Predicting Player Rating in FIFA

Junqi Liao(20650701), Kaiwen Dong(20626786), Raymond Tan(20611791)

07/08/2019

Contents

Abstract

In this article, we decide to carry out the prediction on the European soccer player rating. The data is extracted from the database sqlite downloaded on Kaggle. The main goal is to use various statistics learning techniques we learn in class (GAM, Smoothing Spline, Regression Tree etc.), predict players' overall_rating based on the model we build and see which method we use has the best performance from (e.g. MSE, MSPE etc). Before implementing the algorithms, we will go through each mathematical terminology for better understanding behind its corresponding algorithm. In particular, we will discuss further improvement after we evaluate the model performance.

Introduction

Data Preprocessing

The European soccer rating prediction problem on is based on a past Kaggle competition (https://www.kaggle.com/hugomathien/soccer). This data is extracted from the European Soccer Database with 25k+ matches, 10k+ players and teams attributes for European professional football. Regarding the database, it contains everything from V1 + some fixes (missing games, full team names) and a new table with team attribute. We preprocess the dataset from the file database.sqlite extracted in the data directory. Basically, we join two tables player and player_stats into rating_potential and produce the output of rating_potential.csv file. Noted that the variable gk_reflexes has a partial samples in the range of 65 to 85. In order to reduce the variability and maintain the normality, we basically transform that partial samples into the range of 0 to 20. The response variable overall_rating will not be affected too much because it represents the overall average of each explanatory variable and the average will not change a lot by law of large number. We decide to use this csv file as our main data source for this project. Noticed that we will also use the variable overall_ratingas our target, other attributes as our explanatory variables.

Problem

In this project, we want to apply six modern statistics learning techniques in this article, which are multiple linear regression, LASSO Regression and ridge regression based on the assumption that the model is linear and smooth using general additive model (GAM), gradient boosting (tree-based) and regression tree based on the assumption that the model is non-parametric respectively. Here, we list the assumption and techniques below:

- Linear Model
  - Multiple linear regression
  - LASSO regression
  - Ridge regression
- Non-parametric model
  - General additive model (GAM)
  - Gradient Boosting (Tree-based)
  - Regression tree

Exploratory Data Analysis

Dimension of the Dataframe

We observe that the dimension of the dataframe rating_potential is 10390×33. But we generally consider the subdataframe from the second column to the last column since the second column corre- sponds to the

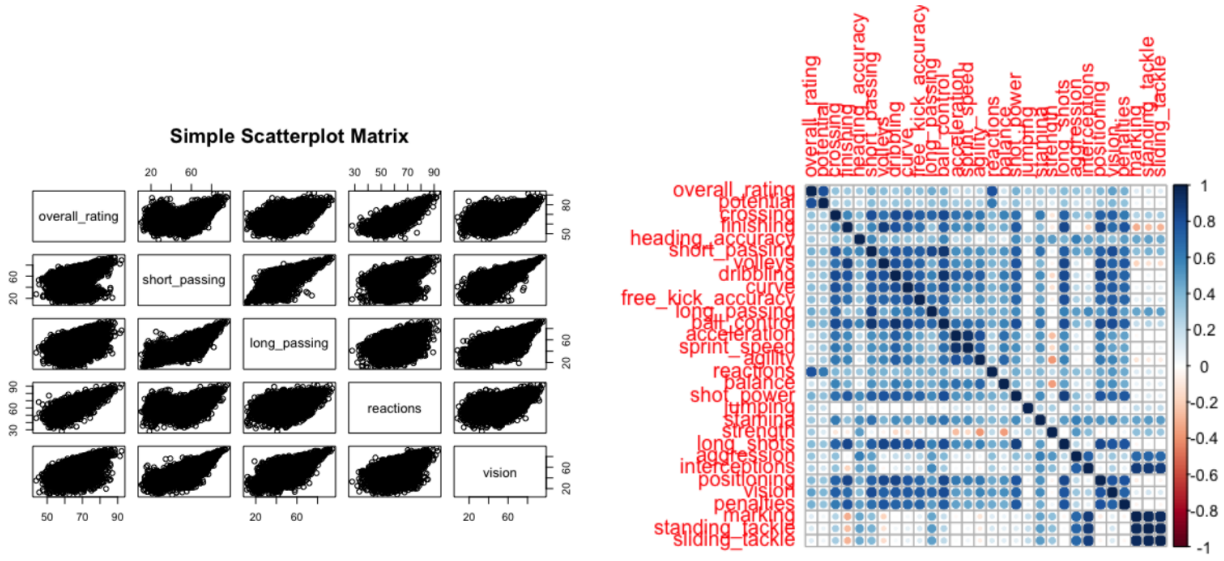


Figure 1: Scatter plot matrix and correlation plot

response variable overall_rating and the subdataframe from the third column to the last column corresponds to input dataframe.

Relationship between Exploratory and Response

We first read in the table rating_potential and display the correlation plot for each explanatory vari- able and corresponding response overall_rating.This will give us some observation on the dataset and perspective on how to build the model, but this correlation only assumes the model is linear. We also want to observe any relationship between the explanatory variable and response variable if the model is non-linear. So we want to build both linear and non-linear model with different assumptions.

From the above scatter plot matrix and correlation plot, we notice that short_passing,long_passing, reactions and vision have strong correlation with overall_rating. This indicates that those ex- planatory variables mentioned above might have a great impact on the response variable overall_rating if the model is linear in parameters.

Linear Models

In this section, we will implement the algorithms based on the assumption that the model is linear in parameters and each variable is identically independent distributed (i.i.d). Before we dig into the regression model, we did some Research and summarized each model we are going to use.

Multiple Linear Regression

Definition

Multiple linear regression is the most common form of linear regression analysis. As a predictive analysis, the multiple linear regression is used to explain the relationship between one continuous dependent variable and

two or more independent variables. The independent variables can be continuous or categorical (dummy coded as appropriate).

Analysis

First we need to do variable selection for multiple linear regression. Since there are more than 30 variables in the model, stepwise regression from both ways is chosen to do the selection since it's more efficient than the best subset selection.

The model involves 26 variables which is still a large model. The mse and mspe are:

```
##           mse     mspe
## [1,] 6.249106 6.486335
```

```
##        potential        reactions         strength           vision
##         1.933699         2.388514         2.281626         5.292583
##     acceleration      gk_reflexes         crossing    short_passing
##         8.789883         1.130511         5.486878        11.862488
##          jumping     long_passing   sliding_tackle        penalties
##         1.394609         5.583540        20.423159         3.715136
##        dribbling    interceptions          volleys       aggression
##        11.695959         7.095458         5.644381         3.412227
##          stamina       long_shots       shot_power            curve
##         3.091090         7.026522         5.434237         4.827894
##          marking  standing_tackle      positioning     ball_control
##        18.575390        27.344903         6.575502        16.637400
## heading_accuracy     sprint_speed
##         3.808084         8.009261
```

which are acceptable. But when we check the variance inflation factor (VIF), the problem of collinearity appears. After we check the variable importance it is quite hard to determine which variable to drop, and thus we choose another way to select variables, which is LASSO.

```
##           mse     mspe
## [1,] 6.275399 6.502008
```

The model we get turns out to be a even larger model than the previous one, containing 22 variables. Since both mse and mspe are close the previous model, we decides to drop some variables from the model based on VIF and their variable importance. The VIF and variable importance are:

```
vif(soccer.lasso)
```

```
##        potential         crossing heading_accuracy    short_passing
##         1.882134         5.391247         3.397916         9.921345
##          volleys        dribbling            curve     long_passing
##         5.178283         8.733323         4.672887         5.434887
##     acceleration     sprint_speed        reactions       shot_power
##         8.763785         7.956275         2.365980         4.375399
##          jumping          stamina         strength       aggression
##         1.391110         3.028884         2.275544         3.394539
##    interceptions           vision        penalties  standing_tackle
##         6.927445         4.834434         3.539196        22.684401
##   sliding_tackle      gk_reflexes
##        18.328067         1.125852
```

It is obvious that there's a huge problem of collinearity between standing tackle and sliding tackle, we drop the less important one which is standing tackle and get the following results:

```
##               mse     mspe
## original 6.275399 6.502008
## updated  6.278059 6.506281
```

```
##        potential         crossing heading_accuracy    short_passing
##         1.879539         5.389306         3.365869         9.819774
##          volleys        dribbling            curve     long_passing
##         5.127665         8.731992         4.672429         5.431697
##     acceleration     sprint_speed        reactions       shot_power
##         8.760854         7.954810         2.337496         4.375395
##          jumping          stamina         strength       aggression
##         1.388328         3.023615         2.266947         3.385045
##    interceptions           vision        penalties   sliding_tackle
##         6.149529         4.829637         3.539185         6.087698
##      gk_reflexes
##         1.123539
```

It seem that there're still some terms with a high VIF, but since all of them have a relatively high variable importance, we decide not to drop them and instead get rid of those variables with low importance to decrease the variability of the model. After some tuning the final multiple linear regression model we get is:

$$\begin{aligned} Overall\_rating = potential + crossing + short\_passing + volleys + \\ dribbling + long\_passing + sprint\_speed + reactions + \\ jumping + stamina + strength + Interceptions + \\ vision + penalties + sliding\_tackle + gk\_reflexes \end{aligned}$$

```
##               mse     mspe
## original 6.275399 6.502008
## updated  6.306575 6.482767
```

with a similar predicting error and a lower testing error compare to the original model, and less collinearity compares to the model selected with stepwise regression.

