At-bat Outcome Prediction Using MLB Pitch Data

Nathaniel Callens Jr, Hunter Lybbert, Jeremy Rawlings, Jake Snow

November 2021

1. Introduction

In July 2021 the Detroit Tigers hosted the Texas Rangers in the middle of a hot, muggy week in Detroit. During the series, the Tigers debuted one of the most dramatic defensive shifts in the modern era against then Rangers slugger Joey Gallo. The Tigers' second baseman was in shallow right field. The shortstop was in shallow right-center. The third baseman was in left field and the left fielder was in left-center. Only one infielder had their feet on the dirt: the first baseman, barely. An entire side of the infield was as open as a 7/11. Tigers pitcher Matt Manning jammed Gallo with a high, inside fastball and what did Gallo do? He popped out to the shortstop...in shallow right-center field. Batters have tendencies, and opposing managers, pitchers, and defenses are learning to take advantage of these tendencies to produce the second most important stat in baseball: the out.

Every year in Major League Baseball, more than 750,000 pitches are thrown over the course of more than 2,500 games. Each pitch has an outcome, and every at-bat ends with either an out, or someone on base (including runs scored). We are interested in being able to predict the outcome of an at-bat based on pitch location, spin rate, velocity, and a host of other features related to individual pitches. Knowing with some degree of certainty what the outcome of a pitch and at-bat will pave the way for defenses to produce more Joey Gallo-like shifts. It also opens doors for hitters and managers to explore unique solutions to produce the single most valuable stat in baseball: runs.

2. Data

In our quest to predict the outcome of at-bats we came across two important datasets on Kaggle[1]. One dataset contained information from every pitch thrown over the last three years, and the other had information about every at-bat over the last three years. Since we are only looking at the outcome of an at-bat, i.e. what happens on the last pitch of the at-bat, we inner merged the two datasets on the ab_id[2] and dropped all pitches except the last pitch from every at-bat. The resulting dataframe represents the last pitch of every at-bat over the last three years in Major League Baseball. There are 50 columns of features that describe such things as (x, y) location at the time the ball crosses the plate, if there are runners on first, second, or

[1] `https://www.kaggle.com/pschale/mlb-pitch-data-20152018`

[2] at-bat ID.

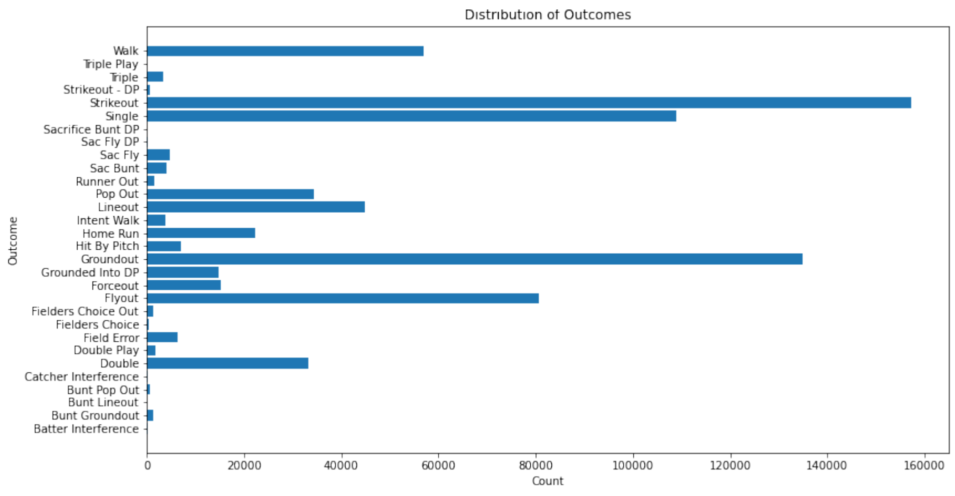


Figure 1: Distribution of at-bat outcomes (events).

third base, the initial velocity of the ball, to name just a few. The column 'event' is the column of labels that we are interested in predicting on. For a distribution of those outcomes, see Figure 1. Additionally, Figure 2 is an abbreviated look at the final dataframe in question and some of its features.

