Baseball Data Analysis

Asaf Drizlikh, Xin Su, Steve Broll

December 14, 2017

## Introduction

The MLB Baseball Hall of Fame is a continuously-developing, prestigious collection of highly-skilled baseball players from the 20th century and early 21st century. However, due to the fact that baseball players have many varying skill sets, being placed in the Hall of Fame is largely a subjective achievement, as there are no explicitly definitive qualifications. However, using linear discriminant analysis and , a statistical analysis method concerned with retaining qualitative results using quantitative data, we can create predictive models for determining a baseball player’s Hall of Fame-related status.

dta = read.csv('https://raw.githubusercontent.com/asafgibor/404finalproject/master/FinalReport/hof\_all.csv')

summary(dta)

## X ID Name HOF   
## Min. : 2 aaronha01: 1 Alomar, Sandy : 2 N:967   
## 1st Qu.: 3192 adairje01: 1 Cruz, Jose : 2 Y: 49   
## Median : 6134 adcocjo01: 1 Roberts, Dave : 2   
## Mean : 6213 ageeto01 : 1 Schofield, Dick: 2   
## 3rd Qu.: 9288 aguaylu01: 1 Aaron, Hank : 1   
## Max. :12440 aldremi01: 1 Adair, Jerry : 1   
## (Other) :1010 (Other) :1006   
## HOF\_n HOF\_mgr FY LY SP   
## Min. : 0.0000 N:1014 Min. :1940 Min. :1961 Min. :10.00   
## 1st Qu.: 0.0000 Y: 2 1st Qu.:1964 1st Qu.:1977 1st Qu.:11.00   
## Median : 0.0000 Median :1974 Median :1988 Median :13.00   
## Mean : 0.9114 Mean :1974 Mean :1987 Mean :13.54   
## 3rd Qu.: 1.0000 3rd Qu.:1986 3rd Qu.:1999 3rd Qu.:15.00   
## Max. :16.0000 Max. :1999 Max. :2008 Max. :25.00   
##   
## n\_33 POS ASG G AB   
## Min. :10.00 Min. :0.0000 N:757 Min. : 140.0 Min. : 252   
## 1st Qu.:11.00 1st Qu.:0.0000 Y:259 1st Qu.: 937.5 1st Qu.: 2624   
## Median :13.00 Median :0.0000 Median :1287.5 Median : 4060   
## Mean :13.54 Mean :0.4163 Mean :1344.2 Mean : 4359   
## 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:1688.2 3rd Qu.: 5661   
## Max. :25.00 Max. :1.0000 Max. :3308.0 Max. :12364   
##   
## R H DB TP   
## Min. : 2.00 Min. : 4.80 Min. : 6.0 Min. : 0.00   
## 1st Qu.: 30.57 1st Qu.: 65.67 1st Qu.:109.0 1st Qu.: 12.00   
## Median : 51.65 Median :106.25 Median :180.0 Median : 25.00   
## Mean : 57.90 Mean :116.67 Mean :200.7 Mean : 30.81   
## 3rd Qu.: 76.55 3rd Qu.:154.38 3rd Qu.:267.0 3rd Qu.: 42.00   
## Max. :229.50 Max. :377.10 Max. :725.0 Max. :177.00   
##   
## HR RBI SB CS   
## Min. : 0.0 Min. : 2.10 Min. : 0.0 Min. : 0.00   
## 1st Qu.: 36.0 1st Qu.: 27.38 1st Qu.: 15.0 1st Qu.: 15.00   
## Median : 84.0 Median : 45.40 Median : 41.0 Median : 31.00   
## Mean :116.4 Mean : 54.60 Mean : 84.6 Mean : 42.02   
## 3rd Qu.:160.0 3rd Qu.: 72.03 3rd Qu.: 103.2 3rd Qu.: 58.00   
## Max. :755.0 Max. :229.70 Max. :1406.0 Max. :335.00   
##   
## BB SO IBB HBP   
## Min. : 17.0 Min. : 35.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 223.8 1st Qu.: 364.8 1st Qu.: 16.00 1st Qu.: 12.00   
## Median : 365.0 Median : 571.0 Median : 30.00 Median : 22.00   
## Mean : 438.1 Mean : 646.2 Mean : 42.20 Mean : 28.77   
## 3rd Qu.: 574.2 3rd Qu.: 845.2 3rd Qu.: 54.25 3rd Qu.: 37.25   
## Max. :2190.0 Max. :2597.0 Max. :293.00 Max. :285.00   
##   
## SH SF GIDP DBTP   
## Min. : 0.00 Min. : 1.00 Min. : 9.00 Min. : 0.70   
## 1st Qu.: 13.00 1st Qu.: 19.00 1st Qu.: 56.00 1st Qu.:12.70   
## Median : 26.00 Median : 32.00 Median : 87.50 Median :21.10   
## Mean : 34.81 Mean : 36.79 Mean : 99.52 Mean :23.15   
## 3rd Qu.: 47.00 3rd Qu.: 49.00 3rd Qu.:131.00 3rd Qu.:30.55   
## Max. :214.00 Max. :128.00 Max. :350.00 Max. :90.20   
##   
## BBSB AVG SLG OBP   
## Min. : 1.90 Min. :161.0 Min. :222.0 Min. :203.0   
## 1st Qu.: 25.60 1st Qu.:249.0 1st Qu.:351.0 1st Qu.:310.0   
## Median : 43.90 Median :261.0 Median :393.0 Median :328.0   
## Mean : 52.27 Mean :261.4 Mean :393.3 Mean :328.3   
## 3rd Qu.: 69.03 3rd Qu.:274.0 3rd Qu.:431.2 3rd Qu.:347.0   
## Max. :359.60 Max. :338.0 Max. :565.0 Max. :421.0   
##   
## AW\_P AW\_C AWD WAR   
## Min. : 0.00 Min. :0.00000 Min. :0.000 Min. :-0.9755   
## 1st Qu.: 0.00 1st Qu.:0.00000 1st Qu.:0.000 1st Qu.: 0.6171   
## Median : 0.00 Median :0.00000 Median :0.000 Median : 1.4069   
## Mean : 1.14 Mean :0.08071 Mean :0.315 Mean : 1.5994   
## 3rd Qu.: 1.00 3rd Qu.:0.00000 3rd Qu.:1.000 3rd Qu.: 2.4016   
## Max. :20.00 Max. :3.00000 Max. :1.000 Max. : 7.5602   
## NA's :1

