Exploring Factors Influencing On-Base Percentage in Modern Baseball

Heidar Gergis

Bachelor's thesis
2024:K20

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematical Statistics

Centrum Scientiarum Mathematicarum

Bachelor's Theses in Mathematical Sciences 2024:K20
ISSN 1654-6229

LUTFMS-4014-2024

Mathematical Statistics
Centre for Mathematical Sciences
Lund University
Box 118, SE-221 00 Lund, Sweden

http://www.maths.lu.se/

Exploring Factors Influencing On-Base Percentage in Modern Baseball

Heidar Gergis

Abstract

This report uses highly detailed data from Statcast from the 2015–2023 seasons to explore and identify the key factors influencing on-base percentage in the American top league, Major League Baseball. On-base percentage is simply the percentage of times a player gets on base, and can be achieved in multiple ways. It is an important metric in baseball, as without getting on base, you cannot score. Using machine learning techniques, we aim to identify these factors and develop models with predictive power. This highly detailed data, with tens of features and more than 1 million rows of tracked events, was then used to develop and implement different machine learning models, such as logistic regression and XGBoost. These models were trained and then tested for performance that took into account the imbalance in the data, such as the F2 score and area under the precision-recall curve (AUC-PR). To aid in the interpretation of the models, SHAP (SHapley Additive exPlanations) values were used to provide insight. Our results show that the XGBoost models significantly outperform the logistic regression model in terms of both F2 score and AUC-PR, achieving high scores of 90.50% and 95.82%, respectively. This can be contrasted with the respective 77.83% and 68.42% for the logistic regression model. We also find that the XGBoost model can be greatly reduced with a feature-selected model, with less than a third of the variables achieving near-identical scores (89.60% and 94.90%, respectively). Feature importance and SHAP analysis showed that factors such as hit location, launch angle, ball-strike count difference, and whether contact was made were the most important and influential factors.

Table of contents

Acknowledgement

I would like to express gratitude to my supervisor, Dr. Linda Hartman, for her guidance and flexibility in the course of this project, allowing me to balance this thesis with my job. I would also like to thank my girlfriend Lisa and my family, for their support throughout this degree project.

Glossary

At bat (AB): The batter's turn at batting

AUC-PR: Area Under the Precision-Recall Curve, a performance measurement for classification problems at various thresholds.

Ball: A pitch outside the strike zone that the batter does not swing at.

Barrel: A batted ball with the optimal combination of exit velocity and launch angle, typically resulting in high-quality contact.

Base: The four points that the batter must touch to score a run, numbered in the order they are placed

Batter: The player up to bat

Batting average (AVG): The number of hits divided by the total number of at-bats, typed as a decimal between .000 and 1.000

Double (2B): A hit where the batter reaches second base.

F2 score: A measure of predictive performance that places more emphasis on recall than precision.

Fielding alignment: The positioning of defensive players on the field.

Foul ball: A ball hit outside the foul lines.

Foul lines: The two lines that meet at a 45 degree angle and are the boundaries of play-

Hard hit: Any ball hit harder than 95 mph (153 km/h).

Hit (H): When a batter makes contact with the ball and reaches base safely.

Hit by pitch (HBP): When the batter gets hit by the ball. Allows free passage to first base.

Home run (HR): A hit between the foul lines and over the center-field fence. Allows the batter and any runners to circle the bases and score runs.

Inning: One turn for each team to both bat and field. A baseball game is comprised of 9 innings.

Launch angle: The vertical angle at which the ball leaves the bat after being hit.

Launch speed: The speed at which the ball leaves the bat after being hit, also known as exit velocity.

LIPS (Late Inning Pressure Situation): A situation in the 7th inning or later where the team is either ahead by one run or behind by three runs or fewer.

Logistic Regression: A statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome.

On-base percentage (OBP): Refers to how frequently a batter reaches base per plate appearance.

Out (O): When the fielding team retires a batter or runner, they lose the ability to score a run until their next turn at bat.

Pitcher (P): The player who throws the ball to the batter.

RISP (Runners in Scoring Position): Runners on second or third base, considered in scoring position because they can typically score on a single.

Run (R): A point scored by a player who advances around all the bases and returns safely to home plate.

Runner: Players on base.

Scoring position: Defined as being on second or third base. It is called scoring position as one can score on a single which is usually impossible to do from first base.

SHAP (SHapley Additive exPlanations): A game theoretic approach to explain the output of any machine learning model.

Single: A hit where the batter reaches single base.

Spray angle: The horizontal angle of the ball when contact is made.

Statcast: A tracking technology that allows for the collection of highly detailed data on various aspects of the game, including pitch velocity, spin rate, and batted ball metrics.

Strike: A pitch inside the strike zone or any pitch that the batter swings at.

Strikeout (K): When the batter gets three strikes at an at bat. The batter is then out.

Strike zone: An imaginary box with the width of the home plate and the height is defined as from the hollow beneath the kneecap to the midpoint between the batter's shoulders and the top of the uniform pants

Sweet spot: Any ball hit at a launch angle between 8 and 32 degrees.

Triple (3B): A hit where the batter reaches third base.

Walk (BB): A walk (or base on balls) occurs when a pitcher throws four pitches out of the strike zone, none of which are swung at by the hitter. Awards free passage to first base.

XGBoost: eXtreme Gradient Boosting, a tree-based machine learning algorithm.

Chapter 1

Introduction

1.1 Introduction to Baseball

Baseball is one of America's most popular sports. Having been created in the 1800s, it has grown to be a global sport with players and leagues in countries in East Asia, and the rest of North America [Nielsen Company, 2023].

It is a sport that is different from others in that it does not have a game clock and play is not continuous; rather, the game consists of discrete events called plate appearances.

Baseball is played on a field, often called the diamond.

In Figure 1.1 one can see the layout of a baseball field. It consists of an infield that is a square with sides of 90 feet (27.4 m) with bases on each corner. This square is covered in grass. The area just outside this square, known as the edge of the infield, is composed of sand. Beyond the infield is the outfield, a large grass area that is the end of the field. The boundaries of play are the foul lines, which extend from home plate at 45-degree angles all the way to the poles on each end, the left-field and right-field poles. This forms a 90-degree area within these foul lines. The last boundary of play is the center-field fence between the two foul poles. The minimum distances shown in the image are due to the fact that each baseball field can look different with different boundaries. It is preferred that the fields fulfill the requirements set forth, with distances of at least 325 ft (99 m) for the foul

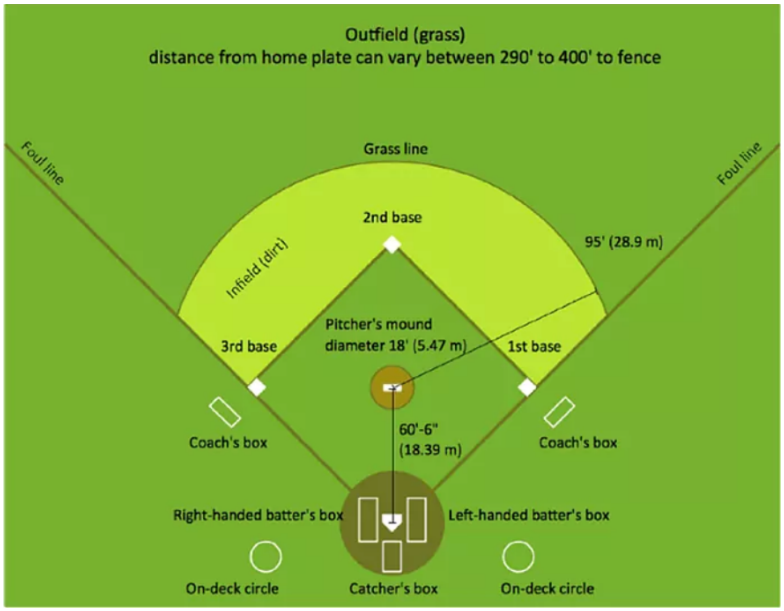


Figure 1.1: Dimensions of a baseball field [Marca, 2022]

lines, and the furthest point of the center-field fence must be more than 400 ft (122 m) from home plate. However, these rules are not strictly enforced, as long as the boundary dimensions are close to these values [Burnes et al., 2024]. This results in baseball fields with differing dimensions, as Figure 1.2 shows the different dimensions of two of the baseball fields in use.

The sport is played between two teams, each with nine players. The goal is to score more runs than the other team. To score a run, a player, the batter, must essentially hit the ball into play and traverse the bases in counterclockwise fashion. The ball must be hit between the two foul lines; otherwise, it is called a "foul ball". The batter starts at home plate, where the pitcher on the fielding team, standing on the pitcher's mound approximately 18.5 m away, throws a ball that the batter tries to make contact with. The event is called a plate appearance. The pitcher's goal is to throw the ball toward the strike zone so that the batter cannot hit it. The strike zone, as displayed in Figure 3.4, is an imaginary box with the width of the home plate, and the height is defined as from the hollow beneath the kneecap to the midpoint between the batter's shoulders and the top of the uniform pants [Burnes et al., 2024].

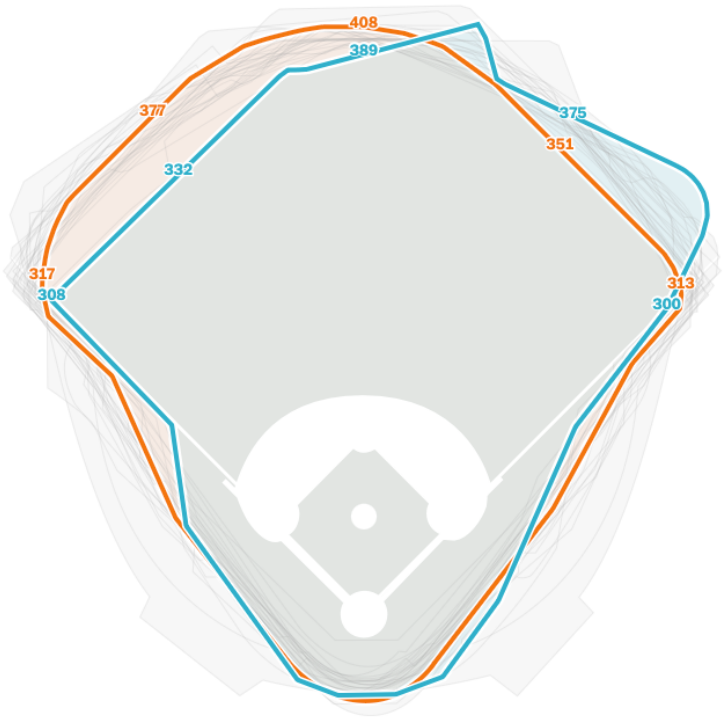


Figure 1.2: The different field dimensions of two baseball fields [Schaul, 2023].

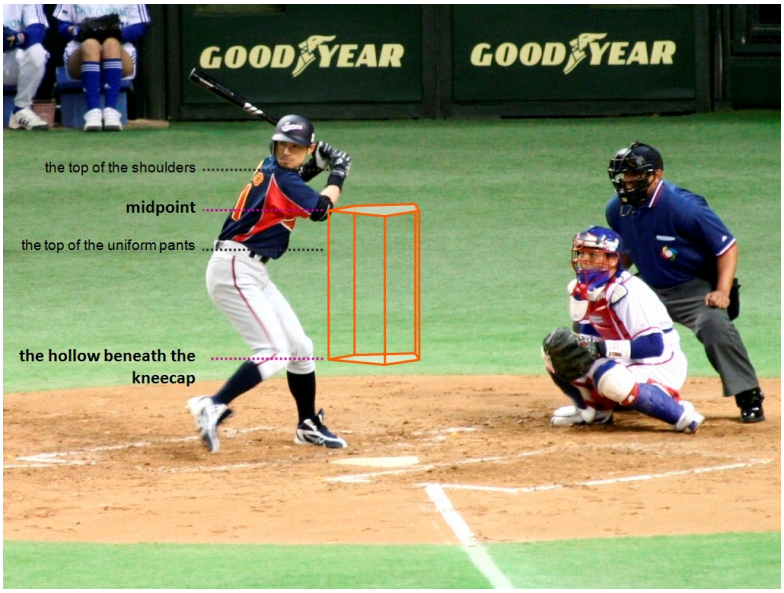


Figure 1.3: Image with a drawing of the strike zone superimposed [Chan, 2008].

A pitch thrown through this area is called a strike. A pitch thrown anywhere that the batter swings and misses is also called a strike. A foul ball is also called a strike, but only if it is the first or second strike. If a batter accumulates three strikes, then they are called out; this specific type of out is called a strikeout. If the ball is thrown outside the strike area without the batter swinging at it the pitch is called a ball. If the pitcher accumulates four balls, then the batter gets to walk freely to first base, this is referred to as a walk or bases on balls. Also, if the batter gets struck directly by a pitch; this is a hit by pitch, and the batter again gets to walk freely to first base.

If and when the batter hits the ball into play, he must then drop the bat and run toward first base. At this point, he becomes a runner. If he reaches the first base before an opponent touches the base with the ball in their hand, the runner is safe. Doing this successfully is called getting a single. And if the batter makes it to second base as a result of the play, that is called a double, and reaching third base as a direct result of the play is called a triple. If the batter hits the ball between the foul lines and over the center-field fence, this is called a home run, and the batter and any runner on base get to freely circle the bases and score runs for their team. These four events;

single, double, triple, and home run are all collectively referred to as hits, not to be confused with the word hit, as in “hitting the ball”.

The fielding team tries to prevent this by getting outs. As previously mentioned, three strikes equal an out,but there are other ways of getting outs. If the ball is caught in the air the batter will be out. If they are tagged with the ball whilst running between bases then they are out. There are more ways to be called out, but they are rare and out of the scope for this report. When the batter is out, he leaves the field and waits until his turn to bat is up again. Once the defense has made three outs, the teams switch sides and the fielding team bats and vice versa. When both teams have batted, this is called an inning. Baseball is typically played over nine innings.

There are no ties, so if the score is even at the end of these nine innings, extra innings are played until a winner is decided. Baseball does not utilize a game clock. The game is over only after nine innings, however long that may take. There is no way to waste time to run out the clock, a team has to make 27 outs and outscore their opponent to win [Burnes et al., 2024].

1.2 The Rise of Statistical Analysis in Baseball

This, coupled with the fact that play comes at these discrete chunks of plate appearances, makes the sport very apt for statistical analysis. Baseball is perhaps the sport with the most statistics and has led the way in sports in using statistics and analytics to guide decisions on squads and game tactics. Other sports, like football, are continuous matches where it is more difficult to extract distinct events. But baseball, like “brännboll” or rounders, which you may have played as a child, is made up of events where one person comes up and tries to hit a ball with a piece of wood thrown by a pitcher. Then the next person comes up and tries to do the same thing. These discrete events make it easier to keep statistics and have also led to many statistical measures such as batting average and on-base percentage, and more advanced statistics such as wins above replacement (WAR), which attempts to quantify the amount of wins a player generates compared to a replacement level player [McLeod, 2023].

This is aided by the fact that since 2015, all stadiums in the American top

league, Major League Baseball now have radar, lidar, and optical tracking systems installed that track plays with high accuracy. The system allows for tracking with very high precision, everything from pitch speed to launch angle to hit distances. This change is so significant that the time since is often referred to as the Statcast era and has started what some in the sport have called an "arms race" of data analysis [A. Chen, 2016].

1.3 Problem Formulation

The most commonly used metric for offensive success for most of the 20th century was the batting average. The batting average is defined as the number of hits divided by the total number of at-bats. At-bats do not include all the batter's appearances at the plate. Some events, like walks and hit by pitches, do not count toward at-bats. At-bats are intended to be plays in which the batter tries to put the ball in play, thus the omission of some events as mentioned above. However, batting average does not take into account other ways of reaching base outside of hits, such as walks. With the rise of statistical analysis in the sport, it was realized that often it was reaching base that was important, not how you reach base. If you can not get on base, you can not score runs. However, you could score runs without hits. And even though more and more complex metrics are being used, a simple one that is highly valued by statistical professionals in the field is on-base percentage (OBP) [Knight, 2009]. On-base percentage is simply the rate at which a batter reaches base per plate appearance. This simplifies the actual calculation a bit because not all plate appearances are included in the calculation. The actual calculation for OBP is:

$$\frac{H + BB + HBP}{AB + BB + HBP + SF} \tag{1.1}$$

where

H = hits

BB = walks

HBP = hit by pitches

AB = at bats

SF = sacrifice flies

The denominator is very similar to plate appearance but does exclude certain infrequent events that the batter can not affect but would negatively affect the OBP. What is often explored with OBP is its effect and correlation to scoring runs, but it is not often explored what factors and variables influence and affect a player's and the sport as whole, OBP. Therefore, the goal of this report is to investigate and explore these factors with the aid of statistical analysis. This thesis aims to identify the key metrics that impact OBP and getting on base. As a lot of different events are included in the calculation for on-base percentage, building a predictive model can be challenging, but it is the goal of this thesis to build a model that can also have predictive power.

Chapter 2

Theory

2.1 Dimensionality Reduction Techniques

Dimensionality reduction is a highly desirable technique for statistical models of large sizes and high dimensionality. Without dimensionality reduction or feature selection, we risk overfitting data and having a model that is too complex for the task at hand. Such a model will not perform well on new unseen data. We also risk the curse of dimensionality. This refers to the fact that after a certain number of dimensions, a model's predictive power will decrease rather than increase [Altman & Krzywinski, 2018].

Dimensionality reduction can be achieved within the process of fitting a prediction model, e.g. using LASSO regression (see Section 2.3.2). Dimensionality reduction techniques can, however, also provide valuable early insights about the data and are often performed as part of the exploratory data analysis. There are many techniques that can be applied in the case of dimensionality reduction, but two common ones are Principal Component Analysis (PCA) and Uniform Manifold Approximation and Projection (UMAP).

2.1.1 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is perhaps the most popular technique for reducing dimensionality. The idea with PCA is to create new variables that are linear combinations of the original variables, such that these new

variables explain most of the variance. These new variables are called principal components, and they are ordered so that the first principal component, denoted PC1, explains most of the variance, PC2 explains the second most, and so on. Worth noting is that PC2 is in the direction perpendicular to PC1, and PC3 is in the direction perpendicular to PC1, PC2 et cetera. An example of PCA and the first two principal components can be seen in Figure 3.11.

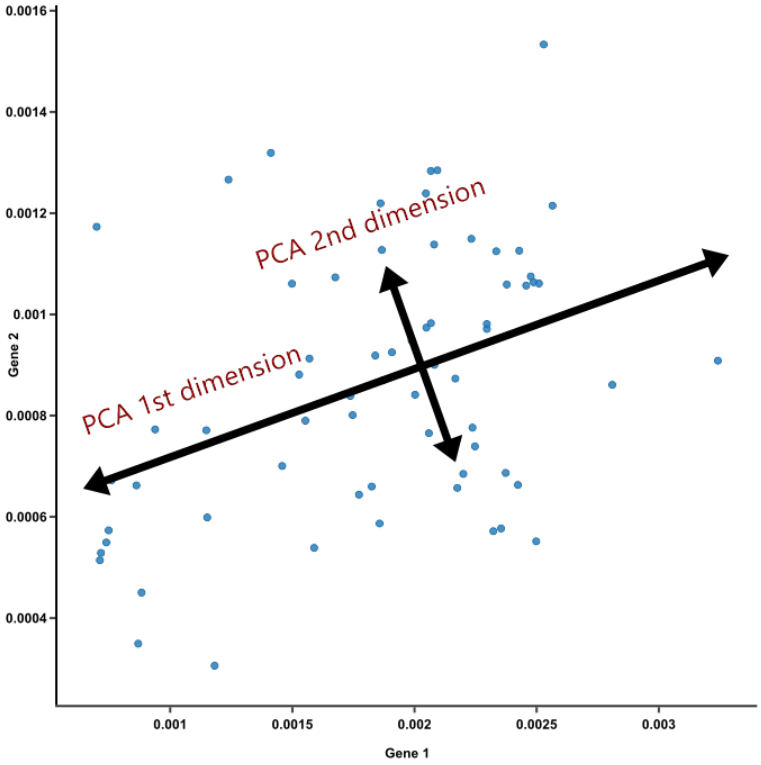


Figure 2.1: An example plot of data with the first two principal components overlaid [Ngo, 2018].

