Artificial Intelligence and Pitch Calling in Baseball

Brianna Butler (bb5943@princeton.edu)
Department of Computer Science, 35 Olden Street
Princeton, NJ 08540 USA

Abstract

For this study, using available official MLB (Major League Baseball) pitching and subsequent batting data captured through Statcast, we analyze current models of baseball strategy and how these strategies have been attempted to be incorporated in many of the proposed hitting AI algorithms. We will attempt to build a very basic pitch calling neural network to explore how different factors come into play (narrowing pitches down to fastball/off-speed, location, etc.), and point out flaws/external factors that will need to be considered before building a full system.

Keywords: Nash equilibrium, neural networks; decision-making; baseball; machine learning

Introduction

In the modern era, artificial intelligence research has skyrocketed in popularity. The curiosity and excitement of the extent of artificial intelligence has spread to the realm of sports. Due to the recent adaptation of use of artificial intelligence in the sports world, there is still a vast amount of work to be done with the effective use of the massive amounts of statistics and data involved in sports. Multiple sports have employed various AI technologies/methods and begun to invest in AI research to further evolve their sport and respective teams/players. In the realm of sports, two main categories are being investigated when it comes to applied AI: (1) injury and risk assessment and (2) sporting performance prediction. According to a 2019 literature review on the subject by João Claudino and colleagues, most literature and research published on the intersection between AI and sports focuses on the latter category.

Choice of AI technologies employed for sports performance can vary by sport. For sports like basketball, handball, soccer, rugby, volleyball, and baseball, most research looks into employing artificial neural networks for sports performance prediction. Another popular AI approach commonly used for sports performance is a decision tree classifier, the choice by many studies focused on American football, Australian football, and field hockey. Other studies (not nearly as many) also utilize the markov process, support vector machine, bayesian network, k-means clustering, and many more. The team sports with the most AI application studies, as of 2019, were soccer, basketball, handball, and volleyball (Claudino et al., 2019).

This leaves America's favorite pastime lacking in the AI department. And although it has a nickname associated with the USA, baseball is an international phenomenon with large fan bases in Korea, Japan, China, the Dominican Republic, Venezuela, etc. Despite lack of published AI research relative to other sports, AI has a valuable role in the sport of baseball. In 2015, the MLB (Major League Baseball, the leading baseball organization in the United States and Canada) introduced Statcast, a tracking system that generates "detailed data pertaining to the movement of both the players and the ball using cameras and radar technology" (Morehouse, 2020). The introduction of Statcast changed the analysis of the game for coaches, players, and even viewers at home. The MLB has also discussed possibly introducing robotic umpires to erase the oh-so infuriating human factor in decision making for strikes and balls.

There has been some discussion in using AI in the realm of making active decisions in baseball strategy (a.k.a in-game decisions). Most information and studies out there describing the realm of baseball strategy and AI focus on the aspect of hitting - when to swing, not swing, bunt, optimize hitting distance, etc. (Silver & Huffman, 2020). There is a lack of consideration on the use of AI in the most important defensive aspect of the game: pitching. Similarly to have an in-game system to analyze and relay when to swing, what if there was a system that could relay to the user on what pitch to throw when, to who, in what situation, with who on base, etc. The MLB commonly runs into issues in which hitters are hitting too well against pitchers and vice-versa (Sanchez, 2021). What-if, rather than changing game dimensions and materials, the MLB were to give pitching staffs this pitch-calling AI to temporarily give them the upperhand?

