MLB Player At Bat Prediction

Using MLB Pitch Data from 2015-2018 to Determine a Player's Performance

Allston Fojas
Data Science
University of California San Diego
La Jolla, CA, USA
amfojas@ucsd.edu

Enrique Sanchez
Data Science
University of California San Diego
La Jolla, CA, USA
ens004@ucsd.edu

Stephanie Moore
Data Science
University of California San Diego
La Jolla, CA, USA
sgm002@ucsd.edu

ABSTRACT

Much like any other sport, baseball is a game of skill, luck, matchups, and many other elements. Major League Baseball (MLB) teams spend millions of dollars every year to develop or maintain the right blend of these elements. This is all done in hopes of winning many games and ultimately a World Series title.

A team's success, however, ultimately comes down to their player's performance during at bats. An at bat is defined to be a batter's turn against a pitcher. Each at bat consists of several attempts of getting on base or hitting a home run. Given that at bats are the only way to gain points in a game, it is critical to maximize the likelihood of a successful at bat.

In this report, we will be building a model that predicts whether an at bat will have success or not. We define a successful (or positive) at bat as one in which the batter either progresses to a base, scores a home run, or gets out but allows a teammate to progress (sacrifice). A failed (or negative) at bat would simply be one in which no progress is made and results in an out.

1 Data Set

We found a Kaggle data set of pitch-level data for every pitch thrown during the 2015-2018 MLB regular seasons[1]. In the pitches data set, pitches are categorized by type (fastball, slider, etc.) and includes information about the current game situation, such as the event description of the result of the at-bat, inning number, number of outs after this at-bat, score for the pitcher's team, which hand pitcher throws with (left or right-handed), which side the batter hits on (left or right), and whether or not if it is the top of the inning. Each row in the pitches data set represents a single pitch.

In addition, there is an ejections data set that contains relevant information about player ejections from the game, such as a human readable format description of the player ejection, event number for ejection, date of the ejection, whether or not the ejection was for arguing balls and strikes, whether or not the ejection was correct, the team for player ejected, and whether that team is the home team.

Another data set included in the Kaggle data set was a games data set, which included information, such as the game attendance from fans, final score for the visiting team, date of the game, length of the game, final score for the home team, start time of the game, various umpires per game, name of the stadium, descriptions of the weather and wind conditions, and the length of delay before the game. Furthermore, there was a data set that contained the player's ID and first and last name.

For the at-bats data set, there are no missing values, and each observation is unique, meaning there is exactly one at-bats ID for each row in that data set. There are 740,389 rows in that data set, 1,688 unique batters, 1,332 unique pitchers, and 9,718 unique games included in that data set. The distribution of pitchers per inning is fairly consistent at around 11.2% for the first eight of nine innings. Then, in the ninth inning, that distribution decreases to about 8.42%. An important note is that a typical game is nine innings, so anything over is considered overtime. Therefore, it makes sense that the distribution of pitchers dramatically decreases after the ninth inning to less than 1%. Another interesting insight is that in terms of pitches, about 73.2% of them are right-handed pitches and about 26.78% of them are left-handed pitches. We see a similar pattern in terms of the side the batter bats on, where about 58.4% of batters stand on the right side and about 41.5% of batters stand on the left side.

In addition, when looking at the distribution of pitches at the top and bottom of the inning, we see that about 50.9% of them occur at the top of the inning and about 49.04% of them occur at the bottom of the inning. It is important to note that only the away team bats at the top of the inning. The home team always bats at bottom of the inning. If the away team fails to take the lead after the top of the 9th inning, the home team automatically wins and will not bat at the bottom of the inning. Hence, this is why we may see more at-bats at the top of the innings. One other fascinating insight is seen from the distribution of the various event categories, in which each one essentially describes whether the player gets on base or is out. Here are a couple of bar charts that visually convey this distribution.

