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 builds and compares linear and suite of machine learning algorithms that predict 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 features that are used to predict their probability of university admission. A regression based approach was used and the dataset was explored for trends. Dataset was cleansed with relevant features and models models. were built using linear regression and a machine learning algorithms. Training and validation datasets were established at a 50:50 proportion at random. Root Mean Square Error and R2 values were used as measures of performance and the results revealed that GLMnet achieved a higher level of accuracy compared any other models. The RMSE values were at 0.065 with an r2 value of 0.887, better than the ensemble or linear regression

Keywords: house rent, machine learning, linear models

Contents

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

Objective

Predict the admission rates of United States Graduate Schools using the academic scores of the applicants and university rankings. Both general linear models and machine learning algorithms will be used and their performances will be compared. Root Mean Square and R2 will be used as the measure of performance for the models.

Materials and Methods

Input Data

The original data is available for public at the following url: www.kaggle.com/mohansacharya/graduate-admissions Since a direct download from kaggle requires an authentication, the whole dataset is uploaded to a github account. The data and codes are downloaded from the following github repository. https://github.com/HexyCodes/Admission.git. This dataset contains, 400 rows of data and 8 features.

```
url="https://raw.githubusercontent.com/HexyCodes/Admission/master/US_grad_admission.csv"
adm=read_csv(url)
```

```
## Parsed with column specification:
## cols(
##   `Serial No.` = col_double(),
##   `GRE Score` = col_double(),
##   `TOEFL Score` = col_double(),
##   `University Rating` = col_double(),
##   SOP = col_double(),
##   LOR = col_double(),
##   CGPA = col_double(),
##   Research = col_double(),
##   `Chance of Admit` = col_double()
## )
```

```
dim(adm) # find the dimensions of the data.frame
```

```
[1] 400 9
```

```
colnames(adm)=c("Serial.No", "GRE.Score", "TOEFL.Score", "University.Rating",
                "SOP", "LOR", "CGPA", "Research", "Chance.of.Admit")
```

Exploratory Data Analysis

In this exploration, the data is analyzed with their summarized characteristics and examined through visualization charts. This step allows to find the patterns, trends and any anamolies that may exist in the data. The serial number column has been removed. This was an obvious choice as this would add unncessary noise to the modeling process.

```
class(adm)
```

```
## [1] "spec_tbl_df" "tbl_df"      "tbl"         "data.frame"
```

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

```
## # A tibble: 5 x 9
##   Serial.No GRE.Score TOEFL.Score University.Rati~   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
## # ... with 1 more variable: Chance.of.Admit <dbl>
```

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

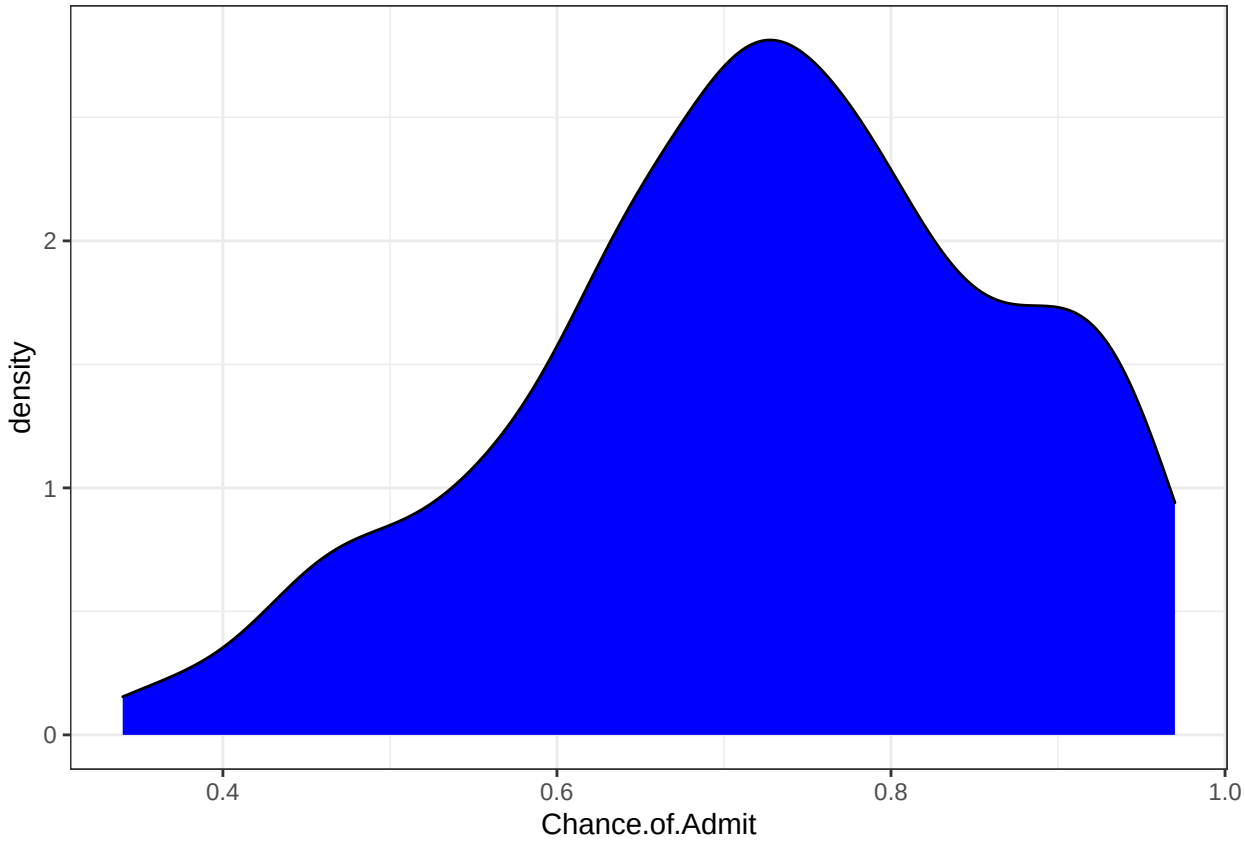
```
## [1] FALSE
```

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

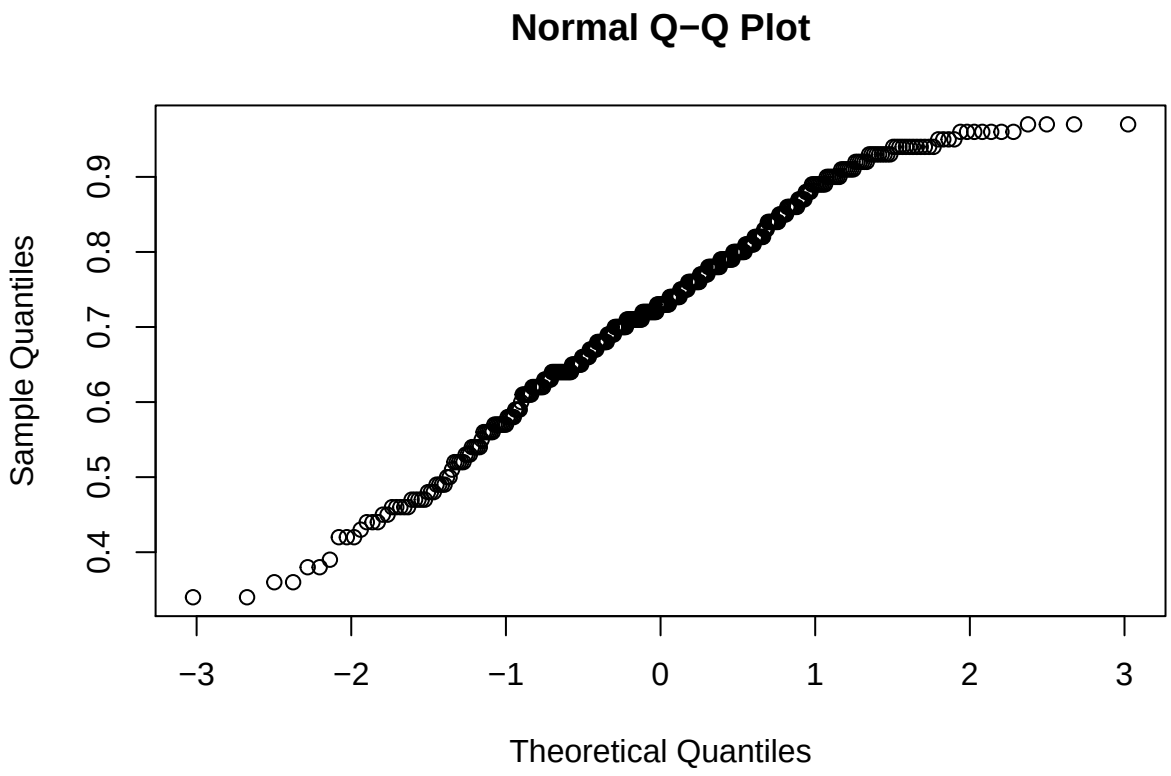
```
##    Serial.No       GRE.Score      TOEFL.Score    University.Rating
##  Min.   :  1.0   Min.   :290.0   Min.   : 92.0   Min.   :1.000
##  1st Qu.:100.8   1st Qu.:308.0   1st Qu.:103.0   1st Qu.:2.000
##  Median :200.5   Median :317.0   Median :107.0   Median :3.000
##  Mean   :200.5   Mean   :316.8   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
```