If we consider a dataset $\mathbf{X}$ with p features, where X_p is the p-th feature, PCA then represents a change of basis to $\mathbf{Z}$ with m features where $m < p$ so for the i-th principal component we get:

$$\mathbf{Z}_i = \phi_{i1}X_1 + \phi_{i2}X_2 + \cdots + \phi_{ip}X_p$$

where ϕ_{ij} are the weights for each X-variable in the newly created Z-basis. To

create the principal components that aim to maximize the variance explained, the covariance matrix is used [Shlens, 2014]. First the data is centered and scaled. Then the estimated covariance matrix $\mathbf{C}$, a $p \times p$ matrix is calculated:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X}$$

On $\mathbf{C}$ eigendecomposition is performed to find both the eigenvalues and eigenvectors. The eigenvectors of the covariance matrix are the principal components and each eigenvalue represents the variance explained by the eigenvector. Since covariance matrices are symmetric, the eigenvectors are orthogonal to each other which is desired and the eigendecomposition becomes $\mathbf{C} = \mathbf{PDP}^T$ where $\mathbf{P}$ are the eigenvectors and $\mathbf{D}$ is a diagonal matrix of the eigenvalues $\lambda_1, \lambda_2 \ldots, \lambda_p$. The eigenvalues and eigenvectors are ordered so that:

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_p$$

Since the maximum number of eigenvalues for a $p \times p$ matrix is p, and we have p features, the sum of the eigenvalues

$$\sum_{i=1}^{p} \lambda_i \tag{2.1}$$

explain all the variance in the dataset. However, if we choose to project the data onto a subset M of the eigenvectors where $M < p$, then the retained variance can be written as:

$$\sum_{i=1}^{M} \lambda_i \tag{2.2}$$

This means that the retained variance can also be expressed as a percentage of the total variance:

$$\%_{var} = \frac{\sum_{i=1}^{M} \lambda_i}{\sum_{i=1}^{p} \lambda_i} \cdot 100 \tag{2.3}$$

This allows us to choose the number of principal components to retain based on the percentage of variance we want to retain, generally this is set to 90-95%. If the PCA is successful, then we can retain the vast majority of the information with only a fraction of the complexity. However, PCA is a linear dimensionality reduction technique it relies on the model being linear and having linear correlations so if the linearity stipulation is not met, then PCA will not work and other techniques should be considered. Other techniques should also be considered if the majority of the variance is not captured by the first few principal components [Dong & McAvoy, 1996].

2.1.2 Uniform Manifold Approximation and Projection (UMAP)

One dimensionality reduction technique for non-linear data is Uniform Manifold Approximation and Projection, or UMAP. It is a popular technique for complex and high-dimensional datasets with non-linear structures. It was created by Leland McInnes et al. in 2018 and excels in speed and efficiency compared to other techniques for non-linear data. The aim is to preserve both the local and global structures of the data [McInnes, 2018].

As the term *manifold* indicates, UMAP uses ideas from the field of manifold learning and topology. A manifold is a topological space that resembles Euclidian space locally, like how a sphere locally and zoomed in looks like a flat plane. In this context, this is essentially describing high-dimensional data with a representation of lower dimensionality [Britannica, 2024]. The mathematical details, however, are complex and beyond the scope of this thesis. Instead, we will focus on the ideas and intuition behind UMAP. UMAP constructs an initial graph representation of the data, where each data point is a node and nodes are connected to each other according to some distance metric. This representation is high-dimensional. The graph is then used to optimize a low-dimensional graph to be, structurally, as similar as possible to the initial high-dimensional graph representation. A more in-depth explanation of UMAP with the mathematical details can be found in Section A.2.

For a dataset as large and complex as ours, with non-linear relationships between the features, UMAP is a powerful technique to find low-dimensional representations and reduce the dimensionality without compromising predictive performance. It is also rather fast and efficient, even with large datasets.

In this project, UMAP and PCA will be used as tools for exploratory data analysis as the reduction from a high-dimensional to a low-dimensional space while preserving the local and global structures of the data will allow us to identify clusters, patterns and relationships that may not be visible in the original data.

2.2 Binary Classification

Whether an event resulted in an on-base appearance or not is a binary classification problem. Binary classification problems are about estimating the probability that an event, Y, occurs or not. The two possible events of Y are often encoded as either 0 or 1 where 1 is the positive class and 0 is the non-positive class. In our case, 1 is that an event resulted in the hitter getting on base $p(x) = P(Y = 1|X = x)$.

2.3 Statistical Models

2.3.1 Logistic Regression

Logistic regression is one of the most commonly used binary classification methods and the logistic function it uses was developed in France in the 1830s and 1840s [Cramer, 2003]. Its use is facilitated by its ease of implementation and the interpretability it provides. To show its properties we can start with the classification problem itself. As previously mentioned, binary classification is trying to predict the outcome of a two-class categorical outcome and the probability of an event resulting in the positive class. Let

$$p(x) = P(Y = 1|X = x)$$

We could create a model to outright predict $p(x)$ but the probability is restricted to the range of 0 to 1 and with such a limited range it would be difficult to identify how the function would behave with a small change in one or more of the predictors. Instead, it is better to consider odds. We can write the odds, the probability of an event occurring compared to the

probability of an event not occurring as:

$$\text{odds} = \frac{p(x)}{1 - p(x)} \tag{2.4}$$

Taking the natural logarithm of the odds, results in log-odds or logit. In logistic regression the logit is

$$\text{logit}(p(x)) = \ln \frac{p(x)}{1 - p(x)} = \beta_0 + \beta_1 x_1 + ... + \beta_p x_p = \beta^T X \tag{2.5}$$

The logit transformation is easier to work with than the probability function as it may provide a linear relationship between the log-odds of an event and the predictors. This is useful as it allows us to use the same principles as linear regression models to estimate the coefficients, and we can also interpret the coefficients of the predictors easily as the change in log-odds per unit of increase in the predictor [Nahhas, 2024].

It is straightforward to derive the probabilities again from the logit function when it is time to interpret the results of the function:

$$p(x) = \frac{e^{\beta^T X}}{1 + e^{\beta^T X}} \tag{2.6}$$

The result is the so-called logistic function. It can take any real input $t = \beta^T X_i$ and output a value between 0 and 1 which means that is well suited for the task of binary classification, as Figure 2.2 shows.

To estimate the parameters of the logistic function maximum likelihood estimation is used. As the name suggests, the objective is to maximize a likelihood function $L(\beta)$ by estimating a parameter vector $\widehat{\beta}$. We want to maximize the conditional probability of the event occurring given the predictors. So for samples labeled as 1 we want to estimate $\widehat{\beta}$ such that $\widehat{p(X)}$ is as close to 1 as possible and for samples labeled as 0 it is as close to 0 as possible. We define x_i as the vector of features for the i-th observation, y_i as the binary outcome for the i-th observation, and s is the set of observations.

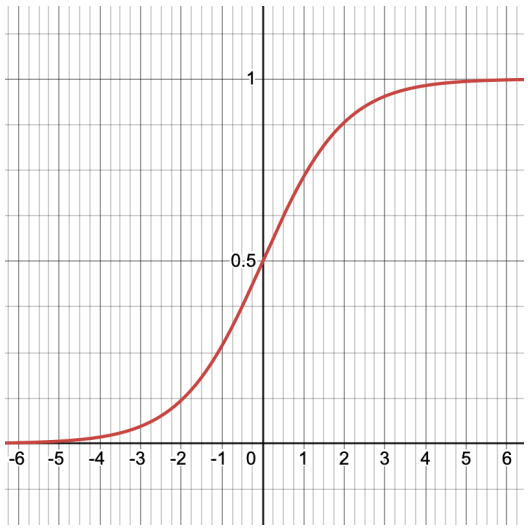


Figure 2.2: The logistic function.

So for N samples we get:

$$\begin{aligned} L(\beta) &= \prod_{s \in y_i = 1} p(x_i) \times \prod_{s \in y_i = 0} (1 - p(x_i)) \\ &= \prod_{s} p(x_i)^{y_i} \times (1 - p(x_i))^{1 - y_i} \\ l(\beta) &= \sum_{i=1}^{N} (y_i \ln p(x_i) + (1 - y_i) \ln(1 - p(x_i))) \end{aligned} \tag{2.7}$$

where $l(\beta)$ is the log-likelihood function [Nguyen, 2020]. In practice however, one often tries to minimize the negative log-likelihood function:

$$-l(\beta) = -\sum_{i=1}^{N} (y_i \ln p(x_i) + (1 - y_i) \ln(1 - p(x_i))) \tag{2.8}$$

2.3.2 Lasso Regularization

Standard logistic regression only seeks to maximize the likelihood function and therefore it might find an optimal solution for the training set but that turns out to be a poor solution for the test set, thus the model would be

overfitting [Brain & Webb, 2002]. It is desirable to instead find a solution that decreases variance and that results in perhaps a worse performance on the training set but generalizes better to the test set and unseen data. This is the objective of regularization. Regularization adds a penalty term to the loss function and aims to discourage the model from fitting the training data too closely and to discourage the model from assigning too much weight to individual coefficients for the predictors.

There are different regularization methods, but Lasso which stands for Least Absolute Shrinkage and Selection Operator is the one we will use. Lasso adds a penalty term that is based on the absolute value of the coefficients and aims to minimize:

$$l_{Lasso}(\beta) = -l(\beta) + \lambda \sum_{j=1}^{p} |\beta_j| \tag{2.9}$$

where λ is a tuning parameter that controls the regularization strength, and p is the number of features. The larger the value of λ the more penalized the coefficients will be and when $\lambda = 0$ the model is the same as standard logistic regression.

An added benefit of Lasso regularization that will be especially beneficial for identifying important factors influencing on-base percentage is that because of the way the penalty term is set up, it can shrink coefficients all the way to 0. This means that Lasso has built-in feature selection, the coefficients being 0 being removed from the model[Galli, 2022]. The feature selection occurs because the Lasso penalty introduces that sharp corners in the constraints where one or more the coefficients are 0, which can be seen in Figure 2.3.

With Ridge regression the penalty term added is based on the squared values of the coefficients:

$$l_{Ridge}(\beta) = -l(\beta) + \lambda \sum_{j=1}^{p} \beta_j^2$$

This created the circular constraints as the figure above shows. The coefficients shrink toward zero but because of the geometry of the constraints, they rarely reach zero.

For the baseball analysis, this is the reason why Lasso's properties are particularly useful. With a large number of predictors, it provides a way to

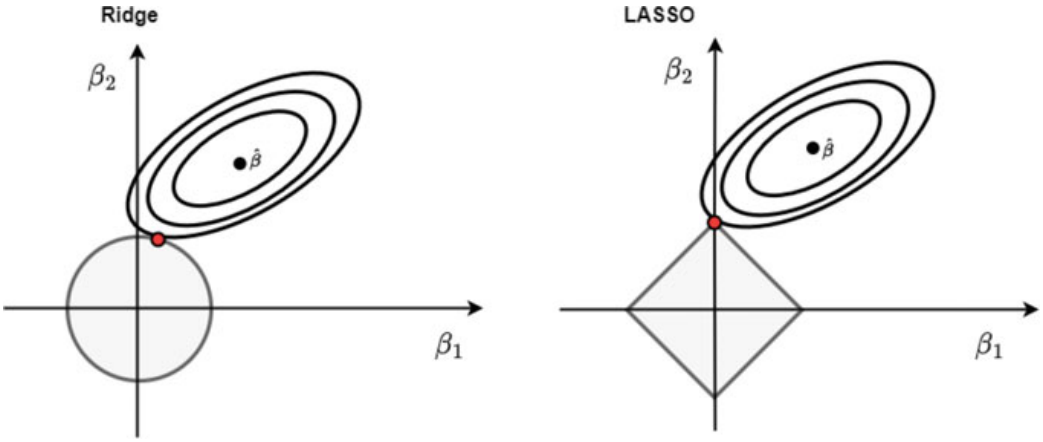


Figure 2.3: Figure illustrating the difference between Ridge and Lasso, showing where the loss function is minimized at one of the corners [Haben et al., 2023].

identify important factors while still using a simple and interpretable model.

2.3.3 XGBoost

XGBoost was created in 2016 [T. Chen & Guestrin, 2016]. Since then, XGBoost has been one of the most popular machine learning libraries and implementations of the library has won numerous machine learning competitions [DMLC, 2023]. Its popularity and design makes it suitable for high-dimensional data and complex datasets. XGBoost stands for eXtreme Gradient Boosting and is a regularized gradient boosting decision tree framework. To understand XGBoost, it is therefore important to understand these terms and the concepts behind them.

At its core, XGBoost uses decision tree models. Decision trees predict the output by evaluating a hierarchical structure of if/else-statements, that is to say they evaluate trees with nodes branching out into two branches. These branch all the way down to the so-called leaf nodes that represent the prediction. Decision trees can be used both for classification and regression problems [Quinlan, 1986]. The tree structure of the model makes it easy to produce graphical representations that make them intuitive and simple to understand, as shown in Figure 2.4.

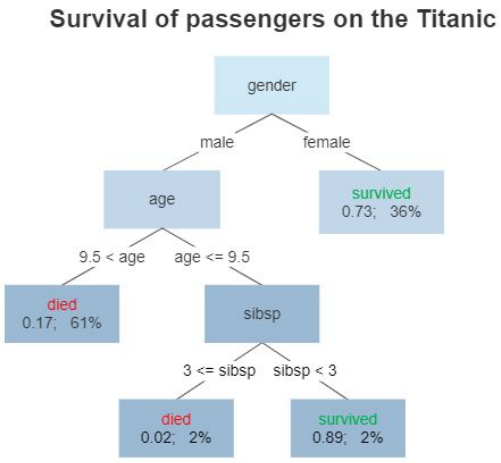


Figure 2.4: A decision tree showing survival probability of passengers on the Titanic depending on the answers to these if/else-statements [Gilgoldm, 2020].

A single decision tree however, can be non-robust and sensitive to changes in the training data. This is due to small variations leading to larger changes to the decision tree itself. Thus, minor changes can drastically affect the model's performance. Therefore ensemble methods are often used. Ensemble methods combine multiple algorithms to create a a single model that is more robust and creates stronger productive models. One method is called boosting. Boosting is a type of method that builds "weak learners" iteratively, with each subsequent learner using the previous learners' residuals to fit the next learner. In this context, a weak learner is machine learning algorithm that performs slightly better than just guessing randomly. The idea can be illustrated and seen in Figure 2.5.

Gradient boosting refers to the method of boosting but where the adding of new learners is governed by a gradient descent algorithm over an objective function. The eXtreme in XGBoost's name is to underscore that this a different and advanced implementation of gradient boosting. As previously stated, XGBoost is regularized which means that it uses regularization. It aims to minimize the following objective function:

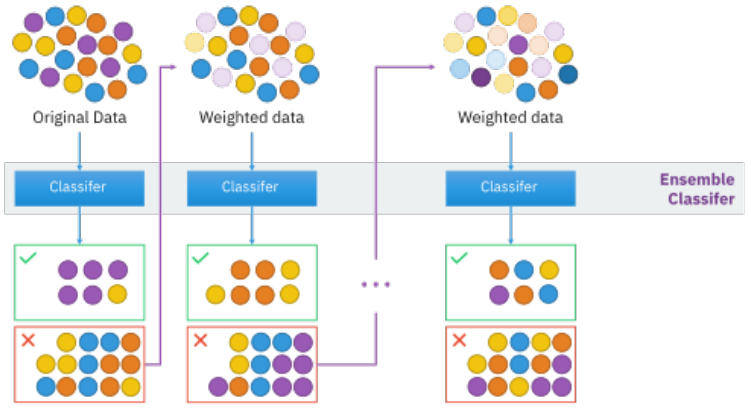


Figure 2.5: An illustration of boosting [Sirakorn, 2020].

$$\text{Obj}(\theta) = L(\theta) + \Omega(\theta) \tag{2.10}$$

Here θ is the model parameters, $\Omega(\theta)$ is the regularization term and $L(\theta)$ a loss function. A tree ensemble model can be defined as:

$$p_i = \sum_{j=1}^{K} f_j(x_i), f_k \in \mathcal{F} \tag{2.11}$$

where p_i is the predicted output for the i-th instance, K is the number of trees created, f_j is an individual tree in the function space $\mathcal{F}$, where $\mathcal{F}$ is the set of possible trees. We let p_i be $p_i^{(t)}$ where t referes to the t-th iteration of the i-th instance. We also add f_t to minimize the objective. For a dataset with m features and n observations this results in:

$$\text{Obj}^{(t)} = \sum_{i=1}^{n} L(y_i, p_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{2.12}$$

From this, we can derive the optimal score of:

$$\text{Obj}^{(t)} = -\frac{1}{2} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} (h_i + \lambda)} \tag{2.13}$$

where g_i and h_i are first two coefficients in the Taylor approximation of $L(y_i, p_i^{(t-1)})$ and λ is a penalty term that controls the regularization strength. For the derivation of the optimal score, see Section A.3.

It is not practical to compute all possible trees, so XGBoost uses a greedy algorithm to calculate the gain in score for the leaf after the split into instances I_L and I_R (the left tree and right tree respectively). That the algorithm is greedy means that it will make the locally optimal choice at each step. We let I be the root node, then the gain is calculated as:

$$\text{Gain} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L}(h_i+\lambda)} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_R}(h_i+\lambda)} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I}(h_i+\lambda)}\right] \tag{2.14}$$

If the value is positive the left split is kept, if the value is negative the right split is kept, if the value is 0 then the tree is grown on both sides [T. Chen & Guestrin, 2016].

The optimizations mentioned, such as the greediness and using the second-order Taylor approximation, are the reason why XGBoost is so efficient and performs well even with large datasets. Furthermore, unlike standard gradient boosting or other tree-based algorithms such as Random Forest, XGBoost includes regularization, which is what we are seeking from Lasso with logistic regression. These properties, and other optimizations that serve to increase speed and efficiency makes XGBoost a promising candidate for the statistical model selection.

2.4 Evaluation Metrics

2.4.1 Accuracy, Precision, Recall, and F2 Score

There is a plethora of evaluation metrics that can be used to judge the performance of a model in a binary classification task.The most common ones are the accuracy, precision, recall. These metrics can all be derived from a confusion matrix, which is a table that shows the performance of the model by its predictions. It is a 2x2 table where the rows show the predictions and the columns show the true values, see Figure 2.6 for an example.

The true positives are the correctly predicted positives, that is the positive class. The true negatives are the correctly predicted negatives. The false

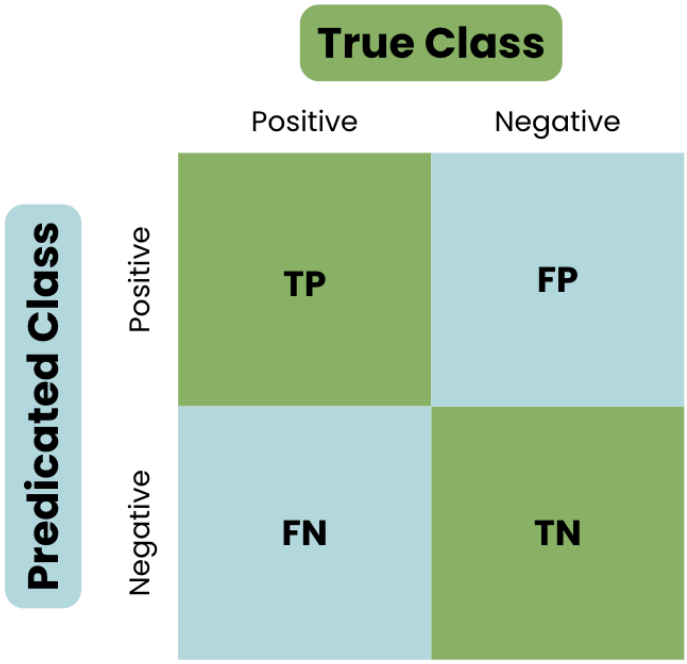


Figure 2.6: How a confusion matrix works [Ahmed, 2023].

positives are the negatives that get incorrectly predicted as positives and vice versa for the false negatives [Powers, 2020]. In the case of on-base events this would result in:

- True positives (TP): the number of on-base events that are correctly predicted as on-base.
- True negatives (TN): the number of on-base events that are correctly predicted as not on-base.
- False positives (FP): the number of on-base events that are incorrectly predicted as on-base.
- False negatives (FN): the number of on-base events that are incorrectly predicted as not on-base.

The accuracy is perhaps the most common one and is written as:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{2.15}$$

We can see that accuracy is a measure of how well the model correctly classifies the data. However, as we will see later, the dataset to be used is imbalanced and accuracy can be misleading in those cases. As an extreme example, if we have a dataset with one class making up 99% of the data a model could get 99% accuracy just by predicting the majority class for every prediction.

