**Enhancing Beat the Streak Odds through Machine Learning: A Tensorflow Neural Network Approach**

Jason Axelberg and Joshua Koskie

*Abstract–*This paper explores the application of machine learning, specifically a TensorFlow neural network approach, to enhance the odds of success in the "Beat the Streak" game created by Major League Baseball. The objective of the game is to beat the longest hitting streak in MLB history, set by Joe DiMaggio at 56 games. The declining popularity of baseball, especially among younger generations, prompted the creation of this interactive game. The paper leverages extensive baseball data, primarily sourced from pybaseball, to develop a predictive model based on batter performance metrics. The study focuses on key variables such as Hits per Game (HPG), At Bats (ABS), Hits, Walks, and Strikeouts to improve the accuracy of predicting a player's likelihood of getting a hit in a given game. The paper also discusses the limitations of the current model and proposes potential enhancements, including the incorporation of ballpark factors, weather conditions, and matchup-based data.

## Introduction

The game of baseball poses a unique challenge for fans participating in "Beat the Streak," where the objective is to predict a player's ability to achieve a hit in consecutive games and surpass the legendary 56-game hitting streak set by Joe DiMaggio. This paper delves into the complexities of this predictive task, leveraging machine learning techniques to enhance accuracy. Drawing on sabermetrics, the study utilizes a dataset sourced from various platforms, including Baseball Reference, Baseball Savant, and FanGraphs, using the pybaseball package. We decided to focus on the batter’s performance metrics, and the model is developed using a TensorFlow neural network with specific attention to feature scaling and architecture. The results are analyzed in comparison to existing implementations, revealing both successes and areas for improvement. The paper concludes with reflections on the limitations of the current model and proposes avenues for future enhancements.

## Background

We are assuming basic knowledge of how the game of baseball is played. A batter, when it’s his turn in the batting order goes up to the plate and has an at bat against the pitcher. The outcomes of the at bat are either he gets out or he doesn’t. A batter can get a hit by putting the baseball in play and reaching base, this is the stat focused on in this problem. Getting a hit in baseball is considered one of the hardest tasks in professional sports. There are countless factors that play into this, which could be an entire writeup in itself.

What is Beat The Streak?  
 Beat The Streak is a game created by major league baseball to increase their interaction among their fanbase, especially among the younger generation(whom the game is getting less popular among). The goal of the game is to beat the longest hitting streak in MLB history which is 56 games. This streak was set by Joe Dimaggio in 1941. This streak is pretty legendary and no one has even gotten close to breaking it since. The next highest streak since then was Pete Rose’s streak of 44 games in 1978 and no one else has gotten closer than that since. In fact every year it is getting harder and harder because batting average has been declining over the last 20 years.

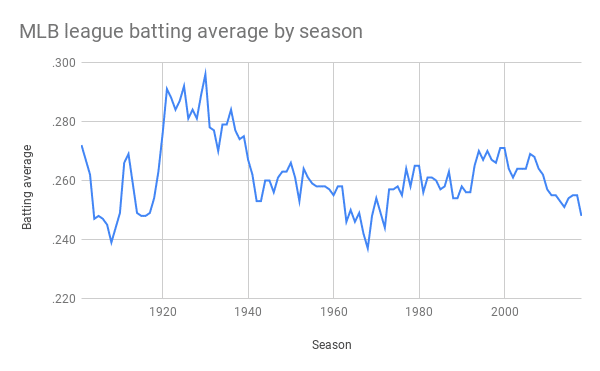


Figure 1

The way the game works is each day you have a chance to pick a hitter to get a hit. If the hitter you picked gets a hit your streak increases one and the goal is to this 57 times in a row. Each day you also have the option to double down, or not pick a hitter to get a hit. So if you don’t like any of the matchups or hitters odds that day you can choose to not pick someone.

What are your odds of beating the streak?

