Classification of Major League Baseball Pitch Outcomes

Seyi Oyesiku
Drexel University - Winter 2023
Philadelphia, Pennsylvania

***Abstract* - MLB pitchers throw a variety of pitches to fool batters and prevent runs. However, not all of these pitches are equally effective. In this study, we propose 3 classification machine learning models to predict the outcomes of pitches based on relevant variables. Specifically, the models put forth are the following: Logistic Regression (LR), Decision Tree Classifier (DTC), and a Random Forest Classifier (RF). After exploring and transforming the dataset, we implemented these models in PySpark to obtain the following outcomes. The Logistic Regression was the worst performing model with an accuracy of 0.51. The Decision Tree did slightly better with an accuracy of 0.55. However, the best model of all of our models was the Random Forest with an accuracy of 0.604. While these accuracy scores may seem low, the task is a difficult one for reasons explored in Section VI. Future work may produce better results according to recommendations set forth in the aforementioned section.**

I. INTRODUCTION

The position of an MLB pitcher may be the single most difficult role in the sport. They shoulder the majority of the responsibility for preventing batters getting on base and runs being scored. To do so, they vary in speed, spin, and position of their pitches. However, some pitches are less effective than others.

Thus, there is a need to understand the types of pitches that will result in different outcomes. For example, a strikeout versus a base hit can create a huge difference in the outcome of a game.

The application of this modeling would be useful for both pitchers and coaches, as it could be implemented real-time during team practice sessions and games. Detailed ball tracking already exists in the form of Statcast, a combination of "camera and radar systems" which provides detailed data like that contained within the dataset of interest [1].

There have been a number of analyses dedicated to understanding the interplay between hitter-batter matchups [2-3] and machine learning algorithms to predict the outcome of games [4-5]. However, there has been relatively little public research dedicated to the specific types of pitches that are most effective on a league-wide level. This is the primary focus of the paper.

II. DATASET

The dataset used is called "MLB Pitch Data 2015-2018" [7]. While it resides on Kaggle, the original author, Paul Schale, scraped the data directly from the MLB website.

The dataset includes many different csv files with game data from the MLB. However, the focus of the project is the "pitches.csv" file. It contains all pitches thrown between 2015 and 2018 - 2,867,154 in total [7]. It gives the speed, spin break angle, and break length among other data for each pitch. Please refer to Appendix 1 for a full list of all 40 variables, complete with descriptions. See Figure 1 below for a sample of the variable descriptions.

APPENDIX 1: "pitches" Dataset

Column Name	Data Type	Column Description
px	continuous	Ball's left-right distance from middle as it crosses plate, feet [6]
pz	continuous	Ball's height as it crosses plate, feet [6]
start_speed	continuous	Starting speed of pitch, mph
end_speed	continuous	Ending speed of pitch, mph
spin_rate	continuous	Rate of spin of ball in flight, rpm [6]
spin_dir	continuous	Direction of spin of ball in flight, degrees [6]
break_angle	continuous	Direction of breaking ball, degrees [6]
break_length	continuous	Greatest deviation from straight-line path of pitcher to home plate, inches [6]

Figure 1: First 8 rows of Appendix 1

III. EXPLORATORY DATA ANALYSIS

The first step of the exploratory data analysis was to create visualizations showing the distribution of all of our categorical features. This included the type variable, which is the target variable the models will be trained against.

The categorical features in this dataset are code, type, pitch_type, nasty, zone, on_1b, on_2b, and on_3b. We decided not to use code because this feature was simply a more detailed version of our target variable type. Pitch type was also not considered because there were many other continuous variables that cover the pitch type. We also have continuous variables that cover the nasty and zone better than using these categorical features, so we disregarded these, as well. Finally, there was no need to use the binary features about who is on each base because they were less directly related to the pitch and its outcome. Type, the target variable, has no null values and is not too imbalanced. The distribution is as follows: 45% strikes, 36% balls, and 18% are ball-in plays. The distribution of each of these different play result types can be seen below.

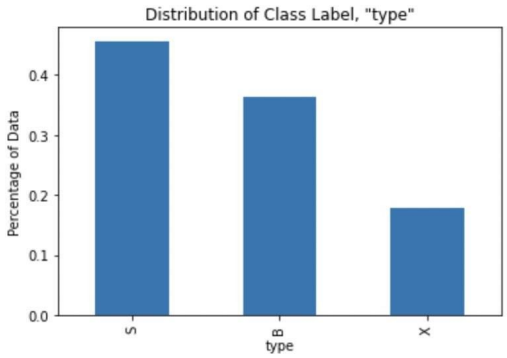

Figure 2: Distribution of Type Values

When performing the exploratory analysis on the continuous variables, we found that there were many more to use. Therefore, we had to create different distribution plots for each of these features in order to understand them in detail.