Given the limitations, we need to look to other metrics that can be more useful and relevant. Thus, we introduce precision and recall.The precision and recall are defined as:

$$\begin{aligned} \text{Precision} &= \frac{\text{TP}}{\text{TP} + \text{FP}} \\ \text{Recall} &= \frac{\text{TP}}{\text{TP} + \text{FN}} \end{aligned} \tag{2.16}$$

From the equations above it is clear that precision is the ratio of true positives to all items that were predicted as positives while recall is the ratio of true positives to all actual positives. In the context of this baseball analysis, precision reveals the share of on-base predictions that are correct while recall

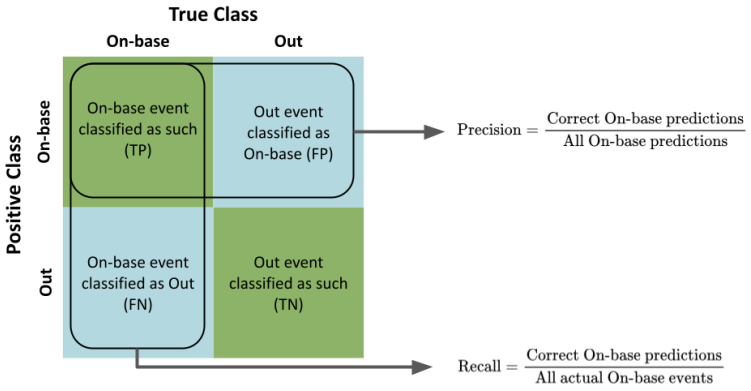


Figure 2.7: Confusion matrix for on-base event classification, with precision and recall for this task added.

reveals the share of actual on-base events that are correctly predicted. A more visual example of this can be seen in Figure 2.7.

Both metrics are relevant for imbalanced data but they should not be used in isolation. For instance, a model that always predicts the positive class will have a recall of 100% and conversely a model that is very selective and only predicts almost certain positive events will have a precision of 100%. The motivation for using both metrics is then that this will give a balanced view of the model's performance as the relationship between them is often inverse, an increase in one is often associated with a decrease in the other. For us, it is more important to correctly identify on-base events but we also want to have a model with predictive power and that can provide insights [Abma, 2009].

With these preferences in mind, we turn to a metric that combines precision and recall while focusing on recall. This metric is called the F2 score:

$$F_2 = 5\frac{\text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}} \tag{2.17}$$

F2 is part of the F-score family and the 2 signifies that this variant places twice as much weight on recall as precision which aligns well with our goals and preferences of identifying on-base events while maintaining a high pre-

dictive power. We see this in Equation 2.17. There, the multiplication of the precision with 4 decreases its relative impact to recall. It is a measure that is well-suited for imbalanced data and for when including more false positives than false negatives is preferred [Van Rijsbergen, 1979]. Thus, the F2 score will be the main metric used for performance evaluation. However, precision and recall are important and useful metric and will also be reported to present a more complete picture of model performance.

2.4.2 Precision-Recall Curve and AUC

To aid evaluation performance and interpretation it is useful to include metric visualizations. As we are using precision and recall, a precision-recall curve or a PR-curve is appropriate. It is constructed by plotting the precision on the y-axis and the recall on the x-axis for different thresholds of classifying predicted probabilities to binary outcomes. For instance, a threshold of 0.5 dictates that probability scores of greater than or equal to this will be classified as positives while those below will be classified as negatives. The PR-curve illustrates the aforementioned trade-off between precision and recall for a given model, see Figure 2.8 for an example. It is also useful for evaluation of different models through the comparison of the Area Under the Curve (AUC). The AUC-PR is the area under the PR-curve and the model with the highest AUC-PR has the best performance as it indicates both high precision and high recall. The maximum area under the curve is 1 while the lowest is 0. Thus, a perfect model would have an AUC-PR of 1 for example [Czakon, 2024].

2.5 Model Interpretability

With machine learning models becoming more and more complex, interpretability of the models often becomes increasingly lost. Advanced algorithms that can capture non-linear relationships, are often referred to as black-box models where results are obtained without any explanation of how. For example, if one compares decision trees to more advanced tree-based algorithms such as the aforementioned XGBoost, the difference becomes stark. Decision tree algorithms output a single tree structure with easily understood decisions and is therefore easier to interpret. XGBoost, however, while often achieving higher accuracy and better performance outputs a model that is

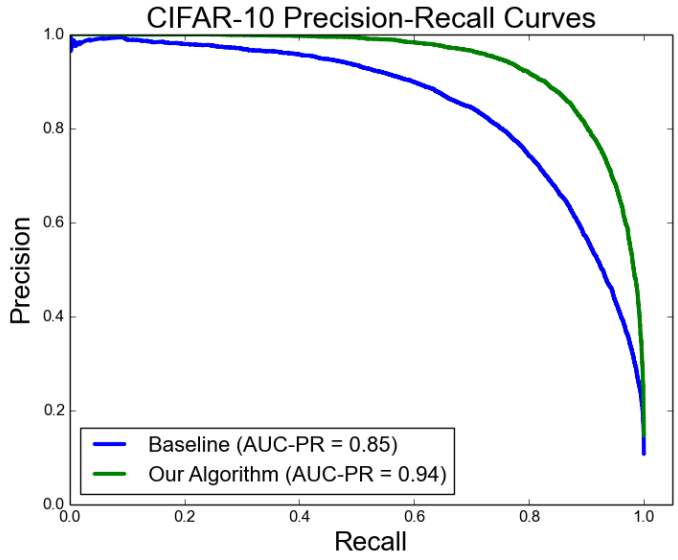


Figure 2.8: A plot showing two different models with corresponding PR-curves and their AUC-PR [Eban et al., 2017].

much more complex. Instead of a single tree, one has to navigate through hundreds or thousands of trees in sequence without clear and cut rules to follow. In many fields, accuracy is not the only metric that is evaluated and instead interpretability is a key factor [Molnar, 2022]. This is especially true for fields that are subject to so-called right to explanation regulations, such as healthcare and finance where an individual has a right to know why a decision was made. This right to explanation is now a law in the European Union with the General Data Protection Regulation (GDPR) [Vollmer, 2023]. In the case of baseball analytics, any model needs to be able to provide actionable insights to coaches and managers and other team staff. With the need for interpretability, there are tools to aid this goal and some of these will be introduced and delved into.

2.5.1 Feature Importance

Feature importance, or variable importance as it sometimes is called, ranks the features of a dataset by how well they aid in predicting the outcome. For linear methods such as logistic regression, when features are standardized

and scaled, this is straightforward as the coefficients for the features directly represent the importance each feature has. XGBoost has built-in methods for feature importance, with gain (see Equation 2.14) being the default one [XGBoost, 2022].

The feature importance method then uses the average gain across all splits to determine an importance score, with higher scores more important. For our analysis, this allows us to identify the most important factors influencing on-base percentage. Potentially, we can also use this to simplify the model and remove features that are unimportant without sacrificing the predictive power of our model. There are however some caveats that introduce limitations to feature importance through XGBoost. The first and perhaps biggest is that feature importance does not show how the factors influence the model, whether they increase the likelihood of an event getting the batter on base or decrease it. Another one is that feature importance only offers a high-level global view of the impact features contribute and does not include information regarding feature interactions. To gain more insight and increase interpretability, we will therefore also look at SHapley Additive exPlanations (SHAP) values.

2.5.2 SHAP Values

SHAP values aim to explain the output of machine learning models. SHAP has its roots in game theory and Shapley values. Shapley values were introduced in 1951 by Lloyd Shapley and this concept won him the Nobel Memorial Prize in Economic Sciences [Nobel Prize Outreach, 2024]. The concept is used in cooperative game theory to fairly divide the payoff of a game between the players and the resulting Shapley values is the expected marginal contribution for a specific player i. This can be written in pseudo mathematical notation as:

$$\phi_i(v) = \frac{1}{\#\text{ of players}} \sum_{\text{S excl. i}} \frac{\text{marginal contribution of i to S}}{\text{number of coalitions excluding i of size |S|}} \tag{2.18}$$

where S is a coalition of players.

In 2017 this concept was extended to the field of machine learning with a

paper by Scott Lundberg and Su-In Lee called "A Unified Approach to Interpreting Model Predictions" where SHAP values were introduced. SHAP values reframed the concept from game theory to machine learning. Instead of calculating the contributions of players to a coalition, SHAP values calculate how features contribute to a prediction. For a specific instance i, where instance refers to a single data point, the SHAP values then become the difference between the average output of the model and the output of the model for i distributed over the features [Lundberg & Lee, 2017]. Thus, they quantify how much the presence of a feature j contributes to a prediction. A positive SHAP value indicates that the feature j increases the prediction, while a negative value indicates that the feature j decreases the prediction. The resulting formula for a single prediction becomes:

$$f(x_i) = \mathbb{E}[f(X)] + \sum_{j=1}^{m} \phi_{ij} \tag{2.19}$$

where $f(x_i)$ is the output of the model for that instance, $\mathbb{E}[f(X)]$ is the expected output for all instances, ϕ_{ij} is the SHAP value for the j-th feature of the i-th instance, and $\sum_{j=1}^{m} \phi_{ij}$ is the sum of the SHAP values for all features for the instance in question.

The additive properties of SHAP values enable global interpretation, that is interpreting the model as a whole. To visualize the impact each feature has on an individual observation, so-called beeswarm plots are used, as displayed in Figure 2.9. In a beeswarm plot, features are organized in rows, ranked by their overall importance. Each dot represents the SHAP value for that feature and instance, ϕ_i, while the color of the dot indicates the feature value with red indicating a higher feature value and blue indicating a lower feature value. Dots with the same SHAP value are stacked on top of each other to show density.

2.6 Model Optimization

To build and optimize a robust and accurate model, we will use hyperparameter tuning with cross-validation. It is standard practice to split the data into a training set and a test set. The training set is used to train the model and the test set is held out until the end to evaluate the model. However,

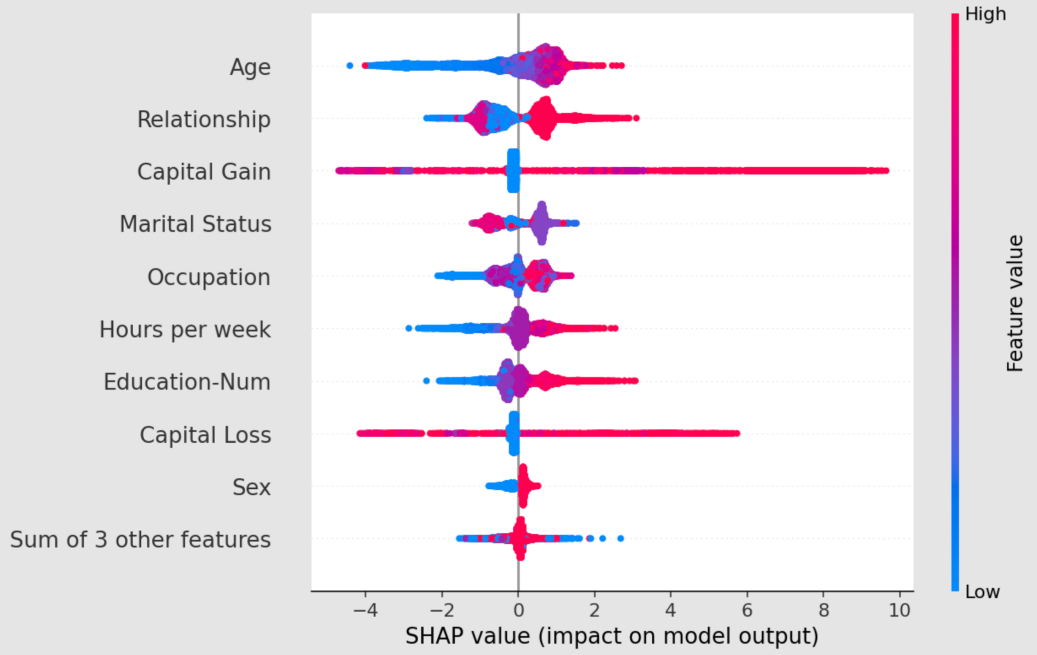


Figure 2.9: A beeswarm plot showing the mean SHAP values for the features [Lundberg, 2018].

the models we will use have many hyperparameters that need to be tuned for good performance and one cannot use the test set to tune them. Instead one uses a validation set to select the best hyperparameters. There are different ways of doing this. One way is to split the training set into a training and validation set once and use the validation set to tune the hyperparameters. Another way is to use cross-validation. Cross-validation is especially useful for limited data but also for large data sets. There are different types of cross-validation, but the most used one is k-fold cross-validation and that is also the one we will use.

In k-fold cross-validation the data training set is split into k subsets of the same size. One subset is used to simulate a validation set and then the model is trained on the remaining $k-1$ subsets. This process is repeated k times with each subset used as a validation set once. The evaluation metrics are then averaged over all iterations. This way, we can tune on the whole training set and avoid overfitting to a specific subset [Bradshaw et al., 2023]. An illustration showing how k-fold cross-validation, with $k = 5$, works can be seen in Figure 2.10.

The two most common hyperparameter tuning methods are grid search and random search. Grid search is an exhaustive search through a previously specified set of values to test. While this is systematic and thorough, it is also computationally expensive and the performance of the optimization depends on the specified grid to search through. For example, if you desire to tune five hyperparameters, with three values each, you will need to test $3^5 = 243$ combinations. To do this with cross-validation and/or large datasets, it quickly becomes impractical [Bergstra et al., 2011].

Random search, on the other hand, randomly samples values to test. This is often more efficient than grid search but it is not as thorough and the algorithm might also spend time and computational resources testing unpromising combinations. For large and complex datasets such as ours grid and random search can be limiting. Instead, we will use Bayesian optimization, a technique that is often preferred for black-box models.

It uses a probabilistic model that is initialized by selecting a small set of random sample points and testing them on the actual model. The results from these points is used to construct a so-called surrogate function which approximates the actual objective function. This is a simpler model that contains estimates using the set of points that were sampled. Together with

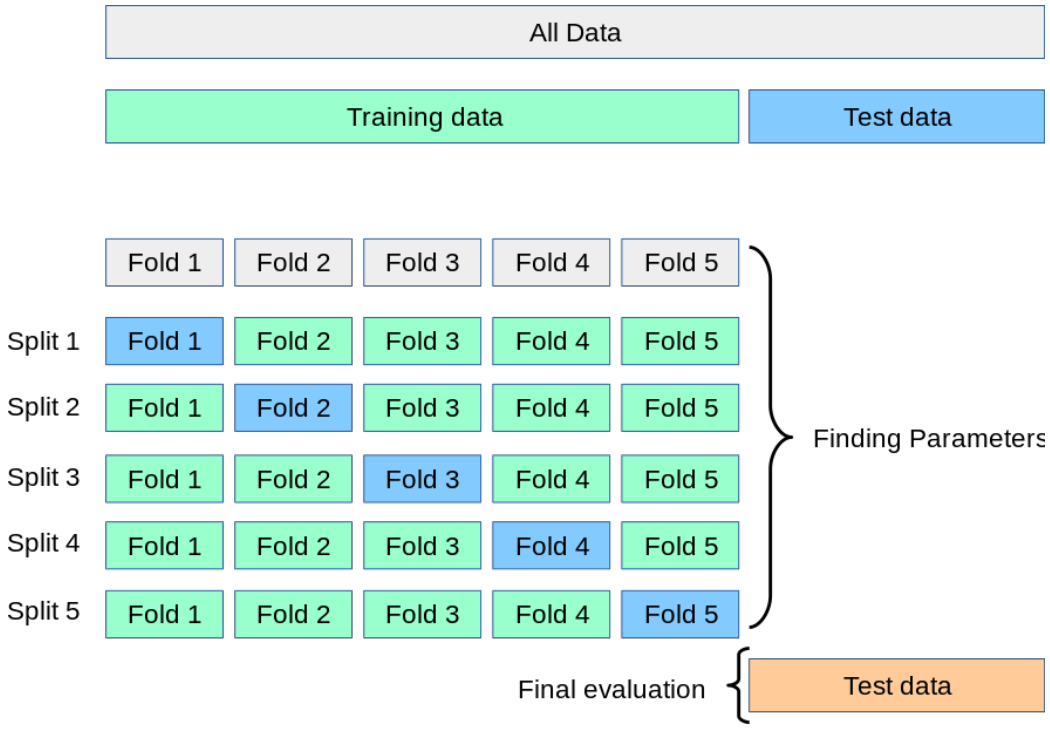


Figure 2.10: An image illustrating how k-fold cross-validation works, here k is 5 [Scikit-learn, n.d.].

surrogate function, a so-called acquisition function is constructed to select future promising points to sample. It does this by using the surrogate function. Often Expected Improvement (EI) is used as an acquisition function:

$$EI(x) = \mathbb{E}[\max(f(x) - f(x^+), 0)] \tag{2.20}$$

where $f(x)$ is the value of the objective function at point x and $f(x^+)$ is the maximum score so far. It is important to note that the acquisition function is a function of the surrogate one. The acquisition function is then used to find x^*, a hyperparameter set that maximizes $EI(x)$, such that:

$$x^* = \arg\max_x EI(x) \tag{2.21}$$

x^* is then evaluated and x^* and the new score is added to the history of samples. The surrogate is trained on the new samples and gets updated. This process is an iterative one and gets repeated for a specified number of iterations [Bergstra et al., 2011].

This is often more efficient than grid search and random search while obtaining better results as they use previous results to guide the search which results in a targeted approach with faster convergence, see Figure 2.11 for an illustration of how the different optimization strategies work.

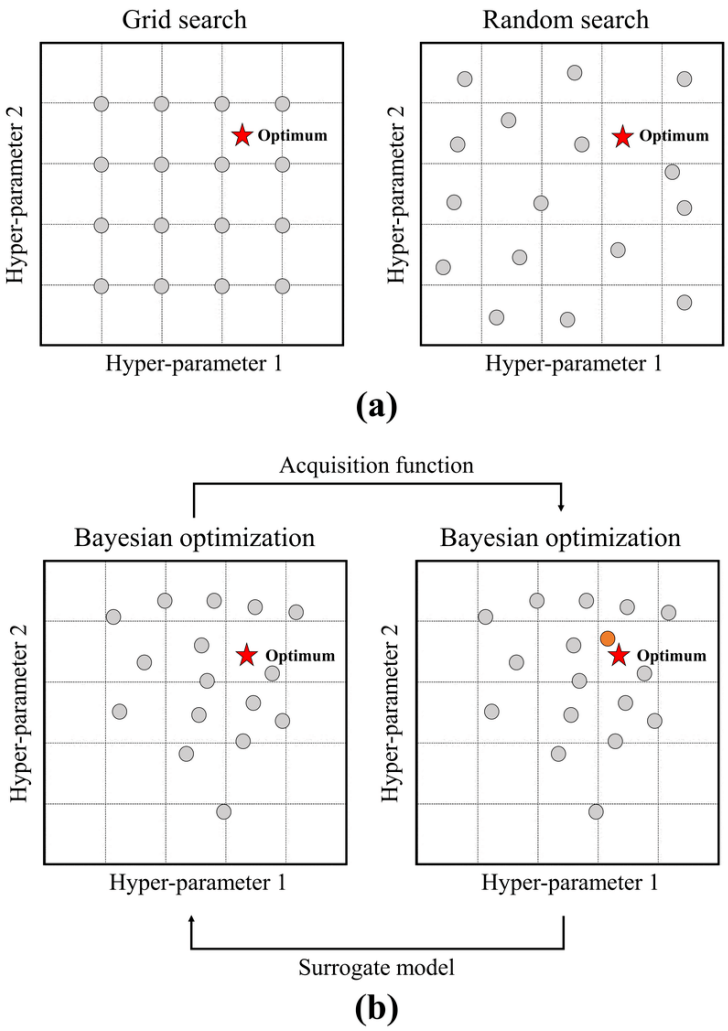


Figure 2.11: An image showing the different optimization strategies [Kim et al., 2021].

Chapter 3

Method

3.1 Data Collection and Preparation

3.1.1 Data Source and Scope

The dataset comes from Statcast. As previously mentioned, the Statcast era is considered to have started in 2015 when the new technology that enables the play tracking was introduced. This makes the scope for the data collection simple to determine. The data is collected from the 2015 MLB season to the most recent completed season, the 2023 MLB season. The data from Statcast is highly detailed and contains pitch-by-pitch data. We are also only using data from regular season games, that is games that are not postseason games. This is customary with American sports as stats for players and teams are colloquially always assumed to be regular season stats and postseason stats and regular season stats are rarely used together. This is because only a select number of teams play in the postseason and as the postseason goes along, that is whittled down further and the number of games are not specified. Rather the games are played in a best-of-X games format.

