Tristan McRae
Introduction to R
Project Writeup

Predicting Fantasy Football Success

Overview

The goal of this project was to predict how well an NFL running back will perform in fantasy football in a given year using only information about how he did the year prior. Models for relationships between one year's statistics and the next year's fantasy success were trained on data from the 2013 and 2014 seasons and tested on data from the 2014 and 2015 seasons.

Dataset Procurement

Many online sources provide running back statistics for fantasy football on a year by year basis. Datasets pertaining to the top 50 point scorers at running back for 2013, 2014 and 2015 were obtained. Each player represented a row of this dataset and the following information each had its own column: player name, games played, rushes, rushing yards, rushing touchdowns, targets, receptions, receiving yards, receiving touchdowns, fumbles lost and points scored.

One of the biggest programming obstacles was associating a player and their statistics with their age. For whatever reason, it is not common practice to include the age of a player in a table recording their playing statistics. Since conventional wisdom says that a player will improve more when they are young and decline as they age, it was important to include age in any model estimating the improvement of a player from one season to the next. Due to the large volume of players to find the ages of, manual lookup and entry was not be feasible. In order to find the missing age information, web scraping techniques had to be used. Every player in this dataset has a Wikipedia article that contains their current age. These Wikipedia articles have predictable URLs based on the players' names and the age data is consistently in the same section of the article written in the same format. A function was written to take a player's name find their Wikipedia article and extract from it their current age. In cases where datasets from past years were used, their age was adjusted by subtracting the number of years since the season in which the dataset was collected.

Once all raw data for a particular season was collected, more columns of data were created for each player. Continuous data was used to create classifications for simplicity of analysis. Each player was given the designation of "young", "normal", or "old" based on their age and a ranking of 1 – 5 based on how many points they scored in a season with the top 10 scorers being given a 1, 11-20 being given a 2 and so on. These classifications were then converted to factors. Pre-existing statistics were used to create both the efficiency statistics of points per game and yards per carry and the aggregate statistics of total yards, total touchdowns and total touches.

In order to track changes for the same player from one season to the next, players in the dataset of one season were matched with their statistics from the following season where applicable. The player's total points, points per game and 1 – 5 ranking from the following season were added to the dataset. A new column was also created in the dataset to track whether the player scored more points per game in the following season then in the season of the dataset in question. A new data frame was

created to house only data for players who appeared in the current and following season's datasets so that there would not be any missing entries.

Results / Analysis

Random Forest analysis was used in an attempt to classify players into what 1 – 5 ranking they would receive the following year based on their total points, yards per carry, total yards, total touches, total touchdowns, points per game and age classification. The model was created using statistics from 2013 and rankings from 2014 and tested on statistics from 2014 and rankings from 2015. Figure 1 shows the confusion matrix from the 2013-2014 data. Players were classified correctly 36% of the time. Figure 2 shows the confusion matrix from the 2014-2015 data. Players here were classified correctly 30% of the time. These correct classifications are significantly better than the 20% correct classifications that random guessing would yield but still far worse than 100%. This is representative of both the unpredictability of the NFL and the extent to which data can be used to see through the randomness.

	1	2	3	4	5
1	4	3	1	0	0
2	2	3	1	1	0
3	0	2	0	2	0
4	1	2	0	3	0
5	0	0	2	1	0

Figure 1: Confusion matrix for 2013-2014 Data

	0	1	2	3	4	5
1	0	1	2	1	1	0
2	0	2	4	2	0	3
3	0	0	2	0	0	0
4	0	2	1	2	2	1
5	0	0	0	0	0	1

Figure 2: Confusion matrix for 2014-2015 Data

When variables in the random forest were plotted according to importance as seen in Figure 3, it becomes clear that points per game (PPG) is the most important factor in predicting future success.

