1. Problem Statement

- 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.

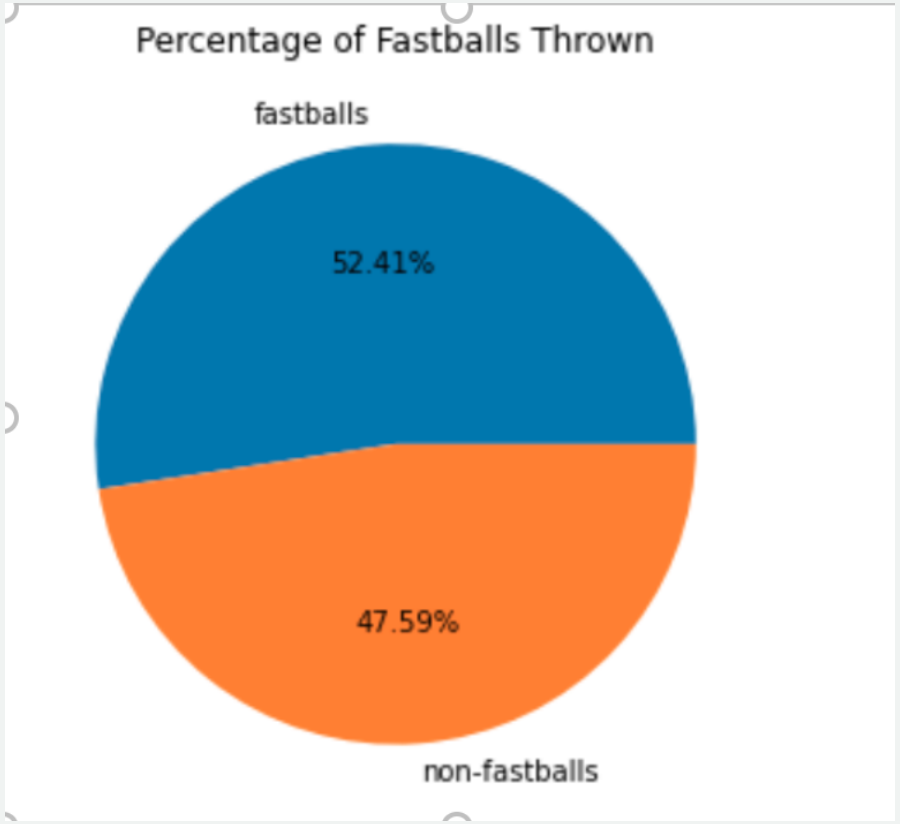
II. The Datasets

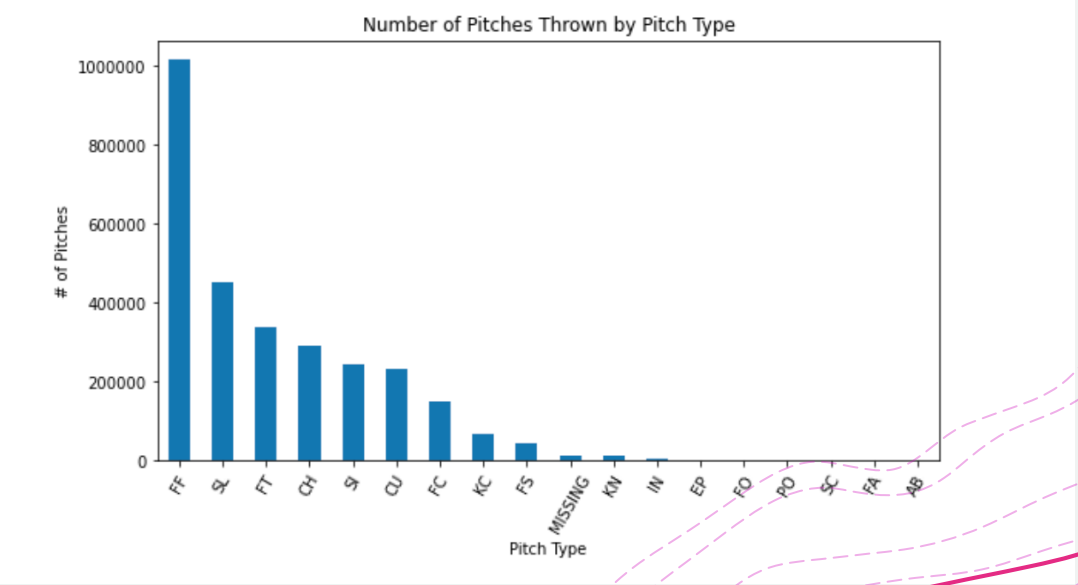
- This project uses the MLB Pitch Data 2015-2018 dataset that is publicly available on
- The first file, pitches.csv, charts various data for each pitch thrown during each of the four seasons from 2015 through
- The second file, atbats.csv, contains various static data for each at-bat from each of the four seasons from 2015 through
- The dataset
- create a new binary variable to classify each of these three types of fastballs as a

