MLB HOF Pitcher Prediction

Patricia McSharry

6/9/2019

Patricia McSharry, HarvardX: PH125.9x Data Science Certificate

MLB Hall of Fame Pitcher Prediction - CYO Captsone Project

This code was developed in R Studio Version 1.2.1335 and R version 3.5.2.

Objective

** Write an algorithn to predict whether a pitcher will be inducted into the Major League Baseball Hall of Fame **

Executive Summary

The purpose of this project is to create an algorithm to predict whether a Major League Baseball (MLB) pitcher will be inducated into the Baseball Hall of Fame (HOF). It uses Random Forest approach, randomForest.

The final model uses the predictors Total Wins and Total Outs Pitched with tuned ntree with an accuracy of 0.9677419

Further analysis of possible improvements are in the conclusion section.

Baseball Hall of Fame

The Baseball Hall of Fame contains the best players and managers.

Hall of Fame Voting

Eligible members of the Baseball Writers' Association of America (BBWAA) cast Hall of Fame votes. Any candidate receiving votes on seventy-five percent (75%) of the ballots cast shall be elected to membership in the National Baseball Hall of Fame. Further rules govern staying on the ballot.

Eligible Candidates

Candidates to be eligible must meet the following requirements, these requirements are accounted for in the algorithm.

1. A baseball player must have been active as a player in the Major Leagues at some time during a period beginning fifteen (15) years before and ending five (5) years prior to election.
2. Player must have played in each of ten (10) Major League championship seasons, some part of which must have been within the period described in 3(A).
3. Player shall have ceased to be an active player in the Major Leagues at least five (5) calendar years preceding the election but may be otherwise connected with baseball.
4. In case of the death of an active player or a player who has been retired for less than five (5) full years, a candidate who is otherwise eligible shall be eligible in the next regular election held at least six (6) months after the date of death or after the end of the five (5) year period, whichever occurs first.
5. Any player on Baseball's ineligible list shall not be an eligible candidate.

Data source

The project uses the Sean Lahman Baseball Database which is avaiable in the Lahman package as a set of R data.frames. It includes data on pitching, hitting and Hall Of Fame and other tables from 1871 through 2018.

The project uses the Master, Pitching and HallOfFame tables.

Pitching Terms Explanation

Basic terms

playerID Player ID code yearID Year of the data W Wins L Losses G Games GS Games Started CG Complete Games SHO Shutouts SV Saves IPOuts Outs Pitched H Hits ER Earned Runs HR Homeruns BB Walks SO Strikeouts

Terms calculated in the code

Terms explained

prefex t - used to denote total for career

tERA - Earned run average for career Definition: 9 X (Total earned runs allowed / total innings pitched) - There are nine inings in a game Code: tERA = (tER/(tIPouts / 3)) *9)

tWP - Winning percentage for career Defintion: total wins / (total wins + total loses) tWP = tW/ (tW + tL))

tWHIP - walks plus hits per inning pitched for career Defintion: total hits + total walks / total innings pitched tWHIP = (tH + tBB) / (tIPouts / 3)

Modeling method

Random forests are a very popular machine learning approach that addresses the shortcomings of decision trees using a clever idea. The goal is to improve prediction performance and reduce instability by averagingmultiple decision trees (a forest of trees constructed with randomness). The model uses the randomForest package.