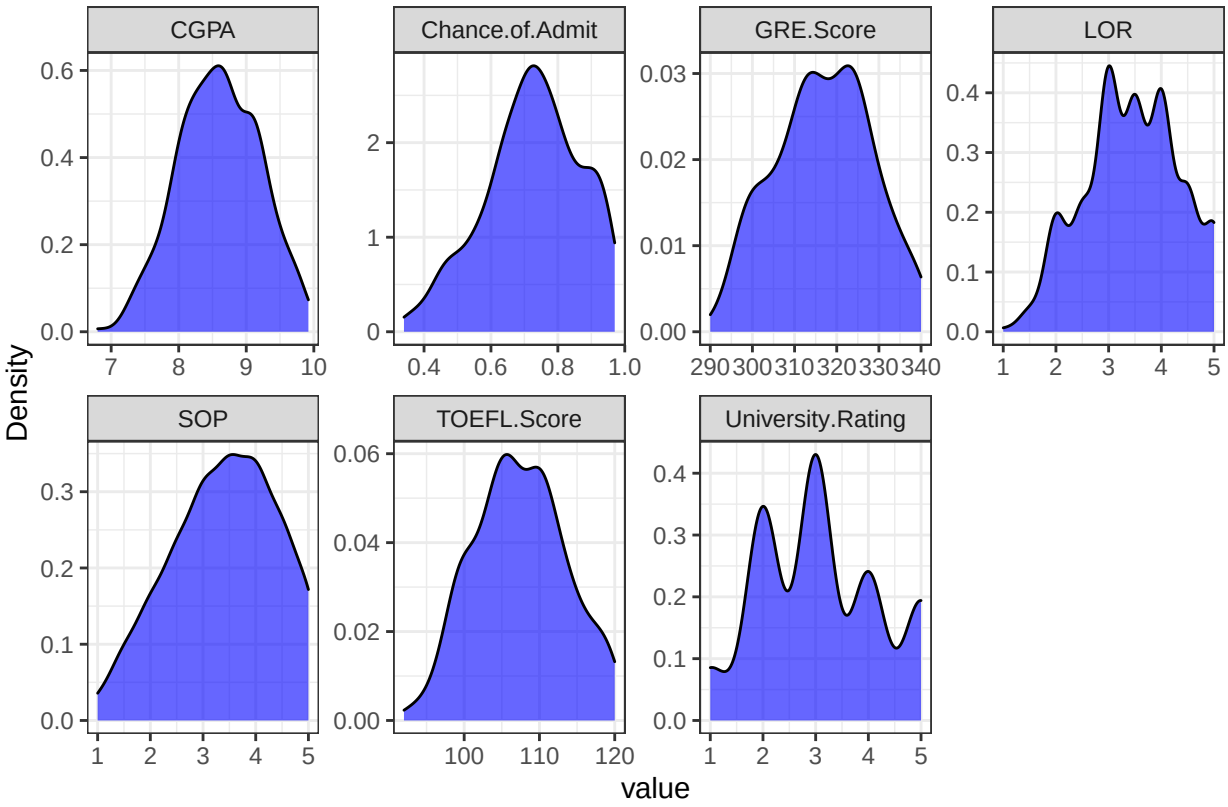
```
adm%>%ggplot(aes(x=Chance.of.Admit))+geom_density(fill="blue") + theme_bw()
```




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




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



Removing Extreme values

Histogram and boxplots of the 'Chance of 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 the 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 without removal of outliers.

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

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

```
#Function to check the outliers
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)
```