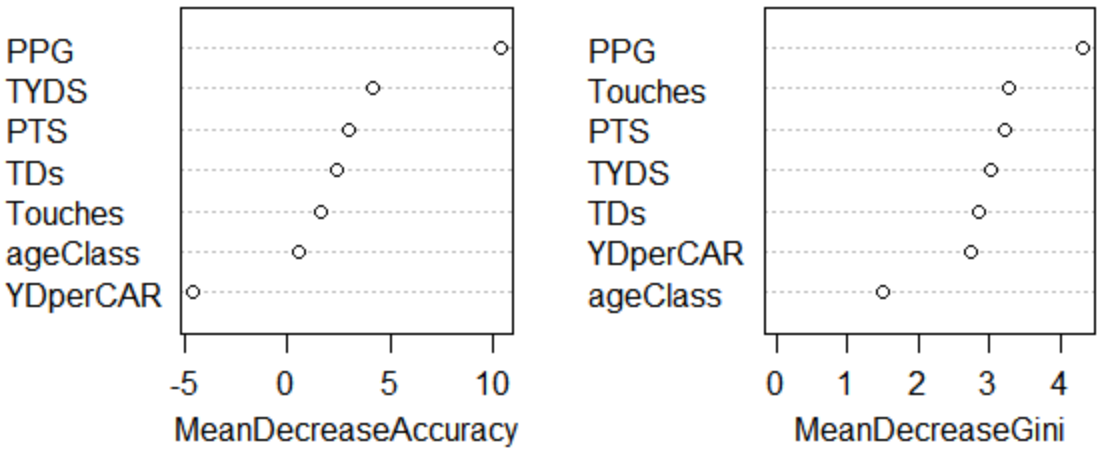

Figure 3: Importance Plot for Random Forest

To simplify the problem, analysis was performed to see how likely a player was to improve their points per game from one year to the next. To find this probability, a generalized linear model was used. In this model, total points, yards per carry, total yards, total touches, total touchdowns, points per game and age classification were used to predict whether or not a player would improve the next year. Figure 4 shows the coefficients found by using this method.

	Estimate	Std. Error	z value	Pr(>|z|)
(Intercept)	-3.79194	5.40419	-0.702	0.483
PTS	0.13995	0.24135	0.580	0.562
YDperCAR	1.14144	1.22484	0.932	0.351
TYDS	-0.01291	0.02333	-0.553	0.580
Touches	0.00562	0.02194	0.256	0.798
TDs	-0.93591	1.41288	-0.662	0.508
PPG	-0.26228	0.34473	-0.761	0.447
ageClassOld	-0.79066	1.17443	-0.673	0.501
ageClassYoung	1.47034	1.19471	1.231	0.218

Figure 4: GLM Coefficients for 2013-2014 PPG Improvement

The direction of these coefficients suggests that young players who score a lot of points, average a lot of yards per carry and get a lot of touches are set to improve. Older players who have more overall yards, touchdowns and points per game tend to regress. This may be due to the fact that players with a lot of touchdowns, which tend to be somewhat random, have artificially high point totals and cannot maintain the scoring. The discrepancy between points per game and total points having opposite effects on improvement likelihood could be due to the fact that the only time these two values diverge is when a player gets injured midseason which could lead to a worse performance the next year if the injury lingers.

When used on the 2013-2014 data it trained on, this methods accurately predicts if a player will improve 71% of the time but that drops to 56% of the time when the same method is used on 2014-2015 data. Interestingly, when age is excluded as a factor, the accuracy of the 2014-2015 predictions rises to 67%. Inspection revealed that this was due to a large number of young players who did moderately well in 2014 regressing in 2015. It is difficult to say if this year is an outlier in this regard but

this suggests that age may be an overrated factor in gauging a player's potential for improvement the following year and trying to account for it only results in overfitting.

