IST707 – Applied Machine Learning

MLB PITCH TYPE PREDICTION

Peter Kucharczuk

Kishan Polekar

Robert Schnoor

Abstract

This research aims at exploring the Baseball (MLB) data and predicting what kind of a pitch will the pitcher throw based on several factors. We started off by looking at the regular season data for training, and post season data for testing our models out. There are 17 different types of pitches a pitcher can throw, and we aim at fine tuning our models, and selecting the best one for the final results. At the end, we used sample inputs for getting a possible pitch type for each pitcher in the post season which will help the teams and players formulate useful gameplans based on known factors in a game situation.

Introduction: MLB Pitch Type Prediction

This report will show how to predict a pitch type in the MLB based on the pitcher, batter, and situation of the game. We reached our findings through descriptive analysis and various data mining and machine learning techniques. Based on the performance metrics of each model, we highlighted the best ones for the purpose of our results.

The dataset used for this report comes from the "pybaseball" package in Python. It consists of every single pitch thrown in the MLB since tracking has allowed. Since the 2023 season just concluded, we wanted to focus our analysis on just that season. We collected regular season and postseason data and removed about 60 unnecessary columns which resulted in just over 30 columns to work with. The columns that were removed either did not have any correlation on pitch type or were metrics that occurred after the pitch was thrown which is not useful to the batter.

The purpose of this analysis is to help an MLB player or team "game plan" for a certain pitcher. In the MLB, a hitter has less than a half-second to react to a pitch. The hitter must read the movement and velocity of the pitch in real time to garner success. That is why our results could be extremely helpful. If a hitter has an idea of what pitch type is coming their way, they can evaluate where the ball will end up before it reaches them. This will allow them to better track the ball and give them a better chance at making solid contact. However, it is extremely hard to predict a pitch type. There are many factors to consider, many of which are not quantifiable. With that being said, there are also many pitch types (about 3-5 for each pitcher) and some of them can be grouped together. So, even if the batter just knows to expect an off-speed pitch or a high-velocity pitch it can still help them greatly.

Data Preparation

As mentioned, we removed unnecessary variables from the dataset. To do this, we first looked at the column names & descriptions and handpicked some columns which were repeated (like the X-axis & Z-axis position of the pitch at different points after the throw, the pitch description, the deprecated spin rate, spin axis, the ball velocity at 50ft from the ground, etc.). We then removed a few columns based on game knowledge and our learnings from the course that we felt would not contribute in any way to the pitch type prediction. Finally, we looked at the heatmap for all variables against the pitches and then removed some other columns that did not relate at all to the pitch type. This allowed the data to be more digestible and easier to work with.