## Data Preparation

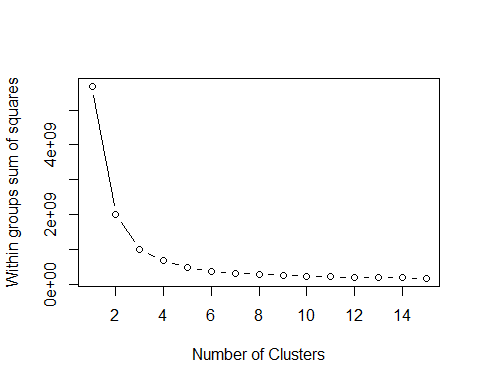
dta\_num <- dta[,c(5,9:11,13:38)]  
dta\_num <- na.omit(dta\_num) # listwise deletion of missing  
dta <- na.omit(dta)  
dim(dta\_num)

## [1] 1015 30

## K-means

K-means clustering is the most popular partitioning method. It requires you to specify the number of clusters to extract. A plot of the within groups sum of squares by number of clusters extracted can help determine the appropriate number of clusters. An important feature in the plot is the bend, similar to a scree test in factor analysis.

# Determine number of clusters  
wss <- (nrow(dta\_num)-1)\*sum(apply(dta\_num,2,var))  
for (i in 2:15) wss[i] <- sum(kmeans(dta\_num, centers=i)$withinss)  
plot(1:15, wss, type="b", xlab="Number of Clusters",ylab="Within groups sum of squares")



# K-Means Cluster Analysis  
fit <- kmeans(dta\_num, 5) # 5 cluster solution  
# get cluster means   
aggregate(dta\_num,by=list(fit$cluster),FUN=mean)

## Group.1 HOF\_n SP n\_33 POS G AB  
## 1 1 0.09345794 11.27103 11.27103 0.5841121 690.9533 1673.864  
## 2 2 2.35294118 16.16471 16.16471 0.2764706 1914.8176 6720.882  
## 3 3 3.85897436 19.53846 19.53846 0.2692308 2518.4359 9202.282  
## 4 4 0.57446809 13.47518 13.47518 0.3581560 1435.1064 4765.344  
## 5 5 0.15867159 12.01845 12.01845 0.4723247 1068.9852 3177.295  
## R H DB TP HR RBI SB  
## 1 19.14346 41.11729 70.84579 9.364486 32.04673 17.90187 21.97664  
## 2 92.90882 183.50529 317.38235 49.323529 204.48235 90.14941 137.20000  
## 3 137.86923 261.01667 452.26923 74.294872 307.34615 129.20513 230.26923  
## 4 62.87163 127.50000 217.93617 34.301418 115.35461 57.26986 94.80851  
## 5 38.30369 81.45092 139.50923 19.878229 74.04428 36.96531 48.68635  
## CS BB SO IBB HBP SH SF  
## 1 13.41589 155.9813 277.5654 13.97664 11.61682 18.04673 12.93458  
## 2 68.07647 685.2353 996.4824 67.41176 43.07647 46.18235 58.50588  
## 3 88.87179 1018.0641 1223.8974 117.92308 57.11538 49.82051 84.07692  
## 4 48.22695 471.2872 696.2340 40.15957 32.96454 40.64894 38.99645  
## 5 28.42804 304.6716 500.0996 29.13653 20.89299 30.37269 26.14022  
## GIDP DBTP BBSB AVG SLG OBP AW\_P  
## 1 39.64953 8.021028 17.79579 243.6776 353.1682 309.1869 0.02336449  
## 2 153.38235 36.670588 82.24353 272.6176 425.9706 340.4353 2.50000000  
## 3 206.26923 52.656410 124.83333 283.1667 448.1410 354.5513 6.20512821  
## 4 106.56383 25.223759 56.60957 267.2234 400.0957 334.3156 0.64539007  
## 5 74.68635 15.938745 35.33579 255.8708 381.7712 322.0627 0.22878229  
## AW\_C AWD WAR  
## 1 0.004672897 0.02803738 0.4078581  
## 2 0.123529412 0.65294118 2.5326984  
## 3 0.615384615 0.89743590 3.6234794  
## 4 0.035460993 0.33333333 1.7530092  
## 5 0.007380074 0.14391144 1.2125420

# append cluster assignment  
dta\_num <- data.frame(dta\_num, fit$cluster)

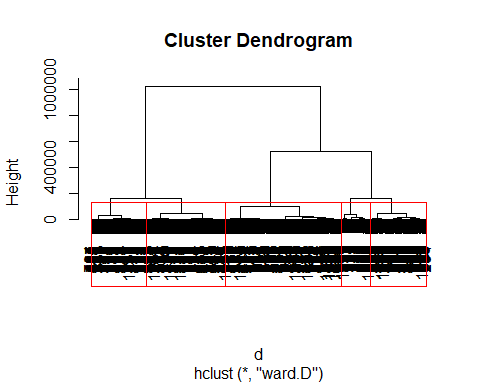
## Hierarchical Agglomerative

There are a wide range of hierarchical clustering approaches.