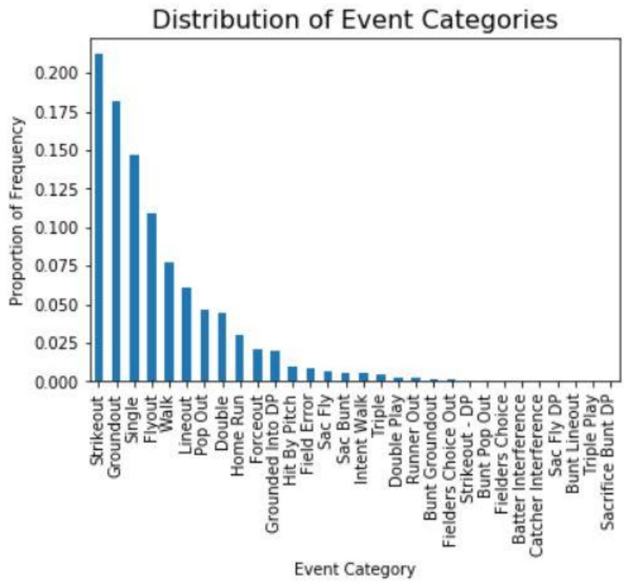

Figure 1: **Bar chart distribution of event categories**

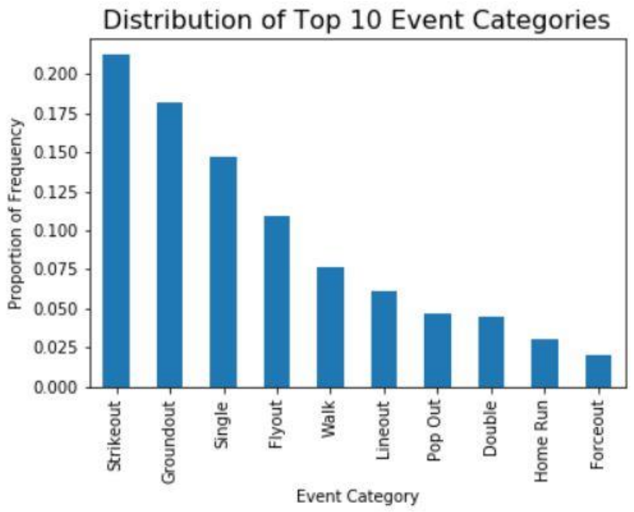

Figure 2: **Bar chart distribution of top 10 event categories**

From the above distributions, we see that strikeouts and groundouts are the two most frequent events. In addition, six of the top ten event categories are types of outs (strikeout, groundout, flyout, lineout, pop out, and forceout), and the remaining four categories are types of plays in which the player gets on a base (single, walk, double, homerun). From these observations, we see that an event in which the player is out is more often to occur than an event in which the player gets on a base. Specifically, after mapping the above batting events to either a positive at-bat (meaning the player gets on base) or a negative at-bat (meaning the player is out), we see that about 65.93% of events are negative events and about 34.06% of events are positive events.

Then, we looked at the player ejections data set to determine which team had the most players ejected from a game. Here are a couple of bar charts that visually convey the distribution of teams that had players ejected from a game.