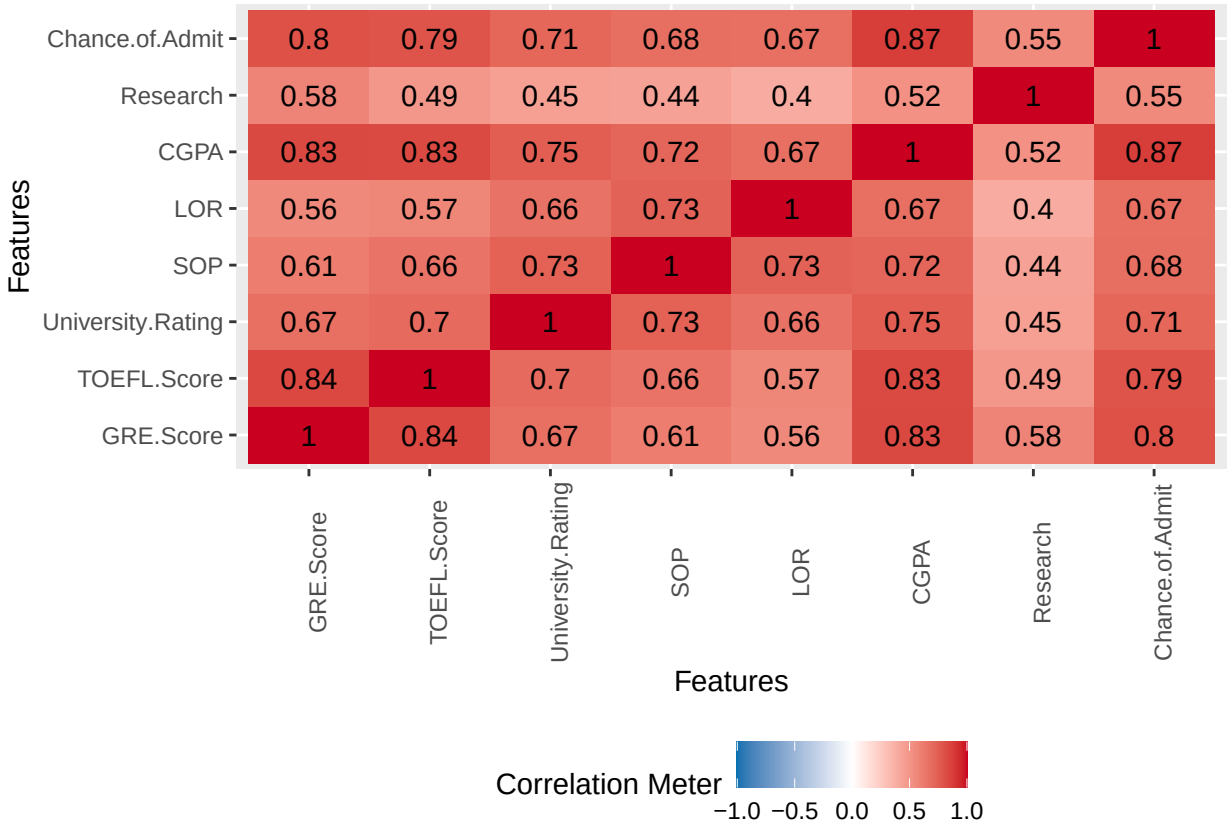
```
  IQR<-(Q3-Q1)
  left<- (Q1-(1.5*IQR))
  print(left)

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


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

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

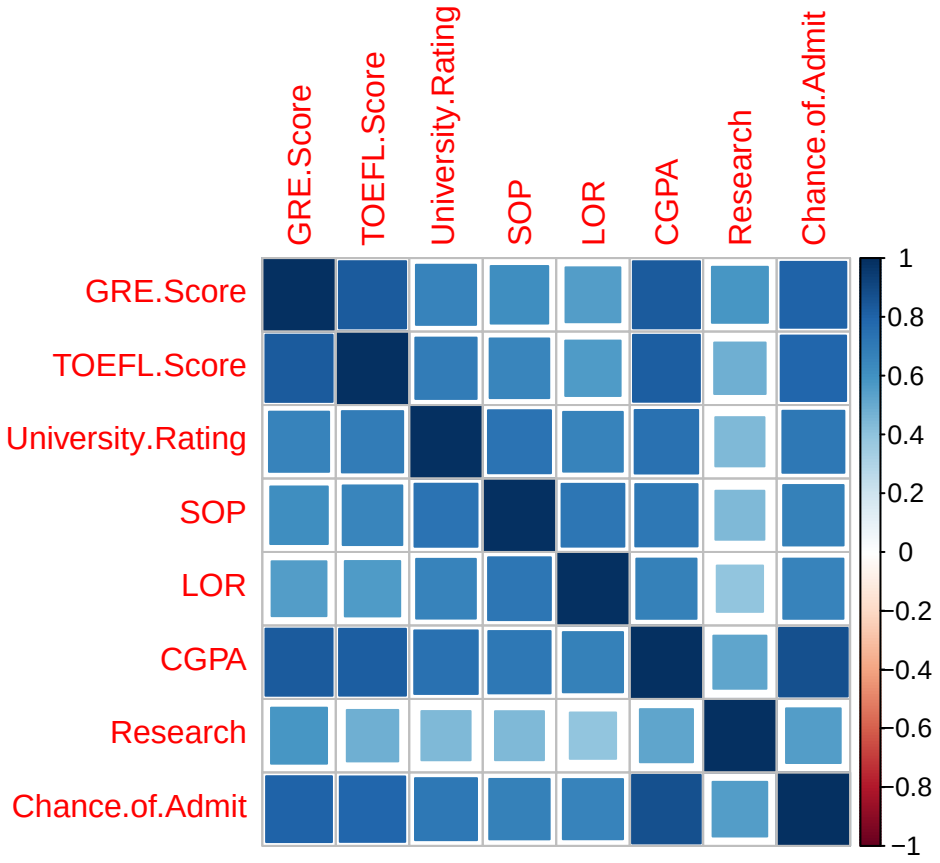
Check for correlations

The dataset is examined for correlations among the different features. There seems to a be strong correlation (>50%) between all of the features, such as GRE. Score, TOEFL. Score, University. Rating, SOP, LOR, CGPA and Research. The boxplot on the important features reveals a positive relationship. 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 a high probability of scoring high in TOEFL and potentially writing a worthy statement of purpose. Therefore, it is essential to examine partial correlation coefficients of these features 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)
```



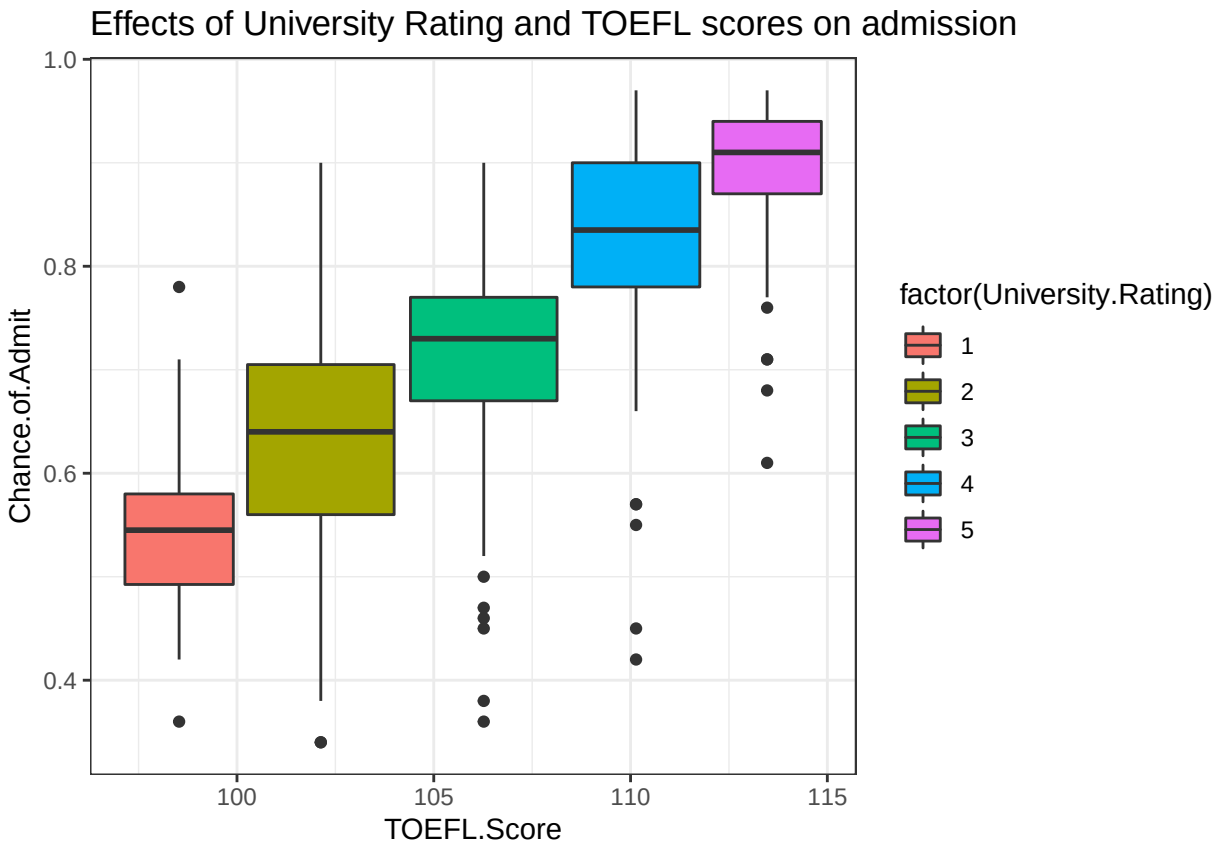

```
# plotting the correlation strength through size of squares.
corrplot(cor(adm_mat), method = "square")
```