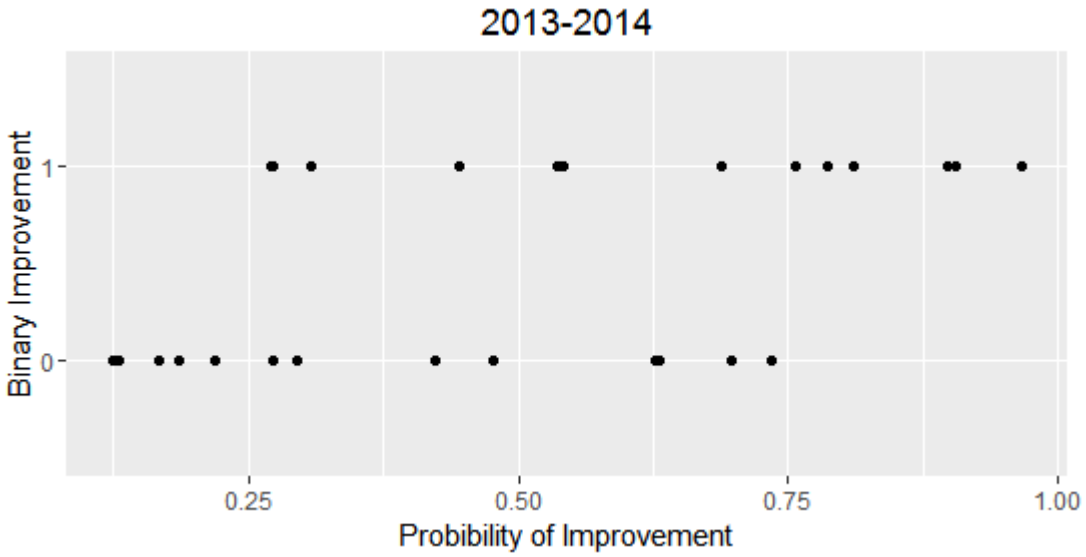

Figure 5: Predicted vs Actual Improvement for Training Data

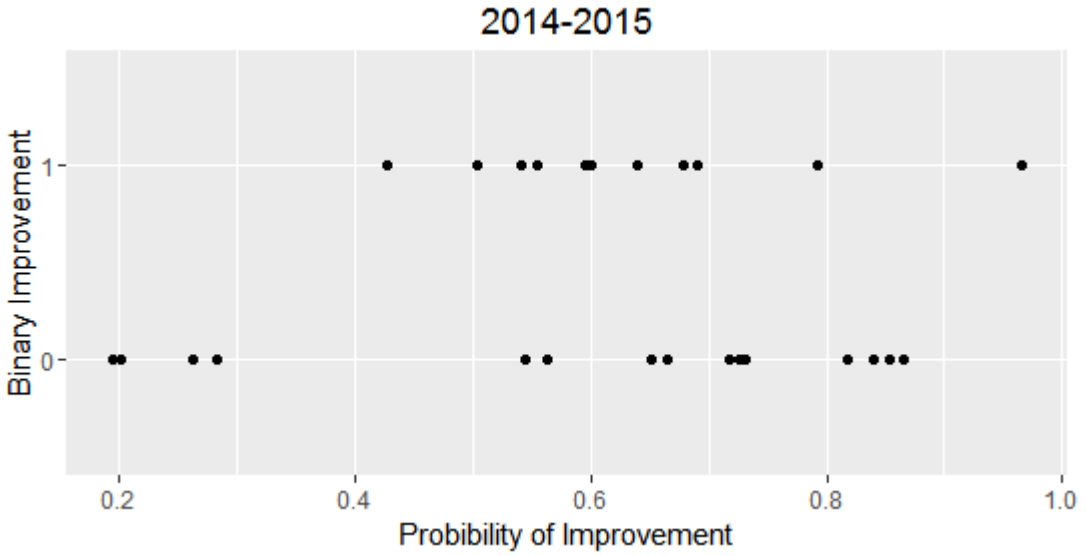

Figure 6: Predicted vs Actual Improvement for Test Data

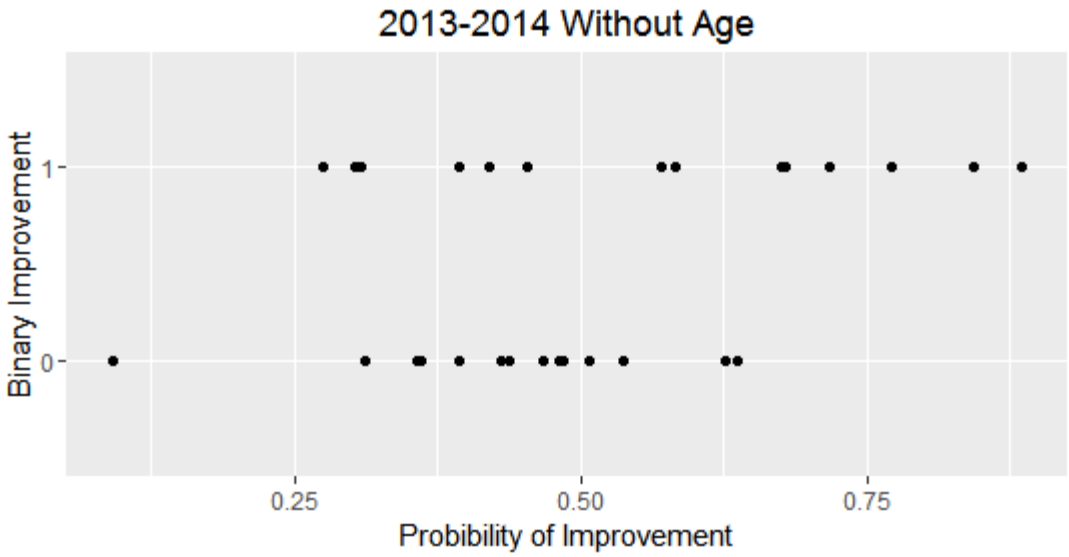

Figure 7: Predicted vs Actual Improvement for Training Data without Accounting for Age