This filtration of the data renders approximately 6 million rows of data, with each row representing a single pitch. However, as not all pitches end with an outcome, we will only use pitches that end with a result. This then leaves us with approximately 1.5 million rows of data. The Statcast data is very detailed and contains in total 92 columns, which means in total 138 million data points [Baseball Savant, n.d.]. The columns include areas such as: -

Game information: Includes general information about the game, such as date, teams involved, location. - Player information: Includes information about the players involved, such as the batter, pitcher, and fielders. - Pitch details: Includes information about the pitch, such as the type of pitch, the release speed, spin rate, release point, and movement. - Play information: Includes information about the play, such as the inning, the at-bat number, the number of balls and strikes, and the pitch number. - Plate appearance outcomes: Provides information about the outcome of the event, such as the result of the play and more advanced metrics from the outcome of the play. - Ball contact information: For events where contact is made with the ball, information about the contact is included, such as launch angle, launch speed, hit coordinates, and more.

For more detailed information about all columns, see Section A.1.

3.1.2 Data Cleaning and Preprocessing

The dataset that resulted from the aforementioned filtration was still in need of cleaning and preprocessing. The exact cleaning code that was used can be found in the appendix. Firstly, by using the formula for on-base percentage Equation 1.1, we only kept events that are part of the OBP calculation. This meant removing events such as errors, ejections and other events not related to OBP. Then columns with only null values were dropped. Furthermore, columns that were dependent on the result of the play so do not give us information for prediction or the columns have information that is not important or inferred from other columns that we do keep were dropped. To have some semblance of identifiability, we created a composite key that held information about game date, game ID, inning, batter ID, pitcher ID, and at-bat number. This was set as an index and the columns that were used to create the index were removed. Thus we avoid data leakage while having a unique identifier for each row.

3.1.3 Feature Engineering

For the feature engineering, which is the process of creating new features from the existing ones, domain knowledge was used to create new features and to aid in reducing the dimensionality of the data as there were features that could be summarized into singular ones [VanderPlas, 2016]. These include:

- A variable 'score_diff' was created by subtracting the score by the batting team from the fielding team. This way, the game score can be captured in one variable and it is more important to capture the relative difference than the two scores themselves. For instance, a score of 13-12 or 1-0 is not as important as the fact that there is only a one run difference. - We create a 'count_diff' variable that is the difference between the number of balls and the number of strikes to capture the pitch count status in one variable. - The data also comes with three columns for indicating if each base is filled. Instead a single column was created to keep track of runners on base. - Statcast has very detailed information about where the ball passes the strike zone, dividing it into 13 zones. For the purpose of this thesis, it is more useful to have a variable that indicates if the ball is in the strike zone or not, thus such a binary variable was created.

MLB and Statcast's own established metrics were added. These metrics are both used for the general public and by MLB teams to aid insights. The following variables were added:

- A variable for runners in scoring position (RISP) was added. Scoring position is defined as being on second or third base. It is called scoring position as one can score on a single which is usually impossible to do from first base [Sutelan, 2014].
- A binary variable for whether a situation is a late inning pressure situation (LIPS): A late inning pressure situation is one in the 7th inning or later where the team is either ahead by one run or behind by three runs or fewer [Brooks, 1989]. This was added to investigate whether a pressure situation aids or not.
- Spray angle: This is the horizontal angle of the ball when contact is made. The calculation of the spray angle came from the formula used in Python Baseball package which in turn was derived from the work of baseball statistician Bill Petti [Petti, 2017]

Lastly binary variables were created to indicate the following launch angle and launch speed metrics that Statcast provides.

- Hard hit: Any ball hit harder than 95 mph (153 km/h). This is kept track of as research shows a higher likelihood for these hard hits to be successful [Mueller, 2015].
- Sweet spot: Any ball hit at a launch angle of between 8 and 32 degrees.
- Barrel : A barrel is a ball hit in such a way that contact of this type

are considered optimal due to their success rate. It is derived using a combination of launch speed and launch angle. An image of the barrel zone can be seen below.

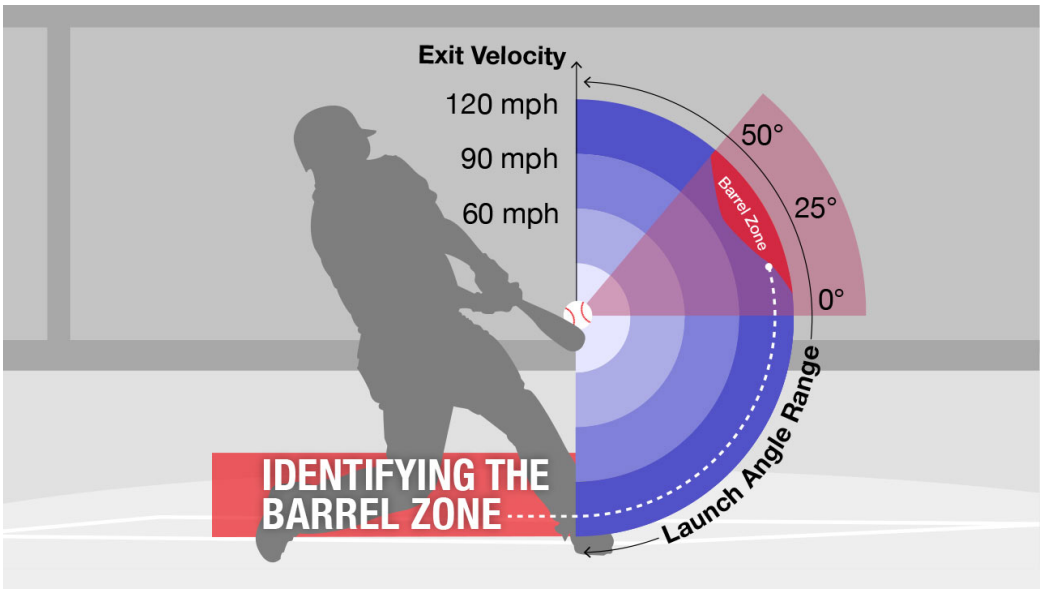


Figure 3.1: The red area shows the combination of launch speed and angle that results in a so-called barrel [Major League Baseball, 2016].

Finally, the events column was transformed into a binary variable that indicated whether an event resulted in the batter getting on base or not. This will be the target variable for the prediction task. We observe that the target variable is imbalanced with approximately 32% of the data belonging to the positive class.

3.1.4 Handling Categorical Features

Categorical features were encoded to either 0 and 1 for binary features while one-hot encoding was utilized for multi-class features, to be used similarly in both logistic regression and XGBoost.

3.1.5 Handling Missing Values

For some variables, imputation was used. For instance, the fielding alignment variables that indicate how the fielding team is aligned it was assumed that from play to play, the alignments do not change. Thus,the missing values there were backfilled using the next available value.

With other variables, the data is only missing for a very small share of rows. This combined with the fact that the Statcast dataset is very large, makes it easier then to simply drop all rows that have missing values.

Early in the process, it was clear that the data could be divided into two groups: events where contact was made with the ball and events where contact was not made. Events where contact is not made are events such as walks and strikeouts. For the non-contact events, of course, the contact related variables are missing. The variables in question relate to launch angle, launch speed, hit coordinates and batted ball type. To handle those, we used the "is_contact" variable was used as an interaction variable, making no contact events 0 and keeping contact events their original value.

3.2 Exploratory Data Analysis

3.2.1 Overview

After the aforementioned data cleaning and transformations, the dataset was ready for exploratory data analysis. The dataset consisted of approximately 1.5 million rows with 47 columns, including the target variable. For detailed information about the columns, see Section A.1. The columns can be grouped into general information, pitch variables, pitch location variables, contact related variables and player related variables such as:

- General information:
  - Outs
  - Home/away team
  - Fielding alignment
  - Score difference
  - Runners on base
  - Count difference
- Player related variables:

  - Batter stance
  - Pitcher throwing arm
- Pitch variables
  - Pitch type
  - Pitch speed
  - Pitch release
  - Pitch movement
- Pitch location:
  - Location across the plate
  - Location on the strike zone
  - Whether the pitch is in the strike zone
- Contact related variables
  - Launch angle and speed
  - Hit coordinates
  - Batted ball type

3.2.2 Pitch Analysis

The analysis started with looking at pitch variables and histograms were plotted for six variables, see Figure 3.2. The top two histograms show the distribution of pitches by release speed and effective speed, both in miles per hour. The next two show the release position of the pitch in the x and z dimension in feet. The last two show the release extension of the pitcher and the spin rate of the pitch in revolutions per minute (RPM).

As the figure shows, release speed and effective speed show similar distributions with the peaks occurring around 90-95 miles per hour. The release position in x-dimension is bimodal, this is likely due to pitcher handedness with the larger peak belonging to right-handed pitchers. On the other hand the release position in the z-dimension looks more normally distributed with a peak around 5.5-6 feet (1.68-1.83 meters), which is to be expected as the variable depends on the height of the pitcher. The same can be seen for the release extension. The spin rate seems more right skewed, with a peak around 2000-2500 RPM.

Due to the bimodal nature of the horizontal release position, it was decided to create a new variables that represented the deviation from the center which might be more important. Thus we took the absolute value of the release position in x-dimension. In the same process we also created a variable that

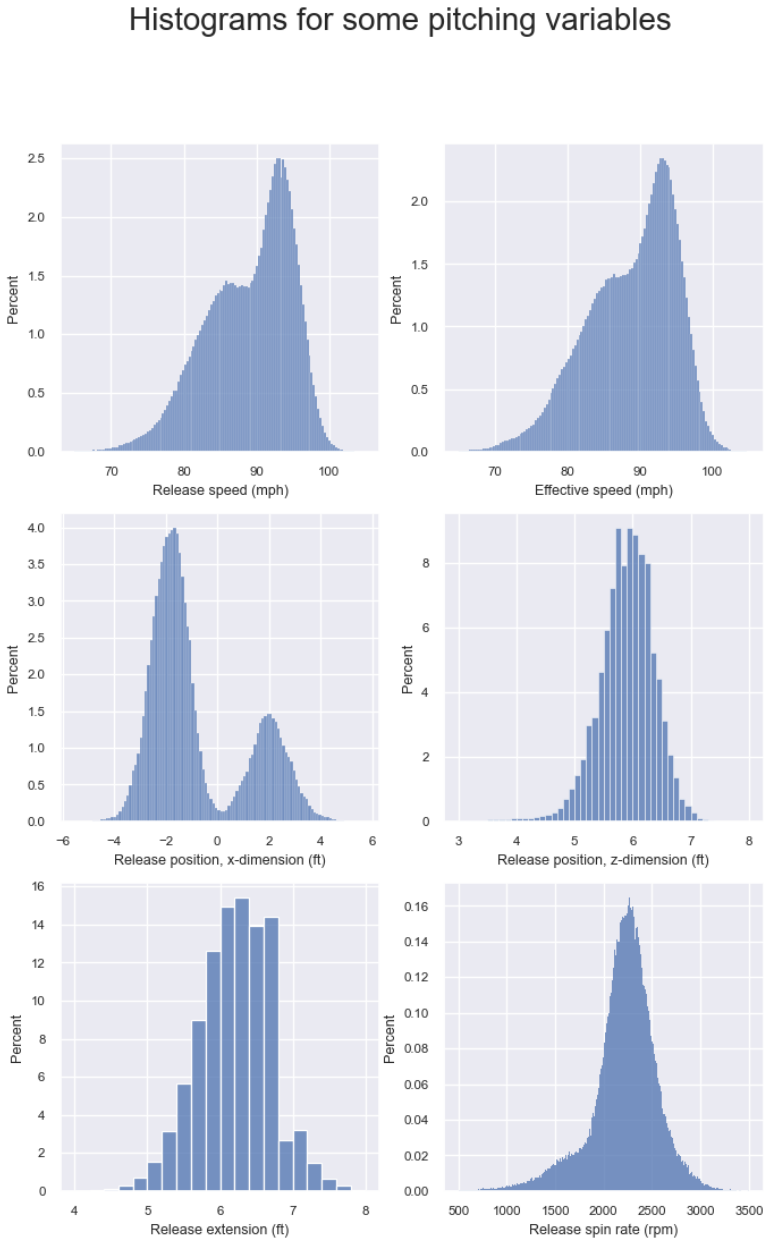


Figure 3.2: The six histograms showing general pitch information.

indicated whether the batter and pitcher were of the same handedness. We did this due to the fact that it is a widely held belief that teams try to use pitchers of the same handedness to maximize their chances of getting the batter out.

The release speed and spin rate are the two main variables generally when looking at pitches, thus we want to investigate further. So distribution plots were made of the pitches by pitch type to get more insight.

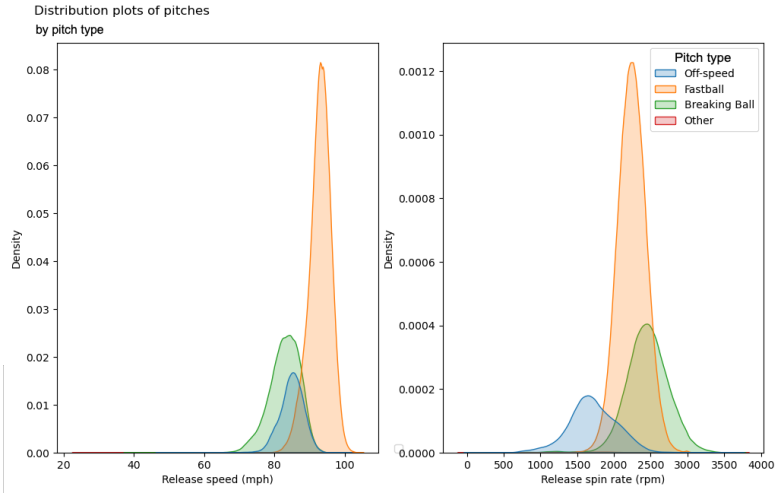


Figure 3.3: Two distribution plots of release speed and release spin rate, grouped by pitch type.

Both plots clearly show that the fastball is the most common pitch by far and has the highest release speeds with speeds up to 100 miles per hour. The breaking balls are the slowest pitches in general but exhibit the highest spin rates with extremes up to 3500 RPM. The 'Other' category is barely shown in the plots, which indicates that the grouping of pitch types done previously is accurate. The results are expected and consistent with the intuition that arises from watching the sport.

Next, the pitch locations were investigated. The pitch locations are classified from the catcher's perspective. We want to see if the zone classification is accurate by Statcast and/or if pitches are called correctly by the umpires, the term used for the referees. Therefore a dual plot was made, a combined scatter plot and distribution plot of the horizontal and vertical coordinates

of the ball as it crosses the plate. This plot can be seen in Figure 3.4. The pitches are then grouped by whether they are in the strike zone or not.

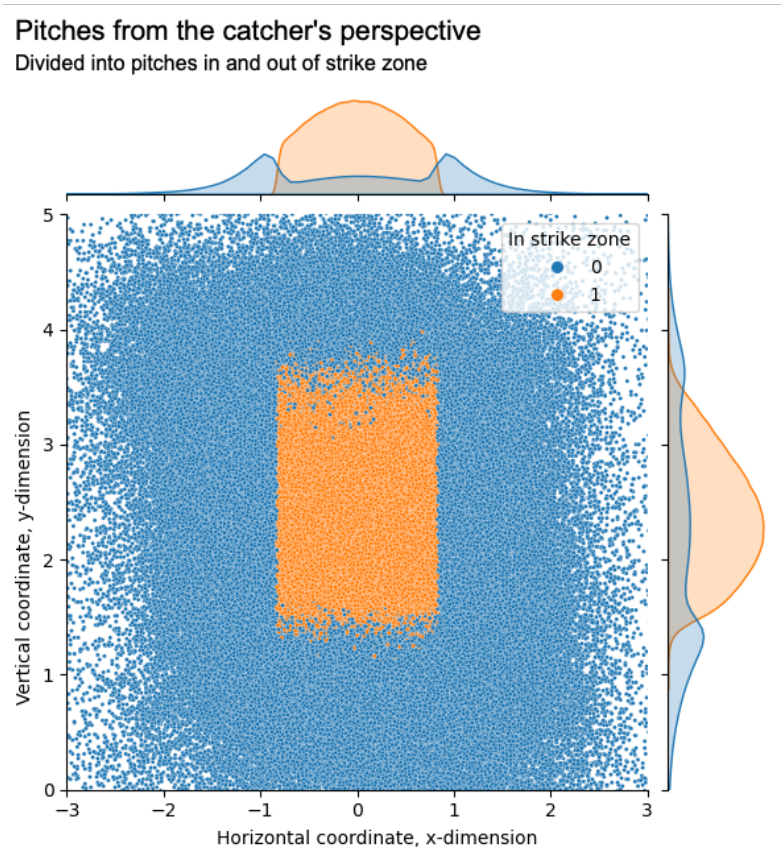


Figure 3.4: Plots showing the pitches from the catcher's perspective as they cross the plate.

In Figure 3.4, we can clearly see the strike zone from the orange points. The plots show that the zone classification is accurate and that the horizontal parts of the strike zone are well defined and very accurate. This is expected as the strike zone's width is defined by the plate width which is constant and the umpires stand behind the plate, giving them a clear view. The vertical borders of the strike zone are more spread out and not as well defined. Again this is expected due to the height of the strike zone being defined by the batter's height as previously mentioned and as can be seen in Figure 3.4.

3.2.3 Batted Ball Analysis

The analysis proceeded with looking at batted balls. A scatter plot was created of the hit locations, grouped by whether the batter got on base or not.

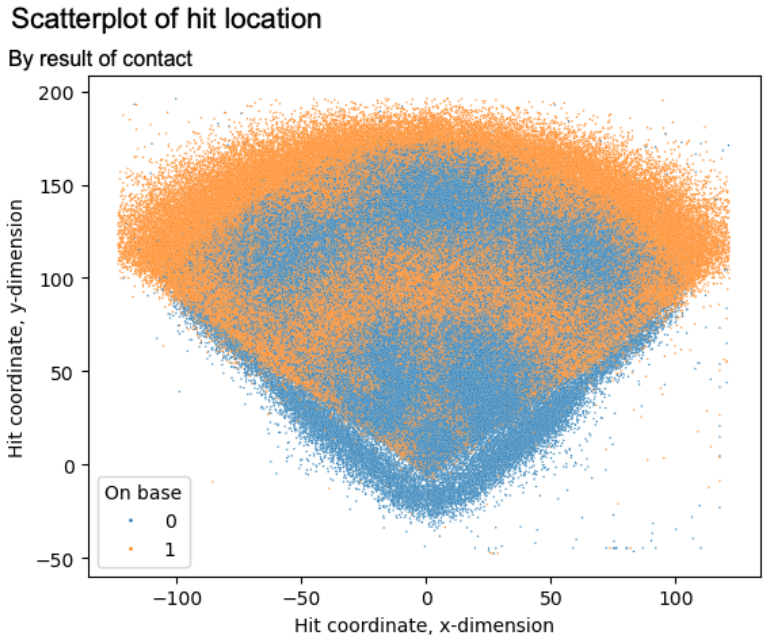


Figure 3.5: Scatter plot of the hit locations by hit coordinates, divided by the result of contact.

The first observation is that the hit coordinates are remarkably well defined and a clear baseball field can be seen with a gap showing foul balls outside the pitch area. It is also visible that balls to the infield are not productive while balls to the outfield are. If we also contrast this with standard fielding alignment, as can be seen in Figure 3.6, that where fielders are the result is usually an out and it is instead required to hit the ball into gaps, where they are too far away to catch the ball, or out of the park for a home run to get on base.

