PREDICTING THE NEXT PITCH IN AN MLB GAME

A PROJECT REPORT

Presented to the Department of Mathematics and Statistics  
California State University, Long Beach  
In Partial Fulfillment of the Requirements for the Degree  
Master of Science in Applied Statistics

BY RJ BURSON

MAY 2022

1. Introduction

The dataset I am using for this project is from baseball savant. The data set contains all pitches from Clayton Kershaw from 2015 and 2016 seasons. The data set contains 999 pitches and 13 variables about each pitch. There is a quote from one of the greatest hitters of all time, Hank Aaron. “Guessing what the pitcher is going to throw is 80% of being a successful hitter. The other 20% is just execution.” I think that this is even more true today with pitchers throwing harder and hitters being able to get less plate appearances per pitcher due to increased bullpen usage. After building 12 models I found that the best one could predict if a fastball or off-speed pitch would be thrown on the next pitch 68% of the time. This would be a useful piece of information for MLB hitters going up to hit at the plate. Another finding during this research was that the most important predictors are how many balls and strikes there are in the count, what the batter handedness is, and what inning it is when trying to guess the next pitch that will be thrown

1. Questions of Interest
2. In this research I will be trying to predict the next pitch that will be thrown
3. Which variables are most useful when trying to predict the next pitch?
4. Analysis

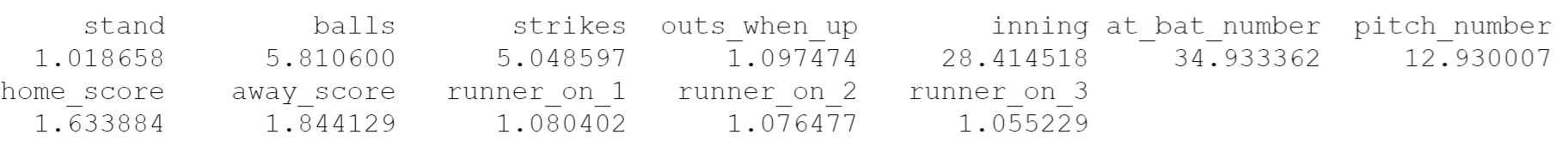
In this analysis I attempt to predict the next pitch using most of the classification models that we have learned in class. I will also see which predictors are important in each of the models built and see if there are variables that are important in many or all the models.

Each row in the data set is an individual pitch thrown by Clayton Kershaw. There are 12 predictor variables and a response factor variable fastball. The response variable is equal to 1 if the pitch is a fastball and the response variable is equal to 0 when it is an off-speed pitch. The first predictor variable is stand which is for the side the batter hits from. It is a factor variable with 1 being a right-handed hitter and 0 being a left-handed hitter. This is important because hitting against a same handed pitcher is usually harder for most hitters. The variables balls and strikes are how many balls and strikes there are in the count. Outs\_when\_up is the number of outs during the plate appearance. The variable inning is what inning it is during the plate appearance. The variable at bat number is the number of batters faced in the game including the current batter. The variable pitch number is what pitch number this is of the current plate appearance. The variable home score and away score indicate the scores for the home and away teams in the game. The variable runner on 1 is a factor variable. If there is a runner on 1st, it is equal to 1 if there is a runner on 1st and 0 if there is not a runner on first. The variable runner on 2 is a factor variable. If there is a runner on 2nd, it is equal to 1 if there is a runner on 2nd and 0 if there is not a runner on 2nd. The variable runner on 3 is a factor variable. If there is a runner on 3rd, it is equal to 1 if there is a runner on 3rd and 0 if there is not a runner on 3rd.

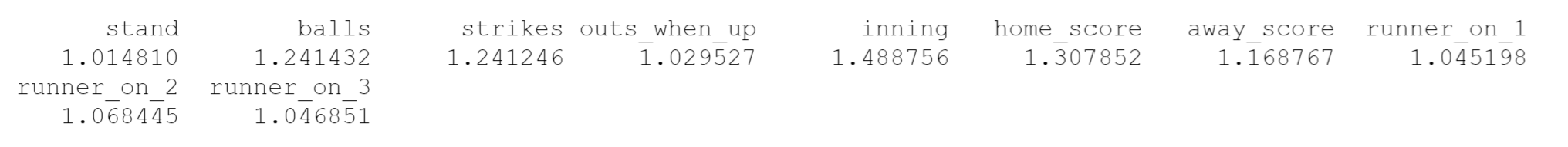
Table

Description automatically generated

The first step I had to complete before attempting any data analysis was splitting the data into training and testing data sets. I split the data with 80% of the observations going into the training set and 20% of the observations going in the testing set. When I checked the model for multicollinearity there looked to be a problem with the inning and at bat number variables.



Chose to drop the pitch number and at bat number variable because they each were correlated with other variables. Pitch number is correlated with the number of balls and strikes in a count in an at bat. At bat number is correlated to inning because they both have to do with how far into the game the at bat is taking place.