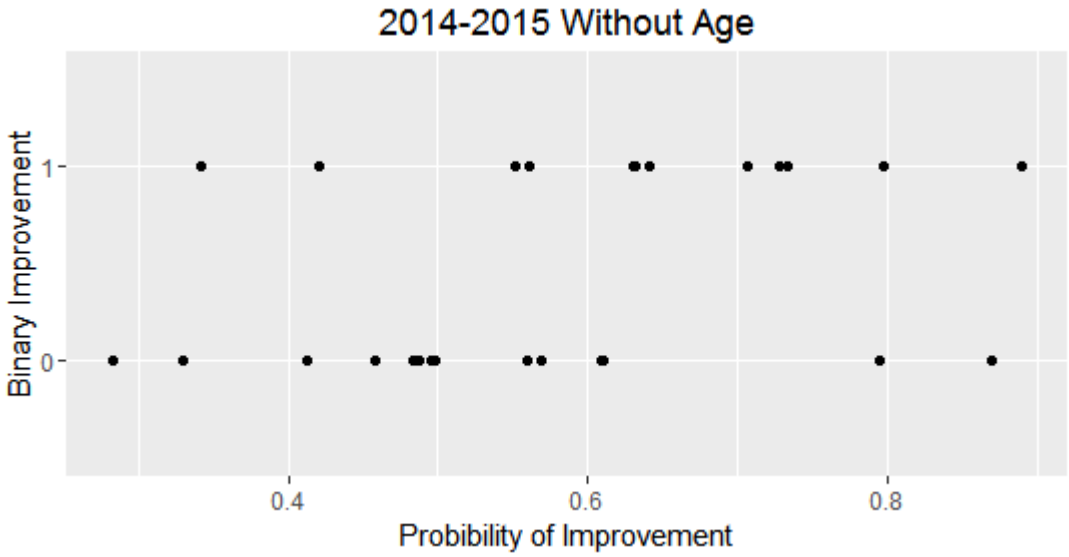

Figure 8: Predicted vs Actual Improvement for Test Data without Accounting for Age

As shown in Figures 5-8 in all cases, the players that, according to the model, are most certain will regress and most certain to improve do in fact regress and improve respectively.

