Samuel Kim

Using Statistical Classification Methods to Predict Hits Using Statcast

Abstract:

Expected Batting Average in Major League Baseball is currently calculated mainly by using two different factors for a batted ball: launch angle (the vertical angle at which the ball leaves a player's bat after being struck) and exit velocity (speed of the baseball as it comes off the bat, immediately after a batter makes contact). However, other factors that may affect whether a ball results in a hit or an out, such as the stadium where the batted ball occurred, the defensive team at the time of the batted ball, and the distance the ball travels in the air, are not analyzed in the model. Therefore, three different statistical learning methods: logistic regression models, generalized additive models, and random forests will be applied to a dataset that contains launch angle, exit velocity, and various other possible factors for every batted ball in the 2022 season. From there, analysis will be done to determine whether models that contain other factors were more effective at predicting hits than the model with just launch angle and exit velocity.

Introduction:

Batting Average (AVG) - which is the rate at which a baseball player gets a hit[1] - has always been looked at as one of the most important statistics that contributes to a Major League Baseball team's success on offense. It is a simple concept: to win, a team must score more runs than the team they are playing, to score more runs than the other team, a team must get more hits than the other team. Although the importance of AVG was universally known in the baseball world, it was much more difficult knowing what factors contributed to a player being more likely to get a hit than any other player. That is, until 2015, when Major League Baseball implemented statcast into every one of their stadiums.

Statcast is a combination of high-speed cameras and a high-accuracy automated tool that is responsible for accurately tracking speed, movement, and spin on each pitch, player, and hit at any point in a MLB game[2]. Quickly after 2015, every MLB team began developing their own analytics teams to use statcast data in hopes of finding undervalued players, and to improve their current players.[3] According to Forbes, the average MLB team is currently worth $2.07 billion and the Miami Marlins are the only team

[1] MLB, "Batting Average: Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/standard-stats/batting-average.
[2] MLB, "Statcast: Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast.
[3] Albert Chen, "How MLB's Statcast Is Changing Game of Baseball," Sports Illustrated (Sports Illustrated, August 26, 2016), https://www.si.com/mlb/2016/08/26/statcast-era-data-technology-statistics.

valued at below $1 billion, with their current franchise evaluation at $990 million[4]. Due to the fact that professional baseball is a multi-billion dollar industry, teams are not willing to disclose any information on how they are using the statcast data to improve their teams.

Currently, the most official public information on how statcast can be used is shown through expected statistics, which is a method used to capture the variance in an outcome in an MLB game by investigating the differences between actual and expected outcomes[5]. To explore the factors that contribute to AVG, we use the Major League Baseball-defined Expected Batting Average (xBA), which is "how often comparable balls -- in terms of exit velocity, launch angle and, on certain types of batted balls, Sprint Speed -- have become hits"[6]. Exit velocity is defined as the "speed of the baseball as it comes off the bat, immediately after a batter makes contact"[7], launch angle is defined as "the vertical angle at which the ball leaves a player's bat after being struck"[8], and sprint speed is defined as "how many feet per second a player runs in his fastest one-second window… The best of these runs, approximately two-thirds, are averaged for a player's seasonal average"[9]. Seasonal sprint speed is not public information, so we will further analyze exit velocity and launch angle.

Due to the nature of baseball, a hitter can do everything correctly in his control; guessing the exact pitch and location, along with producing the most optimal launch angle and exit velocity, but still, be punished by uncontrollable changing factors. One of these possible factors is field dimensions, with every MLB team having different distances to the wall and wall heights, creating instances where a batted ball would have been a home run in a certain stadium, and an out in another[10], causing variance in batted ball outcomes with the same launch angle and exit velocity. Another factor is climate, with different cities having stronger wind or higher elevation, which can both heavily influence how far a baseball will travel,

---

[4] Christina Gough, "MLB Franchise Values US 2022," Statista, November 10, 2022, https://www.statista.com/statistics/193637/franchise-value-of-major-league-baseball-teams-in-2022/.
[5] Sam Bornstein, "A Guide to Baseball's Expected Statistics," Simple Sabermetrics (Simple Sabermetrics, August 6, 2021), https://www.simplesabermetrics.com/post/a-guide-to-baseball-s-expected-statistics.
[6] MLB, "Expected Batting Average (Xba): Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast/expected-batting-average.
[7] MLB, "Exit Velocity (EV): Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast/exit-velocity.
[8] MLB, "Launch Angle (LA): Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast/launch-angle.
[9] MLB, "Sprint Speed (SS): Glossary," MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast/sprint-speed.
[10] Josh Wilson, "Aaron Judge's near-Home Run Just Needed a Convenient Change of Scenery," FanSided (FanSided, October 21, 2022), https://fansided.com/2022/10/21/aaron-judges-near-home-run-just-needed-convenient-change-scenery/.

causing variance in batted ball outcomes with the same launch angle and exit velocity. The skill of defenders also changes frequently, with every MLB team having different players, which causes scenarios where a defender can catch a certain hit, while another defender can't[11]; in turn, creating variance in batted ball outcomes with the same launch angle and exit velocity. After looking at a few different examples of how factors outside of launch angle and exit velocity can affect whether a batted ball was a hit or not, we ask the question: are there any further factors that can be added to create an xBA model that performs better than a model that only contains predictors given in the current Major League xBA formula?

Our data set contains every batted ball in the 2022 regular season, of which there were 108,707 with 10 possible explanatory variables, along with the response variable. Further discussion on the variables will take place in the "Data" section below. We apply three different methods of statistical learning - logistic regression, generalized additive models, and random forests - to our data in order to accomplish the goal of finding any additional factors that can be added to create an xBA model that performs better than a model that contains predictors given in the current Major League xBA formula.