LASSO Regression

Definition

Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of coefficients. This type of regularization can result in sparse models with few coefficients; Some coefficients can become zero and eliminated from the model. Larger penalties result in coefficient values closer to zero, which is the ideal for producing simpler models.

$$min \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \cdot \sum_{j}^{p} |\beta_j|$$

Analysis

We want to compare both of the following cases when we use LASSO:

1. The dataset for explanatory variables is not scaled.
2. The dataset for explanatory variables is scaled.

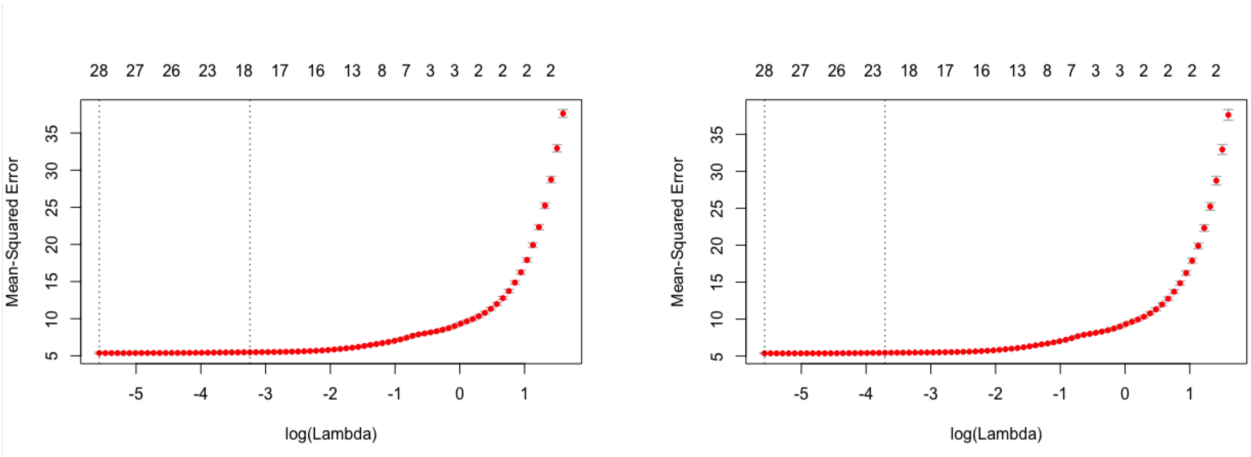


Figure 2: Optimal lambda for xtrain without scaling(Right) and with scaling(Left)

We generally use glmnet package to do cross validation on the training dataset. During this pro- cess, we split the dataset into 80/20, then we estimate the optimal lambda with lambda.min and lambda.1se.We then plot the optimal lambda for the first case and the second case.

From the following figure, we observe that the optimal lambda λ_{1se} for the first the case is relatively larger than the λ_{1se} for the second case. The optimal lambda λ_{min} for both cases are the same. Next, we make a table of MSE and MSPE in order to compare the performance of the models for both cases: We observe that MSE for both cases are smaller than MSPE. MSE and MSPE with scaling are relatively larger than MSE and MSPE without scaling. This indicates that sometimes scaling the explanatory dataset might not be a good choice. MSE and MSPE for both cases are relatively high. So the model might be underfitting. Let's see if the MSE and MSPE would be smaller if we construct non-parametric model to predict it.

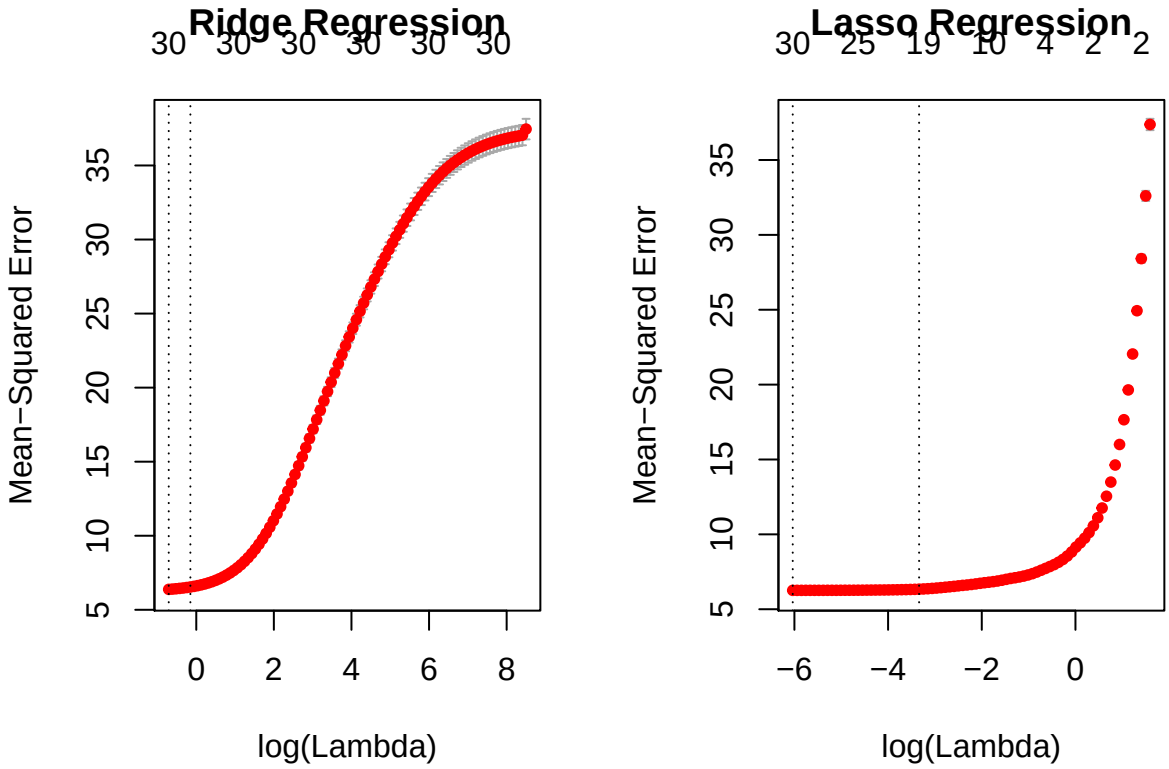
Ridge Regression

Definition

Ridge regression performs L2 regularization, which adds a penalty equal to the square of the magnitude of coefficients. When $\lambda \to 0$, ridge regression results least squares regress(RSS). As $\lambda \to \infty$, all coefficients in ridge regression are shrunk by the same factor. Unlike LASSO regression, ridge regression will not yield to a sparse model since all coefficients are shrunk to zero.

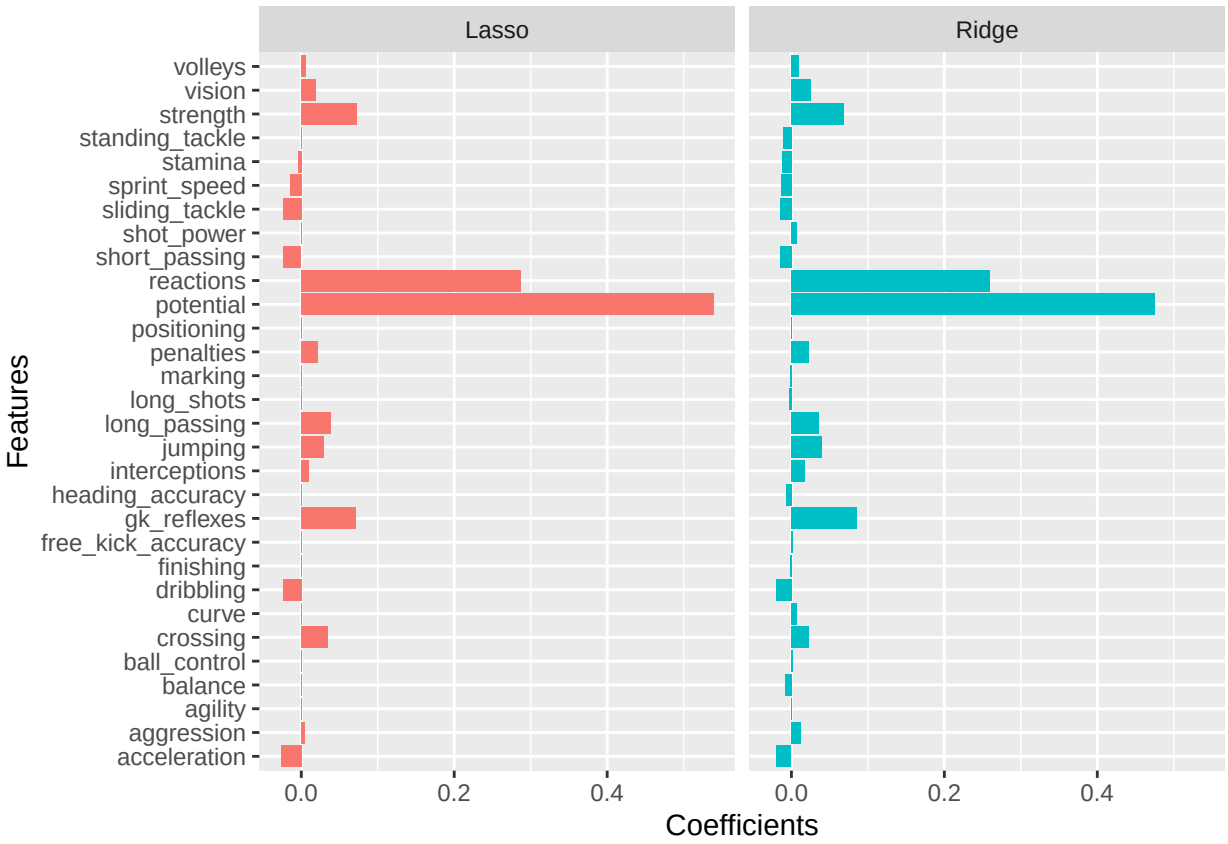
$$min \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2 + \lambda \cdot \sum_{j}^{p} \beta_j^2$$

Analysis



By using the cv.glmnet function wit alpha = 0, we can tune a best lambda range for the ridge regression model by doing a 5-fold cross validation. And with alpha = 1, we can tune a best lambda range for the LASSO regressing model by doing a 5-fold cross validation. In comparisons to LASSO regression, ridge regression has a smoother curve than LASSO regression by controlling lamdba. Unlike ridge regression, LASSo regression increase the MSE dramatically when log(lambda) passes 0.

The MSE and MSPE table shows that LASSo regression with 5-fold cross validation has the best performance, followed by ridge regression with 5-fold cross validation, LASSO regression and ridge regression. The reason is that LASSO eliminated some of the varaibles with coefficients closer to 0, which results in a huge improvemnt in both MSE and MSPE. By doing 5-fold cross valdiation, LASSO impoves a small amount in MSE and MSPE.



