OBP Projection

Terrick Boire

2023-06-05

For this project we had to predict what player`s on base percentage would be after two months on playing. In a previous class we had to use predictive analytics to analyze a data set. So based off of that experience my thought process was to follow the steps I took in that project while also exploring a could of new models from the ones I previously used in this class.Some of the new models I took a look at included a regression tree, random forest, boosting as well as bagging. A small amount of data cleaning was need in order to make this data set usable.

To measure my accuracy I thought using mean square error was the best method. When it comes to mean square error you want it to be as close to zero as possible. As I began and finished my analysis I found that my models were getting more accurate. The most accurate model was the boosted model which gave me an MSE of nearly 0. All of the models did do really well according to MSE as they were all nearly 0. Of the four models the regression tree did the worst with a MSE of 0.0005516988. The random forest and bagging methods amost had the same MSE. The MSE for the random forest was 0.0001405223 and for the bagging model it was 0.0001342595. The best method was the boosting method and that had an MSE extremly close to 0 and it was 2.812912e-28, which is significantly lower that the other observed MSEs. Below you can find my code as well as graphs used to help me analyze and predict the data. I also attached an image of the first 20 OBP observation I predicted as well as the actual value.

	FullSeason_OBP	Tree_Prediction	RandomForest_Prediction	Bagging_Prediction	Boosting_Prediction
1	0.406	0.3561667	0.4022678	0.4011325	0.406
2	0.429	0.3830000	0.4030696	0.4063166	0.429
3	0.315	0.3252308	0.3151293	0.3163034	0.315
4	0.353	0.3561667	0.3532314	0.3522554	0.353
5	0.341	0.3830000	0.3547210	0.3547610	0.341
6	0.412	0.3561667	0.3872277	0.3908326	0.412
7	0.365	0.3561667	0.3592872	0.3596579	0.365
8	0.324	0.2882000	0.3256663	0.3270697	0.324
9	0.348	0.3561667	0.3570867	0.3562490	0.348
10	0.364	0.3561667	0.3566300	0.3587355	0.364
11	0.322	0.3252308	0.3264499	0.3285257	0.322
12	0.389	0.3830000	0.3794746	0.3798697	0.389
13	0.438	0.3830000	0.4067706	0.4041695	0.438
14	0.355	0.3561667	0.3540533	0.3560510	0.355
15	0.313	0.3561667	0.3333826	0.3339243	0.313
16	0.357	0.3561667	0.3514788	0.3510989	0.357
17	0.364	0.3561667	0.3590604	0.3583835	0.364
18	0.355	0.3252308	0.3457280	0.3448385	0.355
19	0.318	0.3561667	0.3350257	0.3348757	0.318
20	0.358	0.3561667	0.3562542	0.3581705	0.358

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse
2.0.0 ──
## ✓ dplyr     1.1.2     ✓ readr     2.1.4
## ✓ forcats   1.0.0     ✓ stringr   1.5.0
## ✓ ggplot2   3.4.2     ✓ tibble    3.2.1
## ✓ lubridate 1.9.2     ✓ tidyr     1.3.0
## ✓ purrr     1.0.1
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all
conflicts to become errors
```

```
library(lubridate)
library(caret)
```

```
## Loading required package: lattice
##
## Attaching package: 'caret'
##
```

```
## The following object is masked from 'package:purrr':
##
##     lift

library(stats)
library(ggplot2)
library(corrplot)

## corrplot 0.92 loaded

library(dplyr)
```

Used https://hastie.su.domains/ISLR2/ISLRv2_website.pdf textbook for reference when coding

Loading the data

```
batting <- read.csv("batting.csv")
```

checking the data for NAs

```
sum(is.na(batting))

## [1] 0
```

Cleaning the Data

Removing the percent symbol from the data

```
batting$MarApr_BB. <- gsub( "%", "", as.character(batting$MarApr_BB.))

batting$MarApr_K. <- gsub( "%", "", as.character(batting$MarApr_K.))

batting$MarApr_LD. <- gsub( "%", "", as.character(batting$MarApr_LD.))

batting$MarApr_GB. <- gsub( "%", "", as.character(batting$MarApr_GB.))

batting$MarApr_FB. <- gsub( "%", "", as.character(batting$MarApr_FB.))