Data:

As mentioned before, Major League Baseball unfortunately defines sprint speed as seasonal sprint speed, which can only be obtained by Major League Baseball and its teams; rendering sprint speed as a factor that can not be considered in our model. However, exit velocity, launch angle, home team, fielding team, batter handedness, pitcher handedness, ball distance traveled in the air (in feet), defensive strategy of the outfield and infield, and whether the hitter's team was losing are all potential factors in hits that are available for every pitch if back end web scraping is done on MLB.com[12]. These factors will look to explain our binomial response variable of whether the actual outcome of the batted ball was a hit. To make sure all of the data is up to date, we consider only the 2022 season, which was the last full season played by the MLB. Furthermore, to keep all of our data as consistent as possible, only regular season games will be used since playoff games have even further non-skill implications, such as a player's mentality that are not quantifiable. The 10 possible explanatory variables and the response variable are

---

[11] Bryce Grafton, "From New York Mets to San Diego Padres: Here Are the Top 5 Defensive Teams in 2022 MLB Season so Far," Sports news (Sportskeeda, May 31, 2022), https://www.sportskeeda.com/baseball/from-new-york-mets-san-diego-padres-here-top-5-defensive-teams-2022-mlb-season-far.

[12] Bill Petti, "How to Build a Statcast Database from Baseballsavant, v3.0," Root Mean Squared Musings (B, April 2, 2021), https://billpetti.github.io/2021-04-02-build-statcast-database-rstats-version-3.0/.

listed below in table 1, along with the five number summaries for quantitative variables, and frequencies for qualitative variables.

Table 1: Summaries For All 11 Variables in Analysis

Variable Name	Type Of Variable	Summary of Variable
Exit Velocity (in MPH)	Quantitative, Explanatory	Min: 5.2 1st Quartile: 79.7 Median: 91 3rd Quartile: 99.2 Max: 122.4 Mean: 88.1
Launch Angle (in degrees)	Quantitative, Explanatory	Min: -90 1st Quartile: -6 Median: 13 3rd Quartile: 31 Max: 90 Mean: 12.4
Distance (in feet)	Quantitative, Explanatory	Min: 0 1st Quartile: 18 Median: 153 3rd Quartile: 285 Max: 504 Mean: 163
Infield Defensive Shift	Qualitative, Explanatory	Standard: 99567 Strategic: 9140
Outfield Defensive Shift	Qualitative, Explanatory	Standard: 103218 Strategic: 5489
Batter Handedness	Qualitative, Explanatory	Right: 66055 Left: 42652
Pitcher Handedness	Qualitative, Explanatory	Right: 79029 Left: 29678
Fielding Team	Qualitative, Explanatory	No Concern For Non-Even Distribution Among All 30 MLB Teams
Home Team	Qualitative, Explanatory	No Concern For Non-Even Distribution Among All 30 MLB Teams
Whether They Were Losing	Qualitative, Explanatory	Yes: 39359 No: 69348
Whether It Was A Hit	Qualitative, Response	Yes: 34645 No: 74062

Although many of the variables are non-evenly and non-normally distributed, all three of our methods - logistic regression, generalized additive models, and random forests - do not assume normality or evenly-distributed data sets; therefore, no further transformation is needed on these variables. Considering that after MLB implemented hawk-eye technology to record their statcast statistics, there was an increase in percentage of batted balls tracked to 99%[13], we did not have any concern with missing data. There is also no concern in reliability for this data set, as all data came from an official Major League Baseball database.

[13] MLB, “Statcast: Glossary,” MLB.com (MLB.com), accessed November 10, 2022, https://www.mlb.com/glossary/statcast.

Methods

As mentioned earlier, three methods of statistical learning - logistic regression, generalized additive models, and random forests - will be applied to the dataset to accomplish the goal of finding the best possible combination of variables to predict hits through expected batting average.

Logistic Regression:

When using a linear model on our binomial response variable, outcome of a batted ball, some estimates may be outside of the [0,1] interval, making the results hard to interpret as probabilities[14]. To counter this issue, we apply a logistic function - which is a function that ensures the outputs are between 0 and 1 - to a linear model. Below is the application of the logistic function to a linear regression model (equation 1), creating a logistic regression model (equation 2):

Equation 1: Linear Regression Model:

$$\pi(y_i) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \dots \beta_v x_{iv} + \in_i$$

Equation 2: Logistic Regression Model:

$$\pi(y_i) = \exp\{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \dots \beta_v x_{iv} + \in_i \}/1 + \exp\{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \dots \beta_v x_{iv} + \in_i \}$$

where $\pi(y_i)$ represents the probability of getting a hit for the i-th batted ball (where i represents a batted ball in our data), $\beta = (\beta_1 + \beta_2 + \beta_3 + \dots \beta_v)$ are the regression coefficients for the explanatory variables $x = (x_{i1} + x_{i2} + x_{i3} + \dots x_{iv})$ for v number of explanatory variables on our i-th batted ball. β_0 represents the value of y_i if all the regression coefficients for the x_i's were 0, and $\in_i$ is the error term for our i-th batted ball. The error term "includes everything that separates your model from actual reality", which means that the term reflects the nonlinearities, unpredictable effects, measurement errors, and omitted variables in our model[15].