Baseball Strategy

One of the drawing factors of baseball is the heavy strategic thinking involved in the playing, coaching, and managing aspect of the sport. Baseball is a heavy statistics sport - abbreviations for many measures of performance are used 24/7 in the viewership of baseball and participation in the sport. In fact, statistics is so important to the sport of baseball that baseball has its own subcategory of statistical analysis: sabermetrics. Coined in 1980 by Bill James, sabermetrics is "the search for objective knowledge about baseball" (SABR, n.d.). Sabermetrics has also been used to define new statistics that have become mainstream for the sport. Standard statistics categories for a baseball game (offensively) include: G (number of games played), AB (number of at-bats had), H (number of hits), 2B (doubles), 3B (triples), HR (home runs), BB (walks), K (strikeouts), and AVG (batting average). An example of a sabermetrics stat that has obtained mainstream familiarity is OPS, "on-base plus slugging", or the sum of a player's on-base percentage and slugging average (SABR, n.d.). There are other traditional statistics and other sabermetrics valuable to the analysis of baseball performance.

Game Theory

For our analysis, we are choosing to investigate machine learning and baseball pitching through the lens of game theory. Pitching, and baseball overall, are heavily dependent on strategy and making decisions based on the actions of the hitter/opponents. The two players in the game of baseball we are focusing on is the pitcher vs. the batter. Baseball is a team sport, but the one-on-one duel between the batter and the pitcher is the one occurring each pitch that is thrown in a 9-inning game.

Nash Equilibrium

Baseball is a zero-sum game. The decision a pitcher makes will have a direct effect on the batter and vice-versa. If a batter gets a hit, it is at the expense of the pitcher and the opposing team. This establishes a Nash equilibrium.

The Nash Equilibrium can help us optimize each pitch decision. In an ideal situation, the pitcher will throw the best sequence of pitches to opposing batters that heighten the number of strikeouts/outs a pitcher obtains each game.

The only downside to utilizing the Nash equilibrium, is that if a pitcher utilizes the same optimized strategy over and over again, the opposing team will take notice and adjust accordingly, rendering the original optimized strategy useless. This is why a mixed strategy approach is much more valuable in the game of baseball - the best pitchers make batters have to guess what pitch is coming.

Mixed Strategy Models As mentioned before, the highlight of baseball is the 1-vs-1 battle between the batter and the pitcher. Each pitcher has a choice of multiple pitches they can throw, and the batter attempts to obtain the best outcome on each pitch.

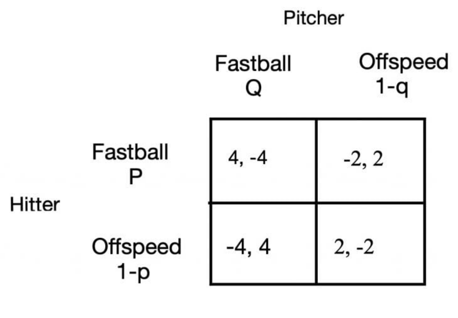
Another point to learn from the application of game theory to baseball is that, pitchers are not the only individuals applying game theory decisions in the game of baseball. Each batter will be adjusting their strategy when they come up to bat, thus, making it critical that the machine learning model that calls pitches is a dynamic system that can store and process data after each at-bat.

A common way of framing this match-up is to simplify the pitcher's pitching arsenal. Pitchers can throw a variety of pitches, a curve ball, a slider, etc. Even the pitch commonly just referred to as the "fastball" can be thrown in a variety of ways: two-seam, four-seam, cutters, etc. This way of framing the battle between batters and pitchers was proposed by Baseball Data Science's Michah Melling. Let's say the pitcher can throw either a fastball or an off-speed pitch. For our analysis, we are also placing the hitter and pitcher in a 3-2 count, a pitch count in baseball that requires to the batter to make a decision that could end the at-bat positively or negatively. The batter can swing and miss, resulting in a strikeout, swing and make contact, resulting in a hit or a field out, or the batter can decided to not swing, and if the pitch is a strike, they strike out. If the pitch is a ball and decide not to swing, then they will be granted a "walk".

