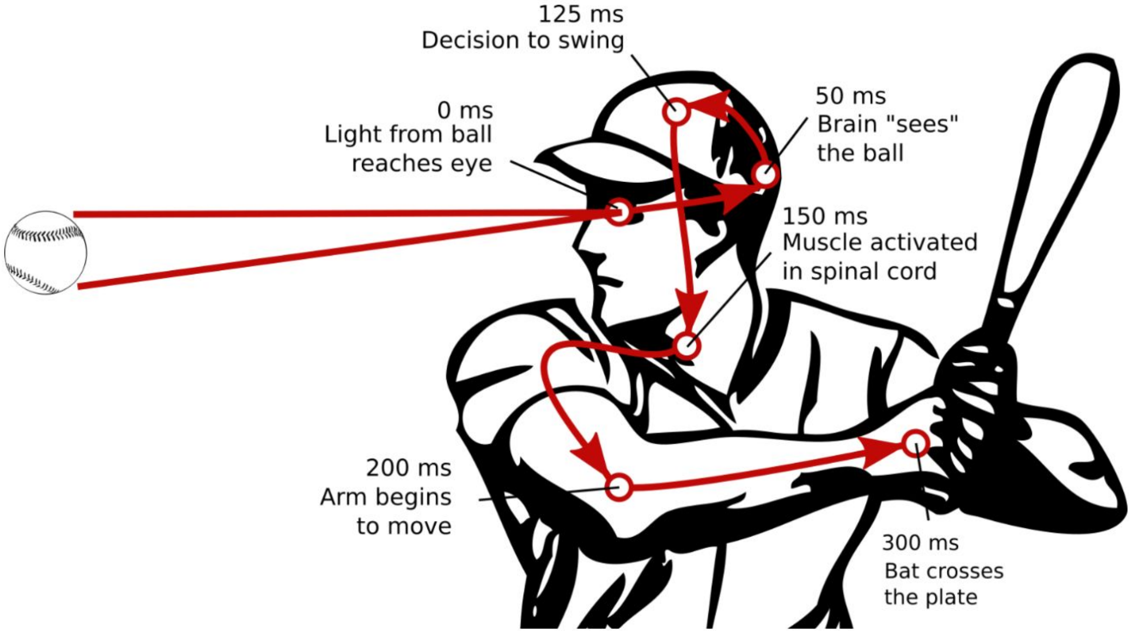
Baseball Pitch Prediction

Albert Cochrane

Why does it matter?

A ball traveling at 100 mph reaches home plate in four tenths of a second (400 ms).



Project Objectives

- Can one could use machine learning to predict what pitch would be thrown, without having to cheat?

- It is fine for a team to give signs to a hitter before a pitch is thrown (i.e. to take the pitch, swing, etc).

- The cheating occurred during the process of actually stealing the signs used by the pitcher and catcher.

Data Overview

We used a Kaggle dataset comprised of 4 csv files, each containing data on a separate aspect of the game.

- At bats
- Pitches
- Players
- Games

The data was originally scraped from the official MLB website

- The majority of the data wrangling was spent filtering and merging the features present in each file that would be relevant to our study.