If you intend to beat the streak using the app’s recommendations the odds are not in your favor. The highest hit percentage they give you on average is 74% doing the math if you have 74% chance every day for 57 days is (0.74)^57 which is 3.5e-8 (0.0000000035) or 1/28,000,000. Your odds are better picking Luis Arraez (The best average hitter in MLB last year) every game who got a hit in 79.9% of his games giving you 1/350,000 odds.

Sports analytics

Baseball is the sport that uses analytics the most, the study of this is called sabermetrics. There are metrics on everything you can think of measured by trackers at every mlb stadium called a trackman. These systems track every measurable you can think of that happens at each pitch of an MLB game. Some examples of this are the release point of the pitch with its xyz axis, the same thing where it crosses the plate, spin axis, spin rate, launch angle, exit velocity… I think you get the point. Teams use all this data to make decisions to see who the best matchup would be (which hitter/lineup should we use against this pitcher, and vice versa) This data is also used by many teams to calculate how much teams pay certain players. The Sabermetrics ppl created a statistic called WAR which means wins above replacement calculated using the equation, with replacement being the average MLB player for the specific position.

WAR = (Batting Runs + Base Running Runs +Fielding Runs + Positional Adjustment + League Adjustment +Replacement Runs) / (Runs Per Win)

The higher the score the more the player helps you win, so the players with higher WAR scores get paid more. You can see the pretty linear connection between these two variables in figure 1 below. As predicted WAR increases the players salary does as well.

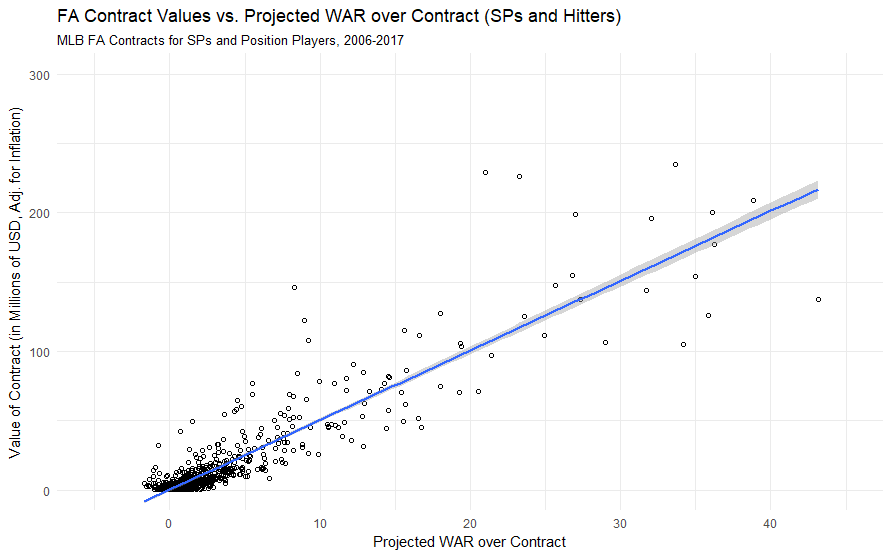


Figure 2

## Dataset

We got our data using a python package called pybaseball which does all the web scraping for us. This package pulls data from three different sites Baseball Reference, Baseball Savant, and FanGraphs. For this project we used the statcast data which pulls basically every possible thing you can record at a baseball game. We pulled this data yearly and each dataset contained 91 rows and depending on the season usually contains 700,000 rows for each pitch thrown that year(except for 2020 because the season was shortened due to COVID). We used these 4 lines of code changing the dates to get the pitches from each season.

from pybaseball import statcast

import pandas as pd

df = statcast(start\_dt ="2023-03-30", end\_dt = "2023-11-01")

df.to\_csv('pbp\_data\_2023.csv', index = False)

Now that we had the data we needed to create parameters that we could feed into our model. First we had to create significant variables we could use as parameters. Some of the key variables found in other models include: batter’s performance, batter’s team performance, opponent starting pitcher’s performance, opponent bullpen’s performance, weather conditions, ballpark factors With the little amount of time we had to get this project done we decided to focus on variables based on the batter’s performance. These variables included

**HPG** - which was a statistic we created that measured on average how many games that player gets a hit in. It is calculated by (games with a hit/games played)

**ABS** - also known as at bats is every time a player does up to the plate and does not walk or have a sacrifice hit. This variable was calculated for the last 10,5, and 3 games to see any trend in how many at bats the player is getting. Our logic was the more at bats a player is getting, the greater chance of them getting a hit.