In this scenario, the pitcher can decide to throw a fastball, which are much easier to control than an off-speed pitch, resulting in a better chance of a strike if the batter decides to not swing, but falling at risk to the batter being more likely to be on time should they choose to swing, resulting in a hard-contact and/or a hit. On the other hand, they can choose to throw an off-speed pitch, which can throw the batter off-balance resulting in weak contact or a swing and miss, but also running the risk of throwing a ball due to off-speed pitches being harder to control. If the batter swings at this ball, this will result in a swing and a miss or a ball that is not hit hard/well (unless the batter is Vladimir Guerrero and can hit a ball that bounces on the ground). If the batter decides to not swing, they will give the batter a free ride to first base.

If the batter were to correctly guess a thrown fastball, they would have better timing, and as a result of the fastball's higher speed than the off-speed pitch, the batted ball will be hit harder and subsequently travel farther. This is the scenario with the highest payoff, or best result, for the batter. If the pitcher were to throw a fastball while the hitter was anticipating an off-speed pitch, then 1) they would be able to locate the fastball better, resulting in a strike if the batter does not swing and 2) throw a pitch much faster than expected by the batter, resulting in a swing and miss or weak contact. Conversely, the largest negative trade-offs can be achieved when choosing the option that results in the largest payoff for the batter and the pitcher each.

This results in a mixed strategy Nash equilibrium. This is because the best strategy for each player in the game depends on the strategy the other player employs. As shown in the table, the probability that a pitcher throws a fastball is "q", and the probability they throw an off-speed pitch is "1-q". Relatively, for hitters, the probability they anticipate a fastball is "p", and the probability they anticipate an off-speed pitch is "1-p"



		Pitcher	
		Fastball Q	Offspeed 1-q
Hitter	Fastball P	4, -4	-2, 2
	Offspeed 1-p	-4, 4	2, -2

Figure 1: Table displaying the decisions a pitcher or a batter can make during a 3-2 count, their respective probabilities, and their respective payoffs.

Here is how the mixed strategy is expressed as suggested by the blog for pitchers choosing the optimal strategy:

Pitcher chooses fastball: $4q + -4(1-q) = 8q - 4$

Pitcher chooses off-speed: $-2q + 2(1-q) = -4q + 2$

$8q - 4 = -4q + 2$

$q = \frac{1}{2}$

Here is how the mixed strategy is expressed as suggested by the blog for hitters choosing the optimal strategy:

Hitter anticipates fastball: $-4p + 2(1-p) = -6p + 2$

Hitter anticipates off-speed: $4p + -2(1-p) = 6p - 2$

$-6p + 2 = 6p - 2$

$p = \frac{1}{3}$

This analysis suggests that in a 3-2 count, hitters should expect a fastball and an off-speed pitch half of the time, while pitchers should throw a majority off-speed pitches ($\frac{2}{3}$ of the time), because it results in lower trade-off should the hitter anticipate the the off-speed pitch (Melling, 2018).

Pitching, as a whole, targets minimizing damage that hitters can do, and thus allowing less runs to give their teams a better shot in the game. This is also shown by the 3-2 count game theory analysis. For whatever pitch calling AI system is made, it must have the same objective: minimize damage. In other words, it must construct a similar form of analysis whenever deciding what pitch to throw against a certain batter - human pitch callers go through the same process in-game.

Machine Learning

Like every other sport on the planet, machine learning is being utilized to further push the limits of the sport of baseball. We have noticed two main categories of artificial intelligence technology are used in the sport of baseball: training technology and prediction technology. For obvious purposes, we will be paying more attention to the latter category for more insight to a possible solution to our problem.

One monumental artificial intelligence technology that exists for baseball predictions and analysis is the Singlearity project by Joshua Silver and Tate Huffman. The Singlearity project focuses on predicting batter-pitcher match-up outcomes. The massive project consisted of a neural network trained on Statcast data from nearly 1.8 million plate appearances across a ten year span in the MLB. For their prediction of run production, they coupled Markov Chains with Singlearity-PA (the official name of the Singlearity neural network technology). Their ability to predict precise outcomes is valued, for our goal is also predict precise outcomes during an at-bat. However, our wanted outcome, in a sense, can be argued that it is more complicated and precise than Singlearity's - they predict at-bats, but we want to know which specific pitch in what specific pitch number of an at bat will lead to a wanted outcome (Silver & Huffman, 2020).