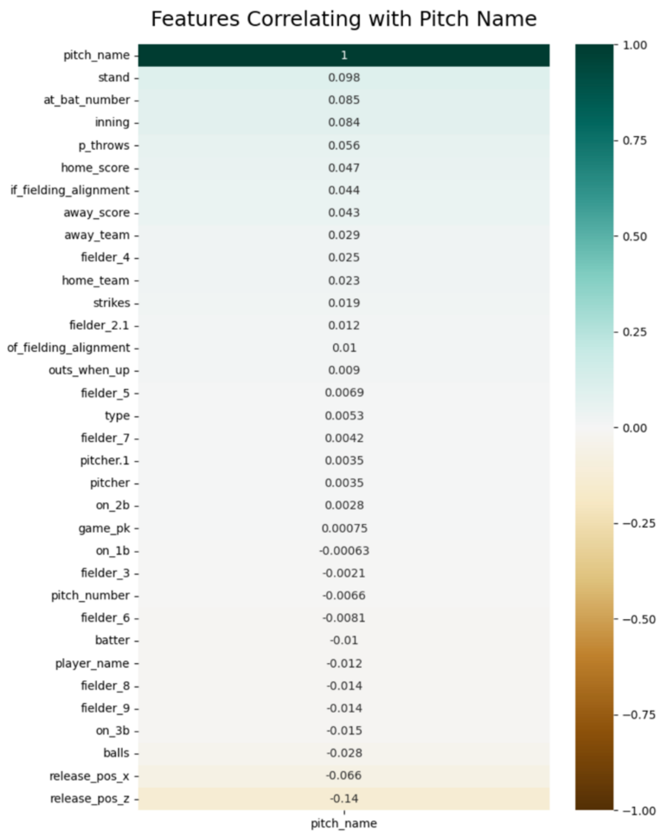

Figure 1: Heatmap of type of pitch against all other columns

With the MLB playoffs having already concluded, we had a built-in test set consisting of the post season data and used the entire regular season for our training dataset. After filtering the data, the train set has 38,456 rows while the test set has 4,532 rows. We identified 14 pitchers that started at least 2 games in the postseason and only included those pitchers in our dataset. Again, we handpicked these 14 pitchers based on the condition mentioned earlier, and on the fact that they had enough pitches per game for our analysis. This allowed us to have enough data points in our test set.

Descriptive Analysis

For our descriptive analysis, we aimed to better understand the data and the pitch arsenals for each pitcher, so we could successfully evaluate the results. This helped in the process of interpreting our results. For example, we better understood which pitchers threw the most pitches and would be harder to predict. Also, we looked at what pitch the pitchers threw in certain situations like strikeouts.

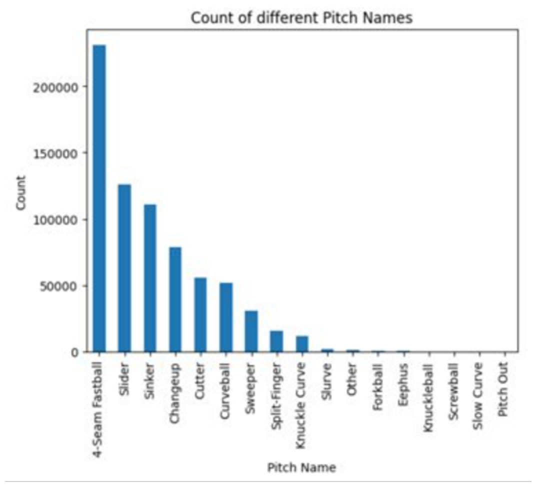

Figure 2: The count of each pitch type in our dataset

The graph above shows the most common pitch types in our dataset. There are 17 different pitch types in our dataset with 4-Seam Fastballs, Sliders, Sinkers, and Changeups appearing the most. This lets us know what to look for in our results. This graph changes drastically when only viewing one pitcher which is why we expect our models to perform better when analyzing only one pitcher.

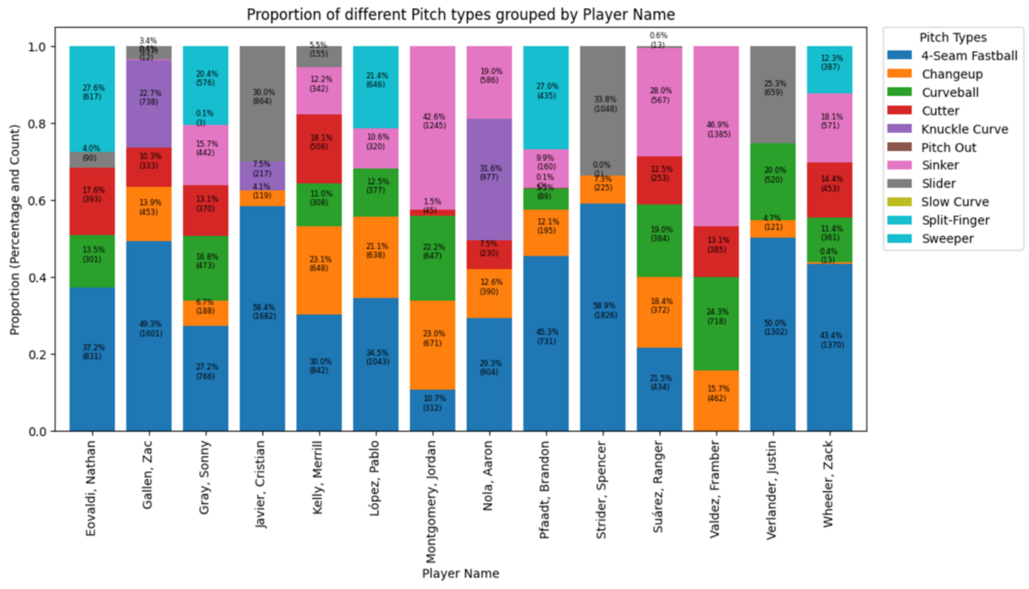

Figure 3: Pitch arsenals for each pitcher in our dataset and how often each pitch is used

The above visualization shows the distribution of pitch types for each pitcher. This visualizes the pitches that each pitcher throws most frequently and the number of pitches that they have in their arsenal.