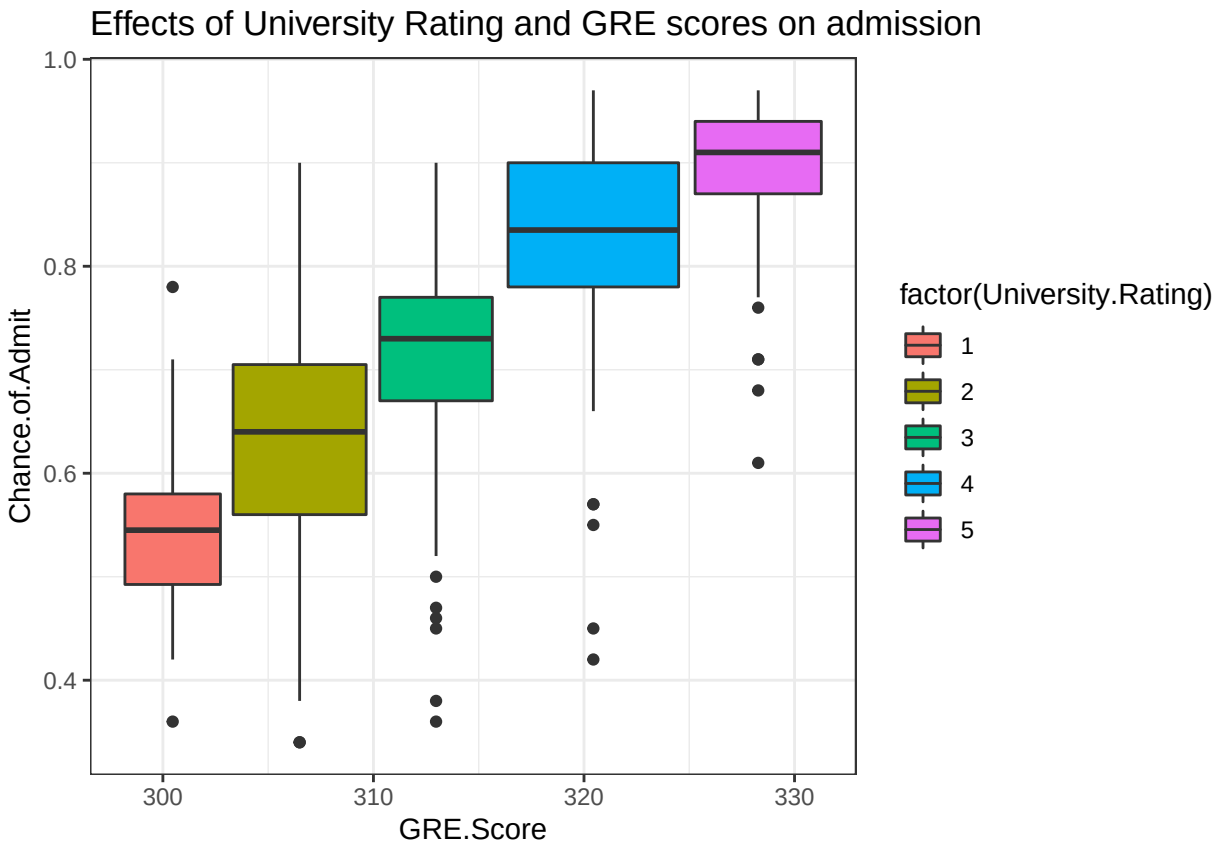
Visualizing the relationship

This step further explores the relationship by visualizing the spread of the features and presents the relationship between combination of features in influencing the admission rates. I explore the impacts of TOEFL. Score, GRE.Score and CGPA on Admission grouped by University Rating. The trend appears to be linear and there is strong evidence that these features are positively related to the admission rates, however their magnitudes vary by Universities.

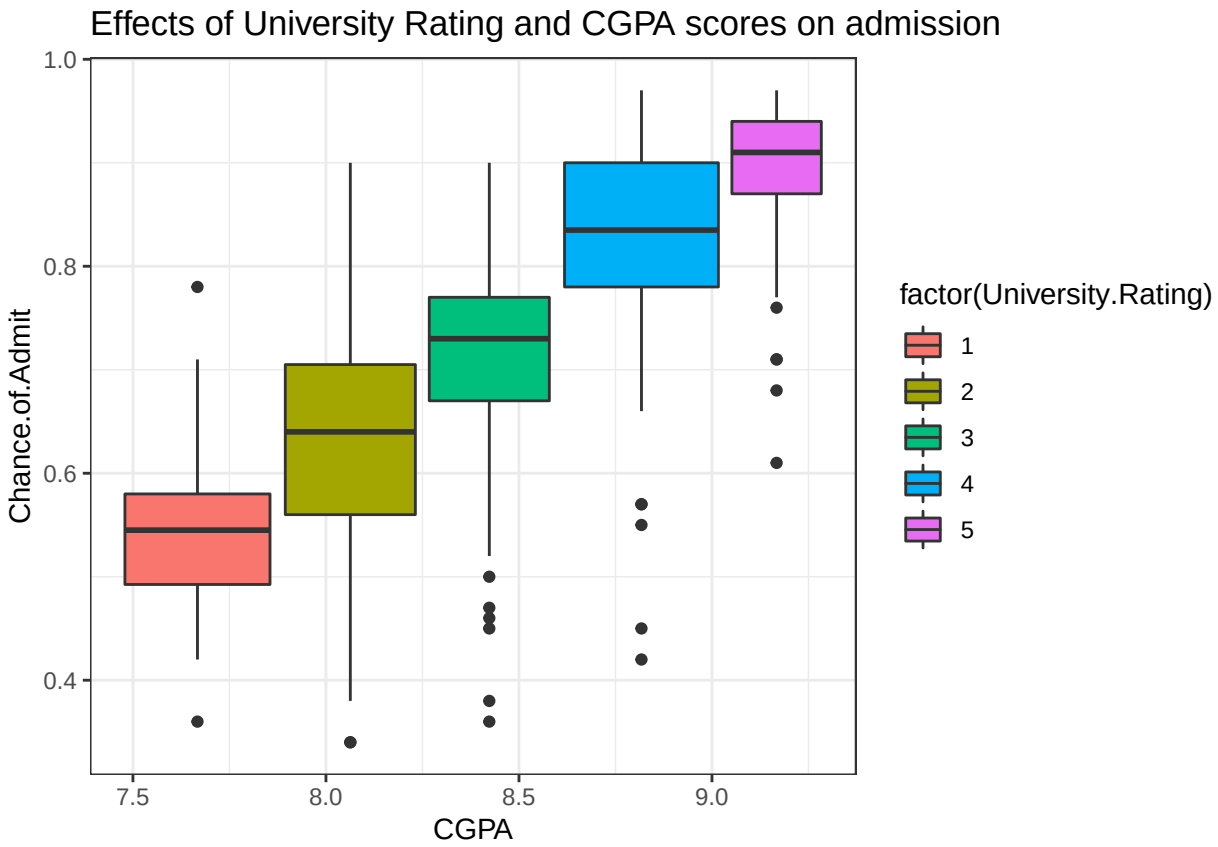
```
adm%>%
  ggplot(aes(x=TOEFL.Score,y=Chance.of.Admit, fill=factor(University.Rating))) +
  geom_boxplot() +theme_bw()+
  ggtitle("Effects of University Rating and TOEFL scores on admission")
```




```
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 individual attributes to the dependent variables of interest. Furthermore, partial correlation ensures that the confounding effects that exist in the variables are eliminated. This probability values from the partial correlation coefficient, shows that the SOP and University.Rating had no influence when partial correlation values were considered (p>0.05). Based on these findings, SOP and 'University.Rating' features will be cleansed from our dataset before the model development.

```
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 = "pandoc", digits=2,
      caption="Partial correlations  of the input dataset features")
```

Table 1: Partial correlations of the input dataset features

	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 = "pandoc", digits=2,
      caption="Probability values  of the input dataset features")
```

Table 2: Probability values of the input dataset features

	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

Model Development

Data Cleansing

Based on the partial correlation coefficient analysis, the SOP and the University Rating features are removed from the dataset. Only this dataset with reduced features will be further used for all of the modeling efforts.

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

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. Fifty percent of the cleansed data was chosen as the validation dataset at random (318) and the rest (82) was saved as the training dataset. This proportion was chosen

based on the strength of the features and their potential relationship with the admission rates. The features selected were GRE.Score, TOEFL.Score, University.Rating, LOR, CGPA and Research

```
set.seed(1, sample.kind = "Rounding")
```

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

```
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
```

```
## Warning: The `i` argument of ``[`()` can't be a matrix as of tibble 3.0.0.
## Convert to a vector.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

Model Performance Metrics

The performance of the models are evaluated through two metrics. First, the Root mean square error (RMSE) and second with R2, coefficient of determination, between the acutals and predicted values.

RMSE

$$RMSE = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}\left(\frac{yhat - y}{\sigma_i}\right)^2}$$

where, yhat is the predicted value of y, y is the actual value, n is the number of samples.

Results

Approach and Model Development

From the datatype of the dependent variable 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 dataset. 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.

Tuning the parameters

This section of the code was run for selecting the optimized parmeters for the Rborist. The pred-Fixed parameter (mtry in RandomForest) and the minNode size model parameters are checked. Given the limitation of the resources, this code is commented out.

