Logistic Regression Modeling on Baseball Data

Jared Berman
Javiy Wang

9 December 2019

Introduction:

Baseball has been a popular sport in different cultures throughout time. In the United States the sport can be traced back to the 19th century where amateurs played with their own informal rules and homemade equipment. But in 1870, a split developed between professional and amateur ballplayers. By the beginning of the 20th century, most major cities in the United States on the east coast had a professional baseball team. The teams were divided into two leagues, the National and American leagues, and during the regular season a team played against other teams within its league. Baseball has also generated national heroes such as Babe Ruth and Jackie Robinson. Then in the 1950s professional baseball had expanded to the west coast of the United States. Professional baseball also began to expand to countries such as Japan, Cuba, and other Caribbean nations at this time as well. Being America's favorite pastime, predicting a valid success from the MLB dataset using logistic regression would be interesting.

Data Description:

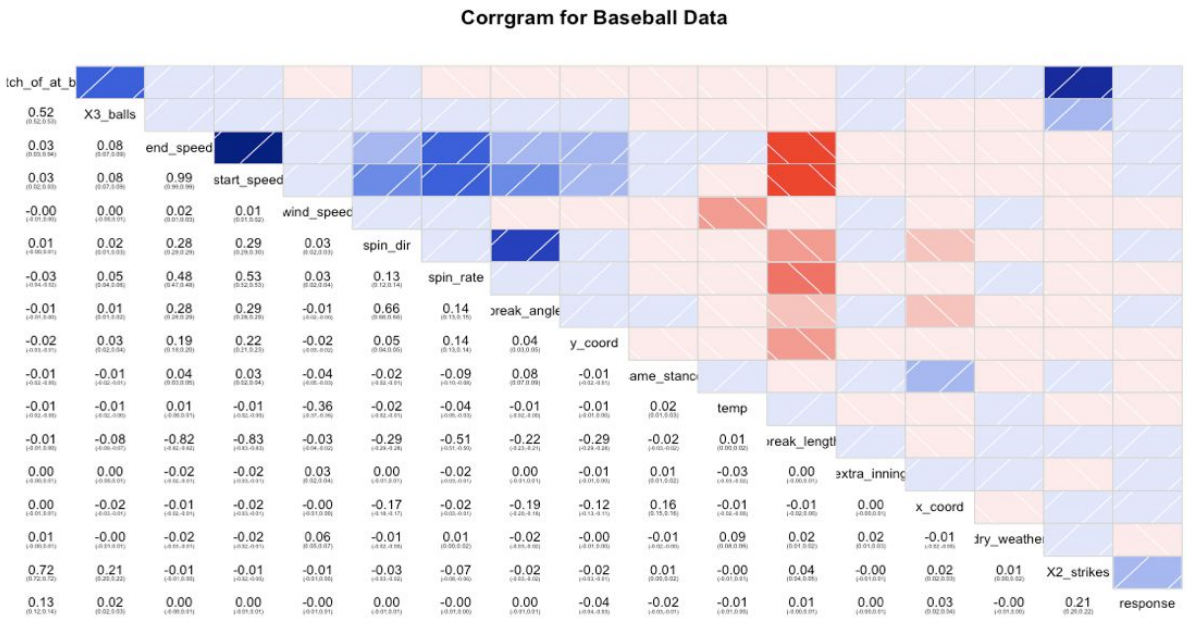
We got the baseball dataset from the kaggle website. The baseball dataset was clean originally but was further cleaned by us to get rid of unnecessary variables. Various columns were dropped, such as the name of the stadium, name of the umpire, etc. because they were irrelevant to the model or had no description to explain what information they contained. The dataset we would use for variable selection and model building included 17 predictor variables and a binary outcome variable. Our final list of predictor variables were as follows:

- start speed: the speed of the pitch when it is thrown
- end speed: the speed of the pitch when it reaches the plate
- X and Y coordinates of the ball: the left-right/up-down locations of the pitch over home plate
- spin rate: the pitches spin rate, measured in RPM
- spin direction: the direction in which the pitch is spinning, measured in degrees
- break angle: the direction in which the pitch curved
- break length: the magnitude in which the pitch curved
- pitch at-bat: the current pitch number of the current player's official turn at-bat
- pitch type: the type of pitch the pitcher throws
- 3 balls: whether or not there are currently three balls
- 2 strikes: whether or not there are currently two strikes
- same stance: whether the pitcher and batter both have the same orientation (both right or both left)
- Extra-inning: if pitch occurred during overtime

- Temperature: the temperature during the game
- wind speed: the speed of the wind during the game
- dry weather: whether the weather was dry or not

Our response variable is defined as success, coded 1, and not a success, coded 0. For the response variable, success is a valid success for the batter's team. This does not necessarily mean the bat coming into contact with the ball, we more so designated a success as something that would be beneficial to the team at bat. In other words, a success could be, for example, when the batter gets four balls and walks to first base while a previous batter was already at first base, allowing this batter to move on to second base. The response variable was created by us because it was not part of the original dataset. A loop in Python was written that went through each row and looked at a pitch to categorize it as either a success (1) or not a success (0). A CSV was created to build a model in R. 3 balls, 2 strikes, same stance, extra-inning, and dry weather are also all binary variables. Pitch type was qualitative and the rest of the predictor variables were quantitative variables.

When exploring the variables as part of our exploratory analysis, we decided to graph the correlation between our response variable and the 17 predictor variables we chose out of the variables initially available to us with this dataset. The graph is presented as follows:

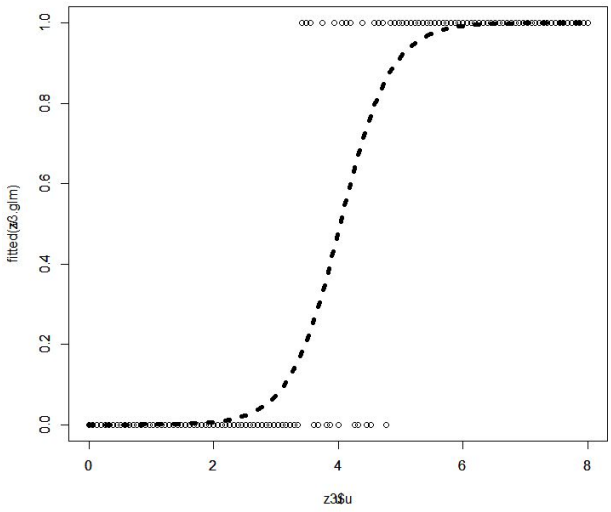


This correlation graph tells us that of the 17 predictor variables we chose, the ones with the highest correlation relative to all the correlations given with the response variable are X2_strikes, x_coord, and the pitch at-bat. We will consider this as we progress through our modelling processes, though this will not immediately make us use these three specific variables for our eventual model.

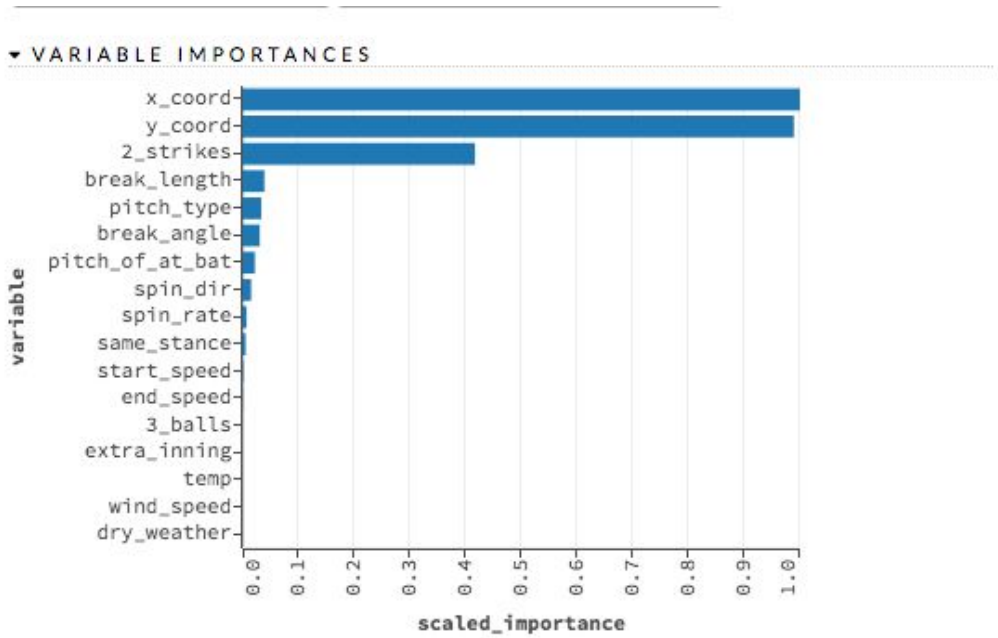
Statistical Methods:

When it comes to this dataset, we decided to use logistic regression to model it. Logistic regression is a statistical method for analyzing a dataset in which the outcome is binary, success and failure. Since we made the response variable binary, we decided to use logistic regression. We then created a training dataset and a testing dataset from our initial three million observations. Both datasets were a random sample of fifty thousand each.

We considered perhaps using a step-function with two levels, f(x) equaling one or zero, but decided against this because it felt too absolute and would not really allow for quantification of uncertainty. Uncertainty quantification is something that tries to determine how likely certain outcomes are, even if some aspects of a system or of a dataset are not precisely known. With logistic regression, only the meaningful variables are included in the model. It is also recommended that the model should have little to no multicollinearity (when one predictor variable can be linearly predicted from the others with a considerable degree of accuracy). Independent variables are also linearly related to the log odds in logistic regression, while the dependent variables have a Bernoulli distribution. Another thing to note is that the error terms in logistic regression do not have constant variance and are also not normally distributed. With linear regression it fits a line to the data, however logistic regression fits an 'S' shaped logistic function to the data, like this graph as an example:



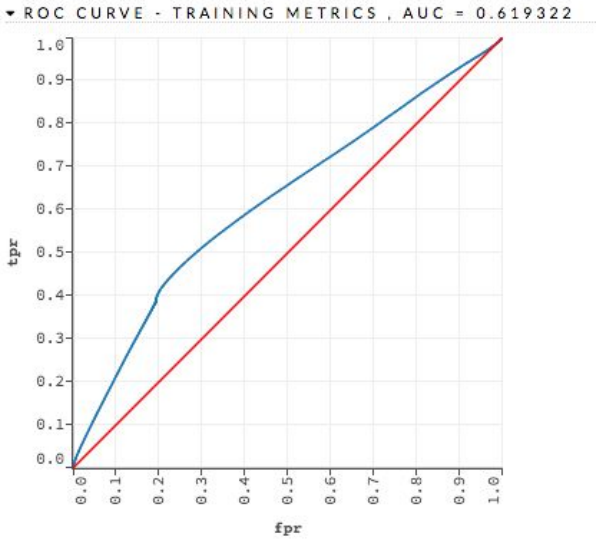
When it comes to determining if explanatory variables in our model are significant, we had one of two options: the Wald's Test and the Likelihood-Ratio Test. We ended up using the Likelihood-Ratio Test for this model. The Likelihood-Ratio Test is a hypothesis test that helps you choose the "best" model between two models, one of which is a subset of the other. To decide which of the 17 predictor variables would be the best to use for our model, we obtained this graph with some R code that utilized the Likelihood-Ratio Test along with a stepwise regression:



As one can see from the graph provided, the three best predictor variables for us would be x_coord, y_coord, and 2_strikes. This means that our binary response variable that we set as a valid success for the batter's team would be most influenced by the left-right location of the pitch over home plate, the up-down location of the pitch over home plate, and whether or not the batter currently has two strikes on him at the moment. This illustrates that where the pitcher throws the ball to the batter in reference to the strike zone normally found in baseball, as well as the stress of being up to bat with two strikes on you and so close to getting out, are what mostly influences whether a success occurs. Now with all this in mind, we find ourselves the ROC curve of the model to help us realize how good or bad it actually is in the grand scheme of things.