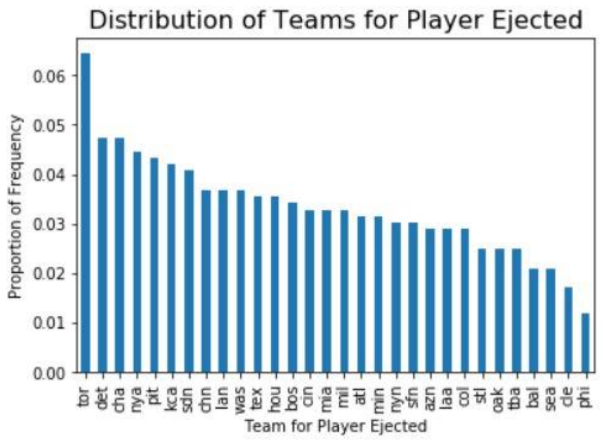

Figure 3: **Bar chart distribution of teams with players ejected**

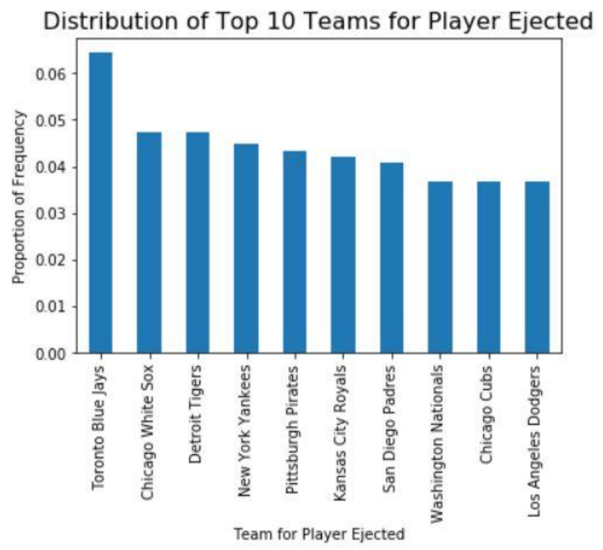

Figure 4: **Bar chart distribution of top 10 teams with players ejected, now shown with human readable team names**

From the above distribution, we see that the Toronto Blue Jays is the team with the highest proportion of players ejected from a game. Interestingly, the Toronto Blue Jays have a proportion of frequency value of over one percent greater than the second team, which is the Chicago White Sox. These two teams have the highest delta/difference in frequency values. As for the Chicago White Sox and the other eight teams in the top 10 teams, each of their respective proportion of frequency values is relatively similar. From the above distribution, we see that the distribution for these nine remaining teams relatively decreases by smaller amounts and eventually levels out.

Next, we combined the at-bats and games data sets to determine if start time had any effect on the distribution of positive and negative events. Here are a couple of bar charts that show the distribution.

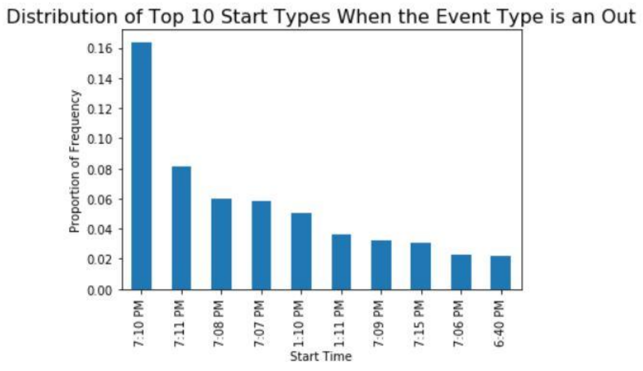

Figure 5: **Bar chart distribution of the top 10 start times when the event type is an out**

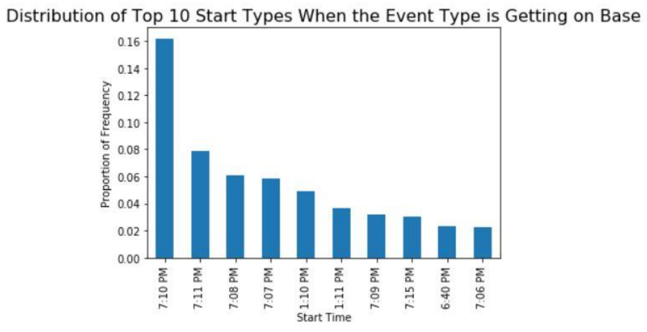

Figure 6: **Bar chart distribution of the top 10 start times when the event type is getting on base**

From the above distributions, we see that the start time has no effect on the resulting event, specifically whether the event results in the player getting out or on base. Comparing the distributions, we see that events resulting in an out occur marginally more often (16.38% for 7:10pm and 8.13% for 7:11pm) in games with the same start times, such as 7:10pm and 7:11pm, than events resulting in the player getting on a base (16.19% for 7:10pm and 7.90% for 7:11pm).