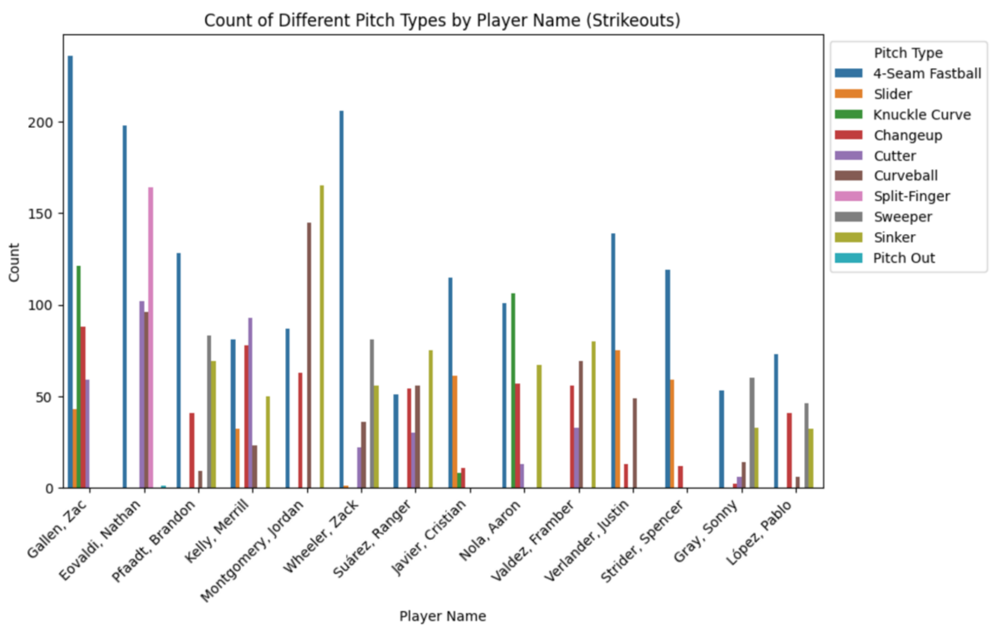

Figure 4: Count of each pitch type for each pitcher on strikeouts

This visualization is important to view because the most important situation to analyze would be with 2 strikes because that is usually when the pitcher is trying to make their best pitch. Each pitcher has an “out” pitch that they feel most confident in. For a situation like that, this would be important for the players and team to understand.

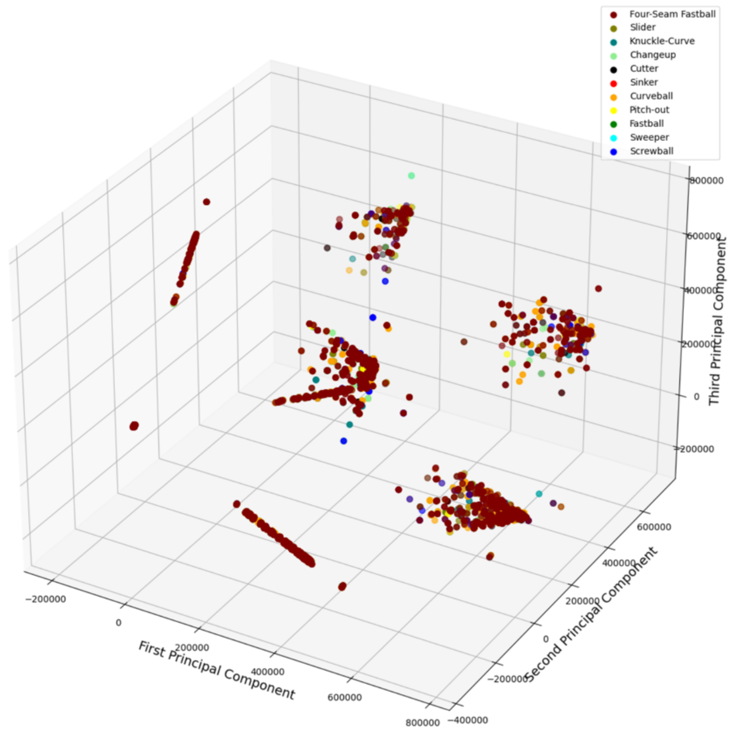

Figure 5: 3-D Principal Component Analysis for our dataset analyzing all the pitch types

Performing Principal Component Analysis on the dataset is important to understand the distribution of the different classes (pitch types) in a reduced dimension. We can see that there are a lot of 4-Seam Fastballs as compared to other pitch types for all the pitchers (as expected) and they do not clearly form a definite cluster (they combine with other pitches nearly each time).