**Logistic Model**

With the new subset of variables free of multicollinearity, I used them to fit a logistic model on the training set. The initial logistic model had 5 predictors that were significant, stand, strikes, balls, inning, and runner on 1. Fit a reduced logistic model with only the variables that were significant in the logistic model summary output.

A picture containing text

Description automatically generated

This model was able to predict 68% of the test data and it had an AUC value of 0.7147746.

**Lasso Model**

Next, I fit a lasso model on the training set to see if any predictor variables weren’t needed and to see which variables are the most important.

Chart

Description automatically generated Text

Description automatically generated with medium confidence

The lasso plot shows the four most significant predictors to be strikes, balls, stand, and inning. None of the coefficients went to zero in the coefficients plot.

**QDA Model**

Since the data set I am using for the data analysis had many training observations I decided to used a QDA model instead of a LDA model.

Chart, line chart, scatter chart

Description automatically generated A picture containing text

Description automatically generated

The QDA model was able to predict the test data set with 0.65% accuracy and had an AUC value of 0.6907206.

**Classification Tree**

Built a classification tree next to try and predict whether a fastball or off-speed pitch was coming next. The classification tree did not perform very well compared to the other models.

Diagram

Description automatically generated Diagram

Description automatically generated

The full tree is on the left and the pruned tree is the tree on the right. This pruned tree model was only able to predict the test data set with 0.62% accuracy and had an AUC value of 0.6372283.

**Bagging Model**

Chart, line chart, scatter chart

Description automatically generatedA picture containing text, clock

Description automatically generated

The bagging model was able to predict the test data set with 0.635% accuracy and had an AUC value of 0.6497585.

Chart, table

Description automatically generated

These graphs show what variables are the most important. From these graphs it looks as though strikes balls, inning, home score, away score, and stand are the important variables.

**Random Forest Model**

A picture containing text, clock

Description automatically generatedChart, line chart, scatter chart

Description automatically generated

The random forest model was able to predict the test data set with 0.635% accuracy and had an AUC value of 0.6640499.

Chart, table

Description automatically generated

Based on these models the important variables are strikes, balls, inning, and home score.

**Boosting Model**

I fit the first boosting model using 100 trees, a shrinkage value of .01 and an interaction depth of 1.

Chart, line chart, scatter chart

Description automatically generatedA picture containing calendar

Description automatically generated

The boosting model was able to predict the test data set with 0.68% accuracy and had an AUC value of 0.7205616.

Table

Description automatically generated

This table shows that strikes, balls, and inning are the three most important predictors with strikes being by far the most important.

**Tuned Boosted Model**

After running the code to check for combinations of variables for the tuned boosted model. The combination with the lowest misclassification rate was the model with 200 trees, a shrinkage value of .04, and an interaction depth of 2.

Chart, line chart

Description automatically generatedA picture containing text

Description automatically generated

The tuned boosting model was able to predict the test data set with 0.68% accuracy and had an AUC value of 0.7074779.

**SVM Model: Linear**

A picture containing text

Description automatically generated Chart, line chart, scatter chart

Description automatically generated

I ran all a linear, radial, and polynomial svm models but chose to use the linear svm model because it predicted the data with the highest accuracy at 68% and had an AUC value of 0.7109501.

1. Conclusion

The best model was a close decision because multiple models had a prediction accuracy of .68%. It can down to the AUC values and the boosted model narrowly beat out the other as the model I chose. The boosting model with 100 trees, shrinkage value of .01, and interaction depth of 1 could predict whether a fastball of off-speed pitch will be thrown 68% of the time and had an AUC value of 0.7205616. When it came to most significant predictors the same variables were consistently the most important across all of the models. Strikes, balls, inning, stand, and the score between the teams were clearly the most significant.

I believe this model could be improved upon by not just having a factor variable indicating if a runner is on base, it may be useful to have a specific player id for the runner on base. If it is a fast or slow runner it could influence the pitches being thrown to the plate. Another possible term/terms to add to the model would be to include the previous pitch or all previous pitches in the at bat to also predict what will be thrown next. I don’t think these results are very general because the data only has pitches specifically thrown by Clayton Kershaw. This same exact code could be applied to another pitcher, and it would be interesting to see how it would perform on them. I think even more interesting and the analysis that would give general results to try and be able to apply to every pitcher would be to get a huge data set of every single pitch in a year in the MLB and predict what pitch will come next.

