Modeling

##Data Cleansing Based on the partial correlation coefficient analysis, the SOP and the University Rating features are removed from the dataset.

```
adm%>%dplyr::select(-SOP, -University.Rating)->admfea # data cleansed after removing SOP
```

Splitting data into training and validation datasets.

The data is split into two datasets. One for training and validation. The training dataset will be used for model development and the validation dataset will only be used for validation of the model as a final step. Twenty percent of the cleansed data was chosen as the validation dataset at random (318) and the rest (82) was saved as the training dataset. The attributes selected were GRE.Score, TOEFL.Score, University.Rating, LOR, CGPA and Research

```
test_indices=createDataPartition(admfea$Chance.of.Admit,
                                 times=1,
                                 p=0.5, # portion of data split into test
                                 list=F)
admfea[-test_indices,]->traindf # dataset reserved for training
admfea[test_indices,]->valdf # dataset held for validation
```

Approach and Model Development

From the data types, boxplots and intial data exploration, it is evident that this is a regression problem. Accordingly, regression based modeling solutions will be explored for model development. Initially, a general linear model will be built using all of the features in the datasets. Following this a feature reduction step will be performed for the linear models using a Stepwise regression. A backward and foward propagated stepwise regression will be performed and the model that exhibits the lowest AIC score will be selected. The AIC refers to the Akaike Information Criteria that defines the performance of the model chosen through a penalization procedure.

```
# cl=makePSOCKcluster(detectCores())
# registerDoParallel(cl)
# #Create control function for training with 10 folds
# #and keep 3 folds for training. search method is grid.
#
# control <- trainControl(method=’repeatedcv’,
#                         number=10,
#                         repeats=3,
#                         search=’grid’)
#
# tunegrid <- expand.grid(predFixed = c(1:5), minNode=1:3)
# rf_gridsearch <- train(Chance.of.Admit ~ .,
#                        data = trainset,
#                        method = ’Rborist’,
#                        metric = c("RMSE"),
#                       tuneGrid = tunegrid)
# print(rf_gridsearch)
# stopCluster(cl)
```

Linear Model

```
mod.lm=lm(Chance.of.Admit~., data=traindf)
pred.lm=predict(mod.lm, newdata=valdf)
RMSE(pred.lm, valdf$Chance.of.Admit)
```

```
## [1] 0.06337683
```

```
stepAIC(mod.lm, direction="both")
```

```
## Start:  AIC=-1077.91
## Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR + CGPA + Research
##
##                Df Sum of Sq     RSS     AIC
## - TOEFL.Score  1   0.00077 0.83303 -1079.7
## <none>                     0.83226 -1077.9
## - GRE.Score    1   0.01578 0.84804 -1076.2
## - Research     1   0.01967 0.85193 -1075.3
## - LOR          1   0.04888 0.88114 -1068.5
## - CGPA         1   0.31951 1.15177 -1015.2
##
## Step:  AIC=-1079.72
## Chance.of.Admit ~ GRE.Score + LOR + CGPA + Research
##
##                Df Sum of Sq     RSS     AIC
## <none>                      0.83303 -1079.7
## + TOEFL.Score  1   0.00077 0.83226 -1077.9
## - Research     1   0.01937 0.85240 -1077.2
## - GRE.Score    1   0.02355 0.85658 -1076.2
## - LOR          1   0.04961 0.88264 -1070.2
## - CGPA         1   0.38380 1.21682 -1006.3
```

```
##
## Call:
## lm(formula = Chance.of.Admit ~ GRE.Score + LOR + CGPA + Research,
##     data = traindf)
##
## Coefficients:
## (Intercept)    GRE.Score          LOR         CGPA     Research
##   -1.213402     0.001827     0.023654     0.146315     0.025734
```

```
mod.lm.step=lm(Chance.of.Admit~GRE.Score+
                 TOEFL.Score+LOR+CGPA, data=traindf)
summary(mod.lm.step)
```

```
##
## Call:
## lm(formula = Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR +
##     CGPA, data = traindf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23796 -0.01987  0.01154  0.03992  0.14565
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.3996427  0.1555292  -8.999  < 2e-16 ***
## GRE.Score    0.0023099  0.0008240   2.803 0.005572 **
## TOEFL.Score  0.0004945  0.0015119   0.327 0.743955
## LOR          0.0252382  0.0069987   3.606 0.000395 ***
## CGPA         0.1449749  0.0168360   8.611 2.48e-15 ***
## ---
## Signif. codes:  0 ’***’ 0.001 ’**’ 0.01 ’*’ 0.05 ’.’ 0.1 ’ ’ 1
##
## Residual standard error: 0.06627 on 194 degrees of freedom
## Multiple R-squared:  0.7925, Adjusted R-squared:  0.7882
## F-statistic: 185.2 on 4 and 194 DF,  p-value: < 2.2e-16
```