# Ward Hierarchical Clustering  
d <- dist(dta\_num, method = "euclidean") # distance matrix  
fit <- hclust(d, method="ward")

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

plot(fit) # display dendogram  
groups <- cutree(fit, k=5) # cut tree into 5 clusters  
# draw dendogram with red borders around the 5 clusters   
rect.hclust(fit, k=5, border="red")



The pvclust( ) function in the pvclust package provides p-values for hierarchical clustering based on multiscale bootstrap resampling. Clusters that are highly supported by the data will have large p values. Be aware that pvclust clusters columns, not rows; data must also be transposed first.

# Ward Hierarchical Clustering with Bootstrapped p values  
library(pvclust)

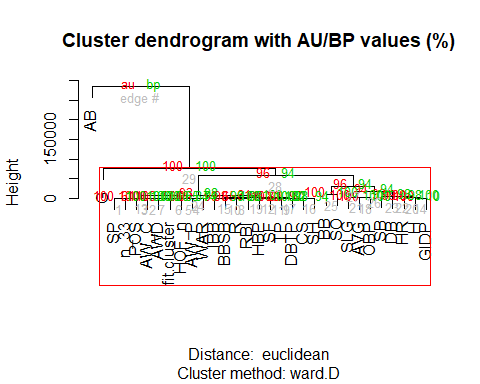
## Warning: package 'pvclust' was built under R version 3.4.3

fit <- pvclust(dta\_num, method.hclust="ward",  
 method.dist="euclidean")

## The "ward" method has been renamed to "ward.D"; note new "ward.D2"

## Bootstrap (r = 0.5)... Done.  
## Bootstrap (r = 0.6)... Done.  
## Bootstrap (r = 0.7)... Done.  
## Bootstrap (r = 0.8)... Done.  
## Bootstrap (r = 0.9)... Done.  
## Bootstrap (r = 1.0)... Done.  
## Bootstrap (r = 1.1)... Done.  
## Bootstrap (r = 1.2)... Done.  
## Bootstrap (r = 1.3)... Done.  
## Bootstrap (r = 1.4)... Done.

plot(fit) # dendogram with p values  
# add rectangles around groups highly supported by the data  
pvrect(fit, alpha=.95)



Exploratory Analysis

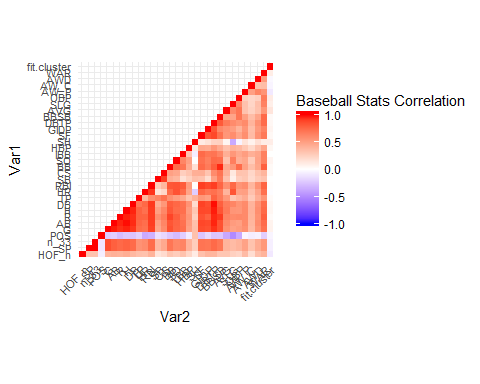
Heatmap

Studying the correlation between each variable in the data set is valuable. If two variables are highly correlated, using them in the same predictive model will generate misleading results.

library(ggplot2)  
library(reshape2)##for melt  
cor\_mat <- cor(dta\_num)  
###half of matrix is redundant, use lower.tri function to set redundant values to NA  
cor\_mat[lower.tri(cor\_mat)] <- NA  
###must melt matrix for ggplot  
melt\_mat <- melt(cor\_mat, na.rm = TRUE)  
##plot  
hmap <- ggplot(data = melt\_mat, aes(Var2, Var1, fill = value)) +  
 geom\_raster(color = "white") + ##faster than geom\_tile()  
 scale\_fill\_gradient2(low = "blue", high = "red", mid = "white", midpoint = 0, limit = c(-1,1), space = "Lab", name = "Baseball Stats Correlation") + ## create gradient, darker approaching +- 1, provides legend   
 theme\_minimal()+ ##white background  
 theme(axis.text.x = element\_text(angle = 45, vjust = 1, size = 8, hjust = 1))+ ##rotate and shrink x axis labels  
 theme(axis.text.y = element\_text(size = 8))+  
 coord\_fixed()

## Warning: Ignoring unknown parameters: colour

plot(hmap)



Principal Components Analysis

This analysis tool is used to reduce the dimensions of the data and determine which variables are the most valueable for predictions.