First, randomForest on the trainPitch training data set with all the predictors was run and total win (tW) and total outs pitched (tIPOUt) were the most important predictors. Tried different models using these predictors and tuned the best model. The final model uses total wins and total innings pitched as the predictors and a tuned ntree paramaters. Accuracy of the final model is 0.9677419.

```
# install packages as needed
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: caret
```

```
## Warning: package 'caret' was built under R version 3.5.3
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
## Warning: package 'ggplot2' was built under R version 3.5.3
```

```
if(!require(Lahman)) install.packages("Lahman", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: Lahman
```

```
## Warning: package 'Lahman' was built under R version 3.5.3
```

```
if(!require(randomForest)) install.packages("randomForest", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: randomForest
```

```
## Warning: package 'randomForest' was built under R version 3.5.3
## randomForest 4.6-14
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
if(!require(dplyr)) install.packages("dplyr", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: dplyr
## Warning: package 'dplyr' was built under R version 3.5.3
##
## Attaching package: 'dplyr'
## The following object is masked from 'package:randomForest':
##
##     combine
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
if(!require(lubridate)) install.packages("lubridate", repos = "http://cran.us.r-project.org")
```

```
## Loading required package: lubridate
## Warning: package 'lubridate' was built under R version 3.5.3
##
## Attaching package: 'lubridate'
## The following object is masked from 'package:base':
##
##     date
```

```
# year of latest data in lahman datset
latestyear = max(Pitching$yearID)

# use Pitching table, which stores data on a yearly basis
# group by playerID and calcuation totals wins, total loses, total saves, total hits, total bases on ba
# total earned runs, total outs pitched, total strike outs, and years played
# per HOF eligibility rules, players need 10 years in the majors to qualify for the HOF, filter out
# players with less than 10 years in the majors
PitchingSum <- Pitching %>%
  group_by(playerID) %>%
  summarize(tW = sum(W), tL = sum(L), tSV = sum(SV),
    tH = sum(H), tBB = sum(BB),  tER = sum(ER),
    tIPouts=sum(IPouts), tSO = sum(SO),
    yearsPlayed = n()) %>%
    filter(yearsPlayed  > 9)
```

```r
# players need to be retired 5 years to be HOF eligible, store in table HOFEligible table
HOFEligible <- Master %>% filter(substring(finalGame,1,4) < toString(latestyear - 5)) %>%
select(playerID)

# code for special rule about a player dying in the past year, store in adjustEligible table
adjustEligible <- semi_join(Pitching, (Master %>% filter(as.numeric(format(Master$deathDate, "%Y")) ==

# adjust the HOFEligible table to account for players dying in the past year
HOFEligibile <- semi_join(HOFEligible , adjustEligible, by="playerID")

# join the PitchingSum table to the HOFEligible table to account for all HOF eligibiliy rules
PitchingSum <- semi_join(PitchingSum,HOFEligible)
```

```
## Joining, by = "playerID"
```

```r
# calculate career ERA (earned run average)
PitchingSum <- PitchingSum %>%
  mutate(tERA = (tER/(tIPouts / 3)) *9)

# calcualte career WHIP (walks plus hits per inning pitched for career)
PitchingSum <- PitchingSum %>%
  mutate(tWHIP = (tH + tBB) / (tIPouts / 3))

# calculate career winning percentage
PitchingSum <- PitchingSum %>%
  mutate(tWP = tW/ (tW + tL))

# the hall of fame has managers and players inducted, create a table of all players in the HOF
IndHOF <- HallOfFame %>% filter(inducted == 'Y' & category=='Player') %>% select(playerID) %>% mutate(i

# creat table PSumHOF by joining PitchingSum and IndHOF, which gives a pitching career level
# table.  induct column is Y if player is in the HOF and N if not
# induct is the column to predict
PSumHOF <-
  left_join(PitchingSum, IndHOF, by='playerID')

# set induct to 'N' if player is not in the HOF
PSumHOF <- PSumHOF %>%
  mutate(induct = coalesce(induct, 'N'))

# induct column needs to be a factor
PSumHOF <- PSumHOF %>%
  mutate(induct = as.factor(PSumHOF$induct))

# create train and test sets
set.seed(1)
test_index <- createDataPartition(y = PSumHOF$induct, times = 1, p = 0.5, list = FALSE)
trainPitch <- PSumHOF[-test_index,]
testPitch <- PSumHOF[test_index,]

# use random forest

# randomForest with all predictors
set.seed(1)
```

```
fit <- randomForest(induct ~
tW + tL + tSV + tH + tBB + tER + tIPouts + tSO + yearsPlayed + tERA + tWHIP + tWP,
data=trainPitch)
pitchhof_results <- data_frame(model = "All Predictors",
  Accuracy = confusionMatrix(predict(fit, testPitch), testPitch$induct)$overall["Accuracy"])
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

```
# print out the table
pitchhof_results %>% knitr::kable()
```

model	Accuracy
All Predictors	0.9631336

```
# list importance of predictors, note that total wins and total innings pitched are the most important
importance(fit)
```

```
##              MeanDecreaseGini
## tW                  14.362833
## tL                   3.379842
## tSV                  2.516652
## tH                   7.986740
## tBB                  3.342200
## tER                  4.301148
## tIPouts             11.705167
## tSO                  5.392015
## yearsPlayed          1.789094
## tERA                 3.097611
## tWHIP                4.949687
## tWP                  3.836056
```

```
# use randomForest with total wins predictor
set.seed(1)
fit2 <- randomForest(induct ~ tW ,  data=trainPitch)

# Using results of training, predict induct in the test set, save accuracy in pitchof_results table
pitchhof_results <- bind_rows(pitchhof_results,
  data_frame(model="Total Wins",
  Accuracy=confusionMatrix(predict(fit2, testPitch), testPitch$induct)$overall["Accuracy"]))

# use randomForest with total innings pitched predictor
set.seed(1)
fit3 <- randomForest(induct ~ tIPouts, data=trainPitch)

# Using results of training, predict induct in the test set, save accuracy in pitchof_results table
pitchhof_results <- bind_rows(pitchhof_results,
  data_frame(model="Total Outs Pitched",
  Accuracy=confusionMatrix(predict(fit3, testPitch), testPitch$induct)$overall["Accuracy"]))

# use randomForest with total wins and total innings pitched predictors
set.seed(1)
fit4 <- randomForest(induct ~ tW + tIPouts, data=trainPitch)
```

```r
# Using results of training, predict induct in the test set, save accuracy in pitchof_results table
pitchhof_results <- bind_rows(pitchhof_results,
  data_frame(model="Total Wins and Total Outs Pitched",
  Accuracy=confusionMatrix(predict(fit4, testPitch), testPitch$induct)$overall["Accuracy"]))

# print accuracy results of the models
pitchhof_results %>% knitr::kable()
```

model	Accuracy
All Predictors	0.9631336
Total Wins	0.9493088
Total Outs Pitched	0.9493088
Total Wins and Total Outs Pitched	0.9662058