1. Appendix

**R Code**

df = as\_tibble(read.csv("C:/Users/RJ Burson/Downloads/savant\_data.csv"))  
  
df = df[,c(18,19,25,26,27,32:36,77:83)]  
  
df$runner\_on\_1= as.factor(ifelse(is.na(df$on\_1b),0,1))  
df$runner\_on\_2= as.factor(ifelse(is.na(df$on\_2b),0,1))  
df$runner\_on\_3= as.factor(ifelse(is.na(df$on\_3b),0,1))  
  
df$stand = as.factor(ifelse(df$stand=="R",1,0))  
  
df = df[,-c(2,6,7,8)]  
  
  
  
df= mutate(df, fastball=ifelse(df$pitch\_name %in% c("4-Seam Fastball"),1,0))  
  
df=df[,-9]  
  
skim(df)

Data summary

|  |  |
| --- | --- |
| Name | df |
| Number of rows | 999 |
| Number of columns | 16 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 4 |
| numeric | 12 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| stand | 0 | 1 | FALSE | 2 | 1: 763, 0: 236 |
| runner\_on\_1 | 0 | 1 | FALSE | 2 | 0: 766, 1: 233 |
| runner\_on\_2 | 0 | 1 | FALSE | 2 | 0: 870, 1: 129 |
| runner\_on\_3 | 0 | 1 | FALSE | 2 | 0: 946, 1: 53 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| balls | 0 | 1 | 0.98 | 0.96 | 0 | 0 | 1 | 2 | 3 | ▇▆▁▅▂ |
| strikes | 0 | 1 | 1.05 | 0.82 | 0 | 0 | 1 | 2 | 2 | ▇▁▇▁▇ |
| game\_year | 0 | 1 | 2015.43 | 0.50 | 2015 | 2015 | 2015 | 2016 | 2016 | ▇▁▁▁▆ |
| outs\_when\_up | 0 | 1 | 0.95 | 0.81 | 0 | 0 | 1 | 2 | 2 | ▇▁▇▁▇ |
| inning | 0 | 1 | 4.09 | 2.16 | 1 | 2 | 4 | 6 | 9 | ▇▇▃▆▂ |
| at\_bat\_number | 0 | 1 | 28.96 | 17.29 | 1 | 15 | 28 | 43 | 80 | ▇▇▇▅▁ |
| pitch\_number | 0 | 1 | 3.21 | 1.80 | 1 | 2 | 3 | 4 | 10 | ▇▇▃▁▁ |
| home\_score | 0 | 1 | 1.26 | 1.66 | 0 | 0 | 1 | 2 | 8 | ▇▂▁▁▁ |
| away\_score | 0 | 1 | 1.01 | 1.45 | 0 | 0 | 1 | 2 | 11 | ▇▁▁▁▁ |
| bat\_score | 0 | 1 | 0.76 | 1.11 | 0 | 0 | 0 | 1 | 5 | ▇▁▁▁▁ |
| fld\_score | 0 | 1 | 1.52 | 1.83 | 0 | 0 | 1 | 2 | 11 | ▇▂▁▁▁ |
| fastball | 0 | 1 | 0.54 | 0.50 | 0 | 0 | 1 | 1 | 1 | ▇▁▁▁▇ |

dim(df)

## [1] 999 16

## Splitting the data set

n = nrow(df)  
prop = .8  
set.seed(123)  
train\_id = sample(1:n, size = n\*prop, replace = FALSE)  
test\_id = (1:n)[-which(1:n %in% train\_id)]  
train\_set = df[train\_id, ]  
test\_set = df[test\_id, ]

## Multicolinearity

# Logistic model  
m1 = glm(fastball~stand+balls+strikes+outs\_when\_up+inning+at\_bat\_number+pitch\_number+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, data = train\_set)  
  
summary(m1)

##   
## Call:  
## glm(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + at\_bat\_number + pitch\_number + home\_score + away\_score +   
## runner\_on\_1 + runner\_on\_2 + runner\_on\_3, data = train\_set)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.9402 -0.3981 0.1224 0.3608 1.0336   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.631075 0.065232 9.674 < 2e-16 \*\*\*  
## stand1 0.086082 0.038007 2.265 0.0238 \*   
## balls 0.025593 0.040170 0.637 0.5242   
## strikes -0.344521 0.043499 -7.920 8.09e-15 \*\*\*  
## outs\_when\_up -0.008997 0.020584 -0.437 0.6622   
## inning 0.054066 0.039203 1.379 0.1683   
## at\_bat\_number -0.010575 0.005427 -1.949 0.0517 .   
## pitch\_number 0.066964 0.031954 2.096 0.0364 \*   
## home\_score 0.026097 0.012163 2.146 0.0322 \*   
## away\_score 0.003002 0.014879 0.202 0.8402   
## runner\_on\_11 0.089445 0.038857 2.302 0.0216 \*   
## runner\_on\_21 -0.034913 0.048964 -0.713 0.4760   
## runner\_on\_31 0.096688 0.071799 1.347 0.1785   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.2031156)  
##   
## Null deviance: 198.35 on 798 degrees of freedom  
## Residual deviance: 159.65 on 786 degrees of freedom  
## AIC: 1008.8  
##   
## Number of Fisher Scoring iterations: 2

varImp(m1)

## Overall  
## stand1 2.2649132  
## balls 0.6371016  
## strikes 7.9202107  
## outs\_when\_up 0.4370655  
## inning 1.3791159  
## at\_bat\_number 1.9485942  
## pitch\_number 2.0956127  
## home\_score 2.1456202  
## away\_score 0.2017666  
## runner\_on\_11 2.3018968  
## runner\_on\_21 0.7130482  
## runner\_on\_31 1.3466479

vif(m1)