**Hits** - this was also counted for the last 10,5 and 3 games to see how the player has been doing and see if that affected the players chance to get a hit

**Walks** - this is counted the same as hits with 10, 5, and 3 games. Our thought was the more a player walks the lower his chances of hits are because he will have less chances in a game if he walks a bunch.

**Strikeouts** - this was counted the same with the 10, 5, and 3 game splits and our logic for this was the same as our logic for walks.

If we had more time there are alot of additional variables we would’ve liked to add to our model’s parameters. A couple of these variables include matchup based variables like a hitter's past performance against the starting pitcher that day and against the other team's bullpen that day. The team’s past performance against the starting pitcher for that day and their bullpen, how the team is doing will affect how many times the batter we pick comes up to bat. We would also like to add how many hits per each start the starting pitcher gives up, how many hits per inning they give up, and how many hits per inning the opposing team’s bullpen gives up. Another thing we would like to add is the park factor which is created by ESPN and measures how much each ballpark favors hitters or pitchers, along with ballpark factor we would also like to add weather and altitude into the equation.These are all factors included in Pedro and Roberto’s model as seen in figure 4 that helped them reach a 78% success rate in their model and we believe would’ve greatly improved our model. One last Idea I can think of doing is training a model for each individual player(if they have enough data which might be kind of hard) instead of all the hitters together because each individual player is unique and should have different trends to their success. Ex. Some players might thrive in cold weather while others might in the warm weather and our model doesn’t take into account that each player is unique.

## Work

Here is the “correlation/importance” graph of each of our variables compared to some other implementations.