Appendix (code)

```
rm(list = ls())

#Read stats from 2015 and scoring data from 2016
RBData2013 = read.csv("C:\\Users\\tm9045\\Desktop\\Stanford\\Fall2016\\R\\2013RBData.csv",
stringsAsFactors=FALSE)
RBData2014 = read.csv("C:\\Users\\tm9045\\Desktop\\Stanford\\Fall2016\\R\\2014RBData.csv",
stringsAsFactors=FALSE)
RBData2015 = read.csv("C:\\Users\\tm9045\\Desktop\\Stanford\\Fall2016\\R\\2015RBData.csv",
stringsAsFactors=FALSE)

library(XML)
library(RCurl)
library(rvest)

specialNames = c("Chris Johnson", "David Johnson", "Andre Brown", "Doug Martin")

#Grabs player age from wikipedia
getAge = function(name){
  search = gsub(' ', "_", name)
  url = paste("https://en.wikipedia.org/wiki/", search, sep="")
  html = read_html(url)
  age = html_nodes(html, ".ForceAgeToShow")
  textAge = html_text(age)
  numOnly = as.integer(substr(textAge, 6, 7))
  gotIt = length(numOnly)
  if (gotIt == 0){
    url = paste("https://en.wikipedia.org/wiki/", search, "_(American_football)",sep="")
    if (is.element(name, specialNames)){
      url = paste("https://en.wikipedia.org/wiki/", search, "_(running_back)",sep="")
    }
    html = read_html(url)
    age = html_nodes(html, ".ForceAgeToShow")
    textAge = html_text(age)
    numOnly = as.integer(substr(textAge, 6, 7))
    gotIt = length(numOnly)
  }
  return(numOnly)
}

#count number of rows in datasets
numRows2013=nrow(RBData2013)
numRows2014=nrow(RBData2014)
numRows2015=nrow(RBData2015)
```

```
#Classify players into draft rounds based on rank
RBData2013$draft = 0
RBData2014$draft = 0
RBData2015$draft = 0

RBData2013$draft = as.integer(RBData2013$draft)
RBData2014$draft = as.integer(RBData2014$draft)
RBData2015$draft = as.integer(RBData2015$draft)

for (i in 1:numRows2013){
  if (1<=160){
    decRound = (i-1)/10+1
    round = floor(decRound)
    RBData2013$draft[i] = round
  } else {
    RBData2013$draft[i] = 17
  }
}
for (i in 1:numRows2014){
  if (1<=160){
    decRound = (i-1)/10+1
    round = floor(decRound)
    RBData2014$draft[i] = round
  } else {
    RBData2014$draft[i] = 17
  }
}
for (i in 1:numRows2015){
  if (1<=160){
    decRound = (i-1)/10+1
    round = floor(decRound)
    RBData2015$draft[i] = round
  } else {
    RBData2015$draft[i] = 17
  }
}
for (i in 1:numRows2016){
  if (1<=160){
    decRound = (i-1)/10+1
    round = floor(decRound)
    RBData2016$draft[i] = round
  } else {
    RBData2016$draft[i] = 17
  }
}

RBData2013$draft = as.factor(RBData2013$draft)
RBData2014$draft = as.factor(RBData2014$draft)
```

```
RBData2015$draft = as.factor(RBData2015$draft)
RBData2016$draft = as.factor(RBData2016$draft)

#Add efficiency statistics
RBData2013$YDperCAR = RBData2013$YDS / RBData2013$RUSH
RBData2013$TYDS = RBData2013$YDS + RBData2013$ReYds
RBData2013$TDs = RBData2013$RuTD + RBData2013$ReTD
RBData2013$Touches = RBData2013$RUSH + RBData2013$REC