batting$MarApr_IFFB. <- gsub( "%", "", as.character(batting$MarApr_IFFB.))

batting$MarApr_HR.FB <- gsub( "%", "", as.character(batting$MarApr_HR.FB))

batting$MarApr_Swing. <- gsub( "%", "", as.character(batting$MarApr_Swing.))

batting$MarApr_O.Swing. <- gsub( "%", "",
```

```
as.character(batting$MarApr_O.Swing.))

batting$MarApr_Z.Swing. <- gsub( "%", "",
as.character(batting$MarApr_Z.Swing.))

batting$MarApr_Contact. <- gsub( "%", "",
as.character(batting$MarApr_Contact.))

batting$MarApr_O.Contact. <- gsub( "%", "",
as.character(batting$MarApr_O.Contact.))


batting$MarApr_Z.Contact. <- gsub( "%", "",
as.character(batting$MarApr_Z.Contact.))

view(batting)
```

Converting the percent values to their actual values

```
batting$MarApr_BB. <- as.numeric(batting$MarApr_BB.)

batting$MarApr_K. <- as.numeric(batting$MarApr_K.)

batting$MarApr_LD. <- as.numeric(batting$MarApr_LD.)

batting$MarApr_GB. <- as.numeric(batting$MarApr_GB.)

batting$MarApr_FB. <- as.numeric(batting$MarApr_FB.)

batting$MarApr_IFFB. <- as.numeric(batting$MarApr_IFFB.)

batting$MarApr_HR.FB <- as.numeric(batting$MarApr_HR.FB)

batting$MarApr_Swing. <- as.numeric(batting$MarApr_Swing.)

batting$MarApr_O.Swing. <- as.numeric(batting$MarApr_O.Swing.)

batting$MarApr_Z.Swing. <- as.numeric(batting$MarApr_Z.Swing.)

batting$MarApr_Contact. <- as.numeric(batting$MarApr_Contact.)

batting$MarApr_O.Contact. <- as.numeric(batting$MarApr_O.Contact.)

batting$MarApr_Z.Contact. <- as.numeric(batting$MarApr_Z.Contact.)

view(batting)
```

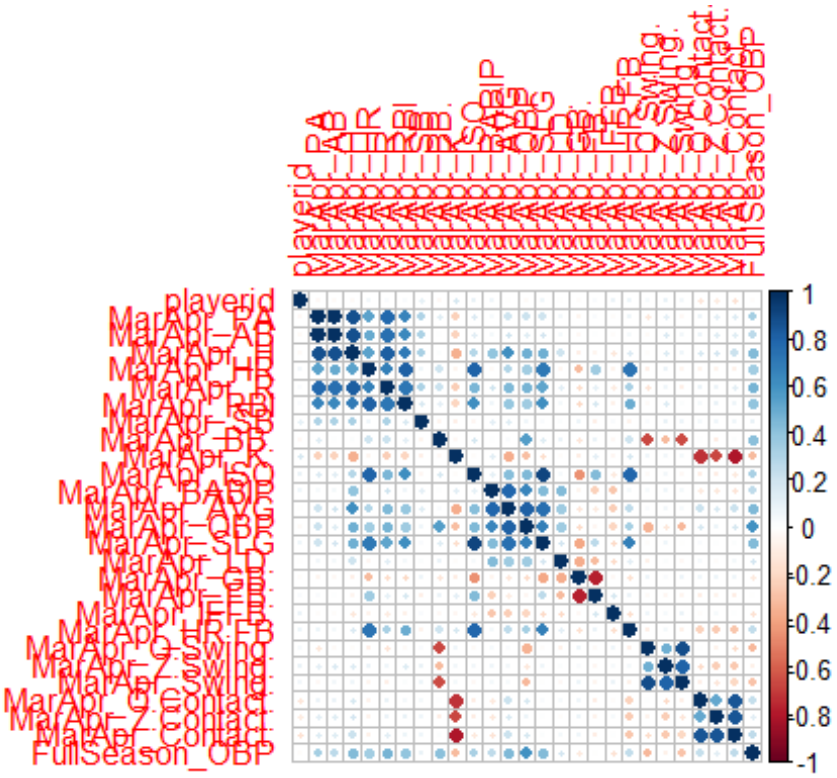
```
batting$MarApr_BB. <- batting$MarApr_BB. / 100