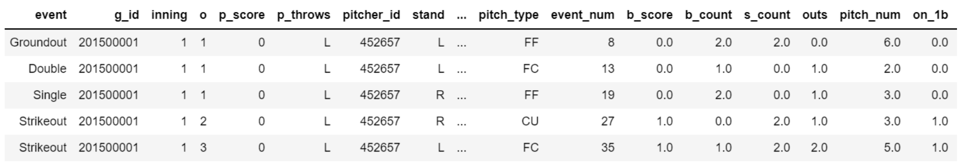
Even though we end up reducing our event space because of the complexity of predicting on 29 events, we still examine the pitch location distribution of certain events. We use the 'px' and 'pz' columns to plot the baseball at the moment it crosses the plate. Figure 3 contains selected pitch placements for flyouts and groundouts, two common outs in baseball.

Two more events that we examine are on opposite ends of the spectrum. Figure 4 contains pitch placements for strikeouts and home runs. The placement distribution of strikeouts is very interesting since most of the pitches are out of the strikezone, possibly indicating that batters are chasing bad pitches out of the zone when they have two strikes.

2.1 Cleaning the data

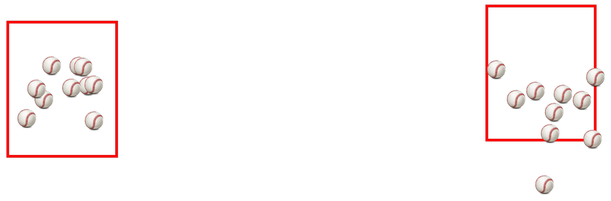
To start off, we needed to clean our 'event' label. Originally, we label encoded events from 1 - 29. Using this method however made training any model take a very long time to compile, so we decided to simplify our event space. Using our domain expertise, we relabeled events in our event space as either 'out' or 'on'.[3] For example, since a groundout, flyout, double play

[3]The key to baseball is getting on base and getting your opponent out.



event	g_id	inning	o	p_score	p_throws	pitcher_id	stand	...	pitch_type	event_num	b_score	b_count	s_count	outs	pitch_num	on_1b
Groundout	201500001	1	1	0	L	452657	L	...	FF	8	0.0	2.0	2.0	0.0	6.0	0.0
Double	201500001	1	1	0	L	452657	L	...	FC	13	0.0	1.0	0.0	1.0	2.0	0.0
Single	201500001	1	1	0	L	452657	R	...	FF	19	0.0	2.0	0.0	1.0	3.0	0.0
Strikeout	201500001	1	2	0	L	452657	R	...	CU	27	1.0	0.0	2.0	1.0	3.0	1.0
Strikeout	201500001	1	3	0	L	452657	L	...	FC	35	1.0	1.0	2.0	2.0	5.0	1.0

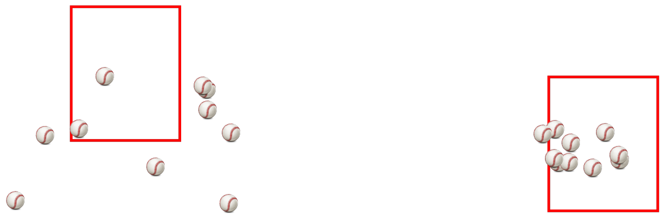
Figure 2: First five rows of the merged dataframe from original datasets of at-bats and pitch information. The full dataframe has 50 columns; some of the ones displayed are the handedness of the pitcher and the number of strikes at the time of the pitch. The 'event' column contains the labels and are the results of the given at-bat (row).



(a) Flyouts

(b) Groundouts

Figure 3: Samples of pitches around the strikezone resulting in (a) flyouts and (b) groundouts. Notice that for (b) pitches are skewed to the bottom of the strike zone, indicating that batters hit the top of the ball, causing a ground out.



(a) Strikeouts

(b) Home runs

Figure 4: Samples of pitches around the strikezone resulting in (a) strikeouts and (b) home runs. Graph (a) is surprising, since this sample has a very high variance. We can infer that batters are chasing these crazier pitches, resulting in strikeouts. Pitches in (b) are skewed towards the inside of the plate, which is a good spot for power hitting, resulting in more home runs.

and strikeout are all outcomes where the batter gets out, we relabeled these events (and others like them) as 1, indicating an out. Events like single, double, and triple (and others like them) are relabeled as 0, indicating the batter got on base. Keeping our event space simple allowed for more generalized models.

In addition to relabeling our event space, we found that some events do not directly result in an out or a runner on base. Some of these events included "Runner Out"[4], "Field Error"[5], and "Intent Walk"[6]. We drop these events that have no bearing on the outcome of 'on base' or 'out'. We as well dropped all null values because we determined we had more than enough data points. Dropping these data points only resulted in a minimal loss of 2% of data, which we consider to be insufficient in the scope of this project.