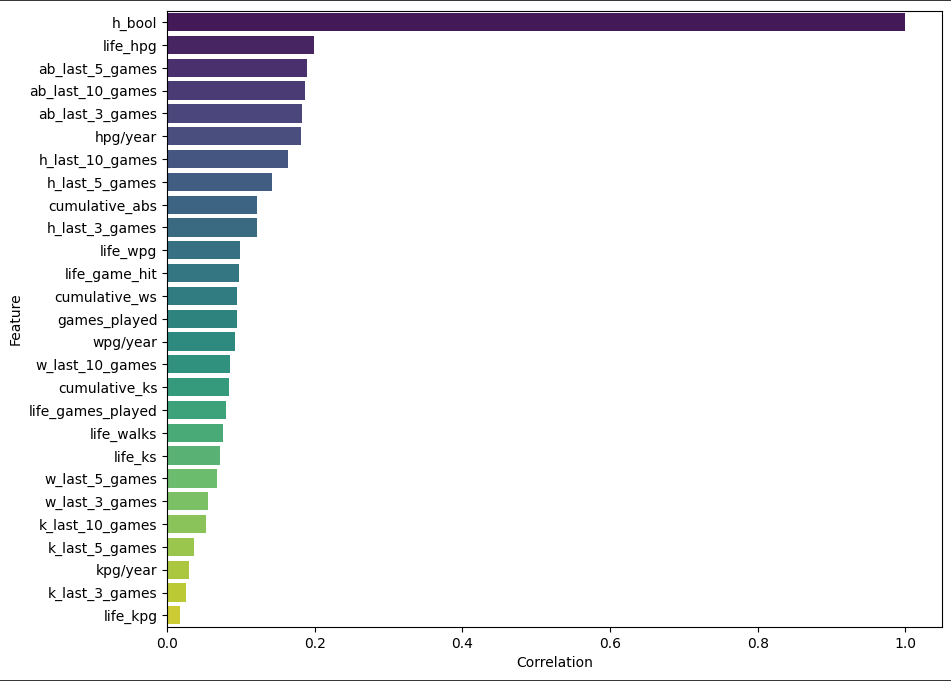


Figure 3

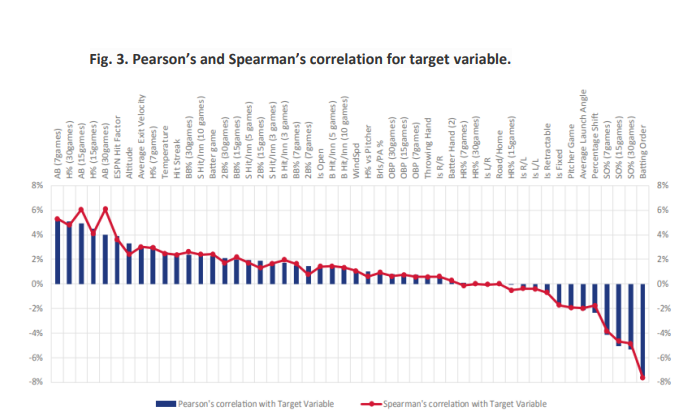
****

Figure 4

We started by splitting the data into training, validation and testing data with 80% training and 10% for validation and testing each. Next a standard scaler was applied to all of the feature data. We used a neural network with 6 hidden layers all using the “tanh” activation and a single node output layer with the “sigmoid” activation layer for binary classification.

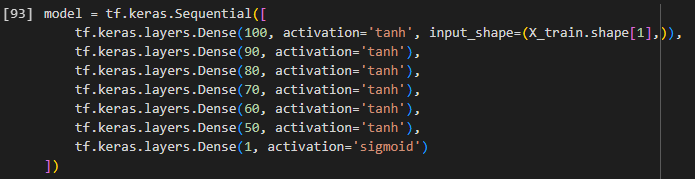


Figure 5

The model was compiled with an “adam” optimizer, “binary\_corossentropy” for the loss and “accuracy” metric for testing. The model was fit on the training data, which was run through 50 epochs with a batch size of 32.

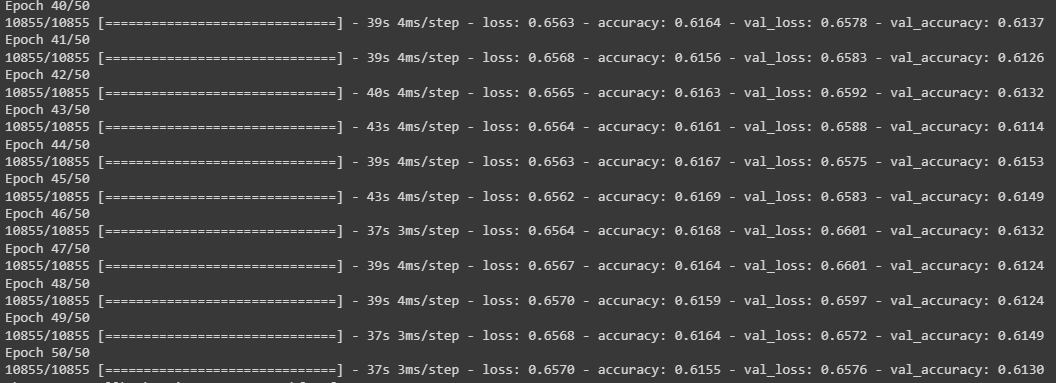


Figure 6

## Results

The model ended with a .6155 training, .6130 validation and .6207 test accuracy. The initial look at these scores didn’t look too good but then we looked at some of the other scores from other implementations and our model wasn’t too far behind. Singularity-BTS has been implementing a model on this problem for over X years and are at .793. Looking at figure X, you can see that just guessing the leadoff hitter gets a better score than our model so there is still some room to grow before this can be confidently implemented.

