Prediction of Home Runs Based on Batted Ball Events Using Random Forest and Naïve Bayes Models

Nicolas Kozachuk[1]

[1] Lehigh University, Bethlehem, PA, USA
ngk324@lehigh.edu

Abstract
In 2015 the MLB implemented Statcast, which is a radar tracking system, in all of its ballparks. The goal of Statcast is to measure the various metrics of baseball games, as baseball is a highly data driven sport. Statcast data is publicly available and has become essential in assessing player performance, as well as consumer engagement from game performances. This paper investigates leveraging Statcast data, along with ballpark dimensions, to predict if a batted ball is a home run by employing two distinct machine learning models, Random Forest and Naive Bayes. Through a comparative analysis of these models, significant insights into the crucial features influencing home run outcomes are discovered. These findings hold promise for player development strategies and performance evaluation methodologies. A performance comparison of the two models will also provide one with a further understanding of the advantages and disadvantages of each model..

Keywords: Random Forest, Naïve Bayes, Baseball, Statcast, Home Run, Exit Velocity, Launch Angle.

1 Introduction

Sabermetrics, or sometimes referred to as moneyball, is the empirical analysis of baseball statistics that measure in-game activity in order to determine player performance, as well as determine the optimal in-game actions that produce a win. The implementation of Statcast in all 30 MLB ballparks in 2015 has revolutionized all aspects of baseball by providing a vast array of baseball data to be used for sabermetric purposes. With advanced sabermetric statistics provided by Statcast, player evaluations and predictions, as well as in-game managerial decisions have been significantly improved. For example, by using Statcast data it was found that a pitcher increasing their pitch spin rate resulted in more success for the pitcher than throwing a pitch faster. Spin rate was not something that was able to be measured before Statcast and revolutionized the way pitchers develop and pitch in game as it is significantly easier to increase a pitches spin rate than it is to throw faster. Additionally, this also revolutionized the way teams assess pitchers to be drafted, as well as the development of pitchers. Looking at the hitting side of sabermetrics, it was found from Statcast that the launch angle of a batted ball was one of the most important factors in hitting a home run.

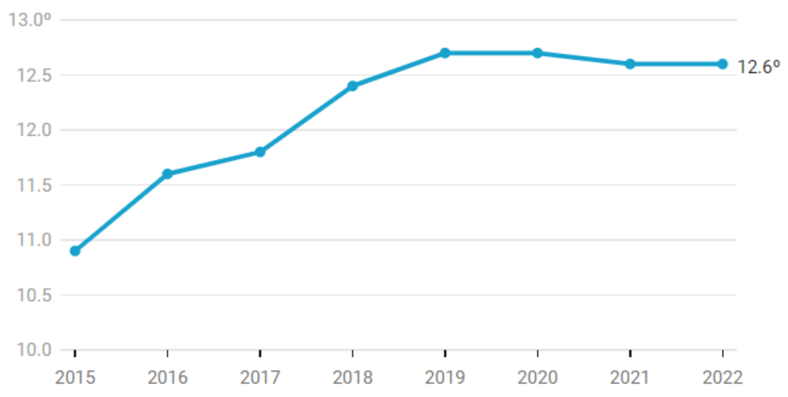


Fig. 1. Plot of MLB League Average Launch Angle from 2015 to 2022

It can be seen in Figure 1 in the Appendix, that the league average launch angle increased each year after Statcast was implemented, until the optimal launch angle was found. With league wide batting averages decreasing due to better pitching, hitting home runs to get an instant run scored in baseball has become more important than ever. This paper will investigate using various different Statcast data to predict if a batter ball is a home run, as well as determine the important features in hitting a home run.

2 Related Work

Few formal papers have been written for the assessment of batted ball outcomes from Statcast statistics, but some do exist. One paper that exists is “Predicting Batted Ball Outcomes in Major League Baseball”[1], which investigates determining the probability of a hit based on the batting statistics of exit velocity, which is the speed of the ball off the bat when being hit, as well as launch angle, which is the angle the ball flies off the bat from a swing. The paper found that the faster the exit velocity, the more likely a hit is, which logically makes sense because a ball hit with greater exit velocity gives fielders less time to react to get the ball and make an out. The paper also found that launch angle has a quadratic effect on the odds of getting a hit. This also makes sense as a batted ball with a low launch angle would result in a ground ball, which does not usually travel far and likely results in an out. A batted ball with a high launch angle often results in a pop up that gives fielders more time to get under the ball to make an out. Thus, there is a middle sweet spot for the launch angle for a batted ball resulting in a hit.