Exploratory Data Analysis

FF - Four Seam Fastball

SL - Slider

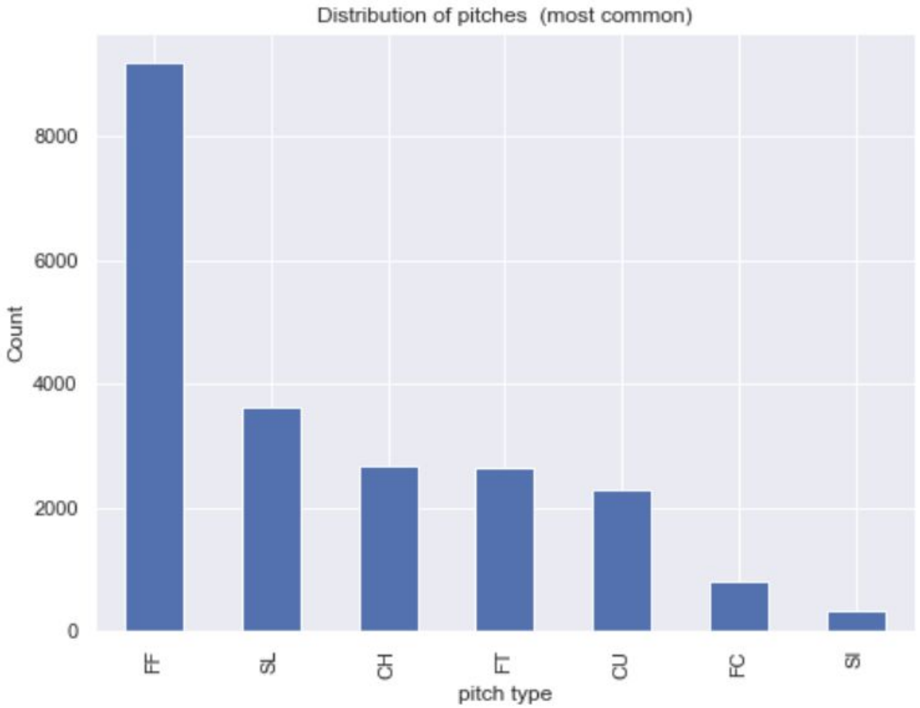
CH - Changeup

FT - Two Seam Fastball

CU - Curveball

FC - Cut Fastball (Cutter)

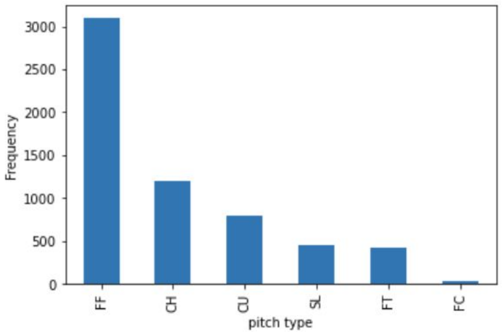
SI - Sinker



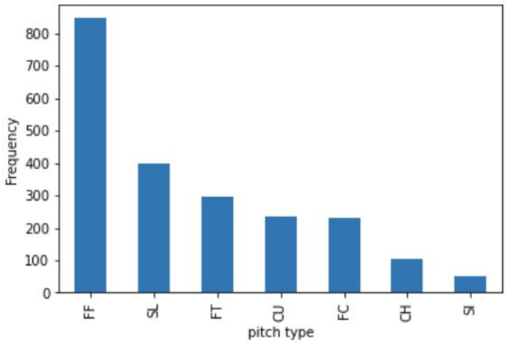
Pitcher/Batter Stance Matchups

The left letter indicates the pitchers dominant hand, the right letter indicates the batters'. R- Right Handed, L-Left Handed