The coefficients for $\beta_0 + \beta_1 + \beta_2 + \beta_3 + \dots \beta_v$ are unknown and have to be estimated based on the given data. The method of maximum likelihood is used to obtain the unknown coefficients. We seek estimates for each of our β_v (where v is an explanatory variable), such that the predicted probability of getting a hit $\widehat{\pi}(x_i)$ (where i represents a batted ball in our data) for each batted ball corresponds as closely as possible to the actual result of the batted ball. Essentially, we attempt to find a value for each of our

[14] Gareth James et al., "Chapter 4: Classification," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).
[15] Stephanie, "Error Term: Definition and Examples," Statistics How To, November 16, 2020, https://www.statisticshowto.com/error-term/.

predicted explanatory variable coefficients, $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots \hat{\beta}_v$ such that plugging in these estimates to equation 2 will result in a number close to 1 for all batted balls that resulted in a hit, and a number close to 0 for all batted balls that resulted in an out. This concept can be visualized in equation 3 as:

Equation 3: Maximum Likelihood Function For Logistic Regression Model

$$L(\beta_1, \beta_2, \beta_3, \ldots \beta_v) = \prod_{i=1}^{n} \pi(x_{iv})^y (1 - \pi(x_{iv})^{1-y}), (y = y_i)$$

where y_i represents an observed batted ball, and x_{iv} is a vector of v variables. The probability was either π, if $y_i = 1$, or $1 - \pi$, if $y_i = 0$ (where i represents a batted ball in our data)[16]. The estimates $\hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3, \ldots \hat{\beta}_v$ are chosen with the motive of maximizing this likelihood function.

We can further manipulate the logistic regression model to find the equation for odds, which is shown through equation 4:

Equation 4: Odds (Divide Equation 2 by $1 - \pi(y_i)$):

$$\frac{\pi(y)}{1-\pi(y)} = \exp\{\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \ldots \beta_v x_{iv} + \epsilon_i\}, (y = y_i)$$

Odds can take on any value between 0 and ∞, with odds of close to 0 and ∞ indicating very low and high probabilities of the batted ball resulting in a hit, respectively. By taking the logarithm of both sides of this equation, we produce an equation for the log odds (logit), which is linear in our variables x_{iv}. It is important to remember that we have not changed any values by transforming our original logistic regression model (equation 2) to our logit model, which means that our model for the logit (equation 5) is still a logistic regression model:

Equation 5: Logit (Take Log of Both Sides of Equation 4):

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i3} + \ldots \beta_v x_{iv} + \epsilon_i$$

however, by having a linear equation for the logit value, analysis is simplified. For example, a one-unit change in x_{iv} results in a change of the logit by β_v (where v is a random explanatory variable). Furthermore, the understanding of the future methods is heavily helped with our logit model above. Lastly, it is important to note that qualitative variables will be treated as dummy variables, with a different value of the regression coefficient β being applied to the variable based on the classification that the variable receives. For visualization purposes, let us imagine that $\beta_3 x_{i3}$ is our qualitative variable home team. Each team will receive a different value of β_3; for example, the Boston Red Sox are represented by

[16] Carnegie Mellon, "Logistic Regression - Carnegie Mellon University," Logistic Regression, accessed November 11, 2022, https://www.stat.cmu.edu/~cshalizi/uADA/12/lectures/ch12.pdf.

β_{3B}, and the New York Yankees are represented by β_{3Y} in a logistic regression model. If the batted ball that is being analyzed happened when the Red Sox were the home team, the above model would be represented by:

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_{3B} x_{i3} + \dots \beta_v x_{iv} + \in_i$$

while a batted ball that occurred when the Yankees were the home team would be shown with:

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_{3Y} x_{i3} + \dots \beta_v x_{iv} + \in_i$$

(note: other variables were left untouched to show how changing the home team effects the model)

Generalized Additive Models:

The Logistic Regression can be defined as a generalized linear model since the outcome y_i always depends on the sum of parameters x (x_{i1}, x_{i2}, x_{i3},... x_{iv}). Given the linear nature of the Logistic Regression model, we further explore non-linear models, particularly generalized additive models, for our next method of statistical learning. For easier visualization, we will first analyze our logit model, but with only one explanatory variable:

Equation 6: Logit Model With One Explanatory Variable

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \in_i$$

In this equation, our variable, x_{i1} is explained by the regression coefficient β_1, creating a linear relationship between our variable x_{i1} and the probability of getting a hit $\pi(y_i)$. There are numerous ways to apply non-linearity to this equation; however, splines were used in our analysis. In order to understand splines, we must first analyze piecewise polynomial regression, which is derived from polynomial regression.

A very basic way of applying non-linearity to an equation is through polynomial regression. An example of an application of polynomial regression comes in equation 7 below, which is a polynomial function of our logit model.

Equation 7: Polynomial Model of Equation 6

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i1}^3 + \dots \beta_d x_{i1}^d + \in_i$$

where d is the highest exponent in our polynomial model. Each of the coefficients β_0, β_1, β_2, β_3, … β_d can then be estimated using maximum likelihood since it is now a logistic regression model just like equation 5, with x_{i1}, x_{i1}^2, x_{i1}^3, … x_{i1}^d as the explanatory variables, rather than x_{i1}, x_{i2}, x_{i3}, … x_{iv}. All coefficients and explanatory variables are considered to create a polynomial curve of d degrees. Since the polynomials are fitted over the entire range of our variable x_{i1}, it may take a high value of d to effectively explain our

variable. However, using a value of d that is higher than 4 can cause the polynomial curve to become over-flexible and take on strange shapes[17]. Furthermore, the value of d is considered the degrees of freedom in our model, and high degrees of freedom in a model can also cause overfitting, which is what happens when the model starts to tailor towards our specific data set, rather than modeling for the true population[18]. This flaw in polynomial regression is accounted for by piecewise polynomial regression. More specifically, we will discuss piecewise cubic polynomials, which starts off as a standard cubic polynomial regression model:

