The Southpaw (dis)advantage

A multivariate analysis of the pitch quality metrics of top-tier baseball players.

David Wouters
R0665814
Applied multivariate statistical analysis

23 Januari 2021

Introduction

Baseball is arguably America's most famous sport, sometimes referred to as "America's pastime". Its turn based style of play lends itself to many different statistics, and in turn has attracted many different statistic enthusiasts.
Among all things that make baseball baseball, the most important aspect is and always will be the batter-pitcher interaction. Every play starts by the pitcher throwing a ball at the strike zone, and the batter trying to bat the ball into play. Describing the nuances of this interaction in its entirety could be a thesis level text, so for the purpose of this assignment it is easily explained as such: the pitcher is trying to do one of two things: either throw the ball past the batter into the strike zone, or throw the ball outside of the strike zone, but make the batter swing at it and miss. In attempting to do so, top tier pitchers have developed a personal arsenal of pitches that all differ in velocity, spin rate and movement (also called break).
An often discussed concept in this pitcher-batter interaction is "the Platoon effect". This states that the batter performs better in matchups where the pitcher is opposite-handed, and pitchers perform better in matchups where the batter is same-handed. This resulted in teams opting to employ left-handed batters, since most pitchers are obviously right-handed. This was then countered by employing more left-handed pitchers, which resulted in left-handed players being completely overrepresented in the baseball community. However the majority of players are still right-handed, so left-handed pitchers still enjoy this Platoon effect far less than right-handed pitchers, so why has the ratio of left-handed pitchers not decreased throughout history? What matters in the end is results, whether pitchers can get batters out. Statistics show that lefties get as many batters out as righties, even though they "suffer" from the Platoon effect more often than not. Is it because they have superior throwing skills, or just because batters are facing something they on average have to deal with less? What is this secret Southpaw advantage?

Objective

In this short analysis, I will first show the existence of the Platoon effect on batting performance versus left/right-handed pitchers. Then I will apply ordination, clustering and classification techniques to pitch quality metrics (that take the batter out of the equation), to see if I can find some underlying structure in the variance of these metrics that represent "handedness" of the pitchers that produced the metrics.

Data

For the batting performance versus left/right-handed pitchers, I used a dataset available on MLB's (=top league of American baseball) website[1]. However this isn't directly available as a csv file, so I used some web scraping and data parsing code I wrote in Python, which you will find in the attachments. Since this dataset didn't contain information about the batter's batting hand, I merged it with a dataset taken from the Lahman R package[2]. The only 2 variables here are batting hand and OBP, which stands for "On Base Percentage. Given that getting on base is a fairly straightforward way of saying an at bat was successful, I chose this variable as a performance statistic for batters.

The pitch quality metric dataset was downloaded from a Statcast powered website[3].It contains the statistics of MLB players active (meaning at least 200 plate appearances) in

2019. The variables are a combination of velocity, spin and break, for 3 types of common pitches: the Fastball, an offspeed pitch and a Breaking ball.

	Fastball	Offspeed	Breaking ball
Velocity	fastball_avg_speed	offspeed_avg_speed	breaking_avg_speed
Spin	fastball_avg_spin	offspeed_avg_spin	breaking_avg_spin
Break (movement)	fastball_avg_break	offspeed_avg_break	breaking_avg_break

Something I will keep in mind during the analysis is that I expect these variables to be highly correlated in both dimensions: The metrics of a particular pitch type will be correlated in some way, and the same metric for the three pitch types will also be correlated, since you'd expect a pitcher with a fast Fastball to also throw his Breaking ball faster than the average pitcher. Not all MLB pitchers have access to each of these pitch types, which creates some missing data. I chose to simply remove those observations, since it concerned only 17 out of 279 observations.

Analysis

The Platoon Effect

The Platoon effect is easily shown by just creating a boxplot of the differences in performance by left/right-handed batters vs left/right-handed pitchers, as seen in **figure 1**.
It clearly shows that performance of left-handed batters is better against right-handed pitchers, and vice versa for right-handed batters.I also created a simple logistic regression model (**Sup figure 1)** to check if the OBP variable can be used to predict handedness of the batter. This showed that for both performances, adding OBP to the intercept was significant.

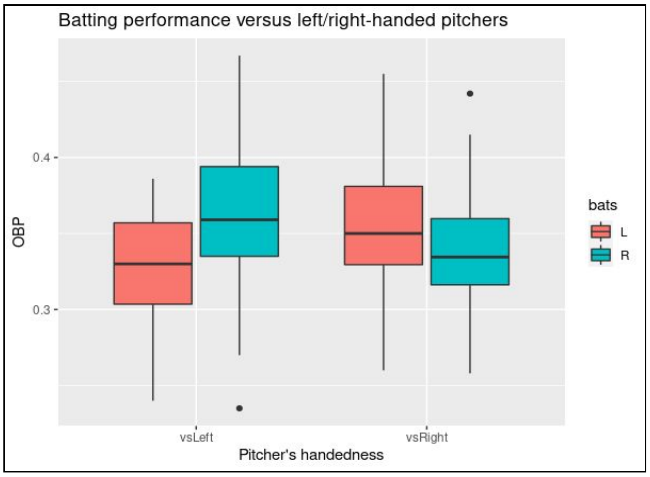


Figure 1: Batting results by "handedness"

Ordination of pitch quality metrics

As mentioned before, I expect the variables to be quite correlated. I investigated this correlation structure with a pairwise scatter plot **(Sup. figure 2)** for all variables, as well as with a biplot **(Figure 3)**, with the code we saw in class instead of the built in biplot() function.
From the pairwise scatter plot, it is clear that there is definitely some positive linear correlation between all speed variables, as well as between the spin and break levels of the

offspeed pitch. There seems to be a negative linear correlation between the speed of a breaking ball and its break.
The biplot shows the expected positive correlation between all 3 speed variables, as well as a positive correlation between most spins and break variables. Only the break of the fastball seems to be more correlated with its speed rather than its break, which makes sense because a fastball shouldn't break all that much.
The biplot also confirms the negative correlation between the break of a breaking ball and its speed. This can be explained by the fact that when a pitcher throws a breaking ball, the goal is to create as much movement as possible on the ball. The slower the ball goes, the more time it has to create movement. It is of note that a lot of information is lost, since most variables are a few pixels from the outer bounds of the circle.