We began with the ‘px’ and ‘pz’ features that are described in Appendix 1. Figures 3 and 4 below show the distribution of these features’ values. We found that the mean of these values was quite small, but the standard deviation was nearly one foot each direction. With the ‘pz’ feature, we found that the average ball was about 2 feet above the dirt, and the standard deviation was nearly a foot above or below. However, we see a normal distribution in both of these features, so they should still work well.

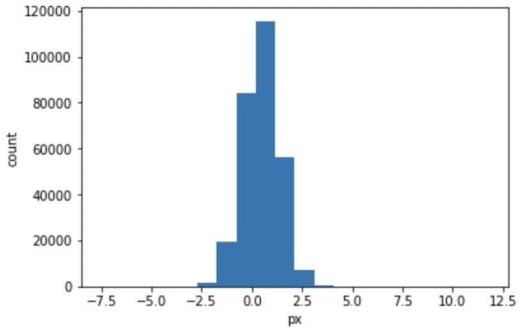

Figure 3: Distribution of ‘px’ Values

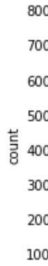

Figure 4: Distribution of ‘pz’ Values The spin rate and spin direction were both important features to use for our analysis. The spin rate is only missing a few values, and average value is 1731 rpm with a standard deviation of 683 rpm. The spin direction is also missing a few rows with an average of 180 degrees in the vertical direction. The standard deviation of the spin direction is 67 degrees. The figures below show the distribution of the spin rate and the spin direction. Here, we can see spin rate and direction are not necessarily normally distributed.

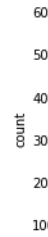

Figure 5: Distribution of Spin Rate Values

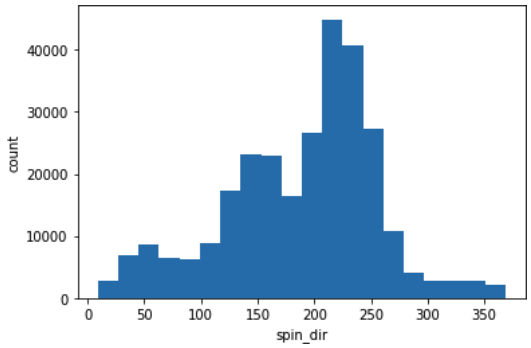

Figure 6: Distribution of Spin Direction Values

The next features we considered were the acceleration values in the x,y, and z-directions. These are labeled as 'ax', 'ay', and 'az'. In the x-direction, we saw there was an average acceleration of 2 feet per second per second in the negative direction (to the catcher's left). In the y-direction, we noticed that there was an average acceleration of 26 feet per second per second slowing down toward the pitcher. The standard deviation was only 4 feet per second per second. Finally, the average acceleration in the z- direction was 23 feet per second per second downwards with a standard deviation of 9 feet per second per second. The figures below show the distribution of these acceleration values.