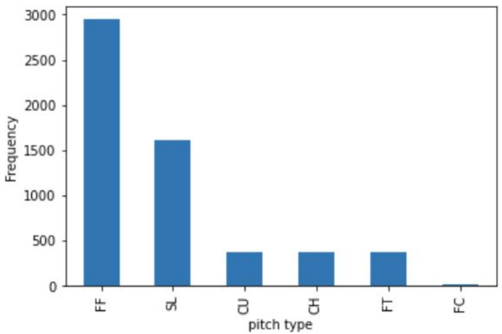
R L



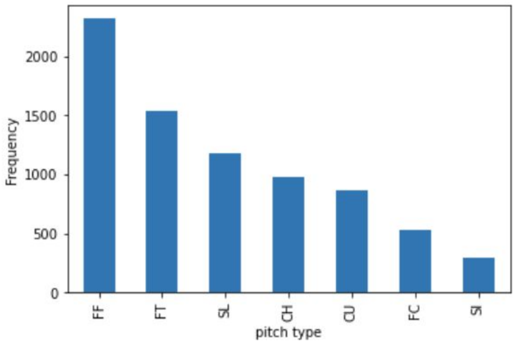
L L



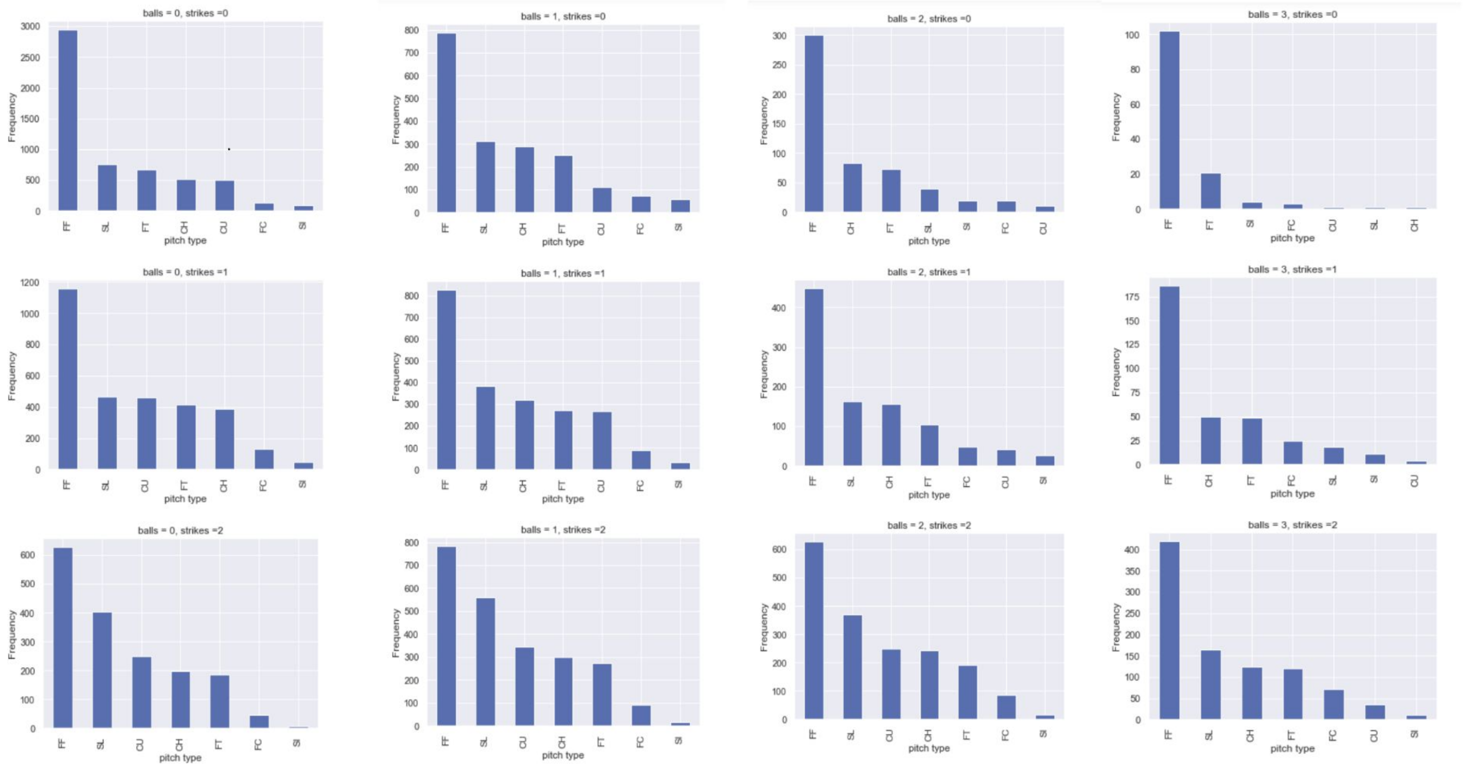
R R



L R



Impact of Balls/Strikes count



Feature Selection

- Any information going into the classifier would be openly available to a hitter/hitting team before a pitch was thrown.