Machine Learning Models

```
knitr::opts_chunk$set(cache=T)

set.seed(1)

cl=makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)

my.con=trainControl(method="cv", number=3,
                    savePredictions = "final", allowParallel = T)
models=caretList(Chance.of.Admit~., data=traindf,
trainControl=my.con,
methodList = c("Rborist",
               "knn",
               "glmnet",
               "xgbLinear",
               "brnn",
               "ridge"),
continue_on_fail = T)
```

```
## Warning in trControlCheck(x = trControl, y = target): trControl$savePredictions
## not ’all’ or ’final’. Setting to ’final’ so we can ensemble the models.

## Warning in trControlCheck(x = trControl, y = target): indexes not defined in
## trControl. Attempting to set them ourselves, so each model in the ensemble will
## have the same resampling indexes.

## [12:44:33] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now de
## [12:44:33] WARNING: amalgamation/../src/learner.cc:480:
## Parameters: { trainControl } might not be used.
##
##   This may not be accurate due to some parameters are only used in language bindings but
##   passed down to XGBoost core.  Or some parameters are not used but slip through this
##   verification. Please open an issue if you find above cases.
##
##
## Number of parameters (weights and biases) to estimate: 7
## Nguyen-Widrow method
## Scaling factor= 0.7
## gamma= 6.5184     alpha= 1.228    beta= 10.6021
## 1 package is needed for this model and is not installed. (elasticnet). Would you like to try
```

```r
models$xgbLinear
```

```
## eXtreme Gradient Boosting
##
## 199 samples
##   5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
## Resampling results across tuning parameters:
##
##   lambda  alpha  nrounds  RMSE        Rsquared   MAE
##   0e+00   0e+00   50      0.08205874  0.7005220  0.06067428
##   0e+00   0e+00  100      0.08205872  0.7005220  0.06067422
##   0e+00   0e+00  150      0.08205868  0.7005220  0.06067416
##   0e+00   1e-04   50      0.08151685  0.7034346  0.06004102
##   0e+00   1e-04  100      0.08151681  0.7034346  0.06004095
##   0e+00   1e-04  150      0.08151678  0.7034345  0.06004090
##   0e+00   1e-01   50      0.07677988  0.7323696  0.05711481
##   0e+00   1e-01  100      0.07677988  0.7323696  0.05711481
##   0e+00   1e-01  150      0.07677988  0.7323696  0.05711481
##   1e-04   0e+00   50      0.08203076  0.7006150  0.06065748
##   1e-04   0e+00  100      0.08203073  0.7006150  0.06065741
##   1e-04   0e+00  150      0.08203069  0.7006150  0.06065735
##   1e-04   1e-04   50      0.08168928  0.7021681  0.06018655
##   1e-04   1e-04  100      0.08168926  0.7021680  0.06018650
##   1e-04   1e-04  150      0.08168923  0.7021680  0.06018645
##   1e-04   1e-01   50      0.07677987  0.7323933  0.05712004
##   1e-04   1e-01  100      0.07677987  0.7323933  0.05712004
##   1e-04   1e-01  150      0.07677987  0.7323933  0.05712004
##   1e-01   0e+00   50      0.08073932  0.7088634  0.05964312
##   1e-01   0e+00  100      0.08073930  0.7088633  0.05964307
##   1e-01   0e+00  150      0.08073926  0.7088633  0.05964301
##   1e-01   1e-04   50      0.08055568  0.7103941  0.05971346
##   1e-01   1e-04  100      0.08055568  0.7103941  0.05971341
##   1e-01   1e-04  150      0.08055566  0.7103941  0.05971336
##   1e-01   1e-01   50      0.07691574  0.7308336  0.05731010
##   1e-01   1e-01  100      0.07691574  0.7308336  0.05731010
##   1e-01   1e-01  150      0.07691574  0.7308336  0.05731010
##
## Tuning parameter ’eta’ was held constant at a value of 0.3
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were nrounds = 50, lambda = 1e-04, alpha
