**Using Machine Learning to Predict Home Runs for a National TV Audience**

Jack R. Puncochar

University of Wisconsin – Green Bay

DS 785: Capstone

Dr. Jennifer Garland

August 7, 2022

**Abstract**

The introduction of the Statcast tracking system increased the prevalence of analytics in baseball. Research emerged claiming that Statcast metrics, exit velocity and launch angle, are strong predictors of home run hitting after contact. However, there is no research exploring the pre-contact probability of hitting a home run. Understanding the probability that a hitter goes yard before the pitch can provide smarter TV audiences with another piece of valuable content. A home run probability model exists at one media company, but the model suffers from overfitting. Also, typical binary classification problems in the sports realm explore problems with balanced classes (i.e., win or lose). This study dealt with extreme class-imbalance (< 5% of batted balls are home runs). Logistic regression and Naïve Bayes were trained on over 70,000 batted balls from the 2021 season using 2019 pitcher and hitter statistics and game state (count, outs) as model features. Both models performed poorly using precision and recall for model assessment. The poor performance was attributed to uncertainty in data that is only known pre-pitch and it was found that home run classification may not be meaningful when the desired output is a probability. Instead, log loss was used for model selection and logistic regression was selected to estimate predicted probabilities on new data. An R Shiny application made it possible to display results of the HR probability model on live pitches. The application was in early stages and needed bug fixes and optimized code prior to being sent to the client. Another limitation with the R Shiny app was a lack of automation in updating as Statcast updated in real-time. The HR probability system was not sent to the client; however, the framework to efficiently process Statcast data and deploy an accurate HR probability model will help them emerge as leaders in the industry, so it is necessary to follow up with the client.

**Keywords**

Imbalanced classes, binary classification, probability, baseball, home runs, Statcast, R Shiny, data processing, data mining, logistic regression, Naïve Bayes, precision, recall, Major League Baseball

**Table of Contents**

ABSTRACT 2

TABLE OF CONTENTS 4

LIST OF TABLES 6

LIST OF FIGURES 6

CHAPTER 1: INTRODUCTION 7

Background 7

Problem Statement 7

Research Purpose 8

Research Question 8

Data Mining 8

Plan for Model Deployment 9

Organization 9

CHAPTER 2: LITERATURE REVIEW 10

CHAPTER 3: DATA COLLECTION AND METHODOLOGY 15

Data Collection 15

Data Scraping 17

Live Data 17

Missing Data 17

Methodology 18

Logistic Regression 18

Naïve Bayes 19

Assessing Model Performance 20

Precision and Recall 20

Log Loss 21

Live Home Run Probability Display 22

CHAPTER 4: RESULTS 22

Logistic Regression 22

Naïve Bayes 25

Log Loss 27

Model Selection 27

Home Run Probability Display 28

Retroactive Model Testing – 2022 MLB Games 30

Other Findings and Discussion 31

CHAPTER 5: RECOMMENDATIONS 33

Conclusion 34

REFERENCES 36

APPENDIX A. R Code 40

APPENDIX B. R Shiny App Code 55

**LIST OF TABLES**

Table Page

1. Home Run Class Imbalance 23
2. Logistic Regression Confusion Matrix 24
3. Naïve Bayes Confusion Matrix 26
4. Model Comparison – Log Loss 27
5. Logistic Regression Significant Coefficients 28

**LIST OF FIGURES**

FigurePage

1. Predicted Probabilities of Hitting a Home Run – Logistic Regression 23
2. Precision-Recall Curve – Logistic Regression 24
3. Predicted Probabilities of Hitting a Home Run – Naïve Bayes 25
4. Precision-Recall Curve – Naïve Bayes 26
5. HR Probability by Pitch – Padres vs. Dodgers 30
6. HR Probability by Pitch – Guardians vs. Angels 31

**Chapter 1: Introduction**

**Background**

In 1980, Bill James defined sabermetrics as “the search for objective knowledge about baseball”. James observed that traditional baseball statistics did not accurately describe a player’s value to his team. Specifically, he found that batting average was flawed because it placed equal value on all four hit outcomes (single, double, triple, home run). After much progress in baseball research, it is now widely accepted that home runs are the most valuable outcome in baseball. Organizations value players that can hit for power over contact and home runs are the most exciting play from a fan’s perspective.

Sabermetrics wouldn’t be where they are today without the dramatic improvement in technology. In 2015, MLB installed tracking technology in all 30 ballparks which paved the way for “the collection and analysis of a massive amount of baseball data” (*Statcast | Glossary*, 2015). Statcast uses cameras and radar to collect information about everything that happens on the field. Player statistics, team records, player and ball location tracking data, and even meta information such as weather, umpires and scorekeepers are examples of information stored on Statcast’s massive database. Statcast made it possible to do amazing things with data. For example, Apple TV+ began incorporating probabilities into their broadcasts during MLB games in 2022. Their broadcasts display probabilities of different events such as reaching base, striking out, or hitting a home run.

**Problem Statement**

The introduction of probabilistic models to the mainstream through national TV broadcasts shows the fast progression of analytics in sports. Other sports’ broadcasts have displayed model probabilities including catch probability in football and face-off probability in hockey. In a race to capture the market share of a smarter audience, media companies have little choice but to utilize applications of machine learning to enhance their product. As legal sports betting increases in the United States, the demand for analytical content will follow. However, with the fast adoption, it isn’t a surprise that there are concerns about the validity of these probabilistic models. Apple TV+’s baseball model comes from nVenue, a tech startup that uses an algorithm with over 120 inputs. Ben Clemens, a baseball writer for Fangraphs, found that the model may suffer from overfitting (Clemens, 2022).

**Research Purpose**

The purpose of this research was to provide sports’ media clients with the framework to implement a successful home run probability model in their broadcasts. Successful models reflect domain knowledge and promote transparency in methodology and results. The Apple TV+ model lacks transparency and there is minimal evidence of its accuracy. Displaying accurate home run predictions gives media companies a competitive advantage.

**Research Question**

Is it possible to establish a repeatable process that utilizes real-time Statcast data to predict the probability that a pitch will result in a home run?

**Data Mining**

All data extraction, processing, and machine learning was completed in R. The *baseballr* package provided specific functions for data extraction. Pitch-by-pitch data was the most granular level of data in Statcast, but several functions made it easier to aggregate data to achieve team or player level statistics. Note that the *baseballr* package also has methods to extract data from websites such as Fangraphs and Baseball Reference. Without the Statcast API, there were alternative ways to get the same data such as writing HTML scraping scripts in Python. Upon completion of data processing, several machine learning models were explored including logistic regression, Naïve Bayes, and boosted aggregation (bagging). Logistic regression and Naïve Bayes were selected due to their ease of interpretability and implementation in a live home run probability model.

**Plan for Model Deployment**

The plan was to deploy the “best” model to a live environment for the client to use. The best model was selected based on model assessment metrics. It was determined that accuracy was insufficient for evaluating a model with a rare outcome. After selecting the best model, “new” data, or real-time Statcast at-bat data, would be used to predict a player’s HR probability live during their at-bat. The predicted probability, along with other game information, would be sent to an R Shiny application that displayed a tidy dashboard. It is then possible to share the Shiny app on GitHub or for a small fee on ShinyApps.io depending on its usage. Additionally, the model could be tested retroactively on completed 2022 games. There were a few key differences between historical data and real-time data that had to be addressed to prevent bugs in testing environment. A successfully deployed model also needed to automatically update as new data came in and store the model’s results. It was equally important to maintain code documentation so that the process was easily repeatable.

**Organization**

Previous literature was reviewed delving into factors that may influence home run hitting, the use of machine learning in predicting baseball outcomes, gaps in current baseball research, and the relevant class imbalance problem. Next, the methodology was outlined, and the results stated with several accompanying data visualizations. Finally, the implications of the results, research limitations, and future recommendations were discussed.

**Chapter 2: Literature Review**

Recent data shows that baseball’s style of play is changing dramatically. Previous research, conducted primarily between the early 2010s through 2018, is useful in its methodology but is quickly becoming outdated. Past methodology can be referenced to model the probability of hitting a home run, including two experiments that modelled the probability of reaching base. Previous studies also delve into factors that contribute to events in a baseball game including the composition of a baseball and the batter/pitcher matchup. A home run probability model greatly relies on the matchup between a pitcher and batter. Further, home run probability models do exist, but they are either inaccurate or they do not measure probability *prior* to contact. The purpose of the literature review is to build on the strengths of previous baseball outcome analysis.

In 2019, home runs spiked dramatically. MLB does not disclose year-to-year changes in baseballs or the manufacturing process, but it was found that the balls had less drag, allowing for balls to fly further than usual. The change in baseballs is also observed by changes to the coefficient of restitution (COR), or the energy that is lost during impact. COR affects the number of home runs hit each season (Kuninaka, 2016). It is difficult to account for year-to-year changes in the composition of a baseball, so using data from the most recent full seasons (2019 and 2021) is a conservative approach to building a home run probability model. In addition to the change in baseballs, the popularity of analytics has influenced recent strategy. Brian Cartwright found that home run, strikeout, and walk rates are way up and batting averages have decreased since 2019. This trend corresponds to pitchers’ increased throwing velocity and batters’ adjustment in their hitting approach (Cooper, 2021). The seemingly dramatic change in baseball’s style of play in recent seasons requires the use of recent data for a real-time home run probability model. Further, the association between weather and hitting has been explored but the conclusion is unclear. Research by Liu (2020) found minimal evidence for weather’s impact on getting a hit in any given game; however, this research examined *all* hits. Home runs are hit higher in the air and may be influenced by windy conditions.