III. Exploratory Data Analysis

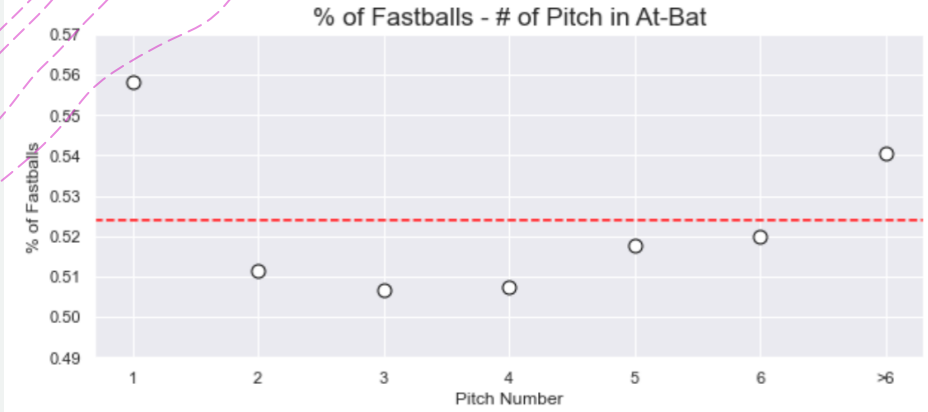
Pre-Pitch Categorical Variables

- The second type of features in the dataset are pre-pitch categorical variables
- continuous variables discussed above; they are known prior to the pitch being thrown
- categorical data variables
- The most significant influence on fastball usage is whether there is at least one runner
- Fastball usage decreases depending upon the number of outs