## stand balls strikes outs\_when\_up inning   
## 1.018658 5.810600 5.048597 1.097474 28.414518   
## at\_bat\_number pitch\_number home\_score away\_score runner\_on\_1   
## 34.933362 12.930007 1.633884 1.844129 1.080402   
## runner\_on\_2 runner\_on\_3   
## 1.076477 1.055229

## multicolinearity is a problem  
  
## Model without at\_bat\_number and pitch\_number. Chose these two variables because pith number would be correlated with the number of balls and strikes in a count in an ab. At bat number would be correlated to innint because they both have to do with how far into the game the ab is taking place.   
  
m1 = glm(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, data = train\_set)  
  
#Attempting to predict the test set  
pred = predict(m1, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
pred = as.factor(ifelse(predict(m1, test\_set, type="response")>optimal,1,0))  
tb = table(pred = pred, truth = test\_set$fastball)  
tb

## truth  
## pred 0 1  
## 0 68 41  
## 1 24 67

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.675

vif(m1)

## stand balls strikes outs\_when\_up inning home\_score   
## 1.014810 1.241432 1.241246 1.029527 1.488756 1.307852   
## away\_score runner\_on\_1 runner\_on\_2 runner\_on\_3   
## 1.168767 1.045198 1.068445 1.046851

## Model without inning and pitch\_number  
  
  
m1 = glm(fastball~stand+balls+strikes+outs\_when\_up+at\_bat\_number+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, data = train\_set)  
  
#Attempting to predict the test set  
pred = predict(m1, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
pred = as.factor(ifelse(predict(m1, test\_set, type="response")>optimal,1,0))  
tb = table(pred = pred, truth = test\_set$fastball)  
tb

## truth  
## pred 0 1  
## 0 61 33  
## 1 31 75

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.68

vif(m1)

## stand balls strikes outs\_when\_up at\_bat\_number   
## 1.014740 1.241175 1.240865 1.037746 1.829892   
## home\_score away\_score runner\_on\_1 runner\_on\_2 runner\_on\_3   
## 1.448541 1.338697 1.051685 1.065453 1.044876

######### Use models without at\_bat\_number and pitch\_number

## Initial logistic model with all variables

# Logistic model  
m1 = glm(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, data = train\_set)  
  
summary(m1)

##   
## Call:  
## glm(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + home\_score + away\_score + runner\_on\_1 + runner\_on\_2 +   
## runner\_on\_3, data = train\_set)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.9528 -0.4087 0.1263 0.3597 0.9585   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.736643 0.054196 13.592 < 2e-16 \*\*\*  
## stand1 0.082932 0.038092 2.177 0.0298 \*   
## balls 0.099646 0.018644 5.345 1.19e-07 \*\*\*  
## strikes -0.263521 0.021658 -12.168 < 2e-16 \*\*\*  
## outs\_when\_up -0.020212 0.020019 -1.010 0.3130   
## inning -0.020465 0.009011 -2.271 0.0234 \*   
## home\_score 0.015612 0.010927 1.429 0.1535   
## away\_score -0.015651 0.011894 -1.316 0.1886   
## runner\_on\_11 0.074221 0.038376 1.934 0.0535 .   
## runner\_on\_21 -0.044621 0.048982 -0.911 0.3626   
## runner\_on\_31 0.100637 0.071809 1.401 0.1615   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.2047963)  
##   
## Null deviance: 198.35 on 798 degrees of freedom  
## Residual deviance: 161.38 on 788 degrees of freedom  
## AIC: 1013.4  
##   
## Number of Fisher Scoring iterations: 2

varImp(m1)

## Overall  
## stand1 2.1771689  
## balls 5.3445676  
## strikes 12.1675558  
## outs\_when\_up 1.0096149  
## inning 2.2711781  
## home\_score 1.4287803  
## away\_score 1.3158511  
## runner\_on\_11 1.9340278  
## runner\_on\_21 0.9109608  
## runner\_on\_31 1.4014608

vif(m1)

## stand balls strikes outs\_when\_up inning home\_score   
## 1.014810 1.241432 1.241246 1.029527 1.488756 1.307852   
## away\_score runner\_on\_1 runner\_on\_2 runner\_on\_3   
## 1.168767 1.045198 1.068445 1.046851

#Attempting to predict the test set  
pred = predict(m1, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
pred = as.factor(ifelse(predict(m1, test\_set, type="response")>optimal,1,0))  
tb = table(pred = pred, truth = test\_set$fastball)  
tb

## truth  
## pred 0 1  
## 0 68 41  
## 1 24 67

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.675

#Plot ROC curve  
glm\_pred\_class = predict(m1, test\_set, type="response")  
pred = prediction(glm\_pred\_class, test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.7176932