Another area where machine learning is being used heavily is the prediction of the next pitch coming from a pitcher. Although neural networks dominate in the world of machine learning and baseball, the most commonly used models for pitch prediction are Support Vector Machines, Classification Trees, and Linear Discriminant Analysis with Classification Trees being shown as the superior method for the multi-class classification required when predicting pitches (Koseler & Stephan, 2018).

Another interesting piece of work we looked into for our investigation was an interesting analysis relative to our topic that was published in Japan. The goal of this analysis was to apply a probabilistic top model to find common sequences against certain categories of hitters with similar tendencies (for example, a pull-side power hitter). Normally, probabilistic top models are applied in natural language processing, but the researchers found their methods to be relatively successful at extracting common pitch sequences for certain types of hitters, effective pitch sequences to strike out hitters, and pitch sequence trends for a specific player against a specific player (Yoshihara & Takahashi, n.d.).

Method

To help supplement our investigation into the use of neural networks in pitching calling for AI, we conducted a small experiment/trial of a neural network trained using relative pitching and hitting data from Statcast. We chose a neural network implementation because neural networks are the most commonly used artificial intelligence model for the sport of baseball (Claudino et al., 2019).

To build the neural network, we proceeded with the same steps as a deep learning neural network tutorial available on Medium, which employs the Tensorflow, Theano, and Keras libraries in Spyder by Anaconda (Mandot, 2017).

Data

The data we used in this experiment was scraped from MLB games from 2015-2018, and compiled into a Kaggle folder. Because it is official MLB data, there were many categories that needed to be cleaned up for the purpose of the neural network that we were looking for: what pitch should I throw when to get the best chance of getting the batter out. Because of this, many aspects of the data had to be removed - we wanted to only include categories that could be directly controlled by the pitcher and categories relating to the current situation the pitcher and batter are in. These are the same categories considered by a pitching coach or a catcher when calling pitches in a game. These were the resulting categories:

- Current count: When calling pitches, the number of balls and strikes can heavily affect the choice of pitch to throw - can the pitcher afford to throw a ball out of the zone in hopes of getting the batter to swing and miss, are they ahead, etc.

- Pitch number of the at bat: Whether a pitch about to be thrown is the first pitch or the fifth pitch of the at bat can be crucial to the choice of pitch thrown. A pitch can be thrown purposely "fat" (down the middle) on the first pitch to immediately get a strike on the batter, etc. This heavily relates to "current count".

- Type of pitch: Should the pitcher throw a fastball, curve, change-up, etc. This information was encoded using the Scikit-learn libray's OneHotEncoder.
- Runners on base: The runners currently on base can affect what kind of pitch is thrown - some runners like to steal when a pitcher throws one of their slower pitches, a pitcher can be more aggressive or more passive towards certain hitters with people on base to minimize damage.
- Outs: The number of outs can affect how aggressive a pitcher is towards a battery. With two outs, pitchers can focus on getting the current batter out in order to get out of the inning.
- Inning Number: Earlier on in the game, pitchers and pitch callers can take more risks with pitches that could result in well hit balls, because there will be plenty of time for their offense to score runs. On the other side, later in the game, a pitcher may be more careful to not allow any home-runs, base runners, etc.
- Pitcher and Batter handedness: The handedness of a pitcher and batter is key in pitch calling. This information was encoded using the Scikit-learn libray's OneHotEncoder.
- Location: This category is not completely controllable, but it is key to pitch calling. Pitchers will attempt to throw their pitches to a spot specified by the pitch caller. No pitcher is 100% accurate in hitting their location on every pitch, but the data is based on pitches that were already thrown and where they ended up position-wise. So, the neural network will be based on the location of a pitch that resulted in the desired effect.