Statcast places a large emphasis on launch speed and launch angle as predictors for hits, and their derived metrics that were added to this dataset through feature engineering use these two variables to derive them. So to

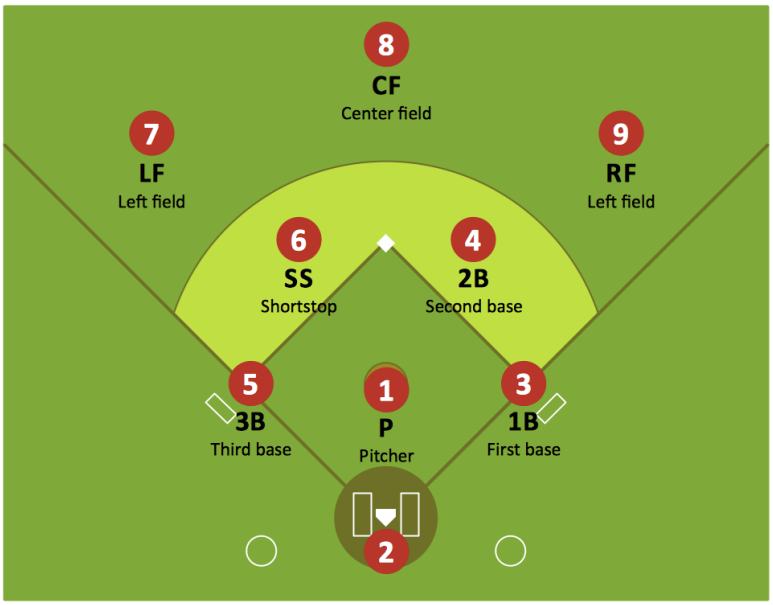


Figure 3.6: Typical fielding alignment in baseball [CS Odessa Corp, n.d.].

investigate this a scatter plot, Figure 3.7,was created with the launch speed on the x-axis and the launch angle on the y-axis, color coded by the result of the contact.

This plot shows some interesting insights. Firstly, there is a clear red cluster showing that home runs are much more likely with a high launch speed of at least 90-95 miles per hour and a launch angle of between 10 and 40 degrees. One can also see that the higher the launch speed, more room is afforded for the launch angle. We can also see that a certain angle a single is more likely, even at lower speeds.

As the launch angle and speed play such big roles in the result of contact, we investigate distribution of them with the corresponding plots as Figure 3.8 shows.

In both plots four peaks can be seen, that are highly out of the norm. Further research showed that is a known phenomenon and comes from Statcast's "no nulls" policy when it comes to launch angle and speed as they are the variables that Statcast places the most emphasis on, as mentioned earlier in Section 3.1.3. Considering this and that both plots have four peaks, it is

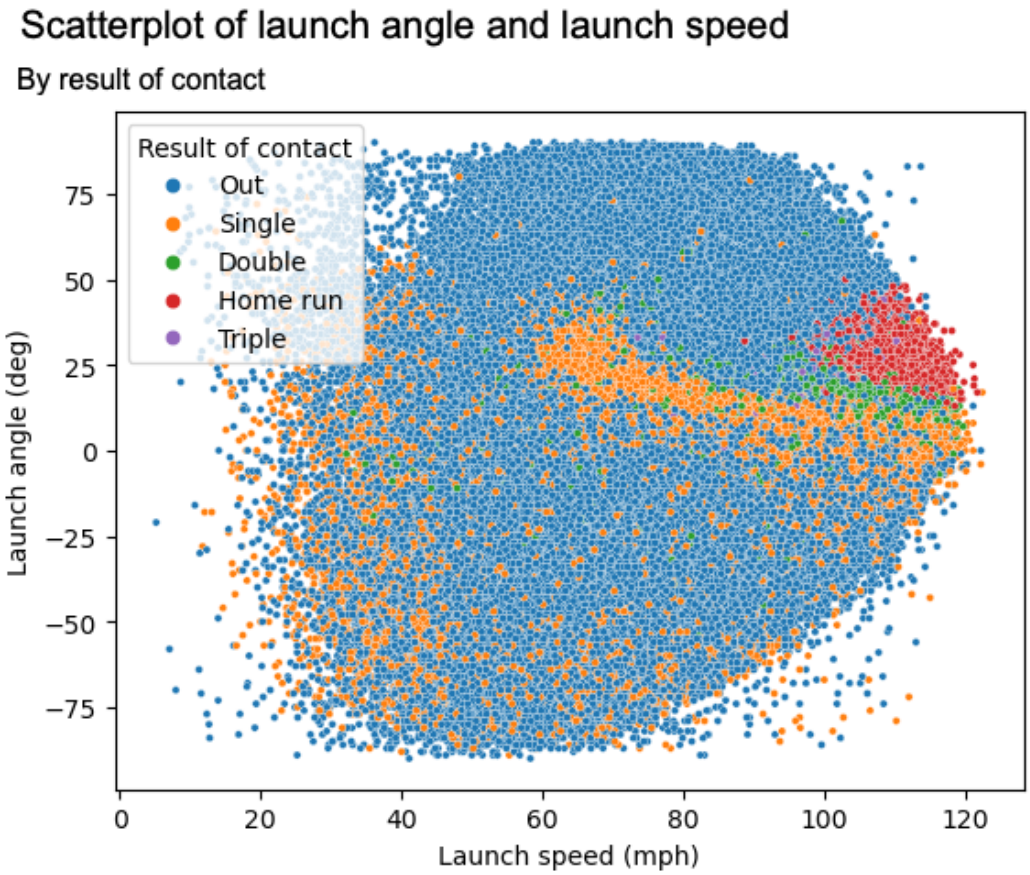


Figure 3.7: Scatter plot of the launch speed and launch angle, grouped by result of contact.

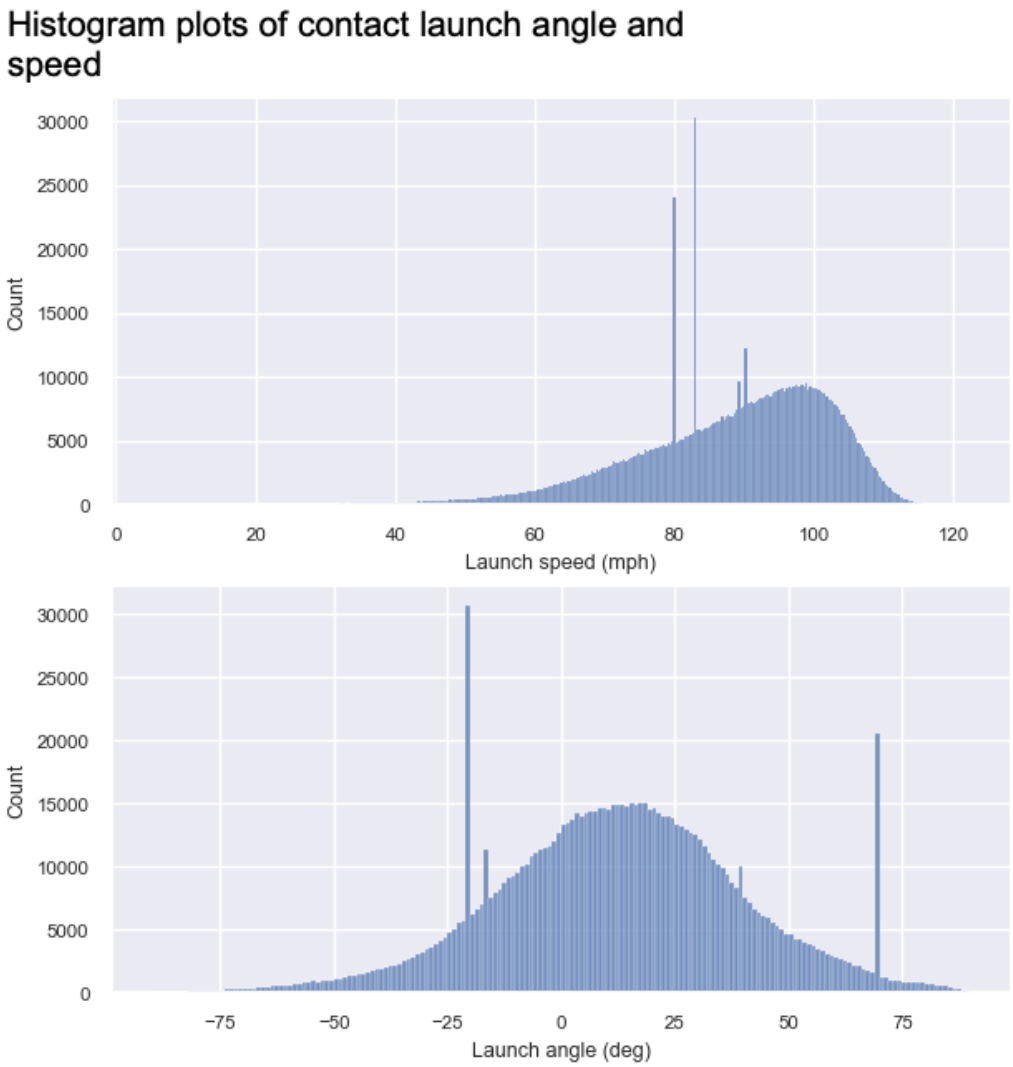


Figure 3.8: Histograms of the launch speed and launch angle.

worth to investigate whether their imputation is made in pairs for the two variables. Thus, the frequency of launch speed and launch angle was listed with the following results of the ten most frequent pairs, in Table 3.1.

Table 3.1: Launch speed and launch angle for the 10 most frequent pairs.

Launch speed (mph)	Launch angle (deg)	Frequency
82.9	-21	24756
80.0	69	19104
90.3	-17	4247
89.2	39	2126
90.4	15	309
91.1	18	293
90.2	-13	256
98.8	17	210
102.8	30	158
93.1	32	114

The four peaks can be clearly seen in the plots with frequencies ranging from 2000 to almost 25000 occurrences. Compared to the fifth most frequent pair, which only appears 309 times in comparison. In total they account for around 5 percent of all occurrences. Considering we have such a large dataset, we can feel confident that their removal from the dataset would not have a significant impact, rather as they are very likely to be default values imputed by Statcast they can present problems if kept. Removing them is the simplest course of action with the size of the dataset and it ensures that we focus on authentic data and we achieve distributions that more closely resemble normality. The confidence is compounded by the fact that that both on-base events and outs are included in these imputed events, so we are not introducing bias into the dataset. After removing the four peaks, the updated histograms can be seen in Figure 3.9.

Comparing this with the scatter plot of the same variables, this looks more reasonable with smoother and more expected distributions. Launch speeds exhibit peaks around 90-95 miles per hour with the distribution being right skewed. The launch angles exhibit more of a normal distribution, peaking around 10 degrees.

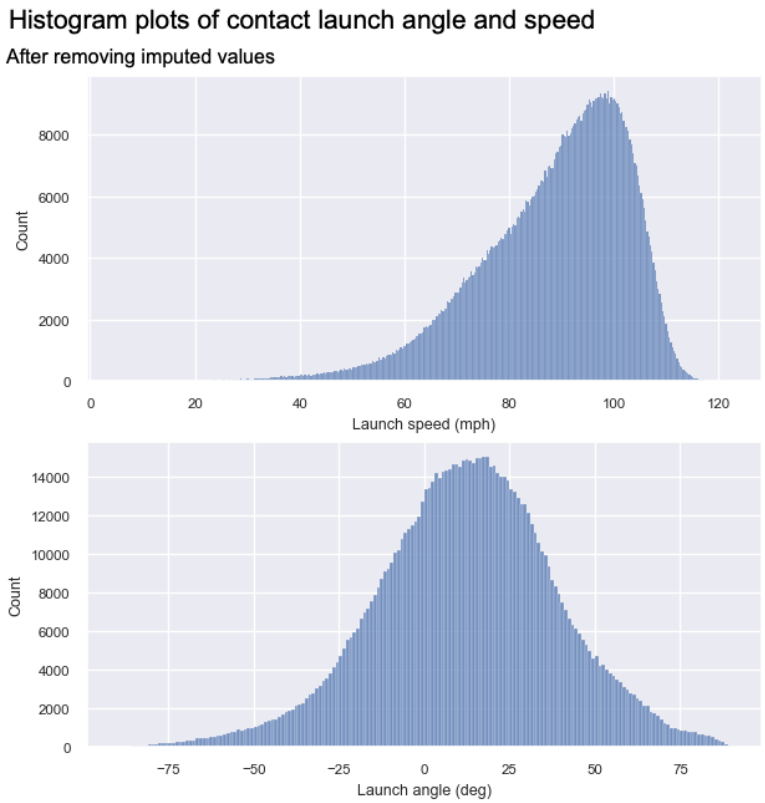


Figure 3.9: Histograms of the launch speed and launch angle after removing the peaks.

3.2.4 Correlation Analysis

We then proceed to look at correlations between the variables through the use of correlation matrices. First, a general correlation matrix is created for all variables to get high level insights, as Figure 3.10 shows.

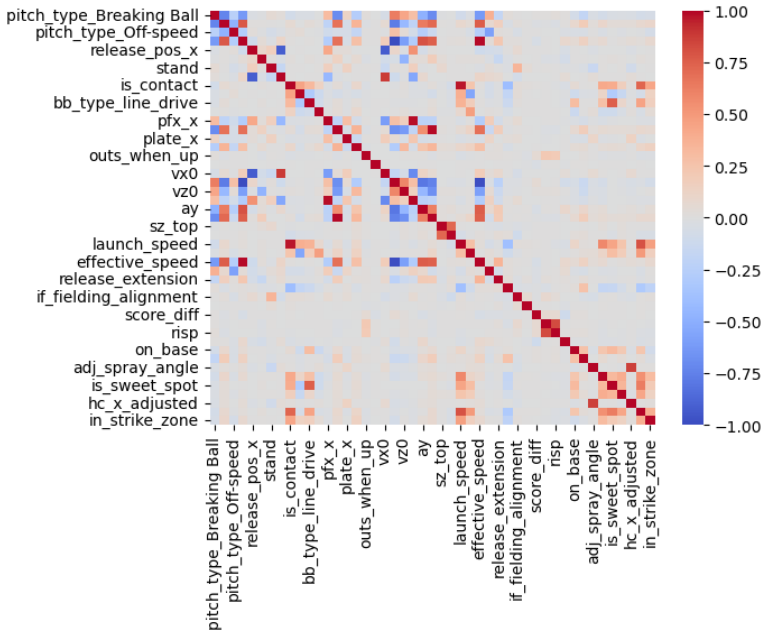


Figure 3.10: Correlation matrix of all variables for the entire dataset.

We see some very boldly colored squares, indicating high positive or negative correlations. Investigating these further, we see that some pitch variables are highly correlated. For instance, release speed and effective speed have a 0.988 correlation while release speed and vy0 have a -0.998 correlation. Considering that these variables are related such as the velocity of a pitch and the release speed the correlation is expected.

The same type of correlation can be seen for the contact related variables. Again, this can be expected due their close relationship. When looking specifically at variables correlation with the target variable we see that the feature engineering has resulted in stronger correlations than their components. The 6 most highly correlated variables to the target variable can be seen in Table 3.2.

Table 3.2: The six most highly correlated variables to the target variable.

Variable	Correlation
is_sweet_spot	0.31
bb_type_line_drive	0.30
count_diff	0.27
hard_hit	0.24
hc_y_adjusted	0.24
is_barrel	0.22

4 of 6 variables are from the feature engineering. We can also look only at non contact events where the correlation for variables such as "in_strike_zone" and "count_diff" is large which is likely due to the non-contact events resulting in walks.

With this correlation analysis, we could glean some insights to guide the rest of the analysis. We see that contact related variables are the most correlated to the target variable. We also see that the feature engineering has been effective and introduces insightful and valuable features to the dataset. Furthermore, we get indications that the dataset is complex and there is multicollinearity in some of the variables.

3.2.5 PCA Implementation and Results

For both PCA and UMAP, as the dataset is large, these operations could not be performed on the entire dataset. Instead, the dataset was sampled for 10% of the data with sampling being stratified by the target variable to ensure that the target variable is evenly distributed. The dataset and all its features was then standardized and scaled. The PCA was then run with a variance threshold of 95% as is commonly done. We found that the number of components needed was 28 with the first two components explaining approximately 25% of the variance and it took 8 components to even surpass 50%. A table of the first five components with their corresponding explained variance ratios and cumulative explained variance ratios can be seen in Table 3.3.

Table 3.3: The first five components with their corresponding explained variance ratios and cumulative explained variance ratios.

	Variance %	Cumulative %
PC1	15.68	15.68
PC2	9.54	25.22
PC3	6.58	31.79
PC4	4.71	36.50
PC5	4.51	41.01

The fact that the first two components only explain 25% of the variance suggests that the dataset might be too complex and not exhibit linear relationships conducive to PCA. Plotting the first two components in with the data points in a scatter plot grouped by the target variables shows the following:

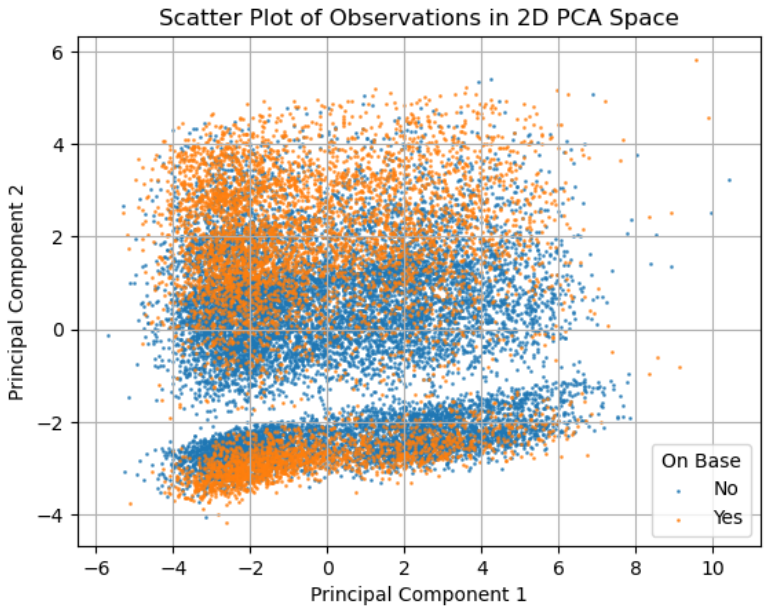


Figure 3.11: Scatter plot of the first two components grouped by the target variable.

We do see a spread in both directions, which signifies that some variance is

captured. However, it is hard to discern any clustering and there does not seem to be any grouping by the target variable. Plotting the most important features in the first two components in a heatmap does show some insights, as presented in Figure 3.12.

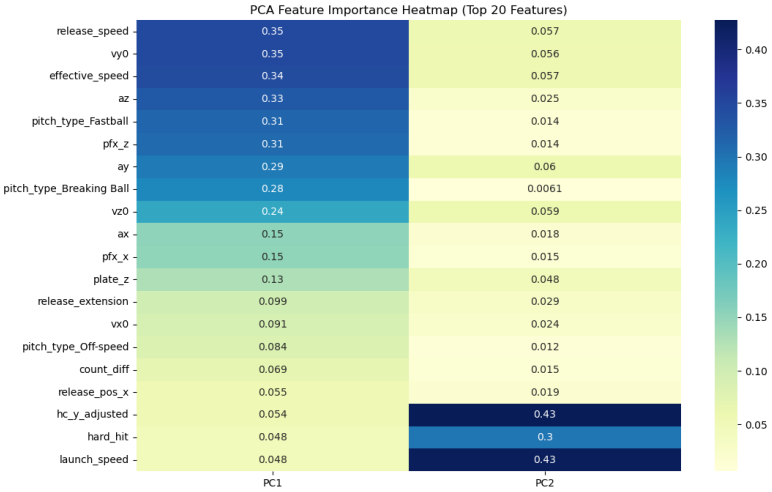


Figure 3.12: Heatmap of the first two components grouped by the target variable.

The more blue the color, the bigger the loading is and we see that for PC1 that pitching variables are the biggest influencers, while PC2 is more influenced by variables related to contact making. It is difficult to discern any deeper insights without investigating further. We do see that some pitch variables are influential, as are the contact variables. It is also clear that the data is complex and the relationships might go beyond the scope of linearity.

3.2.6 UMAP Implementation and Results

Next, the UMAP analysis was done. The same procedure as for the PCA was followed. The data was standardized and scaled. Then the UMAP model was tuned with a range of parameters. Those parameters yielded the following results, as presented in Figure 3.13.