Equation 9: Cubic Polynomial Model (3rd Degree)

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i1}^2 + \beta_3 x_{i1}^3 + \in_i$$

However, instead of each coefficient β_0, β_1, β_2, and β_3 being the same throughout the range of our variable x_{i1}, these values differ based on the value of our variable x_{i1}. The points where the coefficients change are called knots. For example, equation 9 above is a piecewise cubic polynomial with no knots, while the equation for a piecewise cubic polynomial with one knot at point c is written as:

Equation 10: Cubic Piecewise Polynomial Model For One Knot

$$\text{Logit}(\pi(y_i)) = \beta_{01} + \beta_{11} x_{i1} + \beta_{21} x_{i1}^2 + \beta_{31} x_{i1}^3 + \in_i, \text{ if } x_{i1} < c$$

$$\text{Logit}(\pi(y_i)) = \beta_{02} + \beta_{12} x_{i1} + \beta_{22} x_{i1}^2 + \beta_{32} x_{i1}^3 + \in_i,, \text{ if } x_{i1} \geq c$$

where c represents a value that is within the range of x_{i1}. As one can see from equation 10, we are essentially fitting two different cubic polynomial models on our data, one that will be applied on the subset of data where $x_{i1} < c$, and the other on the subset of data where $x_{i1} \geq c$.[19] It is also important to note that the first cubic polynomial function has the coefficients β_{01}, β_{11}, β_{21}, and β_{31}, while the second function has the coefficients β_{02}, β_{12}, β_{22}, and β_{32}, showing that different coefficient values will be applied depending on the value of x_{i1}. There is one major flaw in piecewise polynomials however, which is highlighted by figures in *"An Introduction to Statistical Learning: With Applications in R, 2022 Edition"*[20]. Although the book uses a model that has a continuous response variable rather than a binomial response variable like our model uses, the figures are still applicable when talking about the general flaw of using a piecewise polynomial:

---

[17] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).
[18] Minitab Blog Editor, "Beware of Phantom Degrees of Freedom That Haunt Your Regression Models!," Minitab Blog, accessed November 15, 2022, https://tinyurl.com/mrxhjcxu
[19] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).
[20] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).

Figure 2[21]: Visualization of a Possible Piecewise Cubic Function

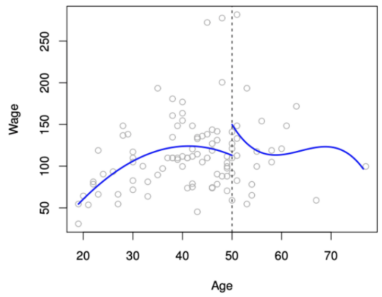
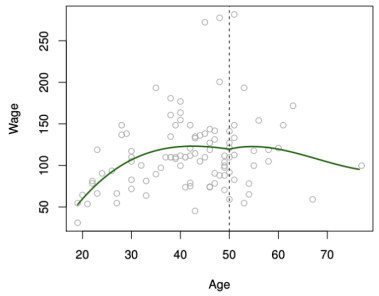


Figure 2 above is a representation of a piecewise function that is attempting to find wage using age, with a knot placed at age = 50. However, one can see from the figure that the function is discontinuous and the curve overall looks out of place. This likely occurred due to the fact that the regression coefficients for the polynomial model for those with an age of less than 50, were drastically different to the regression coefficients for the polynomial for those with an age of more than 50. To fix this possible problem, one can choose to place constraints on the piecewise polynomial. One constraint is that the final fitted curve must be continuous at the knot. By placing this constraint on the curve, we have created a continuous piecewise function and fixed the problem of discontinuity in figure 2.

Figure 3[22]: Visualization of a Possible Continuous Piecewise Cubic Function

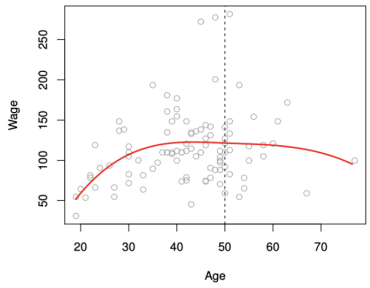


From Figure 3 above, we see a much smoother and more natural-looking curve; however, the v-shape at the knot causes the curve to still look unnatural. One more constraint is placed on the function; this time, not only are the polynomial functions constrained to be continuous at the knot but are also constrained to have first and second derivatives that are continuous at the knot:

---

[21] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).
[22] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).

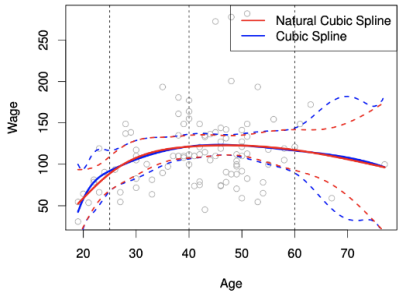
Figure 4[23]: Visualization of a Possible Cubic Spline



As it turns out, constraining the first and second derivative to be continuous at the knot allows for a much smoother and natural curve. This type of constraint on a cubic piecewise function creates a cubic spline. It is important to state that the examples of these constraints were placed on a piecewise cubic function with only one knot. Knots can be increased to create a more flexible piecewise polynomial or spline, and one can require continuity at each knot.

