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

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 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.

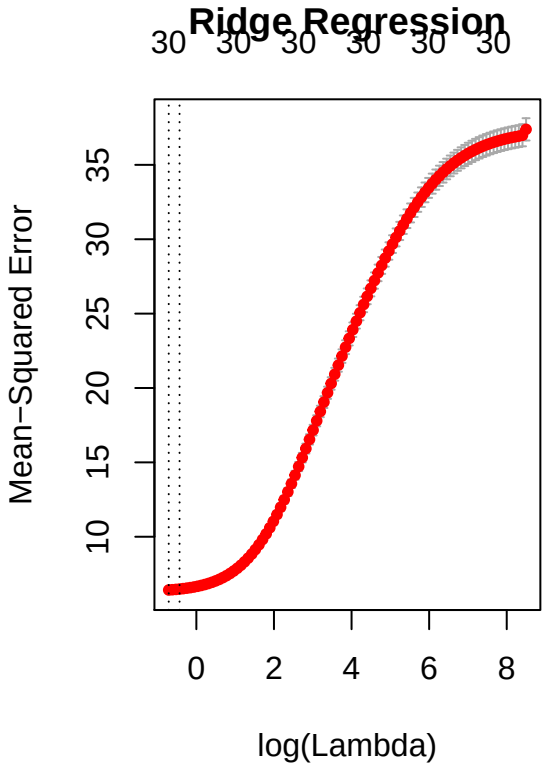
Smoothing Methods

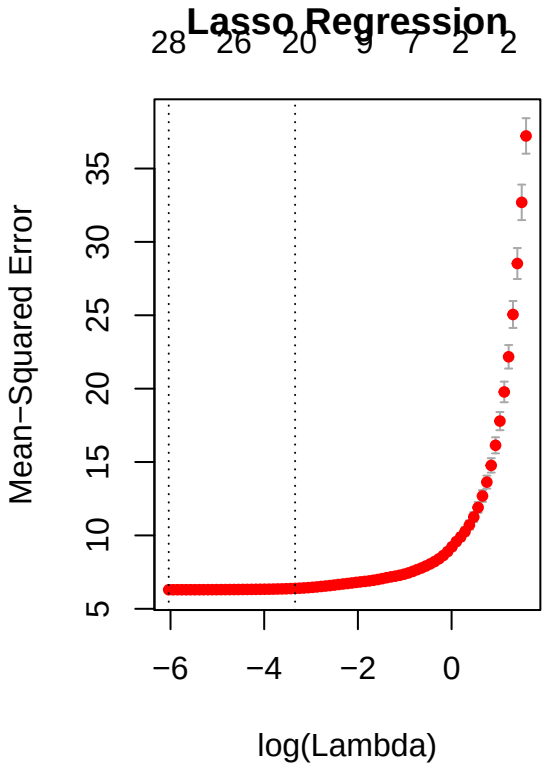
Linear Models

Multiple Linear Regression

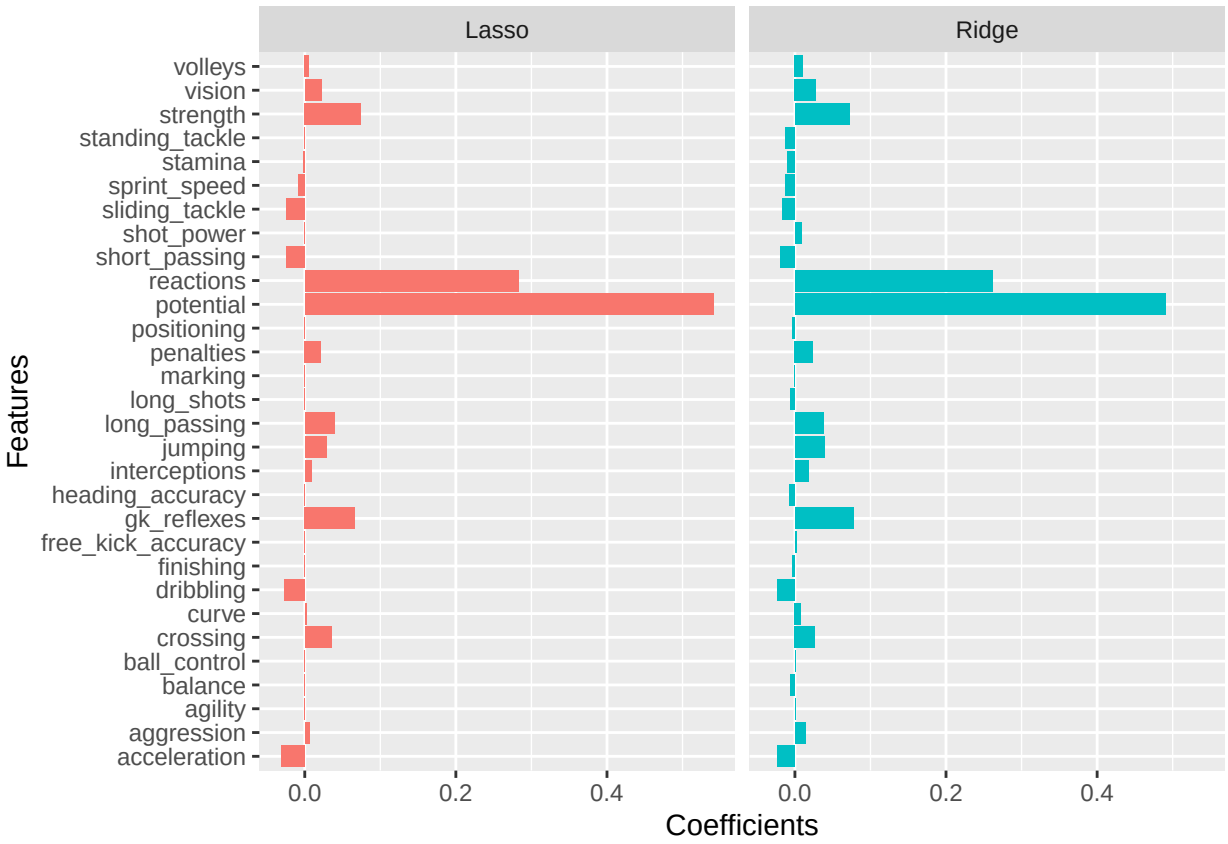
LASSO Regression

Ridge Regression






```
##                MSE      MSPE
## Ridge    22.032908 24.375103
## LASSO     9.046736  9.516999
## RidgeCV   6.447214  6.789598
## LASSOCV   6.351917  6.545837
```



Non-Linear Model

GAM

Tensor Producting Smoothing

Regression Tree

```
##     reactions     potential  ball_control short_passing   positioning
##     185855.75     142082.83      92992.91      74332.56      59964.60
##    shot_power
##      52218.48
```

```
##     reactions     potential  ball_control short_passing   positioning
##     186346.45     142630.24      93827.43      74974.25      60653.96
##    shot_power
##      52881.45
```

```
err1
```

```
##          MSE     MSPE
## RT  10.00523 10.80389
## 1se 10.00523 10.80389
## min 10.00523 10.80389
```

```
err2
```

```
##          MSE     MSPE
```

```
## RT  1.584659 4.559574
## 1se 2.581267 4.527203
## min 1.878952 4.520752
```

Random Forest

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

Conclusion in the context of the problem

Future Work

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/data")
soccer.raw <- read.table("rating_potential.csv",sep = " ",na.strings = "NA")
library(glmnet)
library(data.table)
library(ggplot2)
library(MASS)
library(rpart)
library(rpart.plot)
library(randomForest)
library(caret)
set.seed(123)
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))
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')
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))
```

```r
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$overall_rating ~ ., method = "anova", cp = 0)
par(mfrow=c(1,2))
prune.1se <- pruneTree.1se(rt.cp0)
prune.mincp <- pruneTree.min(rt.cp0)
head(prune.1se$variable.importance)
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$overall_rating)
prune.1se.mspe <- calErr(prune.1se,soccer.test.x,soccer.test$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")

err1
err2
#try 1000 trees
t <- seq(500,2500,500)
p <- length(colnames(soccer.train))
m <- ceiling(c(1,p/5,p/6,p/8))
#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$overall_rating,ntree = 1500,cv.fold = 5)
opt.info <- c(OOB.opt,mspes.opt)
rf.cv$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
```