batting$MarApr_K. <- batting$MarApr_K. / 100

batting$MarApr_LD. <- batting$MarApr_LD. / 100

batting$MarApr_GB. <- batting$MarApr_GB. / 100

batting$MarApr_FB. <- batting$MarApr_FB. / 100

batting$MarApr_IFFB. <- batting$MarApr_IFFB. / 100

batting$MarApr_HR.FB <- batting$MarApr_HR.FB / 100

batting$MarApr_Swing. <- batting$MarApr_Swing. / 100

batting$MarApr_O.Swing. <- batting$MarApr_O.Swing. / 100

batting$MarApr_Z.Swing. <- batting$MarApr_Z.Swing. / 100

batting$MarApr_Contact. <- batting$MarApr_Contact. / 100

batting$MarApr_O.Contact. <- batting$MarApr_O.Contact. / 100

batting$MarApr_Z.Contact. <- batting$MarApr_Z.Contact. / 100

view(batting)
```

Checking for NAs

```
sum(is.na(batting))

## [1] 0
```

Seeing which variables are highly correlated to full season OBP

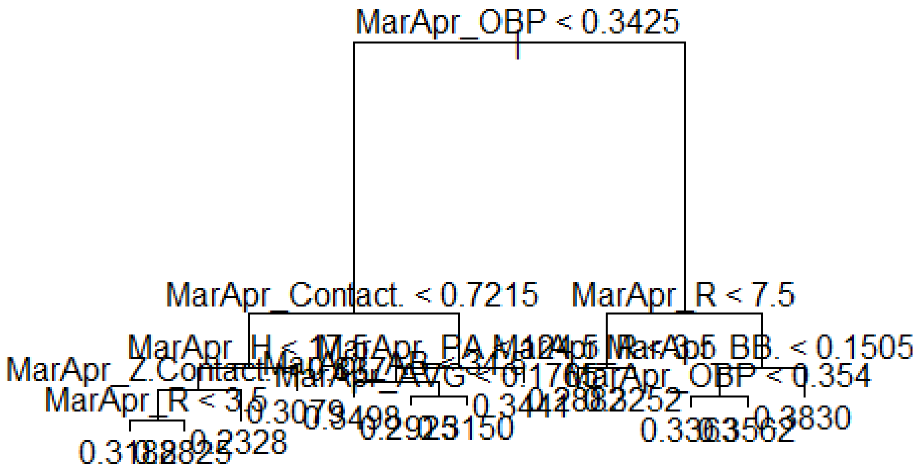
```
b <- batting%>%
  select_if(is.numeric)

batting_cor<- cor(b)

corrplot::corrplot(batting_cor)
```



Creating a training dataset in order to run regression and making a regression tree

```
library(tree)
set.seed (1)

train <- batting

tree.batting <- tree (FullSeason_OBP ~ ., batting )

## Warning in tree(FullSeason_OBP ~ ., batting): NAs introduced by coercion

summary (tree.batting)