Another related paper “Forecasting Batter Performance Using Statcast Data in Major League”[2] was also found. This paper found that a player’s average exit velocity is correlated to a player’s slugging percentage. A player's slugging percentage is related to the number of home runs they hit as slugging percentage is defined as the total number of bases reached divided by the number of at bats. For example, a slugging percentage of 4.0 means the player averages four bases per at bat, which is a home run, and a slugging percentage of 0 means the player never reaches base. The results of both of these papers show that a batted ball's exit velocity and launch angle are important factors in determining if the batted ball is a hit or not. This paper will go into further detail in determining the effect of exit velocity and launch angle, as well as other statistics in determining if a batted ball is a home run.

3 Problem and Methodology

This paper provides an overview of the prediction of batted balls resulting in a home run based on various Statcast statistics. Additionally, this paper will investigate the important features in determining if a batted ball results in a home run or not. The dataset used for this problem is a collection of batted ball outcomes and related statistics for the 2020 season[3]. The dataset contains 25,862 instances of batted ball outcomes, with 24,545 batted ball instances belonging to the no home run class, and 1,317 batted ball instances belonging to the home run class. Additionally, park dimension data is also used as there is some variance in the size of each stadium, which has an effect on if a batted ball is a home run. In order to create predictions, a Random Forest machine learning model is implemented. Random Forest models allow for advanced feature importance analysis, which is greatly advantageous for the goal of determining the important statistics related to hitting a home run. Additionally, the Random Forest feature importance results are helpful in tuning the model by adding features using feature engineering determined from the feature importance. Due to the lowest percentage chance of home runs being hit given a batted ball, the classes of home run and not a home run are extremely disproportionate. In order to validate the features used and the results of the Random Forest model, a Naive Bayes model is also created in order to predict if a batted ball is a home run. 10-fold cross validation is performed on both models in order to determine the validity of the models. A performance comparison between each of the models is also performed in order to analyze the advantages and disadvantages of each model.

4 Background and Theory

4.1 Random Forest

The Random Forest machine learning algorithm used in this study to predict home run occurrences provides a viable solution to the problem due to its resilience to outliers, adaptability to various data structures, and its ability to provide feature importance. The Random Forest algorithm stands out for its capacity to manage complex feature sets, making it an ideal choice for finding patterns within the complex domain of baseball analytics. The random forest algorithm works by combining the predictions from numerous decision trees into a single output. Each decision tree within the forest operates as a single entity, composed of interconnected nodes representing distinct features and potential outcomes. Beginning with a root node, the tree branches into internal nodes, each representing a decision point based on specific feature values. Finally, the branches culminate in leaf nodes, which signify the predicted outcome. By iteratively constructing multiple decision trees with random subsets of features and combining the insights found with them, the random forest algorithm is able to find the optimal combination of factors contributing to predictive accuracy and creates predictions based on these factors. Additionally, the random forest algorithm's ability to output feature importance is essential in identifying the most influential features in determining if a batted ball is a home run. All in all, the random forest algorithm is used as it serves as a powerful tool for prediction given the data used, as well as due to its ability to uncover relationships within baseball data, which can provide valuable insights for player development strategies and performance assessment methodologies.

4.2 Naïve Bayes

The Naive Bayes algorithm employed alongside the random forest in this study, works in a fundamentally different way compared to the random forest. While the random forest is based on decision trees and aggregation of predictions, Naive Bayes relies on probabilistic reasoning and conditional probability. Naive Bayes is particularly advantageous for classification tasks due to its simplicity and efficiency. It works by calculating the probability of a given event or outcome based on the probabilities of related events or features. In the context of predicting home runs, Naive Bayes considers the probability of a batted ball being a home run given the observed features. Despite its simplicity, Naive Bayes assumes that features are conditionally independent, meaning that the presence of one feature does not affect the presence of another, which is a "naive" assumption that does not always hold true in real world scenarios. Despite this simplification, Naive Bayes often performs remarkably well, especially when the features are relatively independent or when there is limited training data available, which exists in this study due to the disproportionate class sizes. All in all, Naive Bayes is a probabilistic algorithm that calculates the likelihood of different outcomes based on observed features, making it a valuable tool for classification tasks such as predicting home runs in baseball. Its simplicity, efficiency, and effectiveness in certain scenarios, as well as its stark difference in how predictions are made, makes it a worthy complement to more complex algorithms like the random forest algorithm.

