**Dataset Compilation/Web Scraping**

Pitchfx is a pitch tracking system that records multiple pitch features including velocity, movement and spin rate for every pitch thrown in a game. The system is currently installed in every MLB Stadium and has been in use since 2006.[[1]](#footnote-1) Given the project aim, Pitchfx data is the perfect source for building a prediction model. There are several online sources for scraping Pitchfx data (Brooks Baseball[[2]](#footnote-2), Baseball Heat Maps[[3]](#footnote-3), PitchRx) that employ various methods and formats to return some or all of the desired information. The challenge was to find an option that combined ease of use with the most comprehensive output possible.

Initially, we tried using PitchRx[[4]](#footnote-4), an R-based package designed specifically for Pitchfx. Unfortunately, the scripts were designed in a way that prevented the retrieval of one comprehensive dataset. Even if we had been able to concatenate the **pitch** and **atbat** subsets, they still lacked a sizeable amount of the features we had earmarked for potential analysis.

Our second approach was to entertain the idea of a customized scraper. This led us to look directly at the master archive itself, which is publicly available through MLB Gameday Directory[[5]](#footnote-5) and consists of a large number of sub-directories, the majority of which are organized by individual year, providing a hierarchical structure for each season’s data, all the way down to the pitch-by-pitch level. Information is stored in XML files, which most often represent a single inning of every game played. Navigating this complex environment would require designing a script to locate and then scrape all desired samples using BeautifulSoup, a Python library for pulling data from select file types (XML, HTML).

Before committing the time and effort to code a custom solution, we conducted one last round of research that ultimately led us to a Python-based parser/scraper[[6]](#footnote-6) that fortunately provided exactly what we needed. The scraper retrieves the features listed below and outputs them via two files (**pitch\_table.csv** and **atbat.csv**) that catalog the information on a per pitch basis:

* Game ID
* Flags
  + Spring Training
  + Regular Season
  + Playoffs etc.
* MLB Game ID
* Game Location Data
* Batter/Pitcher ID data
* Game Situation Data
* Pitch outcome sequence up to that point in the plate appearance
* Flag to designate if the pitch is the last pitch in the plate appearance
* Retrosheet-style event code

**Game Type Description**

The raw dataset includes a feature to describe the exact type of game that produced each individual sample: *Regular Season, Wild-Card Game, Divisional Series, League Championship Series (LCS), World Series, Spring Training* and *Unknown*. In order to keep our analysis limited to official games and lower the number of NaNs, we disregarded the pitches thrown in the *Spring Training* and *Unknown* categories*.*

**Creating Dummy Variables**

After cleaning the dataset, it was necessary to perform some feature engineering on a group of 7 k-class variables in order to run our baseline Logistic Regression model:

* Pitch Type
* Batter Position
* Months
* Year
* Number of Outs
* Ball Count
* Strike Count

The required k-dummy variables were generated using the **One Hot Encoding** module.

**Normalization**

While many of our features were between 0 and 1 (mostly percentages), we had 7 additional variables needed be expressed on a scale of 0 to 1. The features below were normalized using **minmax**. Compared to other normalization techniques, minmax provides a result with smaller standard deviations, which can suppress the effect of outliers.

* Pitcher Age
* Cumulative Pitch Count
* Home Runs Surrendered Per 9 Innings (HR9)
* Strikeout to Walk Ratio (SO/BB)
* Walks + Hits per Innings Pitched (WHIP)
* Earned Run Average (ERA)
* Pitches Per Plate Appearance

**Decision Tree**

In an attempt to improve upon the performance of our baseline model using Logistic Regression, we decided to run several **decision tree** instances, employing different hyperparameters that included *maximum depth, minimum leaf size* and *minimum split size*. A cross-section of 3 values was selected for each attribute and individual trees were then fit and run for each possible permutation, yielding the following accuracy scores:

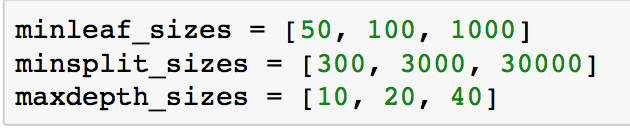


Figure 1

**

Figure 2

For each *min leaf size,* the smaller the *min split size*,the better the score. Moreover, higher *max depths* tend to give higher scores. Lastly, for each *min split size,* the smaller the *min leaf size* the better the score. Given the values above, the highest score was **0.485**.