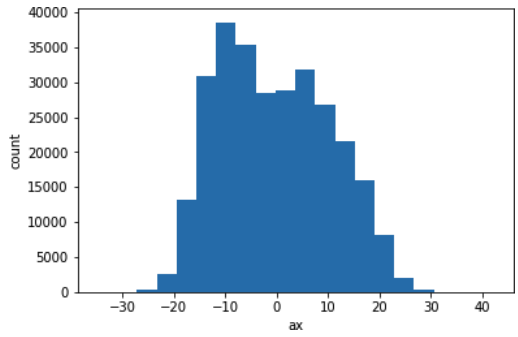

Figure 7: Distribution of the Acceleration in the x-direction

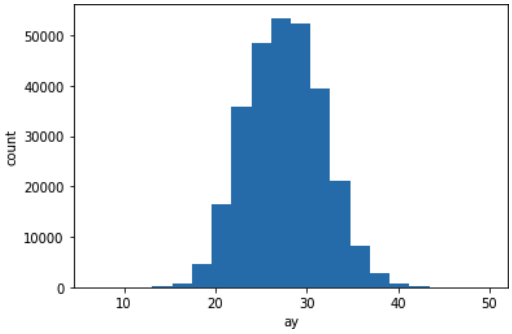

Figure 8: Distribution of the Acceleration in the y- direction

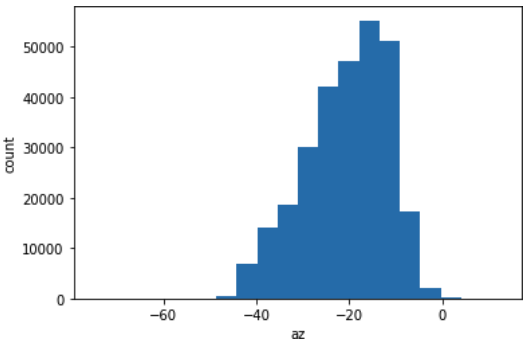

Figure 9: Distribution of the Acceleration in the z- direction

The velocity in the x, y, and z-directions was evaluated next because the velocity of a pitch certainly affects how a batter would hit the ball. In the x-direction, we saw that the average initial pitch release velocity was 2.3 feet per second with a standard deviation of 6 feet per second. In the y-direction, we see that the velocity decreases as it moves toward the catcher at an average value of 128 feet per second with a standard deviation of 9 feet per second. Finally, in the z-direction, we saw that the initial pitch velocity was 4 feet per second downward with a standard deviation of 3 feet per second.

Compared to the mean value, this standard deviation seems quite high. The figures below show the distribution of all of these velocity values.

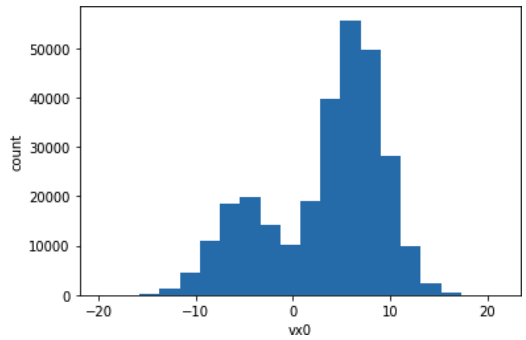

Figure 10: Distribution of Velocity in the x-direction

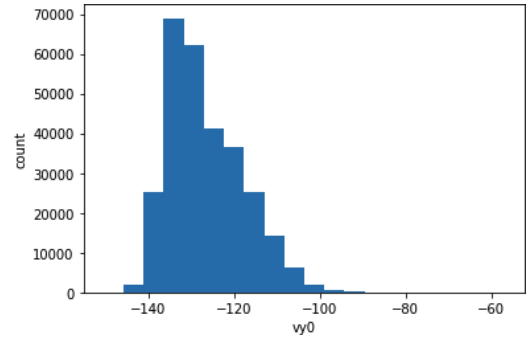

Figure 11: Distribution of Velocity in the y-direction

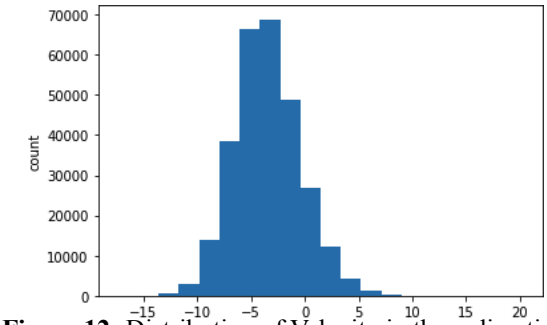

Figure 12: Distribution of Velocity in the z-direction

Finally, we inspected the 'x0' and 'z0' features of the dataset. The mean initial position is about 0.7 ft to the left of the catcher, but the standard deviation is 1.7 feet. With the 'z0' feature, we saw that the average height of the ball was about 5.8 feet with a standard deviation of 3.3 feet downward. The figures below show the distribution of this data.

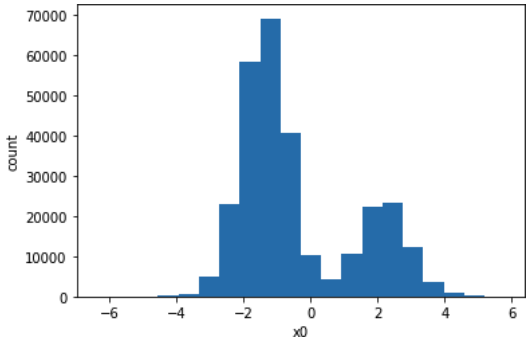

Figure 13: Distribution of the Ball's Initial Position in the x-direction

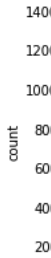

Figure 14: Distribution of the Ball's Initial Position in the z-direction

In summary, though, we can see that there is enough skew and variety in scales that we will need to use robust scaling in order to obtain successful predictions for the model.