5 Implementation

5.1 Data Preprocessing

The implementation of the models first begins with data preprocessing. The dataset of batted balls results contain 24 features, and many of them are not needed. The features of game date, home/away team name, batters' team, pitchers' team, number of outs when batter is up, inning number, strike count, ball count, and the unique batted ball identification number are all removed since they have no effect on whether the batted ball is a home run or not. These features having no effect on a batted ball being a home run or not is known due to previously known knowledge of baseball. Additionally, the features of batter name and pitcher name are removed since the dataset also contains unique player and pitcher identification numbers, which is essentially the same thing. Next, the string features are encoding into integers corresponding to each string. This include the features of pitch type, such as fastball, curveball, etc., as well as the batted balls bearing, which is the horizontal direction classification of the ball leaving the bat.

Next, the stadium dimension dataset was read in and added to the models feature dataframe for each batted ball and the corresponding stadium the game took place in. All in all this results in the features used to be the right, center, and left field wall heights and distance from home plate, if the stadium is covered or not, if the batter is lefty or righty, if the pitcher is left or righty, the bearing of the ball, the pitcher ID, the batter ID, the pitch MPH, the pitched ball position relative to the center of home place plate, the pitch position relative to the distance from the ground, the exit velocity, and the launch angle.

5.2 Random Forest Model Training

The next step was to split the data into training and testing sets, which was done by splitting the data into 90% training data and 10% testing data. A random forest classifier model was then instantiated with the number of decision trees set to be 2,000. The random forest model was then trained using the training data defined and the classification labels, which is a binary value corresponding to if the batted ball was a home run or not. After training the model, predictions were made based on the test dataset, and the accuracy of the model was evaluated by comparing the predicted results of the testing set to the true results. A confusion matrix was then created for visualization purposes and the precision and recall were calculated for each class. Additionally, 10-fold cross validation was performed on the model using the original unsplit dataset features and class labels in order to access the model performance.

Model Tuning

After this, the model's feature importance was computed, outputted, and analyzed in order to tune the model and improve it. After analyzing the feature importance, it was seen that the features of exit velocity and launch angle were the most important features in determining if a batted ball is a home run. Feature engineering was performed by taking the square root, as well as the z-score of the exit velocity and launch angle feature and adding them to the original list of features in order to be used as features for future training. The z-score of these values were chosen as z-score has become a popular statistic in baseball. For example, this can be seen with the new popular advanced analytics of AVG+, ERA+ OBPS+, etc., which reflects the percentage above average(100) that a player is for the given statistic. It was also seen that the pitched ball position relative to the center of home plate and the pitch position relative to the distance from the ground were the next most important, so these values were cubed and also added as features. They were cubed in order to retain the negative values present in the pitched ball position relative to the center of home plate.

5.3 Naïve Bayes Model

Next, once confident in the features used and the random forest model, a Gaussian Naive Bayes model was implemented. The model was fitted using the training data and then used to make predictions on the testing dataset. Like for the random forest model, the accuracy from the results of the predictions from the naive bayes model were calculated, as well as a confusion matrix and the precision and recall values for each class. Additionally, 10-fold cross validation was also run for the naive bayes model in order to determine the validity of the model.

6 Results

Precision is a metric that measures the percentage of a class's predictions that are truly correct. Precision is defined as the ratio of true positives divided by the total number of true positives plus false positives. Recall is a metric that measures how often a machine learning model correctly identifies instances of a class from all the true instances of a class. Recall is defined as the number of true positives divided by the number of true positives plus false negatives.