- This eliminated any of the kinetic measures of the actual pitch such as velocity, spin rate, and break angle.

- Inning
- Outs
- Pitcher’s Stance
- Batter’s Stance
- Balls
- Strikes
- Pitch Count
- Score
- Runners on Base

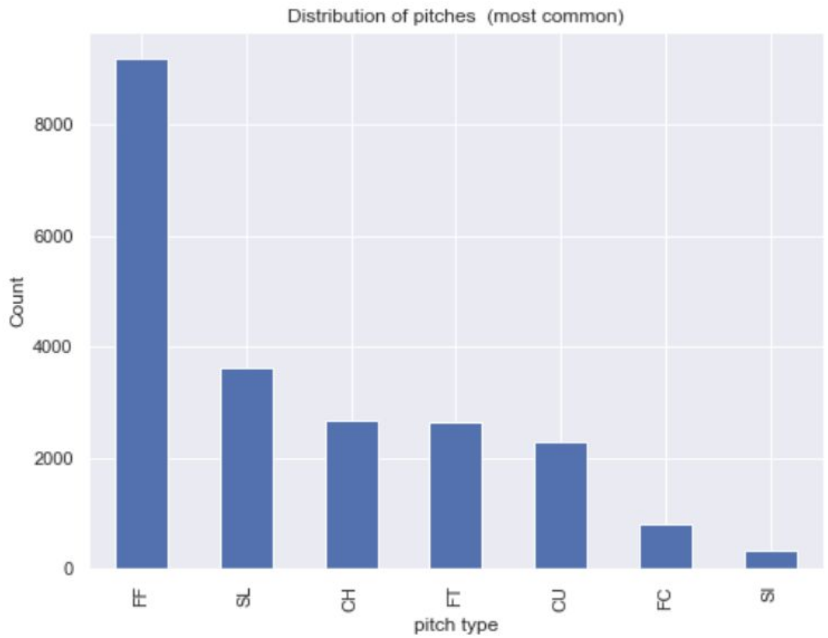
Modeling

To make the actual predictions, we used these 5 classifiers from the Scikit-learn library

1. Dummy (Most Frequent)
2. Logistic Regression
3. Decision Tree
4. Random Forest
5. Gradient Boosting

Creating a Baseline

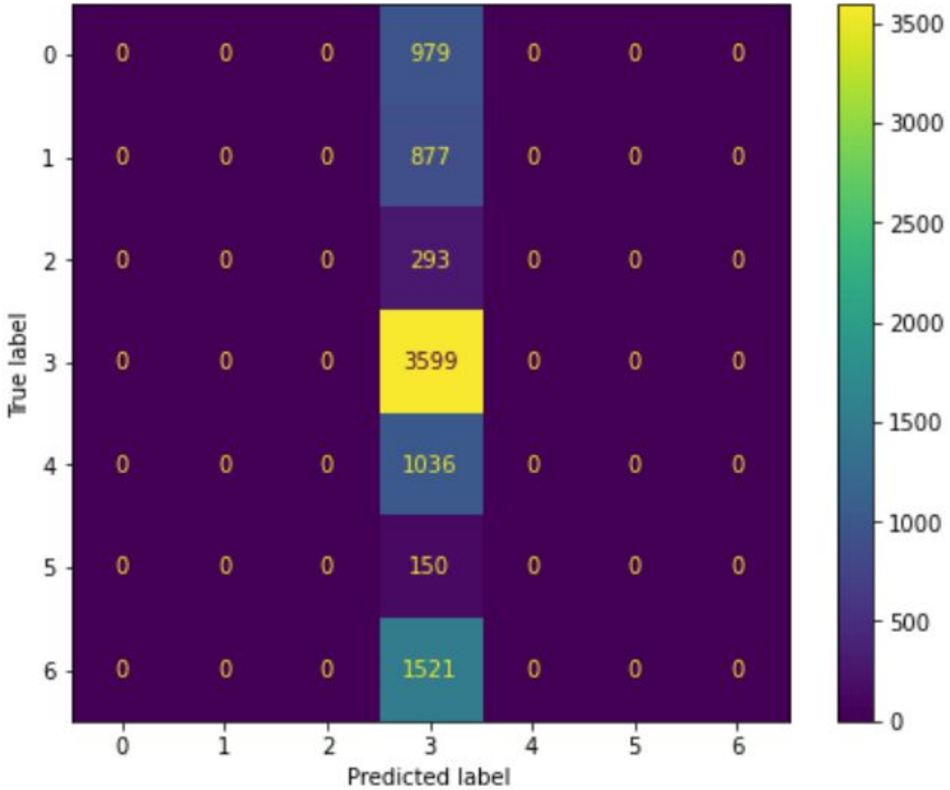
To create a baseline metric to build from, we used a dummy classifier set to predict the most common pitch every time. (4 seam fastball)