A summary of past literature also shows that probability of any outcome greatly depends on the matchup between the pitcher and batter. Pitchers are typically evaluated using *pitch* level data while batter performance is assessed using *at-bat* level data. Research by Sievert and Mills summarized the state of pitching performance analysis. Pitch level data allows pitchers to be evaluated beyond traditional measures of success such as Wins and ERA. Relatively newer measures such as ground ball and fly ball rates, hard hit percentage, and fielding independent pitching (FIP) are predictive of future pitcher ability. For example, fly ball pitchers are more likely to allow home runs (Sievert & Mills, 2016). Further, a study by Powers et al. (2018) used principal component analysis to categorize at-bat results. A key finding was that home runs are considered one of the “three true outcomes”, or outcomes that only rely on the pitcher and batter’s skill. Other outcomes are known as “balls in play”, or balls that require the fielding team to make a play. This is important for the home run probability model because it means that team defensive ability can be ignored.

In addition to these studies, another experiment used a Bayesian approach to model the probability of reaching base using prior batter/pitcher matchup results. The Bayesian hierarchical log5 model used data from 5 pitchers and 15 batters from the Korean Baseball League from 2008-2017. With sufficient data from each of the 75 batter/pitcher matchup pairs, this model performed the best compared to a similar model that only used platoon splits (Doo & Kim, 2018). Platoon splits refer to the handedness of the pitcher and batter. Generally, left-handed pitchers have an advantage over left-handed batters and right-handed pitchers have the same advantage over right-handed batters. It makes sense that historical pitcher/batter matchup data is more predictive of future results for the same matchup; however, it is not feasible to train a home run probability model with individual matchup data because it is common for matchups to have never occurred.

Building on the importance of platoon splits in batter/pitcher matchups, research by Loyer and Sprechini (2011) explored the relationship between conditional probability and batter/pitcher handedness. Specifically, it was observed that Simpson’s Paradox can occur in the case of platoon splits. It is possible for Batter A to have a higher conditional probability, or batting average, against both left and right-handed pitchers while Batter B has the higher overall batting average. The research concluded that the best possible estimate for the probability of a player reaching base is batting average unless new information is introduced. Further, research by Flanagan (1998) examined the relationship between game theory and lineup building strategy. Flanagan found evidence to support the hypotheses that there is a shortage of left-handed pitchers due to biological constraints *and* that it is profitable to have a surplus of left-handed batters as a result. Managers tend to select more left-handed batters knowing that they will face right-handed pitchers, but teams seek left-handed pitching to combat lefty-heavy lineups. Capturing the batter/pitcher matchup is crucial for any home run probability model.

One of the existing home run probability models calculated home run probability *after contact* using factors such as exit velocity, launch angle, and hit distance. Also, it was discovered that batted balls are more likely to be home runs if they are hit to left or right field due to the curvature of the outfield (Kolp, 2018). MLB ballparks are unique playing surfaces in that each team’s field has different dimensions. Kolp’s model did not account for ballpark or defensive factors, but he used the model was to inform viewers of a batted ball’s chance at becoming a home run based on the exit velocity, direction, and launch angle. Also, weekly baseball broadcasts on Apple TV+ display results from nVenue’s models providing viewers with the probability that a player gets on base or hits a home run prior to the pitch. However, their models use over 120 predictors and may suffer from overfitting (Clemens, 2022). The purpose of this research was to tweak existing home run probability models by using factors known *before* the pitch and to reduce overfitting.

There are several examples of using binary classification in baseball research. Koseler et al. (2017) found that the use of support vector machines dominates current baseball classification research. Li et al. (2022) found that SVM performed the best at picking winners of games with 65% accuracy. The experiment also tested artificial neural networks (ANN) and logistic regression. Previous research shows that SVM works for predicting wins and losses or classifying pitch types because categorizing observations into classes is meaningful in those cases. However, SVM lacks a straightforward probability calculation. In addition to model accuracy, careful consideration has been given to model explainability. Akhavan Rahnama et al. (2018) argue that there is “a need for techniques that explain the decision-making process of ‘black-box’ models”. They found that feature importance scores from logistic regression and Naïve Bayes were easily derived, and they provided adjacent solutions for other models. Hosanagar and Jair (2019) argue that “regulators should get high levels of transparency while users accept medium levels”. Both sources acknowledged the need to combat “black-box” models with transparency.

Next, binary classification problems typically have a balanced distribution of observations in each class. Assessing the performance of binary models is straightforward. However, problems that deal with extreme class imbalance must be assessed using alternative methods. Research by Adhikari et al. (2021) found that traditional area-under-curve (AUC) and accuracy statistics are misleading for binary classification when the response class is imbalanced. When the response class is 99/1, predicting every observation as the majority class would yield a 99% accuracy. Adhikari et al. recommends using alternative measures such as precision and recall. These methods focus on the performance when predicting the minority class. Also, research by Liu et al. (2009) found that there are several re-sampling techniques that can improve the performance of machine learning models in imbalanced classification problems. Two methods have drawbacks – *under sampling* results in an information loss from the majority class and *over sampling* requires extensive computation due to the large increase in observations from the minority class. Research by Tian et al. (2010) found that “a combination of two unrelated approaches can perform better than either alone”. Careful thought was given to the application of these sampling techniques in the imbalanced home run data.

**Chapter 3. Data Collection and Methodology**

**Data Collection**

All data for this project was pulled using the *baseballr* package in R. This package provided functions that scraped data from several sources including fangraphs.com, baseball-reference.com, and MLB’s Statcast API. Statcast’s pitch-by-pitch data, aggregated player and team statistics, and weather data were useful for this research. Fangraphs and Baseball Reference provided game information data (dates, teams, parks, and weather information) and additional aggregated player statistics. Since Statcast’s database is so massive, the data scraping process had a learning curve. The R website (CRAN, 2022) provided extensive documentation for the *baseballr* package.

Statcast data dates back to 2015; however, due to the sheer size of one season of data, only 2019 and 2021 data was collected. 2020 was the COVID-shortened season where teams played a non-standard schedule in empty stadiums. 2019 data was used to aggregate player statistics and use as model features. 2021 data was at a pitch-by-pitch level; one pitch was one record that could be predicted by the model. The 2021 season had over 700,000 pitches, more than enough data to train the models, however, the data set was eventually reduced. Some features in the Statcast data included:

* Identification (pitcher/batter ID, game ID)
* Pitch information (velocity, release angle, spin rate, pitch type)
* Batted ball information (launch angle, exit velocity, hit distance)
* Game information (inning, outs, runners on base, count, teams)
* Weather information (temperature, wind direction and speed)

Further, several game information statistics were recorded before *and* after the pitch. For example, the count before the pitch was 2 balls and 1 strike, and the count after was 2 balls and 2 strikes. “Before” stats were more appropriate for a model predicting home runs prior to the pitch. Additionally, 2019 batted ball information was aggregated by player to use as model features for both pitchers and hitters. 2021 pitch-by-pitch data was merged with 2019 statistics. As mentioned, the original data had over 700,000 pitches, but this was reduced to just over 75,000 pitches that were hit into play, or batted balls. The large data set had many identical copies of features which caused model issues. The following batted ball information was used in the model:

* Exit Velocity
* Launch Angle
* Exit Velocity Allowed
* Launch Angle Allowed
* Pitch Speed

A batter’s propensity to hit home runs was measured by their average exit velocity and launch angle from the 2019 season. Likewise, pitcher performance was measured by their exit velocity, launch angle allowed, and pitch speed. The idea was that pitchers who allowed hard-hit balls at certain angles were more likely to allow home runs. Further, research from Chamberlain (2020) found that exit velocity was predictive for hitters and launch angle was predictive for pitchers *after* contact.

***Data Scraping***

Initially, 2019 data was scraped from Baseball Reference using Python and HTML scraping techniques. However, rather late in the process, it was found that the *baseballr* package could do the same work more efficiently. The code to scrape 2019 data was copied from Robert Frey’s YouTube channel and GitHub. Due to the size of the data, it had to be scraped week-by-week and all data frames had to be merged at the end. After the full 2019 pitch data was pulled, player batted ball information was calculated (i.e., 2019 average exit velocity). The Statcast API proved to be effective for merging many sources because the IDs were standard throughout the complete database. Pitcher and batter IDs made it possible to merge historical data with the training data. Game IDs made it possible to merge pitch data with weather data.

***Live Data***

The Statcast database updates in real-time, so it was possible to pull model features to use in a live home run probability model. A slightly altered process was required to obtain data for live MLB games. Several functions were written to extract, clean, merge, and impute missing data before the new record was tested.