##  = 0.1 and eta = 0.3.
```

```
models$knn
```

```
## k-Nearest Neighbors
##
## 199 samples
##   5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
## Resampling results across tuning parameters:
##
##   k  RMSE        Rsquared   MAE
##   5  0.09306460  0.6055756  0.06961677
##   7  0.09018177  0.6273696  0.06787979
##   9  0.08941066  0.6365068  0.06740945
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was k = 9.
```

```
models$Rborist
```

```
## Random Forest
##
## 199 samples
##   5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
## Resampling results across tuning parameters:
##
##   predFixed  RMSE        Rsquared   MAE
##   2          0.07007374  0.7738223  0.05219741
##   3          0.07140460  0.7648293  0.05306859
##   5          0.07411851  0.7471535  0.05481163
##
## Tuning parameter 'minNode' was held constant at a value of 3
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were predFixed = 2 and minNode = 3.
```

```
models$glmnet
```

```
## glmnet
##
## 199 samples
##   5 predictor
```

```
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
## Resampling results across tuning parameters:
##
##   alpha  lambda        RMSE        Rsquared   MAE
##   0.10   0.0002514148  0.06702287  0.7923582  0.04860223
##   0.10   0.0025141476  0.06687991  0.7935777  0.04839192
##   0.10   0.0251414757  0.06810763  0.7930261  0.04933138
##   0.55   0.0002514148  0.06701280  0.7924638  0.04861109
##   0.55   0.0025141476  0.06682233  0.7941896  0.04839073
##   0.55   0.0251414757  0.07051156  0.7943757  0.05291759
##   1.00   0.0002514148  0.06700746  0.7924666  0.04863224
##   1.00   0.0025141476  0.06691587  0.7939357  0.04855497
##   1.00   0.0251414757  0.07423164  0.7888777  0.05742705
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.55 and lambda = 0.002514148.
```

```
models$brnn
```

```
## Bayesian Regularized Neural Networks
##
## 199 samples
##   5 predictor
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 199, 199, 199, 199, 199, 199, ...
## Resampling results across tuning parameters:
##
##   neurons  RMSE        Rsquared   MAE
##   1        0.06776434  0.7872853  0.04909314
##   2        0.06860481  0.7821797  0.04978834
##   3        0.07133352  0.7642993  0.05161799
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was neurons = 1.
```

```
stopCluster(cl)
varImp(models$Rborist)
```

```
## Rborist variable importance
##
##              Overall
## CGPA          100.00
## GRE.Score     42.11
## TOEFL.Score   26.11
## LOR           14.41
## Research       0.00
```

```
varImp(models$glmnet)
```

```
## glmnet variable importance
##
##              Overall
## CGPA        100.0000
## Research     17.1930
## LOR          16.2224
## GRE.Score     0.6967
## TOEFL.Score   0.0000
```

