Predicting Pitch Type – FastBall vs Offspeed

1. Problem Statement

Strategy in Major League Baseball is ever involving. With hitters increasingly focused on launch angles and elevating the ball, it is unsurprising that teams are becoming more reliant on home runs to score. In fact, in 2019 the teams collectively hit an eye-popping 6,776 home runs, which is roughly a 10% increase over the previous record of 6,105 home runs hit in 2017. This trend is very likely to continue in the coming seasons.

It also is generally accepted that fastballs are particularly ripe for batters to hit for home runs given their increased velocity and lack of movement relative to other pitch types. Indeed, Fantasy Labs, a website focused on providing strategic advice to fantasy sports players, noted that fastballs were the most common pitch type to be hit for home runs. (https://www.fantasylabs.com/articles/home-run-trends-part-3-pitch-type/). And simple common sense dictates that a batter will have an even greater chance of hitting a home if he knows that the pitcher is going to throw a fastball - one only needs to watch batting practice or the home run derby to see this confirmed.

The goal of this project is to build a model that attempts to accurately predict when a pitcher will throw a fastball. The targeted clients for this model are professional, semi-professional, and collegiate baseball teams, including the players, managers, and other personnel. The organizations could use this data to inform their batters to anticipate fastballs under certain circumstances, which in turn should lead to the team hitting more home runs. Conversely, organizations could use this information to inform their pitchers as to when fastballs are most often thrown and coach the pitchers to throw a different pitch under those circumstances. Professional sports gamblers could also use the model to inform their wagers. These are just a handful of what is likely many uses for an accurate fastball prediction model.

II. The Datasets

This project uses the MLB Pitch Data 2015-2018 dataset that is publicly available on Kaggle at. I have chosen to build my model using two .csv files from that dataset. The first file, pitches.csv, charts various data for each pitch thrown during each of the four seasons from 2015 through 2018. The second file, atbats.csv, contains various static data for each at-bat from each of the four seasons from 2015 through 2018. The two files have the following dimensions:

pitches.csv == 2,870,000 x 40; and
atbats.csv == 740,000 by 11.

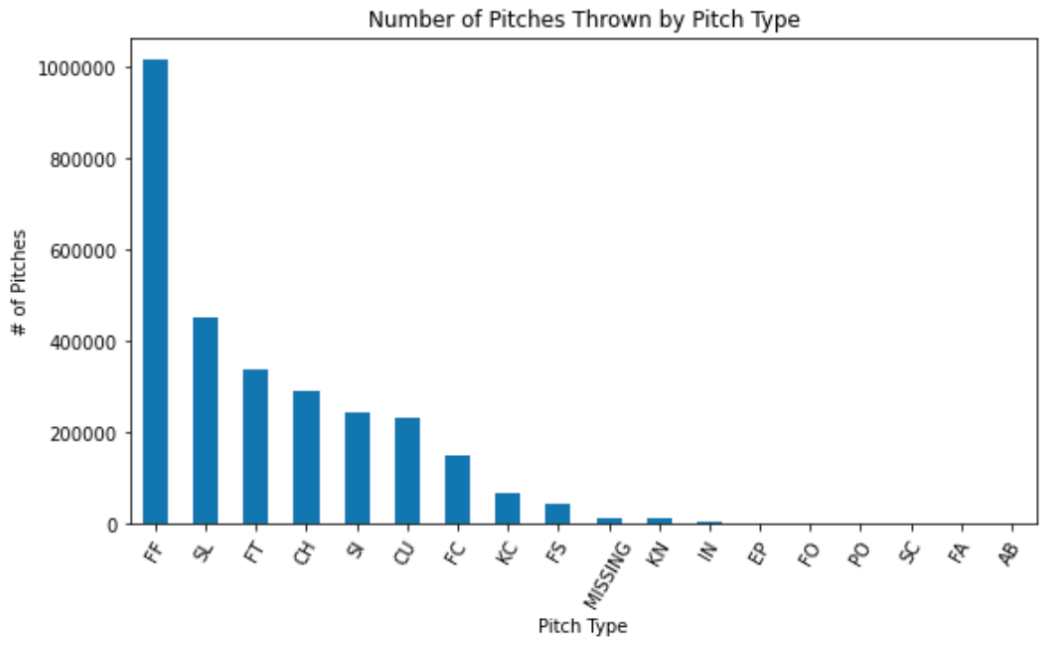
One of the categorical variables in the pitches file, pitch_type, classifies the pitch type for each pitch thrown. The variable contains over fifteen distinct labels, including FF for a four-seam fastball, FT for a two-seam fastball, and FC for a cut-fastball. The dataset also has a label for split-fingered fastballs, but we will treat that category as a non-fastball for the purpose of this project because the pitch behaves and is used more similarly, to other breaking balls (such as a sinker). I will be able to use these labels to create a new binary variable to classify each of these three types of fastballs as a fastballs and the remaining categories of pitches as non-fastballs, which will become the target of my analysis.