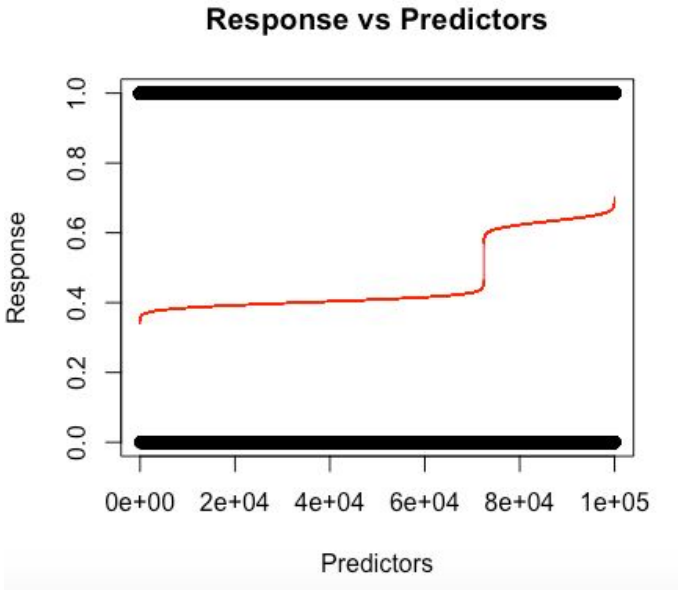
Results:

An ROC curve, or more accurately known as a Receiver Operating Characteristics curve, is found for a model when one needs to check or visualize the performance of the model. It is one of the most important evaluation metrics for checking any classification model's performance. When obtaining an ROC curve, we also receive a parameter known as the AUC, which stands for Area Under the Curve. With the ROC curve graph and the AUC, we are now able to tell how much the model is capable of distinguishing between classes. A high AUC means that the model will be pretty good at predicting failures as failures and successes as successes. The plot and output are given as follows:



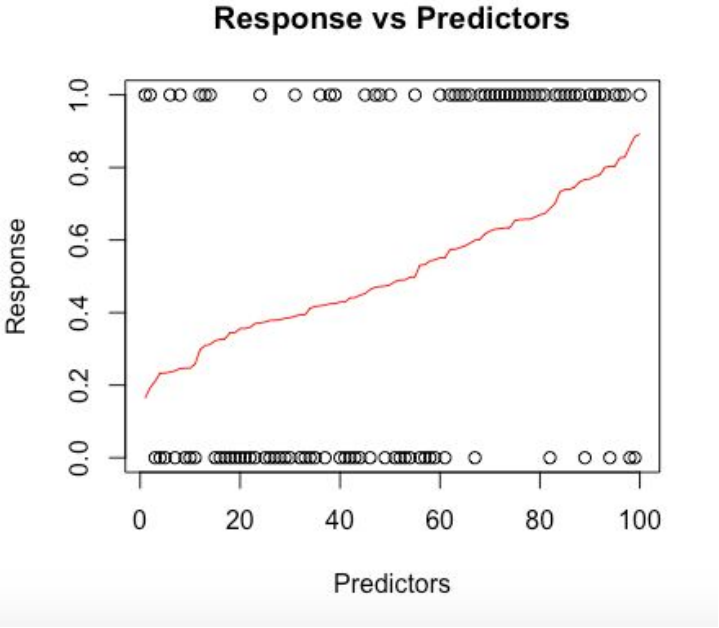
As one can see, our ROC curve is not the best it could possibly be. Typically one would want the curve to be much higher above the red line. Along with this, our AUC is not too high as well, at 0.6193, meaning that it is only going to roughly predict failures as failures and successes as successes with an accuracy of around 62 percent. Thus, our model is definitely not the best it could be, however, it is the best model that can come from a bad situation.