When there are numerous knots, cubic splines can have high variance on the outer range of the predicting variable; in other words, when our variable x_{i1} has a value that is smaller than the placement of the first or last knot. To remedy these issues of high variance, one can place one more constraint on our function, which is to force the function to be linear in the regions where the predicting variable has a value of less than the placement of the first knot, or greater than the placement of the last knot. By adding this third constraint to our function, we create a natural cubic spline.

Figure 5[24]: Overlay of Cubic Spline and Natural Cubic Spline



In Figure 5 above, we see a red line to represent natural cubic splines, and a blue line to represent the cubic spline, along with dotted lines of the same color to represent the confidence bands for their respective splines. There are three knots applied to each of these splines, one at age = 25, one at age = 40,

[23] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).
[24] Gareth James et al., "Chapter 7: Moving Beyond Linearity," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).

and one at age = 60. By analyzing the confidence bands, we see that the bands are a lot more wide in the regions before the first and after the last knot, while the bands for the natural cubic splines are narrow throughout the graph.

By placing several constraints on the piecewise cubic function, we show that the function can be improved to a natural cubic spline to help with issues of variance. Recall how the degrees of freedom in a polynomial model were represented by the highest exponent in the polynomial model. Degrees of freedom in natural cubic splines can be calculated by K + 4, where K are the number of knots in the natural cubic spline. Just like polynomial models, one must find a value for degrees of freedom that is not too high, as that causes overfitting and over flexibility of the spline, while considering a value high enough to explain the variable effectively. The natural cubic spline is most flexible in regions with lots of knots since the regression coefficients can change rapidly. Therefore, one may choose to place more knots in regions that they feel contain the values of x_{i1} that vary most rapidly. There are many ways to determine how many knots should be in the spline, along with where each one should be placed; however, what would be the result if a knot was placed at every value for the variable x_{i1}?

At first glance, it seems that if we were to apply a knot at each value for our variable in our natural cubic spline, it would have far too many degrees of freedom, creating overfitting, over flexibility, and overall an unnatural shape for our spline. However, one can apply a smoothing parameter λ on the natural cubic spline to create a shrunken version of a natural cubic spline, a smoothing spline. Although our data contains a large number of batted balls n; hence, causing the smoothing spline to have n number of degrees of freedom, these n parameters are heavily constrained or shrunk down by the smoothing parameter λ. Therefore, instead of analyzing the degrees of freedom, one uses the value of λ to represent the effective degrees of freedom. With a smoothing spline, effective degrees of freedom are the degrees of freedom that one must control, making sure the value is not too high or too low. We will use cross-validation to find the ideal λ value of each of our smoothing splines. A more in-depth discussion on cross-validation specific to our smoothing splines will be discussed in the cross-validation section. By applying a smoothing spline to our explanatory variable x_{i1}, we have added non-linearity to our logit model given in equation 6. For reference, equation 6 is as follows:

Equation 6: Logit Model With One Explanatory Variable

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1 x_{i1} + \in_i$$

The application of non-linearity to this equation can be written as:

Equation 11: Logit Model With Smoothing Spline Applied to One Explanatory Variable

$$\text{Logit}(\pi(y_i)) = \beta_0 + f_1(x_{i1}, \lambda) + \epsilon_i$$

where the function f represents the smoothing spline, and λ represents the effective degrees of freedom of the smoothing spline applied to the variable x_{i1}. By expanding the above equation to allow for multiple explanatory variables, we create a generalized additive model (GAM). It is important to note that smoothing splines will only be applied to quantitative variables, while qualitative variables will be treated as dummy variables, just like in logistic regression. Our generalized additive model can therefore be represented by:

Equation 12: Generalized Additive Model

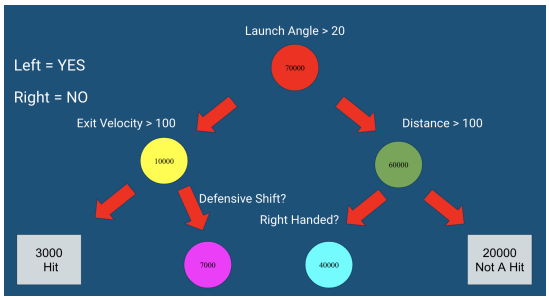
$$\text{Logit}(\pi(y_i)) = \beta_0 + f_1(x^n_{i1}, \lambda) + f_2(x^n_{i2}, \lambda) + f_3(x^n_{i3}, \lambda) + \dots f_v(x^n_{iv}, \lambda) + \beta_1 x^c_{i1} + \beta_2 x^c_{i2} + \beta_3 x^c_{i3} + \dots \beta_v x^c_{iv} + \epsilon_i$$

where x^n represents all numerical (quantitative) variables, and x^c represents all categorical (qualitative) variables. V represents the number of variables for each respective type of explanatory variable.

Random Forest:

For even further expansion on our model, we explore a method of classification based heavily on decision tree modeling: random forests. Decision trees are best explained through an example, consider figure 6 below:

Figure 6: Example Decision Tree



In our figure above, we see that the batted balls in our hypothetical 70000 batted ball dataset are split based on whether their launch angle is greater than 20 degrees. To be specific, the decision tree determined the classifier "launch angle <20" to have the lowest Gini index value based on the dataset. (definition of the Gini index will be discussed below). 10000 batted balls are determined to have a greater launch angle than 20 degrees; therefore, those 10000 are then assessed on the classifier that is considered to have the least Gini index value for batted balls with a higher than 20-degree launch angle. This

classifier is “exit velocity >100”, which means that the decision tree prompts each of the 10000 batted balls on whether their exit velocity was greater than 100 MPH. 3000 are determined to have both a launch angle greater than 20 degrees and an exit velocity greater than 100 MPH, which is enough information for the model to predict a hit for those 3000 batted balls. The 7000 batted balls that do not have an exit velocity of greater than 1000 MPH do not have enough information for the model to predict the result of the batted balls. Therefore, those 7000 batted balls are assessed on the variable that is considered to have the lowest Gini index for batted balls with a higher than 20-degree launch angle, and lower than a 100 MPH exit velocity.

