Predicting Home runs

Ryan Cope
rlcope@wisc.edu

Samuel Gemini
gemini@wisc.edu

Dominic Solberg
dmsolbergr@wisc.edu

December 6, 2021

Abstract

For our project, we investigated at-bat outcomes from the 2020 Major League Baseball (MLB) season if we could come up with a good model for predicting if an at-bat would end in a home run based off of several variables such as pitch type, pitch location, pitch speed, launch angle, launch speed, and we created our own variable which is whether the batter and pitcher were the same hand. After conducting and running several tests and models, we ultimately found that the only parameters that had any predictive power were launch angle and launch speed.

We needed to have two measures of success, one being overall accuracy (accuracy of non home runs as well as home runs) and the other being accuracy of just at bats that ended in home runs. We tried multiple different predictive models, such as KNN, decision tree with a grid search, and XGBoost. The model with the best predictive accuracy was XGBoost. Regarding the overall accuracy, it had a training accuracy of 99.729 percent and test accuracy of 97.165 percent. Regarding the home run accuracy, it correctly guessed 270 out of 395 home runs, which yields an accuracy of 68.354 percent.

1 Introduction

Every year, there are more than 2,400 Major League Baseball MLB games played every year (except for 2020 because of COVID). In those games, a lot of home runs are hit. Home runs are one of the most exciting parts of baseball, but are there patterns associated with these home runs? Given a data set with variables such as pitch count, pitch location, type of pitch, pitch velocity, hit speed, launch angle, etc,. we would like to see if knowing this information helps in predicting whether the batter will hit a home run or not.

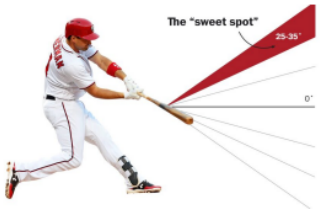

Figure 1: [4]Example illustrating launch angle and the optimal angle for home runs

There have been several studies about launch angle (angle that the baseball leaves the bat) and hitting success, namely home runs. Hitters have noticed an increase in the amount of home runs they hit as well as average once they started trying to hit the ball in the air more. [2]

There are factors other than launch angle at play, for example, people who play baseball know that when the count is 3 balls and 1 strike, do not throw a fastball right down the middle of the plate. The odds are much higher that a home run will be hit. We would like apply this knowledge to see if certain counts, pitch types, and pitch locations that can help us predict whether an at bat ends in a home run or not.

Figure 2: [3]Example demonstrates what pitch location is in reference to the strike zone

2 Related Work

Baseball is the sports analytics Mecca. MLB uses more advanced analytics than any other American sports league; front offices are constantly using models to project how players are going to perform in the coming seasons and to determine their value. Educated baseball analysts no longer use traditional metrics such as Batting Average and Runs Batted In (RBIs); rather, they use advanced statistics that take into account quality of opponent and ballpark characteristics, such as On Base Percentage Plus Slugging Percentage plus (OPS+), weighted Runs Created plus (wRC+), and Wins Above Replacement (WAR).

A good example of the strong and increasing presence of data analytics in baseball is the recent cheating scandal in the Houston Astros' organization. The Astros had cameras placed in center field to take video of pitch signs given by catchers. They used machine learning algorithms to decode the secretive pitch signs with astonishing quickness.

Many data scientists and statisticians have done research and developed machine learning algorithms focusing on baseball analytics. One such research paper, written by Alexander Gow, discusses a project that used machine learning to predict a player's Major League success based on Minor League performance. This statistical analysis used three different tests to determine if a player's MLB success could be predicted using his Minor League statistics.

The first test was a simple linear regression model using scikit learn to test if Minor League wRC+ was a good predictor of MLB success. According to Gow, "The result was a value of 0.05164, meaning that wRC+ only explains about 5 percent of the outcome. With the best possible value being 1, a simple linear regression based upon MILB wRC+ statistics is not a good predictor of future MLB success"[1].

The second test conducted by Gow was a multivariable linear regression model using a decision tree algorithm with XGboost. The prediction accuracy of this model was only about twelve percent, and it was determined that "the model's predictions are not effective"[1].