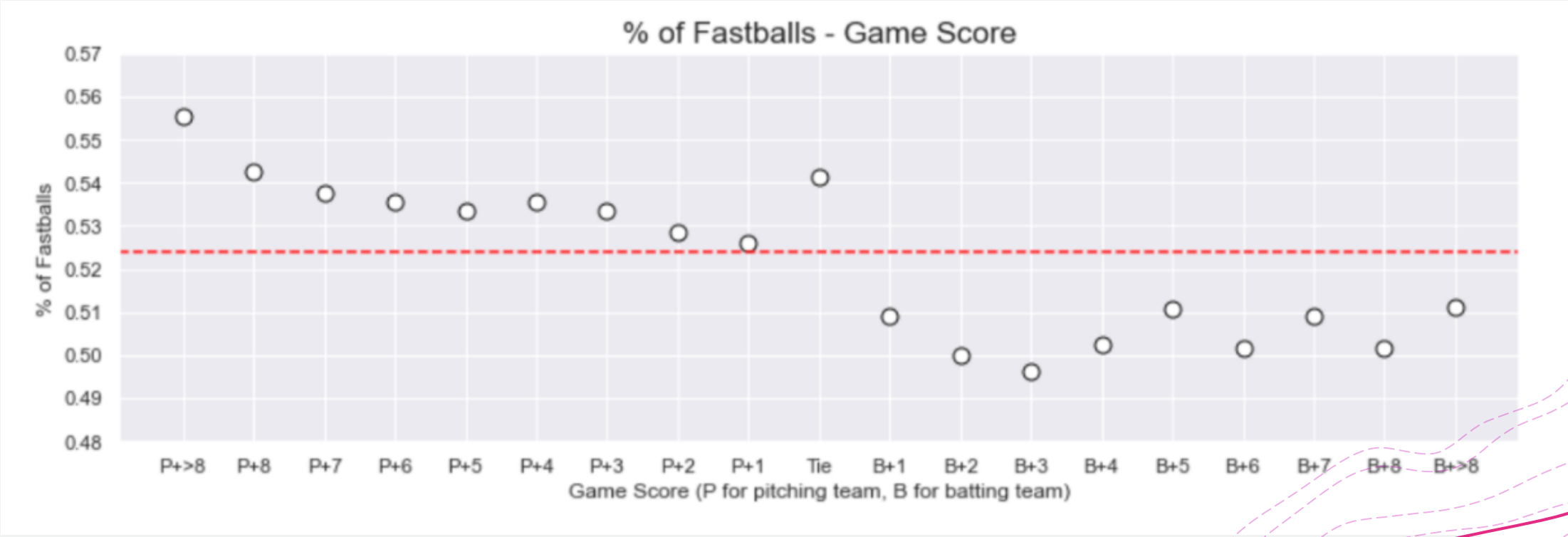
Pitch Sequence

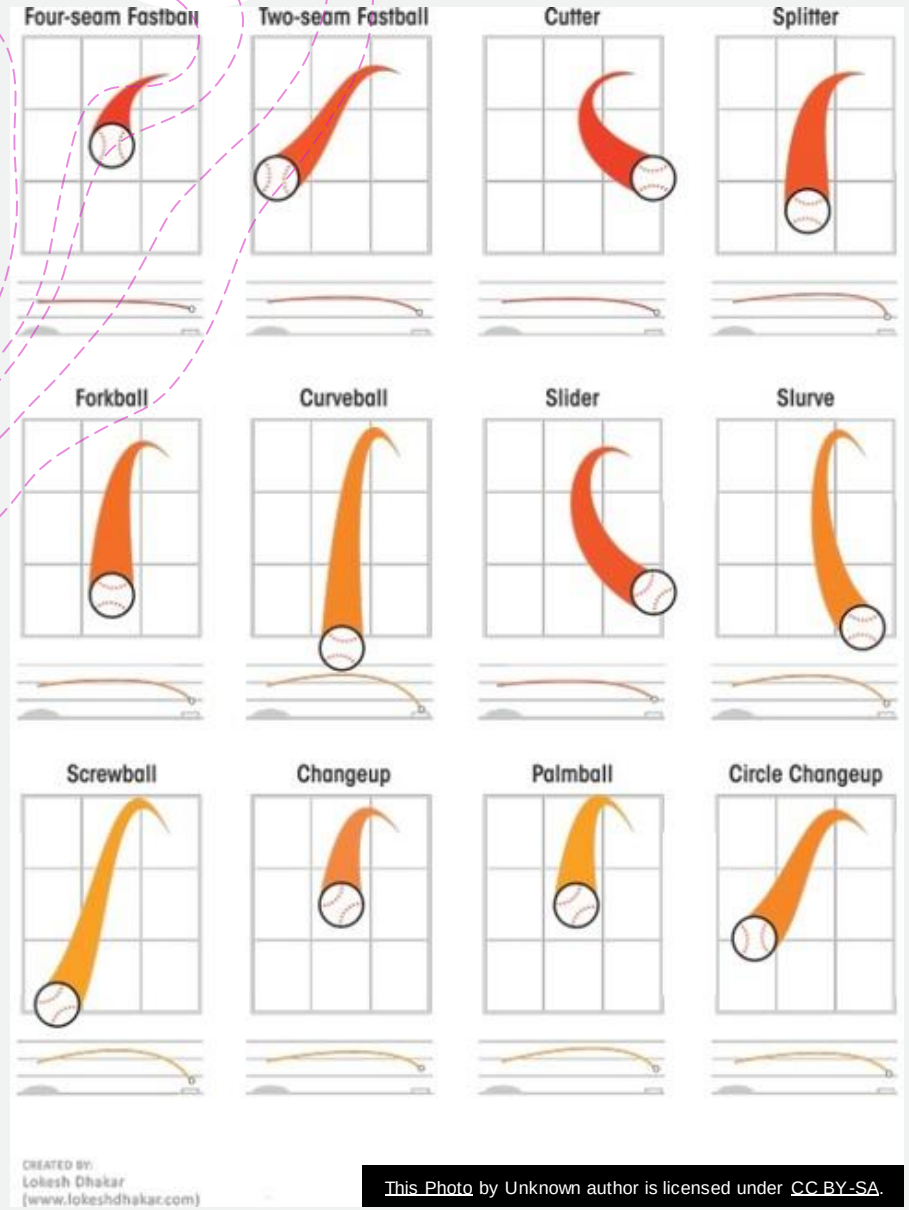


+ Pitchers throw more than average amounts of fastballs by a considerable amount on the
+ first pitch of an at-bat and any pitch after the sixth pitch in a long at-bat
+ Fastballs are thrown higher than average in the early innings and the ninth inning
+ Right-handed pitchers throw fastballs more often than left-handed pitchers regardless of

Game Score

+ Pitchers throw more fastballs than usual when their team is winning, or the game is tied
+ significantly less fastballs when their team is behind
+ game circumstances and the likelihood that a pitcher will throw a fastball on the next



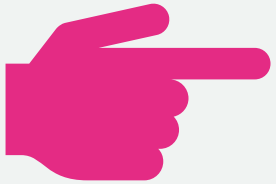


IV. Initial Machine Learning Models Test Run

+ At this point in the project, I decided to feed the dataset into four classifiers to see how they would perform
+ I elected to feed the classifiers approximately 5% of the data. 145,000 pitches
+ 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

V. Additional Feature Engineering

● Inning Pitch Count - this feature provides how many pitches a pitcher has thrown in a particular inning. I could only tally this efficiently on aninning-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.

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 significantly improved models
+ 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

Gradient Boosting Classifier Feature Importance

+ The gradient boosting classifier only produces positive “feature importance” values to each model feature on a normalized scale. These values will indicate to us which features are most important to 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

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%.
- 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.

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 like:
  - 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
  - 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.