Afterwards, we looked at the distribution of events for other metrics, one of which is the venue. Here is a bar chart distribution of both negative and positive events (denoted as 0 and 1, respectively) for each venue.

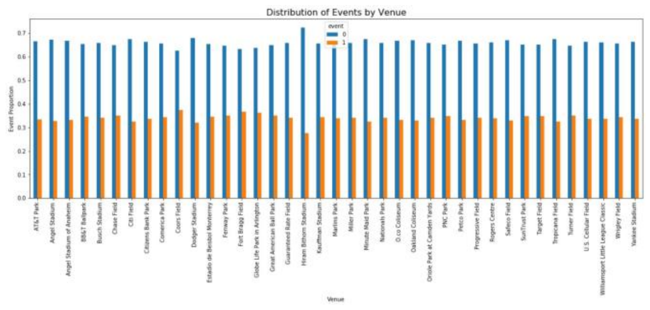

Figure 7: **Bar chart distribution of events by venue**

Clearly, we can see here that different venues result in different proportions of events. This makes sense as some venues are actually more hit or 'home-run friendly' when compared to others. Interestingly, negative events happen most when the stadium is closed (dome or roof closed). Positive events happen most when there is rain or drizzle.

Finally, we looked at if game duration had any effect on the proportion of positive events. Here is a scatterplot of the relationship between these two variables.

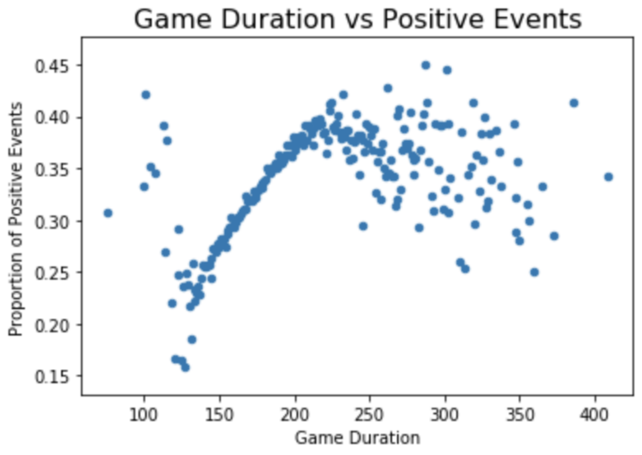

Figure 8: **Scatterplot distribution of game duration vs. positive events**

The above plot is actually very interesting. We can see that for games lasting between ~125 and ~200 minutes, the longer games have higher proportions of positive events. However, once it goes past this ~200 minute mark (possibly overtime), there is a lot more variation. In fact, a possible argument is that this relationship has a downward trend past the ~200 minute mark.

2 Predictive Task

2.1 Feature Engineering

After merging together the data sets that were previously mentioned, we had several features pertaining to the batter, pitcher, pitch information, umpires, weather conditions, and game statistics for every at bat observation. However, much of this information was messy.

For example, the year, month, and day were held inside a single ‘date’ column, the temperature and weather conditions were held within a single ‘weather’ column, and the wind speed and direction were held within a single ‘wind’ column. Each of these columns were subsequently cleaned and each contained feature was given its own respective column in the data set. This information is important as they represent the exterior conditions a batter and pitcher face during an at bat. Strong winds or rain can affect the performance of the batter or pitcher which could ultimately change the outcome.

We also had the names of each of the four umpires of a particular at bat. Given that only the home plate umpire can affect the outcome of an at bat, the first, second, and third base umpire names were removed. In reality, the first base umpire can affect the outcome but it was determined that that this effect was not significant. In addition, after doing some analysis, we found that start times and game delay durations did not affect the outcome of an at bat. Therefore, we did not keep this information in our final data set.