Model with Reduced Features

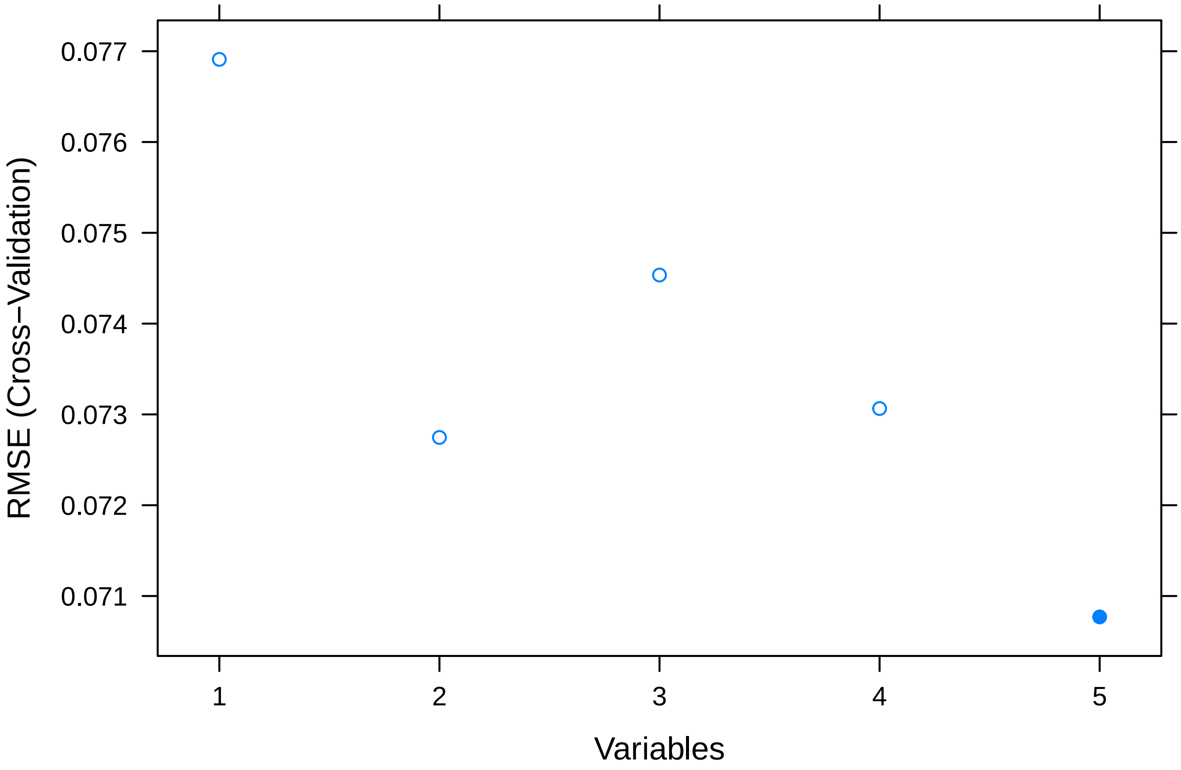
```
# traindf%>%dplyr::select(-Research, -SOP)->traindf_red
# cl=makePSOCKcluster(detectCores()-1)
# registerDoParallel(cl)
# models=caretList(Chance.of.Admit~., data=traindf_red,
# trainControl=my.con,
# methodList = c("Rborist",
#                "knn",
#                "glmnet",
#                "xgbLinear",
#                "brnn"),
# continue_on_fail = T)
# models$xgbLinear
# models$knn
# models$Rborist
# models$glmnet
# models$brnn
# varImp(models$Rborist)
# varImp(models$glmnet)
```