We re-indexed the dataset using the ab_id for better generalization. This allows us to cross-reference the raw datasets using the ab_id to find the pitcher, game, etc. Certain features like top, stand, and p_throws needed to be one-hot encoded.[7] To avoid singular matrices, we dropped the first level while adding a single constant column to account for all dropped levels. Finally, doing a 70-30 split on our data using Scikit-learn's train_test_split method concludes our data cleaning processes.

2.2 Dropped Features

Immediately, there were a few features we could clearly identify as non-integral to our project, such as game id, pitcher id, batter id.[8] Additionally, there are a number of features that are redundant that needed to be dropped. For example, the columns 'x' and 'y' describe the same information as the 'px' and 'pz' columns, the difference being they are using an outdated coordinate system. Another example was the 'o' and 'out' columns. The 'o' column describes the number of outs after the batter, while 'out' describes the number of outs at the time of the at-bat. The 'zone' feature labels the location of pitches about the strikezone. We found this to be redundant to the classification problem at hand because 'px' and 'pz' already describe the

[4]When a runner currently on base gets out independent of the batter.

[5]When a fielder commits an error, resulting in an outcome independent of what the pitch data can tell us.

[6]Technically a pitcher doesn't have to throw a pitch in order to intentionally walk somebody, so we discarded these.

[7]The features 'on_1b', 'on_2b', and 'on_3b' were already one-hot encoded. Therefore 'on_3b' was dropped to be included in the constant feature.

[8]See the conclusion for further applications.

position of the ball when it crosses the plate. We dropped columns with redundant information, keeping columns with more up-to-date data.

The ‘code’ feature records the result of the pitch and cannot be included as a feature. The ’type’ feature simplified pitch outcome codes and therefore must be excluded as well. The ‘event_num’ feature was used for determining when an ejection occured. Ejections do not contribute to the outcome of a pitch. The feature ‘pitch_type’ was dropped because it is linearly dependent of the other features which numerically describe the type of a pitch. Consequently, ‘type_confidence’ was dropped due to the fact that it is a percentage of how confident the pitch was correctly classified. The ‘pitch_num’ feature was also dropped since we are only examining the last pitch of every at-bat. The feature ’top’ indicates which team is on offense, and does not contribute to each at-bat. Dropping these columns reduced our feature space from 50 down to 33 features. Fewer features can help keep our models simpler and more generalized.

2.3 Feature Description

Instead of describing every feature, most of which are pretty self explanatory, we will describe a couple of features of note. ‘Nasty’ is one such interesting feature. This column describes the greatest difference between the trajectory of the pitch and the actual pitch path. In other words, if we were to take the position of the pitcher’s hand at the point of release and drew a straight line to the point where that pitch crossed the plate, ‘nasty’ measures the maximum variation from that line.

Other interesting variables include columns like ‘vx0’, ‘vy0’, and ‘vz0’. These describe the initial velocity of the pitch in the x, y, and z directions, respectively. ‘ax’, ‘ay’, and ‘az’ similarly describe the initial acceleration.

Finally, there are some variables that might be confusing for those unaffiliated with some of the subtleties of baseball. For example, ‘sz_top’ and ‘sz_bot’ describe the bottom and top of the strike zone. According to MLB, the strike zone is defined ‘from the midpoint between a batter’s shoulders and the top of the uniform pants – when the batter is in his stance and prepared to swing at a pitched ball – and a point just below the kneecap.’[21] It becomes apparent reading this description that the strike zone changes batter to batter, making columns defining the strike zone for individual batters critical in considering pitch location. For a more detailed treatment on every feature in the dataset, consider this page[9] under the variables tab.

[9] `http://inalitic.com/datasets/mlb%20pitch%20data.html`

3. Models

3.1 Logistic Regression

With 29 different outcomes in the event space, we thought this classification problem might lend itself well to a logistic regression model using softmax across all possible events. Using the same features mentioned above, and with label-encoded outcomes, we fit the logistic regression model on a train-test split of the cleaned data. The goal of this model is to find the optimal set of $\boldsymbol{\beta}$s by minimizing the categorical cross-entropy loss function (which includes the softmax).