6.1 Random Forest

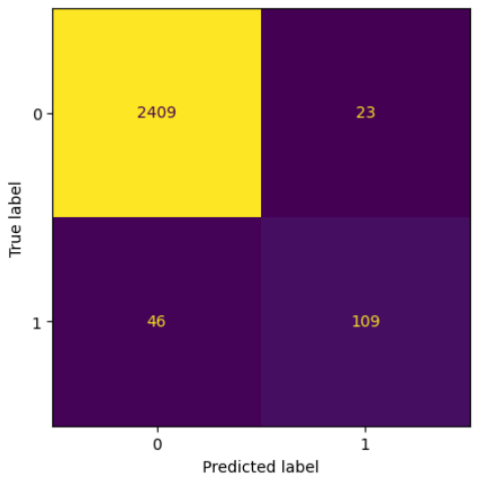
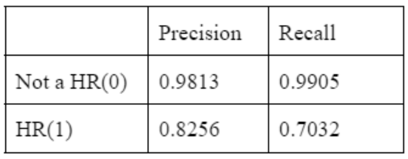


Fig. 2. Confusion Matrix for Random Forest Classifications Predictions: 0 = No Home Run and 1 = Home Run



	Precision	Recall
Not a HR(0)	0.9813	0.9905
HR(1)	0.8256	0.7032

Fig. 3. Precision and Recall Values for Random Forest Classification

One can see the results from the random forest prediction in the confusion matrix shown in Figure 2. One can also see the precision and recall values for each of the predicted classes in Figure 3. It can be seen from Figure 3 that the precision for the not a home run class is 0.9813, and relating this to the confusion matrix can be calculated as 2409 / (2409 + 46). A precision value of 0.9813 is very good and means that with respect to the number of not a home run samples, very few true home runs were classified as not a home run. The recall value for the not a home run class is 0.9905, which is also excellent. This value can be seen from the confusion matrix with the calculation of 2409 / (2409 + 23), and means that out of all of the predicted not a home run class data points, 99.05% of them were correctly classified as not a home run. Some of the misclassification could have possibly resulted from “robbed” home runs, where the batted ball is usually a home run as it is hit over the fence, except the fielder made an excellent play and caused the home run to result in an out.

It can be seen from Figure 3 that the precision for the home run class is 0.8256, and relating this to the confusion matrix can be calculated as 109 / (109 + 23). A precision value of 0.8256 is relatively good, but a deeper analysis shows that with respect to the size of the classes, a precision of 0.8256 is very good. This is due to the classes being extremely unbalanced as the test data contains 2432 samples of no home run data, but only 155 samples of home run data. Since the no home run class data is about 16 times larger than the home run class data, it makes sense that the precision is not very high for the home run class data. As can be seen by the recall of the no home run class, a 0.95% misclassification rate, which is extremely small, of the no run home class, results in 23 misclassified values, corresponds to about 15% of the entire home testing data class. For example, even if all of the 155 home run class testing data was successfully classified as a home run, it would only take 6.43% of the none home class testing data, which evaluates to 155 incorrect no home run class data predicted as a home run, to be misclassified as a home run to result in a recall of 0.5 for the home run class. To put it simply, even if the recall of the home run class is 1.0, and the recall of the no home run class was 0.9357, this would result in a precision of 0.5 for the home run class.

The recall value for the home run class is 0.7032, which is not a horrible result, but also not the best. This value can be seen from the confusion matrix with the calculation of 155 / (155 + 46), and means that out of all of the predicted home run class data points, 70.32% of them were correctly classified as a home run. Some of these misclassifications could be a result of inside the park home runs, where traditionally the hit not going over the fence would mean it is not a home run, but due to some obscure ball bounces or fielder positioning, the batter is able to get a home run without actually hitting the ball over the fence. A recall of 0.7032 is not the best result, but taking into consideration the size of the data, it is a viable result. The dataset contains a total of 25,862 batted ball instances, but only 1,317 of those instances are classified as home run outcomes. 1,317 is not a very large size of data with respect to training a machine learning model, especially for a random forest, so it makes sense the recall is not higher. With more home run class instances, the model would likely perform much better. Taking into account the size of the classes, a recall for the home run class of 0.7032 is actually very good.

The overall accuracy of the Random Forest classifier is very high at 97.33%. This accuracy can be quantified as the number of the number of correct predictions divided by the number of all predictions. It can also be thought of as the weighted average of the recalls of each class. To get a better understanding of the model's performance, a 10-fold cross validation was performed and yielded a value of 0.9735. This further confirms the validity of the model as it has very high accuracy, and shows that overfitting did not occur.