According to the ggplot from above, Lasso regression eliminated 11 variables,finishing and curve and free_kick_accuracy and ball_control and agility and balance and shot_power and long_shots and positioning and marking and standing_tackle, with zero coefficients. By comparing ridge regression to LASSO regression, ridge regression kept all the varaibles and assigns weights to all the coefficients whereas LASSO regression doesn't assign weights to the 6 eliminated variables, but add more weights to the remaining 19 varaibles. The ggplot also shows that reactions and potentials attributes are the most important since the coefficients of them are significantly high.

Non-parametric Model

GAM

Generalized Additive Model is to maximize the quality of prediction of a dependent variable Y from various distributions, by estimating unspecific (non-parametric) functions of the predictor variables which are connected to the dependent variable via a link function.

Definiton

The model relates a univariate response variable, Y , to some predictor variables, xi. An exponential family distribution is specified for Y (for example normal, binomial or Poisson distributions) along with a link function g (for example the identity or log functions) relating the expected value of Y to the predictor variables via a structure such as

$$g(E(Y)) = \beta_0 + f_1(x_1) + f_2(x_2) + ... + f_m(x_m)$$

where the functions fi may be functions with a specified parametric form (for example a polynomial, or an un-penalized regression spline of a variable) or may be specified non-parametrically, or semi- parametrically, simply as 'smooth functions', to be estimated by non-parametric means. If we consider interaction effect between to functions of covariates, then GAM of basis of functions with any two convariates xi, xj can be represented as:

$$g(E(Y)) = \beta_0 + f_i(x_i) + f_j(x_j) + ... + f_{i,j}(x_i, x_j)$$

Analysis

For Generalized Additive Model, we do 85/15 splitting of training set and test set. We will generally implement two GAMs. The first GAM is regressing on bases of functions with only one explanatory and the second GAM is regressing on bases of functions with one or two explanatories. Each two explanatory variables is represented as interaction term. First of all, we look at the first GAM. While building the first GAM, we also want to tune how many degrees of freedom for each explanatory variable term. One way to do this is to use random effects. The basic idea is to generates an i.i.d. Gaussian random effect with model matrix. For example, If g is a factor and x is a numeric, then s(x,g,bs='re') produces an i.i.d. normal random slope relating the response to x for each level of g.

For the second GAM, first we want to explore the interaction terms, but how do we do that? We generally use the earth package to build a multiple adaptive regression spline and see which two variables highly interact with each other. From the R output of multiple hingh functions and their corresponding coefficients, we know that the interaction term could be potential and gk_reflexes, dribbling and strength, dribbling and marking, ball_control and gk_reflexes, ball_control and gk_reflexes.After we find their interaction terms, we use the interaction function ti().From the following MSE and MSPE, we generally know that the second model performs better.

```
##                         MSE     MSPE
## No Interaction   1.640802 1.797277
## With interaction 1.320567 1.467482
```

checking their anova tables, we can see which smooth term has a significant non-linear effect on the response by looking at the p-value. The following are the smooth terms that are not significiant with respect to the non-linear effect.

From above, the smooth term s(free kick accuracy), s(balance) and s(penalties) are not significant. The smooth term s(volleys) and s(agility) are statistically significant. Other smooth terms that are not mentioned are statistically highly significant since their p-value are less than 0.01.

```
##  Smooth Terms          p-value
##  s(volleys)            0.042526
##  s(free_kick_accuracy) 0.141044
##  s(agility)            0.02973
##  s(balance)            0.843734
##  s(penalties)          0.11063
```

Gradient Boosting(Tree -based)

Definition

Gradient Boosting (Tree-based) is a boosting algorithm that fits single decision tree at each iteration. Instead of averaging over all the trees, GB tries to find the best linear combination of fitted trees to explain the training data. As a result of this optimization, the GB model trains much slower but might yields better results. However it is also known to possibly overfit the training data.

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function. The general procedure isusing a training set $\{(x1, y1), (x2, y2), \text{ůůů}(xn, yn)\}$ of known values of x and corresponding values of y, the goal is to find an approximation $F\hat{}(x)$ to a function $F(x)$ that minimizes the expected value of some specified loss function $L(y, F(x))$:

$$\hat{F} = argmin\mathbb{E}_{xy}[L(y, F(x))]$$

Analysis

For gradient Boosting, we do 75/25 splitting. We will generally use two tunning method to choose parameters. The first method is to use caret package. This requires a few minutes to tune the parameters. although the number of iteration is relatively small from the perspective of number of samples and the shrinkage is quite large, the model finally gives us the square root of mean square error and mean square prediction error:

Tunning results using caret package

```
##      Parameters       Values
## [1,] nrounds          150
## [2,] max_depth        3
## [3,] eta              1
## [4,] gamma            0
## [5,] colsample_bytree 0.8
## [6,] min_child_weight 1
## [7,] subsample        1
```

```
##      MSE     MSPE
## 0.890285 1.834120
```

Also, we want to track MSE and MSPE for each iteration, so the following also provides us with some thoughts:

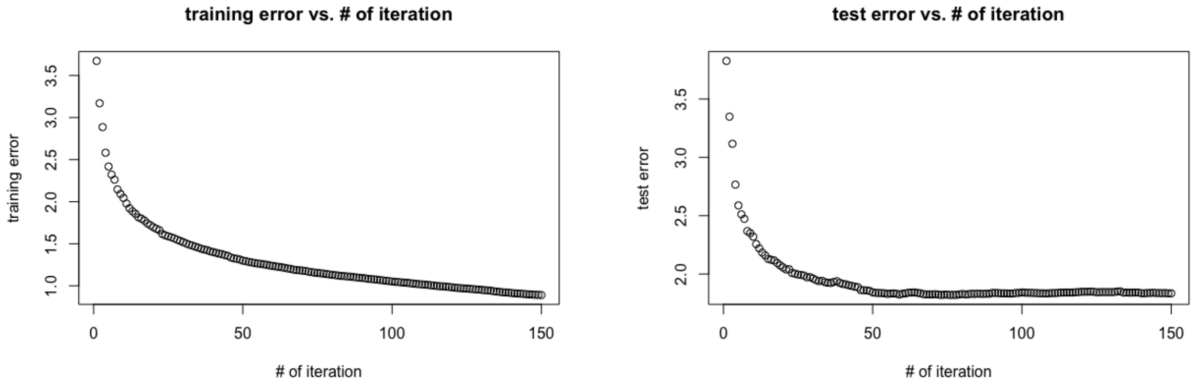


Figure 3: Training Error and Test Error V.S Iterations

As we see in the figure of 2, when starting the iteration, the training error and test error decrease sharply. For training error, it is relatively smaller than test error and slowly decreasing after roughly the iteration of 50. For test error it is almost staying constant after the iteration of 50. This result is reasonable but it is still lack of iteration and the shrinkage is a bit large, so we also want to implement the second method using the grid search method. This method requires cross validation on the training set, setting the candidates for each tuning parameters. It will do grid searching on the whole list of tuning parameters and find the

best candidates for the corresponding tuning parameters. Hence, we decide to set the following candidates of tuning parameters.

Noted that as we have more candidates for tuning the parameters, it will take some time to tune the best candidate. After we find the best candidate, xgb also gives us trained model. This model provides the results of test RMSE for all possible candidates of parameters since we set the metric as RMSE. We want to how test RMSE behaves. Therefore, the plots of MSPE vs. # of iteration and MSPE vs shrinkage are provided below.

The candidates for tuning the parameters are:

- nrounds - [50,150,300,600,1200,2400]
- max_depth - 3
- eta -[0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9,1.0]
- gamma - 0
- colsample_bytree - 0.8
- min_child _weight - 1.0
- subsample - 1.0

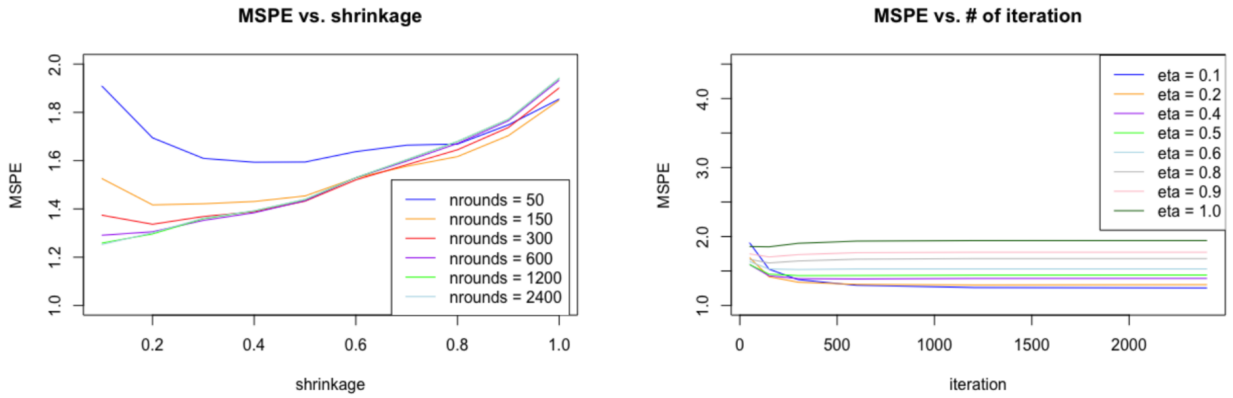


Figure 4: Training Error and Test Error V.S Iterations

From the first figure of 3, if we fix a small shrinkage, when the iteration is small, the MSPE will be larger than others, but fixing a larger shrinkage will make no difference among using each iteration as they are relatively large. From the second figure, if the shrinkage is smaller, the MSPE will decrease at the beginning of the iteration and almost will not decrease after the iteration of 500. Interestingly, MSPE for the shrinkage of 1 even increases a bit and stay almost constant after the iteration of 300. This might indicate that the model is likely to be overfitting. So it seems that smaller shrinkage, larger number of iteration gives us better model performance, but this might take some more time to train the model as the number of iteration is larger. Using the best tuning parameters we find, we have the following training error and test error for the model.

```
##       MSE      MSPE
## 0.4383072 1.3182920
```

From the above result, the second method gives us better performance on finding the best tuning parameter than the performance on the first one. How about their variable importance and model complexity? We plot the variable importance as follows.