Nodes will continue to be split until there is enough information known about every batted ball, or if there is a node small enough to cause concerns of overfitting. If there is a node later on in the tree that analyzes a small subset of batted balls, then there is a danger of those batted balls containing too much information about the data, essentially memorizing which factors contribute to a hit and which factors don’t, rendering the model’s performance to be poor if it is assessed on an unknown dataset[25]. Given that our data contains over 100,000 batted balls, we determine that a good node size to force a prediction is 1000. This means that if there is a split that results in a node that has less than 1000 batted balls, those batted balls will have a prediction forced upon them since they should have enough information for a prediction

The Gini index was mentioned multiple times in our example above and is written as:

Equation 13: Gini Index

$$G = \sum_{k-1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$$

In this equation, $\hat{p}_{mk}$ represents the proportion of observations in the mth node that have the kth classification (either hit or not a hit for our data). If most or all of the $\hat{p}_{mk}$’s are close to zero or one, then the Gini index has a small value. In other words, the variable with the smallest Gini index value at a given node is the variable that has the lowest proportion of observations that would be incorrectly classified. In our example above, by saying that “launch angle <20” had the lowest Gini index at the first node, it means that if we were to classify hits based purely on whether the launch angle was greater than or below 20, that classification would yield the lowest proportion of batted balls that were incorrectly classified as a

[25] Gareth James et al., “Chapter 5: Resampling Methods,” in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).

hit or not a hit. The decision tree looks at every possible split of every independent variable to find the lowest Gini index possible; however, due to this and how much data a decision tree is analyzing at once, decision trees are very prone to overfitting, which causes high variance. Essentially, the decision tree is memorizing the data and creating a model based on the given data; however, when we give a similar dataset with different batted balls, the memorization of the past dataset will prove to be unhelpful in predicting a new dataset. To fix this problem, instead of using just one tree, we have decided to have 501 different trees (can use any number of trees, although more than 500 is encouraged for best results). Each of these trees will be made using a random sample of our data, which means that each of our trees will be looking at a different set of data (although some batted balls may overlap) to make a tree. This expansion of decision trees is called bagging. When one wants to know the prediction of a future batted ball, the batted ball will be run through each of the trees, and receive a prediction of whether it was a hit or not from each tree. The majority decision from the trees will then be applied to be the official prediction of bagging for that batted ball (hence, why we use an odd number, in case there is a tie).

One issue that may stem from bagging is if there is one predictor with a classification with a very low initial Gini index, which would cause all of the decision trees to have the same predictor and classification as the first split. Therefore, we will restrict each of our trees to only look at m number of predictors at each split. That is to say, at each split in a decision tree, the algorithm will randomly select m predictors that can be used at the split. The algorithm will then find the lowest Gini index for a given classification of these m predictors. By applying this value m, we have created a random forest. Cross-validation will be used to find this ideal m value, and a discussion on cross-validation with our random forest models will happen in the cross-validation section.

There is one problem when we try to compare random forests to other statistical learning methods. Both the logistic regression and generalized additive models give an output of the probability of a given batted ball resulting in a hit, while the random forest produces a simple “hit” or “not a hit” prediction for each of our batted balls. Therefore, we find that the proportion of trees that predict a hit for a given batted ball is the probability of that ball being a hit. For example, if a batted ball has 204 trees that predict a hit, and 297 that predict not a hit, that batted ball has a probability of .407 (204/501) of getting a hit.

Models:

For ease of model comparison, we will be comparing five preset models in our analysis. Each of these models will have a different combination and number of possible explanatory variables, and will each be applied to each of our methods in order to create a "final model" for each of our statistical learning methods mentioned above. It is important to note that all of our models contain launch angle and exit velocity, as it is not our goal to replace Major League Baseball's current hit probability calculation since launch angle and exit velocity have proven to be great predictors for hit probability. Instead, we want to find if any further factors lead to a model that performs better. Model 5 represents the Major League Baseball equivalent, as launch angle and exit velocity are the only two predictors.
Cross-validation will also be used to find each method's best model, with more specifics discussed in the cross-validation section. The five models and their predictors are listed below in table 2.

Table 2: Model Parameters for Our 5 Models

Model Number	**Variable 1**	**Variable 2**	**Variable 3**	**Variable 4**	**Variable 5**	**Variable 6**	**Variable 7**	**Variable 8**	**Variable 9**	**Variable 10**
Model 1	Launch Angle	Exit Velocity	Distance	Hand Batter	Hand Pitcher	OF Strategy	IF Strategy	Home Team	Fielding Team	Losing
Model 2	Launch Angle	Exit Velocity	Distance	Hand Batter	Hand Pitcher	OF Strategy	IF Strategy			
Model 3	Launch Angle	Exit Velocity	Distance	Hand Batter				Home Team	Fielding Team	
Model 4	Launch Angle	Exit Velocity	Distance							
Model 5	Launch Angle	Exit Velocity								

How Will The Models Be Assessed?