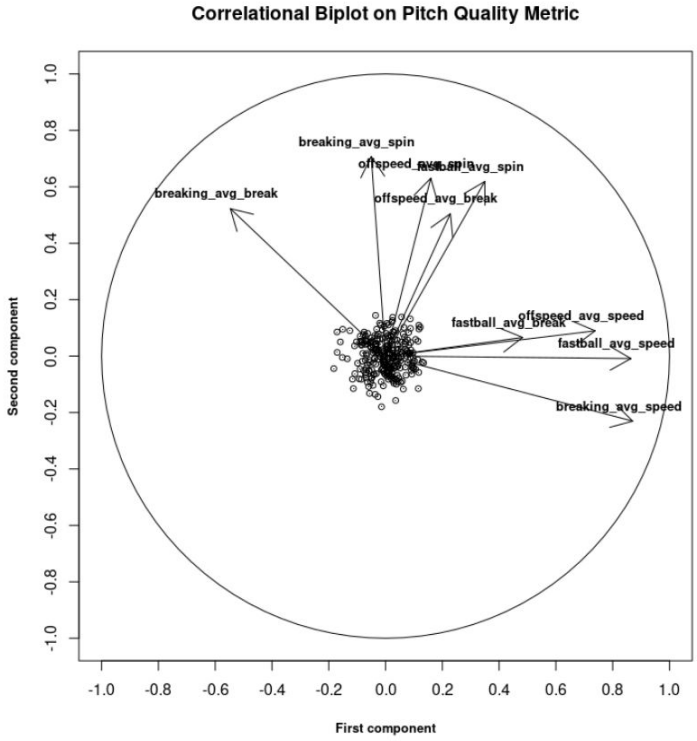
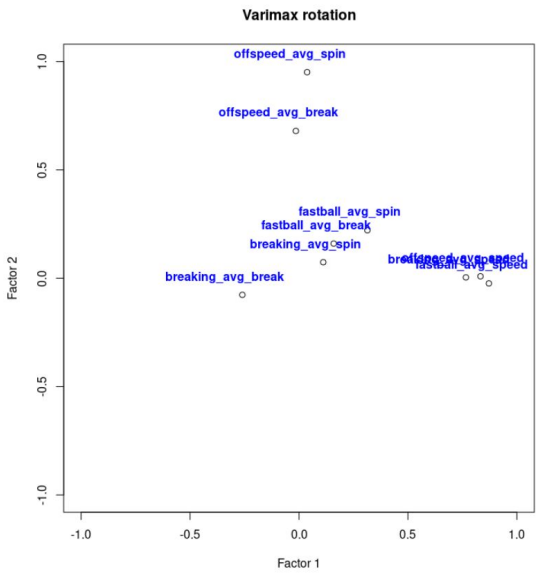


Figure 3: Biplot of correlational structure of the variables.
Since the variables are so correlated with each other, I thought it interesting to perform a factor analysis on the first 4 PC's of the data matrix (determined by a scree plot). **(Figure 5)**



	Factor1	Factor2	Factor3	Factor4
fastball_avg_speed	0.872			0.216
fastball_avg_spin	0.314	0.221	0.214	
fastball_avg_break	0.159	0.160	-0.130	0.963
breaking_avg_speed	0.766		-0.410	
breaking_avg_spin	0.111		0.531	
breaking_avg_break	-0.260		0.956	
offspeed_avg_speed	0.833			
offspeed_avg_spin		0.951		
offspeed_avg_break		0.680		0.264

Figure 5: FA results on the first 4 PC's

An interesting result here is that Factor 2 is highly influenced by offspeed spin and offspeed break. A source[4] that did a similar analysis found that left-handed pitchers outperformed righties in only one thing: the spin and break of their offspeed pitch. This leads me to believe that factor 2 might represent a sort of "handedness" concept.

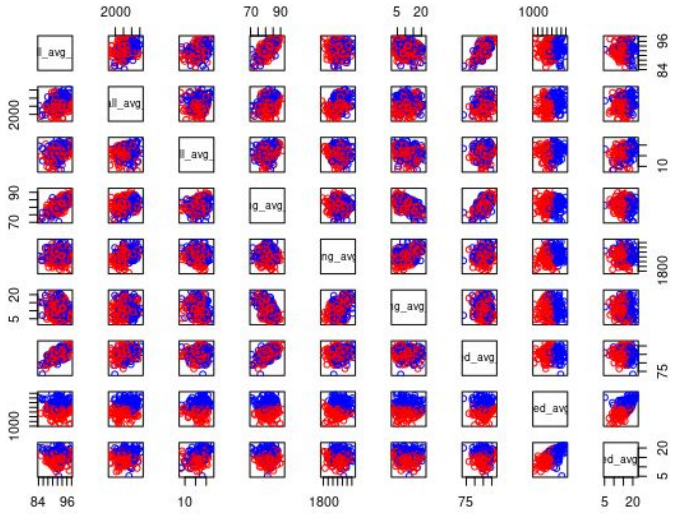
Clustering of pitch quality metrics

The upcoming clustering and classification analyses will all be based around trying to find a grouping structure or a classifier that can successfully split left-handed from right-handed pitchers based on their pitch metrics. The dataset contains 72 and 190 left-and right-handed pitchers respectively. Replotting the paired scatter plot from earlier **(sup. figure 3)** gives me very little hope that I'll find a clean linear clustering between the two on any combination of variables. It is notable however that on almost all variables, the "low" values are almost always lefties, which visualises the concept that lefties indeed have worse pitching quality metrics. For unsupervised clustering I report two methods: Kmeans and Gaussean mixture modelling with Bayesian EM optimization and regularization[5]

When asking Kmeans to find 2 clusters, it seems to find two clusters almost entirely based on an even split in spin and break values of the offspeed pitch, as seen in the group coloured pairs plot **(figure 7)**. This sounds promising at first given that I expect some difference in the spin of the offspeed pitch, however when comparing the clustering assignment to the pitch_hand variable it is clear that his lefty/righty split is not at all the grouping structure Kmeans is picking up, and, no amount of iterations or random starts was able to fix that. It's most likely finding an easy split between pitchers that have a "working" offspeed pitch and the ones that don't.

The gaussian mixture model found that 2 clusters was optimal, and even the split between the two seemed to have the right numbers (80-182), but comparing them with the actual

grouping structure once again revealed that the lefty/righty split is not the grouping structure that is found.



	Cluster 1	cluster2
L	25	47
R	55	135

Figure 7: A: Kmean clustering results, coloured by cluster **B:** Result table comparison of known grouping structure vs clusters created by bayesian clustering.

Classification of pitch quality metrics

The idea here was the following: I suspect that lefties on average have worse pitch statistics. Can I build a classifier that is smart enough to pick up on the subtle and not so subtle differences and classify pitchers' pitch hand solely based on pitch quality metrics? To this end I tried several other non-linear "vanilla" methods, and almost all of them only got half of the lefties correctly classified (**sup. Figure 4)**, all while classifying the righties quite well. This leads me to believe that the difference in handedness in pitch statistics is not quite obvious enough to train a simple classifier to find the difference, and that most classifiers get distracted by the correct classification of righties, which is more easily accomplished since there are more of them. Classification trees showed the most promise **(sup. Figure 5)**, although I believe this might just be a case of overfitting, since using the fitted tree on the pitcher data of 2020 dropped the correct classification of lefties significantly. Interestingly enough: the "negative" branches of the classification tree always points to a left-classification, indicating that indeed their metrics are worse than their right-handed counterparts. The issues I had with classifiers led me to believe that improving a classifier tree with an ensemble method like boosting could increase performance. My attempt at this using Generalized boosted models (gbm package) gave me some nice insight into how the variables work together and how they prioritize a class (**sup. Figure 6)**, but the overall results on the test data only showed mild improvement: 42 out of 53 righties were correctly classified and 14 out of 25 lefties.