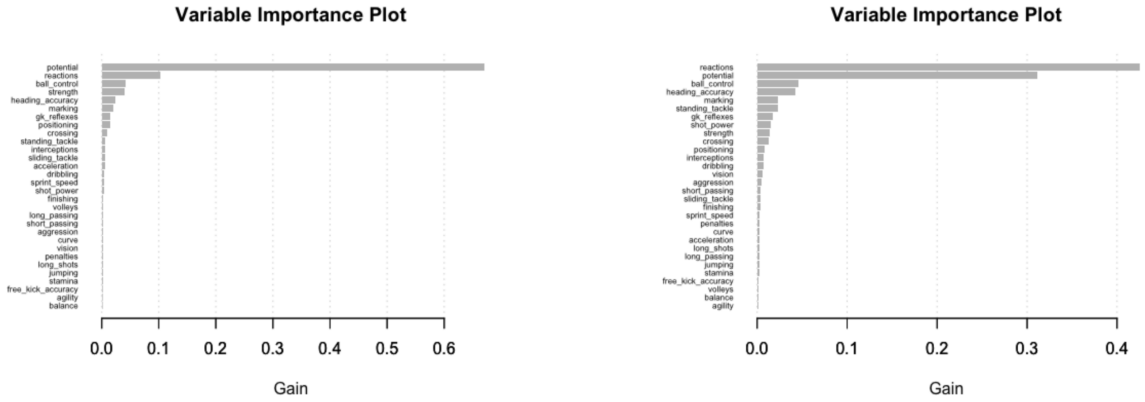


Figure 5: Variable importance for the first model and the second model

From the figure 4, we notice that the variables potential and reactions are relatively important for the first model. Moreover, potential is the most important. However, for the second model, the variable reactions is the most important. These two variable importance make sense since in the soccer match, if a soccer has a potential and a fast reaction, their team is likely to make a better shooting. We also plot the model complexity but both models have almost the same complexity with the leaf depth 4.

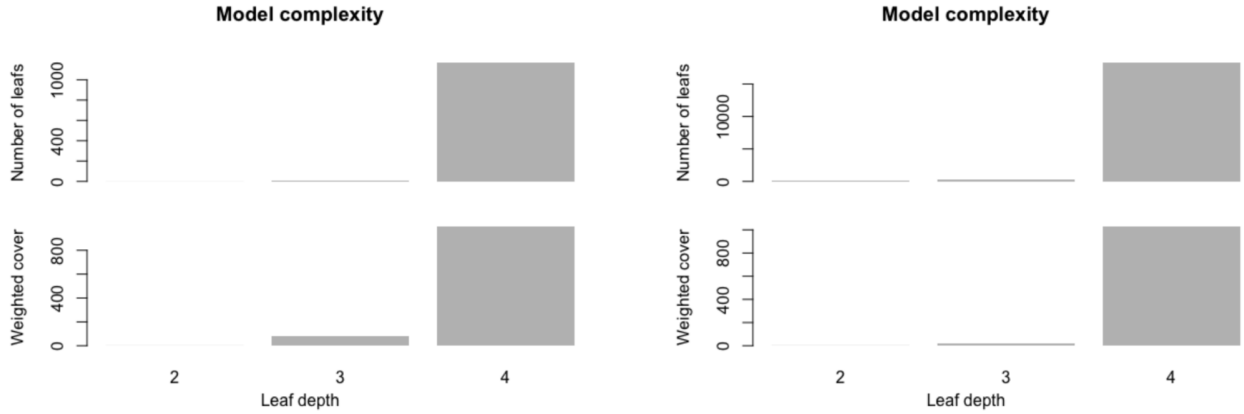


Figure 6: Model complexity for the first model and the second model

Regression Tree

Definition

The regression tree only produces a numerical variable with mixture of continuous and categorical predictor variables. Regression splits features of the data into M disjoint regions $R_1, R_2, ..., R_m$ and fits a piecewise constant-model

$$\mu(x) = \sum_{m=1}^{M} c_m I(x \in R_m)$$

to these M regions. It is designed to approximate continuous valued functions, instead of being used for classification methods.

Analysis

```
## [1] "The top 6 important variables in largest possible regression tree"
```

```
##     reactions     potential  ball_control short_passing   positioning
##     182005.05     131118.58      77255.01      66058.13      64696.09
##    shot_power
##      50039.25
```

```
## [1] "The top 6 important variables in largest possible regression tree for optimal lambda 0.000116591
```

```
##     reactions     potential  ball_control short_passing   positioning
##     188550.97     142559.03      94917.51      73724.71      69692.38
##    shot_power
##      52952.69
```

```
## [1] "The top 6 important variables in largest possible regression treefor lambda min 2.9974781596447
```

```
##     reactions     potential  ball_control short_passing   positioning
##     189082.04     143294.37      95873.25      74684.28      70484.69
##    shot_power
##      53732.57
```

The table from above shows the top 6 important variables in regression tree for both λ_{1se} and λ_{min} with reactions being the most important variable.

```
paste("The MSE and MSPE table for the largest possible tree(cp = 0)")
```

```
## [1] "The MSE and MSPE table for the largest possible tree(cp = 0)"
```

```
err2
```

```
##          MSE     MSPE
## RT  1.580561 4.889147
## 1se 2.444033 5.246653
## min 1.677524 4.922465
```

From the table above, the original largest tree,RT, has the lowest err in terms fo MSE and MSPE, followed by prune with 1se rule, prune by min rule. The reason is that, there exist an over-fitting problem in the original largest tree and regression tree pruning reduces the risk of over-fitting by removing nodes that results in greatest improvement.

Random Forest

###Definiton A random forest is consists of a ensemble of regression trees/decision trees, each capable of producing a response varaible when feeding a set of preditors as input. For regression problem, the response variable classifies which category does the predictor belongs to, by growing simple trees to vote for the most popular category. Forregression problems, the reponses vairables is an estimate of the explanatory variables. Random forest randomly sampling with replacement,boostrapping, of training data when growing and randomly selectes subsets of features when splitting node.