In the previous section, there was a mention of whether we could create a model that "performs better"; however, it is unclear how we will assess model performance. Therefore, we will establish that the assessment of model performance will be derived from the Receiver Operating Characteristic (ROC) curve. In order to fully understand ROC curves, it is important to first discuss thresholds. Essentially, a threshold is the cutoff point of a classification. For example, if there is a threshold of 0.4 in one of our models, any batted ball with a hit probability of greater than 0.4 will be classified as a "hit". At each possible threshold value (which is any real number between 0 and 1), a true positive rate (the rate at which the model correctly predicted a hit) and a false positive rate (the rate at which the model predicted a

hit when the batted ball was actually not a hit) will be calculated, Each of these calculated points will then be graphed, and a curve will be applied on our points, giving us the ROC curve for our model. Once the ROC curve is created, we receive a value that is called the Area Under the Curve (AUC) that essentially tells us the performance of the model. Typically, an AUC from 0.6-0.7 is considered a model that is poor at predicting, an AUC from 0.7-0.8 is considered acceptable, and an AUC from 0.8-0.9 is considered a model that is excellent at predicting. Any AUC value above 0.9 is considered a model that is outstanding at predicting[26]. From there, an ideal threshold value can be assessed as the threshold that produces the highest true positive rate and the lowest false positive rate. Although the official expected batting average model in Major League Baseball only gives a number between 0 and 1 instead of a prediction, having this threshold value allows for our models to have more flexibility in terms of application. From our analysis, not only can we compare our final model to the official Major League Baseball model, but we can also compare our final model to other models that give plain "hit" or "not a hit" predictions.

Cross-Validation

In the sections above, an ideal value for λ (GAMs) and m (random forests) are said to be found using cross-validation. Furthermore, it was also stated that cross-validation would be the strategy to find each method's best model. Although there are many various methods of cross-validation, we will apply k-fold cross-validation [27]. Typically, five or ten folds are recommended; therefore, in our analysis 5-fold cross-validation was used, which means that our dataset will be split into 5 equally sized "folds". Four of the five folds will be treated as the training data, which means that the model will be applied to these folds. Since the model was not applied to the fifth fold, this fifth fold is essentially treated as a "test" dataset to test the accuracy of the model on an unknown dataset. This means that we have five different training datasets, with a test dataset for each of the training datasets. For logistic regression, each of the five main models listed in the models section will be applied to each of the training datasets, which creates five different AUC values for each model when each training model is applied to predict their respective test dataset. The average of these five AUC values is considered to be the AUC value for each model, with the model that produces the highest average AUC being chosen as the best logistic regression model. For the generalized additive models, there must first be cross-validation done to find the ideal λ

[26] Jayawant N. Mandrekar, "Receiver Operating Characteristic Curve in Diagnostic Test Assessment," Journal of Thoracic Oncology (Elsevier, November 20, 2015), https://tinyurl.com/4696ceb5.
[27] Gareth James et al., "Chapter 5: Resampling Methods," in *An Introduction to Statistical Learning: With Applications in R* (Boston: Springer, 2022).

for the smoothing splines applied to each of our quantitative variables: launch angle, exit velocity, and hit distance. Three models will be created, each with a smoothing spline applied to a different quantitative variable, where that variable is the only predictor to hits in the model. λ values ranging from one to eight will be tested on each of these models, creating 24 sub-models, where each quantitative variable model is divided into eight different models, with each model testing a different λ value for the smoothing spline applied to the variable. To be consistent, we will compare AUC values and choose the λ value with the best AUC value for each of our smoothing splines. We will not necessarily pick the λ value with the highest AUC value, as a minimal increase in AUC is not a justification to add more degrees of freedom to the model. Once there is an ideal λ value for each of the smoothing splines applied to our explanatory variables, a smoothing spline with its respective ideal λ value will be applied to each of the quantitative variables in the five main models above. Then, the same logistic regression 5-fold cross-validation process will be repeated with these five generalized additive models, with the model that produces the highest average AUC being chosen as the best generalized additive model. Finally, for random forests, each of the five main models will be split into sub-models, with each model containing a different value of m to represent the number of predictors a random forest will randomly select at each split in their trees. These m values will range from one to the number of predictors in a given model. Once again, AUC will be used to assess which m will be chosen for each model.

Results:

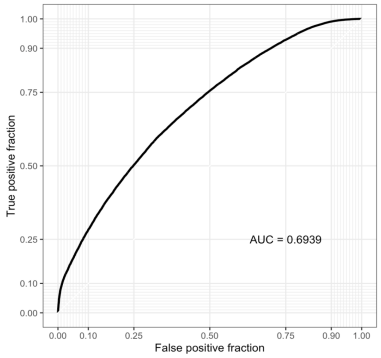
Logistic Regression:

The logistic regression to produce the highest AUC value after cross-validation was model 1, which contained the variables launch angle, exit velocity, distance, the handedness of the batter, the handedness of the pitcher, the outfield defense strategy, infield defense strategy, home team, fielding (defensive) team, and whether or not the player who hit the ball was losing or not. The model can be written in logistic regression form with our logit model:

Equation 14: Final Logit Model

$$\text{Logit}(\pi(y_i)) = \beta_0 + \beta_1\text{LaunchAngle}_i + \beta_2\text{ExitVelocity}_i + \beta_3\text{Distance}_i + \beta_4\text{HandBatter}_i + \beta_5\text{HandPitcher}_i + \beta_6\text{InfieldStrategy}_i + \beta_7\text{OutfieldStrategy}_i + \beta_8\text{HomeTeam}_i + \beta_9\text{FieldingTeam}_i + \beta_{10}\text{Losing}_i + \in_i$$

Figure 7: ROC Curve For Final Logistic Regression Model



The ROC Curve for our final logistic regression model is drawn above; this curve has an AUC value of 0.639, indicating that the logistic regression method is poor at predicting our data. Furthermore, it is found that the best threshold for this model is 0.307, which means that the model should predict hit for every batted ball with a hit probability of higher than 0.307.