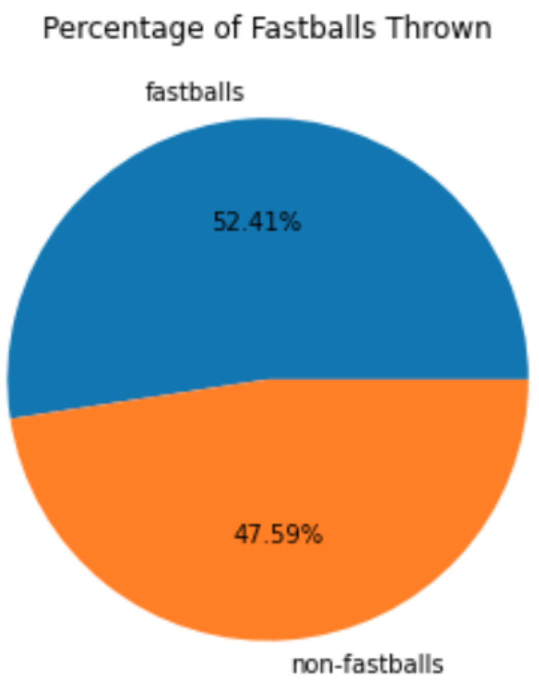
III. Exploratory Data Analysis

Pre-Pitch Categorical Variables

The second type of features in the dataset are pre-pitch categorical variables. These are more important for the building of my model because, unlike the post-release continuous variables discussed above; they are known prior to the pitch being thrown. Some examples of these circumstances are runners on base, the batting count, the game score, the pitch number in the at-bat, the handedness of the pitcher and batter, and the number of outs.

In order to accurately evaluate the various categorical variables, I first needed to identify a default baseline percentage regarding how often a pitcher throws a fastball. This can be done by calculating the percentage of pitches that are labeled as a fastball in the entire dataset:

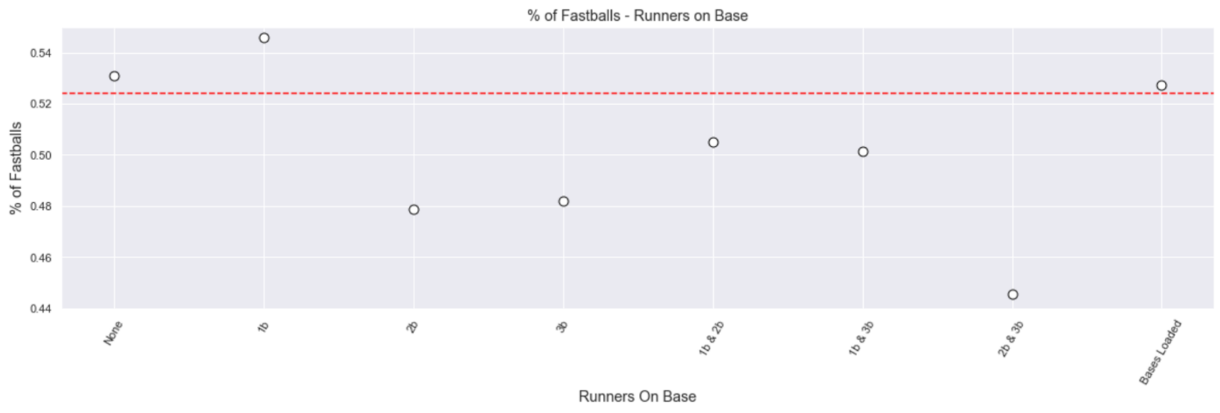




As we can see, a pitcher threw a four-seam, two-seam, or cut-fastball approximately 52.41% of the time. I can use this number as a baseline when analyzing the various categorical data variables. A variable is not overly significant to the extent that it correlates to a fastball usage at or around that same 52.41% frequency - the goal is to identify the circumstances in which that frequency increases (or decreases) materially from that baseline number.

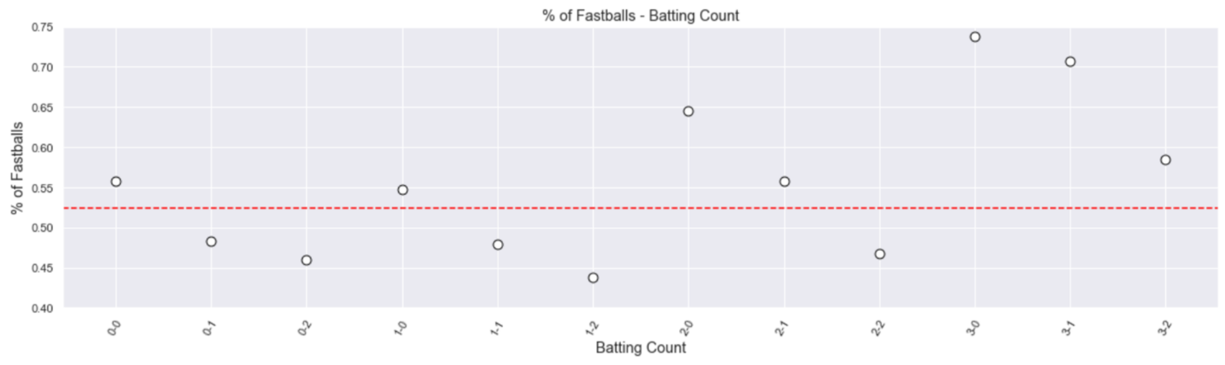
Runners on Base

The most significant influence on fastball usage is whether there is at least one runner in scoring position (on second or third base). Interestingly, however, the fastball usage spikes back up to above average when the bases are loaded, likely because a walk in those circumstances result in a run scored.