***Missing Data***

It was important to figure out how to handle players with insufficient historical data. It was common for players competing in 2022 to not have any 2019 data. There are many rookies and other young players who haven’t had a chance to accrue statistics. Previous baseball research has often described player skill in terms of his relationship to an average player’s statistics. Thus, any missing batted ball information was replaced with replacement-level values obtained by calculating 2019 league *medians*. Any player with missing data would be considered average in the model. Further, weather data could not be obtained for live games because Statcast updates game information at the conclusion of games.

**Methodology**

The purpose of this research was two-fold: 1) to test the performance of different machine learning models on predicting a binary outcome (home run or no home run), and 2) to use the best model in a live home run probability display. There are several algorithms that can predict binary outcomes; however, the response variable for this problem was heavily imbalanced. Additionally, the predicted probability was more informative than the classification groups which ruled out algorithms such as Support Vector Machines. Research by Adhikari et al. (2021) found that logistic regression performed well with *rare* binary outcomes. However, the Naïve Bayes algorithm may perform better than logistic regression if assumptions are met (Chauhan, 2018). An important feature of these models is that the model output is interpretable. Thus, both models were trained using the following set of features:

* Pitcher handedness (L or R)
* Batter handedness
* Outs
* Count
* Batter Avg Exit Velocity
* Batter Avg Launch Angle
* Pitcher Avg Exit Velocity
* Pitcher Avg Launch Angle

***Logistic Regression***

Logistic regression is the most common machine learning method for predicting a binary outcome. Some practical uses of this algorithm include email spam and fraud detection. Logistic regression works for the home run probability model because its output returns a probability value using the sigmoid function (Pant, 2021). This method also has an advantage over other probabilistic methods because the model’s coefficients are interpretable. It is possible to calculate the extent to which a variable change affects the probability of a home run. To improve performance, 10-fold cross validation was used to iteratively split the full data set into a training and test set. For each of the 10 splits, 9/10 of the data was used to train the model to test on the remaining 1/10 resulting in a full set of predicted probabilities. The accuracy of predicted probabilities was tested using log loss and the classification performance was tested using precision and recall. Classifying home runs required finding the optimal decision threshold which is described below.

***Naïve Bayes***

The Naïve Bayes classifier is a probabilistic model based on Bayes Theorem. It generates a posterior probability for a new observation using prior predictor probabilities.

P(A|B) =

This formula can be extended to a full set of predictors. Given that the model relies on conditional probability, Naïve Bayes assumes that all predictors are independent and have the same effect on the response variable. All categorical variables were one-hot encoded only to check correlation between all predictors. Then, the model was trained using 10-fold cross validation on the original predictors mentioned above. Finally, the accuracy of probabilities was assessed using log loss and the classification predictions were summarized and evaluated with the confusion matrix and precision-recall curve after finding the optimal decision threshold.

***Assessing Model Performance***

Throughout the research, it was found that log loss was more informative when assessing the model’s performance. Precision and recall relied on a decision threshold to classify pitches as home run or no home run, but this classification wasn’t exactly meaningful. The purpose of the research was to get as close to the true probability as possible. However, precision and recall curves were still obtained for each model because they provided additional context.

***Precision and Recall***

Typical binary classification algorithms are evaluated using the area under curve (AUC) and accuracy percentages. Research by Adhikari et al. (2021) suggests that *precision* and True Positive Rate (TPR), or *recall*, are more informative when a binary outcome is rare.

*Precision* =

*Recall* =

Precision and recall can be visualized with an area under the precision-recall curve (AUC-PR). Research by Sofaer et al. (2019) found that visualizing model performance with an AUC-PR curve is optimal for a binary classification problem with a rare outcome. Before classifying observations, it was necessary to select an optimal decision threshold. Typical binary classifiers use .5 as the default threshold where observations with probabilities greater than .5 are assigned to one class (y = 1) and all other observations are assigned to the other class (y = 0). However, home run hitting has imbalanced classes as <1% of pitches result in a home run. Thus, it was possible that the optimal threshold was a value other than .5. The following process was used to find the optimal threshold for each machine learning method:

1. Fit the model
2. Predict probabilities
3. Define thresholds (0, .01, .02, …, 1)
4. Evaluate each threshold, select threshold that maximizes *F* score, where

F score =

***Log Loss***

Another way to evaluate model performance when the outcome classes are imbalanced is log loss. Log loss is a function that penalizes predictions that are further from the true outcome. Kulkarni (2021) provides the formulaic definition of log loss:

For example, if the actual outcome is y = 1 and the predicted probability was .1, the resulting log loss would be:

= 2.30

A model that minimizes total log loss was desired. Log loss is directly affected by the predicted probability of hitting a home run. The predicted probability is what is being displayed on the broadcast, so it was important for a model to be as close to the true probability of hitting a home run as possible.

***Live Home Run Probability Display***

The second purpose for the research was to display the best model’s home run probabilities. The *baseballr* package made it possible to pull the live at-bat data from MLB’s Statcast API. Model features were obtained using this package. Several functions were written to extract the data and get it into the format needed for the prediction model. Some quirks in the live data feed were addressed including invalid counts (i.e., 1 ball and 3 strikes). Also, automatically updating the process to obtain real-time data was challenging. Due to limited time, it was more feasible to *manually* update the prediction during live at-bats. Once all the features were obtained, they were plugged into the selected model to produce the home run probability. This probability was displayed neatly in a scoreboard created using R Shiny. It was also possible to retroactively gather at-bat data from recent games. Additional code was written to select random dates, games, and at-bats to collect a large enough data set to test the model and make additional conclusions.

**Chapter 4. Results**

**Logistic Regression**

*Table 1* shows the response variable’s extreme class imbalance. This imbalance identified potential for evaluating performance at different decision thresholds. It also supported the decision to use log loss, precision, and recall, rather than accuracy, to assess the model’s performance.

Graphical user interface, text, application

Description automatically generated

*Figure 1* shows the distribution of predicted probabilities derived from the 10-fold cross validated logistic regression model.

Chart, histogram

Description automatically generated

The distribution shows that using .5 as a decision threshold would be erroneous. However, the distribution does not provide evidence that the logistic model revealed higher predicted probabilities for actual home runs. To investigate the model’s performance further, precision, recall, and F score were calculated at several probability thresholds from (0, .01, .02, …, 1). The optimal threshold was *p* = **.07**, and *Figure 2* shows the precision-recall curve along with the specific precision and recall point that optimized the threshold. The threshold means that any observations with a greater than 7% chance of being a home run would be classified as such. Further, the area under the precision-recall curve (AUC-PR) is .0739.

Chart, scatter chart

Description automatically generated

The low AUC-PR suggests the logistic regression model did not successfully classify pitches.

*Table 2* summarizes the model’s predictions in a confusion matrix. Recall = .233 and precision = .086 at *p* = .07. The model’s accuracy is .84, which sounds like a good number, but a model that predicted all pitches as non-home runs would have had .951 accuracy.

Table

Description automatically generated

**Naïve Bayes**

The 10-fold cross validated Naïve Bayes model produced similar results. *Figure 3* shows the predicted probabilities distribution, which looks like that of the logistic regression model. There doesn’t seem to be a clear shift in probability distribution between home runs and non-home runs.

Chart, histogram

Description automatically generated

*Figure 4* shows the precision-recall curve and the optimal threshold for the Naïve Bayes model.

Chart, scatter chart

Description automatically generated

*Table 3* summarizes the model’s class predictions at an optimal probability threshold of *p* = .07. Recall = .249 and precision = .075 at *p* = .07.

Table

Description automatically generated

Again, these results show poor performance in classifying home runs. There are likely several reasons for this poor performance. First, it is just difficult to classify home runs before the pitch. Successful HR probability models benefit from post-contact exit velocity and launch angle; however, these were not included due to the scope of this model. Also, less than 5% of batted balls are home runs. Even if the clear best hitter is facing the clear worst pitcher, it is still unlikely they will hit a home run, and it is difficult to differentiate a single from a double or home run *prior* to the pitch. Another reason could be that the classification threshold is not meaningful for predicting home run probability. Instead, it may be useful to select a model using log loss.

**Log Loss**

In context of predicting home run probability for broadcast purposes, it may not be necessary to classify pitches as home runs. Instead, the predicted probability is more informative. Using log loss is ideal when the output of a model is a probability. *Table 4* shows the log loss for each method.

Table

Description automatically generated with low confidence

There is no significant difference between models in terms of log loss. Additionally, the logistic regression and bagging models produced 84% and 81.3% accuracy, respectively. Using accuracy vs. precision/recall lead to significantly different conclusions. Future home run classification models should consider alternative model assessment methods in lieu of accuracy.

**Model Selection**

Since both models performed similarly, logistic regression was selected as the final model for the home run probability display. Logistic regression has an advantage due to the ability to interpret model coefficients. *Table 5* shows significant predictors from the logistic regression model and its corresponding effects on the odds of hitting a home run. The most interesting result is that the count greatly influences home run odds. This makes sense because the pitcher’s options change depending on the count. 3-1 and 2-0 counts are favorable to the hitter because the pitcher must throw a strike to keep the at-bat alive increasing the chances of throwing a fastball. Conversely, 0-2 counts favor the pitcher because the pitcher can throw a curveball or slider out of the zone to get the hitter to chase a pitch.