Therefore, our descriptive analysis allowed us to view the data in various ways to better understand it which is always important because once the results are analyzed, it is interesting to see what makes sense and if there are any anomalies.

Models

For our analysis, we tried out a few different models with various fine-tuning parameters for each model. We got a few anticipated results and a few others that we did not expect. The models did take some time to train, and the data points were sufficient since it helped us get a decent sense of the pitch type a pitcher is likely to throw. We used all the 14 pitchers for our initial training & testing of the models. This was about 38,456 rows for training and 4,532 rows for testing.

Multinomial Naïve Bayes

The first model we tried out for our analysis was the multinomial Naïve Bayes. We tried out different parameters (alpha levels) passed into the MultinomialNB model. We used a 5-fold validation through GridSearch which returned us an accuracy score for our training data. We got the best result (among all the alpha values) for an alpha of 0.0001 which was a very slow learning rate. The cross-validation score for this was only 1.4% overall. When we tried this on the test data (we knew it wouldn't give good results but just for numbers, we did) we got a mere 2.6% accuracy. This model did not work well at all. We sort of anticipated this since there were 11 different pitches and 34 columns so the naïve bayes score was expected to be low.

Table 1: Multinomial Naïve Bayes Parameter Tuning Results

Parameter	*Value*	*Cross Validation Score*	*Test Accuracy*	*Precision Score*
alpha	0.0001	*	*	*
	0.001			
	0.01			
	0.1			
	1.0			
Best Scores		**1.41%**	**2.58%**	**0.0034**

Random Forest

The next model we tried out was the random forest classifier. Again, we used the grid search approach to get the best possible parameters for our training data. For this model, we used a 10-fold cross validation technique along with various other parameters that we tried to fine-tune. The scorer we used was accuracy, and the best parameters (that we obtained upon fitting the model to our training dataset) were used on the test dataset to get a sense of how good the model worked on our data. Now, the code only shows a 'max_depth' being an array of 4 values, but a lot of other values were tried out and removed (like 2,3,5,7,9) since running the model again with all the values would take a lot of time. We did a similar thing for other parameters. The best fit we got along with their scores are listed in the table below:

Table 2: Random Forest Classifier Parameter Tuning Results

Parameter	*Value*	*Cross Validation Score*	*Test Accuracy*	*Precision Score*
bootstrap	True	*	*	*
	False			
max_depth	11			
	15	*	*	*
	17			
	19			
min_samples_leaf	5			
	10			
	15			
	20	*	*	*
min_samples_split	10			
	15	*	*	*
	20			
Best Scores		**48.60%**	**44.00%**	**0.3971**

K-Nearest Neighbors

The third model we tried was K-Nearest Neighbors (KNN) classifier. Yet again, we used the grid search approach to get the best possible parameters for our training data using 5-fold cross validation. The scorer we used was accuracy. We experimented with various values of two different parameters (n_neighbors & kind of distance measure). The best fit we got along with their scores are listed in the table below:

Table 3: K Neighbors Classifier Parameter Tuning Results

Parameter	*Value*	*Cross Validation Score*	*Test Accuracy*	*Precision Score*
n_neighbors	2			
	4			
	7			
	11			
	15			
	20	*	*	*
metric	euclidean			
	manhattan	*	*	*
	minkowski			
Best Scores		**30.00%**	**32.88%**	**0.2656**

Support Vector Machine

The final model we tried was Support Vector Machine. Now, for this model, we did not use the usual GridSearchCV function that we used earlier. We tried to experiment with another library called 'Optuna' that works similar to the grid search but is a lot more efficient in trying out different parameter values and finding out the best one among all the possible inputs. This reduced our computation time exponentially since grid search with SVC with the same parameter grid took more than 7 hours to train, and Optuna gave us results in less than an hour. We ran the optimizer for 10 trials and tried to get the best score for each run. The best fit we got along with their scores are listed in the table below:

Table 4: Support Vector Machine Parameter Tuning Results

Parameter	*Value*	*Cross Validation Score*	*Test Accuracy*	*Precision Score*
C	0.0001			
	0.001			
	0.01			
	0.1			
	0			
	1.0			
	10.0			
	100.0			
	1000.0			
	10000.0	*	*	*
gamma	0.0001			
	0.001			
	0.01			
	0.1			
	0.0			
	1.0			
	10.0			
	100.0			
	1000.0	*	*	*
	10000.0			
kernel	linear			
	rbf	*	*	*
	poly			
Best Scores		**35.02%**	**35.02%**	**0.1226**

We also ran all our models individually for each pitcher. For most of the pitchers, the results were worse than everyone together probably because of the small sample sizes for the test set. However, we included three pitchers who had some of the highest results. For example, Aaron Nola got the following results:

- Multinominal Naïve Bayes: 25.5% training accuracy, 28.8% test accuracy, 0.288 recall score
- Random Forest: 33.4% training accuracy, 32.8% test accuracy
- K-Neighbors: Best cross-validation score of 0.19

Conclusion

While our accuracies and numbers aren't great, we needed to also consider a number of outside factors that impacted this data beyond just these numbers, with the largest impact being playoff adrenalin and more consideration and focus with every pitch. With that being said, we tried numerous different combinations of columns, whether that involved adding or removing them, and splitting the data up by pitcher all to try and find the best results. In the end, the random forest ended up being the best model to predict pitch type. Even though the accuracy is not very high, teams can still use this data to point out certain moments in the game where a pitcher is more likely to throw a certain pitch to prepare the hitters in advance. Our final numbers and model parameters are listed below:

Table 5: Results

Model	***Best Parameters***	***Best Cross Validation Score***	***Best Test Accuracy***	***Best Precision Score***
Multinomial Naïve Bayes	'alpha': 0.0001	1.41%	2.58%	0.0034
Random Forest Classifier	**'bootstrap': True, 'max_depth': 15, 'min_samples_leaf': 20, 'min_samples_split': 15**	**48.60%**	**44.00%**	**0.3971**
K Nearest Neighbors Classifier	'metric': 'manhattan', 'n_neighbors': 20	30.00%	32.88%	0.2656
Support Vector Machine	'C': 9178.37, 'gamma': 4423.78, 'kernel': 'rbf'	35.02%	35.02%	0.1226

Results

After fine-tuning our best model (Random Forest Classifier) to our training data, we separated the data for individual pitchers. We planned to predict a pitch type for a particular pitcher based on some sample input and for this, we selected the top 3 pitchers that appeared most in the post season which would help teams formulate better game plans for them. After fitting the random forest model with the best parameters to the individual training data, we predicted a pitch type the pitcher is likely to throw. With a good level of 'correctness', we were able to get different types of pitches for same input values. For example, if we provided a pitch with a certain x & z axis position, innings, outs, etc. we found out that Aaron Nola is most likely to throw a 'Changeup', Zac Gallen would throw a '4-Seam Fastball', while Jordan Montgomery would mostly pitch a 'Cutter'. Our results align somewhat with what was expected, so we could say that our model works well in almost half the situations which is pretty good for teams & players in formulating gameplans against certain pitchers.

Limitations/Further Exploration

Considering how difficult of a task this was, there are limitations that come with our data/analysis. First, there are many outside factors that are not definable or quantifiable that affect a pitcher's decision such as adrenaline or other physical factors on their start day. Second, considering the amount of variability within this space and the number of different pitches for each pitcher, attaining high accuracy levels is very challenging. Therefore, we are satisfied with our results. Furthermore, our test set consists of postseason data which could potentially affect the pitcher's game plan considering the importance of the game. With more time, we could've better analyzed each in-game situation and concluded which ones are easiest to predict. Those results would be useful for teams and players.

References

(Aswani, 2023)

(Stacked Percentage Bar Plot In MatPlotLib, n.d.)

(Rotate tick labels for seaborn barplot, n.d.)

(Move legend outside figure in seaborn tsplot, n.d.)

(Choosing color palettes, n.d.)

(Pandas: How to Find Unique Values in a Column, n.d.)