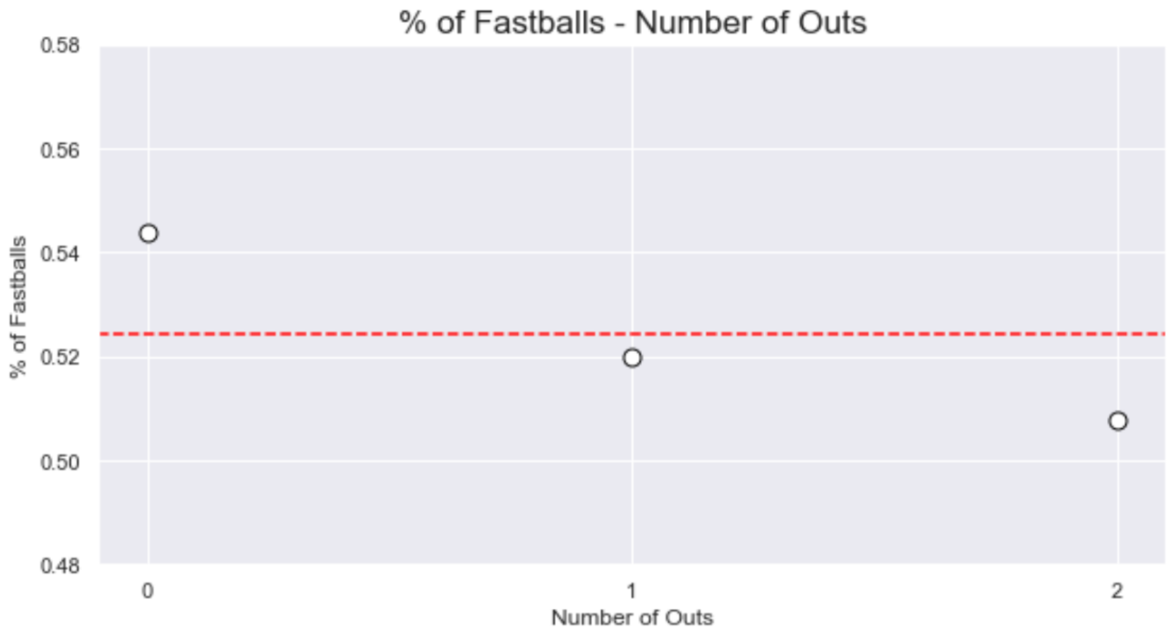
Batting Count

A pitcher's fastball usage decreases when he is ahead in the count and increases when he is behind in the count (likely because he needs to try to avoid walking the batter).



Number of Outs

Fastball usage decreases depending upon the number of outs.

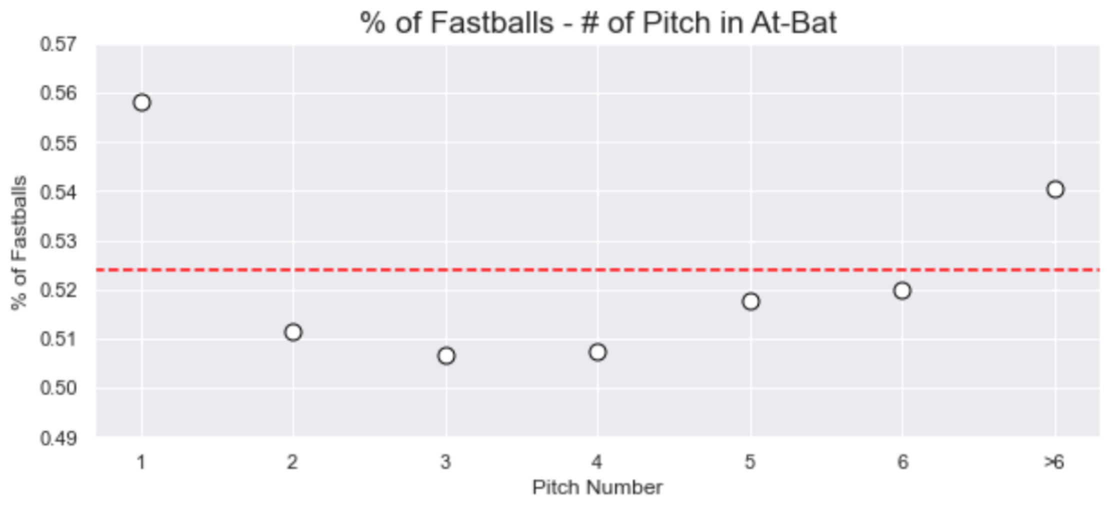


Pitch Sequence

Pitchers throw more than average amounts of fastballs by a considerable amount on the first pitch of an at-bat and any pitch after the sixth pitch in a long at-bat.