Table

Description automatically generated

The odds can be calculated directly from the model’s coefficients using the following formula:

For example, the odds of hitting a home run increase by 57% when the count is 3-1 compared to the start of the at-bat when the count is 0-0. The odds of hitting a home run decrease by 35% with a 0-2 count compared to a 0-0 count. This will be explored further in the live home run results section.

**Home Run Probability Display**

Statcast updates game data in real-time, so it was possible to see a hitter’s home run probability while watching on TV or following along with the MLB.com/gameday live feed. Screenshots below show an example of the R Shiny application with the model’s output and the accompanying plays.

Graphical user interface, application, Word

Description automatically generated

*Screenshot of R Shiny application – Luis Arraez 4.5% HR probability*

Graphical user interface, text, application, chat or text message

Description automatically generated

*Game feed mlb.com/gameday Twins vs. Tigers 8/2/2022*

Several improvements must be made to the system before sending it to media clients. First, the application must be able to update in real-time. Currently, the data can be extracted from Statcast, cleaned, and sent to the Shiny application in a few seconds. However, this requires running code manually. Ideally, the database would update every second or so and the application would only update if the data changed from the previous run.

**Retroactive Model Testing – 2022 MLB Games**

To investigate the logistic regression model further, several random games from the 2022 season were selected. After pulling the necessary model features from Statcast, each pitch was given a home run probability. *Figure 5* shows the results from the Padres vs. Dodgers game on 7/1.

A picture containing timeline

Description automatically generated

The pitches with the highest HR probability occurred when the count was favorable to the hitter. Will Smith’s and Trea Turner’s at-bats show the probability increased as the at-bat progressed due to the count. Next, *Figure 6* shows results from the Guardians vs. Angels game on 4/25.

Graphical user interface

Description automatically generated with medium confidence

Unsurprisingly, the pitch with the highest home run probability occurred with a favorable hitter count. Additionally, the annotated pitches show that the model is picking up on hitter skill because these players are known power hitters.

**Other Findings and Discussion**

Originally, *Run Expectancy* and *Win Expectancy* were features in both models. A team’s expected runs is calculated based on the number of outs and runners on base and regressing on historical runs scored in each game state (Albert, 2015). The win expectancy calculation follows a similar approach, but also accounts for inning and team skill. The performance with these features was outstanding (> .8 recall and precision); however, it was found that those statistics measured the *difference* in run expectancy before and after the pitch. High run expectancy deltas meant that something good happened (runs scored, home run, etc.), so obviously these metrics would be extremely correlated with home run hitting. Removing these features resulted in a dramatic performance decrease. Additionally, there was some evidence that weather information was associated with home run hitting. Specifically, wind blowing towards the outfield fence tended to increase the number of home runs. However, there were some issues with using weather data: 1) this information could not be updated in real-time so it couldn’t be used in a live model and 2) each game only had one static set of weather data, so any weather model would have assumed constant wind speed and direction throughout the game, which does not reflect reality.

Logistic regression and Naïve Bayes performed equally poor. Neither could successfully predict classes and the difference between predicted probabilities for actual home runs and non-home runs was insignificant. There was some slight association between the model’s features and home run hitting. Specifically, known power hitters with favorable counts were more likely to hit home runs according to the model. Future work should do two things. First, investigate the performance of other probabilistic machine learning techniques on imbalanced classes, such as random forest and support vector machines. These methods differ from logistic regression because they are not inherently probabilistic, but it is possible to obtain probabilities using probability calibration with Platt Scaling (ter Braak, 2022). Second, careful consideration should be given to additional features such as ballpark dimensions and weather. A left-handed hitter should be more likely to hit a home run at Yankee Stadium because of the short porch in right field, and even more likely to hit a home run if the wind is blowing out to right field. Statcast makes it possible to extract vast data so there is potential to improve the model. Last, it was mentioned in the literature review that re-sampling techniques could solve issues stemming from class-imbalance. The practice of implementing these sampling algorithms was complex; however, there is strong evidence that doing so would result in better performance.

**Chapter 5. Recommendations**

Statcast’s use in Major League Baseball is already widespread. Teams have analytics departments devoted to writing programs that extract, clean, and analyze vast amounts of data. The data makes it possible to create cool applications like the home run probability model. The current iteration of the home run probability model needs improvement before moving it to production. The R Shiny application that displays HR probability is in its infant stages. It takes a few seconds to pull Statcast data and “send” it to Shiny, and the process is manual. Further, the application has a static HTML aesthetic - utilizing JavaScript can give it an upgraded modern design and allow for constant updates. However, the bones of the system are there:

1. Statcast data processing
2. model trained on historical data
3. live Statcast data processing
4. Run Shiny application

Each step can be tweaked to improve efficiency and model performance. It is necessary to set up a database that tracks home run predictions in real-time. The database must update itself and check against previous versions to update the R Shiny application. An automatically updating database would make it possible to create interesting visuals showing the progression of HR probability as the at-bat progresses. Another possibility is developing a home run over/under expected metric that utilizes the model’s predicted probabilities and compares it to a hitter or pitcher’s actual output. As analytics become mainstream, metrics like these are useful to show as the game progresses.

Further, the application is only as good as the predictions. The purpose of the project was to improve on existing models that had overfitting issues. Although the precision and recall for this model was poor, classifying home runs may not be meaningful to the audience. Therefore, log loss or other probabilistic evaluation metrics should be used to evaluate future models. It is recommended that additional work outlined above be completed prior to moving the model and application to production. The model’s outputs should align with what baseball experts would expect in different situations. For example, should a hitter ever have a greater than 50% chance of hitting a home run? Should the greatest power hitters have a baseline HR probability no matter the situation? These are some questions that should be given critical thought by key stakeholders.

**Conclusion**

There is high demand for analytical content as sports betting legalization spreads through the country. Media companies that welcome the intersection of sports and machine learning will emerge as leaders in the sports content industry. This research paper developed the framework to deploy a home run probability model to live audiences. Key objectives included processing data from the immense Statcast database, training machine learning models that were both interpretable and repeatable, and displaying accurate home run probabilities that align with baseball experts’ expectations. Although the models suffered from poor home run *classification* performance, it was discovered that probability accuracy was more relevant. Predicting home runs before the pitch is difficult because the outcome is rare and there is uncertainty without key post-contact predictors such as exit velocity and launch angle. Statcast makes it possible for countless predictors to be analyzed further to improve future model performance. The process to send model predictions to the R Shiny application must also be refined. After refinement, the steps laid out in this paper can be extended to other outcomes in sports. As national audiences become accustomed to machine learning, careful consideration must be given to the model’s limitations. However, successful applications of probabilistic models on national TV will increase viewership as the average fan becomes smarter.

**References**

A. (2020, July 5). *Practical Guide to deal with Imbalanced Classification Problems in R*. Analytics

Vidhya. <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal->imbalanced-classification-problems/

Adhikari, S., Normand, S. L., Bloom, J., Shahian, D., & Rose, S. (2021). Revisiting performance

metrics for prediction with rare outcomes. *Statistical Methods in Medical Research*, *30*(10), 2352–2366. <https://doi.org/10.1177/09622802211038754>

Akhavan Rahnama, A. H., Butepage, J., Guerts, P., & Bostrom, H. (2018, June). *Evaluating Local*

*Explanations Using White-Box Models*. 1–21. https://doi.org/10.1145/1122445.1122456

Albert, J. (2015). Beyond runs expectancy. *Journal of Sports Analytics*, *1*(1), 3–18.

<https://doi.org/10.3233/jsa-140001>

Chamberlain, A. (2020, July 14). *Launch Angle, Pitch Location, and What Pitchers Can(not)*

*Control*. RotoGraphs Fantasy Baseball. <https://fantasy.fangraphs.com/launch-angle-pitch-location-and-what-pitchers-cannot-control/>

Chauhan, G. (2018, October 8). *All about Naive Bayes - Towards Data Science*. Medium.

https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf

Clemens, B. (2022, May 26). *How Good Are Those Probabilities on the Apple TV+ Broadcasts?*

FanGraphs Baseball. <https://blogs.fangraphs.com/how-good-are-those-probabilities-on-the-apple-tv-broadcasts/>

Cooper, J. J. (2021, May 26). *Home Runs, Strikeouts and Low Averages Are Trending Throughout*

*Baseball*. Baseball America. <https://www.baseballamerica.com/stories/home-runs-strikeouts-and-low-averages-are-trending-throughout-baseball/>

Comprehensive R Archive Network (CRAN). (2022, April 25). *CRAN - Package baseballr*.

Acquiring and Analyzing Baseball Data. <https://cran.r->project.org/web/packages/baseballr/

Doo, W., & Kim, H. (2018). Modeling the probability of a batter/pitcher matchup event: A

Bayesian approach. *PLOS ONE*, *13*(10), e0204874. <https://doi.org/10.1371/journal.pone.0204874>

Flanagan, T. (1998). Game theory and professional baseball: Mixed-strategy models. *Journal of*

*Sport Behavior*, *21*(2), 121.

Hosanagar, K., & Jair, V. (2019, March 19). *We Need Transparency in Algorithms, But Too Much*

*Can Backfire*. Harvard Business Review. Retrieved July 25, 2022, from

<https://hbr.org/2018/07/we-need-transparency-in-algorithms-but-too-much-can->backfire

Kolp, T. (2018). Home Run Probability Based on Hit Distance and Direction.