Feature Reduction

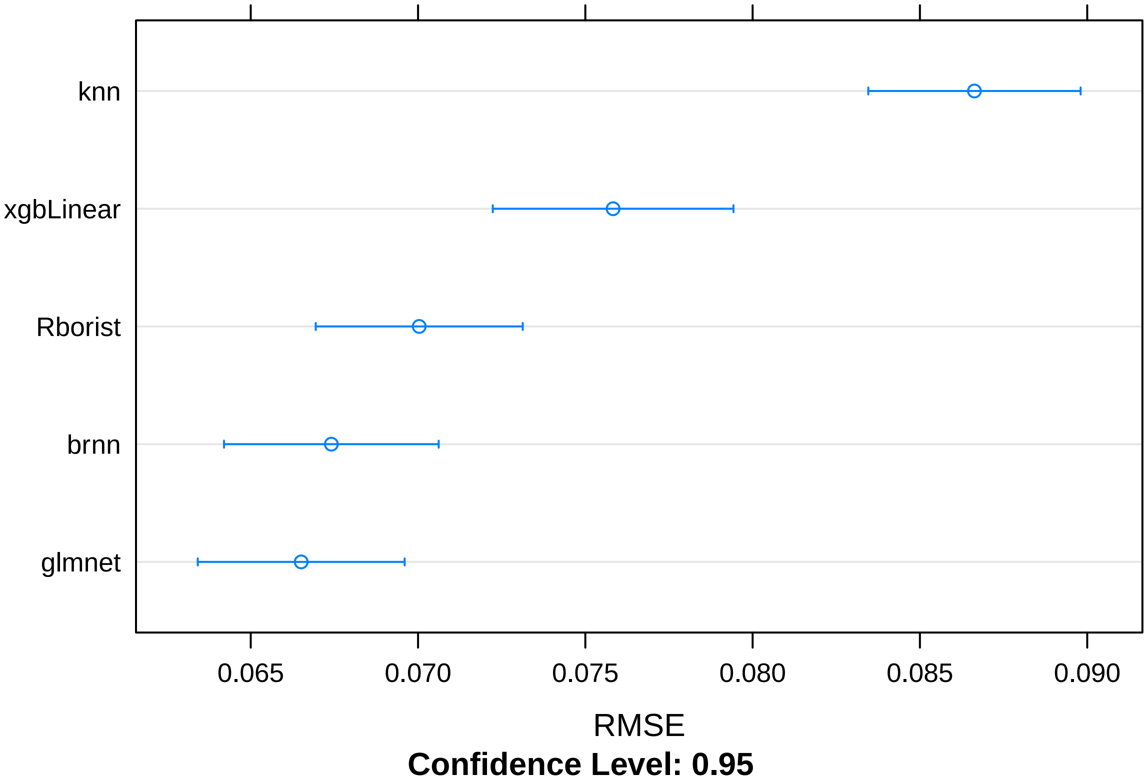
```
set.seed(1)
cl=makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)

ctrl=rfeControl(functions = rfFuncs,
           method = "cv",
           number = 2,
           verbose = F)
subsets=c(1:6)
lmProfile=rfe(traindf[,-6], traindf[,6], sizes=subsets, rfeControl=ctrl)
lmProfile
```

```
##
## Recursive feature selection
##
## Outer resampling method: Cross-Validated (2 fold)
##
## Resampling performance over subset size:
##
##  Variables    RMSE Rsquared     MAE    RMSESD RsquaredSD    MAESD Selected
##          1 0.07691   0.7143 0.05767 0.0076163    0.05154 0.005530
##          2 0.07275   0.7463 0.05572 0.0036125    0.02404 0.005764
##          3 0.07454   0.7453 0.05623 0.0006496    0.01353 0.003492
##          4 0.07306   0.7545 0.05469 0.0026195    0.01914 0.002431
##          5 0.07077   0.7762 0.05266 0.0018916    0.02764 0.004516        *
##
## The top 5 variables (out of 5):
##    CGPA, GRE.Score, LOR, Research, TOEFL.Score
```

```
plot(lmProfile)
```



```
cl=makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)
resamples<-resamples(models)
dotplot(resamples, metric="RMSE")
```