Dummy Classifier

Accuracy: 42.6%

```
DummyClassifier(strategy='most_frequent')
accuracy  0.42566528681253696
Recall  0.42566528681253696
Precision  0.18119093639719935
f1  0.25418439808168586
AUC-ROC-Score  0.5
```



Modeling Performance Summary

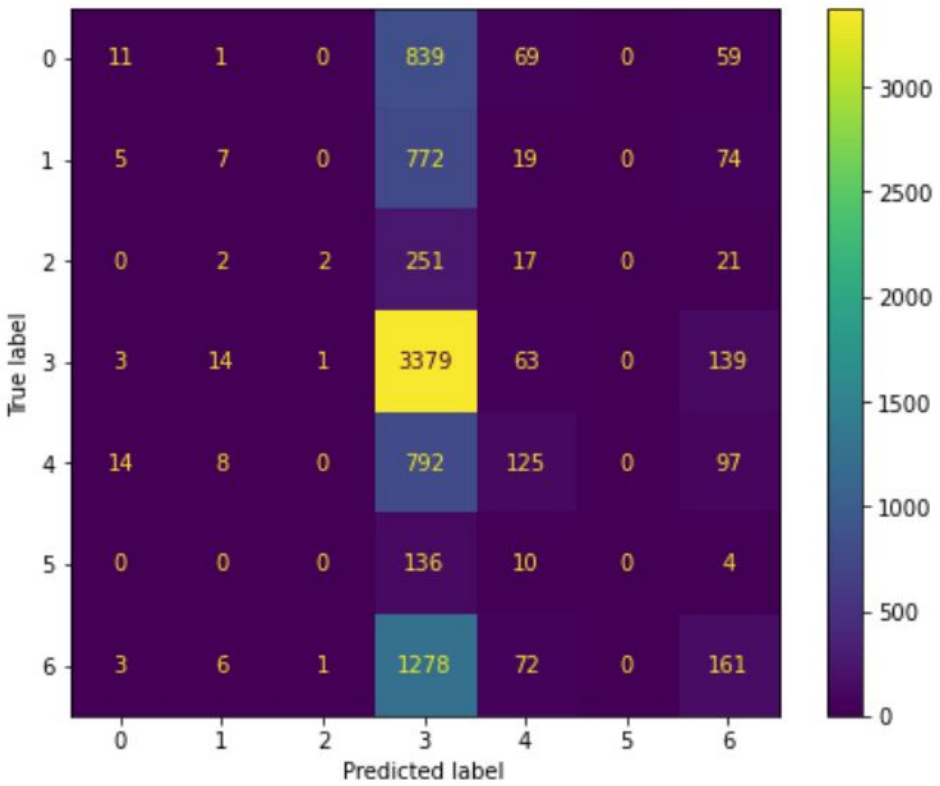
Accuracy Scores of the 5 Classifiers

1.	Dummy (Most Frequent)	42.6 %
2.	Logistic Regression	41.7 %
3.	Decision Tree	32.7 %
4.	Random Forest	36.4 %
5.	Gradient Boosting	43.6 %