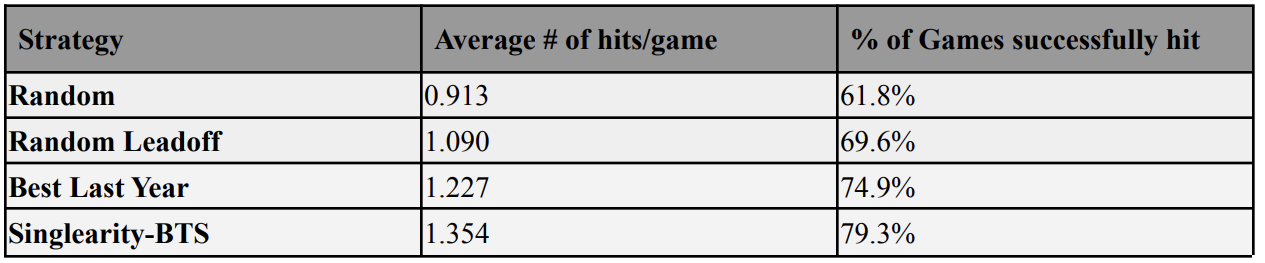


Figure 7

Looking at some actual data points in our testing set we can see that our model is working properly. In figure X, there are 3 players shown with their predicted probability to get a hit above them. Player 1 is a true positive example, he has a high probability to get a hit and you can see that he has some features to back it up including high life\_games\_played and life\_games\_hit which show that this is an experienced player unlike player 2 and 3. Player 2 is a false negative example, he showed a lot of signs that he isn’t a good player to guess to get a hit, but against the odds he got one that day. Player 2 is a great example of how difficult this model is to solve. Player 3 is a true negative example with features that show that this player does not have much experience and has low hitting stats helps the model make a low prediction score on him.

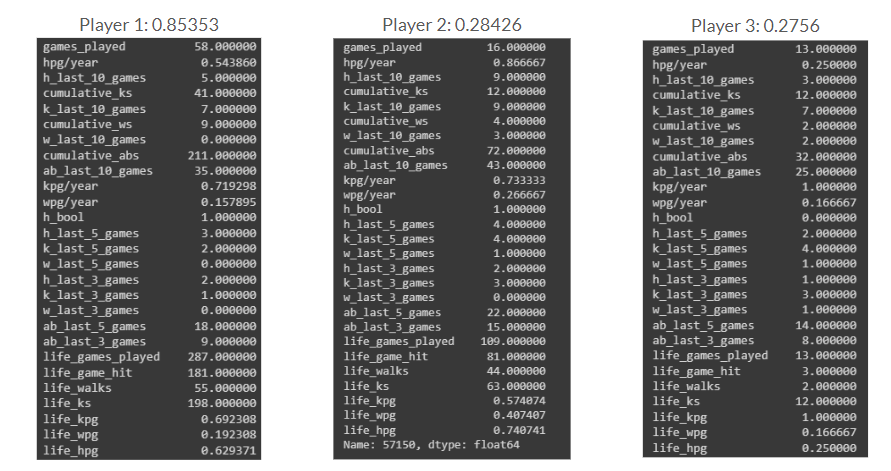


Figure 8

Utilizing SHAP, our model’s features can be visualized. Figure 8 shows the feature values and SHAP values for each feature that was fed into the model.