Koseler, K., & Stephan, M. (2017). Machine Learning Applications in Baseball: A Systematic

Literature Review. *Applied Artificial Intelligence*, *31*(9–10), 745–763. https://doi.org/10.1080/08839514.2018.1442991

Kulkarni, P. (2021, September 22). *Log Loss as a performance metric*. Datasset to Mindset.

https://www.data4v.com/log-loss-as-a-performance-metric/

Kuninaka, H., Kosaka, I., & Mizutani, H. (2016). Home-run probability as a function of the

coefficient of restitution of baseballs. *American Journal of Physics*, 1–18. <https://doi.org/10.48550/1509.07149>

Li, S. F., Huang, M. L., & Li, Y. Z. (2022). Exploring and Selecting Features to Predict the Next

Outcomes of MLB Games. *Entropy*, *24*(2), 288. https://doi.org/10.3390/e24020288

Liu, A. (2020, August 9). *Weather and Wind Data and Methodology*. MLB HIT PREDICTOR.

https://eglouberman.github.io/MLB-hit-predictor/docs/weatherandwind.html

Loyer, M. W., & Sprechini, G. D. (2011). Can the Probability of an Event Be Larger or Smaller

Than Each of Its Component Conditional Probabilities? *CHANCE*, *24*(1), 44–53. <https://doi.org/10.1080/09332480.2011.10739851>

Pant, A. (2021, December 7). *Introduction to Logistic Regression - Towards Data Science*.

Medium. <https://towardsdatascience.com/introduction-to-logistic-regression->66248243c148

Powers, S., Hastie, T., & Tibshirani, R. (2018). Nuclear penalized multinomial regression with an

application to predicting at bat outcomes in baseball. *Statistical Modelling*, *18*(5–6), 388–410. https://doi.org/10.1177/1471082x18777669

Sievert, C., & Mills, B. (2016). Using Publicly Available Baseball Data to Measure and Evaluate

Pitching Performance. *Handbook of Statistical Methods and Analyses in Sport*, 39–64.

Sofaer, H. R., Hoeting, J. A., & Jarnevich, C. S. (2019). The area under the precision‐recall curve

as a performance metric for rare binary events. *Methods in Ecology and Evolution*, *10*(4), 565–577. <https://doi.org/10.1111/2041-210x.13140>

*Statcast | Glossary*. (2015). MLB.Com. Retrieved 2022, from

<https://www.mlb.com/glossary/statcast>

ter Braak, L. (2022, June 21). *Introduction to Probabilistic Classification: A Machine Learning*

*Perspective*. Medium. <https://towardsdatascience.com/introduction-to-probabilistic->classification-a-machine-learning-perspective-b4776b469453

Robert Frey. (2020, January 15). *Get Full Year Statcast Data with Some Simple Steps* [Video].

YouTube. https://www.youtube.com/watch?v=swJr4u-HYr0

Tian, J., Gu, H., & Liu, W. (2010). Imbalanced classification using support vector machine

ensemble. *Neural Computing and Applications*, *20*(2), 203–209. https://doi.org/10.1007/s00521-010-0349-9

Xu-Ying Liu, Jianxin Wu, & Zhi-Hua Zhou. (2009). Exploratory Undersampling for Class-

Imbalance Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550. https://doi.org/10.1109/tsmcb.2008.2007853

Z. (2021, December 13). *Logistic Regression Explained - Towards Data Science*. Medium.

<https://towardsdatascience.com/logistic-regression-explained-9ee73cede081>

**Appendix A.**

**R Code**

### All project code ###

### Scraping Statcast, Data Analysis, Machine Learning, Data Visualizations ###

# Load necessary packages

library(baseballr)

library(tidyverse)

library(ggplot2)

library(gt)

library(PRROC)

library(ipred)

library(rpart)

library(caret)

library(reshape2)

# set wd

setwd('C:/Users/punco/OneDrive/Desktop/DS 785 Assignments')

# initialize data frame for live pitch tracking (way up here so I don't write over it)

all\_live\_pitches <- data.frame()

# Scrape Statcast w/ existing code (2021) provided by Robert Frey's YouTube video

# Source: https://github.com/robert-frey/YouTube/tree/master/Obtain%20Full%20Year%20Statcast%20Data%20with%20Some%20Simple%20Steps

#load Statcast data week by week, since it can only load 10 days at a time or 40,000 observations

#scrape\_statcast\_savant scrapes data from Savant given the game dates and the player types

date328407 = baseballr::scrape\_statcast\_savant(start\_date = '2019-03-28',

end\_date = '2019-04-07', player\_type = 'batter')

date408414 = baseballr::scrape\_statcast\_savant(start\_date = '2019-04-08',

end\_date = '2019-04-14', player\_type = 'batter')

date415421 = baseballr::scrape\_statcast\_savant(start\_date = '2019-04-15',

end\_date = '2019-04-21', player\_type = 'batter')

date422428 = baseballr::scrape\_statcast\_savant(start\_date = '2019-04-22',

end\_date = '2019-04-28', player\_type = 'batter')

date429505 = baseballr::scrape\_statcast\_savant(start\_date = '2019-04-29',

end\_date = '2019-05-05', player\_type = 'batter')

date506512 = baseballr::scrape\_statcast\_savant(start\_date = '2019-05-06',

end\_date = '2019-05-12', player\_type = 'batter')

date513519 = baseballr::scrape\_statcast\_savant(start\_date = '2019-05-13',

end\_date = '2019-05-19', player\_type = 'batter')

date520526 = baseballr::scrape\_statcast\_savant(start\_date = '2019-05-20',

end\_date = '2019-05-26', player\_type = 'batter')

date527602 = baseballr::scrape\_statcast\_savant(start\_date = '2019-05-27',

end\_date = '2019-06-02', player\_type = 'batter')

date603609 = baseballr::scrape\_statcast\_savant(start\_date = '2019-06-03',

end\_date = '2019-06-09', player\_type = 'batter')

date610616 = baseballr::scrape\_statcast\_savant(start\_date = '2019-06-10',

end\_date = '2019-06-16', player\_type = 'batter')

date617623 = baseballr::scrape\_statcast\_savant(start\_date = '2019-06-17',

end\_date = '2019-06-23', player\_type = 'batter')

date624630 = baseballr::scrape\_statcast\_savant(start\_date = '2019-06-24',

end\_date = '2019-06-30', player\_type = 'batter')

date701707 = baseballr::scrape\_statcast\_savant(start\_date = '2019-07-01',

end\_date = '2019-07-07', player\_type = 'batter')

date708714 = baseballr::scrape\_statcast\_savant(start\_date = '2019-07-08',

end\_date = '2019-07-14', player\_type = 'batter')

date715721 = baseballr::scrape\_statcast\_savant(start\_date = '2019-07-15',

end\_date = '2019-07-21', player\_type = 'batter')

date722728 = baseballr::scrape\_statcast\_savant(start\_date = '2019-07-22',

end\_date = '2019-07-28', player\_type = 'batter')

date729804 = baseballr::scrape\_statcast\_savant(start\_date = '2019-07-29',

end\_date = '2019-08-04', player\_type = 'batter')

date805811 = baseballr::scrape\_statcast\_savant(start\_date = '2019-08-05',

end\_date = '2019-08-11', player\_type = 'batter')

date812818 = baseballr::scrape\_statcast\_savant(start\_date = '2019-08-12',

end\_date = '2019-08-18', player\_type = 'batter')

date819825 = baseballr::scrape\_statcast\_savant(start\_date = '2019-08-19',

end\_date = '2019-08-25', player\_type = 'batter')

date826901 = baseballr::scrape\_statcast\_savant(start\_date = '2019-08-26',

end\_date = '2019-09-01', player\_type = 'batter')

date902908 = baseballr::scrape\_statcast\_savant(start\_date = '2019-09-02',

end\_date = '2019-09-08', player\_type = 'batter')

date909915 = baseballr::scrape\_statcast\_savant(start\_date = '2019-09-09',

end\_date = '2019-09-15', player\_type = 'batter')

date916922 = baseballr::scrape\_statcast\_savant(start\_date = '2019-09-16',

end\_date = '2019-09-22', player\_type = 'batter')

date923929 = baseballr::scrape\_statcast\_savant(start\_date = '2019-09-23',

end\_date = '2019-09-29', player\_type = 'batter')

#combine all data into one data frame (729,793 rows)

SavantData19 = rbind(date328407, date408414, date415421, date422428, date429505,

date506512, date513519, date520526, date527602, date603609,

date610616, date617623, date624630, date701707, date708714,

date715721, date722728, date729804, date805811, date812818,

date819825, date826901, date902908, date909915, date916922,

date923929)

#write file to a csv on your machine (csv is a comma separated value excel file)

write.csv(SavantData19,"SavantHittingData19.csv", row.names = F)

#read in the csv file that you just created to the dataframe

SavantData19 = read.csv("SavantHittingData19.csv", stringsAsFactors = F)