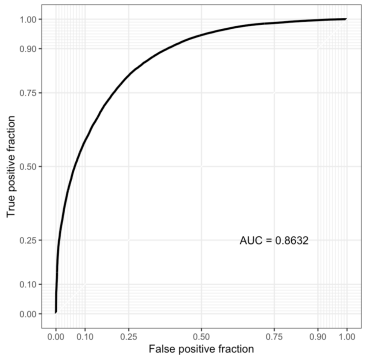
Generalized Additive Model:

Through cross-validation, it was found that a smoothing spline with a λ value of five should be applied to each of our explanatory variables. The generalized additive model to produce the highest AUC value after cross-validation was model 4, which contained launch angle, exit velocity, and distance traveled in the air. In fact, models 1, 2, 3, and 4 all produced the same AUC value, which points to the fact that the dummy variables may not add much to our generalized additive model. Therefore, we use the model with the least amount of predictors, limiting degrees of freedom to avoid any potential issues of overfitting. The model can be written in a generalized additive form as:

Equation 15: Final Generalized Additive Model

$$\text{Logit}(\pi(y_i)) = \beta_0 + f_1(\text{LaunchAngle}, 5) + f_2(\text{ExitVelocity}, 5) + f_3(\text{Distance}, 5) + \in_i$$

Figure 8: ROC Curve For Final Generalized Additive Model

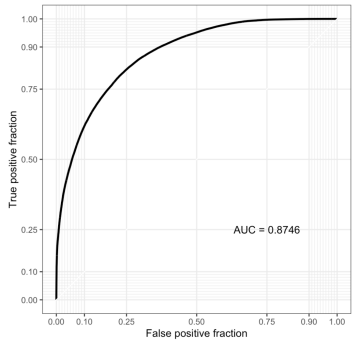


The ROC Curve for our final generalized additive model is drawn above; this curve has an AUC value of 0.8632, indicating that the generalized additive model method is good at predicting our data. It is found that the best threshold for this model is 0.342, which means that the model should predict a hit for every batted ball with a hit probability of higher than 0.342.

Random Forest Model:

The random forest model to produce the highest AUC value after cross-validation was model 4 just like the generalized additive models. For recall, this model contained launch angle, exit velocity, and distance traveled in the air.

Figure 9: ROC Curve For Final Random Forest Model:



The ROC Curve for our final random forest model is drawn above; this curve has an AUC value of 0.8746, indicating that the random forest model method is good at predicting our data. The best threshold for this random forest model ended up being 0.322, which means that the model should predict hit for every batted ball with a hit probability of higher than 0.322. Although our final random forest model had three total predictors, it was found through cross-validation that the best performing model had an m value of 2, which means that in each split in our trees in our random forest, two of the three predictors will randomly be selected for possible candidates to split the data.

Conclusion

From our results, we see that the model that performed best according to AUC for our data is a random forest model that contains the predictors of launch angle, exit velocity, and distance traveled in the air. In this random forest model, each split in the trees will randomly select two of the predictors to be a potential classification for a split. When looking at the Major League Baseball equivalent model of launch angle and exit velocity as the only predictors, we see that the statistical learning method that produced the highest AUC for this model is also random forests. It is important to note the caveat that this

is the best Major League Baseball equivalent model that could be made, as seasonal sprint speed is not public data. With this caveat, we see that a random forest with only launch angle and exit velocity as predictors has an AUC value of 0.8609, about 0.01 less than the AUC value of our best performing model. There are no well-documented hypothesis tests to test whether this difference is substantial; however, we see that using a random forest model using Major League Baseball's current predictors still creates a model that is seen as a good predictor of hits, given our data. If one finds that the slight increase in AUC is enough to warrant using our best predicting model, as opposed to Major League Baseball's model, it can be very valuable in trading for new players. If a Major League Baseball team is trading a player in exchange for another player, they will likely want to know the most accurate model for assessing that player's performance, giving them even a slight edge over other teams. For example, let's say that the New York Yankees and Boston Red Sox are making a trade and each of the teams determines that the expected batting average is to be the best statistic to analyze player performance. However, the New York Yankees use a random forest model with the predictors from the official Major League Baseball calculation of expected batting average, while the Boston Red Sox use our random forest model with distance traveled in the air included. We will establish that the player the New York Yankees are trading will be player A and the player the Boston Red Sox are trading will be player B. The New York Yankees do past batted ball analysis on each of these players, discovering that player A has an expected career batting average of .320, while player B has an expected career batting average of .322. Meanwhile, the Boston Red Sox also do past batted ball analysis on each of these players and discover that player A has an expected career batting average of .324, while player B has an expected career batting average of .319. Given the results of their analysis, both teams are happy to make the trade; however, if one of the models that the teams used is not the best performing, then that slight change in expected batting average could mean that they actually traded a better player in exchange for a worse player.

The only problem with the above scenario is that teams likely will not be using the expected batting average as the only statistic to assess a player's ability. Therefore, the natural next step would be to explore further into the world of expected statistics in baseball. Other expected statistics in baseball include but are not limited to: expected slugging percentage, expected weighted on-base average, and expected catch probability. One can apply all of these expected statistics to a response variable such as salary, to find a model that is best at predicting how much a player should be worth. From there, one can then find the expected statistics that are best for player assessment.