RBData2013$PPG = RBData2013$PTS / RBData2013$GMS

RBData2014$YDperCAR = RBData2014$YDS / RBData2014$RUSH
RBData2014$TYDS = RBData2014$YDS + RBData2014$ReYds
RBData2014$TDs = RBData2014$RuTD + RBData2014$ReTD
RBData2014$Touches = RBData2014$RUSH + RBData2014$REC
RBData2014$PPG = RBData2014$PTS / RBData2014$GMS

RBData2015$YDperCAR = RBData2015$YDS / RBData2015$RUSH
RBData2015$TYDS = RBData2015$YDS + RBData2015$ReYds
RBData2015$TDs = RBData2015$RuTD + RBData2015$ReTD
RBData2015$Touches = RBData2015$RUSH + RBData2015$REC
RBData2015$PPG = RBData2015$PTS / RBData2015$GMS

#add columns to data corresponding to player's score and draft rank next year
RBData2013$scoreNext = 0
RBData2013$draftNext = 0
RBData2013$ppgImprove = 0

RBData2014$scoreNext = 0
RBData2014$draftNext = 0
RBData2014$ppgImprove = 0

RBData2015$scoreNext = 0
RBData2015$draftNext = 0
RBData2015$ppgImprove = 0

#fill in 2016 score column in 2015 data
for (i in 1:numRows2013){
 if(RBData2013$Player[i] != "Giovani Bernard"){
  RBData2013$age[i] = getAge(RBData2013$Player[i]) - 3
 } else{
  RBData2013$age[i] = 24 - 3
 }
 for (j in 1:numRows2014){
  if (RBData2013$Player[i] == RBData2014$Player[j]) {
   RBData2013$scoreNext[i] = RBData2014$PTS[j]
   RBData2013$draftNext[i] = RBData2014$draft[j]
   RBData2013$ppgNext[i] = RBData2014$PPG[j]
   if (RBData2014$PPG[j]>RBData2013$PPG[i]){
```

```
      RBData2013$ppgImprove[i] = 1
     }
    }
   }
}
for (i in 1:numRows2014){
  if(RBData2014$Player[i] != "Giovani Bernard"){
    print(RBData2014$Player[i])
    RBData2014$age[i] = getAge(RBData2014$Player[i]) - 2
  } else{
    RBData2014$age[i] = 24 - 2
  }
  for (j in 1:numRows2015){
    if (RBData2014$Player[i] == RBData2015$Player[j]) {
      RBData2014$scoreNext[i] = RBData2015$PTS[j]
      RBData2014$draftNext[i] = RBData2015$draft[j]
      RBData2014$ppgNext[i] = RBData2015$PPG[j]
      if (RBData2015$PPG[j]>RBData2014$PPG[i]){
        RBData2014$ppgImprove[i] = 1
      }
    }
  }
}

for (i in 1:numRows2013){
  if (RBData2013$age[i]<=24){
    RBData2013$ageClass[i]="Young"
  }
  else if (RBData2013$age[i]>=28){
    RBData2013$ageClass[i]="Old"
  }
  else{
    RBData2013$ageClass[i]="Normal"
  }
}

for (i in 1:numRows2014){
  if (RBData2014$age[i]<=24){
    RBData2014$ageClass[i]="Young"
  }
  else if (RBData2014$age[i]>=28){
    RBData2014$ageClass[i]="Old"
  }
  else{
    RBData2014$ageClass[i]="Normal"
  }
}
```

```
RBData2013$ageClass = as.factor(RBData2013$ageClass)
RBData2014$ageClass = as.factor(RBData2014$ageClass)

RBData2013$draftNext = as.factor(RBData2013$draftNext)
RBData2014$draftNext = as.factor(RBData2014$draftNext)

RBData2013$ppgImprove = as.factor(RBData2013$ppgImprove)
RBData2014$ppgImprove = as.factor(RBData2014$ppgImprove)

#make new dataframes only including players who have scores in consecutive years