Conclusion

My main goal in this project was to see if I can validate the findings of this article[4] that left-handed pitchers have worse pitches, by finding an underlying grouping structure based on pitch quality metrics, either supervised or unsupervised. Unsupervised clustering failed to

find a grouping structure that could realistically be related to “handedness”, and supervised classifiers had a decent overall success classification rate, but largely due to correctly classifying righties, but not lefties. This leads me to believe that the difference in pitching metrics between left-handed and right-handed pitchers is not obvious enough to be picked up by “simple” clustering/classification methods. By interpreting the results I did find a lot of indications that Lefties do indeed overall have worse pitch quality metrics, indicating that their survival in the roster depends on something else than the platoon effect or superior pitching capabilities.

References:

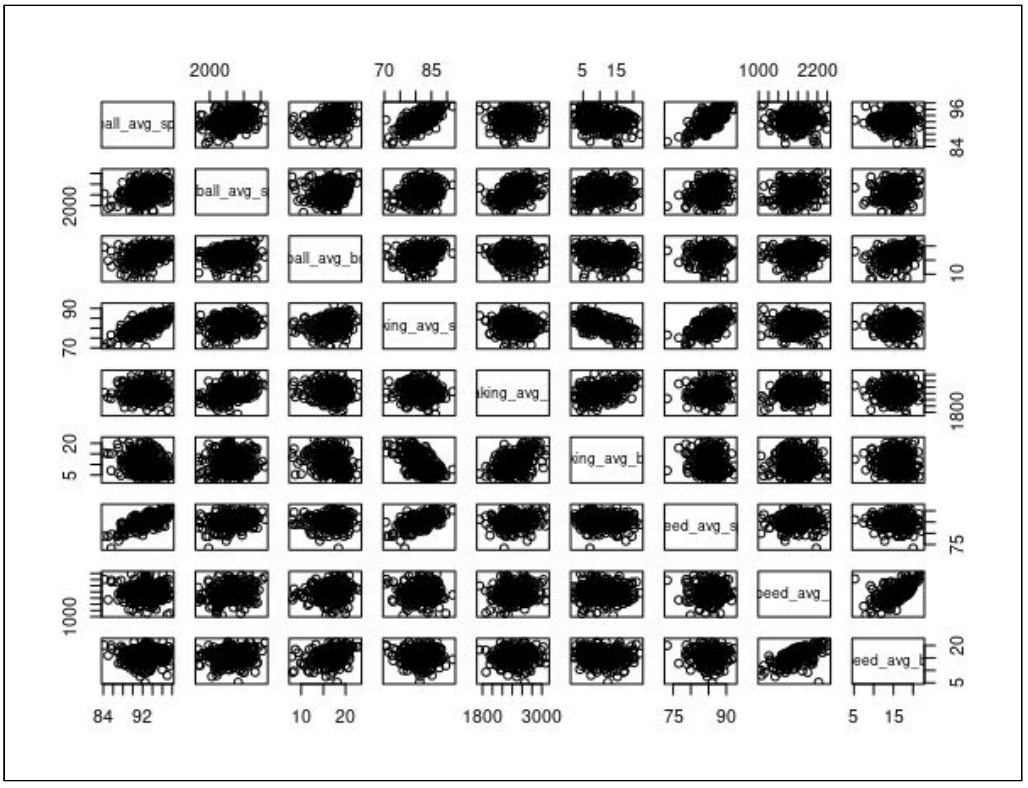
https://blogs.fangraphs.com/the-southpaw-advantage/

1. MLB Stats | Baseball Stats. *MLB.com* https://www.mlb.com/stats.
2. Friendly, M. *et al. Lahman: Sean ‘Lahman’ Baseball Database*. (2021).
3. Statcast Custom Leaderboards. *baseballsavant.com* https://baseballsavant.mlb.com/leaderboard/custom?year=2019&type=pitcher&filter=&sort=4&sortDir=asc&min=q&selections=xba,xslg,xwoba,xobp,xiso,exit_velocity_avg,launch_angle_avg,barrel_batted_rate,&chart=false&x=xba&y=xba&r=no&chartType=beeswarm.
4. The Southpaw Advantage. *FanGraphs Baseball* https://blogs.fangraphs.com/the-southpaw-advantage/.
5. Scrucca, L., Fop, M., Murphy, T., Brendan & Raftery, A., E. mclust 5: Clustering, Classification and Density Estimation Using Gaussian Finite Mixture Models. *R J.* **8**, 289 (2016).