We then proceed to start creating logistic graphs. Presented below we have the plot of our logistic model with all 3 million observations in our dataset:



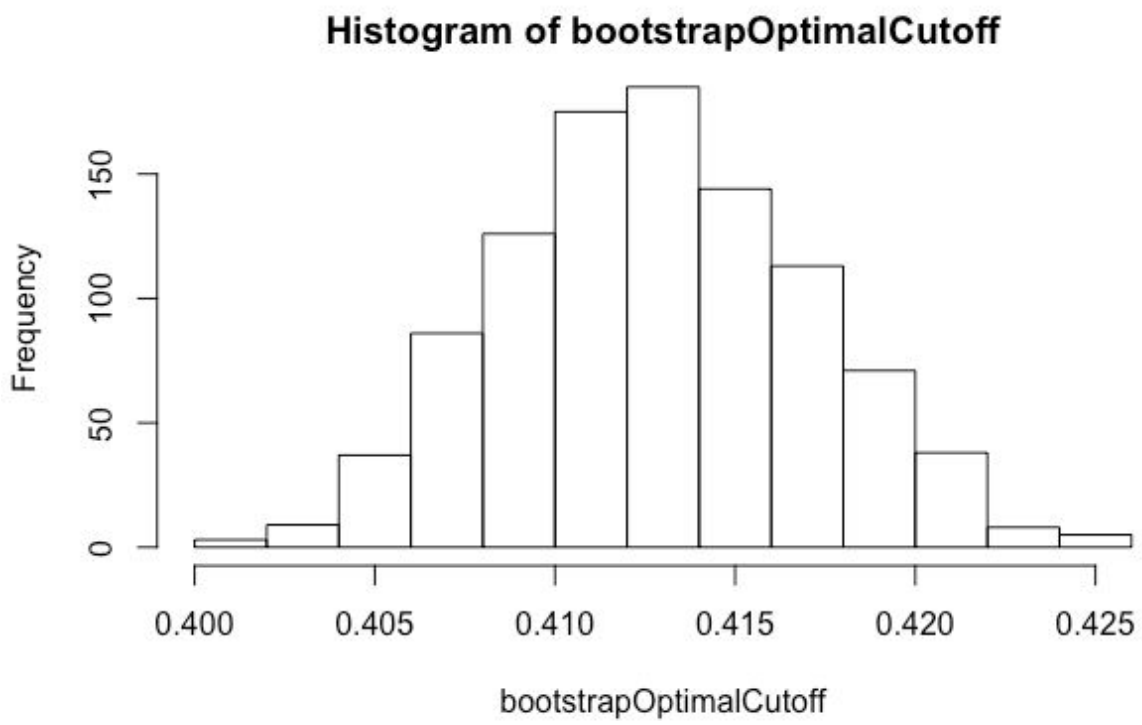
As one can see, the graph looks a bit off when attention is drawn to the red lines odd movement and shape across the graph. We assumed at this point that the reason for this odd plot is due to the sheer quantity of observations present in our baseball dataset. Thus we coded and

then plotted another graph, this time only utilizing the first one hundred observations we have in the dataset for better visualization. The plot respectively is presented as follows:



Now this graph looks a bit more like what one would normally see with a logistic plot. With the graph of the first one hundred observations now at hand, we can see why the initial plot looked so strange at first. It appears to be that the data is just very densely packed in certain parts, which is something bound to happen with such a large dataset, and the main reason our initial plot contained such an odd looking line.