Lastly, Gow attempted to create a deep neural network using the keras library in Python: "After building and training the neural network, the training accuracy was 0.8800 and the training loss was 0.3040. Furthermore, my testing accuracy was 0.8299 and testing loss was 0.5138 (Figure 3, Figure 4). Clearly neural network predictions are far more accurate than those made by the scikit learn linear and XGBoost models"[1].

To state the obvious, the work done by Alexander Gow is similar to this project because it is focused on baseball analytics, specifically MLB. It is attempting to predict MLB outcomes (in this case, the performance of upcoming Minor League players) using advanced baseball analytics and predictive machine learning models. While it is not predicting the exact same aspect of baseball as this project, the general principles are the same; this project attempts to use MLB data to predict an outcome, i.e. whether a home wun was hit. In addition, some of the methods used in Gow's work are very similar to those used in this project. For example, Gow uses a gradient boosted decision tree algorithm to create a multivariable linear regression model. Utilization of the decision tree method was very important to this project.

3 Proposed Method

Our initial thoughts about how to do this project was to make several different models. We thought ones that would work well are decision trees, k-nearest neighbors, and XGBoost models. These models provide different levels of complexity when predicting the labels for given data, with XGBoost being the most complex and k-nearest neighbors being the simplest. We also predict accuracy values will be higher in more complex models.

3.1 Decision Trees

In our case, a decision tree classifier is an algorithm which takes several feature variables as input, and creates a binary tree with nodes based on the values of each input variable. We optimized our decision tree using grid search, where we are looking for the optimal number splits and the optimal depth of the tree. We set the seed to be 1 for this model, as well as for all following models.

3.2 K-Nearest Neighbors

K-Nearest Neighbor algorithms follow a rather simple concept. Assume that we are looking at two different, numeric feature variables and fitting labels based on them. We can assign one variable to the x-axis of a plot, and another variable to the y-axis of the plot. We can give labels to new data points based on what the 'k' nearest neighbor data points are labeled as, where 'k' is some number that we can choose. The nearest neighbor data points are determined using the euclidean distance, which is calculated using the formula below. When looking at 'k', we decided to manually check for the best performing value.

3.3 XGBoost

XGBoost is an open-source implementation of gradient boosting. Gradient boosting is generally a method which combines 'weak learner' models into a 'strong learner' model. Multiple rounds of boosting are done to optimize a differentiable loss function of a weak learner, and this gives us an additive model consisting of multiple weak learners. Overall, this process can be summarized as the building of further decision trees based on the errors made in previous trees.

3.4 Possible Issues

An issue that we thought of immediately, before we even created any models, was that most at bats in baseball do not end up in home runs. This being the case, our model could literally predict everything to be a non home run and have over 90 percent accuracy. Furthermore, we needed a way to be able to measure the accuracy of at bats that ended in home runs, as well non home runs because there is the obvious skew. To solve this problem, we decided to use a confusion matrix as that shows a 2x2 grid of the predictions a model makes. One box is the number of true positives (correctly predicted non home runs), false positives (home runs that were predicted as non home runs), true negatives (correctly predicted home runs), and false negatives (non home runs that were predicted as home runs). This way we can look at each sub category's accuracy.

Another issue we predicted was if the data was actually non-conclusive towards predicting home runs. In baseball, home runs are rather rare to begin with, so it could be possible that there is no discernible pattern to predict home runs with our given data.

4 Experiments

The K-Nearest Neighbors, Decision Tree, and XGBoost models were all tested similarly to evaluate how well they can predict what a hit baseball will become. We used the mlxtend and sci-kit learn packages to create our models.

4.1 Dataset

For our data we searched Kaggle and found a dataset called 'Baseball'. This is a collection of all at bats from the 2020 season where the ball was put in play (no walks, strikeouts, hit by pitches, etc.). The unfortunate thing about the 2020 season was that due to COVID-19, the normal 162 game season for each team had to be reduced to only 60 games because of the late start in the season. This obviously leads to a much smaller sample size, however, it was still sufficient since there are a lot of at bats per game.