##
## Regression tree:
## tree(formula = FullSeason_OBP ~ ., data = batting)
## Variables actually used in tree construction:
## [1] "MarApr_OBP"        "MarApr_Contact."   "MarApr_H"
## [4] "MarApr_Z.Contact." "MarApr_R"          "MarApr_PA"
## [7] "MarApr_AB"         "MarApr_AVG"        "MarApr_BB."
## Number of terminal nodes:  13
## Residual mean deviance:  0.0005751 = 0.1765 / 307
## Distribution of residuals:
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -0.0652800 -0.0152700 -0.0001667  0.0000000  0.0149100  0.0582000
```

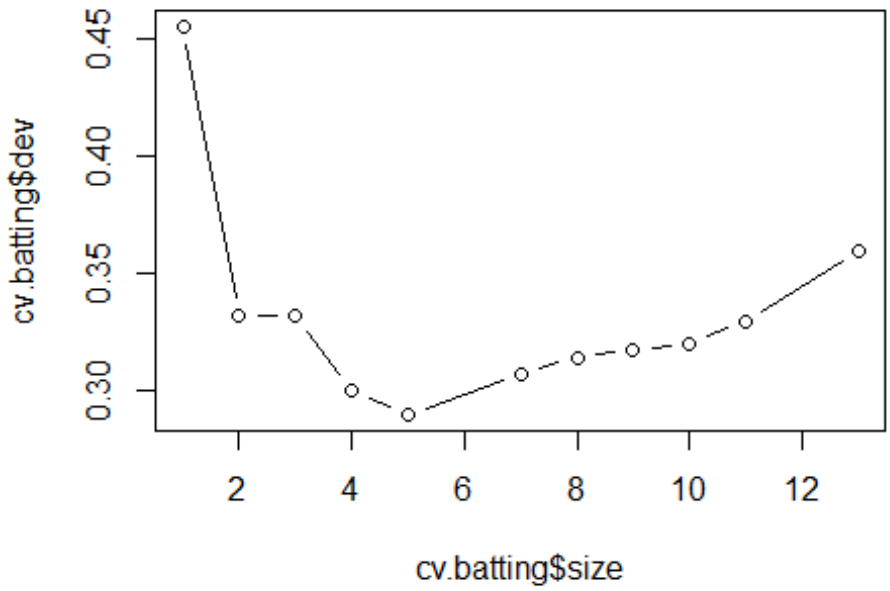
```r
plot (tree.batting)
text (tree.batting , pretty = 0)
```



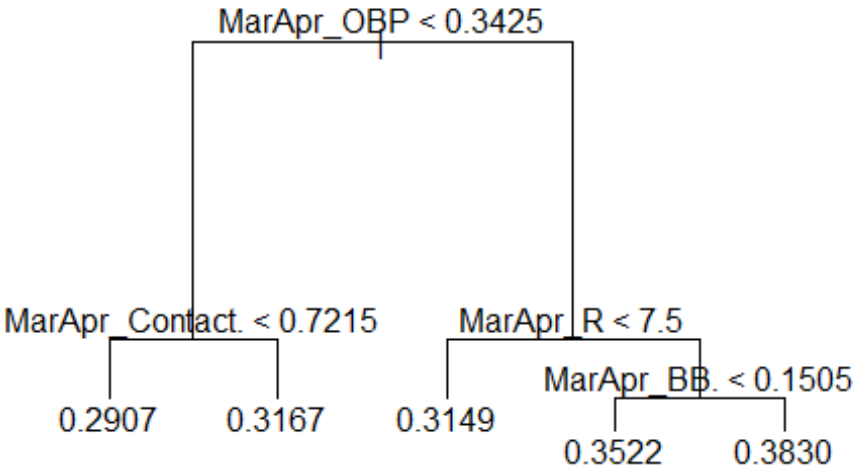
Pruning the tree

```r
cv.batting <- cv.tree(tree.batting)

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion
```

```
## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

## Warning in tree(model = m[rand != i, , drop = FALSE]): NAs introduced by
## coercion

## Warning in pred1.tree(tree, tree.matrix(nd)): NAs introduced by coercion

plot(cv.batting$size , cv.batting$dev, type = "b")
```

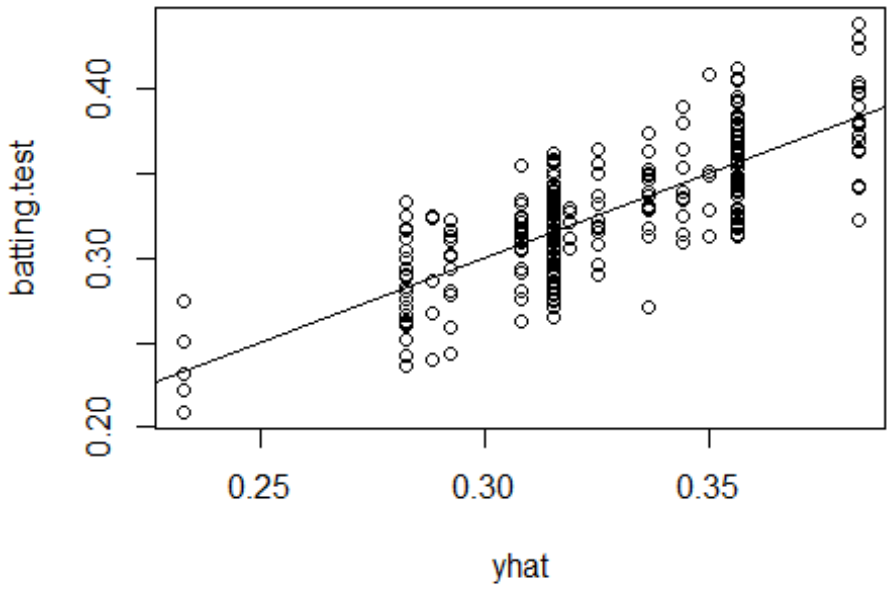