Analysis

```
##
##          OOB      MSPE
## opt.info 2.436565 2.314814
```

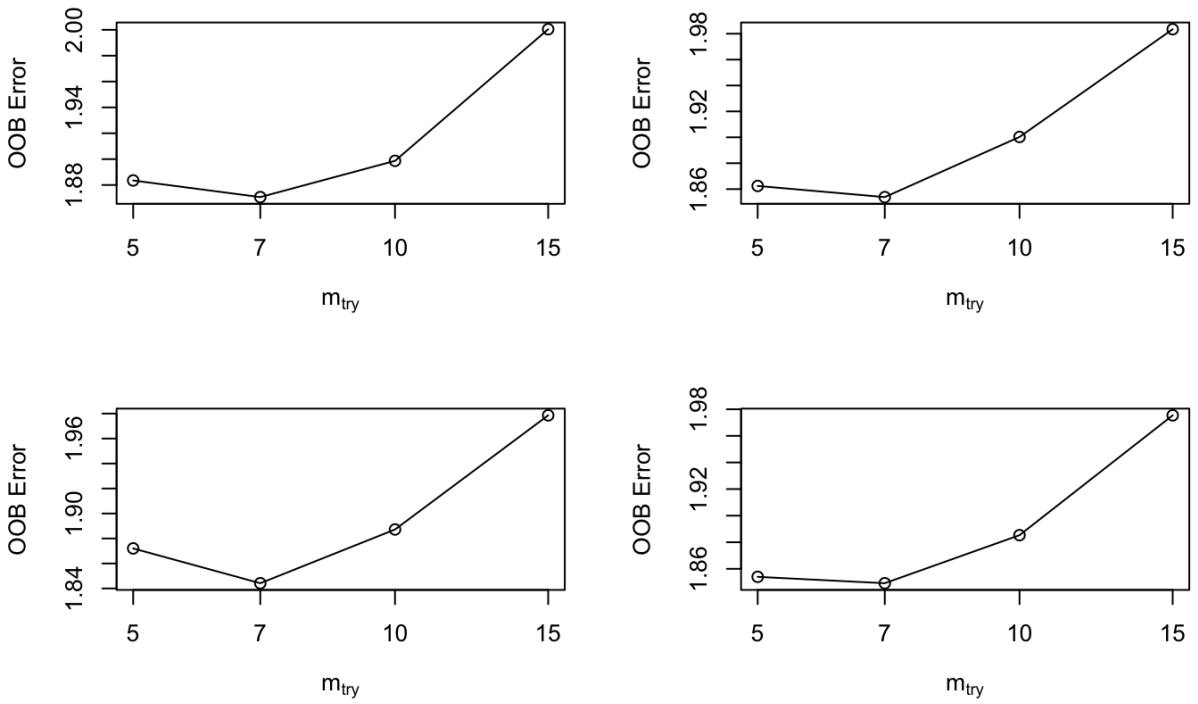


Figure 7: Scatter plot matrix and correlation plot

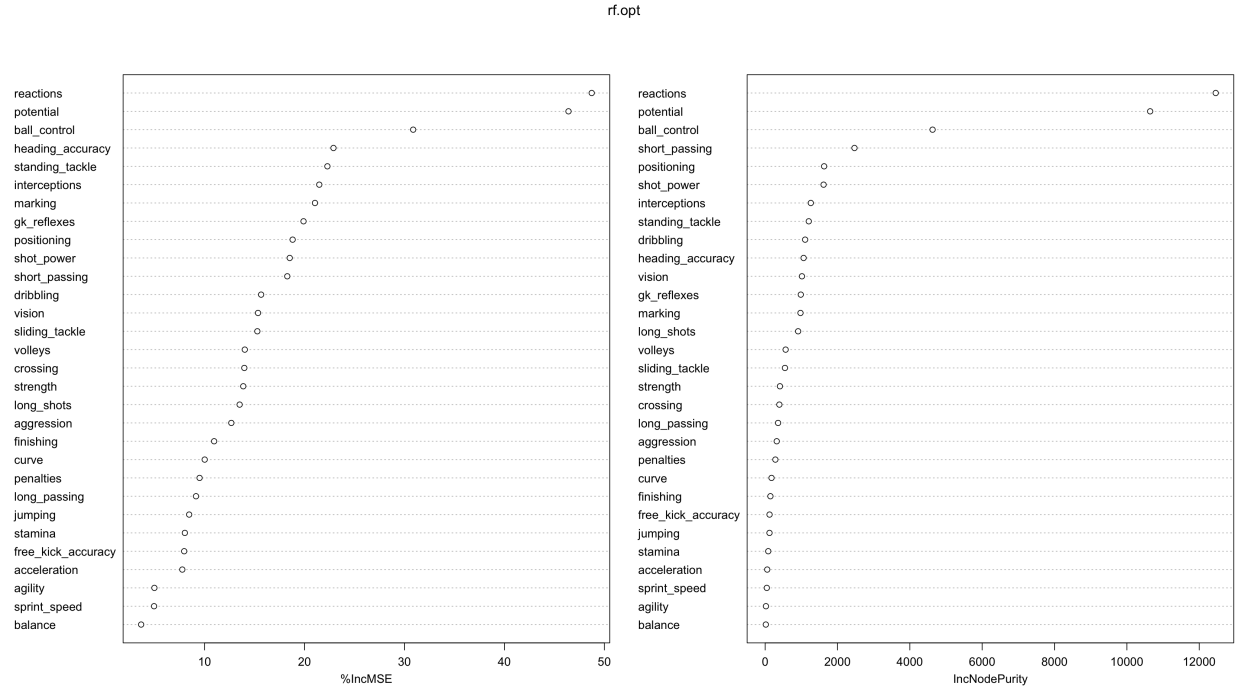


Figure 8: Variance Importance plot for tunned mtry=7 and ntree=1500

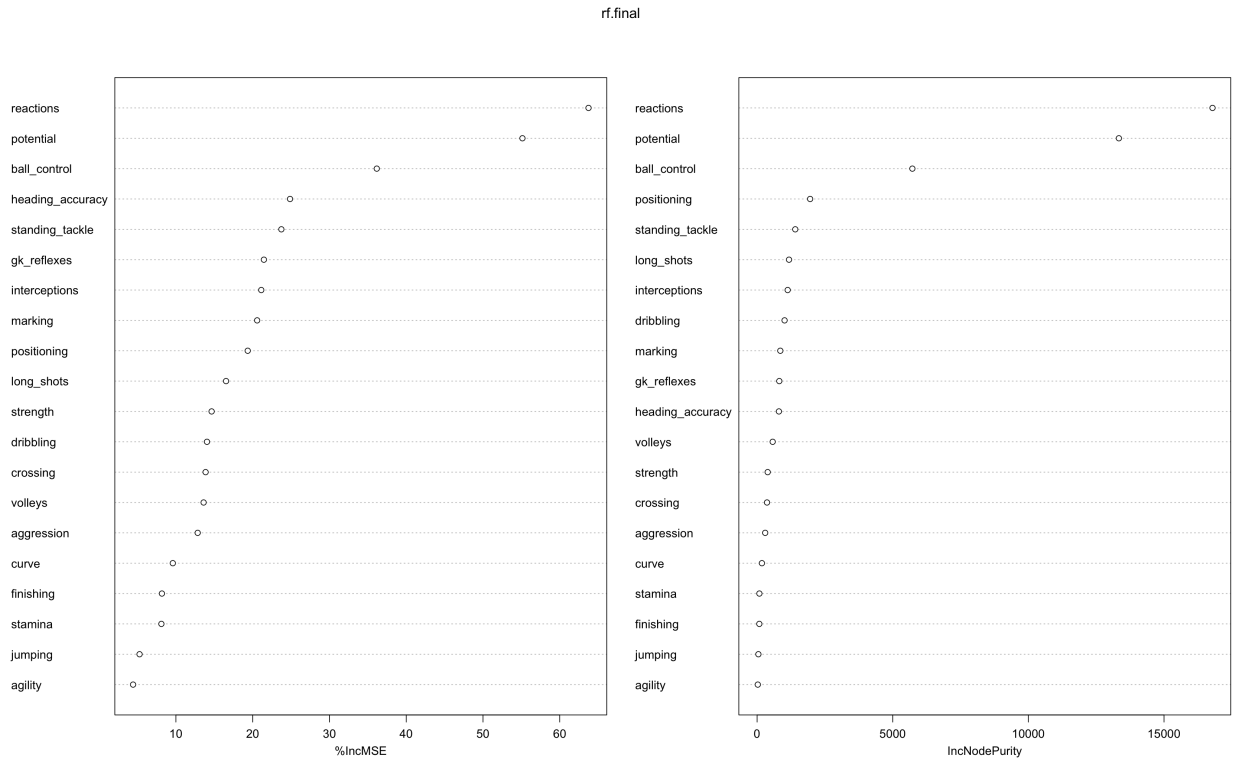


Figure 9: Variable Importance for tunned mtry=7 and ntree=1500 with 15 variables

The 15 most importance variables in the final random forest model are selected by using 5-fold cross validation are: over_rating ~ reations + potential + ball_control + positioning + standing_tackle + gk_reflexes + interceptions+ heading_accuracys + marking + dribbling + crossing + long_shots + volleys + stamina + agility

Statistical Conclusions

- LASSO Regression
  - LASSO Regression might not greatly fit the model compared with non-parametric regression.
  - There is not too much difference between MSE and MSPE whether scaling or not if using LASSO Regression.
- Gradient Boosting (Tree-based)
  - Gradient boosting model with Lower shrinkage and larger number of iteration performs better with respect to its prediction.
  - Tuning parameters using grid search method could bring better model performance in prediction.
  - Both variables potential and reactions are important to the gradient boosting model.
- Generalized Additive Model
  - Simply using the basis functions of smooth term will ignore the interaction effect among pairs of explanatory variables.
  - Regressing on the basis of functions considering interaction is better than regressing on the basis of functions without considering interaction.

- Smooth terms with p-value greater than 0.05 indicates they are not significant to the model with respect to their non-linear effect.

Conclusion in the context of the problem

Future Work

For future work, we would like to explore deeper into the LASSO Regression, since LASSO Regression has a feature of L1 regularization, so could we combine its feature to penalize a non-parametric spline? Regarding to the grid search method and using caret package to do parameter tuning, it is very time consuming to tune large number of parameters, could we do optimization to decrease its running time? In terms of the GAM, is there an approach to search the interaction terms via GAM?

Contribution

Appendix

Variables

- player_name: The name of the player
- finishing: The accuracy of shots using foot, inside the penalty area
- dribbling: The ability to kepp possesion of the ball.
- ball_control: The ability to keep your ball under your feet with velocity.
- reactions: How quickly a player respinds a situation.
- stamina: Determine the rate at which a player will tire during a game.
- interceptions: The ability to intercepts a pass where the ball is going and stop it from going there.
- marking: The ability to track and defend an opposing player.
- overall_rating: The rating of the player based on all attributes.
- heading_accuracy: The accuracy of he player either a pass or a shot by using head.
- curve: The ability to shoot the ball in a curved shape.
- acceleration: Increase in the rate of speed of a player.
- balance: The ability to maintain balance after a physical challenge.
- strength: The ability to win a physical challenge.
- positioning: The ability to read the game offensively, get into good positions, make effective runs, and avoid getting caught offside.
- standing_tackle:The ability of the player to time standing tackles so that they win the ball rather than give away a foul.
- potential: A peak in overall rating that a player could reach.
- short_passing: The ability to perform a pass in short distance.
- free_kick_accuracy: The accuracy of a direct free kick on goal.(Free kick: an unimpeded kick of the stationary ball awared to one side as a penalty for a foul by the other side)
- sprint_speed: The maximum speed over a short distance of a player.
- shot_power: How hard can the player hits the ball when taking a shot at goal.
- long_shots: The accuracy of shots from outside of the penalty area.
- vision: The player's awareness of the position of his team mates & opponents around him.
- sliding_tackle: The ability of the player to time sliding tackles so that they win the ball rather than give away a foul.
- crossing: The accuracy of the player crosses the ball.
- volleys: The accuracy of a player strike or hit the ball at goal before it touches the ground.
- long_passing: The ability to perform a long pass in the air and on the ground to his teammate.
- agility: The ability of a player to move or turn in game.
- jumping: The vertical distance of a player can jump from the ground.
- aggression: The frequency & aggression of jostling, tackling & slide tackling.
- penalties: The ability to take penalties.
- gk_reflexes: The ability to react a ball in movement at goal by the goal keeper.