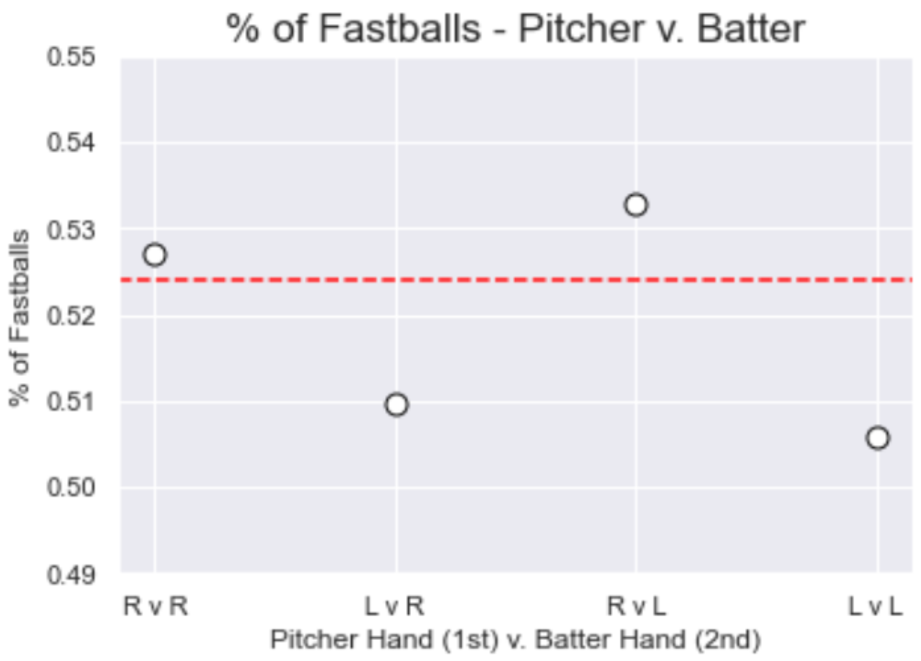
Inning

Fastballs are thrown higher than average in the early innings and the ninth inning. The middle innings appear to be when the least number of fastballs are thrown.

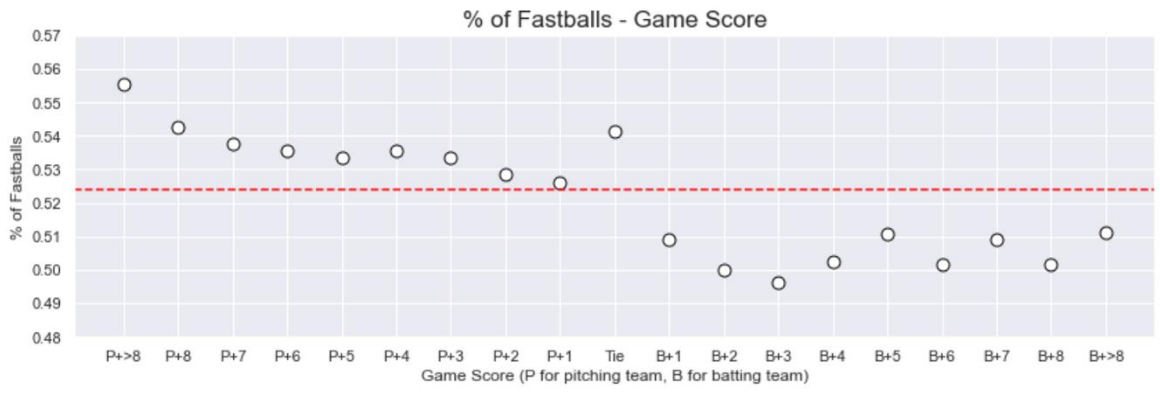


Pitcher/Batter Dominant Hand

Right-handed pitchers throw fastballs more often than left-handed pitchers regardless of the handedness of the batter.



Game Score
Pitchers throw more fastballs than usual when their team is winning, or the game is tied and significantly less fastballs when their team is behind.



In summary, it certainly appears as though some correlation exists between certain game circumstances and the likelihood that a pitcher will throw a fastball on the next pitch. I therefore am hopeful that a model can predict when a fastball will be thrown at a rate higher than the default 52.41 percent.

IV. Initial Machine Learning Models Test Run

At this point in the project, I decided to feed the dataset into four classifiers to see how they would perform. Because my dataset has over 2.8 million samples and would be a significant time investment to train the models on the full dataset, I elected to feed the classifiers approximately 5% of the data (145,000 pitches) and confirmed that the sample data had the same 52/48% split between fastballs and non-fastballs.