```
prune.batting <- prune.tree (tree.batting , best = 5)
plot(prune.batting)
text(prune.batting , pretty = 0)
```



Making predictions on the based on the tree

```
yhat <- predict (tree.batting , newdata = batting)

## Warning in pred1.tree(object, tree.matrix(newdata)): NAs introduced by
coercion

batting.test <- batting[ , "FullSeason_OBP"]

plot (yhat , batting.test)

abline (0, 1)
```




```
mean ((yhat - batting.test)^2)

## [1] 0.0005516988
```

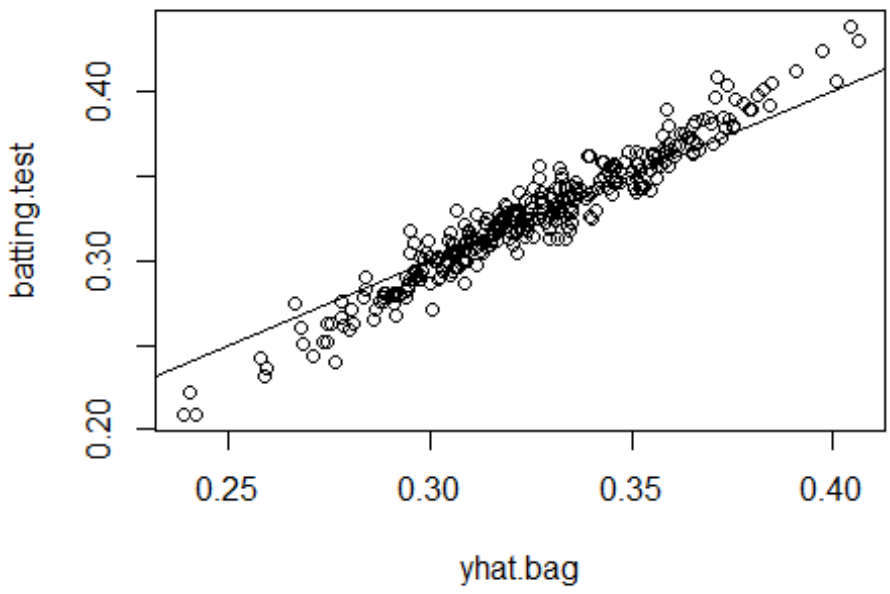
Creating a random forests

```
library (randomForest)

## randomForest 4.7-1.1

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

set.seed (1)

bag.batting <- randomForest(FullSeason_OBP~., data = batting, mtry = 10,
importance = TRUE)

bag.batting

##
## Call:
##  randomForest(formula = FullSeason_OBP ~ ., data = batting, mtry = 10,
importance = TRUE)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 10
##
##           Mean of squared residuals: 0.000814254
##                     % Var explained: 42.32
```

Predicting values from bagging as well as finding MSE

```
yhat.bag <- predict (bag.batting , newdata = batting)

plot (yhat.bag , batting.test)

abline (0, 1)
```




```
mean ((yhat.bag - batting.test)^2)
```

```
## [1] 0.0001342595
```

Predicting values from the random forest as well as MSE