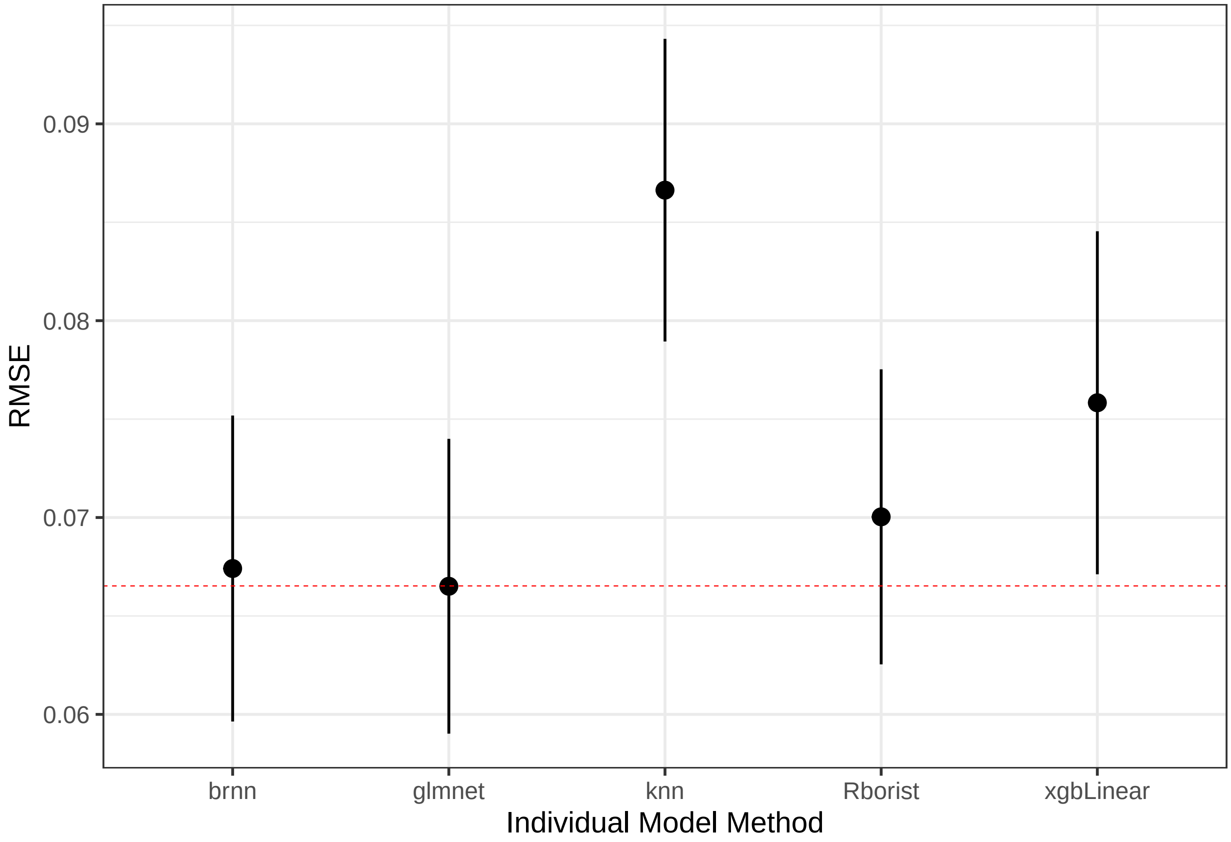
# Results

```
cl=makePSOCKcluster(detectCores())
registerDoParallel(cl)
ens=caretEnsemble(models, metric="RMSE", trControl=my.con)
summary(ens)
```

```
## The following models were ensembled: Rborist, knn, glmnet, xgbLinear, brnn
## They were weighted:
## -0.0257 0.1918 0.0711 1.3128 -0.0134 -0.5298
## The resulting RMSE is: 0.0665
## The fit for each individual model on the RMSE is:
##     method       RMSE      RMSESD
##    Rborist 0.07003540 0.007494216
##        knn 0.08663003 0.007687303
##     glmnet 0.06650807 0.007488198
##  xgbLinear 0.07583018 0.008713204
##       brnn 0.06740915 0.007771737
```

```
plot(ens)
```

```
## Warning: Duplicated aesthetics after name standardisation: size
```



#Validation

In this step, the model are evaluated for their performance. The models are used to predict the admission probabilities of the students with the validation dataset which was reserved from participating in the model developent process. The actual values of the admission rates were compared to the predicted values from various models. Both RMSE an the R2 values were evaluated.

```
#Prediction from Linear model
pred.lm.step=predict(mod.lm.step, newdata = valdf)

#Predicted from Rborist

 predicted.Rborist=predict(models$Rborist, newdata=valdf)

#Prediction from Ensemble of the models
 predicted.ens=predict(ens, newdata=valdf)
```

```
## [12:44:42] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now de
```

```
 data.frame(Rborist=RMSE(predicted.Rborist, valdf$Chance.of.Admit),
            Ensemble=RMSE(predicted.ens, valdf$Chance.of.Admit),
            Linear=RMSE(pred.lm.step, valdf$Chance.of.Admit))
```

```
##      Rborist   Ensemble     Linear
## 1 0.06975151 0.05834425 0.06418429
```

```
data.frame(Rborist=cor(predicted.Rborist, valdf$Chance.of.Admit),
           Ensemble=cor(predicted.ens, valdf$Chance.of.Admit),
           Linear=cor(pred.lm.step, valdf$Chance.of.Admit))
```

```
##     Rborist  Ensemble    Linear
## 1 0.8720163 0.9116409 0.8940309
```

Conclusion

Comparison was made between a linear regression model and a ensemble technique with machine learning algorithms for their abilities to predict the admission probability of applicants to US graduate schools. The applicant characteristics such as GRE.Score, TOEFL.Score, CGPA, LOR and SOP were positively correlated with the admission probabilities. Both linear regression and an ensemble of maching learning algorithms both predicted the outcomes with great accuracy (RME =0.065 and R2 of 0.88). For this given problem, either of the approaches will work. Nevertheless, owing to the simplicy of linear model, for practical purposes, the linear model approach is favored.