I also used only the pre-pitch categorical features discussed in the previous section. Including the post-release continuous features would no doubt dramatically improve model performance, but would defeat the purpose of this exercise by using information not available to the batter prior to the pitch being thrown.

The initial models yielded the following accuracy on the test set from the sample data:
- Logistic Regression Classifier: 0.5572
- SGDClassifier: 0.5442
- Random Forest Classifier: 0. 5280
- Gradient Boosting Classifier: 0. 5510

The results were not encouraging. The classifiers performed only marginally better than the default 52.41% accuracy that you could achieve simply by guessing fastball every time. So, it appeared at least on the surface that Major League Baseball pitchers were fairly adept at mixing up their pitch selection in the various game circumstances that I fed into my models.

I had not, however, fed any information into the models regarding the previous pitch thrown by the pitcher. That information would be known by the batter and was available in my dataset, so I decided to add some additional features to the dataset fed into the models.

V. Additional Feature Engineering

I was fairly confident that information about the previous pitch in a pitch sequence should improve the accuracy of my models. The major obstacle to this approach, however, was that building previous pitch features would involve having to iterate over the samples to include the data only when relevant (i.e., only to pitches that are not the first pitch of a particular at-bat), which was very time-consuming and computationally expensive given that there are over 2.8 million pitches in my dataset. I ultimately decided to add the following features:

● Inning Pitch Count - this feature provides how many pitches a pitcher has thrown in a particular inning. I could only tally this efficiently on an inning-by-inning basis (rather than a pitch count for the entire game) given how the data was aggregated in my dataset.

● Previous Pitch Type - this feature provides the pitch type of the immediate previous pitch thrown to the batter using the pitch type labels (Once this data was compiled, I converted it to categorical features using dummy variables)

● Vertical/Horizontal Previous Pitch Location - these two features provide the vertical and horizontal pitch location of the immediate previous pitch to a batter in a particular at-bat. These were built off of the ‘px’ and ‘pz’ continuous variables discussed above.

I ended up settling on using only the previous pitch type and location features due to how computationally expensive it was to build these features over a 2.8 million sample dataset. Had I not been under such a time constraint, I would have built a previous pitch feature for each of the continuous variables discussed in the exploratory data analysis section of this report. I ultimately chose to simply use the previous pitch type label knowing that the particular label somewhat aggregates that data into one feature.

VI. Second Machine Learning Models Test Run

Having added the previous pitch features to the dataset, I fed the updated 5% sample data into some models to see if I had improved performance. Here are the resulting test set accuracy numbers for each model:

- Logistic Regression Classifier: 0.601517
- SGDClassifier: 0.591917
- Random Forest Classifier: 0.569545
- Gradient Boosting Classifier: 0.601848