The data set also contained information on whether the at bat was performed at the top/bottom of an inning and the team name of the home/away team. Since we know that only away teams bat at the top of the inning, we were able to determine the team name of the batter and pitcher for a particular observation. This information was transformed into two new columns and replaced the previously mentioned information.

The pitch information provided by Kaggle details the speed, spin, location and many other attributes of a pitch for a particular at bat. However, about 30 of these attributes could not be deciphered so we decided to ignore them and keep only those that were known. Instead of working with the raw pitch data, we decided to engineer a couple features out of them and the other information in our data set. The first feature we engineered was a Strikeout Percentage. This feature represents the historical strikeout percentage of a batter. The higher this percentage, the more likely a batter is to strike out for an at bat. Given that a strikeout is the most common event during an at bat, we believed that this feature would come useful. The second feature we created was a Defensive Efficiency Ratio (DER). This feature was inspired by literature, as mentioned below, and is used to determine the defensive efficiency of the team the batter is up against. The formula for this feature is as follows:

$$1 - \frac{hits\ -home\ runs}{plate\ appearances\ -\ bases\ on\ walks\ -\ strikeouts\ -\ hits\ by\ pitches\ -\ home\ runs}$$

Naturally, a batter who meets an opposing team with a good defensive ability will have a hard time getting on base so it makes sense to include such a feature.

Given that the data set we are working with has several different possible events (strikeout, double play, single, etc.), these events were mapped to either a 0 (failure) or 1 (success). Note that this is what we are going to be attempting to predict. From this, we were now working with a binary classification problem.

As an additional and final feature, we included the average event for every batter so that we have a better idea of the proportion of times a batter got on base.

To summarize, these are the features our final data set contained: batter_id, batter_position, team, inning, post_outs, pitcher_id, pitcher_position, opponent, opponent_score, game_id, attendance, year, month, day, game_duration, temperature, weather_condition, wind_speed, wind_direction, umpire_HP venue_name, team_final_score, opp_final_score, DER, strikeout_%, av_event, and event (y).

2.2 Prediction

We will attempt to predict whether an at bat will have success (positive at bat) or result in an out (negative at bat) using the features mentioned above. Categorical features will be one-hot encoded to ensure our model is able to make such predictions.

As was noted in our analysis of the data set, 65.93% of at bats were positive (1) and the other 34.06% were negative (0). Therefore, we can infer that a trivial binary classifier (classifying every at bat as 0) will have an accuracy of approximately 66%. We will build a model that outperforms this baseline by learning on the features used above.

Although there is an imbalance in the percentage of positive and negative at bats, the imbalance is not so drastic so using accuracy to evaluate our model should be reasonable. If we were working with a multi-class classifier and there was an imbalance between the classes in the data, then this may not be a reasonable approach.

Nonetheless, we will still assess the validity of our predictions by determining the proportion of positive and negative at bats our classifier predicts. If we find that our classifier disproportionately predicts 0, we may need to alter our model so that it minimizes the number of false 0 predictions or minimizes the False Negatives.

3 Model

Since our predictive task is a binary classification problem to predict whether the batter has a "positive at bat", it makes sense to use a logistic regression model since this model is best suited for binary classification problems. As we learned in class, logistic regression aims to model probabilities, specifically p(label|data), by training a classifier that outputs a 1 if the current feature times theta is greater than some threshold, and outputs a 0 otherwise. Logistic regression converts a real-valued expression into a probability by using a sigmoid function.

There are two main ways that we optimized our model. The first was performing feature engineering to ensure we were feeding the right features into our model. If we found that a feature was redundant or added no additional information, this feature was removed. The features we ultimately worked with are the features mentioned in the previous section. The second method of optimizing our model was by modifying our model's parameters through a grid search approach. We tuned these parameters to maximize the accuracy on the validation set (20% of the data) but at the same time paid close attention to the number of false negatives, the precision, and the recall of our model.

