HOF Batter Prediction

Jacob Berkowitz

5/16/2018

# Goal

We want to predict whether a player will be a hall of famer based on his career stats.

## Step 1: Manipulate data and create usable dataframe

data("HallOfFame")  
#get dataset  
df<-data\_frame(HallOfFame)

## Warning: `data\_frame()` is deprecated as of tibble 1.1.0.  
## Please use `tibble()` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_warnings()` to see where this warning was generated.

#only players  
df1<-HallOfFame%>%  
 dplyr::filter(category=='Player')%>%  
 group\_by(playerID)%>%  
 filter(yearID==last(yearID))%>%  
 mutate(rate=votes/ballots)  
#add info  
df2<-left\_join(df1,People,by="playerID")  
#fix batting  
bat<-Batting  
index<-is.na(bat)  
bat[index]<-0  
bat<-bat%>%  
 group\_by(playerID)%>%  
 summarise(seasons=n(),  
 hits=sum(H),  
 AB=sum(AB),  
 X2B=sum(X2B),  
 X3B=sum(X3B),  
 HR=sum(HR),  
 K=sum(SO),  
 BB=sum(BB),  
 Games=sum(G),  
 SB=sum(SB))%>%  
 mutate(avg=hits/AB)%>%  
 mutate(slug=((hits-X2B-X3B-HR)+2\*X2B+3\*X3B+4\*HR)/AB)%>%  
 mutate(kRate=K/AB)%>%  
 ungroup()%>%  
 mutate(BBperK=BB/K)

## `summarise()` ungrouping output (override with `.groups` argument)

#join batting  
df3<-left\_join(bat,df2,by='playerID')%>%  
 mutate(yearRetired=substr(finalGame,1,4))  
  
#Appearances  
app<-Appearances%>%  
 group\_by(playerID)%>%  
 summarize(pitch=sum(G\_p))

## `summarise()` ungrouping output (override with `.groups` argument)

dfBat<-left\_join(df3,app,by='playerID')%>%  
 ungroup%>%  
 filter(pitch<50)  
  
dfBat$eligible<-as.numeric(ifelse(is.na(dfBat$inducted),0,1))  
dfBat$inducted<-as.factor(ifelse(dfBat$inducted=='Y',1,0))  
dfBat$inducted[is.na(dfBat$inducted)]<-0  
dfBat<-subset(dfBat, !is.nan(dfBat$avg))  
  
dfBat<- select(dfBat, eligible, inducted, eligible, hits, X2B, X3B, AB, SB, HR, BB, K, Games)

## Step 2: Split data into training and testing

## 75% of the sample size  
smp\_size <- floor(0.75 \* nrow(dfBat))  
  
## set the seed to make the partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(dfBat)), size = smp\_size)  
  
train <- dfBat[train\_ind, ]  
test <- dfBat[-train\_ind, ]  
  
train.control <- trainControl(method = "cv", number = 10)

## Step 3: Fit our inital logistic regression using 10 fold cross validation

tenfld <- caret::train(inducted~.,data=train, method = "glm", trControl = train.control,na.action=na.exclude)  
# Summarize the results  
print(tenfld)

## Generalized Linear Model   
##   
## 10152 samples  
## 10 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 9136, 9137, 9136, 9137, 9136, 9137, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.9927113 0.6435452

summary(tenfld)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.604 0.000 0.000 0.000 3.171   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -2.421e+01 7.756e+02 -0.031 0.97510   
## eligible 1.921e+01 7.756e+02 0.025 0.98024   
## hits 3.635e-03 1.377e-03 2.640 0.00829 \*\*   
## X2B -3.258e-03 2.803e-03 -1.162 0.24504   
## X3B 1.569e-02 4.340e-03 3.616 0.00030 \*\*\*  
## AB -2.000e-04 5.936e-04 -0.337 0.73612   
## SB 4.484e-04 1.024e-03 0.438 0.66154   
## HR 9.283e-03 2.053e-03 4.521 6.15e-06 \*\*\*  
## BB 1.436e-03 6.150e-04 2.336 0.01951 \*   
## K -2.076e-03 6.560e-04 -3.165 0.00155 \*\*   
## Games -1.777e-03 1.600e-03 -1.111 0.26676   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1338.96 on 10151 degrees of freedom  
## Residual deviance: 370.72 on 10141 degrees of freedom  
## AIC: 392.72  
##   
## Number of Fisher Scoring iterations: 22

## Step 4: Check results on test data

test$pred\_log<-predict(tenfld, newdata = test)  
  
table(test$inducted,test$pred\_log)

##   
## 0 1  
## 0 3348 5  
## 1 14 18

Great model accuracy, but not false negative rate is too high. Let’s make another model of only eligible players.

## Step 5: Adjust model

dfBat2<-subset(dfBat, dfBat$eligible==1)  
  
#TT split  
  
## 75% of the sample size  
smp\_size <- floor(0.75 \* nrow(dfBat2))  
  
## set the seed to make the partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(dfBat2)), size = smp\_size)  
  
train <- dfBat2[train\_ind, ]  
test <- dfBat2[-train\_ind, ]  
  
train.control <- trainControl(method = "cv", number = 10)  
  
tenfld <- caret::train(inducted~ . -eligible,data=train, method = "glm", trControl = train.control,na.action=na.exclude)  
# Summarize the results  
print(tenfld)

## Generalized Linear Model   
##   
## 608 samples  
## 10 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 548, 547, 547, 548, 547, 547, ...   
## Resampling results:  
##   
## Accuracy Kappa   
## 0.8833272 0.5631029

summary(tenfld)

##   
## Call:  
## NULL  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.9600 -0.4546 -0.2435 -0.1086 2.6620   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -5.5523074 0.6587824 -8.428 < 2e-16 \*\*\*  
## hits 0.0031719 0.0014153 2.241 0.025019 \*   
## X2B -0.0039504 0.0027568 -1.433 0.151865   
## X3B 0.0165220 0.0045719 3.614 0.000302 \*\*\*  
## AB 0.0002553 0.0006285 0.406 0.684577   
## SB -0.0002082 0.0010143 -0.205 0.837380   
## HR 0.0075151 0.0020677 3.635 0.000278 \*\*\*  
## BB 0.0015588 0.0006622 2.354 0.018584 \*   
## K -0.0019013 0.0006706 -2.835 0.004583 \*\*   
## Games -0.0025056 0.0017377 -1.442 0.149328   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 583.87 on 607 degrees of freedom  
## Residual deviance: 344.41 on 598 degrees of freedom  
## AIC: 364.41  
##   
## Number of Fisher Scoring iterations: 6

test$pred\_log<-predict(tenfld, newdata = test)  
#test$classify<-ifelse(test$pred\_log>=0.5,1,0)  
table(test$inducted,test$pred\_log)

##   
## 0 1  
## 0 156 4  
## 1 20 23

Hm, our false negative rate is still really high. Let’s try playing around with the type of classifier.

pred<-predict(tenfld, newdata = test, type='prob')  
test$pred\_yes<-pred$`1`  
test$classify<-ifelse(test$pred\_yes>=0.30,1,0)  
  
table(test$inducted,test$classify)

##   
## 0 1  
## 0 144 16  
## 1 12 31

There, that’s much better of a true negative rate.