The dataset included the following parameters:
Bip ID: unique identifier of ball in play
Game Date: date of game (YYYY-MM-DD)
Home Team: home team abbreviation
Away Team: away team abbreviation
Batter Team: batter's team abbreviation
Batter Name: batter's name
Pitcher Name: pitcher's name
Batter ID: batter's unique identifier
Pitcher ID: pitcher's unique identifier
isbatterlefty: binary encoding of left-handed batters
ispitcherlefty: binary encoding of left-handed pitchers

BB Type: batted ball type classification

Bearing: horizontal direction classification of ball leaving the bat (i.e. 'left' ball is traveling to the left side of the field)

pitch name: name of pitch type thrown

park: unique identifier of park venue

inning: inning number within game

outswhenup: current number of outs

balls: current number of balls

strikes: current number of strikes

plate x: ball position left(-) or right(+) of center plate (feet)

plate z: ball position above home plate (feet)

pitch mph: speed of pitched ball (miles per hour)

launch speed: speed of ball leaving the bat (miles per hour)

launch angle: vertical angle of ball leaving the bat (degrees relative to horizontal)

ishomerun: binary encoding of home runs

4.2 Software

All coding was done in Python 3.8.8 in Jupyter notebook. We used the following Python libraries: numpy and pandas for data manipulation, matplotlib for general plotting, sklearn, mlxtend and xgboost for building the machine learning models.

5 Results and Discussion

5.1 Feature Variables

We initially wanted to check if we could get any mileage out of just using pitch location to predict home runs. Our findings were not surprising, we did not even need to create a model to determine that pitch location alone would have minimal, if any, predictive power. All that had to be done was look at a scatter plot of the pitch location with the color of the point indicating whether it was a home run or not. There were so many overlapping points that there could not possibly be any predictive power just based on pitch location. To describe it in a different way, non-home runs were totally eclipsing home runs. Figure 3 shows the scatter plot previously described.

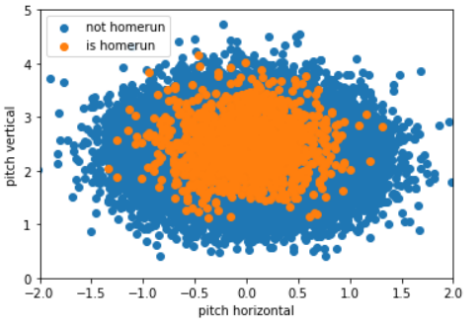

Figure 3: Scatter plot of pitch-locations: an orange point indicates a home run; a blue point indicates a non-home-run. Virtually every orange home-run point is overlapping with a blue non-home-run point.

Next, we thought it might be useful to see if the pitch location along with the pitch type would be useful in addition to a new binary parameter we created which was 'same hand players'. In baseball, it is a known fact that when the pitcher and batter are of opposite hand dominance (i.e. a righty pitcher against a lefty hitter), it is generally easier for the batter to have a successful at-bat and to hit for power. So it is logical to think that the binary parameter "same hand players" could be helpful for predicting home runs.

The use of the additional predictor variables mentioned above did not improve the predictive power very much. Regardless of the method used, the model predicted very few home runs, and those few home run predictions were very inaccurate. Some of the methods used didn't predict any home runs, at all. One of the methods that gave this result was the decision tree; others were k-nearest-neighbor methods where k was relatively large – around 6 or higher. In fact, our first model, using a grid search cross validation decision tree predicted 0 home runs. But since most at bats do not end in home runs, we still had an accuracy slightly north of 94 percent (94.08).

Most combinations of predictor variables that we tried did not yield a model with significant, or any, predictive power, which is intuitive. For example, as mentioned before, there are so many pitches in every part of the zone, and the vast majority of at bats do not end in home runs, but some do end in home runs even when the parameter

values are the same, making it nearly impossible to make a distinction. Adding pitch type, while slightly more specific, still runs into the same issue because there could be 100 instances of a 4 seam fastball in the same spot but only one of them will be a home run. That is just how baseball is. Pitch speed would also not help because it is highly correlated with pitch type. Other variables, such as balls, strikes, pitcher ID, and player ID, were added to the model, and no notable improvements to prediction accuracy were made.