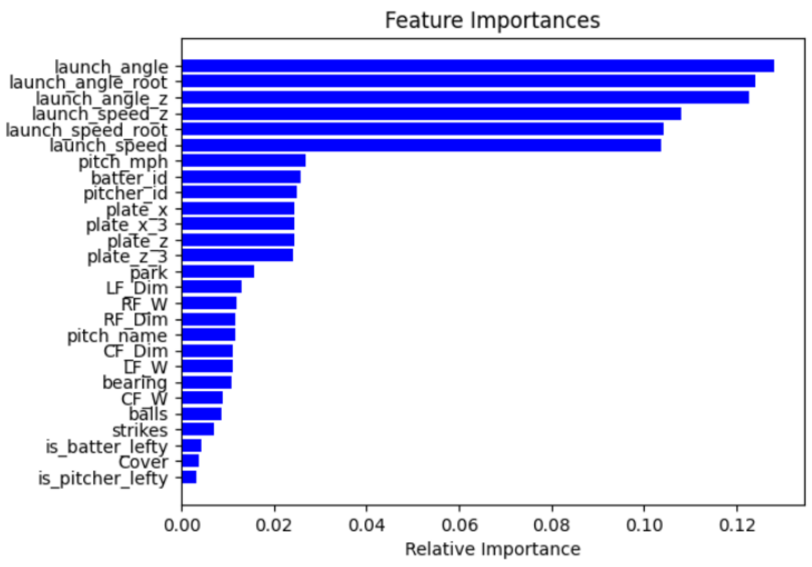


Fig. 4. Feature Importance from Tuned Random Forest Classifier Model

The feature importance generated from the Random Forest model can be seen above in Figure 4. It can be seen that the exit velocity(launch speed) and launch angle are the most important features by a large margin, as well as their corresponding features that were engineered. It can be seen that the next most important features are pitch speed and batter/pitcher ID. Performing feature engineering on the pitch speed was not necessary as this is also related to the exit velocity, which already has engineering features. The batter and pitcher ID is also an important feature as some batters are much more likely to hit home runs compared to others, and some pitchers are much more likely to give up home runs than others. These features could not be used for further feature engineering because they are discrete values that correspond to individual players. The next most important features are pitch_x and pitch_z, which are the horizontal and vertical pitch location with respect to home plate, as well as their engineered features. It makes sense these are important features because the pitch location has a big effect of determining where the ball is hit, as outside pitches usually go in the opposite direction, pitches down the middle are usually hit up the middle, and pitchers inside are usually pulled in the direction the batter bats. This can be used to determine the general area the ball is hit, which can then be compared with the stadium dimensions, which are also the next most important features. This is important as there is a dramatic difference in the distance to the center field fence compared to left and right field, meaning balls hit to center field have to be hit much harder and at a more optimal launch angle to be a home run. Additionally, stadium dimensions have great variance, so knowing which direction the ball is hit and what stadium is being played in is important in determining if a batted ball is a home run. In addition, traditionally hits from balls thrown near the center of the strike zone are more likely to be home runs due to solid contact being made compared to balls hit that are on the edge or outside of the strike zone due to the lack of solid contact.

6.2 Naïve Bayes

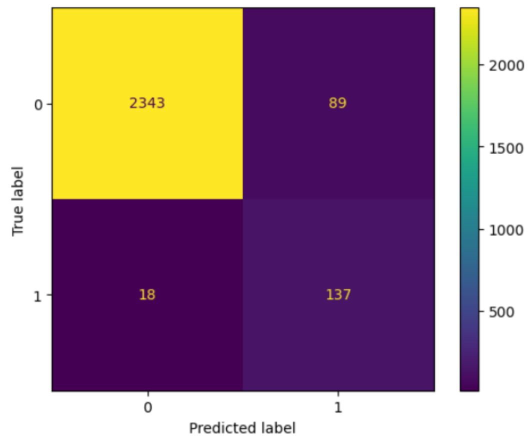
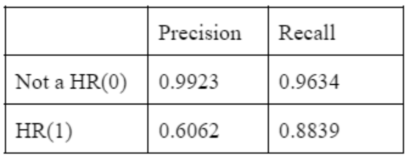


Fig. 5. Confusion Matrix for Naïve Bayes Classifications Predictions: 0 = No Home Run and 1 = Home Run