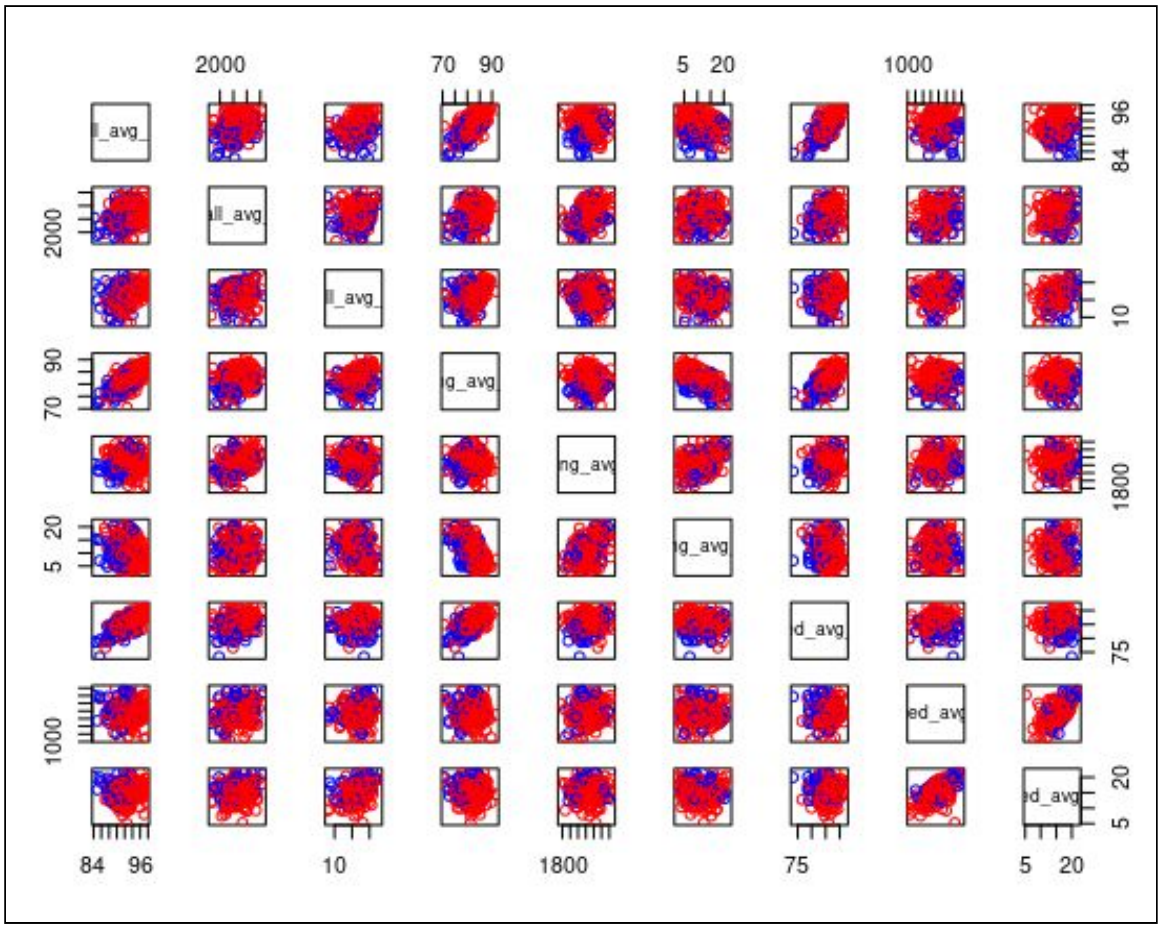
Supplementary images

Coefficients: Estimate Std. Error z value Pr(>|z|) (Intercept) 5.430 1.971 2.756 0.00586 ** OBP -14.172 5.628 -2.518 0.01180 *	Coefficients: Estimate Std. Error z value Pr(>|z|) (Intercept) -6.686 1.955 -3.420 0.000627 *** OBP 20.942 5.716 3.664 0.000249 ***
OR 2.5 % 97.5 % (Intercept) 1.248206e-03 2.016435e-05 4.572518e-02 OBP 1.243941e+09 3.516195e+04 2.255852e+14	OR 2.5 % 97.5 % (Intercept) 2.282626e+02 5.629456e+00 1.362383e+04 OBP 6.998685e-07 6.154785e-12 2.856534e-02

Sup. Figure 1: Output of logistic regression with model: bats ~ OBP



Sup Figure 2: pairwise scatterplot between all 9 pitch quality petric variables



Sup. figure 3: Paired scatter plot of all observations, coloured by handedness: blue = left-handed, red = right-handed.