pca = princomp(dta[,c(5,9,11,13:34,38)]) #Select numeric columns  
summary(pca)

## Importance of components:  
## Comp.1 Comp.2 Comp.3 Comp.4  
## Standard deviation 2339.7531774 252.58465759 1.650494e+02 1.161489e+02  
## Proportion of Variance 0.9779186 0.01139663 4.866206e-03 2.409864e-03  
## Cumulative Proportion 0.9779186 0.98931526 9.941815e-01 9.965913e-01  
## Comp.5 Comp.6 Comp.7 Comp.8  
## Standard deviation 1.047001e+02 5.804014e+01 3.788169e+01 28.673021687  
## Proportion of Variance 1.958199e-03 6.017548e-04 2.563429e-04 0.000146862  
## Cumulative Proportion 9.985495e-01 9.991513e-01 9.994076e-01 0.999554487  
## Comp.9 Comp.10 Comp.11 Comp.12  
## Standard deviation 2.527310e+01 2.354931e+01 2.006323e+01 1.624688e+01  
## Proportion of Variance 1.140984e-04 9.906473e-05 7.190588e-05 4.715224e-05  
## Cumulative Proportion 9.996686e-01 9.997677e-01 9.998396e-01 9.998867e-01  
## Comp.13 Comp.14 Comp.15 Comp.16  
## Standard deviation 1.326102e+01 1.258564e+01 1.141157e+01 9.058676e+00  
## Proportion of Variance 3.141352e-05 2.829522e-05 2.326236e-05 1.465859e-05  
## Cumulative Proportion 9.999181e-01 9.999464e-01 9.999697e-01 9.999843e-01  
## Comp.17 Comp.18 Comp.19 Comp.20  
## Standard deviation 6.601362e+00 4.094912e+00 3.742533e+00 2.442204e+00  
## Proportion of Variance 7.784483e-06 2.995380e-06 2.502039e-06 1.065434e-06  
## Cumulative Proportion 9.999921e-01 9.999951e-01 9.999976e-01 9.999987e-01  
## Comp.21 Comp.22 Comp.23 Comp.24  
## Standard deviation 2.221854e+00 1.375781e+00 6.310633e-01 3.605425e-01  
## Proportion of Variance 8.818479e-07 3.381124e-07 7.113911e-08 2.322072e-08  
## Cumulative Proportion 9.999996e-01 9.999999e-01 1.000000e+00 1.000000e+00  
## Comp.25 Comp.26  
## Standard deviation 4.662011e-07 0  
## Proportion of Variance 3.882480e-20 0  
## Cumulative Proportion 1.000000e+00 1

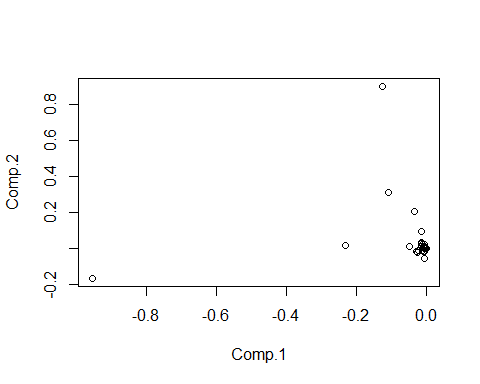
#The first PC explains an astonishing 98 % of the variance  
  
#Coefficients for each variable in first PC  
loadings(pca)[,1]

## HOF\_n SP POS G AB   
## -0.0004532118 -0.0010246307 0.0000450809 -0.2316730380 -0.9554730383   
## R H DB TP HR   
## -0.0148133089 -0.0276933280 -0.0481789689 -0.0080314192 -0.0340238502   
## RBI SB CS BB SO   
## -0.0139920624 -0.0252377229 -0.0099435165 -0.1068048056 -0.1240069402   
## IBB HBP SH SF GIDP   
## -0.0120633207 -0.0058655329 -0.0043073900 -0.0088144860 -0.0213276382   
## DBTP BBSB AVG SLG OBP   
## -0.0056210388 -0.0132042528 -0.0051911489 -0.0123463632 -0.0059000219   
## WAR   
## -0.0003994239