	Precision	Recall
Not a HR(0)	0.9923	0.9634
HR(1)	0.6062	0.8839

Fig. 6. Precision and Recall Values for Naïve Bayes Classification

One can see the results from the prediction of the Naive Bayes Model in the confusion matrix shown in Figure 5. One can also see the precision and recall values for each of the predicted classes in Figure 6. It can be seen from Figure 6 that the precision for the not a home run class is 0.9923, and relating this to the confusion matrix can be calculated as 2343 / (2343 + 18). A precision value of 0.9923 is excellent and means that with respect to the number of not a home run samples, very few true home runs were classified as not a home run. The recall value for the not a home run class is 0.9634, which is also very good. This value can be seen from the confusion matrix with the calculation of 2343 / (2343 + 89), and means that out of all of the not a home run testing data points, 96.34% of them were correctly classified as not a home run. Again, some of the misclassification could have possibly resulted from "robbed" home runs, where the batted ball is usually a home run as it is hit over the fence, except the fielder made an excellent play and caused the home run to result in an out.

It can be seen from Figure 6 that the precision for the home run class is 0.6062, which when initially looking at it, is not a good result. This value can be seen from the confusion matrix with the calculation of 137 / (137 + 89). A precision value of 0.6062 is not very good, but a deeper analysis shows that with respect to the size of the classes, a precision of 0.6062 is not bad. As explained earlier, this is due to the classes being extremely unbalanced as the test data contains 2432 samples of no home run data, but only 155 samples of home run data. Since the no home run class data is about 16 times larger than the home run class data, it makes sense that the precision is not very high for the home run class data, as for the no home run class, a recall or 0.9634 results in 89 misclassifications, which since there are only 155 test instance of the home run class, the precision is bound to be small. Even if all 155 instances of the home run class was correctly classified, the precision would only be 0.6352.

The recall value for the home run class is 0.8839, and relating this to the confusion matrix can be calculated as 137 / (137 + 18). This means that out of all of the predicted home run classifications, 88.39% of them were correctly classified as a home run. Again, some of these misclassifications could be a result of inside the park home runs, where traditionally the hit not going over the fence would mean it is not a home run, but due to some obscure ball bounces or fielder positioning, the batter is able to get a home run without actually hitting the ball over the fence. A recall of 0.8839 is very good, especially when taking into account the size of the class. Of the 155 instances of the home run class, only 18 are misclassified as no home run.

The overall accuracy of the Naive Bayes classifier is high at 95.85%. This accuracy can be quantified as the number of the number of correct predictions divided by the number of all predictions. It can also be thought of as the weighted average of the recalls of each class. To get a better understanding of the model's performance, a 10-fold cross validation was performed and yielded a value of 0.9563. This further confirms the validity of the model as it has very high accuracy, and shows that overfitting did not occur.

6.3 Model Comparison

Overall, it can be seen that the Random Forest model performs better when classifying none home runs as it only misclassified 23 instances and has a recall of 0.9905 compared to the Naive Bayes model, which misclassified 89 instances of data samples from the no home run class and has a recall of 0.9634. Thus, it can then be seen that the Naive Bayes model performs better for the home run class, with only 18 instances of misclassification and a recall of 0.8839, compared to 46 misclassifications and a recall of 0.7032 for the Random Forest classifier model. All in all, I would conclude that even though the overall accuracy and cross validation score for the Naive Bayes model is slightly lower than the Random Forest model, it has better performance. This is because of the Naive Bayes improved recall for the small class. When classification datasets are extremely disproportionate, precision becomes a less important metric, as even the slightest amount of misclassification of the larger class results in a significant reduction in the precision of the smaller class. This is most prominently noticeable in the Random Forest results, that can be seen in Figure 2, where of the 2,432 no home run class instances, only 23 instance are misclassified as home runs, which is a very small percentage and corresponds to a recall of 0.9905, but 23 is a much larger percentage compared to 155, so even if all of the home run instance are correctly classified, the largest the precision could be for the home run classification is 0.87.

If one takes the unweighted average of the recalls, the Random Forest model has a value of 0.847, while the Naive Bayes model has a value of 0.9237. With extremely disproportionate class sizes, this is how the performance should be evaluated. Compared to the standard definition of accuracy, which can also be represented as the weighted average of the recalls, the much larger class has a significantly greater weight, and in this case the no home run class has a weight of about 96% since the no home run class contains 96% of the total instance, compared to 4% for the home run class. Truly, the weights should be 50/50 split, as although the home run class is extremely small, predictions for the home run class are increasingly important due to the small number of instances it has, as a single correct classification of the home run class is more important than a single correct classification for the no home run class due to the larger percentage a single instance of the home run class effects the performance of the class prediction, as well as due to the smaller sized class for dataset that is used to train the model.