IV. METHODOLOGY

A. Feature Engineering and Selection

To perform feature selection, we first had to consider dealing with the missing values in the dataset by firstly dropping all the rows that have missing values using the ***na.drop()*** function. Then we also had to consider filtering the remaining data to include only pitches that were measured from the standard mound distance of 50 feet.

Next, we removed features which 1) would be less relevant for our analysis or 2) overlapped heavily with other variables in the dataset. The ones we did not include were ***break_y, sz_bot, type_confidence, x,y,y0, nasty, zone, code, pitch_type, event_num, b_score, s_count, outs, pitch_num, on_1b, on_2b, and o_3b***.

Then, we created the ***delta_speed*** variable to encapsulate the velocity change of the pitches rather than having the ***end_speed*** as a separate column.

Specifically, we made use of the ***withColumn*** function to create and label our desired column.

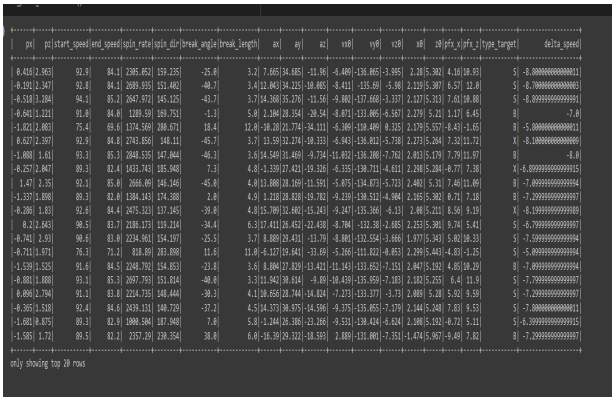

Figure 15: Display of dataframe after creating new values

In order to examine numeric data for multicollinearity, we iterated through all features to obtain correlations for variable pairs in the dataset. Specifically, we used Pearson correlation and set a threshold at 0.8 as a sign of multicollinearity [8].

After taking a look at the results, we found the following suspects: ***start_speed, break_length, start, speed, vy0, break_angle,ax, break_angle, pfx_x, break_lenght, az, break_lenght, vy0, break_lenght pfx_z, ax, pfx_x, ay, delta_speed, az, pfx_z, vx0, x0***. As such, we will be removing the ***start_speed, break_angle, break_length, pfx_x, and pfx_z*** columns from the dataset. We are keeping the ay and delta_speed, as well as vx0 and x0, despite a high Pearson correlation score. Ay is acceleration-related on only the y axis, while delta_speed is velocity related which makes measurement for both of them slightly different.

After dropping all the high multicollinearity columns, we set up a string indexer for our sole categorical variable and label, ***type_target.***

Since all of our features were numeric in nature, we were free to make use of ***RobustScaler*** on the entire feature column. Finally, we assembled a pipeline for the transformation of data prior to modeling consisting of:

1. A string indexer
2. A vector assembler
3. A robust scaler

We started by creating a list for columns to convert and feature columns. We then created stages for the data transformation and pipeline processes enabling us to start splitting our data into train and test dataframes, as well as the application of the appropriate pipelines. The labels that we used for each value in the target column are the following: 0 for Strikes, 1 for Balls, and 2 for any ball in-play results, i.e. anytime that a batter hits the ball and it stays in play.

B. Machine Learning Models

We decided to implement three machine learning models that could accurately predict the pitch outcomes. Generally, we set up our model, created a parameter grid, and set up a crossvalidator to evaluate against a multiclass classification evaluator.

1. Logistic regression: First, we set our maxIter to 10. We set the regParam values to range from 0 to 0.1 in increments of 0.01. Then, we set the elasticNetParam to be [0,1] and incorporated that into our parameter grid.

2. Decision Tree: We set up the Decision Tree parameter grid with a max depth of 3, 5, and 7. The bins were 8, 16, and 32, and we added both the Gini and Entropy impurities.

3. Random Forest: The Random Forest model that we used had 10 trees. In the parameter grid, we used both the Gini impurity and the Entropy impurity. We varied the max depth from 5 to 8 to 10, and built the grid.