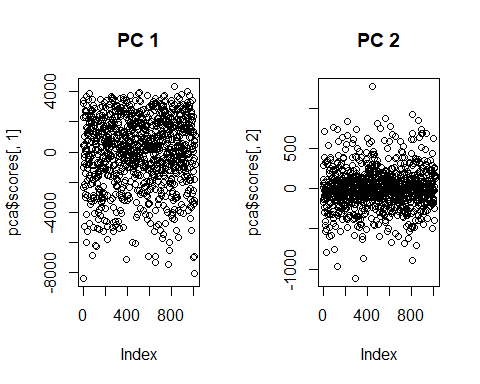
#In the first PC, AB's are weighted much more heavily than any others, almost 8 times as much as the next highest  
  
#Coefficients for each variable in second PC  
loadings(pca)[,2]

## HOF\_n SP POS G AB   
## -0.0009130850 0.0002695459 -0.0004557172 0.0159198495 -0.1652367494   
## R H DB TP HR   
## 0.0125661643 -0.0143465483 0.0125820592 -0.0153437455 0.2054184260   
## RBI SB CS BB SO   
## 0.0324450316 -0.0201921338 -0.0106659427 0.3140571132 0.9024472657   
## IBB HBP SH SF GIDP   
## 0.0249123484 0.0146957823 -0.0543215878 0.0060948271 -0.0088520557   
## DBTP BBSB AVG SLG OBP   
## -0.0002761686 0.0293864979 -0.0164760095 0.0969028879 0.0210229389   
## WAR   
## 0.0008895106

#In the second PC, SO is by far the highest, more than 3 times as high as the next main group of AB, HR, and BB   
  
#First two PC's  
plot(loadings(pca))

 Most coefficients are very close to 0, interestingly, there is no extreme outlier in the top left, meaning that extreme outliers for one PC are not extreme outliers for the other.

#Side by side plot of player's PC scores  
par(mfrow = c(1,2))  
plot(pca$scores[,1], main = "PC 1")  
plot(pca$scores[,2], main = 'PC 2')

 The first PC’s scores are skewed to the negative end, while the second PC’s scores are evenly spread around 0 There is a lot more information from the first PC, and it probably would be more useful in categorizing HOF type players

Main Statical Analysis

Linear Discriminant Analysis

library(MASS)  
  
## Extract a few offensive statistics (numerical variables).  
dta\_st <- na.omit(dta[, c("HOF", "OBP", "SF","SLG","SB","SH")])  
  
## Variable declarations  
thresh\_seq = seq(from = .05, to = .95, by = .05) # list of threshold values  
n <- nrow(dta) # stores number of rows in the data set  
  
lda\_out <- lda(HOF ~., data=dta\_st, CV = TRUE)  
sens\_lda <- spec\_lda <- bacc\_lda <-NULL  
for(i in 1:length(thresh\_seq)) {  
 class\_lda <- lda\_out$posterior[,2] > thresh\_seq[i]  
 # filters results by players who are actually in the HOF (Y), then checks if TRUE was predicted. The average of the correct YES predictions defines sensitivity  
 sens\_lda[i] <- mean(class\_lda[dta\_st$HOF == 'Y'] == TRUE)   
 # filters results by players who are actually NOT in the HOF (N), then checks if FALSE was predicted. The average of the correct YES predictions defines sensitivity  
 spec\_lda[i] <- mean(class\_lda[dta\_st$HOF == 'N'] == FALSE)  
 bacc\_lda[i] <- (sens\_lda[i] + 3\*spec\_lda[i]) / 4 # balance accuracy calculation is weighted  
 results1 <- as.data.frame(cbind(sens\_lda, spec\_lda, bacc\_lda))  
}  
  
spec2 <- 1-spec\_lda  
roc\_dat1<-data.frame(x=spec2,y=sens\_lda)  
  
roc1 <- ggplot(roc\_dat1) + geom\_point(aes(roc\_dat1$x,roc\_dat1$y),size=2) + labs(title= "LDA ROC Curve",   
 x = "False Positive Rate (1-Specificity)",   
 y = "True Positive Rate (Sensitivity)")  
results1