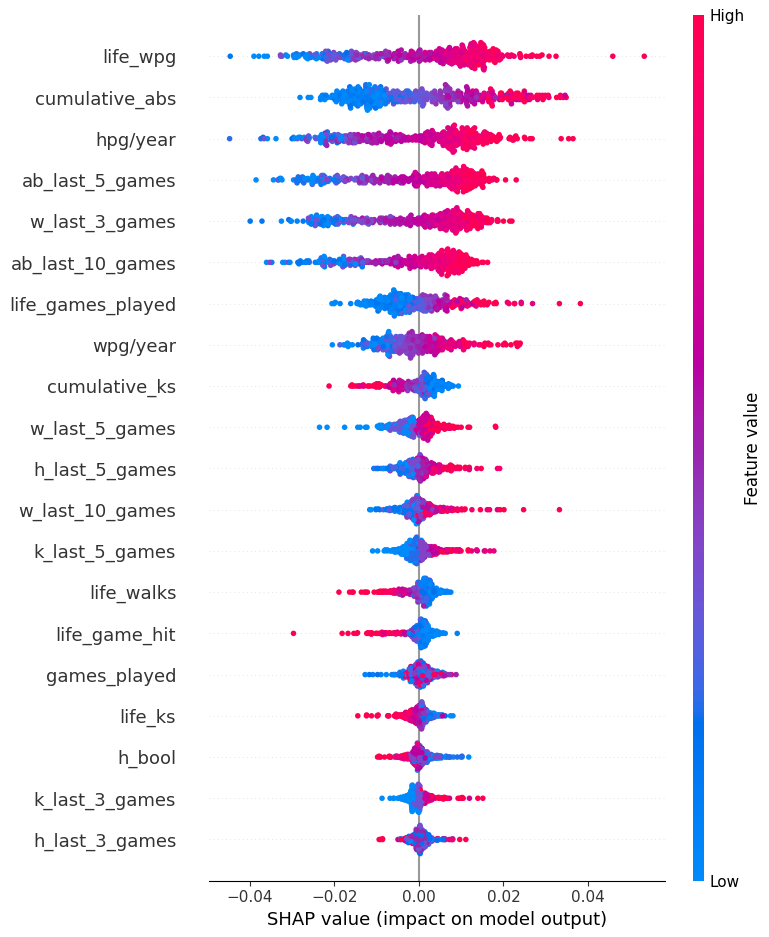


Figure 9

Figures 10 and 11 show how the model takes the average prediction score for the whole testing set as a starting point and then adds each feature's impact to the prediction to calculate the final prediction.