# End of source's code

# scrape 2019 hitter statistics

hitter\_stats\_2019 <- mlb\_stats(stat\_type = 'season', stat\_group = 'hitting', season = 2019, player\_pool = "All")

hitter\_stats\_2019 <- tibble(hitter\_stats\_2019)

# Scrape 2019 pitcher statistics

pitcher\_stats\_2019 <- mlb\_stats(stat\_type = 'season', stat\_group = 'pitching', season = 2019, player\_pool = "All")

pitcher\_stats\_2019 <- tibble(pitcher\_stats\_2019)

# Average exit velocity by hitter

ev\_df <- SavantData19 %>% drop\_na(launch\_speed) %>% dplyr::group\_by(batter, player\_name) %>% dplyr::summarise(avg\_EV = mean(launch\_speed)) %>%

dplyr::arrange(desc(avg\_EV))

# Launch angle by hitter

angle\_df <- SavantData19 %>% drop\_na(launch\_angle) %>% dplyr::group\_by(batter, player\_name) %>% dplyr::summarise(avg\_angle = mean(launch\_angle)) %>%

dplyr::arrange(desc(avg\_angle))

# Add pitcher name to pitcher stats data

SavantData19 <- left\_join(SavantData19, pitcher\_stats\_2019 %>% dplyr::select(player\_id, player\_full\_name), by=c("pitcher"="player\_id"))

# EV allowed -- pitcher

ev\_allowed <- SavantData19 %>% drop\_na(launch\_speed) %>% dplyr::group\_by(pitcher, player\_full\_name) %>% dplyr::summarise(avg\_EV\_allowed = mean(launch\_speed)) %>%

dplyr::arrange(desc(avg\_EV\_allowed))

# Launch angle allowed -- pitcher

angle\_allowed <- SavantData19 %>% drop\_na(launch\_angle) %>% dplyr::group\_by(pitcher, player\_full\_name) %>% dplyr::summarise(avg\_angle\_allowed = mean(launch\_angle)) %>%

dplyr::arrange(desc(avg\_angle\_allowed))

# average Release speed -- pitcher

avg\_velocity <- SavantData19 %>% drop\_na(release\_speed) %>% dplyr::group\_by(pitcher, player\_full\_name) %>% dplyr::summarise(avg\_velo = mean(release\_speed)) %>%

dplyr::arrange(desc(avg\_velo))

# Merge all hitter stats

hitters\_df\_all <- merge(ev\_df, angle\_df, by = c("batter", "player\_name"))

hitters\_df\_all <- left\_join(hitters\_df\_all, hitter\_stats\_2019 %>% dplyr::select(player\_id, plate\_appearances), by=c("batter"="player\_id"))

# Merge all pitcher stats

#put all data frames into list

pitcher\_df\_list <- list(ev\_allowed, angle\_allowed, avg\_velocity)

#merge all data frames in list

pitchers\_df\_all <- pitcher\_df\_list %>% reduce(full\_join, by=c('pitcher', 'player\_full\_name'))

pitchers\_df\_all <- left\_join(pitchers\_df\_all, pitcher\_stats\_2019 %>% dplyr::select(player\_id, innings\_pitched), by=c("pitcher"="player\_id"))

########################

#### WEATHER DATA ####

weather\_payload <- get\_game\_info\_sup\_petti() %>% dplyr::filter(substr(game\_date, 1,4)==2021 & game\_type=="R") %>% dplyr::select(game\_pk, venue\_name,

venue\_id, temperature,

other\_weather, wind)

weather\_data <- tibble(weather\_payload)

# Wind Speed and Direction

weather\_data <- weather\_data %>% mutate(WindSpeed = as.numeric(sub(" .\*", "", wind)),

WindDirection = sub(".\*, ", "", wind))

# Wind Category (Clear, cloudy, rain, dome)

weather\_data <- weather\_data %>% mutate(WeatherCategory = case\_when(other\_weather %in% c("Sunny", "Clear") ~ "Clear",

other\_weather %in% c("Cloudy", "Partly Cloudy", "Overcast") ~ "Clouds",

other\_weather %in% c("Drizzle", "Rain", "Snow") ~ "Precip",

TRUE ~ "Dome/Roof"),

DirectionCategory = case\_when(WindDirection %in% c("Out To CF", "Out To RF", "Out To LF") ~ "Out",

WindDirection %in% c("In From CF", "In From RF", "In From LF") ~ "In",

WindDirection %in% c("None", "Calm") ~ "No wind",

WindDirection %in% c("L To R", "R To L") ~ "Crosswind",

TRUE ~ "Other"))

# couple quick weather plots

ggplot(weather\_data, aes(WindSpeed)) + geom\_histogram() + facet\_wrap(~venue\_name, ncol= 6) + theme\_minimal()

ggplot(weather\_data, aes(DirectionCategory)) + geom\_bar() #+ facet\_wrap(~venue\_name, ncol= 6) + theme\_minimal()

## DATA ANALYSIS ##

# Read in 2021 Kaggle pitch by pitch data

pitches <- read.csv("pitches2021.csv")

# Merge pitches and weather data

pitches <- left\_join(pitches, weather\_data %>% dplyr::select(game\_pk, venue\_name, temperature, WeatherCategory, WindSpeed, DirectionCategory), by="game\_pk")

# Only want batted balls

pitches2 <- pitches %>% dplyr::filter(events %in% c("single", "double", "triple", "force\_out", "grounded\_into\_double\_play", "home\_run",

"sac\_fly\_double\_play", "fielders\_choice\_out", "field\_out", "field\_error", "fielders\_choice",

"triple\_play", "sac\_fly"))

# Home Run binary variable

pitches2 <- pitches2 %>% dplyr::mutate(home\_run = ifelse(events=='home\_run',1,0))

# remove unnecessary columns from pitches data

pitches3 <- pitches2 %>% dplyr::select(batter, pitcher, events, stand, p\_throws, count, outs\_when\_up, inning, home\_run, venue\_name, temperature,

WeatherCategory, WindSpeed, DirectionCategory)

# Merge 2019 hitter stats with 2021 pitches

pitches4 <- left\_join(pitches3, hitters\_df\_all, by = "batter")

# Merge 2019 pitcher stats with 2021 pitches

pitches5 <- left\_join(pitches4, pitchers\_df\_all, by = "pitcher")

# rename batter and pitcher name columns

pitches5 <- pitches5 %>% dplyr::rename(BatterName = player\_name, PitcherName=player\_full\_name)

# REMOVE NAs

pitches5 <- pitches5[!is.na(pitches5$innings\_pitched),]

pitches5 <- pitches5[!is.na(pitches5$plate\_appearances),]

#########

## Machine Learning Models ##

# log loss function

LogLoss <- function(pred, res){

(-1/length(pred)) \* sum (res \* log(pred) + (1-res)\*log(1-pred))

}

# logistic regression

set.seed(3)

# remove id and other non-model features

pitches6 = pitches5 %>% dplyr::select(-c(batter, pitcher, events, inning, BatterName, plate\_appearances, PitcherName, innings\_pitched))

# unlist "list" variables

pitches7 <- pitches6 %>% unnest(avg\_EV) %>% unnest(avg\_angle) %>% unnest(avg\_EV\_allowed) %>% unnest(avg\_angle\_allowed) %>% unnest(avg\_velo)

# Remove rows with any NAs

hr\_data <- hr\_data[complete.cases(hr\_data),]

# 10 fold CV

set.seed(3)

n = dim(hr\_data)[1]

nfolds = 10

groups = rep(1:nfolds, n)

cvgroups = sample(groups, n)

# initialize prediction vector

CVpredictions = numeric(length = n)

for(ii in 1:nfolds) {

groupii = (cvgroups == ii)

trainset = hr\_data[!groupii,]

testset = hr\_data[groupii,]

# Logistic regression model

fit <- glm(home\_run ~ stand + p\_throws + avg\_EV + avg\_angle + avg\_EV\_allowed + avg\_angle\_allowed + avg\_velo + count + outs\_when\_up,

data = trainset, family = 'binomial')

predicted = predict(fit, newdata = testset, type = "response") # predict for test set

CVpredictions[groupii] = predicted

}

# calculate log loss - logistic regression

LogLoss(CVpredictions, hr\_data$home\_run)

# final model to test on new predictions

final\_model\_fit <- glm(home\_run ~ stand + p\_throws + avg\_EV + avg\_angle + avg\_EV\_allowed + avg\_angle\_allowed + avg\_velo + count + outs\_when\_up,

data = hr\_data, family = 'binomial')

hr\_data$hr\_prob <- CVpredictions

log\_model\_data <- hr\_data %>% dplyr::select(home\_run, hr\_prob) # data for model assessment