Model based discriminant analysis

	L	R
L	33	39
R	15	175

Flexible discriminant analysis

	L	R
L	24	28
R	14	176

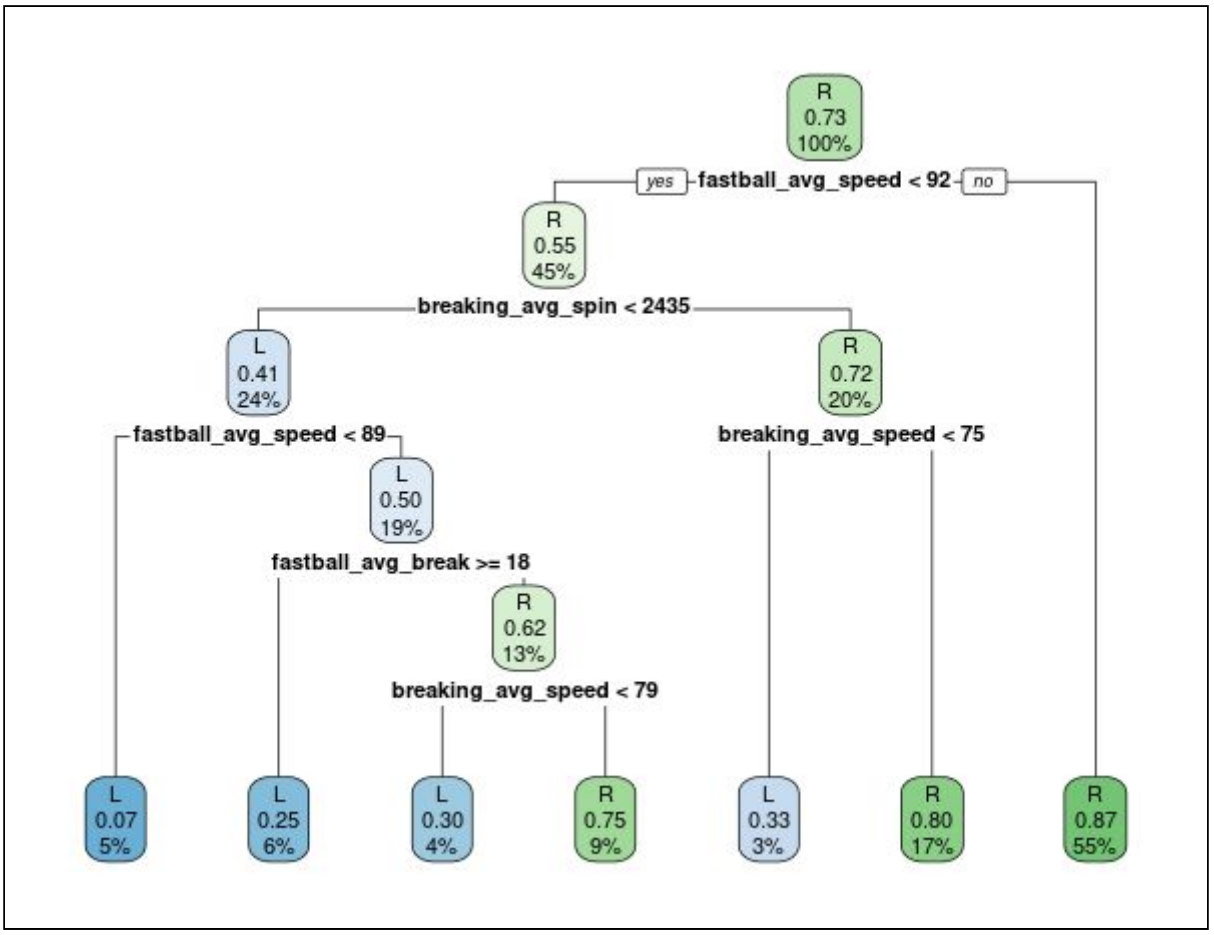
Naive Bayes classifier

	L	R
L	31	27
R	41	163

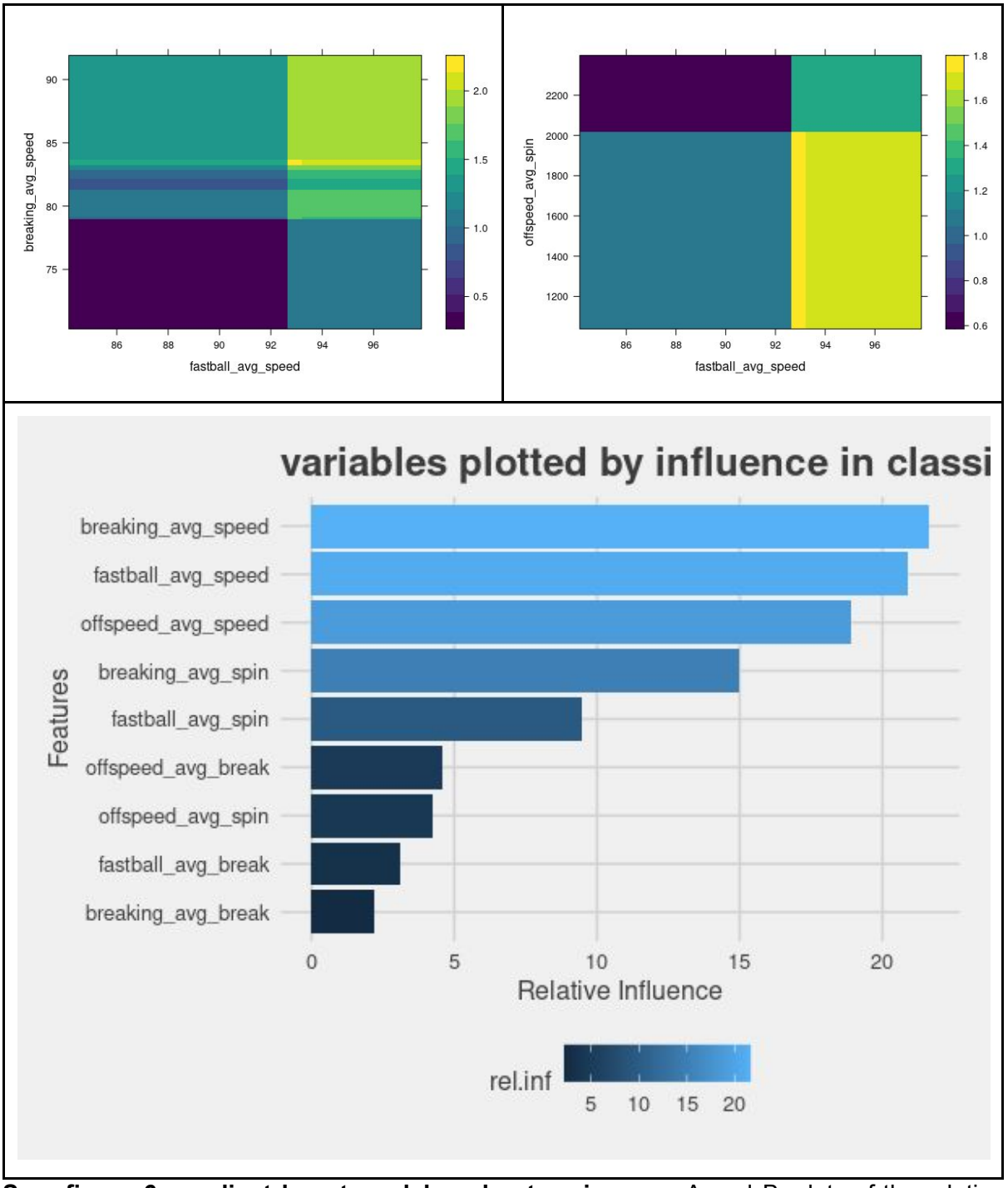
K-Nearest Neighbour (k=5)

	L	R
L	28	13
R	44	177

Sup. Figure 4: Classification outcomes of 4 “vanilla” classifiers. Rows are the actual known grouping structure, and columns are the classification results of the respective classifying methods.



Sup. Figure 5: Classification tree created with rpart. Each node contains: current classification, probability, and percentage of observations in that node. “Negative” branches (eg. lower speed, less break etc.) always seem to point towards a L classification.



Sup. figure 6: gradient boost model exploratory images: A and B: plots of the relative score variables assign an observation based on their respective values. A score of above 1 is a definite vote for Right-handed, while a lower score might contribute more to left-handed. A look at all combinations of these plots showed that most increasing values pulled the score towards right-handed classification (like the pattern of A), and already at a pretty low threshold, which could explain the model's tendency to classify R so easily.. The only exception was offspeed spin, where a higher value pulled the score towards left-handed.
C: Relative influence that every variable had in the final boosted model. It is clear that the speed variables hold the most weight in classification.