With all this in mind, we will now start calculating our optimal cutoff probability. We will create a histogram to see what it would be. To do this, we will be using the bootstrap method. The bootstrap method is a resampling technique used to estimate statistics on a population by sampling the dataset with replacement. It is a method that can be used to estimate summary statistics such as the mean or standard deviation of data. For our purposes we will be utilizing the bootstrap method to obtain our optimal cutoff probability. To find this, we take re-samples from our sample and then use these re-samples to calculate estimates for our population statistic. Re-sampling will produce a distribution of mean or medians which will form a distribution that we then plot into a histogram. Our sample consists of one thousand observations of our dataset when using this method. The histogram is now presented as follows:



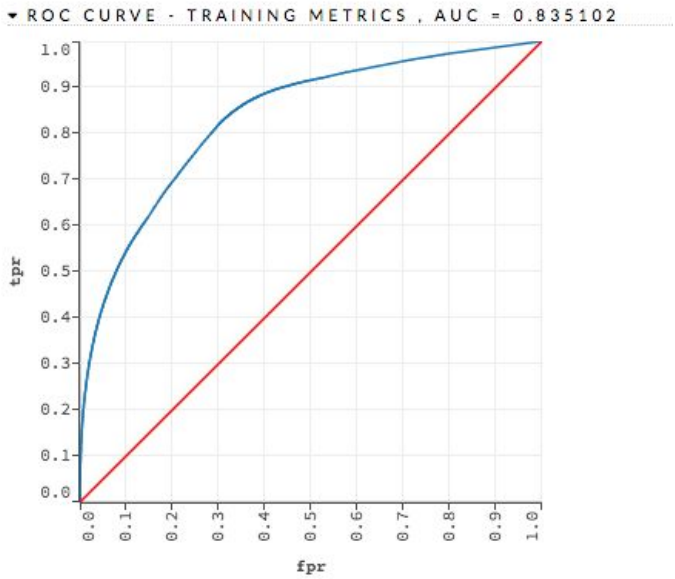
The histogram presented here is quite smooth and very normal, as one can see. This also illustrates to us that the optimal cutoff lies somewhere between .400 and .425.

Results Using GBM:

As stated before, the ROC curve is not the best it could be and our model could only predict a true success or true failure 62 percent of the time. However, by utilizing an R package called "h2o," we are able to use a pre-built in function called "gbm" rather than "glm". A GBM stands for a Gradient Boosted Machine and it is a technique for regression, for which it produces a prediction model by using multiple learning algorithms in order to obtain better predictive performance. The most common technique a GBM utilizes are decision trees, which are yes or no questions, and it is used to classify the data. Once one decision tree is used, GBM's find outliers and continue using better decision trees in order to obtain an even better predictor model than the first decision tree. It is also not uncommon for a GBM to make thousands of decision trees in order to obtain the best model. Essentially, we utilized a gradient boosted machine and we are able to obtain a better ROC curve with an AUC score of 0.8351. As a result, this new model was able to predict a true success or true failure 84 percent of the time as seen in the image below.

Now one might have two questions: First, if the GBM's continuously use decision trees until the AUC is maximized, why does the AUC not approach 1? Second, why did our group not use this technique from the beginning in order to obtain the best model? Well to answer the first question one must know that data will never be perfect and that there will always be outliers even with the best predictive model. GBM's keep improving on decision trees in order to increase our AUC. However, after each consecutive tree, there is a smaller increase for our AUC,

it converges to a certain number before ever reaching 1. Now why did we not use this method from the start? It comes down to the disadvantages of using GBM's. They are computationally expensive, less interpretive, and can cause overfitting. As stated before GBM's require thousands of decision trees, which can be time and memory exhaustive. GBM's are less interpretable for the fact that it is running thousands of decision trees and we do not get to see each decision our GBM makes in order to make the more predictive model. Therefore, at the end, our group used the GBM model in our to double check and make sure that its predictor variables matched with the predictor variables we obtained using stepwise regression.



Conclusion:

In conclusion, by running a logistic regression model with our 17 predictor variables, we found, through our ROC curve, that our AUC was .619322. We determined that our relationship regarding our predictors and response variable was discrete-binary, which supported our choice of a logistic regression model. The model performance is shown above and in the end we found our best predictors to be the X and Y coordinates of the ball as well as 2 strikes. The X and Y coordinate variables were the top two predictors of the three, with 2 strikes following as our third. The first two align with what one would naturally assume to be the case-how far off a pitch is from the plate will impact the result of the next play in a baseball game. Along with this, 2 strikes made sense due to the stress and anxiety a baseball player at bat would have if they were one strike away from getting an out.