V. RESULTS

We used mlib's evaluator for multiclass metrics to obtain the following types of metrics on the models for train and test: confusion matrix, accuracy, weighted recall, weighted precision, weighted F(1) score, and by-class metrics of the aforementioned.

The first model, logistic regression, performed the worst. The tables below show the scores on the training and test data.

DT Train	Precision	F1	Recall	Acc.
Overall				0.555
Weighted Overall	0.473	0.495	0.555	
Strikes	0.523	0.641	0.827	
Balls	0.637	0.551	0.485	
In-Play	0.0	0.0	0.0	

Table 3: Decision Tree Results with the Training Data

Table 1: Logistic Regression Results with the Training Data

LR Test	Precision	F1	Recall	Acc.
Overall				0.519
Weighted Overall	0.472	0.439	0.519	
Strikes	0.489	0.635	0.906	
Balls	0.675	0.407	0.291	
In-Play	0.0	0.0	0.0	

Table 2: Logistic Regression Results with the Test Data

DT Test	Precision	F1	Recall	Acc.
Overall				0.554
Weighted Overall	0.472	0.493	0.554	
Strikes	0.521	0.639	0.827	
Balls	0.637	0.549	0.483	
In-Play	0.0	0.0	0.0	

Table 4: Decision Tree Results with the Test Data

The difference in accuracy values between the train and test runs were minimal. So, the model performs poorly, but generalizes well. One thing we do see is that this model had a decently high recall score when predicting strikes, which is a strength.

Next, the decision tree model performed slightly better than the logistic regression, per the results shown below. However, we still just saw a total accuracy value around 55% for both the train and test sets.

We see that this model's F(1) score for strikes is very similar to LR's, but it is much higher for balls. So, we see notable improvement in its predictive power for the second class. Note that this model still is very poor at predicting the third and final class, In- Play.

Finally,we come to the decision tree model. This has the highest accuracy value of all that we have built in the study, 0.604. It is now much better than chance at correctly predicting classification in this task.

Table 5: Random Forest Results with the Training Data

Table 6: Random Forest with the Test Data Note that, while it still cannot distinguish in-play outcomes, it is much better than DT for predicting strikes and balls. This makes sense, as a random forest will usually perform better than just one decision tree. Finally, just like the other models, it did the best when trying to predict the strikes.

VI. CONCLUSION AND FUTURE WORK

Predicting the outcome of a pitch gives an edge to batters and pitchers alike looking to up their game. Thus, this paper sets forth classification models to answer this need. The result of these efforts unveiled the complexity of this task, though. The best- performing model, the Random Forest (RF) Classifier, still struggles to achieve high accuracy and cannot predict In-Play outcomes with any degree of certainty.

There are a number of reasons why this modeling effort failed to perform well. First, many strikes and balls can look similar to balls that are put in play. Specifically, this derives from the individual variations in batters - one batter can hit a 98 mph fastball, while another cannot. Second, the modeling effort here assumes that pitches are independent, which is a naive assumption. For example, a batter may not swing at an easily-hittable strike because the three prior pitches were all balls. Finally, the task of predicting a multiclass classification task is far more difficult than a binary one.

In terms of future work, there is far more that can be done to improve the modeling efforts. First, even though the labels did not seem very imbalanced,
class-weighting could be undertaken. Second, there are some variables that may not have a linear relationship with the class label. So, further feature transformations could be undertaken. Finally, pulling in more features could better account for the individuality of batter-pitcher dynamics.

REFERENCES

1. "Statcast." *MLB*, https://www.mlb.com/glossary/statcast. Accessed 20 February 2023.
2. Silver, Joshua. “Singlearity: Using A Neural Network to Predict the Outcome of Plate Appearances.” *Baseball Prospectus*, 9 July 2020. https://www.baseballprospectus.com/news/article/59993/singlearity-using-a-neural-network-to-predict-the-outcome-of-plate-appearances/. Accessed 20 February 2023.
3. Pemstein, Jonah. “The Outcome Machine: Predicting At Bats Before They Happen.” *FanGraphs: Community Research*, 23 October 2014. https://community.fangraphs.com/the-outcome-machine-predicting-at-bats-before- they-happen/. Accessed 20 February 2023.
4. Nourse, Garrett. “A Machine Learning Algorithm for Predicting Outcomes of MLB Games.” *Towards Data Science*, 6

October 2021. https://towardsdatascience.com/a-machine-learning-algorithm-for-predicting-outcomes-of-mlb-games-fa17710f3c04. Accessed 20 February 2023.