RBData2013relevant = RBData2013[1,]
for (i in 2:numRows2013){
  if (RBData2013$scoreNext[i] != 0){
    RBData2013relevant = rbind(RBData2013relevant,RBData2013[c(i),])
  }
}

RBData2014relevant = RBData2014[1,]
for (i in 2:numRows2014){
  if (RBData2014$scoreNext[i] != 0){
    RBData2014relevant = rbind(RBData2014relevant,RBData2014[c(i),])
  }
}

#Use randomforrest to see which factors most affect draft position
library(randomForest)
set.seed(1)
library(party)

draft.rf = randomForest(draftNext ~ PTS + YDperCAR + TYDS + Touches + TDs + PPG + ageClass,
data=droplevels(RBData2013relevant), importance=TRUE, proximity=TRUE)
importance(draft.rf)
plot(draft.rf, log="y")
print(draft.rf)
varImpPlot(draft.rf)
ct = ctree(draftNext ~ PTS + YDperCAR + TYDS + Touches + TDs + PPG + ageClass, data =
RBData2013relevant)
plot(ct, main="Conditional Inference Tree")
prediction = predict(draft.rf, RBData2014relevant)
t = table(prediction, RBData2014relevant$draftNext)
print(t)


improve.rf = randomForest(ppgImprove ~ PTS + YDperCAR + TYDS + Touches + TDs + PPG + ageClass,
data=droplevels(RBData2013relevant), importance=TRUE, proximity=TRUE)
importance(improve.rf)
plot(improve.rf, log="y")
print(improve.rf)
```

```
varImpPlot(improve.rf)
ct = ctree(ppgImprove ~ PTS + YDperCAR + TYDS + Touches + TDs + PPG + ageClass, data =
RBData2013relevant)
plot(ct, main="Conditional Inference Tree")
prediction = predict(improve.rf, RBData2014relevant)
t = table(prediction, RBData2014relevant$ppgImprove)
print(t)

#% chance that each player will improve or not the next year
impFit2013 = glm(ppgImprove ~ PTS + YDperCAR + TYDS + Touches + TDs + PPG + ageClass, data =
RBData2013relevant, family = "binomial")
summary(impFit2013)
coef(impFit2013)
RBData2013relevant$ppgPredict= predict(impFit2013, newdata = RBData2013relevant, type =
"response")

numCorrect2013=0
for (i in 1:nrow(RBData2013relevant)){
  if (round(RBData2013relevant$ppgPredict[i])==RBData2013relevant$ppgImprove[i]){
    numCorrect2013 = numCorrect2013+1
  }
}
accuracy2013 = numCorrect2013/nrow(RBData2013relevant)
accuracy2013

RBData2014relevant$ppgPredict= predict(impFit2013, newdata = RBData2014relevant, type =
"response")

numCorrect2014=0
for (i in 1:nrow(RBData2014relevant)){
  if (round(RBData2014relevant$ppgPredict[i])==RBData2014relevant$ppgImprove[i]){
    numCorrect2014 = numCorrect2014+1
  }
}
accuracy2014 = numCorrect2014/nrow(RBData2014relevant)
accuracy2014

library(ggplot2)
ggplot(data = RBData2013relevant, aes(x = ppgPredict, y = ppgImprove)) + geom_point() + ggtitle("2013-
2014 Without Age") + labs(x="Probibility of Improvement",y="Binary Improvement")
ggplot(data = RBData2014relevant, aes(x = ppgPredict, y = ppgImprove)) + geom_point() + ggtitle("2014-
2015 Without Age") + labs(x="Probibility of Improvement",y="Binary Improvement")
```