```r
# the total wins and total innings pitched model had the best results, tune this model
bestparm <- seq(1, 2, 1)

# tune mtry
accuracy <- sapply(bestparm,function(sz) {
 set.seed(1)
 fitbest <- randomForest(induct ~ tW + tIPouts, mtry=sz, data=trainPitch)
 accuracy=confusionMatrix(predict(fitbest, testPitch), testPitch$induct)$overall["Accuracy"]
} )

# accuracy is less than previous max, use default mtry
bestmtry = bestparm[which.max(accuracy)]
bestmtry
```

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

```r
max(accuracy)
```

```
## [1] 0.9662058
```

```r
# tune ntree
bestparm <- seq(1, 500, 1)
accuracy <- sapply(bestparm,function(sz) {
 set.seed(1)
 fitbest <- randomForest(induct ~ tW + tIPouts, ntree=sz, data=trainPitch)
 accuracy=confusionMatrix(predict(fitbest, testPitch), testPitch$induct)$overall["Accuracy"]
} )

# superior accuracy to using default for ntree, use tuned ntree for better results, ntree is 24
bestntree = bestparm[which.max(accuracy)]
bestntree
```

```
## [1] 24
```

```r
max(accuracy)
```

```
## [1] 0.9677419
```

```r
# tune maxnodes
bestparm <- seq(10,200,10)
```

```
accuracy <- sapply(bestparm,function(sz) {
 set.seed(1)
 fitsz <- randomForest(induct ~ tW + tWP +tSV, maxnodes=sz, data=trainPitch)
 accuracy=confusionMatrix(predict(fitsz, testPitch), testPitch$induct)$overall["Accuracy"]
} )

# accuracy is less than previous max, use default mtry
bestmax = bestparm[which.max(accuracy)]
bestmax
```

```
## [1] 40
```

```
max(accuracy)
```

```
## [1] 0.9646697
```

```
# run total wins and total innings pitched model using best ntree value
set.seed(1)
fit5 <- randomForest(induct ~ tW + tIPouts, ntree=bestntree, data=trainPitch)

# Using results of training, predict induct in the test set, save accuracy in pitchof_results table
pitchhof_results <- bind_rows(pitchhof_results,
  data_frame(model="Total Wins and Total Outs Pitched with tuned ntree ",
  Accuracy=confusionMatrix(predict(fit5, testPitch), testPitch$induct)$overall["Accuracy"]))

# tuned total wins and total outs pitched model produces highest accuracy
pitchhof_results %>% knitr::kable()
```

model	Accuracy
All Predictors	0.9631336
Total Wins	0.9493088
Total Outs Pitched	0.9493088
Total Wins and Total Outs Pitched	0.9662058
Total Wins and Total Outs Pitched with tuned ntree	0.9677419

```
# create a list of pitchers for whom the algorithm predicted induct incorrectly
# it will be discussed in the conclusion section
y_hat <- predict(fit5, testPitch)
indx <- (y_hat != testPitch$induct)
predErr <- testPitch[indx,]
predErr <- inner_join(predErr, Master, by="playerID")
errList <- select(predErr, c(nameLast,nameFirst, finalGame, induct))
print(errList, n=100)
```

```
## # A tibble: 22 x 4
##    nameLast  nameFirst finalGame  induct
##    <chr>     <chr>     <chr>      <fct>
##  1 Bender    Chief     1925-07-21 Y
##  2 Bunning   Jim       1971-09-03 Y
##  3 Clemens   Roger     2007-09-16 N
##  4 Coveleski Stan      1928-08-03 Y
##  5 Dean      Dizzy     1947-09-28 Y
##  6 Drysdale  Don       1969-08-05 Y
##  7 Gossage   Rich      1994-08-08 Y
```

```
##  8 Haines    Jesse     1937-09-10 Y
##  9 Hunter    Catfish   1979-09-17 Y
## 10 John      Tommy     1989-05-25 N
## 11 Koufax    Sandy     1966-10-02 Y
## 12 Lemon     Bob       1958-07-01 Y
## 13 Moyer     Jamie     2012-05-27 N
## 14 Mullane   Tony      1894-07-26 N
## 15 Mussina   Mike      2008-09-28 N
## 16 Newhouser Hal       1955-05-03 Y
## 17 Sutter    Bruce     1988-09-09 Y
## 18 Vance     Dazzy     1935-08-14 Y
## 19 Waddell   Rube      1910-08-01 Y
## 20 Walsh     Ed        1917-09-11 Y
## 21 Wells     David     2007-09-28 N
## 22 Wilhelm   Hoyt      1972-07-10 Y
```

Conclusion

Comments on several players whose induction was incorrectly predicted:

Roger Clemens - was invovled in the steroid scandal and lied to Congress about it. The BBWWA writers will never vote him in, regardless of his stellar stats.

Mike Mussina - was voted into the Hall for the class of 2019.

The model could be further refined to split pitchers into relievers and starters. In that case Rich (Goose) Gossage and Bruce Sutter be correctly predicted as they were relievers.