## sens\_lda spec\_lda bacc\_lda  
## 1 0.8775510 0.8726708 0.8738909  
## 2 0.7959184 0.9161491 0.8860914  
## 3 0.6938776 0.9378882 0.8768855  
## 4 0.6530612 0.9461698 0.8728926  
## 5 0.6326531 0.9554865 0.8747782  
## 6 0.5918367 0.9596273 0.8676797  
## 7 0.5306122 0.9658385 0.8570319  
## 8 0.5102041 0.9699793 0.8550355  
## 9 0.4285714 0.9792961 0.8416149  
## 10 0.4081633 0.9834369 0.8396185  
## 11 0.4081633 0.9855072 0.8411713  
## 12 0.3469388 0.9875776 0.8274179  
## 13 0.3265306 0.9906832 0.8246451  
## 14 0.2653061 0.9927536 0.8108917  
## 15 0.2653061 0.9927536 0.8108917  
## 16 0.2244898 0.9958592 0.8030169  
## 17 0.2244898 0.9989648 0.8053461  
## 18 0.1428571 0.9989648 0.7849379  
## 19 0.1020408 0.9989648 0.7747338

Generalized Linear Model

## Extract a few offensive statistics (numerical variables).  
dta\_st <- dta[, c("HOF", "OBP", "SF","SLG","SB","SH")]  
  
## Variable declarations  
fit\_pred <- matrix(rep(0,n), nrow=n, ncol=1)  
  
for(i in 1:n) {  
 fit <- glm(HOF ~., data = dta\_st[-i,], family = binomial()) #uses all values but one as training data  
 fit\_pred[i] <- predict(fit, newdata = dta\_st[i,], type = 'response') # uses remaining value to predict  
}  
  
sens\_glm <- spec\_glm <- bacc\_glm <-NULL  
for(i in 1:length(thresh\_seq)) {  
 class\_glm <- fit\_pred > thresh\_seq[i]  
 # filters results by players who are actually in the HOF (Y), then checks if TRUE was predicted. The average of the correct YES predictions defines sensitivity  
 sens\_glm[i] <- mean(class\_glm[dta\_st$HOF == 'Y'] == TRUE)  
 # filters results by players who are actually NOT in the HOF (N), then checks if FALSE was predicted. The average of the correct YES predictions defines sensitivity  
 spec\_glm[i] <- mean(class\_glm[dta\_st$HOF == 'N'] == FALSE)  
 bacc\_glm[i] <- (sens\_glm[i] + 3\*spec\_glm[i]) / 4   
 results2 <- as.data.frame(cbind(sens\_glm, spec\_glm, bacc\_glm))  
}  
  
spec2 <- 1-spec\_glm  
roc\_dat2<-data.frame(x=spec2,y=sens\_glm) # data for roc curve  
  
roc2 <- ggplot(roc\_dat2) + geom\_point(aes(roc\_dat2$x,roc\_dat2$y),size=2) + labs(title= "GLM ROC Curve",   
 x = "False Positive Rate (1-Specificity)",   
 y = "True Positive Rate (Sensitivity)")  
  