The UMAP model does exhibit some clustering, however it does not seem to be able to capture the structures and complexities that separate the classes of the target variable. However, the fact there is clustering does lend some

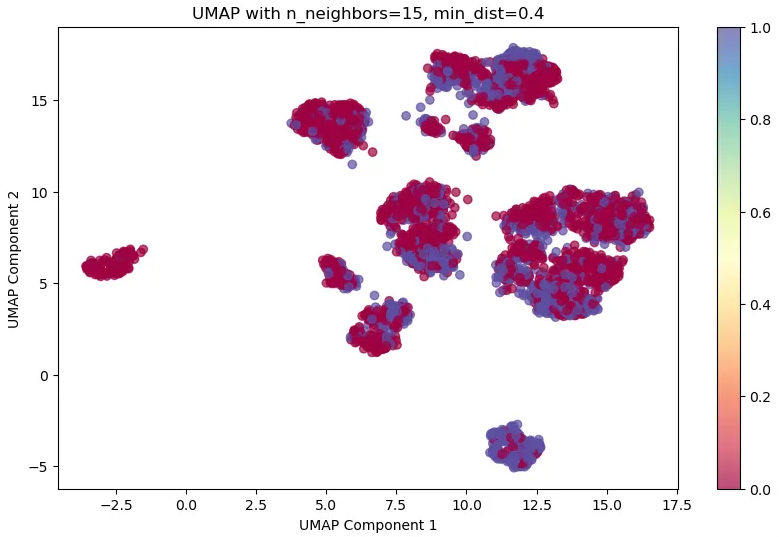


Figure 3.13: Heatmap of the first two components grouped by the target variable.

promise to the idea of predicting on-base events and/or finding important features.

3.2.7 Conclusions from PCA and UMAP

Both techniques indicate that the dataset is highly complex. The PCA required 28 components to reach 95% of explained variance and the first two components only explained 25% of the variance. This signifies that the dataset does have relationships that are not linear. The same conclusion can be drawn from the PCA scatter plot in Figure 3.12, where no clear clustering could be seen. UMAP revealed some clustering that PCA did not, which again indicates that non-linear relationships exist in the data. However, the clustering could not be separated by the target variable. Thus, the fact that UMAP and PCA did not show any clear separations and that the PCA required many components to reach the desired explained variance ratios led us to determine that it would be more beneficial to focus on Lasso regularization for logistic regression and feature selection for XGBoost for any potential dimensionality reduction.

While PCA and UMAP were not used further, they did glean some insights.

Both techniques confirmed the suspicion of complexity in the data and that non-linear relationships exist. Furthermore, PCA did provide some insight from the heatmap in Figure 3.12, that shows that pitching variables are important and so are contact related variables. This aided in the understanding of the data and helped guide future decisions.

3.3 Model Development

3.3.1 Model Training and Evaluation

3.3.1.1 Data Splitting (Train/Test)

The dataset was split into a training set and a test set with 70% of the data being used for training and 30% for testing, which resulted in approximately 1 million rows in the training set and more than 400,000 rows in the test set. As the dataset is imbalanced, we used stratified sampling to ensure that that the target variable is evenly distributed.

For the feature selection in Section 3.3.1.5.2 we used a 10% sample of the training data to be able to run the code with cross-validation described in Section 3.3.1.2. This was necessary due to the computational intensity required to perform the iterative feature selection process. For all other model training and evaluation, we used the entire dataset.

3.3.1.2 Cross Validation

As the dataset is so large, we used a 3-fold cross validation instead of a 5 or 10-fold cross validation. However, we are confident that the results become robust with the combination of folds and the amount of data. Again, we make sure to stratify the data for the target variable. We used this strategy in both the model development and hyperparameter tuning.

3.3.1.3 Hyperparameter Tuning

As previously mentioned, the model optimization strategy chosen was Bayesian optimization. For that we used the Optuna package which is a hyperparameter optimization framework. One uses Optuna by defining an objective with a hyperparameter search space which Optuna then optimizes. The tuning was then done with the aforementioned 3-fold cross validation

and the evaluation metric used was the mean F2 score. We set Optuna to run 50 trials which is a resource intensive operation that required enabling parallelization. The parameters that yielded the best F2 score were then saved and used for the final model.

3.3.1.4 Logistic Regression Implementation

For the logistic regression we wanted to develop some initial models to serve as baselines and to give direction to the areas of improvement and which hyperparameters to tune. The first model implemented was a bare bones logistic regression with no parameters specified except for the computer performance related ones, such as the number of processing cores to use and the maximum number of iterations. Then a lasso logistic regression was implemented, that had the additional hyperparameter of which penalty to use. The final initial logistic regression model was one with class weighting. Class weighting, as the name dictates, assigns weights to each class to give more importance to the minority classes with the aim of making the model give equal attention to both classes. This was done as the dataset is imbalanced and we do not want the model to favor one class over the other.

3.3.1.4.1 Hyperparameter Tuning

The logistic regression model was tuned with the following hyperparameters and search spaces, as presented in Table 3.4.

Table 3.4: The hyperparameters that were tuned for the logistic regression model.

Parameter	Description	Search Space
alpha	The Lasso regularization parameter, the higher the value the more regularization is applied	0.00001 to 10 in log scale
learning_rate	The learning rate schedule, that determines the step size of each iteration	{optimal, invscaling, constant, adaptive}

Parameter	Description	Search Space
eta0	The starting learning rate	0.00001 to 10 in log scale
class_weight	The weighting of the classes	{balanced, None}

with the obtained hyperparameters presented in Section A.4.

3.3.1.4.2 Feature Selection

For the feature selection process the intrinsic capabilities of Lasso regularization were used and the final model was trained with the obtained optimal regularization strength, which then determines if any features received coefficients of 0.

3.3.1.5 XGBoost Implementation

Like the logistic regression, an initial XGBoost model was created with minimal hyperparameter specification. That model was then evaluated with the 3-fold cross validation and the requisite metrics were calculated. The model was then optimized.

3.3.1.5.1 Hyperparameter Tuning

The XGBoost model was tuned with the following hyperparameters and search spaces, as presented in Table 3.5.

Table 3.5: The optimal hyperparameters for the XGBoost model.

Parameter	Description	Search Space
max_depth	Maximum depth of the tree	3 to 8
learning_rate	The learning rate schedule, that determines the step size of each iteration	0.001 to 1 in log scale

Parameter	Description	Search Space
n_estimators	The number of trees to build	100 to 1000
min_child_weight	The minimum number of samples required to split a node	1 to 6
gamma	The minimum loss reduction required to make a split	0.01 to 1 in log scale
reg_alpha	The Lasso regularization parameter, the higher the value the more regularization is applied	0.01 to 10 in log scale
scale_pos_weight	The weighting of the classes	{balanced, None}
subsample	Subsample ratio of the training instances, used to prevent overfitting	0.7 to 0.9
colsample_bytree	Subsample ratio of columns when constructing each tree, again to prevent overfitting	0.5 to 0.7

The optimal hyperparameters are, like the logistic regression tuning, presented in Section A.4.

3.3.1.5.2 Feature Selection

The feature importances were extracted from the optimal XGBoost model and then an iterative process was used to run cross-validation on the optimal model, increasing the number of features used. The first run was only done with the feature with the highest gain and the last run was done with all features. From there we selected a threshold that resulted in a balance of feature selection and model performance.

3.4 Model Evaluation and Interpretation

3.4.1 Performance Metrics

For testing against the test set, three models were evaluated. The first model was the optimized logistic regression model. The second model was the optimized XGBoost model. The third model was the optimized XGBoost model after feature selection. As previously mentioned, the F2 score was the main metric used with all models also being evaluated with AUC-PR, precision and recall. For the AUC-PR, precision and recall a precision-recall curve was created with all three models featured and the area under the precision-recall curve was calculated. To further aid in interpretation, confusion matrices were also created for each model.

3.4.2 Feature Importance Analysis

The feature importance analysis was done with using the feature importance values from both XGBoost functions and they were plotted in bar charts to aid in extracting insights regarding the most influential factors. We also compared the results of the two XGBoost models with each other to see if whether the most important features of the full model ended up being the same as the features in reduced model or if the dimensionality reduction could yield a different result and new insights.

3.4.3 SHAP Value Analysis

For a more nuanced and detailed analysis of the model outputs of the best performing model SHAP values were used. To evaluate the model as a whole, a beeswarm plot was created with SHAP values for the top features and interaction plots were created to see how the features interacted with each other.

3.5 Tools and Libraries Used

Python was used for the analysis with the following libraries being used:

- Scikit-Learn and XGBoost: for machine learning
- Pandas: for data analysis and manipulation

- Seaborn and Matplotlib: for data visualization
- Numpy: for data manipulation
- PyBaseball: for data collection
- SHAP: for SHAP values

Chapter 4

Results

4.1 XGBoost Feature Selection Results

Using the optimal XGBoost model and the process outlined in Section 3.3.1.5.2, F2 scores for each run are presented in Figure 4.1.

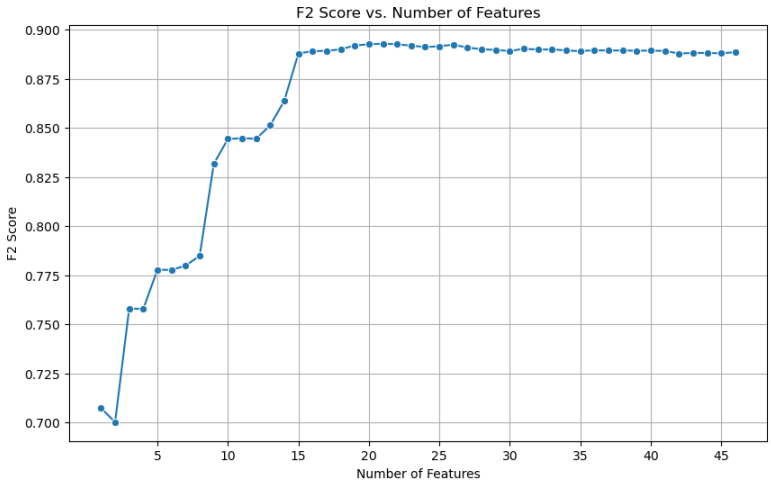


Figure 4.1: Line plot of the F2 score for each run with an increasing number of features used.

As can be seen in Figure 4.1 the F2 score reaches near optimal values at 15 features at 88.79%.Additional features thereafter have no significant impact

on the model performance with a peak of 89.29%.

Thus, for a feature-selected model, the number of features was chosen to be 15 with the features being listed in Table 4.1.

Table 4.1: The features used for the feature-selected model.

Rank	Feature
1	in_strike_zone
2	count_diff
3	is_contact
4	is_sweet_spot
5	hard_hit
6	is_barrel
7	pitch_type_Fastball
8	bb_type_ground_ball
9	hc_y_adjusted
10	launch_angle
11	bb_type_popup
12	bb_type_line_drive
13	launch_speed
14	plate_x
15	hc_x_adjusted

4.2 Lasso Feature Selection Results

With the process outlined in Section 3.3.1.4.2, the optimal model was obtained with a regularization strength that selected 42 out of 46 features. The features are listed in Table 4.2, ranked by their absolute value of the coefficient.

Table 4.2: The features selected by Lasso.

Rank	Feature	Rank	Feature
1	release_speed	22	hc_x_adjusted
2	hc_y_adjusted	23	launch_angle
3	is_contact	24	plate_z

Rank	Feature	Rank	Feature
4	bb_type_ground_ball	25	az
5	effective_speed	26	pfx_z
6	bb_type_line_drive	27	pitch_number
7	vy0	28	runners_on_base
8	launch_speed	29	bb_type_popup
9	count_diff	30	plate_x
10	in_strike_zone	31	release_spin_rate
11	hard_hit	32	score_diff
12	is_sweet_spot	33	home_away
13	is_barrel	34	outs_when_up
14	adj_spray_angle	35	same_hand
15	pitch_type_Off-speed	36	release_pos_z
16	pitch_type_Breaking Ball	37	if_fielding_alignment
17	ay	38	ax
18	sz_top	39	release_pos_x_abs
19	pitch_type_Fastball	40	lips
20	sz_bot	41	release_pos_x
21	release_extension	42	of_fielding_alignment

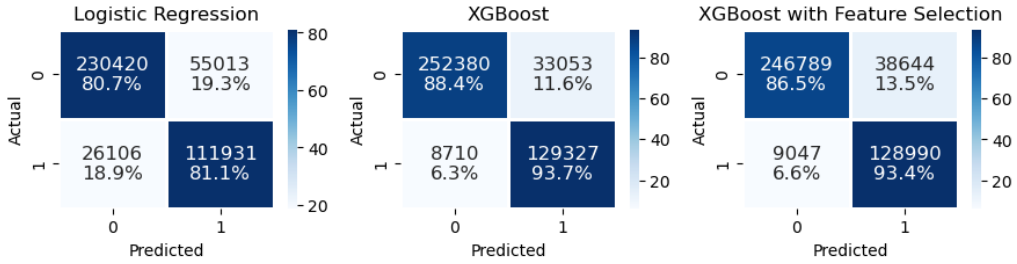
4.3 Model Performance Results

4.3.1 Confusion Matrices

In Figure 4.2 the confusion matrices for each model can be seen, with counts and percentages for the true positives, true negatives, false positives and false negatives.

4.3.2 Performance Metrics

This table shows the F2 score, AUC-PR, precision and recall for each model.



(a) (b) (c)

Figure 4.2: Results for the confusion matrices for each model.

Table 4.3: Performance metrics for each model

Model	F2	Precision	Recall	AUC-PR
Logistic Regression	77.83%	67.05%	81.09%	68.42%
XGBoost	90.50%	79.64%	93.69%	95.82%
XGBoost after Feature Selection	89.60%	76.95%	93.45%	94.90%

We can clearly see that the XGBoost models perform better than the logistic regression model in all metrics. The full XGBoost model achieves the highest scores with an F2 score of 90.50%, precision of 79.64% and recall of 93.69%. The feature-selected model achieves a slightly worse performance with the precision being the most affected at 76.95%, almost three percentage points lower than the full model. However, these differences are very minute. The high F2 and recall scores indicate that the XGBoost models are very good at predicting on-base events, which is very important in baseball strategy as teams always look for scoring opportunities and do not want to miss those opportunities. The AUC-PR for the logistic regression model is 68.42% while the AUC-PR for the XGBoost model is 95.82%. The XGBoost model after feature selection achieves a slightly lower, but still very impressive, AUC-PR of 94.90%. The fact that the feature-selected model is so close to the full model with less than a third of variables shows that the feature selection process by XGBoost seems effective and this can potentially offer significant computational advantages with the same predictive power.

4.3.3 Precision-Recall Curve and AUC-PR

Figure 4.3 shows the precision-recall curve obtained by each model and the selected precision and recall for each model marked with a circle.

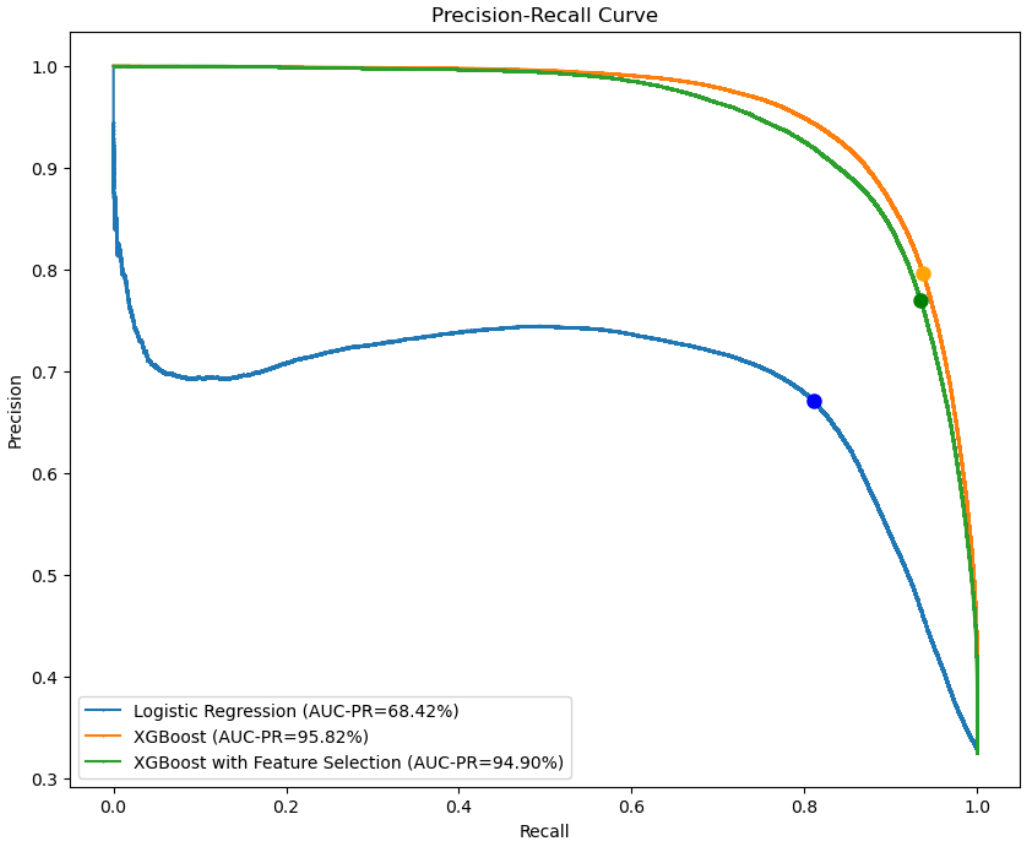


Figure 4.3: Precision-Recall Curve and AUC-PR

We see similar results in the precision-recall curve as Figure 4.3 shows where XGBoost models obtain better results at each point of the curve with the feature-selected model slightly differing at the very top of the precision and recall. Looking at the area under the curve for each model, the differences appear as starkly. The logistic regression's precision drops off very quickly as recall is increased, eventually stabilizing at approximately 70% until recall reaches above 80%.

The performance difference between the XGBoost models and the logistic

regression model made it a reasonable choice to then only focus on the XGBoost models for the feature importance analysis and the SHAP analysis.

4.4 Feature Importance Results

For this project and the dataset used, this resulted in the selection of 15 features that this iterative process indicated would yield similar performance to the XGBoost model with all features.

Figure 4.4 shows the relative feature importance for each model with the top 15 features being shown for the full model and all 15 features being shown for the feature-selected model.

Looking at the feature importance results in Figure 4.4 we see that there are differences with the reduced model valuing certain features more than the full model. This might be the model adjusting to the absence of the features that the complete model still has.

For the reduced model, the 'is_contact' variable is assigned significantly higher importance compared to the full model, whereas the full XGBoost model places more weight on the 'in_strike_zone' and 'count_diff' variables. The importance placed by both models on the resulting features aligns well with baseball intuition, that getting in a favorable count (having more balls than strikes) is important, as is making contact with the ball.

4.5 SHAP Results

Figure 4.5 and Figure 4.6 show the bee swarm plots for the full and feature-selected models respectively with the features ranked by their mean absolute SHAP values and the plots are color coded by the feature values with higher feature values being colored in red and lower feature values being colored in blue.

The two bee swarm plots look very similar with the five most important features being the same:

- hc_y_adjusted (hit coordinate, y-dimension)
- launch_angle (launch angle)

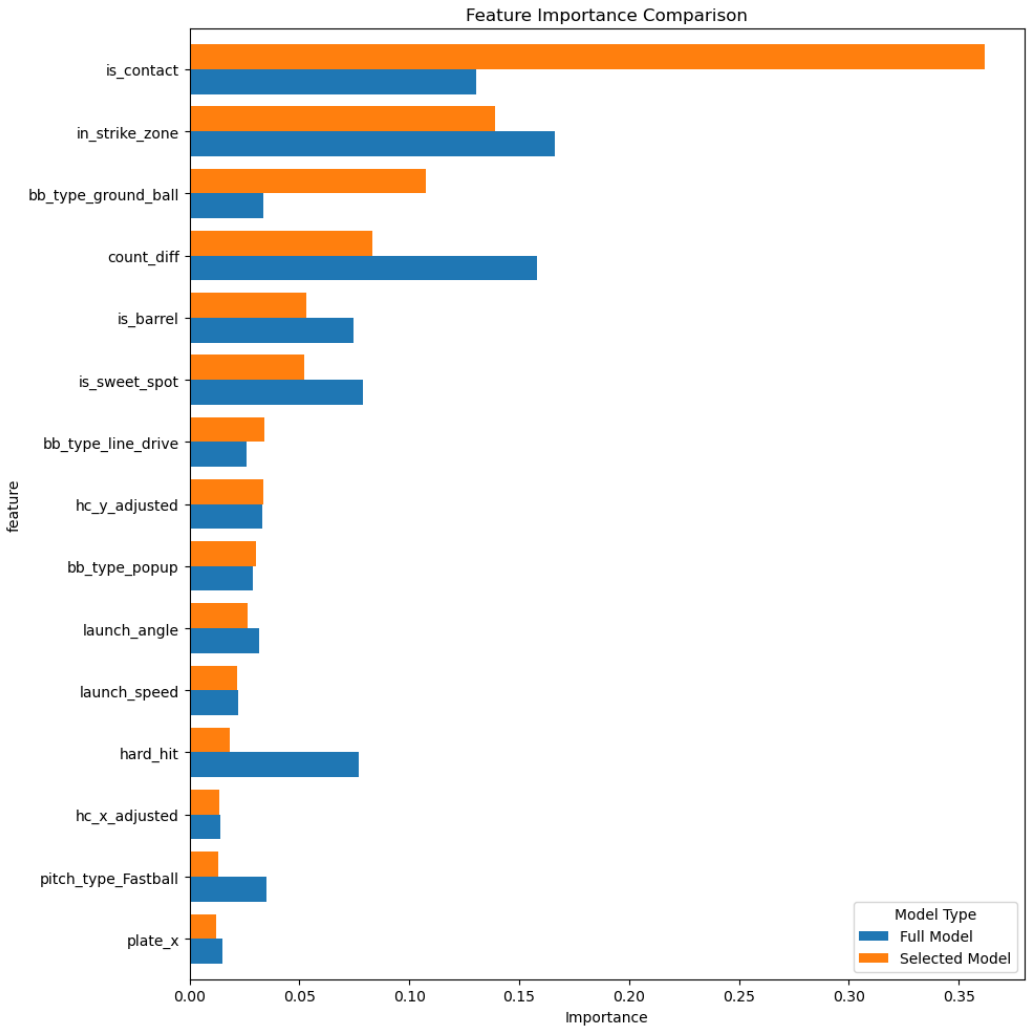


Figure 4.4: Feature Importances for the two XGBoost models.