```
# 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

The general linear model considers Chance.of.Admit as the dependent variable and all of the other features as the independent variables. The linear model summary showed that the GRE and TOEFL scores were not significantly correlated with the predicted variable. This seems unreasonable and potentially confounding or multicolinearity effects are masking their effects. Therefore, a stepwise regression procedure was implemented to identify the most influential factors. The results revealed that Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR + CGPA were chosen as the factors for the model. Accordingly, a lm.model.step was constructed. Note that the R2 values were similar on both of the models.

```
#General linear model
mod.lm=lm(Chance.of.Admit~., data=traindf)
summary(mod.lm)
```

```
##
## Call:
## lm(formula = Chance.of.Admit ~ ., data = traindf)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23633 -0.02649  0.00890  0.03769  0.15988
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.2737301  0.1744318  -7.302 7.24e-12 ***
## GRE.Score    0.0014463  0.0009021   1.603 0.110516
## TOEFL.Score  0.0020728  0.0014942   1.387 0.166967
```

```
## LOR          0.0240550  0.0070353   3.419 0.000766 ***
## CGPA         0.1431870  0.0169714   8.437 7.59e-15 ***
## Research     0.0104297  0.0115710   0.901 0.368516
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06315 on 193 degrees of freedom
## Multiple R-squared:  0.8146, Adjusted R-squared:  0.8098
## F-statistic: 169.6 on 5 and 193 DF,  p-value: < 2.2e-16
```

```
#predicting results from the model
pred.lm=predict(mod.lm, newdata=valdf)
RMSE(pred.lm, valdf$Chance.of.Admit)
```

```
## [1] 0.0655762
```

```
#Stepwise Regression procedure
stepAIC(mod.lm, direction="both")
```

```
## Start:  AIC=-1093.45
## Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR + CGPA + Research
##
##               Df Sum of Sq     RSS     AIC
## - Research     1  0.003240 0.77298 -1094.6
## - TOEFL.Score  1  0.007675 0.77742 -1093.5
## <none>                     0.76974 -1093.5
## - GRE.Score    1  0.010251 0.77999 -1092.8
## - LOR          1  0.046627 0.81637 -1083.7
## - CGPA         1  0.283897 1.05364 -1033.0
##
## Step:  AIC=-1094.61
## Chance.of.Admit ~ GRE.Score + TOEFL.Score + LOR + CGPA
##
##               Df Sum of Sq     RSS     AIC
## - TOEFL.Score  1  0.007085 0.78007 -1094.8
## <none>                     0.77298 -1094.6
## + Research     1  0.003240 0.76974 -1093.5
## - GRE.Score    1  0.014482 0.78747 -1092.9
## - LOR          1  0.048971 0.82195 -1084.4
## - CGPA         1  0.295041 1.06802 -1032.3
##
## Step:  AIC=-1094.79
## Chance.of.Admit ~ GRE.Score + LOR + CGPA
##
##               Df Sum of Sq     RSS     AIC
## <none>                     0.78007 -1094.8
## + TOEFL.Score  1   0.00709 0.77298 -1094.6
```

```
## + Research     1   0.00265 0.77742 -1093.5
## - GRE.Score    1   0.03368 0.81375 -1088.4
## - LOR          1   0.05524 0.83531 -1083.2
## - CGPA         1   0.35233 1.13240 -1022.6

##
## Call:
## lm(formula = Chance.of.Admit ~ GRE.Score + LOR + CGPA, data = traindf)
##
## Coefficients:
## (Intercept)    GRE.Score          LOR         CGPA
##   -1.367540     0.002218     0.025851     0.151528
```

```
#Predicting results from stepwise regression
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
## -0.243878 -0.026094  0.007705  0.037273  0.156825
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.3432107  0.1563998  -8.588 2.87e-15 ***
## GRE.Score    0.0016591  0.0008703   1.906 0.058068 .
## TOEFL.Score  0.0019876  0.0014905   1.333 0.183936
## LOR          0.0245705  0.0070086   3.506 0.000565 ***
## CGPA         0.1449726  0.0168472   8.605 2.58e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.06312 on 194 degrees of freedom
## Multiple R-squared:  0.8139, Adjusted R-squared:   0.81
## F-statistic:   212 on 4 and 194 DF,  p-value: < 2.2e-16
```

Machine Learning Models

For machine learning, the following models were chosen: Rborist, which is a fast implementation of the random forest, k-nearest neighbours, neural net implmentation of glm (glmnet), gradient boost algorithm (xgbLinear), bayesian regularized neural network (brnn) and ridge regression. All of these models were run in the CareEnsemble package. This packages allows simulataneous runs of various ML algorithms, collect and integrate the results and above all, conducts an ensemble

evaluation of the model results based on their performance. In this case, the RMSE will be used to evaluate the performance of the models by defaults and the results are weighted according to the model RMSE values.

```
set.seed(1, sample.kind = "Rounding")
```

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

```
cl=makePSOCKcluster(detectCores()-1) #setting up clusters for parallel processing
registerDoParallel(cl) # register the clusters for parellel processing

my.con=trainControl(method="cv", # choose cross validation
                    number=3, # number of times the process is run
                    savePredictions = "final", allowParallel = T)
models=caretList(Chance.of.Admit~., data=traindf,
trainControl=my.con,
methodList = c("Rborist",  # list of ML models chosen
               "knn",
               "glmnet",
               "xgbLinear",
               "brnn",
               "ridge"),
continue_on_fail = T) # making sure all models are running without error
```

```
## 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.
```

```
## [22:17:00] WARNING: amalgamation/../src/objective/regression_obj.cu:170: reg:linear is now de
## [22:17:00] 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= 6.7034     alpha= 1.6038   beta= 12.0324
```

```
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      0.08063889  0.7109212  0.05853937
##   0e+00   0e+00  100      0.08063882  0.7109211  0.05853927
##   0e+00   0e+00  150      0.08063874  0.7109211  0.05853918
##   0e+00   1e-04   50      0.08058688  0.7117638  0.05854805
##   0e+00   1e-04  100      0.08058681  0.7117637  0.05854795
##   0e+00   1e-04  150      0.08058673  0.7117637  0.05854785
##   0e+00   1e-01   50      0.07457953  0.7465389  0.05386809
##   0e+00   1e-01  100      0.07457953  0.7465389  0.05386809
##   0e+00   1e-01  150      0.07457953  0.7465389  0.05386809
##   1e-04   0e+00   50      0.08059165  0.7113760  0.05860778
##   1e-04   0e+00  100      0.08059158  0.7113759  0.05860768
##   1e-04   0e+00  150      0.08059151  0.7113758  0.05860759
##   1e-04   1e-04   50      0.08057632  0.7115631  0.05863154
##   1e-04   1e-04  100      0.08057625  0.7115630  0.05863144
##   1e-04   1e-04  150      0.08057617  0.7115630  0.05863134
##   1e-04   1e-01   50      0.07457143  0.7465915  0.05385579
##   1e-04   1e-01  100      0.07457143  0.7465915  0.05385579
##   1e-04   1e-01  150      0.07457143  0.7465915  0.05385579
##   1e-01   0e+00   50      0.07871170  0.7225830  0.05702163
##   1e-01   0e+00  100      0.07871164  0.7225830  0.05702156
##   1e-01   0e+00  150      0.07871158  0.7225829  0.05702149
##   1e-01   1e-04   50      0.07872647  0.7221352  0.05712342
##   1e-01   1e-04  100      0.07872644  0.7221352  0.05712337
##   1e-01   1e-04  150      0.07872640  0.7221352  0.05712333
##   1e-01   1e-01   50      0.07439612  0.7473093  0.05330426
##   1e-01   1e-01  100      0.07439612  0.7473093  0.05330426
##   1e-01   1e-01  150      0.07439612  0.7473093  0.05330426
##
## 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 = 0.1, 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  0.08658990  0.6601111  0.06548773
##   7  0.08444153  0.6718092  0.06346668
##   9  0.08393995  0.6743269  0.06253178
##
## 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          0.06905907  0.7774871  0.04954272
##   3          0.07031808  0.7701666  0.05065089
##   5          0.07271757  0.7561023  0.05272761
##
## 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   0.0002568282  0.06495079  0.8067987  0.04742797
##   0.10   0.0025682822  0.06490501  0.8071440  0.04730252
##   0.10   0.0256828219  0.06613262  0.8042026  0.04734771
##   0.55   0.0002568282  0.06495173  0.8068342  0.04740708
##   0.55   0.0025682822  0.06497479  0.8069613  0.04731463
##   0.55   0.0256828219  0.06797812  0.8074955  0.04961532
##   1.00   0.0002568282  0.06496929  0.8067131  0.04742983
##   1.00   0.0025682822  0.06508041  0.8066258  0.04737834
##   1.00   0.0256828219  0.07188462  0.8010876  0.05383877
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.1 and lambda = 0.002568282.
```

```
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        0.06508188  0.8054340  0.04721833
##   2        0.06648537  0.7965571  0.04807210
##   3        0.06747989  0.7899181  0.04880175
##
## RMSE was used to select the optimal model using the smallest value.
## The final value used for the model was neurons = 1.
```

```
stopCluster(cl)
registerDoSEQ()
rm(cl) # removing cluster
varImp(models$Rborist) # variable importance for Rborist
```

```
## Rborist variable importance
##
```

```
##              Overall
## CGPA          100.00
## GRE.Score      95.78
## TOEFL.Score    44.33
## LOR            13.91
## Research        0.00
```