############# Reduced glm model ####################  
  
  
m1\_reduced= glm(fastball~stand+balls+strikes+inning+runner\_on\_1, data = train\_set)  
summary(m1\_reduced)

##   
## Call:  
## glm(formula = fastball ~ stand + balls + strikes + inning + runner\_on\_1,   
## data = train\_set)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -0.9468 -0.4078 0.1431 0.3596 0.9362   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.718251 0.051496 13.948 < 2e-16 \*\*\*  
## stand1 0.080315 0.038127 2.107 0.0355 \*   
## balls 0.096376 0.018497 5.210 2.41e-07 \*\*\*  
## strikes -0.260619 0.021658 -12.033 < 2e-16 \*\*\*  
## inning -0.019032 0.007419 -2.565 0.0105 \*   
## runner\_on\_11 0.070916 0.037827 1.875 0.0612 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for gaussian family taken to be 0.2056229)  
##   
## Null deviance: 198.35 on 798 degrees of freedom  
## Residual deviance: 163.06 on 793 degrees of freedom  
## AIC: 1011.7  
##   
## Number of Fisher Scoring iterations: 2

varImp(m1\_reduced)

## Overall  
## stand1 2.106503  
## balls 5.210239  
## strikes 12.033290  
## inning 2.565141  
## runner\_on\_11 1.874751

#Attempting to predict the test set  
pred = predict(m1\_reduced, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
pred = as.factor(ifelse(predict(m1\_reduced, test\_set, type="response")>optimal,1,0))  
tb = table(pred = pred, truth = test\_set$fastball)  
tb

## truth  
## pred 0 1  
## 0 68 40  
## 1 24 68

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.68

#Plot ROC curve  
glm\_pred\_class = predict(m1\_reduced, test\_set, type="response")  
pred = prediction(glm\_pred\_class, test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, line chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.7147746

## Lasso regression to deal with multicollinearity and see which variables to keep.

xmat = model.matrix(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, train\_set)[,-1]  
  
  
y = train\_set$fastball  
xmat = apply(xmat, 2, function (x) scale(x, center=FALSE))  
  
mod.lasso = glmnet(xmat, y, alpha=1, family="binomial")  
  
plot(mod.lasso, xvar = "lambda", label = TRUE)

Chart

Description automatically generated

#Predicting lasso model  
set.seed(123)  
cv.out = cv.glmnet(xmat, y, alpha=1, nfolds=10, family=binomial)  
best.lambda = cv.out$lambda.min  
  
  
pcoefs = predict(mod.lasso, s = best.lambda, type = "coefficients")  
pcoefs

## 11 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) 1.07147232  
## stand1 0.33928128  
## balls 0.63917468  
## strikes -1.59612006  
## outs\_when\_up -0.10813462  
## inning -0.43904829  
## home\_score 0.14103854  
## away\_score -0.13032258  
## runner\_on\_11 0.17224682  
## runner\_on\_21 -0.06898433  
## runner\_on\_31 0.10993770

mod.lasso.best = glmnet(xmat, y, alpha=1, lambda = best.lambda)  
yhat.lasso.best = predict(mod.lasso.best, newx = xmat, type = "response")#predict.glmnet  
  
  
#### Create confusion matrix for test set  
  
new\_xmat = model.matrix(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, test\_set)[,-1]  
  
y = test\_set$fastball  
  
  
pred = predict(mod.lasso.best, newx = new\_xmat, type = "response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
pred = as.factor(ifelse(predict(mod.lasso.best, new\_xmat, type="response")>optimal,1,0))  
tb = table(pred = pred, truth = test\_set$fastball)  
tb

## truth  
## pred 0 1  
## 0 62 39  
## 1 30 69

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.655

## Since the data set has many training observation use QDA model instead of LDA.

train\_qda\_fit = qda(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3, data = train\_set)  
  
# Confusion matrix for test set  
qda\_pred\_class = predict(train\_qda\_fit, test\_set)$class  
tb = table(predict\_status = qda\_pred\_class,  
true\_status=test\_set$fastball)  
tb

## true\_status  
## predict\_status 0 1  
## 0 58 36  
## 1 34 72

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.65

# ROC plot for test set  
qda\_pred = predict(train\_qda\_fit, test\_set)  
qda\_pred\_post = qda\_pred$posterior[,2]  
pred = prediction(qda\_pred\_post, test\_set$fastball)  
perf = performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

## AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.6907206

## Classification tree

mod.tree = tree(fastball ~ stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set)  
summary(mod.tree)

##   
## Regression tree:  
## tree(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + home\_score + away\_score + runner\_on\_1 + runner\_on\_2 +   
## runner\_on\_3, data = train\_set)  
## Variables actually used in tree construction:  
## [1] "strikes" "inning" "balls"   
## Number of terminal nodes: 6   
## Residual mean deviance: 0.2035 = 161.3 / 793   
## Distribution of residuals:  
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## -0.8516 -0.4271 0.1484 0.0000 0.3467 0.7400

plot(mod.tree)  
text(mod.tree, pretty = 0)