Certain statistics commonly used for baseball analysis that we wanted to exclude in our analysis was ERA, batting average... basically any stat that did not provide any information to the specifications of an at bat was deemed useless for our task. A pitcher/catcher/pitch caller can directly choose which pitch to throw, and they'll have access to the memory of what the hitter they're facing did in their previous at bats.

The only downside to using this data is that pitch type is assumed from data collected through Statcast. One pitcher's splitter may read, statistically like a slider in speed, movement, and spin. All of these pitches have been assumed when it comes to their categorical label of pitch type.

Another downside to cleaning the data this way is that, for the sake of ease and time, pitcher and batter labels were removed as well. This is because the time to encode each would have contribute to a much longer time spent on cleaning data. Also, another downside to this, is that the neural network would not have the specific functions wanted in our proposed artificial intelligence pitch caller. We wanted pitch suggestions to be batter and pitcher specific, but with the way the data was handled, the neural network is more-so looking for the best pitches to call in specific situations and at-bats of a game.

Results

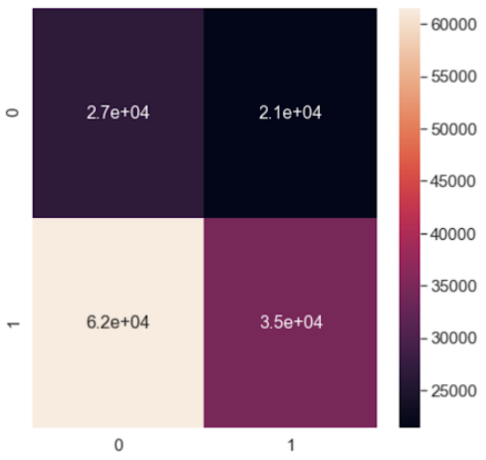


Figure 2: The resulting confusion matrix from our trial.

Our resulting neural network model was shown to not be very accurate at all. We tested the accuracy by choosing a random subset of the original pitch data and comparing it to the predictions from the classifier.

The overall accuracy of the model was about 42.5%. We compared our neural network's output to a subset of the original data (the "true" output). This is consistent, however, with findings relating to the process of baseball and pitch calling. Out of all the found papers and studies we have investigated, none have been able to create an accurate neural network model that can help call pitches in baseball. Our findings were the same.

This phenomenon could be attributed to the choice of AI model. Other events and activities and baseball can be accurately modeled and predicted using neural networks, but the very nature of pitch calling could be ill-fitted for(simpler) neural network models.

Method Problems

One downside to this method of building a neural network is the fact that the accuracy testing of the model was done using a subset of the original pitch data. There is no guarantee that the original pitches from the data set were the optimal pitches to call during the time they were thrown. So, testing our model against this subset was not 100% reliable, because we do not have a training subset that contains only the most optimal pitches. However, our results could still be interpreted as our neural network performed relatively poorly compared to original-human pitch callers in the MLB.

Another downside (and benefit) to our model was choosing the most important relative statistics for building our neural network. This points to an important design consideration when building the model necessary for pitch-calling - the inclusion of unnecessary categories in the data can sully the output of the neural network.

Alternative Direction - Recommendation System

Based off of our ”Method Problems” section and analysis, there seems to be a possibility that, one day, we can build a neural network whose output is the most optimal pitch to throw during a specific at-bat.

However, I believe there is a faster and relatively accurate way to bring a pitch-calling AI to the market/MLB: a recommendation system. Recommendation systems are machine learning systems that are commonly used in consumer-product relations, but I foresee value in applying the models of these recommendation systems to baseball and pitch calling. Like the Yoshihara study, there is value in exploring machine learning models that are commonly used in seemingly unrelated tasks.