In the end, there was a number of other variables we decided not to even consider due to either the dataset not explaining what exactly those variables were, as well as variables we felt were superfluous like the name of the stadium a current game is being played in.

As one can easily tell, our model was not the best. We believe that a problem with our idea is the way we ended up classifying what was a success and what was a failure. We were too ambitious, and after all our trials and tribulations we believe that we probably should have only

looked at cases where the batter actually swung. But, we knew right away that, with our particular dataset, a nonlinear model would be a good idea.

Code Appendix:

Code For GBM Model And Most of The Graphs

```
library(h2o)
h2o.init(nthreads = -1)
path = "/Users/JaviyWang/Desktop/Importing Data In R/statProject.csv" newpitches =
h2o.importFile(path)
myY = "response"
myX = setdiff(names(newpitches), myY)
newpitches[,myY] = as.factor(newpitches[,myY])
gbm_m = h2o.gbm(x = myX, y = myY, training_frame = newpitches, nfolds = 5)
```

Code For Both Logistic Plots

```
pitches <- import('~/Desktop/Importing Data In R/statProject.csv')
##new data set with the first 100,000 rows
newpitches = pitches[1:100,]
actual = newpitches$response
glm_model = glm(response~x_coord+y_coord+newpitches$`2_strikes`, data = newpitches,
family = "binomial")
# glm_model = glm(response~x_coord+y_coord+`2_strikes`, data = newpitches, family =
"binomial")
# Predict with all predictors
pred = predict(object = glm_model, newdata = newpitches, type = "response")
new_df = data.frame(pred = pred, actual = actual)
new_df = new_df[order(new_df$pred),]
plot(x=1:length(new_df$pred), y = new_df$actual, main = "Response vs Predictors", xlab =
'Predictors', ylab = 'Response')
lines(x=1:length(new_df$pred),y=new_df$pred, "l", col="red")
```

Code/Algorithm For Our Stepwise Regression

```
usedPredictors <- c()
currentPredString = '1'
for(i in 1:17){
  currentMisClassError <- 1
  nextBestPred = ''
  for(predictor in setdiff(setdiff(names(newpitches), 'response'), usedPredictors)){
    workingPredString = paste(currentPredString, predictor, sep = ' + ')
    formula = paste('response ~ ', workingPredString)
    mod <- glm(formula, data = newpitches, family = binomial(link = 'logit'))
    predicted <- plogis(predict(mod, testDf))
    optCutOff <- optimalCutoff(testDf$response, predicted)[1]
    MCE <- misClassError(testDf$response, predicted, threshold = optCutOff)
```

```
    if(MCE < currentMisClassError){
      nextBestPred = predictor
      currentMisClassError = MCE
    }
  }
  thing1 = paste('response ~ ', currentPredString)
  thing2 = paste(thing1, nextBestPred, sep = ' + ')
  oldMod <- glm(thing1, data = newpitches, family = binomial(link = 'logit'))
  newMod <- glm(thing2, data = newpitches, family = binomial(link = 'logit'))
  if(anova(oldMod, newMod, test = 'LRT')[2,5] >= .05){
    print('anova broke it')
    break
  }
  currentPredString <- paste(currentPredString, nextBestPred, sep = ' + ')
  usedPredictors = append(usedPredictors, nextBestPred)
  print(i)
}
print(currentPredString)
```

Code/Algorithm For Our Bootstrap Procedures

```
bootstrapOptimalCutoff <- c()
for(i in 1:1000){
  tempDf <- df[sample(nrow(df), 50000, replace = TRUE), ]
  tempMod <- glm(response ~ X2_strikes + y_coord + x_coord, data = tempDf, family =
binomial(link = 'logit'))
  tempPredicted <- plogis(predict(tempMod, testDf))
  bootstrapOptimalCutoff <- append(bootstrapOptimalCutoff, optimalCutoff(testDf$response,
tempPredicted)[1])
}
```