R Code

All code can also be found in the attached files, for improved readability

```
################################
## Confirming the Platoon Effect ###
################################

###################
# Preparing the data #
######################
library(dplyr)
#importing the data I webscraped from mlb.com/stats/
versusLeft <- read.csv("data/versusLeft.csv")
versusRight <- read.csv("data/versusRight.csv")
versusLeft <- versusLeft[,c("FIRSTNAME", "LASTNAME", "OBP")]
versusRight <- versusRight[,c("FIRSTNAME", "LASTNAME", "OBP")]

##This data doesn't have the batting hand yet, for that we merge with Lahman database
library(Lahman)
##contains all batters
data(Batting)
Batting <- Batting[Batting$yearID==2019,]
##contains bathand info and names
data(Master)
merged <- merge(Batting, Master)
merged <- merged[,c("bats","nameFirst","nameLast")]
colnames(merged)[2] <- "FIRSTNAME"
colnames(merged)[3] <- "LASTNAME"
df = merged[!duplicated(merged$LASTNAME, merged$LASTNAME),]

withBattingHand <- merge(versusLeft, df, by = c("LASTNAME", "FIRSTNAME"))
battingVersusLeft <- withBattingHand[!duplicated(withBattingHand$LASTNAME,
withBattingHand$LASTNAME),]
battingVersusLeft <- battingVersusLeft[!battingVersusLeft$bats=="B",]

withBattingHand <- merge(versusRight, df, by = c("LASTNAME", "FIRSTNAME"))
battingVersusRight <- withBattingHand[!duplicated(withBattingHand$LASTNAME,
withBattingHand$LASTNAME),]
battingVersusRight <- battingVersusRight[!battingVersusRight$bats=="B",]

summary(battingVersusLeft)
##39 Lefties, 65 Righties
hist(battingVersusLeft$OBP)
summary(battingVersusRight)
hist(battingVersusRight$OBP)
##Both OBP distributions seem normally distributed enough
```

```
#creating a list of dataframes
dfList <- NULL
dfList[[1]] = battingVersusLeft
dfList[[2]] = battingVersusRight
library(data.table)
combinedBatting <- rbindlist(dfList, idcol = "origin")
combinedBatting$origin <- factor(combinedBatting$origin, levels=c(1,2),labels = c("vsLeft",
"vsRight"))

table(combinedBatting$origin, combinedBatting$bats)

#####################
## Exploring the data ##
#####################
library(ggplot2)
ggplot(combinedBatting, aes(x=origin, y=OBP, fill=bats)) +
  geom_boxplot() +
  ggtitle("Batting performance versus left/right-handed pitchers") + xlab("Pitcher's
handedness")

###################################
## Exploring the platoon effect ##
###################################

tempLeft <- battingVersusLeft[battingVersusLeft$bats=="R" |
battingVersusLeft$bats=="L",c("OBP", "bats")]
tempRight <- battingVersusRight[battingVersusRight$bats=="R" |
battingVersusRight$bats=="L",c("OBP", "bats")]
#Logistic regression for batting vs left
left.log <- glm(bats ~ OBP, data=tempLeft, family=binomial(link="logit"))
summary(left.log)
exp(cbind(OR =left.log$coefficients, confint(left.log)))

#Logistic regression for batting vs right
right.log <- glm(bats ~ OBP, data=tempRight, family=binomial(link="logit"))
summary(right.log)
exp(cbind(OR =right.log$coefficients, confint(right.log)))

anova(left.log, test="Chisq")
anova(right.log,test="Chisq")

############
## Trees ##
############
##here I build a tree to see if based on OBP and batting hand, the tree can guess if this stat
combination is from the vsLeft or vsRight table
library(rpart)
library(rpart.plot)
```

```
combinedBatting <- combinedBatting[combinedBatting$bats=="R" |
combinedBatting$bats=="L"]
obp.tree<-rpart(origin~OBP + bats,combinedBatting,method="class")
rpart.plot(obp.tree,yesno=T)

rpart.pred <- predict(pruned.tree,type="class")
table(combinedBatting$origin, rpart.pred, dnn=c("From","Classified into"))
#it has about a 70% chance of getting it correct, which is pretty cool.
summary(obp.tree)


############################
## Ordination of pitching data ##
############################

##loading the data
library(dplyr)
pitchers <- read.csv("data/baseballsavant_2019.csv")
pitchers <- as.data.frame(pitchers)
names(pitchers)
##checking missing values
for(i in 1:ncol(pitchers)) {
  print(sum(is.na(pitchers[,i])))
}
pitchers <- subset(pitchers, select=-X)
pitchers <- na.omit(pitchers)

##Exploratory analysis
attach(pitchers)

##PCA
##selecting the numerical columns
mat <- pitchers[,7:15]
dim(mat)
pairs(mat)
##linear correlation between spin rate and break, and also between the different speeds
##little correlation between speed and spin, which is good
#install.packages("plot.matrix")
library(plot.matrix)
plot(cor(mat))
##from this I can see that there is a  negative correlation between breaking avg speed and
offspeed avg speed
pca <- princomp(mat)
plot(pca)
##it's clear that the first 3-4 pca's are probably just picking up on the three different types of
pitches
##It's probably just catching up on the 3 types of pitches
pca$loadings
```

```
screeplot(pca, type="lines")
summary(pca)
library(ggplot2)
qplot(pca$scores[,1], pca$scores[,2], colour=pitchers$pitch_hand)

#############
## BIPLOT ##
#############

library(plotrix)
PCA.biplot<- function(x) {
  #x is the matrix to biplot, x is numeric, thus variables on ratio or interval scales
  #x has dimnames(x)[[2]] defined (and eventually dimnames(x)[[1]] defined)
  xm<-apply(x,2,mean)
  y<-sweep(x,2,xm)
  ss<-(t(y)%*%y)
  s<-ss/(nrow(x)-1)
  d<-(diag(ss))^(-1/2)
  e<-diag(d,nrow=ncol(x),ncol=ncol(x))
  z<-y%*%e   #variables of z are unit length scaled
  r<-t(z)%*%z
  q<-svd(z)
  gfd<-((q$d[1])+(q$d[2]))/sum(q$d)
  gfz<-(((q$d[1])^2)+((q$d[2])^2))/sum((q$d)^2)
  gfr<-(((q$d[1])^4)+((q$d[2])^4))/sum((q$d)^4)
  l<-diag(q$d,nrow=ncol(x),ncol=ncol(x))
  R.B<-q$u        #scores matrix
  C.B<-q$v%*%l    #loadings
  #possibility to stretch scores by a scale factor
  #scalefactor<-3.5
  #R.B<-q$u *scalefactor

  par(mar=c(4,4,4,4),pty='s',oma=c(5,0,0,0),font=2)
  plot(R.B[ ,1],R.B[ ,2],axes=F,xlim=c(-1,1),ylim=c(-1,1),xlab=' ',ylab=' ',cex=.8)
  mtext('First component',side=1,line=3,cex=.8)
  mtext('Second component',side=2,line=3,cex=.8)
  title("Correlational Biplot on Pitch Quality Metric")
  axis(1,at=c(-1,-.8,-.6,-.4,-.2,0,.2,.4,.6,.8,1),cex=.8)
  axis(2,at=c(-1,-.8,-.6,-.4,-.2,0,.2,.4,.6,.8,1),cex=.8)
  box( )

  points(R.B[,1],R.B[,2],pch=".")


  points(C.B[,1],C.B[,2],pch=".")
  text(C.B[,1]-.05,C.B[,2]+.05,as.character(dimnames(x)[[2]]),cex=0.8)

  for (i in seq(1,nrow(C.B),by=1))
```

```
    arrows(0,0,C.B[i,1],C.B[i,2])

  #Draw circle unit
  draw.circle(0,0,1,border='black')
}

p.mat<- as.matrix(mat)
par(mfrow=c(1,1))
PCA.biplot(p.mat)

######################
## Factor analysis ##
######################

##WORKING FA from source####################
pit.fmle.r <- factanal(mat,factors=4,scores="Bartlett",rotation="varimax")
pit.fmle.r

plot(pit.fmle.r$loadings[,1],
     pit.fmle.r$loadings[,2],
     xlab = "Factor 1",
     ylab = "Factor 2",
     ylim = c(-1,1),
     xlim = c(-1,1),
     main = "Varimax rotation")

text(pit.fmle.r$loadings[,1]-0.08,
     pit.fmle.r$loadings[,2]+0.08,
     colnames(mat),
     col="blue")
################
## ----->> Factor 2 might be the difference in lefties/righties we see


###############################
## Clustering of pitch metric data ##
###############################


## This Rscript uses several (blind) clustering methods to see if any of them pick up the
difference in lefties and righties as a cluster
library(plot.matrix)
pitchers <- read.csv("data/baseballsavant_2019.csv")
for(i in 1:ncol(pitchers)) {
  print(sum(is.na(pitchers[,i])))
}
pitchers <- subset(pitchers, select=-X)
pitchers <- na.omit(pitchers)
```

```
table(pitchers$pitch_hand)
##72 lefties, 190 righties

##exploring known grouping structure "Pitch hand"
par(mfrow=c(1,1))
group <- NA
group[pitchers$pitch_hand =="R"] <- 2
group[pitchers$pitch_hand == "L"] <- 1
pairs(pitchers[,7:15], col = c("blue","red")[group])
##It's obvious that I'm not going to find a nice linear split anywhere.
##lefties seem to be on the "lower" side of all of the variables tho

##############################
## Hierarchical Clustering ##
##############################
pit.mat <- as.matrix(pitchers[,7:15])
rownames(pit.mat) <-pitchers$last_name
colnames(pit.mat) <- colnames(pitchers)[7:15]
pit.clus <-hclust(dist(pit.mat), method="average")
plot(pit.clus)
##it looks like there is an outlier the causes an immediate split, let's take that one out
pit.mat_no_outliers <- pit.mat[!rownames(pit.mat)%in%"Baez",]
pit.clus2 <-hclust(dist(pit.mat_no_outliers), method="average")
plot(pit.clus2)
##looks a bit better, i'll continue without the outlier
pit.cutTree <- cutree(pit.clus2,k=2) ## --> contains index vector you can use to color your
plots
pairs(pitchers[,7:15], col = c("red","purple")[pit.cutTree])
#It's obvious that hierarchical clustering at k=2 level finds a completely different seperation

############################
## Non-Hierarchical Clustering ##
############################
pit.kmeans <- kmeans(pit.mat, 2)
pairs(pitchers[,7:15], col= c("blue","red")[pit.kmeans$cluster] )
#def not a perfecct group structure
library(cluster)