Diagram

Description automatically generated

set.seed(123)  
cv.out = cv.tree(mod.tree, K = 6)  
  
plot(cv.out$size, cv.out$dev, type = "b")

Chart, line chart

Description automatically generated

prune.mod = prune.tree(mod.tree,  
best = cv.out$size[which.min(cv.out$dev)])  
plot(prune.mod)  
text(prune.mod, pretty = 0)

Chart

Description automatically generated with medium confidence

#### See how it predicts test data  
yhat.test = predict(prune.mod, newdata = test\_set)  
y.test = test\_set$fastball  
#mse for train set  
mean((y.test-yhat.test)^2)

## [1] 0.2331208

# Confusion matrix  
  
pred = predict(prune.mod, newdata = test\_set)  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_class\_tree = as.factor(ifelse(predict(prune.mod, test\_set)>optimal,1,0))  
 tb = table(pred = yhat.test\_class\_tree, true=test\_set$fastball)  
 tb

## true  
## pred 0 1  
## 0 73 57  
## 1 19 51

(tb[1,1] + tb[2,2])/sum(tb)

## [1] 0.62

#Plot ROC curve  
tree\_pred\_class = predict(prune.mod, test\_set)  
pred = prediction((tree\_pred\_class), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, line chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.6372283

## Bagging

p = ncol(train\_set) - 1  
set.seed(123)  
bag\_fit = randomForest(fastball~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, mtry = p, importance = TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer  
## unique values. Are you sure you want to do regression?

## Warning in randomForest.default(m, y, ...): invalid mtry: reset to within valid  
## range

pred = predict(bag\_fit, test\_set, type = "class")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_bag = as.factor(ifelse(predict(bag\_fit, test\_set, type = "class")>optimal,1,0))  
tb\_bag = table(pred = yhat.test\_bag,  
true = test\_set$fastball)  
tb\_bag

## true  
## pred 0 1  
## 0 42 23  
## 1 50 85

(tb\_bag[1,1] + tb\_bag[2,2])/sum(tb\_bag)

## [1] 0.635

#Plot ROC curve  
yhat.test\_bag = predict(bag\_fit, test\_set, type = "class")  
pred = prediction(as.numeric(yhat.test\_bag), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.6497585

#Which variables are most important  
importance(bag\_fit, type=2)

## IncNodePurity  
## stand 7.742259  
## balls 18.580214  
## strikes 30.954625  
## outs\_when\_up 15.839115  
## inning 30.970014  
## home\_score 22.783911  
## away\_score 18.937126  
## runner\_on\_1 8.155922  
## runner\_on\_2 6.043113  
## runner\_on\_3 3.454568

varImpPlot(bag\_fit, main = "Variable Importance (Bagging)")

Table

Description automatically generated

## Random Forest

p = ncol(train\_set) - 1  
set.seed(123)  
rf\_fit = randomForest(fastball ~ stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set,  
mtry = round(sqrt(p)), importance = TRUE)

## Warning in randomForest.default(m, y, ...): The response has five or fewer  
## unique values. Are you sure you want to do regression?

pred = predict(rf\_fit, test\_set, type = "class")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_rf = as.factor(ifelse(predict(rf\_fit, test\_set, type = "class")>optimal,1,0))  
tb\_rf = table(pred = yhat.test\_rf,  
true = test\_set$fastball)  
tb\_rf

## true  
## pred 0 1  
## 0 43 24  
## 1 49 84

(tb\_rf[1,1] + tb\_rf[2,2])/sum(tb\_rf)

## [1] 0.635

#Plot ROC curve  
yhat.test\_rf = predict(rf\_fit, test\_set, type = "class")  
pred = prediction(as.numeric(yhat.test\_rf), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.6640499

#Which variables are most important  
importance(rf\_fit, type=2)

## IncNodePurity  
## stand 6.303025  
## balls 17.563429  
## strikes 31.572042  
## outs\_when\_up 13.869866  
## inning 26.389285  
## home\_score 19.357902  
## away\_score 16.752153  
## runner\_on\_1 6.557679  
## runner\_on\_2 5.571960  
## runner\_on\_3 3.070403

varImpPlot(rf\_fit, main = "Variable Importance (Random Forest)")

Table

Description automatically generated

## Boosting

set.seed(123)  
boost\_fit = gbm(fastball~ stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , train\_set, n.trees = 100,  
shrinkage = 0.1, interaction.depth = 1,  
distribution = "bernoulli")  
  
  
pred = predict(boost\_fit, test\_set, type = "response")

## Using 100 trees...

optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_boost = as.factor(ifelse(predict(boost\_fit, test\_set, type = "response")>optimal,1,0))

## Using 100 trees...

tb\_boost = table(pred = yhat.test\_boost,  
true = test\_set$fastball)  
tb\_boost

## true  
## pred 0 1  
## 0 55 27  
## 1 37 81

(tb\_boost[1,1] + tb\_boost[2,2])/sum(tb\_boost)