Using Scikit-learn's LogisticRegression package we tried to fit a model on the training data and labels and found that the model did not converge to a solution after 24 hours of running. Because our labels are label encoded and our features have different ranges of values, we used lbfgs as a solver which only supports an L2-regularization as a penalty. We were restricted in our choice of solvers due to the fact that we are dealing with a multi-class classification problem. We set multi_class to 'multinomial' and started training. Because the model was running for over 24 hours with no results or convergence, we abandoned this idea and consequently decided to greatly simplify our model in order to obtain interpretable results.

Our simplified data still did not meet some of the requirements to use other solvers in Scikit-learn's LogisticRegression package. We proceeded with lbfgs and L2-regularization, hoping that our reduced event space will aid in the time to convergence. This time the model successfully found an optimal set of coefficients for a binary classification problem using a softmax function. Finally scoring the model on the test set, we achieved 70.6% accuracy in distinguishing between which pitches led to batters getting on base or out.

A quick inspection of the training and testing set data revealed that 68% of the points are classified as 'out'.[10] Therefore, if we simply guessed that every at-bat resulted in an out, we would only be 2 percentage points less accurate than the logistic regression model described here. Still, with so many features to regress on it is impressive that the optimal arrangement of coefficients for the softmax yields a better accuracy than guessing, even if it isn't by much.

Figure 5 shows the ROC curve corresponding to our successfully trained model.

[10]This 68% was true for both the training and testing set.

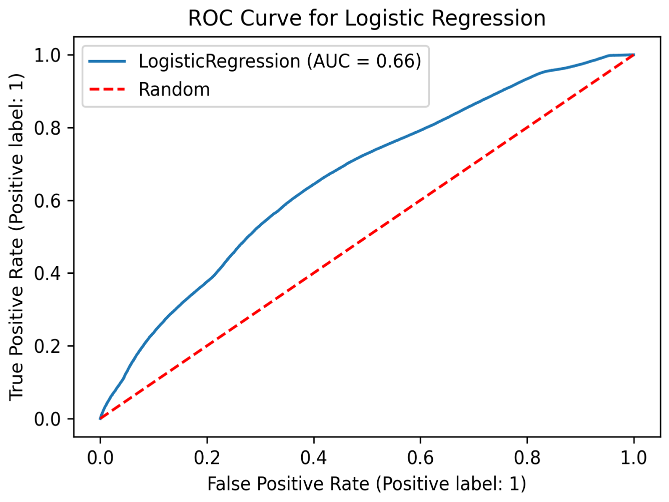


Figure 5: ROC curve for the logistic regression model trained above with AUC score of 0.66. The placement of the curve above the diagonal line indicates that the model is better than random at assigning novel datapoints to either ‘on-base’ or ‘out’.

3.2 Random Forest

We wanted to use a random forest classifier here because the advantages it provides from aggregating many classification trees. An advantage of a random forest is that it avoids overfitting by having many classification trees vote, balancing out the weaknesses of each tree and narrowing in on the true solution.

Using Scikit-learn, we trained several random forests with various hyperparameters. We tried various number of classifiers, max depth, and min samples in a leaf. After measuring each models accuracy on the test set out best performing model had a score of 74.68%. This model was trained with the following parameters: 100 estimators, no max depth, and minimum samples in a leaf 1.

The random forest also helped us identify which features were most important in predicting the outcome of the play, either on base or out. We looked at the 10 most important features and saw some interesting similarities between them all. For example, several of these features were about the location of the pitch leaving the pitchers hand and when it crossed home plate. A few others had to do with the velocity and acceleration of the

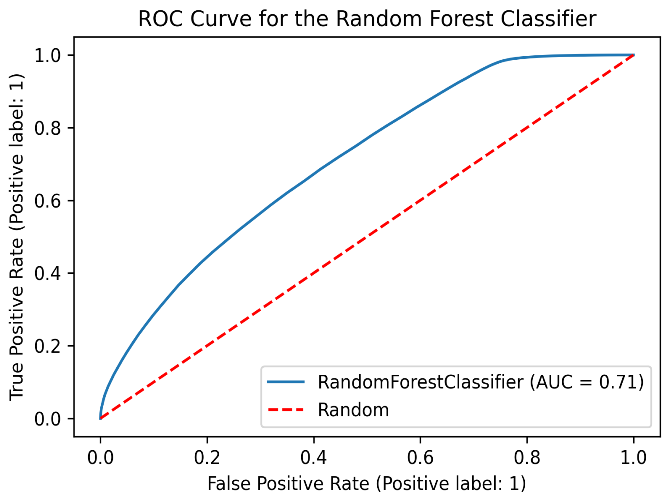


Figure 6: ROC curve for the Random Forest model. The curve is above the diagonal line indicating a better than random model. Notice the area under the curve (AUC) for the logistic regression model was only 0.66 while the random forest model was higher with 0.71

pitch. A possible explanation is that pitches thrown to specific locations at top speeds tend to be harder to hit and therefore harder to get on base. The least important features were the inning, location of base runners, and the score of the game. We can infer from this that the outcome of the current at-bat is independent of the results of previous at-bats.