R-Code

```
knitr::opts_chunk$set(echo = TRUE)
setwd("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction")
soccer.raw <- read.table("data/rating_potential.csv",sep = " ",na.strings = "NA")
library(glmnet)
library(data.table)
library(ggplot2)
library(MASS)
library(rpart)
library(rpart.plot)
```

```r
library(randomForest)
library(caret)
library(dplyr)
library(car)
set.seed(123)
# ---
# title: "Soccer Player Rating Prediction"
# author:
# Junqi Liao 20650701
# Raymond Tan
#
#
# date: "31 July 2019"
# output: csv file
# ---
#


# Importing data and join together player_attributes with player

library(RSQLite)
library(dplyr)

# extract the zip database file in the data folder and set your own db path
con <- dbConnect(SQLite(), dbname="/Users/peterliao/Desktop/stat/stat444/project/data/database.sqlite")
dbListTables(con)

# processing the data
player<- tbl_df(dbGetQuery(con,"SELECT * FROM player"))
player_stats<- tbl_df(dbGetQuery(con,"SELECT * FROM Player_Attributes"))

# join player and player_stats into one table
joint_player_stats<-  player_stats %>%
  rename(player_stats_id = id) %>%
  left_join(player, by = "player_api_id")

# check dimension
dim(joint_player_stats)

#average of overall rating , average of potential
rating_potential<-aggregate(cbind(overall_rating,potential,crossing,finishing,heading_accuracy,short_pas
                                  agility,reactions, balance  ,
                                  shot_power,jumping,stamina  ,
                                  strength,long_shots,aggression    ,
                                  interceptions,positioning,vision   ,
                                  penalties,marking,standing_tackle ,
                                  sliding_tackle,gk_reflexes)~factor(player_name),data=joint_player_sta
colnames(rating_potential)[1]<-"player_name"

# check dimension again
dim(rating_potential)

# some observations for specific attributes
```

```r
hist(rating_potential[,2:32]$gk_reflexes, xlab='gk_reflexes', main='Histogram of gk_reflexes before tra

transform.gk_reflexes<-function(x){
  ifelse(x > 25,x-55,x)
}
rating_potential[,2:32]$gk_reflexes<- transform.gk_reflexes(rating_potential[,2:32]$gk_reflexes)
hist(rating_potential$gk_reflexes,xlab='gk_relexes',main='Histogram of gk_reflexes after transformation

# check if there is NA
colSums(is.na(rating_potential))
# dbWriteTable(con, 'rating_potential',rating_potential,overwrite = TRUE)
# write the table rating_potential.csv to Github for teammate use
write.table(rating_potential, file = "rating_potential.csv")
dbDisconnect(con)

# ---
# title: "Soccer Player Rating Prediction"
# author:
# Junqi Liao 20650701
# Raymond Tan
#
#
# date: "31 July 2019"
# output: data plot
# ---
#

# Visualize the data plot and see if each player attribute has correlation to each other
library(RSQLite)
library(dplyr)

# extract the zip database file in the data folder and set your own db path
con <- dbConnect(SQLite(), dbname="/Users/peterliao/Desktop/stat/stat444/project1/data/database.sqlite")
dbListTables(con)

rating_potential<- tbl_df(dbGetQuery(con,"SELECT * FROM rating_potential"))

# correlation plot
library(corrplot)
corrplot(cor(rating_potential[,-c(1,32)]))

# since short_passing, long_passing, reactions, vision have a higher correlation,
# we pair them with the response variable overall_rating and see the scatterplot
# matrix
pairs(overall_rating~short_passing  +  long_passing+reactions+
        vision,data=rating_potential,
      main="Simple Scatterplot Matrix")
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
soccer=read.csv(file="/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/data/ra
soccer=soccer[-1]


# Function for calculating mse and mspe
```

```r
mse=function(x,y){
  mean((x-y)^2)
}

# Creating training and test sets

as.data.table(soccer)
soccer$set <- ifelse(runif(n=nrow(soccer))>0.85, yes=2, no=1)
soccer.train <- soccer[which(soccer$set==1),]
soccer.test <- soccer[which(soccer$set==2),]
soccer.train$set=NULL
soccer.test$set=NULL

mse.step=cbind(1,1)
colnames(mse.step)=c("mse","mspe")

mse.lasso=cbind(1,1)
colnames(mse.lasso)=c("mse","mspe")

mse.lasso1=cbind(1:2,1)
colnames(mse.lasso1)=c("mse","mspe")
rownames(mse.lasso1)=c("original","updated")

# Defining mininum model and scope
min.model=lm(overall_rating~1,data=soccer.train)
max.model=formula(lm(overall_rating~.,data=soccer.train))

# Stepwise using both forward and backward with BIC
soccer.step=step(min.model,direction = "both",scope = max.model,trace=F,criteria=BIC)

# Calculate msem mspe and vif
soccer.step.predict=predict(soccer.step,newdata=soccer.test[-1])
soccer.step.predict1=predict(soccer.step,newdata=soccer.train[-1])

mse.step[1]=mse(soccer.train$overall_rating,soccer.step.predict1)
mse.step[2]=mse(soccer.test$overall_rating,soccer.step.predict)
mse.step
vif(soccer.step)
## Variable selection using LASSO
soccer.train.x <- soccer.train[,2:length(soccer.train)]
lasso.model <- glmnet(as.matrix(soccer.train.x),soccer.train$overall_rating,alpha = 1)

# Fitting the model with lasso's selection
soccer.lasso=lm(overall_rating~potential+crossing+heading_accuracy+short_passing+volleys+
                  dribbling+curve+long_passing+acceleration+sprint_speed
                +reactions+shot_power+jumping+stamina+strength+aggression+interceptions+
                  vision+penalties+standing_tackle+sliding_tackle+gk_reflexes,data=soccer.train)

# Calculate mse,mspe and vif.
soccer.lasso.predict=predict(soccer.lasso,newdata=soccer.test[-1])
soccer.lasso.predict1=predict(soccer.lasso,newdata=soccer.train[-1])

mse.lasso[1]=mse(soccer.train$overall_rating,soccer.lasso.predict1)
```

```
mse.lasso[2]=mse(soccer.test$overall_rating,soccer.lasso.predict)
mse.lasso1[1,1]=mse.lasso[1]
mse.lasso1[1,2]=mse.lasso[2]
mse.lasso
vif(soccer.lasso)
soccer.lasso2=update(soccer.lasso,.~.-standing_tackle)

soccer.lasso2.predict=predict(soccer.lasso2,newdata=soccer.test[-1])
soccer.lasso2.predict1=predict(soccer.lasso2,newdata=soccer.train[-1])

mse.lasso1[2,1]=mse(soccer.train$overall_rating,soccer.lasso2.predict1)
mse.lasso1[2,2]=mse(soccer.test$overall_rating,soccer.lasso2.predict)

mse.lasso1
vif(soccer.lasso2)
soccer.lasso1=update(soccer.lasso,.~.-standing_tackle-acceleration-heading_accuracy-
                       shot_power-curve-aggression)

soccer.lasso1.predict=predict(soccer.lasso1,newdata=soccer.test[-1])
soccer.lasso1.predict1=predict(soccer.lasso1,newdata=soccer.train[-1])

mse.lasso1[2,1]=mse(soccer.train$overall_rating,soccer.lasso1.predict1)
mse.lasso1[2,2]=mse(soccer.test$overall_rating,soccer.lasso1.predict)

mse.lasso1
# ---
# title: "Soccer Player Rating Prediction"
# responsiblity: LASSO Regression
# author:
# Junqi Liao 20650701
#
#
#
# date: "31 July 2019"
# output: data plot
# ---
#

# Visualize the data plot and see if each player attribute has correlation to each other
library(RSQLite)
library(dplyr)
library(glmnet)
library(caret)

set.seed(123)
# extract the zip database file in the data folder and set your own db path
con <- dbConnect(SQLite(), dbname="/Users/peterliao/Desktop/stat/stat444/project1/data/database.sqlite")
dbListTables(con)

rating_potential<- tbl_df(dbGetQuery(con,"SELECT * FROM rating_potential"))

rating_potential$set <- ifelse(runif(n=nrow(rating_potential))>0.8, yes=2, no=1)
df.rating_potential <- as.data.frame(rating_potential)
```

```r
ytrain <- as.matrix(rating_potential[which(rating_potential$set==1),2])
xtrain <- as.matrix(rating_potential[which(rating_potential$set==1),c(3:32)])
xtrain.scaled <- scale(xtrain)
ytest <- as.matrix(rating_potential[which(rating_potential$set==2),2])
xtest <- as.matrix(rating_potential[which(rating_potential$set==2),c(3:32)])
xtest.scaled <- scale(xtest)

lasso.1 <- glmnet(y=ytrain, x= xtrain, family="gaussian")

# Coefficent path without scaling
plot(lasso.1)

lasso.2 <- glmnet(y=ytrain, x= xtrain.scaled, family="gaussian")

# Coefficent path with scaling
plot(lasso.2)

# use cross validation to estimate optimal lambda
cv.lasso.1 <- cv.glmnet(y=ytrain, x= xtrain, family="gaussian")
cv.lasso.2 <- cv.glmnet(y=ytrain, x= xtrain.scaled, family="gaussian")

plot(cv.lasso.1)
plot(cv.lasso.2)

# Predict both halves using first-half fit
predict.train <- predict(cv.lasso.1, newx=xtrain)
predict.test <- predict(cv.lasso.1, newx=xtest)
MSE.lasso <- mse(ytrain, predict.train)
MSPE.lasso <- mse(ytest,predict.test)

predict.train.scaled <- predict(cv.lasso.2, newx=xtrain.scaled)
predict.test.scaled <- predict(cv.lasso.2, newx=xtest.scaled)
MSE.lasso.scaled <- mse(ytrain, predict.train.scaled)
MSPE.lasso.scaled <- mse(ytest,predict.test.scaled)
t<-as.table(rbind(c(MSE.lasso,MSE.lasso.scaled),c(MSPE.lasso,MSPE.lasso.scaled)))
colnames(t) <- c("without scaling","with scaling")
rownames(t) <- c("MSE","MSPE")

# variable importance function
varImp <- function(object, lambda = NULL, ...) {

  ## skipping a few lines

  beta <- predict(object, s = lambda, type = "coef")
  if(is.list(beta)) {
    out <- do.call("cbind", lapply(beta, function(x) x[,1]))
    out <- as.data.frame(out)
  } else out <- data.frame(Overall = beta[,1])