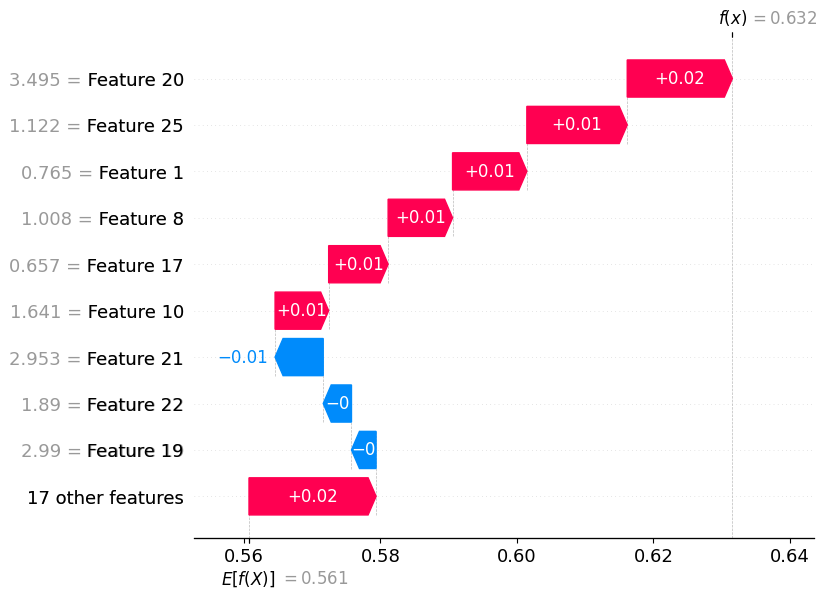


Figure 10

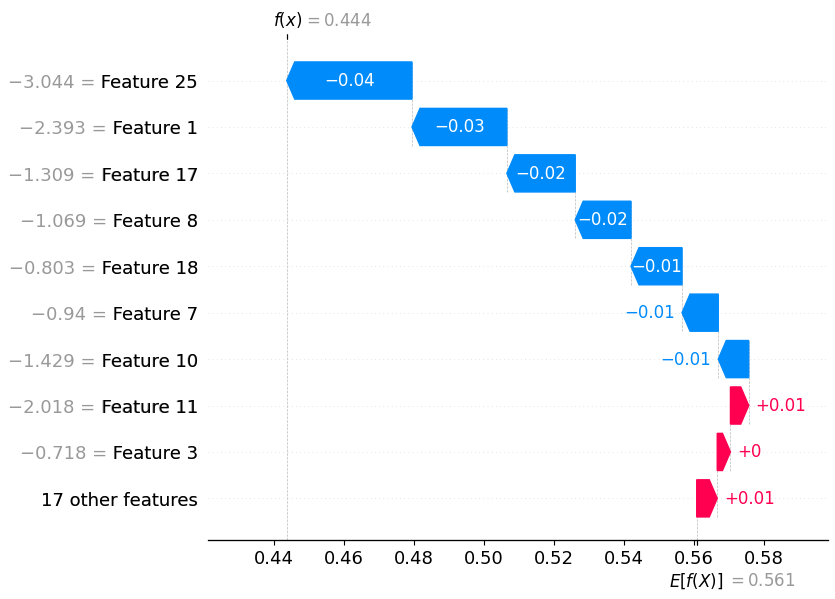


Figure 11

## Summary

The bottleneck of our implementation is the data we are using and the model that we chose. The addition of ballpark factors, weather, and matchup based data will improve the model. Incorporated time-space into our neural network would also improve the model. Doing this would replicate what we did by creating data for the past 3, 5, and 10 games but with a lot more relevance. Using time-space would allow the model to find player streaks which play a huge role in the game of baseball. This change would give our model the extra power to get over the accuracy of just guessing the leadoff batter.

## Conclusion

Predicting a player's likelihood of getting a hit in a baseball game proves to be a challenging task, as evidenced by the complexities of the "Beat the Streak" game. This study employs a TensorFlow neural network approach, emphasizing batter performance metrics, to enhance prediction accuracy. While the initial model achieves moderate success, there is acknowledgment of its limitations, particularly the absence of factors such as ballpark conditions, weather, and matchup-based data. The conclusion highlights the potential for future improvements by incorporating these additional variables and introducing a time-space element into the neural network architecture. By addressing these aspects, the model's predictive power could be significantly enhanced, paving the way for a more robust and accurate tool for baseball enthusiasts participating in "Beat the Streak."

## References

Figure 2 was pulled from a fangraphs article on the comparison between War and free agent contracts- <https://community.fangraphs.com/on-war-its-linearity-and-efficient-free-agent-contracts/>

Python Package we used to pull our data

<https://www.google.com/url?q=https://github.com/jldbc/pybaseball&sa=D&source=docs&ust=1706388706057039&usg=AOvVaw2KdLmHyRqkUCc7iwLgsIcP>

Figure 4 comes from this peer reviewed paper by Pedro Alceo and Roberto Henriques. It comes from page 22

<https://run.unl.pt/bitstream/10362/142740/1/Beat_the_Streak_Prediction_of_MLB_Base_Hits_Using_Machine_Learning.pdf>

Figure 7 we got out of the Singlearity journal by Joshua Silver

<https://www.singlearity.com/static/assets/DiMaggioBeatTheStreak.pdf>

## Additional Documents

**Final.ipynb** contains all of the code used in the machine learning aspect of this paper and **data\_manipulation.ipynb** was used in creating the year by year parameter for the model and these are both attached within this submission.

**all\_years\_data.csv** is the data that we used and is also attached