As we previously suspected, we did run into issues with our models disproportionately predicting 0's (negative events) and having a high number of false negatives. We attempted to fix the issue by balancing the class weight of our logistic classifier but this actually resulted in too many 1's (positive events) being predicted or a high number of false positives. We ultimately settled for the model with the parameters, as determined by our grid search, that showed the best tradeoff between accuracy, precision, and recall.

In addition to using a logistic regression model, we used a linear SVC model and a random forest classifier. A linear SVC model fits to the provided data, returning a "best fit" hyperplane that divides, or categorizes, the data. In our case, a linear SVC model returns a "best fit" hyperplane that categorizes positive and negative events. In contrast, a random forest classifier is an ensemble learning method that fits several decision tree classifiers on various sub-samples of the data set and uses averaging to improve the predictive accuracy and control over-fitting.

Initially, the purpose of using a random forest classifier was to get feature importances for each feature. However, we quickly disregarded the random forest classifier for several reasons. This model was heavily overfitting our training set and underfitting our validation/test sets. We could not find our way around this issue. Also, given the vast number of features we were training on after one hot encoding our categorical features, training time was expensive and grid search would take several hours to finalize. Ultimately, this model was unable to outperform our original logistic classifier.

As for the linear SVC model, the results appeared to be promising. The models accuracy, although inconsistent in its results, did appear to outperform our logistic classifier model. However, we realized that we were again running into the issue of a high false negative rate. The issue here was actually more extreme than we had experienced with the logistic classifier. We were able to correct the issue to an extent but were unable to get it to outperform our logistic classifier. Given this issue and the inconsistency in the models performance, we disregarded this model.

4 Literature

As was previously mentioned, the data set for this project came from Kaggle[2]. It contains MLB pitch data from the 2015-2018 regular season. Pitches are categorized by type (fastball, slider, etc.) and also contain game situation information (umpire, weather conditions, time of day, inning, etc.). This data set has been used for pitch type prediction, strike zone analysis, and pitch trajectory analysis.

In the past, most research surrounding this field has been in predicting the winner of entire games. This has been done with ensemble learning (using model trees, artificial neural networks, and support vector machines) that predicts players' future performance. Chain modeling has also been used to predict scores of teams. Despite the importance of individual player performance to the outcome of the game, little analysis has been done on individual batter-pitcher matchups[5].

Some studies that have been conducted for predicting matchups outcomes in MLB use different probability models[3]. This study used data from the 2012 season that contained specific matchups: batter, pitcher, times faces, and the result of the pitch (single, triple, walks, out, etc)[4].

Another study used a Bayesian hierarchical log5 model to predict the results of batter-pitcher matchups event. They used data of regular-season baseball games in the Korea Baseball Organization (KBO) league from April 2008 to October 2017 (www.statiz.co.kr). They finalized their model based off of their findings that a general log5 model is more flexible, but the estimation of coefficients was not as accurate due to the insufficient amount of data available in practice. They also included a new variable in their model that represented the defensive ability of the pitcher's team because this was also a factor of the matchup result.

Currently, the methods employed to study this type of data take the batting statistics and transform them into a measure of "offensive production" [6]. Models have included different player statistic features like total runs, on-base percentage, strikeout, and many more. Studies have used probability models to find a "general matchup formula" [3], and logarithmic models [5]. These studies are part of a bigger research question that searched for the best predictor of team efficiency, and team wins. Finding the best combination of these features is one of the key challenges to this research area. An additional challenge surrounding this topic includes finding reliable data. The datasets used in studies have had to be filtered to use only pitchers / batters who have passed a threshold of times of appearances. This decreases the amount of usable data.

The conclusions of studies have shown that there is an efficient way to estimate the probability that team X beats team Y, but our research findings attempted to expand that to predicting if a batter is 'successful' against a pitcher (a successful at bat as one in which the batter either progresses to a base, scores a home run, or gets out but allows a teammate to progress) [3]. The Bayesian Log5 model was one attempt to predict with decent accuracy if a batter would get on base. However, this model can be improved by considering other significant variables. One example of such a variable to further improve the predictive performance is a batters' pitch recognition skills.