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

```
## glmnet variable importance
##
##               Overall
## CGPA         100.0000
## LOR           17.4840
## Research       7.2316
## TOEFL.Score    0.5354
## GRE.Score      0.0000
```

Feature Reduction

The varImp procedure from the ensemble reveals different feature importance based on the model. This step attempts to explore whether a feature reduction is possible to achieve without compromising the performance of the model using the random Forest function. A recursive feature reduction method is implemented to check the feature reduction opportunities.The recursive feature eliminated Research feature from the modelo. Since the ensemble of models prefer to have different input features, all features were left in the model.

```
set.seed(1, sample.kind = "Rounding")
```

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

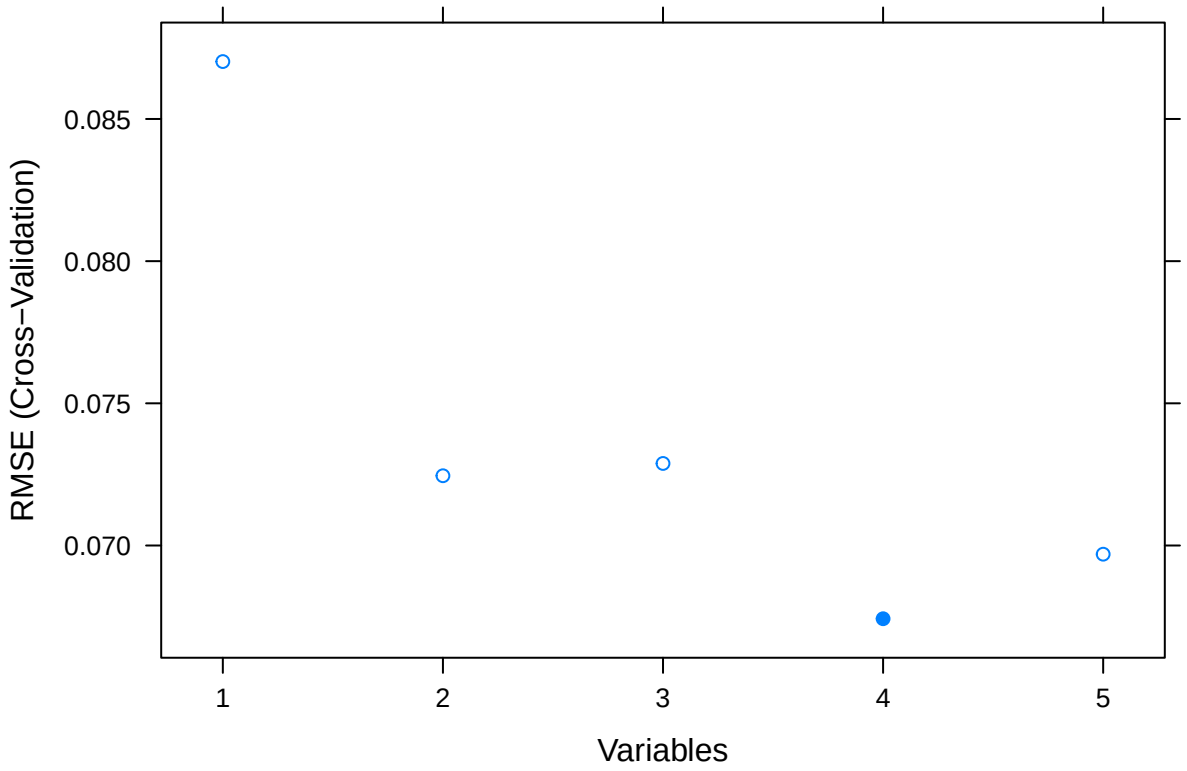
```
cl=makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)

ctrl=rfeControl(functions = rfFuncs, #random forest function
           method = "cv", # cross validation
           number = 2, # times
           verbose = F)
subsets=c(1:5)
registerDoSEQ() # avoid warning of parallel clusters not existing
lmProfile=rfe(as.matrix(traindf[,-6]),
              as.matrix(traindf[,6]),
              sizes=subsets,
              rfeControl=ctrl) # recursive feature elimination
```

```
## Warning in rfout$mse/(var(y) * (n - 1)/n): Recycling array of length 1 in vector-array arithm
##   Use c() or as.vector() instead.
```

```
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 0.08702   0.6491 0.06394 0.024352    0.16673 0.019171
##          2 0.07245   0.7511 0.05325 0.007942    0.04575 0.003054
##          3 0.07289   0.7493 0.05217 0.008247    0.04638 0.002315
##          4 0.06742   0.7857 0.04788 0.009911    0.05166 0.002970        *
##          5 0.06969   0.7769 0.05048 0.009760    0.04848 0.003595
##
## The top 4 variables (out of 4):
##    CGPA, GRE.Score, TOEFL.Score, LOR
```

```
plot(lmProfile)
```