3.2.1 Gradient Boosting

An additional attempt to build upon our Random Forest model was to implement Gradient Boosting. This didn't prove fruitful as the score did not exceed the accuracy of our original model.

3.3 Naive Bayes

Naive Bayes turned out to be extremely ineffective. An initial fit and score of the train and test data produced a whopping 33% accuracy. Adjusting hyperparameters boosted the model to the 35-38% accuracy range. Upon further inspection, the results actually make a lot of sense. Scikit-learn's implementation of Naive Bayes assumes a distribution of data. For example, there exists GaussianNB or BinomialNB. These models assume the data to be normally or binomially distributed respectively. Our data has both continuous and discrete data. Picking a Gaussian Naive Bayes model to fit the data on assumes that the inning number or the home team's score is normally distributed, which is nonsense. Since this problem isn't particularly suited for Naive Bayes to begin with, we spare ourselves the work of constructing a model by hand that would account for the difference in data types, but it could be something to look in to for future analysis.

4. Ethics

A debate on ethics may not lend itself quite as easily to a sports project as it might to a project using medical data. Regardless, there are still ethical concerns that should be raised.

At the 2022 Winter Meetings, MLB's top brass will gather like they do every year to discuss potential rule changes. Among the rule changes being presented is one that would ban the defensive shift that has been so effective in getting batters out [Sau18]. The single biggest reason that such a change is being considered? Boredom [Bic21]. Over-analysing the sport means that players' athletic abilities are being put on the back-burner in favor of math and data. Some argue that it makes the sport better, fostering a more strategic environment where players and coaches can prepare more precisely and anybody has a shot at winning if they have the right data. Fans complain about excessive shifts, rising strikeout numbers and boring, low-scoring games. Regardless of your opinion on the subject, there is no denying the effect that data has had and will have on baseball and sports in general.

As data scientists dipping our toes in the pool of sports data, we need to ask ourselves where the line ought to be drawn. When has our predictive power gotten good enough, and we need to stop and let players just do their thing? When will sports become so deconstructed and predictive that they stop being fun? It may seem like the very distant future, but as we improve our models, these are important questions to keep in mind.

5. Conclusion

We sought to predict the outcome of an at-bat based on several data features and slid into many problems on our way to home-base. Just like any game of baseball we got on base with our successful models and got out with failed models. We simplified our label space due to over-complexity and down-sampled our feature space to exclude unnecessary features. Our best model predicted the at-bat outcomes with 74.68% accuracy. This is better than guessing (50-50) and better than predicting all ‘out’ (68-32). With this analysis, we believe our model did not overfit and there does exist a correlation between pitch features and the at-bat outcome.

In our data cleaning process, we dropped all data points containing null values. Further analysis needs to be done to determine if the null values contribute to some hidden trend. This might provide some additional insights to our project.

Further application of this paper would include applying this to a specific batter or pitcher. Defenses could filter the at-bat and pitches data set by the batter_id and analyze how this specific batter reacts to different pitches, what pitch can get him out, or where his bats tend to end up in the outfield, etc. This would allow teams to study the opposing team batters to better prepare defenses to get more outs, as well as letting offenses filtering by pitcher_id and analyzing the tendencies of a certain opposing pitcher. This could allow for batters to prepare for certain pitches or expect other tendencies which could lead to more runs. We believe that the greatest value herein resides in these further applications.

References

[Sau18] Patrick Saunders. "Banning the shift in MLB generates growing, heated baseball debate". In: (2018). URL: https://www.denverpost.com/2018/12/16/banning-shift-mlb-baseball-debate/.

[Bic21] Dan Bickley. "MLB's obsession with analytics is killing what's great about baseball". In: (2021). URL: https://arizonasports.com/story/2549057/mlbs-obsession-with-analytics-is-killing-whats-great-about-baseball/?show=comments.

[21] "MLB's Strike Zone Official Ruling". In: (2021). URL: https://www.mlb.com/glossary/rules/strike-zone.