# class outcome distribution

classes\_tbl <- tibble(Outcome = c("No Home Run", "Home Run"), Proportion = as.numeric(prop.table(table(hr\_data$home\_run))))

classes\_gt <- gt(classes\_tbl) %>% tab\_header(title = "Table 1. Home Run Class Imbalance", subtitle = "proportion of batted balls that were home runs (2021 season)") %>%

fmt\_number(columns = "Proportion", decimals = 3) %>% cols\_align(align="center")

classes\_gt

# histogram of predicted probs - Logistic Regression

ggplot(log\_model\_data, aes(x=hr\_prob)) +

geom\_histogram(data=subset(log\_model\_data,home\_run==0), aes(fill=factor(home\_run)),alpha=0.5) +

geom\_histogram(data=subset(log\_model\_data,home\_run==1), aes(fill=factor(home\_run)),alpha=0.5) + theme\_minimal() +

labs(title = "Figure 1. Predicted Probabilities of Hitting a Home Run - Logistic Regression", x = "Predicted Probability", y = "Count") +

xlim(c(0, .25)) +

theme(plot.title = element\_text(size=22), axis.title.x = element\_text(size=15), axis.title.y = element\_text(size=15), text = element\_text(size=20),

legend.position = c(.75, .5), legend.background = element\_rect(fill="light grey", size=2, linetype="solid", color = "light grey"),

legend.title = element\_text(size=15, face="bold")) + scale\_fill\_manual(name="Home Run", values = c("red","green"), labels=c("No", "Yes"))

# decision thresholds

thresholds <- seq(0, 1, by=.01)

# precision and recall lists

prec\_list = numeric()

rec\_list = numeric()

F\_score\_list = numeric()

for(i in thresholds) {

# re-classify observations based on threshold

# confusion matrix

conf\_mat <- as.matrix(table(factor(testing$bayes\_prob > i, levels=c(F, T)), testing$home\_run))

# precision

precision = conf\_mat[4] / (conf\_mat[4] + conf\_mat[2])

# recall

recall = conf\_mat[4] / (conf\_mat[4] + conf\_mat[3])

# F score

F\_score = 2 / ((1/recall) + (1/precision))

# append to lists

prec\_list = c(prec\_list, precision)

rec\_list = c(rec\_list, recall)

F\_score\_list = c(F\_score\_list, F\_score)

}

# optimal threshold logistic regression

opt\_ind = which.max(F\_score\_list)

# confusion matrix at optimal threshold (GT package)

as.matrix(table(factor(testing$bayes\_prob > thresholds[opt\_ind], levels=c(F, T)), testing$home\_run))

log\_tbl = tibble(Predicted = c("Non HR", "HR"), "Non HR"=c(15190,2853), HR=c(692, 230))

gt(log\_tbl) %>% tab\_spanner(label="Actual Outcome", columns=c("Non HR", "HR")) %>%

tab\_header(title = "Table 3. Confusion Matrix", subtitle="Naive Bayes Model with p=.07 Threshold") %>% cols\_align(align="center") %>%

fmt\_number(columns=2:3, use\_seps = TRUE, decimals = 0)

# gg plot Precision-Recall Curve Logistic Regression

# used package to find AUC for PR curve

hr <- log\_model\_data$hr\_prob[log\_model\_data$home\_run==1]

no\_hr <- log\_model\_data$hr\_prob[log\_model\_data$home\_run==0]

pr <- pr.curve(scores.class0 = hr, scores.class1 = no\_hr, curve = T) #.0739

plot(pr)

# create data frame

plot\_df <- data.frame(p=prec\_list, r=rec\_list, f=F\_score\_list)

# plot PR curve with AUC pulled from above plot

ggplot(plot\_df, aes(r, p)) + geom\_point(size=3, color = "#454545") +

labs(title = "Figure 2. Precision-Recall Curve", x = "Recall", y = "Precision", subtitle = "AUC = .0739; optimal threshold = .07") + theme\_minimal() +

scale\_y\_continuous(breaks=seq(0,1,.1)) + scale\_x\_continuous(breaks=seq(0,1,.1)) +

geom\_point(aes(x=r[opt\_ind], y=p[opt\_ind]), pch=18, size=8, color='#ff0000') +

theme(plot.title = element\_text(size=22), axis.title.x = element\_text(size=15), axis.title.y = element\_text(size=15),

text = element\_text(size=17))

### Naive Bayes ###

hr\_data <- pitches7[!is.na(pitches7$avg\_EV\_allowed),]

# combine pitcher-batter matchup

bayes\_data <- hr\_data %>% dplyr::select(-c(venue\_name, temperature, WindCategory, WindSpeed, WindDirection))

# one hot encoding for categorical predictors

bayes\_data\_encoded <- bayes\_data %>% mutate(ID = seq(1, nrow(bayes\_data), 1)) # add ID variable for melt

bayes\_data\_encoded <- bayes\_data\_encoded %>% dplyr::select(stand, p\_throws, count, outs\_when\_up, ID) # only categorical variables

bayes\_data\_encoded <- dcast(data = melt(bayes\_data\_encoded, id.vars = "ID"), ID ~ variable + value, length) # one-hot encode

bayes\_data\_final <- cbind(bayes\_data\_encoded, bayes\_data %>% dplyr::select(-c(stand, p\_throws, count, outs\_when\_up)))

# correlation

cor(bayes\_data\_final[,-c(1,21)])

#split data into training and test data sets

indxTrain <- createDataPartition(y = bayes\_data$home\_run,p = 0.75,list = FALSE)

training <- bayes\_data[indxTrain,]

testing <- bayes\_data[-indxTrain,]

x <- training[,-5] # no ID or response variable

y <- training$home\_run

model = train(x,factor(y),'nb',trControl=trainControl(method='cv',number=10))

bayes\_pred <- predict(model, newdata=testing, type="prob")

testing$bayes\_prob <- bayes\_pred[,2]

LogLoss(testing$bayes\_prob, testing$home\_run)

# histogram of predicted probs - Naive Bayes

ggplot(testing, aes(x=bayes\_prob)) +

geom\_histogram(data=subset(testing,home\_run==0), aes(fill=factor(home\_run)),alpha=0.5) +

geom\_histogram(data=subset(testing,home\_run==1), aes(fill=factor(home\_run)),alpha=0.5) + theme\_minimal() +

labs(title = "Figure 3. Predicted Probabilities of Hitting a Home Run - Naive Bayes", x = "Predicted Probability", y = "Count") +

xlim(c(0, .25)) +

theme(plot.title = element\_text(size=22), axis.title.x = element\_text(size=15), axis.title.y = element\_text(size=15), text = element\_text(size=20),

legend.position = c(.75, .5), legend.background = element\_rect(fill="light grey", size=2, linetype="solid", color = "light grey"),

legend.title = element\_text(size=15, face="bold")) + scale\_fill\_manual(name="Home Run", values = c("red","green"), labels=c("No", "Yes"))

# gg plot Precision-Recall Curve naive bayes

# used package to find AUC for PR curve

hr <- testing$bayes\_prob[testing$home\_run==1]

no\_hr <- testing$bayes\_prob[testing$home\_run==0]

pr <- pr.curve(scores.class0 = hr, scores.class1 = no\_hr, curve = T) #.0703

plot(pr)

# create data frame

plot\_df <- data.frame(p=prec\_list, r=rec\_list, f=F\_score\_list)

# plot PR curve with AUC pulled from above plot

ggplot(plot\_df, aes(r, p)) + geom\_point(size=3, color = "#454545") +

labs(title = "Figure 4. Precision-Recall Curve", x = "Recall", y = "Precision", subtitle = "AUC = .0703; optimal threshold = .07") + theme\_minimal() +

scale\_y\_continuous(breaks=seq(0,1,.1)) + scale\_x\_continuous(breaks=seq(0,1,.1)) +

geom\_point(aes(x=r[opt\_ind], y=p[opt\_ind]), pch=18, size=8, color='#ff0000') +

theme(plot.title = element\_text(size=22), axis.title.x = element\_text(size=15), axis.title.y = element\_text(size=15),

text = element\_text(size=17))

# log loss table

log\_loss\_tbl = tibble(Model = c("Logistic Regression", "Naive Bayes"), "Log Loss"=c(.1914,.1922))

gt(log\_loss\_tbl) %>%

tab\_header(title = "Table 4. Log Loss") %>% cols\_align(align="center")

# Logistic Regression Model Coefficients Interpretation table

# Logit to probability function source: ("https://sebastiansauer.github.io/Rcode/logit2prob.R")

logit2prob <- function(logit){

odds <- exp(logit)

prob <- odds / (1 + odds)

return(odds)

}

logit2prob(coef(final\_model\_fit))

coef\_tbl = tibble(Variable = c("BatHand Right", "Exit Velo", "Launch Angle", "Launch Angle Allowed", "Count 0-2", "Count 2-0", "Count 3-1"),

Coefficient=c(.145,.132,.0416,.0261,-.4245,.3916,.4495),

Odds=c(1.16, 1.14, 1.04, 1.03, .65, 1.48, 1.57)) %>% arrange(desc(Odds))

gt(coef\_tbl) %>%

tab\_header(title = "Table 5. Logistic Regression Significant Coefficients", subtitle="with corresponding odds change") %>% cols\_align(align="center")

# LIVE HOME RUN PROBABILITY

# Get weather data from game

get\_weather\_data <- function(game\_pk) {

weather = get\_game\_info\_mlb(game\_pk=game\_pk) %>% dplyr::select(game\_pk, game\_date, venue\_name, temperature, other\_weather, wind)

return(tibble(weather))

}

# update statcast data by searching game date (would be today's date) and home team

get\_live\_probs <- function(date, home\_team) {

# get game packs for date

game\_packs <- baseballr::get\_game\_pks\_mlb(date=date, level\_ids = 1)