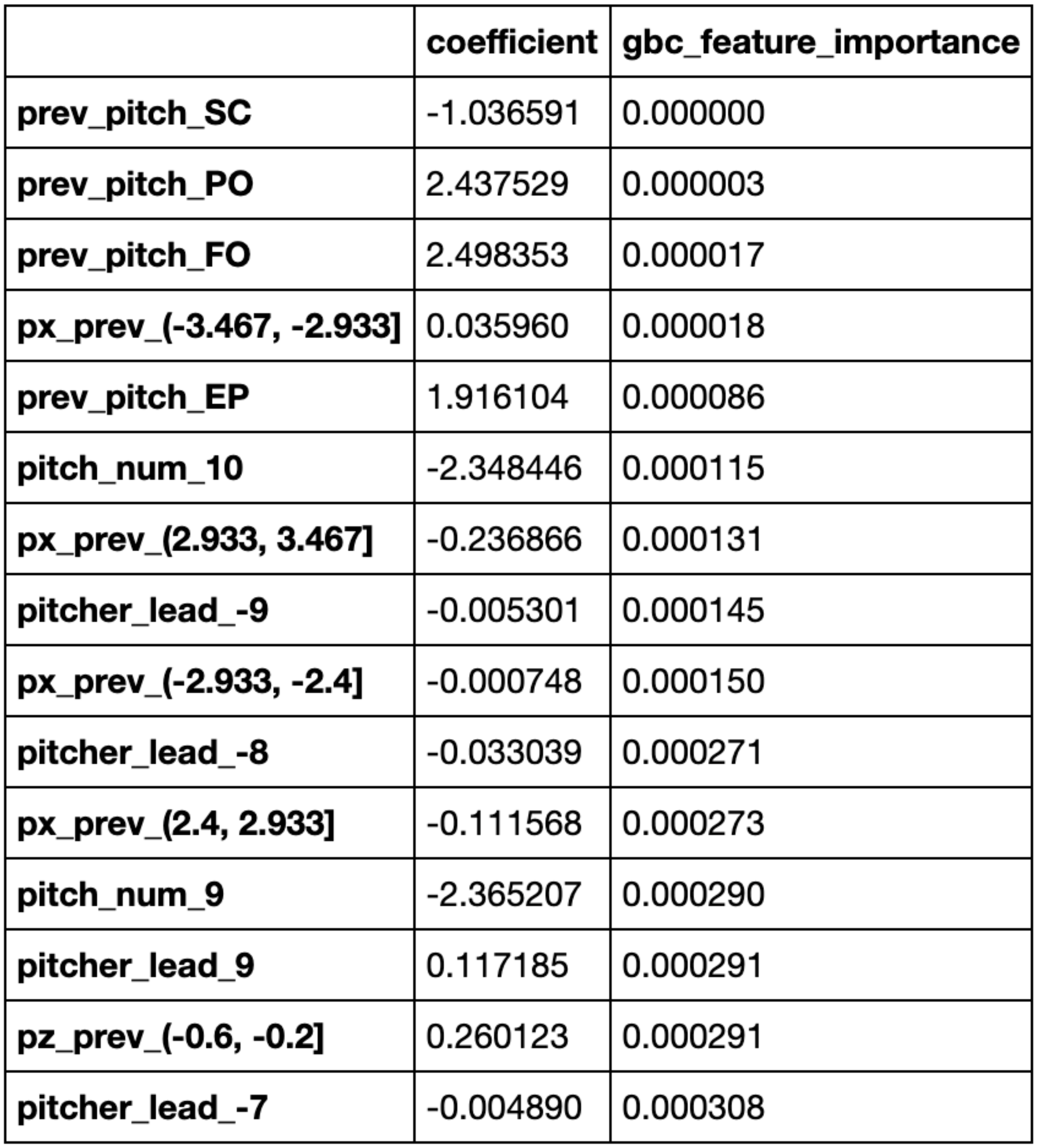
As we can see, adding features regarding the previous pitch in a pitch sequence significantly improved models. Given that the Logistic Regression and Gradient Boosting Classifiers were able to produce an accuracy of over 0.6, I elected to use those two models going forward.

Gradient Boosting Classifier Feature Importance

The gradient boosting classifier only produces positive "feature importance" values to each model feature on a normalized (0-1) scale.

These values will indicate to us which features are most important to the model in making its key decisions within the decision trees, with a higher value indicating that the feature is more influential. Here are the 15 most (on the top) and least (on the bottom) features in the model:

	coefficient	gbc_feature_importance
prev_pitch_SI	0.231926	0.379442
prev_pitch_FF	2.742564	0.080825
b_count_3	0.724995	0.057462
s_count_2	-0.937326	0.054237
prev_pitch_FT	2.808360	0.043790
prev_pitch_FC	3.038671	0.037677
inning_pitch_count	-0.094548	0.029650
s_count_1	-0.600940	0.021992
b_count_1	-0.199580	0.020831
prev_pitch_KN	0.085609	0.019515
on_2b	-0.131415	0.015927
pitch_num_2	-2.442173	0.015921
prev_pitch_SL	2.232262	0.014552
b_count_2	0.051795	0.014136
prev_pitch_IN	-6.606109	0.012344