## [1] 0.68

#Plot ROC curve  
yhat.test\_boost = predict(boost\_fit, test\_set, type = "response")

## Using 100 trees...

pred = prediction(as.numeric(yhat.test\_boost), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.7205616

#Which variables are most important  
  
summary(boost\_fit)

Chart

Description automatically generated

## var rel.inf  
## strikes strikes 52.753676  
## balls balls 16.484287  
## inning inning 10.569079  
## away\_score away\_score 5.058207  
## stand stand 4.597029  
## home\_score home\_score 4.353955  
## runner\_on\_1 runner\_on\_1 2.592056  
## runner\_on\_3 runner\_on\_3 1.368325  
## outs\_when\_up outs\_when\_up 1.113881  
## runner\_on\_2 runner\_on\_2 1.109507

## Boosting performs slightly better than bagging and random forest.

## Tune boosted Model

grid = expand.grid(  
n.trees\_vec = c(100, 200, 300, 400),  
shrinkage\_vec = c(0.2, 0.1, 0.06, 0.05, 0.04, .02, .01),  
interaction.depth\_vec = c(1, 2, 3),  
miss\_classification\_rate = NA,  
time = NA  
)  
  
  
### Commented this out because the rmd file wouldnt not knit. ######  
#set.seed(1)  
#for(i in 1:nrow(grid)){  
#time = system.time({  
#boost\_fit = gbm(fastball~ ., train\_set,  
#n.trees = grid$n.trees\_vec[i],  
#shrinkage = grid$shrinkage\_vec[i],  
#interaction.depth = grid$interaction.depth\_vec[i],  
#distribution = "bernoulli", cv.folds=5)  
  
#grid$miss\_classification\_rate[i] =  
#boost\_fit$cv.error[which.min(boost\_fit$cv.error)]  
#grid$time[i] = time[["elapsed"]]  
#}  
#)  
  
#}  
  
head(grid %>% arrange(miss\_classification\_rate))

## n.trees\_vec shrinkage\_vec interaction.depth\_vec miss\_classification\_rate time  
## 1 100 0.2 1 NA NA  
## 2 200 0.2 1 NA NA  
## 3 300 0.2 1 NA NA  
## 4 400 0.2 1 NA NA  
## 5 100 0.1 1 NA NA  
## 6 200 0.1 1 NA NA

## Tune Boosted Model  
  
set.seed(123)  
boost\_fit = gbm(fastball~ stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , train\_set, n.trees = 200,  
shrinkage = 0.04, interaction.depth = 2,  
distribution = "bernoulli")  
  
  
pred = predict(boost\_fit, test\_set, type = "response")

## Using 200 trees...

optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_boost = as.factor(ifelse(predict(boost\_fit, test\_set, type = "response")>optimal,1,0))

## Using 200 trees...

tb\_boost = table(pred = yhat.test\_boost,  
true = test\_set$fastball)  
tb\_boost

## true  
## pred 0 1  
## 0 62 34  
## 1 30 74

(tb\_boost[1,1] + tb\_boost[2,2])/sum(tb\_boost)

## [1] 0.68

#Plot ROC curve  
yhat.test\_boost = predict(boost\_fit, test\_set, type = "response")

## Using 200 trees...

pred = prediction(as.numeric(yhat.test\_boost), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.7074779

## SVM

# Linear  
set.seed(123)  
tune\_svm = tune(svm, fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "linear",  
ranges = list(cost = seq(.01, 10, length.out=10)))  
summary(tune\_svm)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 2.23  
##   
## - best performance: 0.2452484   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.01 0.2488288 0.04018481  
## 2 1.12 0.2452499 0.03616243  
## 3 2.23 0.2452484 0.03612139  
## 4 3.34 0.2452750 0.03614145  
## 5 4.45 0.2452760 0.03613286  
## 6 5.56 0.2452962 0.03613835  
## 7 6.67 0.2452715 0.03612557  
## 8 7.78 0.2452834 0.03613086  
## 9 8.89 0.2452876 0.03612474  
## 10 10.00 0.2452795 0.03613643

svm\_fit = svm(fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "linear", cost = 2.23,  
scale = FALSE)  
  
summary(svm\_fit)

##   
## Call:  
## svm(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + home\_score + away\_score + runner\_on\_1 + runner\_on\_2 +   
## runner\_on\_3, data = train\_set, kernel = "linear", cost = 2.23,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: linear   
## cost: 2.23   
## gamma: 0.09090909   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 584

#Predicting the test set  
pred = predict(svm\_fit, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_svm = as.factor(ifelse(predict(svm\_fit, test\_set, type = "response")>optimal,1,0))  
tb\_svm = table(pred = yhat.test\_svm, truth = test\_set$fastball)  
tb\_svm

## truth  
## pred 0 1  
## 0 61 34  
## 1 31 74

(tb\_boost[1,1] + tb\_boost[2,2])/sum(tb\_boost)