Using information gained from the results in Figure 2, we adjust the hyperparameters to improve our model selection. Figure 4 shows that the highest score from this round was **0.486** (a slight improvement)with a *max depth* of 30 and *min split size* of 300.

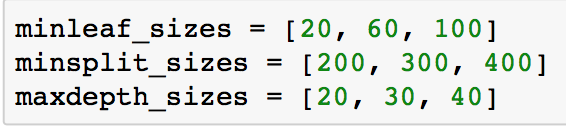


Figure 3



Figure 4

After settling on *max depth* and *min split size* as 20 and 300, respectively, we need to determine the optimal *min leaf size*. The additional tests show that the smaller *min leaf size* has performed better:

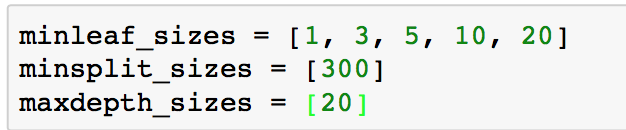


Figure 5



Figure 6

To conclude, the best result **0.486** (which outperforms the Logistic Regression Model) and log-loss **1.409**, is achieved using:

* *max depth size* = 20
* *min leaf size* = 5
* *min split size* = 300

Although, 1 and 3 perform slightly better than *min leaf size 5,* in order to reduce the complexity of the model, we used *min leaf size* 5 in our final model. The below figure summarizes the feature importance for our final decision tree models. The top three most important features are:

* **pct\_SI** = % of Sinkers thrown by pitcher in previous year (Sinkers/Total Pitches)
* **pct\_SL** = % of Sliders thrown by pitcher in previous year (Sliders/Total Pitches)
* **pct\_FT** = % of 2-Seam Fastballs thrown by pitcher in previous year (2-Seam Fast/Total Pitches)

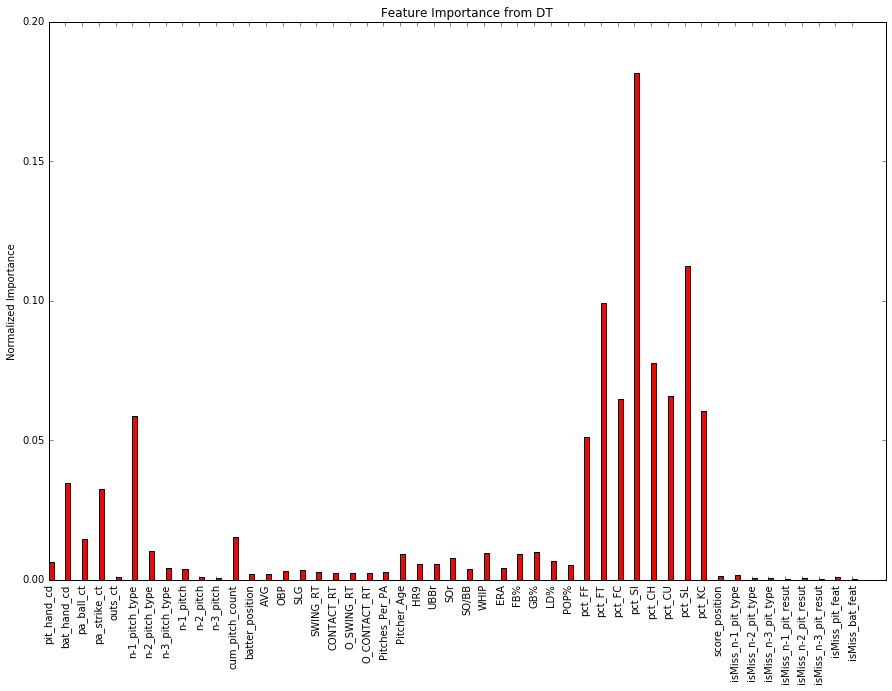


Figure 7

1. <http://www.fangraphs.com/library/misc/pitch-fx/> [↑](#footnote-ref-1)
2. <http://www.brooksbaseball.net/> [↑](#footnote-ref-2)
3. <http://www.baseballheatmaps.com/> [↑](#footnote-ref-3)
4. <http://cpsievert.github.io/pitchRx/> [↑](#footnote-ref-4)
5. <http://gd2.mlb.com/components/game/mlb/> [↑](#footnote-ref-5)
6. <http://www.beyondtheboxscore.com/2015/9/24/9374949/a-new-python-based-pitchf-x-parser-scraper> [↑](#footnote-ref-6)