For the proposed recommendation system, we are heavily basing it on the machine learning item recommendation system proposed by Paul Bertens and colleagues in 2018. This original item recommendation system was constructed for video games, but we foresee a benefit in reframing the system for baseball and pitch calling. One aspect of this pre-proposed model that was appealing to us was the fact that their system had to account for long-time gamers with years-worth of data, like a baseball player will play nearly 200 games in a single season and play multiple seasons, resulting in a very large training dataset for the system.

Another aspect of the proposed recommendation system that caused interest was the conversion of the time-series to a single vector for use in the DNN involved in their model. Similar to how a gamer will slightly adjust behaviors over the course of their lifetime, baseball players can make constant decision and physical changes over the course of their career.

In their model, they had to create training sub-samples for each player with tons of playing activity and multiple purchases (Bertens, Guitart, Chen, & Perianez, 2018). For our model, batters and pitchers have a variety of actions they can conduct throughout a single game. Hitters can strike out, ground out, fly out, etc., and, while pitchers love to get strikeouts, sometimes the most optimal play to reduce pitch count and increase pitcher longevity can be the ground out (especially in the case of double plays) or fly out.

This will result in multiple labels or prediction targets for hitters and pitchers. Earlier, we discussed the handling of the time series, and this is where it would come into play. We agree with their proposed solution: we could take a subsample of a hitter's at-bat performances up until time t (say a specific at-bat against a certain pitcher) and then find out the result and statistics of their at-bat. This could result in multiple sub-samples for each player to improve the training data set and reduce over-fitting.

Creating the Model Had we had more time to conduct our study, we would have carried out this proposed system ourselves. However, because of relative experience building coding machine learning models on our own requiring a steep learning curve, we are simply proposing this possible system without the chance of testing it out.

- Output: For each hitter, we could generate the probability that they will have a out-resulting outcome on a certain pitch from a certain pitcher. We could train the model over all hitters in the MLB. Once the model is done, it can begin to recommend which pitch to throw when to certain batters during a game.
- Input: This part is heavily influenced by the Bertens system. We could take the time-series data obtained for each hitter and proceed with the aforementioned conversion of them into single vectors. These vectors will all be contained in a mini-batch, and groups of mini-batches are generated per epoch. The model can then be trained over these batches (Bertens et al., 2018).

In their study, they compared the performances of a DNN versus an ERT using this framework with the ERT resulting in slightly better results (Bertens et al., 2018). The ERT also scaled up more easily. In our opinion, in a future study, both models should be explored for the pitch calling system, but the aspect of the ERT scaling better is intriguing due to the amount of data that can be collected for a single player in the MLB (this data could also include their data from minor league games).

One aspect the Bertens paper does not mention is the aspect of new players. Players are ”welcomed to the show” or invited to come play in the major league for the first time quite often. In this case, the system should call upon similar hitters (many MLB scouts compare hitters to past players for analysis purposes) to recommend what kind of pitches to throw them until they have in-game data to add to the pitch calling system.

Discussion

Once there is a model found that can accurately complete the task of pitch calling, the sport of baseball, and even the act of watching baseball, can change forever. There are many benefits to employing artificial intelligence to advance pitch calling.

As mentioned before, there has been some success with predicting the next pitching using an artificial neural network, but there is a key commonality amongst all of these (relatively) successful models: they predict the throwing of a fastball vs. an off-speed pitch. These neural networks only have to predict one of two different categorical values. For the model we attempted to build, we incorporated every pitch type record under Statcast, which includes an arsenal of at least 13 different pitches.

There is a chance that the key to the perfect machine learning model for this task could be hiding amongst a model that seems like it would not ”fit” the task. A great example of this was the Yoshihara study mentioned earlier. With the use of a model primarily used for text analysis in natural language processing, they were able to accurately find trends in pitching sequences for certain pitchers and situations (Yoshihara & Takahashi, n.d.). If we had more time, we would have applied different models to our data to see which best fulfilled the task of recommending us another pitch based on previous data and inputs.

Other successful predictive neural network models in baseball predict plate appearance outcomes - a larger scale prediction than a singular pitch (Silver & Huffman, 2020). Hitters see five pitches on average per at bat (Abbatine, 2019).