clusplot(pit.mat,pit.kmeans$cluster,stand=TRUE,main="k-means clustering on pitchers
data")
table(pitchers$pitch_hand,pit.kmeans$cluster)
##left vs right is also here not the type of split that is being picked up by kmeans

############################
## Bayesian clustering with EM ##
############################
library(mclust)
fit <- Mclust(pit.mat)
plot(fit)
```

```
###choose option 2 here to look at a similar classification scheme as i've been doing with
the pairs plot
summary(fit)
table(pitchers$pitch_hand, fit$classification)



####################################
## Classification of pitching metric data ##
####################################

#The idea of this script is to see if i can make a supervised classifier that can tell the
difference between a left handed
#or right handed pitcher based on their pitch metrics.
pitchers <- read.csv("data/baseballsavant_2019.csv")
for(i in 1:ncol(pitchers)) {
  print(sum(is.na(pitchers[,i])))
}
pitchers <- subset(pitchers, select=-X)
pitchers <- na.omit(pitchers)
table(pitchers$pitch_hand)

############
## TREES ##
############
attach(pitchers)
library(rpart)
library(rpart.plot)
rpart.tree<-rpart(pitch_hand~ fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break,pitchers,method="class")
rpart.plot(rpart.tree, yesno=TRUE)
printcp(rpart.tree)
plotcp(rpart.tree)

rpart.pred <- predict(rpart.tree,type="class")
table(pitchers$pitch_hand, rpart.pred, dnn=c("From","Classified into"))
##alright alright this tree is starting to do a lil better: 38-34

##checking overfitting by applying this tree on the pitching dataset of 2020
pitchers_2020 <- read.csv("data/baseballsavant_2020.csv")
##how many Left and Right handed pitchers are there actually
pitchers_2020 <- subset(pitchers_2020, select=-X)
pitchers_2020 <- na.omit(pitchers_2020)
rpart.pred <- predict(rpart.tree, pitchers_2020,type="class")
table(pitchers_2020$pitch_hand, rpart.pred, dnn=c("From","Classified into"))
```

```
###################################
## Attempt at boosting the tree ##
###################################
library(rsample)
library(gbm)
library(caret)
pitchers$pitch_hand <- as.numeric(as.factor(pitchers$pitch_hand))-1
##0 = left, 1 = right
##split data in training and test
pit.split <- initial_split(pitchers, prop = .7)
pit.train <- training(pit.split)
pit.test  <- testing(pit.split)
attach(pit.train)

##try a random hyper parameter config
gbm.fit <- gbm(
  formula = pitch_hand ~ fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break,
  data = pit.train,
  distribution = "bernoulli",
  n.trees = 10000,
  interaction.depth = 1,
  shrinkage = 0.001,
  cv.folds = 5,
  n.cores = NULL, # will use all cores by default
  verbose = FALSE
)
print(gbm.fit)
##RMSE
sqrt(min(gbm.fit$cv.error))
gbm.perf(gbm.fit, method = "cv")

##grid searching for the best parameters
hyper_grid <- expand.grid(
  shrinkage = c(.01, .1, .3),
  interaction.depth = c(1, 3, 5),
  n.minobsinnode = c(5, 10, 15),
  bag.fraction = c(.65, .8, 1),
  optimal_trees = 0,
  min_RMSE = 0
)

# randomize data before using train.fraction
random_index <- sample(1:nrow(pit.train), nrow(pit.train))
random_pit_train <- pit.train[random_index, ]

##grid search to find the best parameters
```

```
for(i in 1:nrow(hyper_grid)) {

  # train model
  gbm.tune <- gbm(
    formula = pitch_hand ~ fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break,
    distribution = "bernoulli",
    data = random_pit_train,
    n.trees = 5000,
    interaction.depth = hyper_grid$interaction.depth[i],
    shrinkage = hyper_grid$shrinkage[i],
    n.minobsinnode = hyper_grid$n.minobsinnode[i],
    bag.fraction = hyper_grid$bag.fraction[i],
    train.fraction = .75,
    n.cores = NULL, # will use all cores by default
    verbose = FALSE
  )

  # add min training error and trees to grid
  hyper_grid$optimal_trees[i] <- which.min(gbm.tune$valid.error)
  hyper_grid$min_RMSE[i] <- sqrt(min(gbm.tune$valid.error))
}

##this shows the best 10 model configs, I choose the parameters of the top model
##This is a stochastic algorithm so if you rerun this, the parameters might differ
hyper_grid %>%
  dplyr::arrange(min_RMSE) %>%
  head(10)

gbm.fit.final <- gbm(
  formula = pitch_hand ~ fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break,
  distribution = "bernoulli",
  data = pit.train,
  n.trees = 6,
  interaction.depth = 5,
  shrinkage = 0.3,
  n.minobsinnode = 15,
  bag.fraction = .65,
  train.fraction = 1,
  cv.folds = 5,
  n.cores = NULL, # will use all cores by default
  verbose = FALSE
)
```

```
summary(
  gbm.fit.final,
  cBars = 10,
  method = relative.influence, # also can use permutation.test.gbm
  las = 2
)
##predict using the final model on the training data
pred <- predict.gbm(gbm.fit.final, n.trees = gbm.fit.final$n.trees, pit.test, type="response")
calibrate.plot(
  pit.test$pitch_hand,
  pred,
  distribution = "bernoulli",
  replace = TRUE,
  line.par = list(col = "black"),
  shade.col = "lightyellow",
  shade.density = NULL,
  rug.par = list(side = 1),
  xlab = "Predicted value",
  ylab = "Observed average",
  xlim = NULL,
  ylim = NULL,
  knots = NULL,
  df = 6,
)

## exploring results
##calculating RMSE
caret::RMSE(pred, pit.test$pitch_hand)
##plotting decision making of variables
plot.gbm(gbm.fit.final, i.var=c(1,4))
plot.gbm(gbm.fit.final, i.var=c(1,8))

##exploring relative influence of all variables
library(dplyr)
effects <- tibble::as_tibble(gbm::summary.gbm(gbm.fit.final,
                                              plotit = FALSE))
effects %>% utils::head()
library(ggthemes)
effects %>%
  # arrange descending to get the top influencers
  dplyr::arrange(desc(rel.inf)) %>%
  # sort to top 10
  dplyr::top_n(9) %>%
  # plot these data using columns
  ggplot(aes(x = forcats::fct_reorder(.f = var,
                                      .x = rel.inf),
             y = rel.inf,
             fill = rel.inf)) +
```

```
  geom_col() +
  # flip
  coord_flip() +
  # format
  scale_color_brewer(palette = "Dark2") +
  theme_fivethirtyeight() +
  theme(axis.title = element_text()) +
  xlab('Features') +
  ylab('Relative Influence') +
  ggtitle("variables plotted by influence in classification score")

#I choose to put an arbitrary threshold on 0.7, this is a part that i do not 100% understand, so there is bias in this decisian
table(pit.test$pitch_hand==0,pred<0.7)

#############################################################################
## End of boosting attempt: This turned out to be quite a topic, so i'm not 100% sure I did everything correctly ##
#############################################################################

##trying another tree building method with pruning
library(tree)
attach(pitchers)
pitchers$pitch_hand <- as.factor(pitchers$pitch_hand)
tree.tree <-tree(pitch_hand~fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+ breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+ offspeed_avg_spin+ offspeed_avg_break,
data=pitchers,method="recursive.partition",split="deviance")
plot(tree.tree)
text(tree.tree)
##here i have unnecessary splits in the end, so i'm gonna have to do some pruning
tree.tree.cv <- cv.tree(tree.tree,FUN=prune.tree)
plot(tree.tree.cv$size,tree.tree.cv$k,type="l")
plot(tree.tree.cv$size,tree.tree.cv$dev,type="l",xlab="Number of end nodes",ylab="Deviance")
#use info from these plots to then prune the actual tree
pruned.tree<-prune.tree(tree.tree,best=3)
plot(pruned.tree)
text(pruned.tree)
summary(pruned.tree)
tree.pred <- predict(pruned.tree,type="class")
table(pitchers$pitch_hand, tree.pred, dnn=c("From","Classified into"))
##here it seems that the process really seems to prefer classifying things into R instead of R, since it's dominating.
##this might be a clue that pointing the tree towards classifying L correctly (like boosting) might improve performance.
##40-32
```

```
##############################
## Multiple vanilla ML Classifiers: ##
##############################

#Mclust
library(mclust)
clas <- MclustDA(pitchers[,7:15], pitchers$pitch_hand)
summary(clas)
##33-39
plot(clas)
##choose 2 for a pairs-classification plot like previously

##flexible discriminant analysis
library(mda)
# fit model
fit <- fda(pitch_hand~fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break, data=pitchers)
summary(fit)
predictions <- predict(fit, pitchers[,7:15])
table(pitchers$pitch_hand, predictions)
##24-48

##KNN
# fit model
attach(pitchers)
library(caret)
fit <- knn3(pitch_hand~fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break, data=pitchers, k=5)
summary(fit)
predictions <- predict(fit, pitchers[,7:15], type="class")
table(pitchers$pitch_hand, predictions)
##28-44

##Naive Bayes
library(e1071)
# fit model
fit <- naiveBayes(pitch_hand~fastball_avg_speed+ fastball_avg_spin+ fastball_avg_break+
breaking_avg_speed+ breaking_avg_spin+ breaking_avg_break+ offspeed_avg_speed+
offspeed_avg_spin+ offspeed_avg_break, data=pitchers)
summary(fit)
predictions <- predict(fit, pitchers[,7:15])
table(pitchers$pitch_hand, predictions)
##31-41
```