5. Huang, Mei-Ling and Yun-Zhi Li. “Use of Machine Learning and Deep Learning to Predict the Outcomes of Major League Baseball Matches.” *Applied Sciences*, vol. 11, no. 10, May 2021, p. 4499. *Crossref*, https://www.mdpi.com/2076-3417/11/10/4499. Accessed 20 February 2023.
6. Fast, Mike. “MVN: A PITCHf/x primer.” *Fast Balls*, 14 January 2008. https://fastballs.wordpress.com/category/pitchfx-glossary/. Accessed 20 February 2023.
7. Schale, Paul. MLB Pitch Data 2015-2018. *Kaggle*, 2020. Web. https://www.kaggle.com/datasets/pschale/mlb-pitch-data-20152018. Accessed 20 February 2023.
8. Shrestha, Noora. “Detecting Multicollinearity in Regression Analysis.” *American Journal of Applied Mathematics and Statistics, 2020, Vol. 8, No. 2, 39-42*. Web. http://article.sciappliedmathematics.com/pdf/ajams-8-2-1.pdf. Accessed 11 March 2023.

APPENDIX 1: "pitches" Dataset

Column Name	Data Type	Column Description
px	continuous	Ball's left-right distance from middle as it crosses plate, feet [6]
pz	continuous	Ball's height as it crosses plate, feet [6]
start_speed	continuous	Starting speed of pitch, mph
end_speed	continuous	Ending speed of pitch, mph
spin_rate	continuous	Rate of spin of ball in flight, rpm [6]
spin_dir	continuous	Direction of spin of ball in flight, degrees [6]
break_angle	continuous	Direction of breaking ball, degrees [6]
break_length	continuous	Greatest deviation from straight-line path of pitcher to home plate, inches [6]
break_y	continuous	Distance from home plate to point in pitch trajectory where break_length occurred, feet [6]
ax	continuous	Acceleration of pitch at release, feet per second per second x-axis [6]
ay	continuous	Acceleration of pitch at release, feet per second per second y-axis [6]
az	continuous	Acceleration of pitch at release, feet per second per second z-axis [6]
sz_bot	continuous	Distance from ground to bottom of current batter's strike zone, feet [6]
sz_top	continuous	Distance from ground to top of current batter's strike zone, feet [6]
type_confidence	continuous	Confidence in pitch_type classification, value multiplied by 1.5 if pitch is known to be frequently-used pitch [6]
vx0	continuous	Velocity of pitch at release, feet/second x-axis [6]
vy0	continuous	Velocity of pitch at release, feet/second y-axis [6]
vz0	continuous	Velocity of pitch at release, feet/second z-axis [6]
x	continuous	From legacy Gameday coordinate system, horizontal position as ball crosses plate [6]
x0	continuous	Ball's left-right distance from middle at initial measurement point, feet [6]

y	continuous	From legacy Gameday coordinate system, vertical position as ball crosses plate [6]
y0	n/a (one value)	Distance from home plate where “initial” values are recorded, feet [6]
z0	continuous	Height of pitch at release, feet [6]
pfx_x	continuous	Horizontal movement of pitch thrown, inches [6]
pfx_z	continuous	Vertical movement of pitch thrown, inches [6]
nasty	discrete	0-100 scale dealing with movement and speed of pitch
zone	discrete	1-14 scale bucketing pitches based on where they land
code	categorical	Records result of pitch in highly detailed manner
type	categorical	Rollup of “code” variable where it only represents strikes (S), balls (B), and in-play (X)
pitch_type	categorical	Type of pitch thrown
event_num	discrete	Event number for finding ejections in other datasets, which is not of use for this analysis
b_score	discrete	Score of the batter’s team
ab_id	discrete	At-bat ID for finding at bats in other datasets, which is not of use for this analysis
b_count	discrete	Number of balls in a given at-bat
s_count	discrete	Number of strikes in a given at-bat
outs	discrete	Number of outs before ball is thrown in a given at-bat
pitch_num	discrete	Number of pitches thrown in a given at-bat
on_1b	categorical	Binary value, 1 if runner on first base, else 0
on_2b	categorical	Binary value, 1 if runner on second base, else 0
on_3b	categorical	Binary value, 1 if runner on third base, else 0

*Note: y dimension is called z dimension in PITCHf/x coordinate system [6]