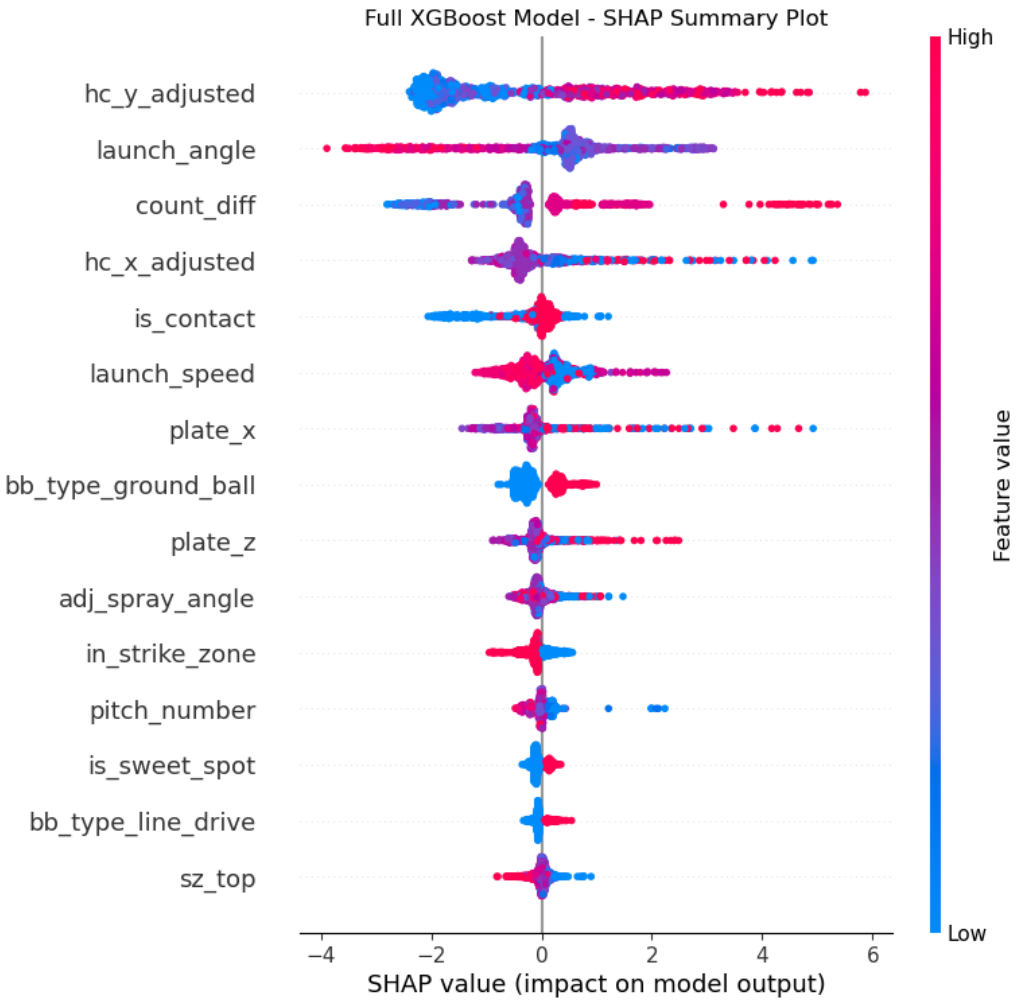


Figure 4.5: SHAP bee swarm plot for the full XGBoost model.

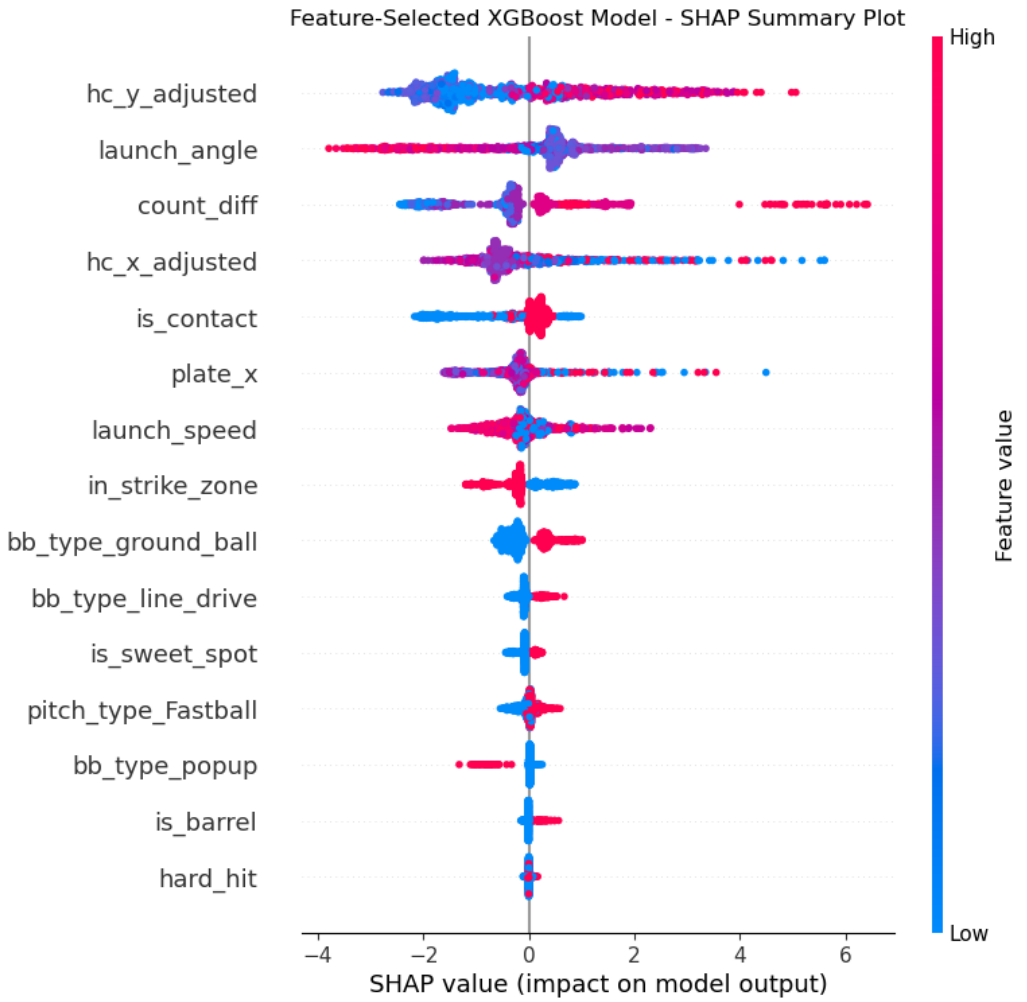


Figure 4.6: SHAP bee swarm plot for the feature-selected XGBoost model

- count_diff (count difference, i.e. number of balls minus number of strikes)
- hc_x_adjusted (hit coordinate, x-dimension)
- is_contact (whether contact was made with the ball)

The plots show that for both 'hc_y_adjusted' and 'count_diff' the higher feature values are the more positive impact on OBP. Launch angle exhibits more of a non-linear relationship with the target variable with extreme values at both ends having a negative impact. This is also consistent with baseball intuition, that a ball launched at a higher angle gives fielders more time to catch the ball and the ball cannot go so far. Equally disadvantageous a low angle seems to be, where hitting the ball at an angle of less than 0 degrees is not traveling far and easily being recovered by the fielders. This also explains the rationale for the sweet spot metric created by Statcast that, as mentioned in Section 3.1.3, is an angle of between 8 and 32 degrees.

In summary, the similarity of the two bee swarm plots do indicate again that the feature selection was successful and that the key features identified are robust and that the reduced model simplified but did not experience any significant loss in performance and can quite accurately capture the essence of the full dataset and the full model.

There is some difference to feature importance provided by XGBoost with the two SHAP plots being very similar while the feature importances for the two models being more different. It does then seem that the SHAP values are more stable than the gain metric from XGBoost and captures complexities and interactions between the features more accurately.

Figure 4.7 shows the interaction plots for the full model. They show the interaction effect between features in the model. The color indicates direction of interactions, with red indicating positive interactions while blue indicates negative interactions. Positive interactions in this context refer to the features interacting to increase the prediction while negative interactions indicating the opposite.

The plot shows for example that 'hc_y_adjusted' and launch angle have strong self-interactions, that is that those features as we have found out have strong effects on the predictions. They also have strong interactions with each other, suggesting that the launch angle has a strong effect on the distance in the y-direction. In summary, one can also see from the interactions that they

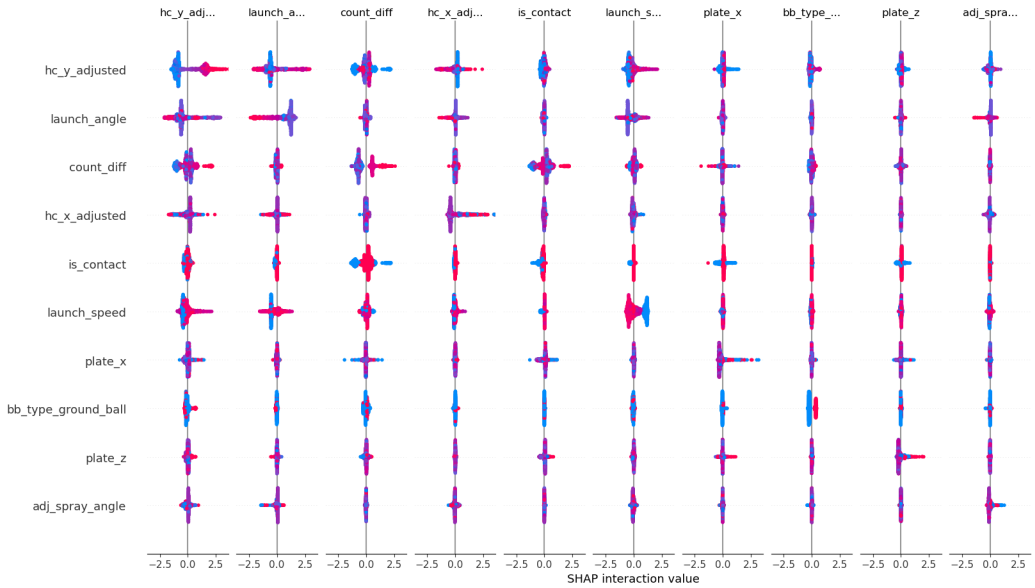


Figure 4.7: SHAP Interaction Plot

are complex and non-linear which justifies the use of XGBoost and explains its' advantage over the logistic regression model.

Chapter 5

Discussion

5.1 Insights for Baseball Strategy

There are some insights that can be drawn and are relevant to baseball strategy. The first is that we obtain a greatly reduced model that indicates that a certain subset of features dictate outcomes more than others and should be regarded more by teams. Not all factors can be affected by the teams but certain ones are and these factors are more important for teams to focus on to maximize their chances of getting on base.

One of these factors is the importance of making contact with the ball. This might seem obvious but there are different approaches. Some players aim for making hard contact, potentially swinging and missing, which is often called power hitting. Some players on the other hand prioritize the contact itself and try to not strike out. Contact-focused players do not generally always hit the ball hard, but their approach might lend itself to getting on base. We do also see that the hit coordinates matter a lot and that as Figure 3.5 shows, the further away the ball is hit does not translate to a higher probability of getting on base and that placement matters. This can conversely guide defensive baseball strategy as the effect of these variables could inform defensive positioning in an effort to get batters out.

We also do see the importance of count difference with more balls than strikes increasing the probability of getting on base. This suggests that batters need to have plate discipline and to be able to get the count difference in their

favor is important. This means not swinging at every pitch as every swing is a strike if missed and that pitch selection and identification matters. Being able to identify pitches that are outside the strike zone and should not be hit at could get the count in the batter's favor which in turn possibly forces the pitcher to pitch more hittable balls in the strike zone to avoid giving the batter a walk.

5.2 Limitations & Future Opportunities

The limitations in this project are also opportunities for the future. While each event contains some information about previous events, such as the count difference that provides information about the previous pitches in the at bat, the models built in this project do not use information from previous events more than these high-level insights. Future work could include more temporal analysis, such as looking at whether these features' effects change over time, or looking at performances over time, or taking into account previous events in more detail. An example would be seeing the impact of pitch sequences, such as what impact can be seen if the pitcher has thrown the same pitch multiple times in a row.

Another opportunity is to look at more player grouped metrics and develop models that look at player types and what player types are more likely to get on base. One could then look into the possibility of using so-called rate statistics, which are the percentage of actions occurring, such as whiff rate which indicates the percentage of pitches that a batter swings and misses at.

Finally, there are opportunities and insights that can be gained from using causal inference techniques. While SHAP values are a great way to understand the impact of features on the prediction, they do not provide any information about causal relationships and instead provide more correlational information. Causal inference techniques can then be used instead to provide such insights regarding the true causal structures and account for confounding effects. This could help stakeholders and analysts more deeply understand the mechanisms at play.

References

Abma, B. J. M. [2009]. Evaluation of requirements management tools with support for traceability-based change impact analysis. *University of Twente*. https://www.utwente.nl/en/eemcs/trese/graduation/_projects/2009/Abma.pdf

Ahmed, N. A. [2023]. *What is A Confusion Matrix in Machine Learning? The Model Evaluation Tool Explained*. https://www.datacamp.com/tutorial/what-is-a-confusion-matrix-in-machine-learning

Altman, N., & Krzywinski, M. [2018]. The curse(s) of dimensionality. *Nature Methods*, *15*[6], 399–400. https://doi.org/10.1038/s41592-018-0019-x

Baseball Savant. [n.d.]. *Statcast Search CSV Documentation*. Retrieved April 27, 2024, from https://baseballsavant.mlb.com/csv-docs

Bergstra, J., Bardenet, R., Bengio, Y., & Kégl, B. [2011]. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, *24*. https://papers.nips.cc/paper/_files/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf

Bradshaw, T. J., Huemann, Z., Hu, J., & Rahmim, A. [2023]. A Guide to Cross-Validation for Artificial Intelligence in Medical Imaging. *Radiology: Artificial Intelligence*, *5*[4], e220232. https://doi.org/10.1148/ryai.220232

Brain, D., & Webb, G. I. [2002]. The Need for Low Bias Algorithms in Classification Learning from Large Data Sets. In G. Goos, J. Hartmanis, J. Van Leeuwen, J. G. Carbonell, J. Siekmann, T. Elomaa, H. Mannila, & H. Toivonen [Eds.], *Principles of Data Mining and Knowledge Discovery* [Vol. 2431, pp. 62–73]. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45681-3_6

Britannica. [2024]. *Manifold | Differential Geometry, Topology & Algebra | Britannica*. https://www.britannica.com/science/manifold

Brooks, H. [1989]. The Statistical Mirage of Clutch Hitting. *Baseball Re-*

search Journal. http://research.sabr.org/journals/the-statistical-mirage-of-clutch-hitting

Burnes, C., DeWitt, B., Gallen, Z., Johnson, G., Merrifield, W., Miller, B., Monfort, D., Shapiro, M., Slater, A., Stanton, J., Werner, T., & Mifsud, P. V. [2024]. *Official Baseball Rules 2024 Edition*. https://mktg.mlbstatic.com/mlb/official-information/2024-official-baseball-rules.pdf

Chan, M. [2008]. *Explanation of strike zone*. https://commons.wikimedia.org/w/index.php?curid=5591282

Chen, A. [2016]. How MLB's Statcast is changing game of baseball. *Sports Illustrated*. https://www.si.com/mlb/2016/08/26/statcast-era-data-technology-statistics

Chen, T., & Guestrin, C. [2016]. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Coenen, A., & Pearce, A. [2018]. Understanding UMAP. In *Google PAIR*. https://pair-code.github.io/understanding-umap/

Cramer, J. S. [2003]. The Origins of Logistic Regression. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.360300

CS Odessa Corp. [n.d.]. Baseball Diagram. In *https://www.conceptdraw.com*. Retrieved August 3, 2024, from https://www.conceptdraw.com/examples/baseball-positions

Czakon, J. [2024]. F1 Score vs ROC AUC vs Accuracy vs PR AUC: Which Evaluation Metric Should You Choose? In *neptune.ai*. https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc

DMLC. [2023]. Machine Learning Challenge Winning Solutions. In *GitHub*. https://github.com/dmlc/xgboost/tree/master/demo

Dong, D., & McAvoy, T. J. [1996]. Nonlinear principal component analysis—Based on principal curves and neural networks. *Computers & Chemical Engineering*, *20*[1], 65–78. https://doi.org/10.1016/0098-1354(95)00003-K

Eban, E. E., Schain, M., Mackey, A., Gordon, A., Saurous, R. A., & Elidan, G. [2017]. *Scalable Learning of Non-Decomposable Objectives* [arXiv:1608.04802]. arXiv. http://arxiv.org/abs/1608.04802

Galli, S. [2022]. *Feature selection with Lasso in Python*. https://www.blog.trainindata.com/lasso-feature-selection-with-python/

Gilgoldm. [2020]. *English: A tree showing survival of passengers on the Titanic ("sibsp" is the number of spouses or siblings aboard)*. https:

//commons.wikimedia.org/wiki/File:Decision/_Tree.jpg
Haben, S., Voss, M., & Holderbaum, W. [2023]. Load Forecasting Model Training and Selection. In *Core Concepts and Methods in Load Forecasting* [pp. 107–127]. Springer International Publishing. https://doi.org/10.1007/978-3-031-27852-5_8
Kim, G., Lee, S. Y., Oh, J.-S., & Lee, S. [2021]. Deep Learning-Based Estimation of the Unknown Road Profile and State Variables for the Vehicle Suspension System. *IEEE Access*, *9*, 13878–13890. https://doi.org/10.1109/ACCESS.2021.3051619
Knight, M. [2009]. Prospectus Idol Entry: Why is On Base Percentage King? In *Baseball Prospectus*. https://www.baseballprospectus.com/news/article/8938/prospectus-idol-entry-why-is-on-base-percentage-king/
Lundberg, S. [2018]. *Beeswarm plot*. https://shap.readthedocs.io/en/latest/example/_notebooks/api/_examples/plots/beeswarm.html
Lundberg, S., & Lee, S.-I. [2017]. *A Unified Approach to Interpreting Model Predictions* [arXiv:1705.07874]. arXiv. http://arxiv.org/abs/1705.07874
Major League Baseball. [2016]. *Statcast: Barrel Zone*. https://www.mlb.com/video/statcast-barrel-zone-c1152989483
Marca. [2022]. MLB Field Dimensions: What are the dimensions of a baseball field? In *MARCA*. https://www.marca.com/en/mlb/2022/03/10/622931ce22601dc5528b45e2.html
McInnes, L. [2018]. *How UMAP Works — umap 0.5 documentation*. https://umap-learn.readthedocs.io/en/latest/how/_umap/_works.html
McInnes, L., Healy, J., & Melville, J. [2020]. *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction* [arXiv:1802.03426]. arXiv. https://doi.org/10.48550/arXiv.1802.03426
McLeod, J. [2023]. Sabermetrics 101: Understanding the Calculation of WAR. In *Samford University*. https://www.samford.edu/sports-analytics/fans/2023/Sabermetrics-101-Understanding-the-Calculation-of-WAR
Molnar, C. [2022]. *Interpretable machine learning: A guide for making black box models explainable* [Second edition]. Christoph Molnar.
Mueller, B. [2015]. *The Importance of Hard-Hit Percentage*. https://community.fangraphs.com/the-importance-of-hard-hit-percentage/
Nahhas, R. W. [2024]. *Introduction to Regression Methods for Public Health Using R*. https://www.bookdown.org/rwnahhas/RMPH/
Ngo. [2018]. Principal component analysis explained simply. In *BioTuring's Blog*. https://blog.bioturing.com/2018/06/14/principal-component-analysis-explained-simply/

Nguyen, M. [2020]. *A guide on data analysis*. Bookdown. https://bookdown.org/mike/data/_analysis/

Nielsen Company, T. [2023]. *Tops of 2023: Sports | Nielsen*. https://www.nielsen.com/insights/2023/tops-of-2023-sports/

Nobel Prize Outreach. [2024]. *Lloyd S. Shapley – Facts*. https://www.nobelprize.org/prizes/economic-sciences/2012/shapley/facts/

Petti, B. [2017]. Research Notebook: New Format for Statcast Data Export at Baseball Savant. In *The Hardball Times*. https://tht.fangraphs.com/research-notebook-new-format-for-statcast-data-export-at-baseball-savant/

Powers, D. M. W. [2020]. *Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation* [arXiv:2010.16061]. arXiv. http://arxiv.org/abs/2010.16061

Quinlan, J. R. [1986]. Induction of decision trees. *Machine Learning*, *1*[1], 81–106. https://doi.org/10.1007/BF00116251

Schaul, K. [2023]. How far must a ball travel to be a home run? Depends on the MLB stadium. In *Washington Post*. https://www.washingtonpost.com/sports/interactive/2023/mlb-field-dimensions/

Scikit-learn. [n.d.]. 3.1. Cross-validation: Evaluating estimator performance. In *scikit-learn*. Retrieved August 2, 2024, from https://scikit-learn.org/stable/modules/cross/_validation.html

Shlens, J. [2014]. *A Tutorial on Principal Component Analysis* [arXiv:1404.1100]. arXiv. http://arxiv.org/abs/1404.1100

Sirakorn. [2020]. *English: Illustration of a boosting method for ensemble learning*. https://commons.wikimedia.org/wiki/File:Ensemble/_Boosting.svg

Sutelan, E. [2014]. *Taking a Closer Look at Hitting with Runners in Scoring Position*. https://community.fangraphs.com/taking-a-closer-look-at-hitting-with-runners-in-scoring-position/

Van Rijsbergen, C. J. [1979]. *Information Retrieval*. Butterworth and Co. https://www.dcs.gla.ac.uk/Keith/Preface.html

VanderPlas, J. [2016]. *Python data science handbook: Essential tools for working with data* [First edition]. O'Reilly.