## [1] 0.68

#Plot ROC curve  
yhat.test\_svm = predict(svm\_fit, test\_set, type = "response")  
pred = prediction(as.numeric(yhat.test\_svm), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.7109501

# Radial  
set.seed(123)  
tune\_svm = tune(svm, fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "radial",  
ranges = list(cost = seq(.01, 10, length.out=10)))  
summary(tune\_svm)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 1.12  
##   
## - best performance: 0.2398083   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.01 0.3391164 0.06705224  
## 2 1.12 0.2398083 0.03255317  
## 3 2.23 0.2522085 0.03656717  
## 4 3.34 0.2570502 0.03638437  
## 5 4.45 0.2605187 0.03822182  
## 6 5.56 0.2639628 0.04023175  
## 7 6.67 0.2678500 0.04190360  
## 8 7.78 0.2720752 0.04268107  
## 9 8.89 0.2762337 0.04338643  
## 10 10.00 0.2803118 0.04476210

svm\_fit = svm(fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "radial", cost = 1.12,  
scale = FALSE)  
  
summary(svm\_fit)

##   
## Call:  
## svm(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + home\_score + away\_score + runner\_on\_1 + runner\_on\_2 +   
## runner\_on\_3, data = train\_set, kernel = "radial", cost = 1.12,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: radial   
## cost: 1.12   
## gamma: 0.09090909   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 620

#Predicting the test set  
pred = predict(svm\_fit, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_svm = as.factor(ifelse(predict(svm\_fit, test\_set, type = "response")>optimal,1,0))  
tb\_svm = table(pred = yhat.test\_svm, truth = test\_set$fastball)  
tb\_svm

## truth  
## pred 0 1  
## 0 41 21  
## 1 51 87

(tb\_boost[1,1] + tb\_boost[2,2])/sum(tb\_boost)

## [1] 0.68

#Plot ROC curve  
yhat.test\_svm = predict(svm\_fit, test\_set, type = "response")  
pred = prediction(as.numeric(yhat.test\_svm), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, line chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.6778382

#Polynomial  
  
set.seed(123)  
tune\_svm = tune(svm, fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "polynomial",  
ranges = list(cost = seq(.01, 10, length.out=10)))  
summary(tune\_svm)

##   
## Parameter tuning of 'svm':  
##   
## - sampling method: 10-fold cross validation   
##   
## - best parameters:  
## cost  
## 1.12  
##   
## - best performance: 0.2400414   
##   
## - Detailed performance results:  
## cost error dispersion  
## 1 0.01 0.3934810 0.07131131  
## 2 1.12 0.2400414 0.03270791  
## 3 2.23 0.2491385 0.04214047  
## 4 3.34 0.2566958 0.04504410  
## 5 4.45 0.2646345 0.04827098  
## 6 5.56 0.2719296 0.05157933  
## 7 6.67 0.2760023 0.05321714  
## 8 7.78 0.2796102 0.05372459  
## 9 8.89 0.2834359 0.05438313  
## 10 10.00 0.2869237 0.05588314

svm\_fit = svm(fastball ~stand+balls+strikes+outs\_when\_up+inning+home\_score+away\_score+runner\_on\_1+runner\_on\_2+runner\_on\_3 , data = train\_set, kernel = "polynomial", cost = 1.12,  
scale = FALSE)  
  
summary(svm\_fit)

##   
## Call:  
## svm(formula = fastball ~ stand + balls + strikes + outs\_when\_up +   
## inning + home\_score + away\_score + runner\_on\_1 + runner\_on\_2 +   
## runner\_on\_3, data = train\_set, kernel = "polynomial", cost = 1.12,   
## scale = FALSE)  
##   
##   
## Parameters:  
## SVM-Type: eps-regression   
## SVM-Kernel: polynomial   
## cost: 1.12   
## degree: 3   
## gamma: 0.09090909   
## coef.0: 0   
## epsilon: 0.1   
##   
##   
## Number of Support Vectors: 637

#Predicting the test set  
pred = predict(svm\_fit, test\_set, type="response")  
optimal = optimalCutoff(test\_set$fastball, pred)[1]  
yhat.test\_svm = as.factor(ifelse(predict(svm\_fit, test\_set, type = "response")>optimal,1,0))  
tb\_svm = table(pred = yhat.test\_svm, truth = test\_set$fastball)  
tb\_svm

## truth  
## pred 0 1  
## 0 47 23  
## 1 45 85

(tb\_svm[1,1] + tb\_svm[2,2])/sum(tb\_svm)

## [1] 0.66

#Plot ROC curve  
yhat.test\_svm = predict(svm\_fit, test\_set, type = "response")  
pred = prediction(as.numeric(yhat.test\_svm), test\_set$fastball)  
perf= performance(pred, "tpr", "fpr")  
plot(perf, main = "ROC Curve")  
abline(0, 1, lty=3)

Chart, line chart, scatter chart

Description automatically generated

#AUC value  
auc = as.numeric(performance(pred, "auc")@y.values)  
auc

## [1] 0.672806