Eventually, the variables launch angle and launch speed were added, and the model was run again. There was one issue that we ran into, and that was missing values for some of the launch speeds. Our thought is that radar gun must have failed or someone else failed to pick up the speed. Nevertheless, we had to remove all rows with missing values. This still left us with a fairly large sample size, which was reduced from 46244 data points to 25866. Launch angle and launch speed vastly improved the predictive power of the model. The prediction accuracy improved by over 2 percent for all methods tested: decision tree, k-nearest-neighbors, and XGBoost. This suggests that launch angle and launch speed have a substantial impact on predicting home runs.

One final predictor variable was added to see if it would improve the prediction accuracy even further. This variable is called 'bearing'; it indicates the direction that the ball is hit. This is a valuable piece of information because the distance a ball must be hit to get over the fence is different for different areas of the ballpark. This means that the required launch angle and launch speed for the ball to go over the fence is different depending on the direction that the ball is hit. The addition of this feature only slightly increased the performance of the models, by less than a 1 percent increase.

After these final changes were made to the feature set, we were able to create our final models and generate our accuracy values from those.

5.2 Model Results

Considering specifically the K-Nearest-Neighbors model, we found that the value of k which produced the highest test accuracy was k = 4. For k = 4, the training accuracy was 97.802 percent, while the test accuracy was 97.088. These values are very close, so we can assume

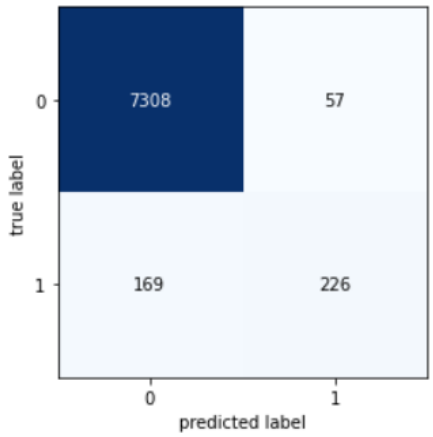

Figure 4: Confusion matrix for the K-Nearest-Neighbors model with k = 4. 0 is the label for non home runs and 1 is the label for home runs.

that the model is predicting accurately based off of the training data. When looking only at home run predictions, there were 226 correctly predicted home runs and a total of 395 predicted home runs, as you can see from Figure 4, which shows the confusion matrix for this K-Nearest-Neighbors model. This gives a home run prediction accuracy of 57.215 percent.

For the best performing decision tree, with a depth of 6 and 3 splits, the training accuracy was 97.509 percent, and the test accuracy was 97.023 percent. For the home run accuracy using the test data set, there were 248 correctly predicted home runs and 148 false home run predictions, for a total of 395 predicted home runs. These numbers are shown in Figure 5, the confusion matrix for the decision tree model. This predicted home run accuracy was 62.785 percent.

The XGBoost model had a training set accuracy of 99.729 percent and a test set accuracy of 97.165 percent. There were 270 correctly predicted home runs, 125 false home runs, and 395 total home run predictions. These numbers are shown in Figure 6, the confusion matrix for the XGBoost model. The home run prediction accuracy was 68.354 percent.

From Table 5.3, it clear that the overall accuracy is very high. All models had accuracies over 97 percent, with

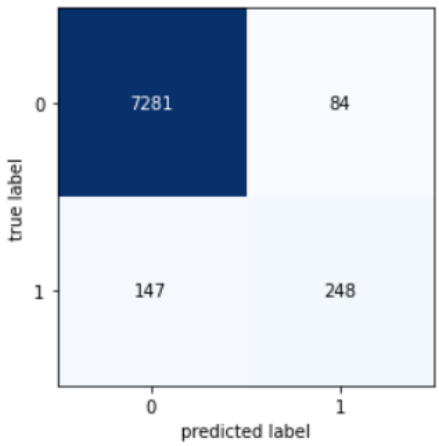

Figure 5: Confusion matrix for the decision tree model. 0 is the label for non home runs and 1 is the label for home runs.