These conclusions are different to our own because our data set was actually much larger. Bayesian hierarchical model only used resampling with only 20 players (5 pitchers, 15 hitters) and tried to find the best was to specifically predict those players[5]. This model was evaluated using MSE (.0464) rather than accuracy, precision and recall like we did. The probability of MLB data from 2012 only used data from batters who had at least 502 plate appearances and pitchers who have faced at least 502 batters because that is the minimum number of plate appearances to be eligible for the Major League Baseball batting title. Additionally, since this model made probability models of individual matchups, they had a much smaller sample size than we did. This model also ignored 'walks' because they are affected but other factors (baserunner configuration, number of outs) and not just the ability of the batter or pitcher [3]. The results of this model were evaluated using $\chi 2$ tests p values [3]. The similarities in our model's conclusion with those of the literature we reviewed were that more specific data would help to improve these models.

It is also interesting to note that research has also shown that teams that are especially efficient are not likely to sustain those levels the next year.[6] This brings light to the difficulty of trying to find the most predictive parameters for matchups as they are year-to-year.

5 Results and Conclusions

Our data set was split into a training, validation, and test set consisting of 60%/20%/20% of the data respectively. Our best performing model was determined to be a simple logistic regression model with a C (regularizer) value of .0001. This C value, as we used SKlearn, represents the inverse of lambda. Using such a low C value means that we are reducing the complexity of our model or heavily penalizing complexity. This makes sense as our model ultimately has thousands of features to train on.

After training our model, we achieved an accuracy of 75.9%, precision of .78, recall of .42, and F1 score of .54 on our validation set. An accuracy of 76%, precision of .79, recall of .41, and F1 score of .54 was achieved on our test set. This is an improvement over the trivial model with an accuracy of 66% and a precision, recall, and F1 score of 0.

The linear SVC model we created was inconsistent in its results but on average we saw about a ~75-76% accuracy, precision of .9, recall of .1, and F1 score of .18 on the test set. Clearly we can see its higher False Negative Rate (FNR) relative to that of the logistic regression model, given its recall. Although it may outperform our logistic classifier at times in terms of accuracy, we decided that the variability in the precision and recall was a shortcoming that made it inferior to the simple but consistent logistic classifier.

We believe that the SVC performed poorly/ inconsistently given the size and difficulty of distinguishing positive and negative at bats. When performing our EDA, we found that there really was not a feature that could easily distinguish between these different outcomes. This makes us believe that there is a lot of overlap between these two outcomes which would make it difficult for the SVC given its algorithm. This may be why we see so many false negatives. Logistic regression on the other hand does not necessarily face these same challenges given its emphasis on likelihood rather than classification error.

Our feature representation is too large to display, but we are confident that our features, given the data, are ideal for this predictive task. The majority of our work was spent on cleaning and developing these features. As was already mentioned, features that offered little value (or did not have a description that identified their purpose in the dataset) were thrown out and we kept only those that we deemed important.

In future research, collecting and including more specific data would help improve the models in this type of research. Information about whether walks are intentional or using a quantifiable measure of a batter's pitch recognition skills would influence a predictor. As mentioned in the literature section, these types of prediction models are difficult because of the variation in team performance year-to-year. A model of this kind may be better suited for individual team predictions so it could pick up on the specifics of a smaller amount of players instead of trying to generalize to the entire MLB.

REFERENCES

[1] Index of /Components/Game/Mlb/, gd2.mlb.com/components/game/mlb/.
[2] https://www.kaggle.com/pschale/mlb-pitch-data-20152018#atbats.csv
[3] https://sabr.org/research/matchup-probabilities-major-league-baseball
[4] https://www.retrosheet.org/game.htm, https://www.baseball-reference.com/
[5] https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0204874
[6] https://diamond-mind.com/blogs/baseball-articles/43548612-2007-team-efficiency?_pos=1&_sid=6f4eb6aa7&_ss=r