results2

## sens\_glm spec\_glm bacc\_glm  
## 1 0.91836735 0.8954451 0.9011757  
## 2 0.89795918 0.9327122 0.9240240  
## 3 0.81632653 0.9523810 0.9183673  
## 4 0.71428571 0.9585921 0.8975155  
## 5 0.67346939 0.9710145 0.8966282  
## 6 0.55102041 0.9741201 0.8683452  
## 7 0.53061224 0.9761905 0.8647959  
## 8 0.48979592 0.9792961 0.8569210  
## 9 0.38775510 0.9824017 0.8337400  
## 10 0.38775510 0.9865424 0.8368456  
## 11 0.38775510 0.9896480 0.8391748  
## 12 0.36734694 0.9917184 0.8356256  
## 13 0.28571429 0.9958592 0.8183230  
## 14 0.26530612 0.9979296 0.8147737  
## 15 0.24489796 0.9979296 0.8096717  
## 16 0.20408163 0.9979296 0.7994676  
## 17 0.14285714 0.9989648 0.7849379  
## 18 0.12244898 1.0000000 0.7806122  
## 19 0.06122449 1.0000000 0.7653061

Sensitivity and Specificity are based the rate of true positives and true negatives, respectively. In a confusion matrix, which compares predictions to actual values, the effectiveness of a model can be easily determined. A perfect model would only result in a confusion matrix that only features data on the diagonal. This would signify that every positive (bottom right) and negative (top left) prediction was correct.

avglda <- mean(sens\_lda)  
avgglm <- mean(sens\_glm)  
  
lda\_out <- lda(HOF ~., data = dta\_st)  
lda\_pred <- predict(lda\_out, newdata = dta\_st)$class   
   
confmat <- table(lda\_pred, dta\_st$HOF)  
confmat

##   
## lda\_pred N Y  
## N 951 27  
## Y 15 22

avglda

## [1] 0.443609

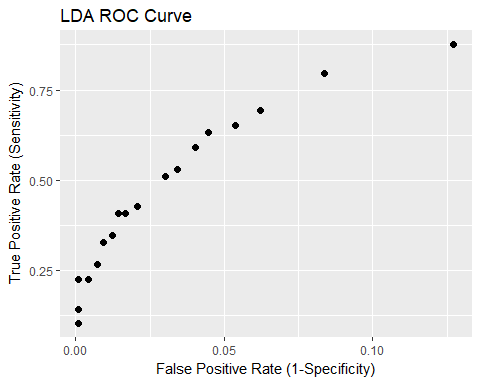
avgglm

## [1] 0.4446831

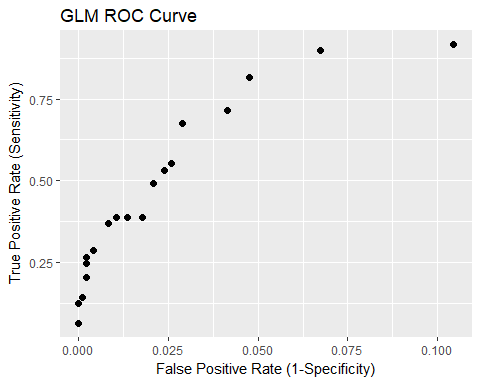
This confusion matrix shows how, for example, 15 players were incorrectly predicted to not be in the hall of fame. The numbers on the diagonal represent correct predictions.

Results and Conclusion

plot(roc1)



plot(roc2)



Comparing Receiver Operator Characteristic (ROC) curves is a fast way to compare models and gain a general understanding of their effectiveness. An ideal ROC curve will maximize sensitivity, while minimizing 1-speciicity. This would manifest in a graph as the curve only being present on the leftmost side for as much of the y-axis as possible.

The glm graph shows that it was able to achieve slightly better rates for true positives and negatives. This is a valuable result that can be quickly observed from an ROC curve. Using this information, optimizations for threshold values and variables can improve sensitivity, specificity, and balanced accuracy. The glm prediction was able to generate an accuracy of over 90%. which the lda prediction wasn’t able to do with the given model.

Specificity was similar between the two methods, which is to be expected given the fact that the vast majority of players are not in the Hall of Fame. Predicting no for every player would give good specificity, but clearly wouldn’t be an effective tool. The average sensitivity was also extremely close between the two models. This is reflected in the ROC curves as well. Paying more attention to correlation between variables in the model could improve the distinction between the two methods in the future.