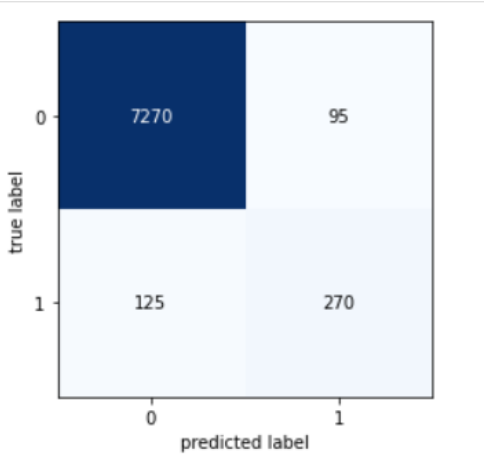

Figure 6: Confusion matrix for the XGBoost model. 0 is the label for non home runs and 1 is the label for home runs.

the XGBoost model having the highest accuracy at 97.165 percent. K-Nearest Neighbors and the decision tree models were both close in accuracy, and while we expected the decision tree to perform better than the KNN model, the opposite was actually found. This overall accuracy value means that for approximately 97 percent of hit baseballs, any of the models will correctly predict if the ball will be a home run or a regular hit.

5.3 Accuracy Comparison

Overall accuracy is a decent metric to determine the models' accuracy, although we are more interested in how well it can predict home runs specifically, because they are statistically much rarer. This home run accuracy is summarized for the test data in Table 5.3. This data covers a larger spread, where the K-Nearest Neighbors model performed the worst with an accuracy of 57.215 percent, the XGBoost model performed the best with a 68.354 percent accuracy, and the decision tree fell in the middle with an accuracy of 62.745. These results actually confirmed our earlier prediction that XGBoost would perform the best and that KNN would perform the worst. It was found that the XGBoost model predicted 44 more correct home runs than the KNN model and 22 more correct home runs than the decision tree. We decided to run Z-score tests for differences between proportions, where the proportions are the accuracy values, and found that there is a statistically significant difference when comparing all accuracies against each other. This allows us to conclude that the XGBoost model was the best at correctly predicting home runs.

Method	Overall Accuracy
Decision Tree	97.023%
KNN	97.088 %
XGBoost	97.165 %

Table 1: This table shows the accuracy, for predicting if a hit baseball will be a home run or not, for each of our models on the testing subset of data.

6 Conclusions

Essentially, what we found is that we cannot get any predictive power from 'pre hit' parameters. We are calling parameters 'pre hit' if they can be found before the ball is hit, examples of these being pitch location, pitch type,

Method	Home Run Accuracy
Decision Tree	62.745%
KNN	57.215 %
XGBoost	68.354 %

Table 2: This table shows the accuracy of predicted home runs. Accuracy is calculated using number of true home runs divided by total number of home runs predicted. This data is from each of our models using the testing subset of data.

pitch speed, handedness, etc. However, we did find that launch angle and launch speed (post hit) are adequate predictors for determining when a home run will be hit. Our findings ultimately will not help pitching or betting strategy because the only parameters that have any predictive power are the ones that happen after the fact.

7 Acknowledgements

We gratefully thank our advisor Dr. Sebastian Raschka for all of his guidance and support. His assistance with code problems and supplying resources for us to find datasets was extremely helpful.

To that point, Kaggle is where we found our data source. Our dataset was supplied by 'jcraggy', a Kaggle community member.

8 Contributions

For our modeling methods, we all participated in the idea phase of making models and data visualizations, but Ryan did most of the actual coding for our model. Dominic found the dataset and initially read in all the data. Sam looked for and found related work as well as pictures for visual aids.

For the paper writing, everyone was responsible for different sections, but were at liberty to edit and revise any section we wanted. Dominic's specific areas of the paper were the Abstract, Introduction, Acknowledgement, and some of the results/discussion/experiment sections. Ryan detailed the Proposed Method and Experiment sections as well as most of the results/discussion. Sam focused on the related work and data visualization/visual aids and references. Ultimately, we all did a little bit of everything because everyone had to proof read the work and paper and add suggestions/comments.

References

[1] A. Gow. Using machine learning to predict mlb success based on milb performance. *IPHS 300: Artificial Intelligence for the Humanities: Text, Image, and Sound*, (18), 2019.

[2] D. Kagan. What is the best launch angle to hit a home run? *The Physics Teacher*, 48(4):250, 2010.

[3] S. Kelley. *Pitch Track*. Apr 2018.

[4] D. Steinin. Why mlb hitters are suddenly obsessed with launch angles. 2017. accessed 10/10/2021.