  out <- abs(out[rownames(out) != "(Intercept)",,drop = FALSE])
  out
}

# plot variable importance
```

```
vari<-varImp(cv.lasso.2,lambda = cv.lasso.1$lambda.min)
sort(vari[,1])
varImp(cv.lasso.2,lambda = cv.lasso.1$lambda.1se)
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
mse <- function(y,yhat) {
  return( mean( (y - yhat)^2 ))
}

soccer <- soccer.raw
soccer$player_name <- NULL
# soccer <- soccer[which(soccer$gk_reflexes <= 20),]
# soccer$gk_reflexes <- NULL
soccer$set <- ifelse(runif(n=nrow(soccer)) > 0.85,yes = 2,no = 1)
#Split data into training set and testing set
soccer.train <- soccer[which(soccer$set == 1),]
soccer.test <- soccer[which(soccer$set == 2),]
soccer.train$set <- NULL
soccer.test$set <- NULL
soccer.train.x <- soccer.train[,2:length(soccer.train)]
soccer.test.x <- soccer.test[,2:length(soccer.test)]

ridge.info <- c()
ridge.cv.info <- c()

lass.info <- c()
lass.cv.nfo <- c()

#Fit ridge model
ridge_model <- glmnet(as.matrix(soccer.train.x),soccer.train$overall_rating,alpha = 0)

#Calculate MSE and MSPE for ridge model
train.pred <- as.matrix(cbind(const=1,soccer.train.x)) %*% coef(ridge_model)
test.pred <- as.matrix(cbind(const=1,soccer.test.x)) %*% coef(ridge_model)
ridge_model.mse <- mse(train.pred,soccer.train$overall_rating)
ridge_model.mspe <- mse(test.pred,soccer.test$overall_rating)
ridge.info <- c(ridge_model.mse,ridge_model.mspe)

#Fit Lasso model
lasso_model <- glmnet(as.matrix(soccer.train.x),soccer.train$overall_rating,alpha = 1)
train.pred <- as.matrix(cbind(const=1,soccer.train.x)) %*% coef(lasso_model)
test.pred <- as.matrix(cbind(const=1,soccer.test.x)) %*% coef(lasso_model)
lasso_model.mse <- mse(train.pred,soccer.train$overall_rating)
lasso_model.mspe <- mse(test.pred,soccer.test$overall_rating)
lasso.info <- c(lasso_model.mse,lasso_model.mspe)


#Fit model with cross validation
ridge_model <- cv.glmnet(as.matrix(soccer.train.x),soccer.train$overall_rating,alpha = 0,nfolds = 5)
lasso_model <- cv.glmnet(as.matrix(soccer.train.x),soccer.train$overall_rating,alpha = 1,nfolds = 5)

best_lambda.ridge <- ridge_model$lambda.1se
best_lambda.lasso <- lasso_model$lambda.1se
par(mfrow=c(1,2))
```

```r
plot(ridge_model,main = "Ridge Regression")
plot(lasso_model,main = "Lasso Regression")
ridge_coeff <- ridge_model$glmnet.fit$beta[,ridge_model$glmnet.fit$lambda == best_lambda.ridge]
lasso_coeff <- lasso_model$glmnet.fit$beta[,lasso_model$glmnet.fit$lambda == best_lambda.lasso]

train.pred <- predict(ridge_model,as.matrix(soccer.train.x),s = "lambda.1se")
test.pred <- predict(ridge_model,as.matrix(soccer.test.x),s = "lambda.1se")
ridge_model.cv.mse <- mse(train.pred,soccer.train$overall_rating)
ridge_model.cv.mspe <- mse(test.pred,soccer.test$overall_rating)
ridge.cv.info <- c(ridge_model.cv.mse,ridge_model.cv.mspe)

train.pred <- predict(lasso_model,as.matrix(soccer.train.x),s = "lambda.1se")
test.pred <- predict(lasso_model,as.matrix(soccer.test.x),s = "lambda.1se")
lasso.cv.mse <- mse(train.pred,soccer.train$overall_rating)
lasso.cv.mspe <- mse(test.pred,soccer.test$overall_rating)
lasso.cv.info <- c(lasso.cv.mse,lasso.cv.mspe)

err <- as.table(rbind(ridge.info,lasso.info,ridge.cv.info,lasso.cv.info))
colnames(err) <- c("MSE","MSPE")
rownames(err) <- c("Ridge","LASSO","RidgeCV","LASSOCV")

err
#Compare Coefficients:
coeff <- data.table(Lasso = lasso_coeff,Ridge = ridge_coeff)
coeff[,Features :=names(ridge_coeff)]
to_plot <- melt(data = coeff,id.vars = 'Features',variable.name = 'Model',value.name = 'Coefficients')
 ggplot(to_plot,aes(x=Features,y=Coefficients,fill=Model)) + coord_flip() + geom_bar(stat = 'identity')

## Generalized additive model

name.soccer<-names(df.soccer)[2:32]

fm.s.temp <- paste('s(', name.soccer[-1], ')', sep = "", collapse = ' + ')
fm.s <- as.formula(paste('overall_rating ~', fm.s.temp))

m.s <-gam(fm.s,data=soccer.train[,2:32],method="REML")
plot(m.s)
summary(m.s)
pred.gam.s <- predict(m.s,newdata=soccer.test[,2:32])
predTrain.gam.s <- predict(m.s,newdata = soccer.train[,2:32])
sqrt(mse(soccer.test[,2],pred.gam.s))
sqrt(mse(soccer.train[,2],predTrain.gam.s))
gam.check(m.s)
plot(m.s)
earth.soccer<-earth(soccer.train[,3:32],soccer.train$overall_rating,degree=2,pmethod="backward")
summary(earth.soccer)

fm.ti.inter <- as.formula(paste('overall_rating ~',fm.s.temp,
'+ ti(potential,gk_reflexes) + ti(dribbling,strength) + ti(dribbling,marking) + ti(ball_control,gk_refl
fm.ti.inter
m.ti.inter <- gam(fm.ti.inter,data=soccer.train[,2:32],family = gaussian)
pred.gam.ti.inter <- predict(m.ti.inter,newdata = soccer.test[,2:32])
predTrain.gam.ti.inter <- predict(m.ti.inter,soccer.train[,2:32])
```

```
sqrt(mse(soccer.test[,2],pred.gam.ti.inter))
sqrt(mse(soccer.train[,2],predTrain.gam.ti.inter))
gam.check(m.ti.inter)
anova(m.ti.inter)

#Extraced output from file:Gam.rmd
err.gam <- as.table(rbind(c(1.640802,1.797277),c(1.320567,1.467482)))
colnames(err.gam) <- c("MSE","MSPE")
rownames(err.gam) <- c("No Interaction","With interaction")
smooth.gam <- as.table(cbind(c("s(volleys)","s(free_kick_accuracy)","s(agility)","s(balance)","s(penalt
colnames(smooth.gam) <- c("Smooth Terms","p-value")
rownames(smooth.gam) <- c("","","","","")
err.gam
smooth.gam
# ---
# title: "Soccer Player Rating Prediction"
# author:
# Junqi Liao 20650701
#
#
#
# date: "31 July 2019"
# output: data plot
# ---
#
set.seed(123)
# Visualize the data plot and see if each player attribute has correlation to each other
library(RSQLite)
library(dplyr)
library(mgcv)
library(gbm)
library(caret)
library(xgboost)
library(Metrics)

# extract the zip database file in the data folder and set your own db path
con <- dbConnect(SQLite(), dbname="/Users/peterliao/Desktop/stat/stat444/project1/data/database.sqlite"
dbListTables(con)

rating_potential<- tbl_df(dbGetQuery(con,"SELECT * FROM rating_potential"))

rating_potential$set <- ifelse(runif(n=nrow(rating_potential))>0.75, yes=2, no=1)
df.rating_potential <- as.data.frame(rating_potential)
nrow(as.matrix(df.rating_potential[which(df.rating_potential$set==1),c(3:32)]))

# check dimension
dim(df.rating_potential)

# split the dataset into training set and test set
xtrain<-as.matrix(df.rating_potential[which(df.rating_potential$set==1),c(3:32)])
ytrain<-df.rating_potential[which(df.rating_potential$set==1),2]
dtrain<- xgb.DMatrix(data=xtrain,label=ytrain)
xtest<-as.matrix(df.rating_potential[which(df.rating_potential$set==2),c(3:32)])
```

```
ytest<-df.rating_potential[which(df.rating_potential$set==2),2]
dtest<- xgb.DMatrix(data=xtest,label=ytest)

# tune parameters using gradient boosting
caret::train(method="xgbTree",x=xtrain,y=ytrain)

# define parameters from tuning results
params = list(metric=list("rmse","auc"),
              tuneLength=150,
              model="xgb",
              max_depth=3,
              eta=1,
              cvfolds=5,
              gamma = 0, colsample_bytree = 0.8, min_child_weight = 1,subsample=1)


watchlist<- list(train = dtrain, eval = dtest)

rating.boost <-xgb.train(booster="dart",data=dtrain,nrounds = 150,params = params,metrics=list("rmse","a

# compute predicted value for training set and test set
pred.test.rating<-predict(rating.boost,xtest)
pred.train.rating<-predict(rating.boost,xtrain)

# evaluate MSPE and sMSE
sqrt(mse(ytest,pred.test.rating))
sqrt(mse(ytrain,pred.train.rating))

plot(rating.boost$evaluation_log$iter,rating.boost$evaluation_log$train_rmse,xlab='# of iteration',ylab=
plot(rating.boost$evaluation_log$iter,rating.boost$evaluation_log$eval_rmse,xlab='# of iteration',ylab=

# plot variable importance
xgb.plot.importance(xgb.importance(model=rating.boost),xlab='Gain',main='Variable Importance Plot')

# plot model complexity
xgb.plot.deepness(rating.boost)

# save the model locally for self using purpose
xgb.save(rating.boost,'rating.boost.Rdata')

# more exploration on grid search method
cv.ctrl <- trainControl(method = "repeatedcv", repeats = 1,number = 5)

xgb.grid <- expand.grid(nrounds = c(50,150,300,600,1200,2400),
                        max_depth = 3,
                        eta =seq(0.1,1,0.1),
                        gamma = 0,
                        colsample_bytree = 0.8,
                        min_child_weight=1,
                        subsample=1
)

xgb_tune <-train(overall_rating ~.,
```

```
                  data=df.rating_potential[which(df.rating_potential$set==1),c(2:32)],
                  method="xgbTree",
                  metric = "RMSE",
                  trControl=cv.ctrl,
                  tuneGrid=xgb.grid
)

pred2.test.rating <- predict(xgb_tune, xtest)
pred2.train.rating <-predict(xgb_tune, xtrain)
sqrt(mse(ytest,pred2.test.rating))
sqrt(mse(ytrain, pred2.train.rating))

# plot variable importance
xgb.plot.importance(xgb.importance(model=xgb_tune$finalModel),xlab='Gain',main='Variable Importance Plo

# plot model complexity
xgb.plot.deepness(xgb_tune$finalModel)

df.n50fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 50),c(1,7,8)]
df.n150fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 150),c(1,7,8)]
df.n300fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 300),c(1,7,8)]
df.n600fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 600),c(1,7,8)]