# if there are live games, choose 1 game

one\_game\_pack <- game\_packs %>% filter(status.detailedState == "In Progress", teams.home.team.name==home\_team) %>% pull(game\_pk)

# Get statcast pbp data for that game

one\_game <- baseballr::get\_pbp\_mlb(game\_pk = one\_game\_pack)

#get\_weather\_data(one\_game)

return(tibble(one\_game))

}

# filter live statcast game data

filter\_statcast\_game <- function(tbl\_obj) {

filtered <- tbl\_obj %>% dplyr::select(about.inning, result.awayScore, result.homeScore, matchup.batter.id, matchup.batter.fullName,

matchup.pitcher.fullName, matchup.pitcher.id, count.strikes.start, count.balls.start,

matchup.batSide.code, matchup.pitchHand.code, count.outs.start, home\_team, away\_team, batting\_team,

fielding\_team, result.event, atBatIndex, pitchNumber)

# convert team names to formatted strings following filename format

filtered <- filtered %>% dplyr::mutate(home\_team = gsub(" ", "-", tolower(home\_team)),

away\_team = gsub(" ", "-", tolower(away\_team)))

# Rename columns

filtered <- filtered %>% dplyr::rename(stand = matchup.batSide.code, p\_throws=matchup.pitchHand.code, batterID=matchup.batter.id,

pitcherID=matchup.pitcher.id, outs\_when\_up=count.outs.start) %>%

mutate(count=paste(count.balls.start, count.strikes.start, sep="\_")) %>% dplyr::select(-c(count.balls.start, count.strikes.start))

return(filtered)

}

# IMPUTE MISSING DATA (rookies, players with insufficient data, etc.) with MEDIAN STATS

# GET MEDIAN STATISTICS

# Average exit velocity by hitter

median\_ev <- SavantData19 %>% drop\_na(launch\_speed) %>% dplyr::summarise(median\_EV = median(launch\_speed))

# Launch angle by hitter

median\_angle <- SavantData19 %>% drop\_na(launch\_angle) %>% dplyr::summarise(median\_angle = median(launch\_angle))

# EV allowed -- pitcher

median\_ev\_allowed <- median\_ev

# Launch angle allowed -- pitcher

median\_angle\_allowed <- median\_angle

# average Release speed -- pitcher

median\_velocity <- SavantData19 %>% drop\_na(release\_speed) %>% dplyr::summarise(median\_velo = median(release\_speed))

# merge data data

merge\_stats <- function(live\_data) {

# filter

filtered <- filter\_statcast\_game(live\_data)

# merge hitter data

merge1 <- left\_join(filtered, hitters\_df\_all %>% dplyr::select(batter, avg\_EV, avg\_angle), by=c("batterID"="batter"))

# merge pitcher data

merge2 <- left\_join(merge1, pitchers\_df\_all %>% dplyr::select(pitcher, avg\_EV\_allowed, avg\_angle\_allowed, avg\_velo), by=c("pitcherID"="pitcher"))

return(merge2)

}

# predict pitch

predict\_pitch <- function(live\_game\_data, live=TRUE) {

# merge

if(live==TRUE){

play <- merge\_stats(live\_game\_data)[1,] # only top record if live

}

else {

play <- merge\_stats(live\_game\_data) # all game records if not live

# remove invalid counts

play <- play %>% filter(!count %in% c("4\_0", "4\_1", "4\_2", "0\_3", "1\_3", "2\_3", "3\_3"))

}

# impute missing values

# Replace NAs

play$avg\_EV[is.na(play$avg\_EV)] <- as.numeric(median\_ev)

play$avg\_angle[is.na(play$avg\_angle)] <- as.numeric(median\_angle)

play$avg\_EV\_allowed[is.na(play$avg\_EV\_allowed)] <- as.numeric(median\_ev\_allowed)

play$avg\_angle\_allowed[is.na(play$avg\_angle\_allowed)] <- as.numeric(median\_angle\_allowed)

play$avg\_velo[is.na(play$avg\_velo)] <- as.numeric(median\_velocity)

new\_pred <- predict(final\_model\_fit, newdata=play, type = "response")

# data frame with useful info and prediction

pred\_df <- data.frame(play, new\_pred)

return(pred\_df)

}

# run this code for live games \*\*only works if there are live games (set detailedState=="Final" in line 436 if no live games)

live\_game\_data <- get\_live\_probs("2022-08-03", "Minnesota Twins")

live\_pitch <- predict\_pitch(live\_game\_data)

all\_live\_pitches <- rbind(all\_live\_pitches, live\_pitch)

####

# Retroactive testing

random\_dates <- c("2022-04-25", "2022-04-29", "2022-05-16", "2022-07-01", "2022-07-17")

# get game packs from random date

game\_pks\_random\_date <- baseballr::get\_game\_pks\_mlb(date=random\_dates[4], level\_ids = 1) %>% pull(game\_pk)

# Pick random game from random date

random\_game <- sample(game\_pks\_random\_date, 1)

# Get pbp data from that game

pbp\_random <- tibble(baseballr::get\_pbp\_mlb(game\_pk = random\_game) %>% filter(type=="pitch"))

# predict probs for random game

probs\_random <- predict\_pitch(pbp\_random, live=FALSE)

# remove duplicates

probs\_random <- probs\_random %>% distinct() %>% arrange(atBatIndex, pitchNumber) %>% mutate(pitchIndex = row\_number())

# plot game hr probability

# title

game\_title <- paste0(probs\_random$batting\_team, " vs. ", probs\_random$fielding\_team, " HR Probability by Pitch")[1]

# label index for conditionally annotated points

probs\_random <- probs\_random %>% mutate(labelIndex = ifelse(new\_pred < .01, 1, 0))

# at-bat description

probs\_random <- probs\_random %>% mutate(ab\_desc = paste0(matchup.batter.fullName, " ", result.event, " on a ", count, " count"))

# plots random game HR probability by pitch

ggplot(probs\_random, aes(pitchIndex, new\_pred)) + geom\_line(color = "#454545", size=2) + theme\_minimal() +

labs(title=game\_title, x="Pitch Number", y="HR Probability", subtitle = random\_dates[4]) +

theme(plot.title = element\_text(size=22), axis.title.x = element\_text(size=15), axis.title.y = element\_text(size=15), text = element\_text(size=17)) +

geom\_text(aes(pitchIndex, new\_pred, label=ab\_desc), data = probs\_random[probs\_random$labelIndex==1,],

nudge\_x = -25, nudge\_y = .005, fontface="bold", color="#EF4444", size=5)

**Appendix B**

**R Shiny Code for HR Probability Display**

# R Shiny application that displays results of logistic regression HR probability model

# Load packages

library(shiny)

# UI

ui <- fluidPage(

titlePanel("Scoreboard"),

fluidRow(

column(4,

h1("Outs"),

h3(uiOutput("outs"))

),

column(4,

h1("Balls"),

h3(uiOutput("balls"))

),

column(4,

h1("Strikes"),

h3(uiOutput("strikes"))

)

),

br(),

fluidRow(

column(4,

h1("Pitching"),

h3(uiOutput("pitching\_info"))

),

column(4,

h1("At-Bat"),

h3(uiOutput("hitting\_info"))

),

column(4,

h1("HR Probability"),

h3(uiOutput("hr\_prob"))

)

),

br(),

fluidRow(

column(4,

h1("Inning:")

),

column(4,

h1(uiOutput("away\_score"))

),

column(4,

h1(uiOutput("home\_score"))

)

),

fluidRow(

column(4,

h1(uiOutput("inning"))

),

column(4,

imageOutput("Away\_Team\_Image")

),

column(4,

imageOutput("Home\_Team\_Image")

)

)

)

# Server

server <- function(input, output, session) {

output$inning <- renderText({

live\_pitch$about.inning

})

output$away\_score <- renderText({

live\_pitch$result.awayScore

})

output$home\_score <- renderText({

live\_pitch$result.homeScore

})

output$pitching\_info <- renderText({

paste0(live\_pitch$matchup.pitcher.fullName, " (", live\_pitch$p\_throws, ")")

})

output$hitting\_info <- renderText({

paste0(live\_pitch$matchup.batter.fullName, " (", live\_pitch$stand, ")")

})

output$hr\_prob <- renderText({

paste(round(live\_pitch$new\_pred, 3))

})

output$Home\_Team\_Image <- renderImage({

filename = normalizePath(file.path("./www/", paste(paste0("mlb-", live\_pitch$home\_team, "-logo", sep=""), ".png", sep="")))

return(list(src = filename, alt=live\_pitch$home\_team, width=100, height=100))

}, deleteFile = FALSE)

output$Away\_Team\_Image <- renderImage({

filename = normalizePath(file.path("./www/", paste(paste0("mlb-", live\_pitch$away\_team, "-logo", sep=""), ".png", sep="")))

return(list(src = filename, alt=live\_pitch$away\_team, width=100, height=100))

}, deleteFile = FALSE)

output$balls <- renderText({

sub("\_.\*", "", live\_pitch$count)

})

output$strikes <- renderText({

sub(".\*\_", "", live\_pitch$count)

})

output$outs <- renderText({

live\_pitch$outs\_when\_up

})

}

shinyApp(ui, server)