Gradient Boosting Classifier

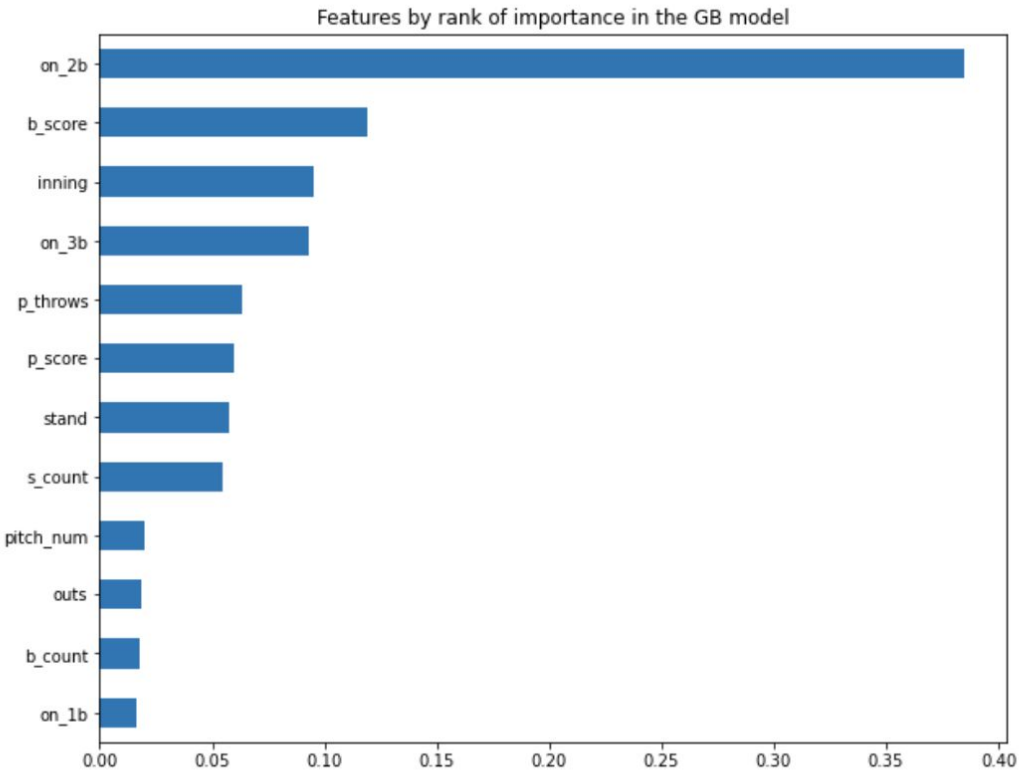
Accuracy: 43.6%

```
GradientBoostingClassifier()
accuracy  0.4358367829686576
Recall  0.4358367829686576
Precision  0.35798477594163086
f1  0.31460025947998804
AUC-ROC-Score  0.7283790206471588
```



Gradient Boosting Classifier

Which features had the most impact on its predictions?



Complete Model Metrics

Metrics used weighted averages

	model	accuracy	recall	precision	f1_score	roc_auc_score
0	DummyClassifier(strategy='most_frequent')	0.426321	0.426321	0.181749	0.254851	0.500000
1	LogisticRegression()	0.416821	0.416821	0.259054	0.277740	0.677615
2	DecisionTreeClassifier()	0.327386	0.327386	0.322496	0.324071	0.547766
3	RandomForestClassifier()	0.364458	0.364458	0.324882	0.337940	0.644192
4	GradientBoostingClassifier()	0.437210	0.437210	0.357987	0.314837	0.722891