df.n1200fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 1200),c(1,7,8)]
df.n2400fixed<-xgb_tune$results[which(xgb_tune$results$nrounds == 2400),c(1,7,8)]
plot(df.n50fixed$eta,df.n50fixed$RMSE,type='l',col='blue',ylim = c(1,2),xlab='shrinkage',ylab='MSPE',ma
lines(df.n150fixed$eta,df.n150fixed$RMSE,type='l',col='orange')
lines(df.n300fixed$eta,df.n300fixed$RMSE,type='l',col='red')
lines(df.n600fixed$eta,df.n600fixed$RMSE,type='l',col='purple')
lines(df.n1200fixed$eta,df.n1200fixed$RMSE,type='l',col='green')
lines(df.n2400fixed$eta,df.n2400fixed$RMSE,type='l',col='lightblue')
# Add a legend
legend(0.67, 1.52, legend=c("nrounds = 50", "nrounds = 150",
                        "nrounds = 300", "nrounds = 600",
                        "nrounds = 1200", "nrounds = 2400"),
       col=c("blue", "orange","red","purple", "green","lightblue"), lty=1:1)

df.eta0.1fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.1),c(1,7,8)]
df.eta0.2fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.2),c(1,7,8)]
df.eta0.3fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.3),c(1,7,8)]
xgb_tune$results$eta[1] == 0.1
df.eta0.4fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.4),c(1,7,8)]
df.eta0.5fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.5),c(1,7,8)]
df.eta0.6fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.6),c(1,7,8)]
df.eta0.7fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.7),c(1,7,8)]
df.eta0.8fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.8),c(1,7,8)]
df.eta0.9fixed<-xgb_tune$results[which(xgb_tune$results$eta == 0.9),c(1,7,8)]
df.eta1.0fixed<-xgb_tune$results[which(xgb_tune$results$eta == 1),c(1,7,8)]
plot(df.eta0.1fixed$nrounds,df.eta0.1fixed$RMSE,type='l',col='blue',ylim=c(1,4.5),xlab='iteration',ylab=
lines(df.eta0.2fixed$nrounds,df.eta0.2fixed$RMSE,type='l',col='orange')
lines(df.eta0.4fixed$nrounds,df.eta0.4fixed$RMSE,type='l',col='purple')
lines(df.eta0.5fixed$nrounds,df.eta0.5fixed$RMSE,type='l',col='green')
lines(df.eta0.6fixed$nrounds,df.eta0.6fixed$RMSE,type='l',col='lightblue')
lines(df.eta0.8fixed$nrounds,df.eta0.8fixed$RMSE,type='l',col='grey')
```

```
lines(df.eta0.9fixed$nrounds,df.eta0.9fixed$RMSE,type='l',col='pink')
lines(df.eta1.0fixed$nrounds,df.eta1.0fixed$RMSE,type='l',col='darkgreen')
legend(1850,4.63, legend=c("eta = 0.1", "eta = 0.2",
                           "eta = 0.4", "eta = 0.5",
                           "eta = 0.6", "eta = 0.8",
                           "eta = 0.9", "eta = 1.0"),
       col=c("blue", "orange","purple", "green","lightblue","grey","pink","darkgreen"), lty=1:1)

##Extract output from file:gradboost.R
Parameters <- c("nrounds","max_depth","eta","gamma","colsample_bytree","min_child_weight","subsample")
Values <- c(150,3,1.0,0,0.8,1.0,1.0)
out <- as.table(cbind(Parameters,Values))
rownames(out) <- c()

err.xgboost <- as.table(c(0.890285,1.834120))
rownames(err.xgboost) <- c("MSE","MSPE")

err.xgboost1 <- as.table(c(0.4383072,1.318292))
rownames(err.xgboost1) <- c("MSE","MSPE")
out
err.xgboost
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
err.xgboost1
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
calErr <- function(tree,data,y) {
  y.hat <- predict(tree, newdata = data)
  return(mse(y,y.hat))
}

rt <- rpart(data = soccer.train, soccer.train$overall_rating ~ . , method = 'anova')
pruneTree.1se <- function(tree) {
  tree$cptable[,c(2:5,1)]
  tree.cpt <- tree$cptable
  minrow <- which.min(tree.cpt[,4])
  se.row <- min(which(tree.cpt[,4] < tree.cpt[minrow,4] + tree.cpt[minrow,5]))
  cplow.1se <- tree.cpt[se.row,1]
  cpup.1se <- ifelse(se.row==1, yes=1, no = tree.cpt[se.row-1,1])
  cp.1se <<- sqrt(cplow.1se*cpup.1se)
  prune.1se <- prune(tree,cp = cp.1se )
  return(prune.1se)
}

pruneTree.min <- function(tree) {
  tree$cptable[,c(2:5,1)]
  tree.cpt <- tree$cptable
  minrow <- which.min(tree.cpt[,4])
  cplow.min <- tree.cpt[minrow,1]
  cpup.min <- ifelse(minrow==1, yes = 1, no = tree.cpt[minrow-1,1])
  cp.min <<- sqrt(cplow.min * cpup.min)
  prune.mincp <- prune(tree,cp = cp.min)
  return(prune.mincp)
```

```
}

prune.1se <- pruneTree.1se(rt)
prune.mincp <- pruneTree.min(rt)


# prp(rt, type=1, extra=1, main="Original full tree")

#Calculating Unpruned Regression tree MSE and MSPE
rt.mse <- calErr(rt,soccer.train.x,soccer.train$overall_rating)
rt.mspe <- calErr(rt,soccer.test.x,soccer.test$overall_rating)
rt.info <- c(rt.mse,rt.mspe)


#Calculating pruned Regression tree MSE and MSPE
prune.1se.mse <- calErr(prune.1se,soccer.train.x,soccer.train$overall_rating)
prune.1se.mspe <- calErr(prune.1se,soccer.test.x,soccer.test$overall_rating)
prune.1se.info <- c(prune.1se.mse,prune.1se.mspe)


prune.mincp.mse <- calErr(prune.mincp,soccer.train.x,soccer.train$overall_rating)
prune.mincp.mspe <- calErr(prune.mincp,soccer.test.x,soccer.test$overall_rating)
prune.mincp.info <- c(prune.mincp.mse,prune.mincp.mspe)




err1 <- as.table(rbind(rt.info,prune.1se.info,prune.mincp.info))
colnames(err1) <- c("MSE","MSPE")
rownames(err1) <- c("RT","1se","min")
par(mfrow=c(1,2))
plotcp(prune.1se)
plotcp(prune.mincp)
#Build Regression tree as large as possible
rt.cp0 <- rpart(data = soccer.train, soccer.train$overall_rating ~ ., method = "anova", cp = 0)
par(mfrow=c(1,2))
prune.1se <- pruneTree.1se(rt.cp0)
prune.mincp <- pruneTree.min(rt.cp0)
paste("The top 6 important variables in largest possible regression tree")
head(rt$variable.importance)
paste("The top 6 important variables in largest possible regression tree for optimal lambda",cp.1se)
head(prune.1se$variable.importance)
paste("The top 6 important variables in largest possible regression treefor lambda min",cp.min)
head(prune.mincp$variable.importance)


rt.cp0.mse <- calErr(rt.cp0,soccer.train.x,soccer.train$overall_rating)
rt.cp0.mspe <- calErr(rt.cp0,soccer.test.x,soccer.test$overall_rating)
rt.cp0.info <- c(rt.cp0.mse,rt.cp0.mspe)


#Calculating pruned Regression tree MSE and MSPE
prune.1se.mse <- calErr(prune.1se,soccer.train.x,soccer.train$overall_rating)
```

```r
prune.1se.mspe <- calErr(prune.1se,soccer.test.x,soccer.test$overall_rating)
prune.1se.info <- c(prune.1se.mse,prune.1se.mspe)


prune.mincp.mse <- calErr(prune.mincp,soccer.train.x,soccer.train$overall_rating)
prune.mincp.mspe <- calErr(prune.mincp,soccer.test.x,soccer.test$overall_rating)
prune.mincp.info <- c(prune.mincp.mse,prune.mincp.mspe)


err2 <- as.table(rbind(rt.cp0.info,prune.1se.info,prune.mincp.info))
colnames(err2) <- c("MSE","MSPE")
rownames(err2) <- c("RT","1se","min")


paste("The MSE and MSPE table for the largest possible tree(cp = 0)")
err2

opt.info <- c(2.436565,2.314814)
#try 1000 trees
t <- seq(500,2500,500)
p <- length(colnames(soccer.train))
m <- ceiling(c(1,p/5,p/6,p/8))
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
rf.oob.table <- as.table(rbind(c("OOB","MSPE"),opt.info))
colnames(rf.oob.table) <- c("","")
rf.oob.table
#Optimal mtry = 7, Optimal ntrees = 1500
rf.opt <- randomForest(soccer.train.x,soccer.train$overall_rating, mtry = 7, ntree = 1500,importance = 
#Calculate OOB error
OOB.opt <- sqrt(mse(soccer.train$overall_rating,rf.opt$predicted))
yhat<- predict(rf.opt,soccer.test)
mspes.opt <- sqrt(mse(soccer.test$overall_rating,yhat))

rf.cv <- rfcv(trainx = soccer.train.x,trainy = soccer.train$overall_rating,ntree = 1500,cv.fold = 5)
opt.info1 <- c(OOB.opt,mspes.opt)
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
knitr::include_graphics("/Users/Raymond/Desktop/Raymond Tan/HW/4B/STAT444/soccer-rating-prediction/repo
rf.cv$error.cv

temp <- soccer.train.x
temp$shot_power <- NULL
temp$short_passing <-NULL
temp$vision <- NULL
temp$sliding_tackle <- NULL
temp$free_kick_accuracy <- NULL
temp$acceleration <- NULL
temp$sprint_speed<- NULL
temp$penalties <- NULL
temp$long_passing <- NULL
temp$balance <- NULL


rf.final <- randomForest(temp,soccer.train$overall_rating, mtry = 7, ntree = 1500,importance = TRUE,kee
```

```
OOB <- sqrt(mse(soccer.train$overall_rating,rf.final$predicted))
yhat <- predict(rf.final,soccer.test)
mspes <- sqrt(mse(soccer.test$overall_rating,yhat))


final.info <- c(OOB,mspes)
OOB.table <- as.table(rbind(opt.info,final.info))
colnames(OOB.table) <- c("OOB","MSPE")
rownames(OOB.table) <- c("31 vars","15 vars")
OOB.table
```

Reference

Statistics How To. (n.d). Ridge regression. Retrieved on July 22nd, 2019
From: https://www.statisticshowto.datasciencecentral.com/ridge-regression/.

Statistics How To. (n.d). What is lasso regression? Retrieved on July 21st, 2019
From: https://www.statisticshowto.datasciencecentral.com/lasso-regression/.

Statistics Solutions. (n.d). What is multiple linear re- gression? Retrieved on July 21st, 2019
From: https://www.statisticssolutions.com/what-is-multiple-linear-regression/.

Statsoft. (n.d). General additive model. Retrieved on Aug 8th, 2019
From: http://www.statsoft.com/Textbook/Generalized-Additive-Models.

Feature importance in random forests when features are correlated
From: http://corysimon.github.io/articles/feature-importance-in-random-forests-when-features-are-correlated/

Statistics What is random forest?
From: http://www.statsoft.com/Textbook/Random-Forest