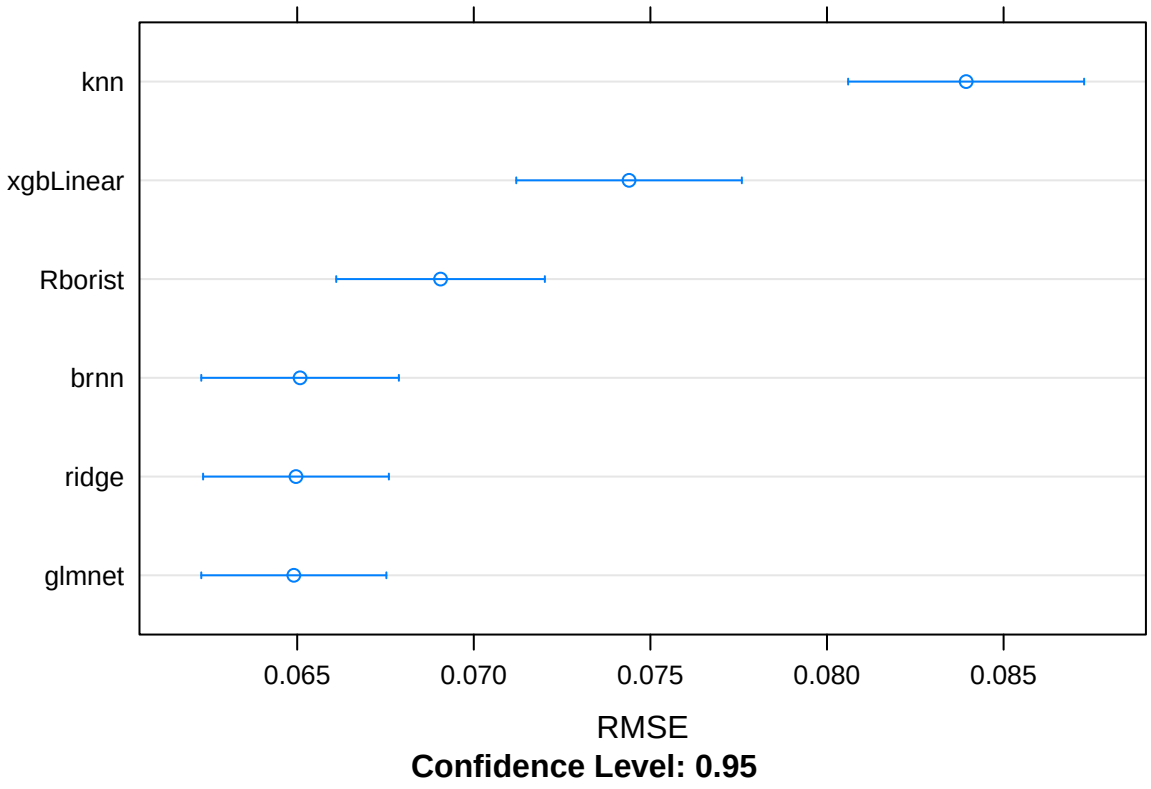


The lmProfile chose 4 models. Nonetheless, owing to multiple models involved in the ensemble, I have chosen to include all five features in the model.

Cross validation

The following plot demonstrates the performance of the machine learning models with the cross-validation dataset. The brnn and glmnet both registered the lowest RMSE values in this step. The knn model registerd the highest RMSE value. Glmnet which is a form of the general linear model outperformed all other models in the cross-validation.

```
cl=makePSOCKcluster(detectCores()-1)
registerDoParallel(cl)
#retrieve resamples from cross validation
resamples<-resamples(models)
dotplot(resamples, metric="RMSE")
```




```
summary(resamples)
```

```
##
## Call:
## summary.resamples(object = resamples)
##
## Models: Rborist, knn, glmnet, xgbLinear, brnn, ridge
## Number of resamples: 25
##
## MAE
```

```
##                 Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## Rborist   0.04087272 0.04650832 0.04837651 0.04954272 0.05306860 0.06023343
## knn       0.05062886 0.05933917 0.06375611 0.06253178 0.06639950 0.07113544
## glmnet    0.04017813 0.04434533 0.04664781 0.04730252 0.04891365 0.05898725
## xgbLinear 0.04162523 0.04886742 0.05361918 0.05330426 0.05739933 0.06370941
## brnn      0.03957659 0.04443392 0.04668988 0.04721833 0.04853141 0.05911275
## ridge     0.04039075 0.04496411 0.04685090 0.04749992 0.04864804 0.05971269
##           NA's
## Rborist      0
## knn          0
## glmnet       0
## xgbLinear    0
## brnn         0
## ridge        0
##
## RMSE
##                 Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## Rborist   0.05888242 0.06372738 0.06814136 0.06905907 0.07374948 0.08176403
## knn       0.06539133 0.08009600 0.08325039 0.08393995 0.08835281 0.09735443
## glmnet    0.05327690 0.05940444 0.06664181 0.06490501 0.06906988 0.07601822
## xgbLinear 0.06361614 0.06704974 0.07401360 0.07439612 0.08120587 0.08926259
## brnn      0.05298541 0.05856899 0.06685617 0.06508188 0.06916823 0.07734920
## ridge     0.05347100 0.05981185 0.06631941 0.06496590 0.06934683 0.07660255
##           NA's
## Rborist      0
## knn          0
## glmnet       0
## xgbLinear    0
## brnn         0
## ridge        0
##
## Rsquared
##                Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## Rborist   0.6565284 0.7323901 0.7859326 0.7774871 0.8201107 0.8508976    0
## knn       0.5287886 0.6358892 0.6652869 0.6743269 0.7100946 0.8093582    0
## glmnet    0.6955158 0.7847301 0.8143828 0.8071440 0.8376338 0.8644957    0
## xgbLinear 0.6395155 0.7255789 0.7519061 0.7473093 0.7941205 0.8330314    0
## brnn      0.6895725 0.7784176 0.8148005 0.8054340 0.8355610 0.8667472    0
## ridge     0.6937151 0.7860366 0.8144191 0.8066642 0.8386284 0.8631465    0
```