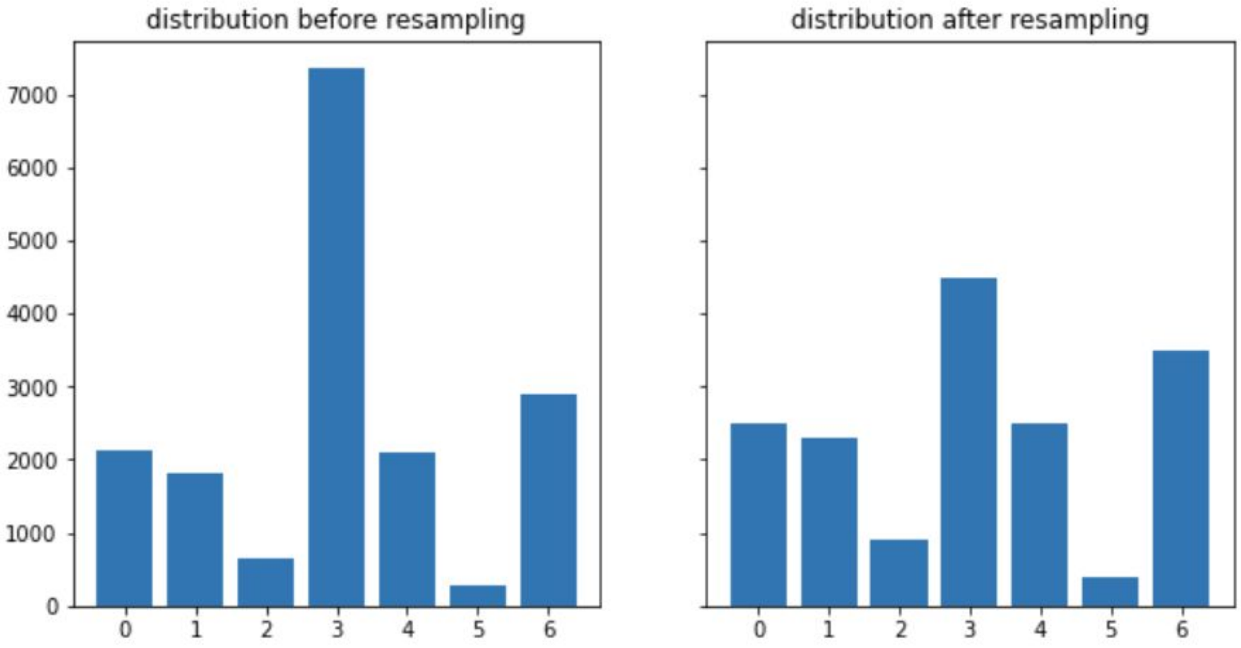
The metric that matters here is accuracy score.
The other metrics are useful to distinguish the models and show their final distribution tendencies.

Model Tuning

- Hyperparameter tuning in this project was pursued but ultimately yielded no significant impact on our metrics.
- Different sampling techniques were also employed due to the imbalanced classes

- Because none of the initial models showed promising metrics, in a practical setting it would probably be more wise to consider changing the scenario/question being asked in the first place.

SMOTE & Undersampling



Despite multiple attempts using this approach, the performance actually declined from the base models.

Potential Future Improvements

- Ideally, every at bat and game should almost be treated like a time series, because pitches don’t happen in a vacuum.
- What a pitcher throws is influenced by what pitches the hitter has already seen, both in that game and in previous encounters.
- Trying to evaluate pitchers in the aggregate was really only done for the sake of computational benefit.
- It would make more sense to build a model for each individual pitcher, potentially adding levels of specificity a priori (individual matchups, righties vs lefties etc).

Final Remarks

- Because none of the initial models showed promising metrics, in a practical setting it would probably be more wise to consider changing the scenario/question being asked in the first place.
- Data analysis is pervasive in the world of baseball. It is kind of telling that even in this modern era of analytics, a team found tremendous success using a cheating scandal that involved banging a trashcan to signal hitters.