```
set.seed (1)

rf.batting <- randomForest (FullSeason_OBP ~ ., data = batting, mtry = 5,
importance = TRUE)

yhat.rf <- predict (rf.batting, newdata = batting)

mean ((yhat.rf - batting.test)^2)
```

```
## [1] 0.0001405223
```

Checking the importance of each variable

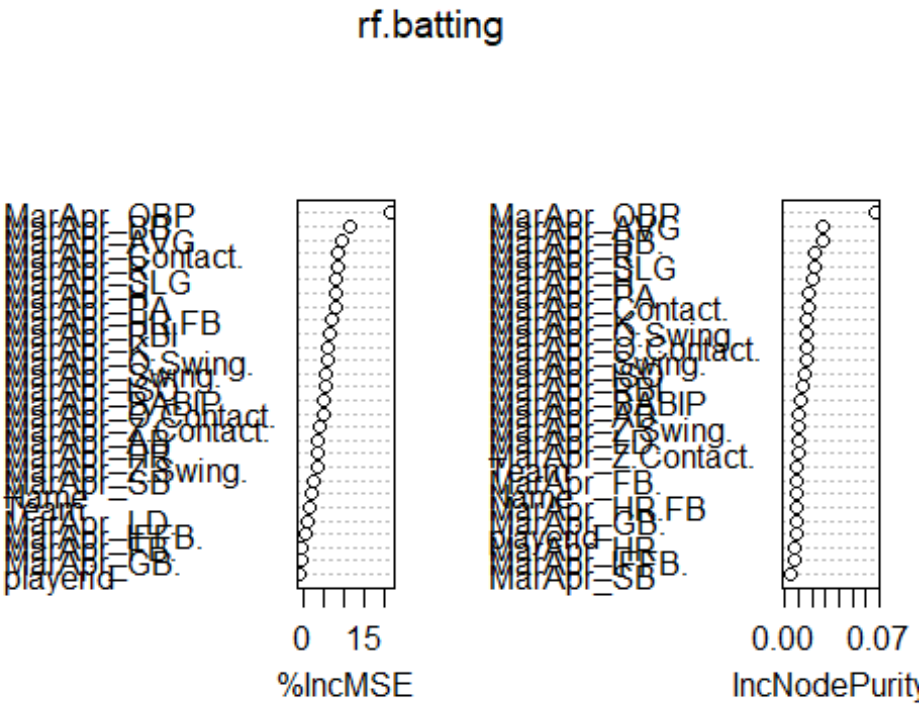
```
importance(rf.batting)
```

```
##                      %IncMSE IncNodePurity
## playerid          -0.53748526   0.007991481
## Name               1.97941192   0.008752540
## Team               1.63994913   0.009046640
## MarApr_PA          8.04245408   0.018102349
```

```
## MarApr_AB           3.96196351   0.010295263
## MarApr_H            8.04584243   0.020724019
## MarApr_HR           3.87098852   0.007556245
## MarApr_R            8.51344212   0.022613447
## MarApr_RBI          6.69428183   0.012757692
## MarApr_SB           2.75748366   0.003756126
## MarApr_BB.         11.71271648   0.028141465
## MarApr_K.           6.39189222   0.016372253
## MarApr_ISO          5.73798847   0.013958028
## MarApr_BABIP        5.27786315   0.011574785
## MarApr_AVG          9.84494569   0.028682710
## MarApr_OBP         21.98449393   0.068135274
## MarApr_SLG          8.12538816   0.021420455
## MarApr_LD.          1.31715898   0.009362325
## MarApr_GB.         -0.42418041   0.008191399
## MarApr_FB.         -0.09439536   0.008774449
## MarApr_IFFB.        0.79593152   0.007016134
## MarApr_HR.FB        7.40234653   0.008408952
## MarApr_O.Swing.     6.06491793   0.016252269
## MarApr_Z.Swing.     3.67872196   0.009553508
## MarApr_Swing.       5.91861275   0.015828296
## MarApr_O.Contact.   5.06299556   0.015904835
## MarApr_Z.Contact.   4.42324747   0.009125142
## MarApr_Contact.     8.65348544   0.016808111
```

Plotting the importance

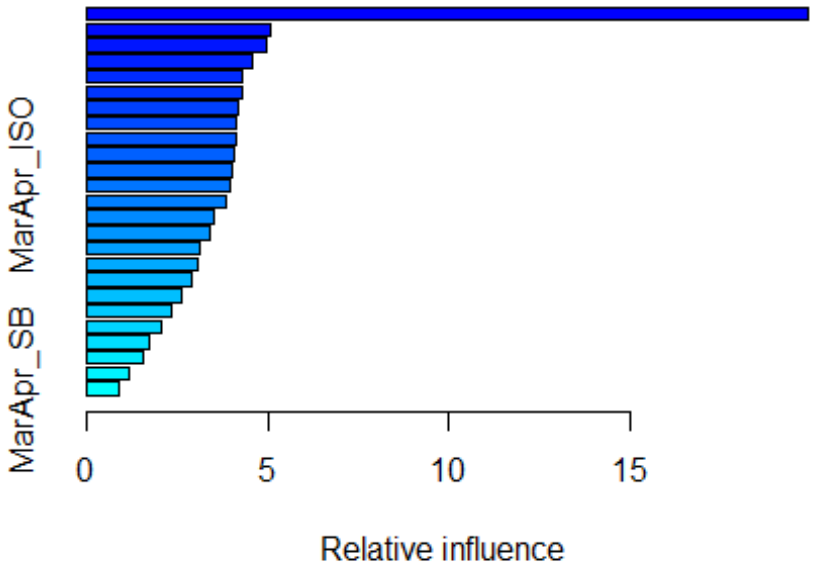
```
varImpPlot(rf.batting)
```



Boosting to see if I can get more accurate results

Removing non numeric variables for boosting