Vollmer, N. [2023]. *Recital 71 EU General Data Protection Regulation (EU-GDPR)*. SecureDataService. https://www.privacy-regulation.eu/en/recital-71-GDPR.htm

XGBoost. [2022]. *Understand your dataset with XGBoost — xgboost 2.1.1 documentation*. https://xgboost.readthedocs.io/en/stable/R-package/di

scoverYourData.html

Appendix A

Appendix

A.1 Statcast CSV documentation

Column name	Definition
pitch_type	The type of pitch derived from Statcast.
game_date	Date of the Game.
release_speed	Pitch velocities from 2008-16 are via Pitch F/X, and adjusted to roughly out-of-hand release point. All velocities from 2017 and beyond are Statcast, which are reported out-of-hand.
release_pos_x	Horizontal Release Position of the ball measured in feet from the catcher's perspective.
release_pos_z	Vertical Release Position of the ball measured in feet from the catcher's perspective.
player_name	Player's name tied to the event of the search.
batter	MLB Player Id tied to the play event.

Column name	Definition
pitcher	MLB Player Id tied to the play event.
events	Event of the resulting Plate Appearance.
description	Description of the resulting pitch.
spin_dir	Deprecated field from the old tracking system.
spin_rate_deprecated	Deprecated field from the old tracking system. Replaced by release_spin
break_angle_deprecated	Deprecated field from the old tracking system.
break_length_deprecated	Deprecated field from the old tracking system.
zone	Zone location of the ball when it crosses the plate from the catcher's perspective.
des	Plate appearance description from game day.
game_type	Type of Game. E = Exhibition, S = Spring Training, R = Regular Season, F = Wild Card, D = Divisional Series, L = League Championship Series, W = World Series
stand	Side of the plate batter is standing.
p_throws	Hand pitcher throws with.
home_team	Abbreviation of home team.
away_team	Abbreviation of away team.
type	Short hand of pitch result. B = ball, S = strike, X = in play.
hit_location	Position of first fielder to touch the ball.
bb_type	Batted ball type, ground_ball, line_drive, fly_ball, popup.
balls	Pre-pitch number of balls in count.

Column name	Definition
strikes	Pre-pitch number of strikes in count.
game_year	Year game took place.
pfx_x	Horizontal movement in feet from the catcher's perspective.
pfx_z	Vertical movement in feet from the catcher's perpsective.
plate_x	Horizontal position of the ball when it crosses home plate from the catcher's perspective.
plate_z	Vertical position of the ball when it crosses home plate from the catcher's perspective.
on_3b	Pre-pitch MLB Player Id of Runner on 3B.
on_2b	Pre-pitch MLB Player Id of Runner on 2B.
on_1b	Pre-pitch MLB Player Id of Runner on 1B.
outs_when_up	Pre-pitch number of outs.
inning	Pre-pitch inning number.
inning_topbot	Pre-pitch top or bottom of inning.
hc_x	Hit coordinate X of batted ball.
hc_y	Hit coordinate Y of batted ball.
tfs_deprecated	Deprecated field from old tracking system.
tfs_zulu_deprecated	Deprecated field from old tracking system.
fielder_2	Pre-pitch MLB Player Id of Catcher.
umpire	Deprecated field from old tracking system.
sv_id	Non-unique Id of play event per game.
vx0	The velocity of the pitch, in feet per second, in x-dimension, determined at y=50 feet.

Column name	Definition
vy0	The velocity of the pitch, in feet per second, in y-dimension, determined at y=50 feet.
vy0	The velocity of the pitch, in feet per second, in z-dimension, determined at y=50 feet.
ax	*The acceleration of the pitch, in feet per second per second, in x-dimension, determined at y=50 feet.
ay	The acceleration of the pitch, in feet per second per second, in y-dimension, determined at y=50 feet.
az	The acceleration of the pitch, in feet per second per second, in z-dimension, determined at y=50 feet.
sz_top	Top of the batter's strike zone set by the operator when the ball is halfway to the plate.
sz_bot	Bottom of the batter's strike zone set by the operator when the ball is halfway to the plate.
hit_distance	Projected hit distance of the batted ball.
launch_speed	Exit velocity of the batted ball as tracked by Statcast. For the limited subset of batted balls not tracked directly, estimates are included based on the process described here.
launch_angle	Launch angle of the batted ball as tracked by Statcast. For the limited subset of batted balls not tracked directly, estimates are included based on the process described here.

Column name	Definition
effective_speed	Derived speed based on the the extension of the pitcher's release.
release_spin	Spin rate of pitch tracked by Statcast.
release_extension	Release extension of pitch in feet as tracked by Statcast.
game_pk	Unique Id for Game.
pitcher	MLB Player Id tied to the play event.
fielder_2	MLB Player Id for catcher.
fielder_3	MLB Player Id for 1B.
fielder_4	MLB Player Id for 2B.
fielder_5	MLB Player Id for 3B.
fielder_6	MLB Player Id for SS.
fielder_7	MLB Player Id for LF.
fielder_8	MLB Player Id for CF.
fielder_9	MLB Player Id for RF.
release_pos_y	Release position of pitch measured in feet from the catcher's perspective.
estimated_ba_using_speedangle	Estimated Batting Avg based on launch angle and exit velocity.
estimated_woba_using_speedangle	Estimated wOBA based on launch angle and exit velocity.
woba_value	wOBA value based on result of play.
woba_denom	wOBA denominator based on result of play.
babip_value	BABIP value based on result of play.
iso_value	ISO value based on result of play.
launch_speed_angle	Launch speed/angle zone based on launch angle and exit velocity. 1: Weak 2: Topped 3: Under 4: Flare/Burner 5: Solid Contact 6: Barrel
at_bat_number	Plate appearance number of the game.

Column name	Definition
pitch_number	Total pitch number of the plate appearance.
pitch_name	The name of the pitch derived from the Statcast Data.
home_score	Pre-pitch home score
away_score	Pre-pitch away score
bat_score	Pre-pitch bat team score
fld_score	Pre-pitch field team score
post_home_score	Post-pitch home score
post_away_score	Post-pitch away score
post_bat_score	Post-pitch bat team score
if_fielding_alignment	Infield fielding alignment at the time of the pitch.
of_fielding_alignment	Outfield fielding alignment at the time of the pitch.
spin_axis	The Spin Axis in the 2D X-Z plane in degrees from 0 to 360, such that 180 represents a pure backspin fastball and 0 degrees represents a pure topspin (12-6) curveball
delta_home_win_exp	The change in Win Expectancy before the Plate Appearance and after the Plate Appearance
delta_run_exp	*The change in Run Expectancy before the Pitch and after the Pitch

A.2 UMAP Explanation

To construct the high-dimensional graph, UMAP uses a concept called simplices. A simplex is essentially a way of representing a k-dimensional object, where a k-simplex is k dimensional. Hence, a 0-simplex is a point, a 1-simplex is a line, a 2-simplex is a triangle et cetera. Each k-dimensional simplex is formed by using $k+1$ $k-1$-dimensional simplices. This means that a 1-simplex is formed by using 2 0-simplices and a 2-simplex is formed by using 3 1-simplices. This process then makes it easy to generalize to higher and

higher dimensional simplices.

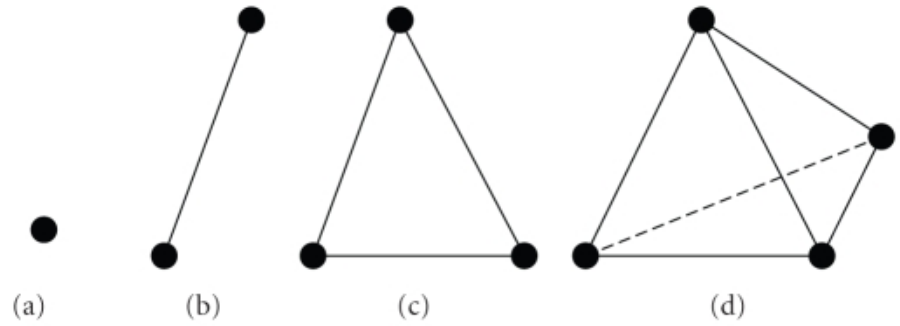


Figure A.1: The four first simplices [McInnes, 2018]

We consider the data points in our dataset to be 0-simplices and connect them to their neighboring data points to form 1-simplices in a so-called simplicial complex. With that, the topology of the manifold can be approximated. To connect each data point, UMAP creates radii that extend from the data point outwards and when data points overlap, they get connected. Then all connections get weights that indicate connection probability. Thus, when points are far apart, the weight is low and closer points are higher weighted [Coenen & Pearce, 2018].

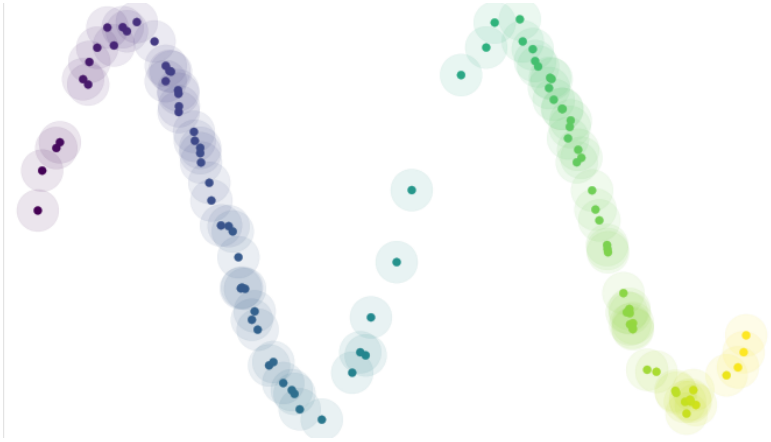


Figure A.2

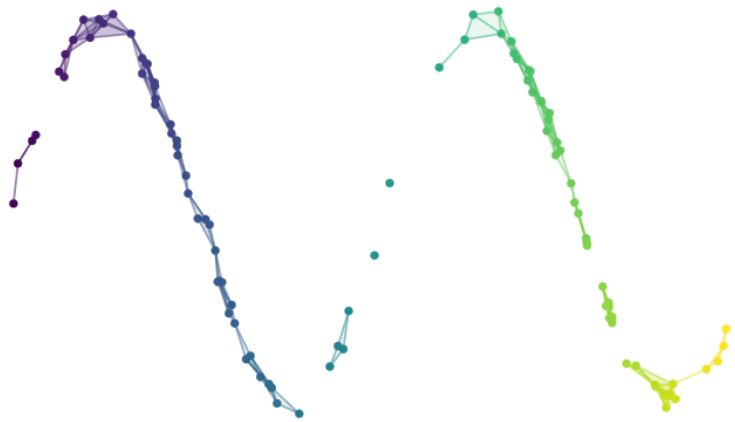


Figure A.3: Images showing the creation of radii from each data point and the subsequent formation of the simplical complex [McInnes, 2018].

To find the low-dimensional graph representation, we seek to minimize a loss function that is based on the cross entropy loss. We define the set of all possible 1-simplices as E and the weight functions $w_H(e)$, $w_L(e)$ to be the weights of the 1-simplex e in the high-dimensional and low-dimensional graph representations respectively. The loss function will then be similar to previously discussed loss functions, specifically we get:

$$\sum_{e \in E} w_H(e) \ln\left(\frac{w_H(e)}{w_L(e)}\right) + (1 - w_H(e)) \ln\left(\frac{1 - w_H(e)}{1 - w_L(e)}\right)$$

There are two terms here. The first term $w_H(e) \ln\left(\frac{w_H(e)}{w_L(e)}\right)$ will be minimized as $w_L(e)$ increases, which indicates inter-point distance to be small, whereas the second term $(1 - w_H(e)) \ln\left(\frac{1-w_H(e)}{1-w_L(e)}\right)$ will be minimized as $w_L(e)$ decreases, which indicates inter-point distance to be large. This is the balance that the loss function needs to achieve and minimize and when that is done, the low-dimensional graph representation is found [McInnes et al., 2020].

A.3 XGBoost Mathematical Details

$$\Omega(f_t) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \tag{A.1}$$

where γ, is the complexity control parameter, T is n the number of leaves in the tree, λ is a regularization term, and w_j are the leaf weights [T. Chen & Guestrin, 2016]. Equation A.1 can be unwieldy to work with and therefore XGBoost instead uses the second-order Taylor approximation of the loss function to simplify calculations. Doing that results in:

$$\text{Obj}^{(t)} = \sum_{i=1}^{n} \left[L(y_i, p_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) \tag{A.2}$$

where

$$g_i = \frac{\partial}{\partial p_i^{(t-1)}} L(y_i, p_i^{(t-1)})$$
$$h_i = \frac{\partial^2}{\partial p_i^{2(t-1)}} L(y_i, p_i^{(t-1)})$$

When removing constant terms, Equation A.2 simplifies to:

$$\text{Obj}^{(t)} = \sum_{i=1}^{n} [g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t) \tag{A.3}$$

This is the objective function to minimize. We can re-formulate the model by defining I_j as the instance of all data points of the j-th tree. Then each f_t becomes ω_j and to find the minimum we derive the objective function with respect to ω_j:

$$\frac{\partial}{\partial \omega_j} \text{Obj}^{(t)} = \sum_{i \in I_j} [g_i + (h_i + \lambda)\omega_j] = 0$$
$$\omega_j^* = \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} (h_i + \lambda)}$$

A.4 Hyperparameter Tuning Results

For the logistic regression model the optimal hyperparameters were, as presented in Table A.2:

Table A.2: The optimal hyperparameters for the logistic regression model.

Parameter	Optimal value
alpha	0.000034
learning_rate	adaptive
eta0	0.00346
class_weight	balanced

These hyperparameters resulted from the following optimization history, as presented in Figure A.4:

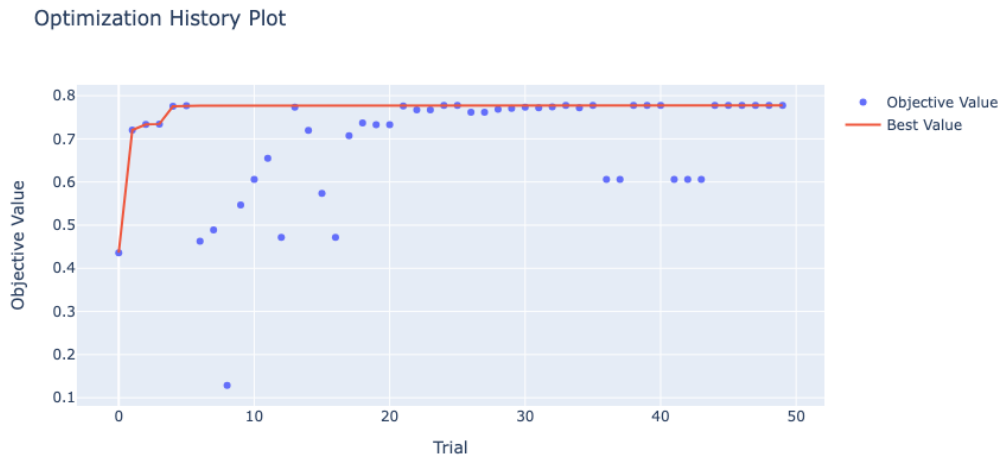


Figure A.4: Hyperparameter Optimization History

where the objective function was the mean F2 score. We can see that the algorithm reached near optimal objective values already in the first 10 trials and that after approximately 20 trials it very rarely explored unfavorable regions of the parameter space.

Looking at the XGBoost model the optimal hyperparameters were, as presented in Table A.2::

Table A.3: The optimal hyperparameters for the XGBoost model.

Parameter	Optimal value
max_depth	7
learning_rate	0.0958
n_estimators	803
min_child_weight	1
gamma	0.4267
reg_alpha	9.4585
scale_pos_weight	4.9604
subsample	0.7873
colsample_bytree	0.6680

Again we present the optimization history, as shown in Figure A.5:

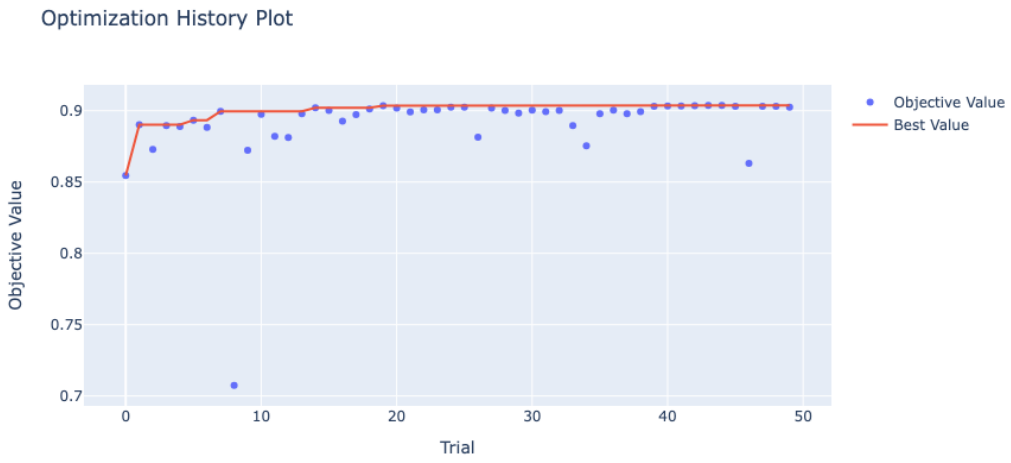


Figure A.5: Hyperparameter Optimization History

We again see that the algorithm is very quick with reaching near optimal objective values and with XGBoost it is even quicker in abandoning unfruitful paths.