Additionally, the advantages in Naive Bayes can also be seen in the computation time. The Random Forest classification model takes about 2 minutes to train and calculating its cross-validation score takes about 10 minutes, while for Naive Bayes, both training and cross validation take under a second. This is a significant improvement in the computation time, which would be even more exaggerated for larger data sizes, which would be optimal and important in the improvement of the model.

7 Summary and Related Work

Overall, it can be determined that the Naive Bayes classifier is the better model to use for prediction if a batter ball is a home run or not. This is due to the significantly improved computation time, as well as the improved classification accuracy. Although the Random Forest classifier is slightly better at classifying the no home run class, the Naive Bayes model is significantly better at classifying home runs. Taking these two factors into account, the Naive Bayes model is the better model to use, especially if there is larger datasets. Although the Naive Bayes model is the preferred model, the Random Forest model still has its advantages of being able to determine feature importance, which was one of the goals of this report.

The Random Forest model showed that the most important features in predicting if a batted ball is a home run are the exit velocity and the launch angle, with launch angle being the most important. This corresponds to the findings found in the paper “Predicting Batted Ball Outcomes in Major League Baseball”[1], which also determined that launch angle and exit velocity are the most important features in determining if a batted ball is a home run or not. This can be seen in Figure 3, where the relative importance value of launch angle and exit velocity, as well as their engineered features, are significantly higher than every other feature. Additionally, the results also match the findings from "Forecasting Batter Performance Using Statcast Data in Major League"[2], which concluded that launch angle was the single most important feature in determining if a batted ball is a home run or not. This again can be seen in Figure 3, which shows that launch angle is the most important feature.

8 Conclusion

In conclusion, this report investigates advanced baseball analytics, particularly focusing on the prediction of home runs using Statcast data and machine learning models. Leveraging the collection of information provided by Statcast for the 2020 season, this study employs Random Forest and Naive Bayes algorithms to predict whether a batted ball results in a home run. Through a thorough analysis, significant insights into the pivotal features influencing home run outcomes are discovered.

The Random Forest model was used due to its ability to handle complex feature sets, and demonstrates high precision and recall for classifying non-home run instances. However, due to the disproportionate class sizes, its performance in classifying home runs is relatively lower. On the other hand, the Naive Bayes model, despite its simplicity, showcases superior recall for the home run class, and thus achieves a more balanced performance across both classes. Furthermore, the feature importance analysis produced by the Random Forest model shows that exit velocity and launch angle are integral in determining home run outcomes, aligning with prior research findings. This highlights the importance of these metrics in player evaluation and development strategies.

In conclusion, while the Naive Bayes model emerges as the preferred choice for its computational efficiency and enhanced performance in classifying home runs, the Random Forest model remains a useful model due to its feature importance determination. Overall, this study contributes to the ever-expanding field of baseball analytics, offering valuable insights into the predictive modeling of home runs and emphasizes the significance of Statcast data in advancing player assessment methodologies and techniques for improving player performance.

References

1. *Predicting Batted Ball Outcomes in Major League Baseball*, www.causeweb.org/usproc/sites/default/files/usclap/2019-1/Predicting Batted Ball Outcomes in Major League Baseball.pdf. Accessed 13 May 2024.

2. *Forecasting Batter Performance Using Statcast Data in Major* ..., library.ndsu.edu/ir/bitstream/handle/10365/28371/Taylor_ndsu_0157N_11679.pdf?sequence=1. Accessed 13 May 2024.

3. Kraggle. “Baseball.” *Kaggle*, 28 July 2021, www.kaggle.com/datasets/jcraggy/baseball/data

4. R, Sruthi E. “Understand Random Forest Algorithms with Examples (Updated 2024).” *Analytics Vidhya*, 19 Apr. 2024, www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/.

5. Zhang, Zixuan. “Naive Bayes Explained.” *Medium*, Towards Data Science, 14 Aug. 2019, towardsdatascience.com/naive-bayes-explained-9d2b96f4a9c0.