Before the modern era, most managers and coaches in baseball relied on gut feeling and chance to make decisions that could impact the course of the game. Every Major League Baseball team utilizes AI technology to run performance analysis on swings and individual pitches for training purposes. Most D1 teams have migrated to technologically supplemented training as well, resulting in faster, stronger, and more reactive players than ever before.

Baseball game management itself (a.k.a the act of coaching) is becoming increasingly more technological - very few decisions today are made without some form of technological analysis/tool helping guide managers and coaches towards victory. Pitch calling is one of the only tasks in baseball that there is not a commercial product on the market that utilizes AI to make the user (pitch-caller) more successful. As mentioned before, there are plenty of apps and devices that can help hitters learn the best decisions to make when hitting against a certain pitcher, but not vice-versa.

Another area of baseball that can benefit from pitch-calling AI is sports betting. Sports betting can be a controversial topic, but it is a highly lucrative part of baseball and helps with engagement with the sport. With pitch-calling AI, sports betters can have access to the tech, and utilize it with their own data to inform their decisions when placing bets.

Conclusion

Overall, this study ended up becoming an analysis of what components would need to go into a system that can call pitches using artificial intelligence. Although the neural network ended up not being successful, the process of creating the neural network provided us more insight into how complicated baseball statistics can be, and how the vastness of the statistics can complicate the data cleaning process. Also, despite not having the tools or time to build the recommendation system, the framework we came up is still worth exploring further. Overall, using artificial intelligence to call pitches in baseball (and subsequently, softball) is a heavily under-explored field that could be valuable to expand.

References

Abbatine, T. (2019). *The anatomy of a great at-bat.* Retrieved from https://www.baseballamerica.com/stories/the-anatomy-of-a-great-at-bat/

Bertens, P., Guitart, A., Chen, P. P., & Perianez, A. (2018). A machine-learning item recommendation system for video games. *2018 IEEE Conference on Computational Intelligence and Games (CIG).*

Claudino, J. G., de Oliveira Capanema, D., de Souza, T. V., Serrão, J. C., Pereira, A. C. M., & Nassis, G. P. (2019). Current approaches to the use of artificial intelligence for injury risk assessment and performance prediction in team sports: a systematic review. *Sports Medicine - Open*, *5*.

Flanagan, T. (1998). Game theory and professional baseball: Mixed-strategy models. *Journal of Sports Behavior*, *21*.

Koseler, K., & Stephan, M. (2018). A survey of baseball machine learning: A technical report. *Miami University: Computer Science and Software Engineering Technical Reports.*

Mandot, P. (2017). *Build your first deep learning neural network model using keras in python.* Retrieved from https://medium.com/@pushkarmandot/build-your-first-deep-learning-neural-network-model-using-keras-in-python-a90b5864116d

Melling, M. (2018). *Game theory applications in baseball.* Retrieved from https://www.baseballdatascience.com/game-theory-applications-in-baseball/

Morehouse, J. (2020). *Machine learning implementations in baseball: An algorithmic prediction of the next pitch* (Masters dissertation). Mathematics and Statistics, University of New Brunswick, Fredericton, Brunswick.

SABR. (n.d.). *A guide to sabermetric research.* Retrieved from https://sabr.org/sabermetrics

Sanchez, S. (2021). *There are still a lot of home runs.* Retrieved from https://blogs.fangraphs.com/there-are-still-a-lot-of-home-runs/

Schale, P. (2020). *Mlb pitch data 2015-2018.* Retrieved from https://www.kaggle.com/pschale/mlb-pitch-data-20152018

Silver, J., & Huffman, T. (2020). Baseball predictions and strategies using explainable ai. *MIT Sloan: Sports Analytic Conference*, *15*.

Yoshihara, K., & Takahashi, K. (n.d.). Pitch sequences in baseball: Analysis using a probabilistic topic model.