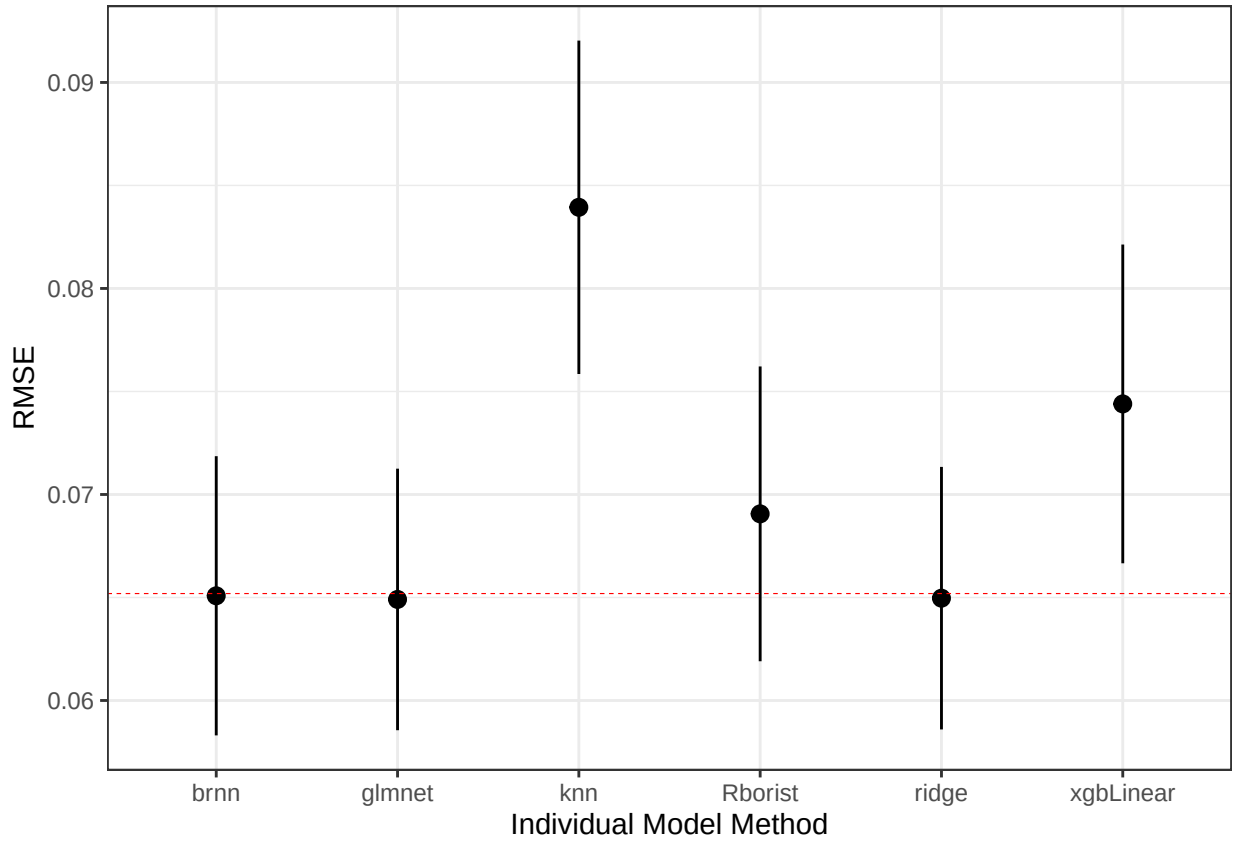
Ensemble models

Ensembling is a technique where the model results are weighed according to their performance. The caretEnsemble function performs this step and we can predict the performance of the ensemble models with the cross-validation dataset. The performance results are similar to those found from the individual models.

```
cl=makePSOCKcluster(detectCores())
registerDoParallel(cl)
#
ens=caretEnsemble(models, metric="RMSE",
                  trControl=my.con) # Run Ensemble model to gather the best result
summary(ens)
```

```
## The following models were ensembled: Rborist, knn, glmnet, xgbLinear, brnn, ridge
## They were weighted:
## -0.0129 0.0473 -0.0222 1.7555 0.0816 0.3364 -1.1817
## The resulting RMSE is: 0.0652
## The fit for each individual model on the RMSE is:
##     method       RMSE      RMSESD
##    Rborist 0.06905907 0.007154672
##        knn 0.08393995 0.008092117
##     glmnet 0.06490501 0.006347584
##  xgbLinear 0.07439612 0.007735828
##       brnn 0.06508188 0.006776304
##      ridge 0.06496590 0.006371914
```

```
plot(ens)
```



Validation

In this step, the model are evaluated for their performance against a new and independent dataset. 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.

```
######### Results- Validation ############

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

#Predicted from each ML model

 predicted.Rborist=predict(models$Rborist, newdata=valdf)
 predicted.knn=predict(models$knn, newdata=valdf)
 predicted.glmnet=predict(models$glmnet, newdata=valdf)
 predicted.xgbLinear=predict(models$xgbLinear, newdata=valdf)
 predicted.brnn=predict(models$brnn, newdata=valdf)
 predicted.ridge=predict(models$ridge, newdata=valdf)




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


 # Construct the table to output th results.
 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))->rmses

 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))->r2s

 Results=data_frame(Model="Linear-Stepwise-Selected",
                    RMSE=round(RMSE(pred.lm.step, valdf$Chance.of.Admit), 3),
                    R2=round(cor(pred.lm.step, valdf$Chance.of.Admit), 3))
```

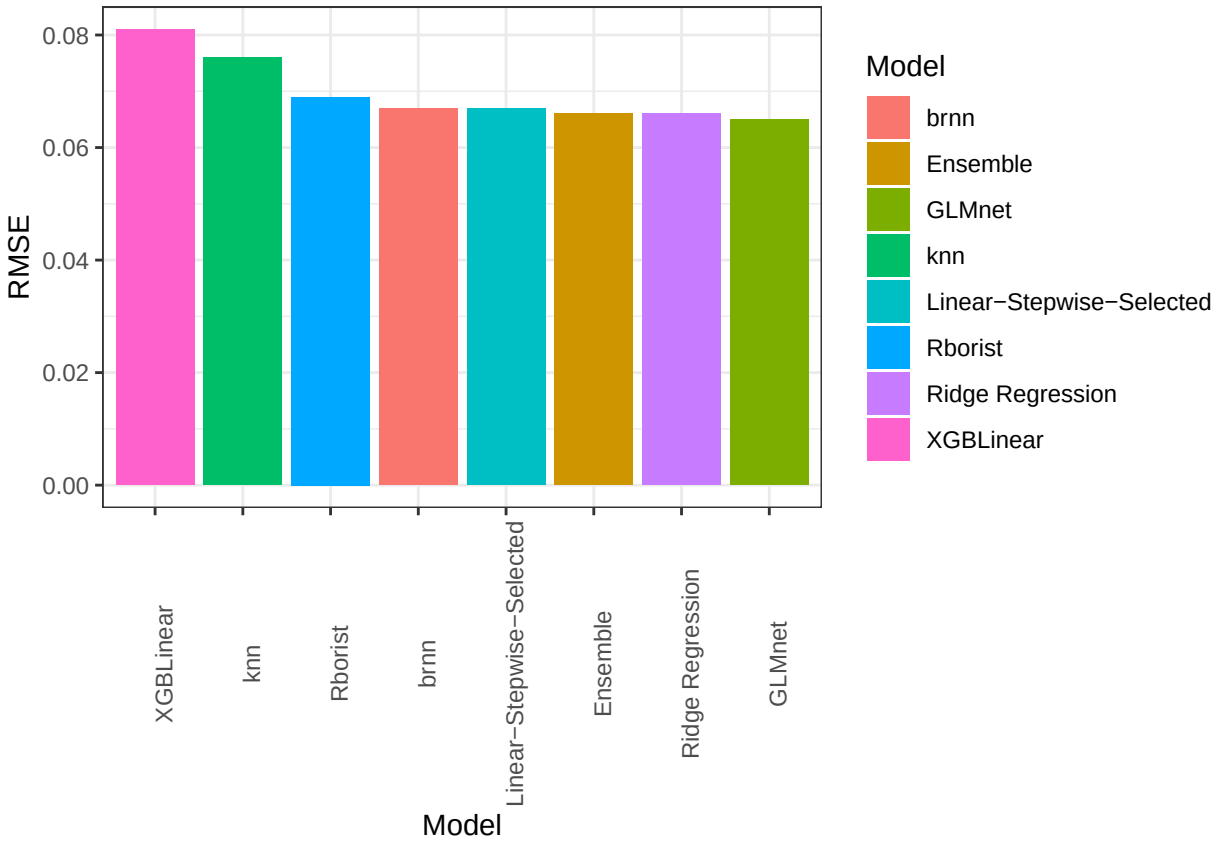
```
## Warning: `data_frame()` is deprecated as of tibble 1.1.0.
## Please use `tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
Results[2,]=list(Model="knn",
                    RMSE=round(RMSE(predicted.knn, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.knn, valdf$Chance.of.Admit), 3))
Results[3,]=list(Model="GLMnet",
                    RMSE=round(RMSE(predicted.glmnet, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.glmnet, valdf$Chance.of.Admit), 3))

Results[4,]=list( Model="XGBLinear",
                    RMSE=round(RMSE(predicted.xgbLinear, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.xgbLinear, valdf$Chance.of.Admit), 3))
Results[5,]=list(Model="brnn",
                    RMSE=round(RMSE(predicted.brnn, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.brnn, valdf$Chance.of.Admit), 3))
Results[6,]= list(Model="Ridge Regression",
                    RMSE=round(RMSE(predicted.ridge, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.ridge, valdf$Chance.of.Admit), 3))

Results[7,]= list(Model="Ensemble",
                    RMSE=round(RMSE(predicted.ens, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.ens, valdf$Chance.of.Admit), 3))
Results[8,]=list(Model="Rborist",
                    RMSE=round(RMSE(predicted.Rborist, valdf$Chance.of.Admit), 3),
                    R2=round(cor(predicted.Rborist, valdf$Chance.of.Admit), 3))




kable(Results, format="pandoc",
      digits=3, caption = "Performance of the different models: RMSE and R2 values" )
```

Table 3: Performance of the different models: RMSE and R2 values

Model	RMSE	R2
Linear-Stepwise-Selected	0.067	0.882
knn	0.076	0.840
GLMnet	0.065	0.887
XGBLinear	0.081	0.825
brnn	0.067	0.879
Ridge Regression	0.066	0.885
Ensemble	0.066	0.884
Rborist	0.069	0.873

```
Results%>% ggplot(aes(reorder(Model, -RMSE), RMSE, fill=Model))+geom_bar(stat="identity")+theme_
  theme(axis.text.x = element_text(angle=90))+xlab("Model")
```



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. GLMnet outperformed all of the other models, with a RMSE of 0.065 and a R2 value of 0.887. This model was closely followed by Ridge regression and Ensemble. It is clear that, in this case, invidividual models can perform better than the ensemble technique.

Limitations

Owing to the limited sample sizes, this model has its' limitations. Tuning of parameters for invidividual models can offer opportunity to improve the performance of the models.