```
batting_boost <- batting[-c(1:3)]

library (gbm)

## Loaded gbm 2.1.8.1

set.seed (1)

boost.batting <- gbm (FullSeason_OBP ~ ., data = batting_boost,
distribution = "gaussian", n.trees = 5000,
interaction.depth = 4, shrinkage = 0.2, verbose = F)

summary(boost.batting)
```




```
##                                   var    rel.inf
## MarApr_OBP                 MarApr_OBP 19.8963652
## MarApr_O.Swing.       MarApr_O.Swing.  5.0525397
## MarApr_IFFB.             MarApr_IFFB.  4.9627735
## MarApr_BB.                 MarApr_BB.  4.5863060
## MarApr_Swing.           MarApr_Swing.  4.3252766
## MarApr_O.Contact. MarApr_O.Contact.  4.3036456
## MarApr_Contact.       MarApr_Contact.  4.2008047
## MarApr_FB.                 MarApr_FB.  4.1513261
## MarApr_RBI                 MarApr_RBI  4.1294944
## MarApr_PA                   MarApr_PA  4.1030489
## MarApr_Z.Swing.       MarApr_Z.Swing.  4.0183693
## MarApr_ISO                 MarApr_ISO  3.9733375
## MarApr_BABIP             MarApr_BABIP  3.8346059
## MarApr_Z.Contact. MarApr_Z.Contact.  3.5270631
## MarApr_GB.                 MarApr_GB.  3.3812539
## MarApr_LD.                 MarApr_LD.  3.1517433
## MarApr_R                     MarApr_R  3.0660269
## MarApr_AVG                 MarApr_AVG  2.9257748
## MarApr_HR.FB             MarApr_HR.FB  2.6009148
## MarApr_K.                   MarApr_K.  2.3352498
## MarApr_H                     MarApr_H  2.0817642
## MarApr_SLG                 MarApr_SLG  1.7372775
## MarApr_AB                   MarApr_AB  1.5660502
## MarApr_HR                   MarApr_HR  1.1800184
## MarApr_SB                   MarApr_SB  0.9089699
```

#predicting values for boosting as well as finding MSE

```
yhat.boost <- predict (boost.batting , newdata = batting, n.trees = 5000)

mean ((yhat.boost - batting.test)^2)

## [1] 2.812912e-28
```

Creating a dataset for the actual and predicted vaules

```
Accuracy <- bind_cols(batting$FullSeason_OBP,yhat, yhat.rf, yhat.bag)

## New names:
## • `` -> `...1`
## • `` -> `...2`
## • `` -> `...3`
## • `` -> `...4`
```

Renaming the columns code from https://sparkbyexamples.com/r-programming/rename-column-in-r/

```
colnames(Accuracy)[1] = "FullSeason_OBP"
colnames(Accuracy)[2] = "Tree_Prediction"
colnames(Accuracy)[3] = "RandomForest_Prediction"
colnames(Accuracy)[4] = "Bagging_Prediction"
```

`