Python Code (used for web scraping mlb.com/stats/)

```
import requests
import pandas as pd

dataframeDict = {}
versusList = ["vr","vl"]
# battingList = ["l", "r"]
for versus in versusList:
    dataframes = []
    for i in range(1,7):
        url = 'https://www.mlb.com/stats/2019?split=' + versus + '&page=' + str(i)
        html = requests.get(url).content
        df_list = pd.read_html(html)
        df = df_list[-1]
        dataframes.append(df)
    dataframeDict['versus_{}'.format(versus)] = pd.concat(dataframes)
for k, v in dataframeDict.items():
    v.to_csv('{}.csv'.format(str(k)))




#importing necessary packages
import pandas as pd
import os


##read in data
dfVersusLeft = pd.read_csv("versus_vl.csv")
dfVersusRight = pd.read_csv("versus_vr.csv")
dfVersusList = [dfVersusLeft, dfVersusRight]


##reading and shifting the columns a bit
#creating new columns for later
for df in dfVersusList:
    df.drop("Unnamed: 0", inplace=True, axis=1)
    df.rename(columns={"caret-upcaret-downOPS": "OPS"}, inplace=True)
    df["FIRSTNAME"] = df["PLAYER"]
    df["LASTNAME"] = df["PLAYER"]



##Parsing the name column since it is ineligable and I need last name identifiers to merge
with another
##database in R
def extractLastName(string):
```

```
    temp = string[:len(string) // 2]
    return temp

def extractFirstName(string):
    temp = string[:-1]
    return temp

for df in dfVersusList:
    for i,player in enumerate(df["PLAYER"]):
        if i <10:
            new_player = player[1:len(player)-6]
        elif i<100:
            new_player = player[2:len(player)-7]
        else:
            new_player = player[3:len(player)-8]
        splitName = new_player.split(" ")
        df["FIRSTNAME"].replace({player: extractFirstName(splitName[0])},inplace=True)
        df["LASTNAME"].replace({player: extractLastName(splitName[1])},inplace=True)
        df["PLAYER"].replace({player: new_player}, inplace=True)


dfVersusList[0].to_csv("versusLeft.csv")
dfVersusList[1].to_csv("versusRight.csv")
```