	coefficient	gbc_feature_importance
prev_pitch_SC	-1.036591	0.000000
prev_pitch_PO	2.437529	0.000003
prev_pitch_FO	2.498353	0.000017
px_prev_(-3.467, -2.933]	0.035960	0.000018
prev_pitch_EP	1.916104	0.000086
pitch_num_10	-2.348446	0.000115
px_prev_(2.933, 3.467]	-0.236866	0.000131
pitcher_lead_-9	-0.005301	0.000145
px_prev_(-2.933, -2.4]	-0.000748	0.000150
pitcher_lead_-8	-0.033039	0.000271
px_prev_(2.4, 2.933]	-0.111568	0.000273
pitch_num_9	-2.365207	0.000290
pitcher_lead_9	0.117185	0.000291
pz_prev_(-0.6, -0.2]	0.260123	0.000291
pitcher_lead_-7	-0.004890	0.000308

In analyzing the features that the Gradient Boosting Classifier assigned the least amount of importance, it becomes clear that most if not all of the features tend to be outlier values. Among the features are previous pitch types that are rarely thrown (screwballs, pitchouts), game scores in which the pitcher's team is winning (or losing) by a wide margin, extreme previous pitch location values, and the last pitches of abnormally long at-bats (9 or greater). It may make sense to consider dropping these outlier features in future iterations of this model, though I would not expect that to significantly impact the model's overall performance given the relative infrequency of these features in the dataset and the low importance the model assigns to them.

IX. Summary/Conclusions

This project demonstrates that it is possible to analyze pre-pitch circumstances to predict whether a pitcher is going to throw a fastball as the next pitch at a higher accuracy rate than the default frequency percentage of 52.41%. It is important to note that the models struggled to outperform the default rate without information regarding the previous pitch despite observing correlations between fastball frequency and certain of these circumstances (runners on base, batting count, pitch sequence, game score, etc.). It is interesting to note that, while correlations between certain game circumstances and fastball frequency do exist, the same overall game circumstances did not significantly improve the algorithms' ability to predict fastballs when considering all of that data in the aggregate.

Once we included information about the pitcher's inning pitch count, the immediate previous pitch type thrown, and the location of the immediate previous pitch, the models' respective accuracy improved significantly by approximately 5-8%. Teams therefore should be able to use one of the final models to correctly identify when a pitcher will throw a fastball or non-fastball more often than competitors not using such a model.

The Gradient Boosting Classifier should be used for most circumstances given its overall superior performance. As explained in detail above, the Gradient Boosting Classifier performed substantially better in identifying non-fastballs and generated higher accuracy numbers on most of the model features. It also appears to have more potential to improve through future work.

That being said, a user should consider using the Logistic Regression Classifier under circumstances in which the user is concerned only (or primarily) with identifying when a fastball will be thrown and is less concerned about mis-identifying non-fastballs as fastballs.

X. Future Work for Potential Model Improvements

I believe that I potentially could improve the performance of the final models generated by this project with some additional work. The main hurdles that I encountered with this project was the large size the data (over 2.8 million pitches) and the limited computational power and time available. With more time and access to more computational power, I think the models could potentially be improved through exploration of one or more of the following:

● Adding more previous pitch data as additional features. I was only able to add certain previous pitch features to the dataset given how much time/computational power it took to build them. If I had more time and computational power, I would suggest adding the other previous pitch data available for the immediate previous pitch (pitch speed, break length, spin rate, etc.). I also would suggest adding the same information for each previous pitch for an entire at-bat (rather than only the data from the immediate previous pitch). I believe this has the potential to improve the model performance given how significantly the previous pitch features improved my model in this project.

● Tuning the hyperparameters with a larger portion of the data. I only used approximately 5% of the data to tune my model hyperparameters given my time and computational power constraints.