What’s In A Pitch?

Insights from PITCHf/x

Introduction

PITCHf/x is a pitch-tracking system installed in every Major League Baseball (MLB) stadium. Cameras track pitch movement, and MLB Advanced Media makes that data (and more) publicly available.

PITCHf/x permits a finer-grained analysis of pitching, with breakdowns of velocity, movement, spin, and location.

The Data

Brooks Baseball (http://www.brooksbaseball.net/) provides a hand-corrected version of the PITCHf/x data. I found a Scrapy-based project on GitHub (https://github.com/mattdennewitz/mlb-brooks-pitch-importer) to download the data.

The Data

- Every pitch thrown in every game between 2010 and 2015
- Date, park, players, position, spin, velocity, strike zone dimensions
    - Pitch type
- A LOT OF DATA

Preprocessing

- Filtering of rare pitcher-batter ‘matchups’
  - 1 matchup = 1 pitch thrown by Pitcher P to Batter B
  - Only events involving players with >= 30 matchups in the data were kept
  - Noise reduction
- Factorization of categorial columns

Modeling

- 3 Classes (Hit, Strike, Ball)
- Heterogenous data

Answer: Random Forests

Modeling

Aside: I looked into SVMs, but they took so long to train that they weren't considered in depth.

Analysis & Results

Goal: determine what aspects of a pitch are most important to the outcome.

Plan: consider different groups of features, progressively informing models to increase predictive power. We'll infer the importance of features by inspecting the weights our models assign to them, as well as the success of those models.

Analysis & Results

We use F1 as a scoring metric because it balances precision and recall and handles multiple classes well.

Since the focus is on the features as opposed to sheer predictive power, we will construct all our models with the same parameters.

Analysis and Results

Model I: “Pre-pitch” data – what could be reasonably inferred before the ball leaves the pitcher’s hand (but including release position).

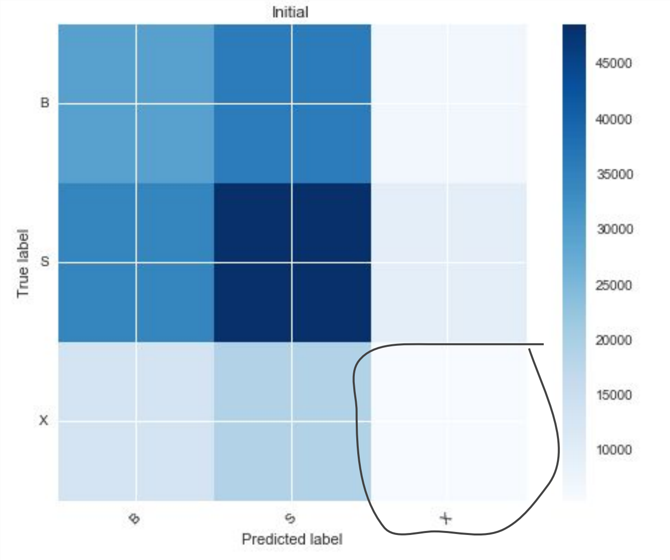
Columns: `ab_count, pitcher_id, batter_id, factorized_stand, strikes, balls, factorized_p_throws, x0, y0, z0, factorized_park`

Analysis & Results – Model I

Results: Not good!

F1: 0.413

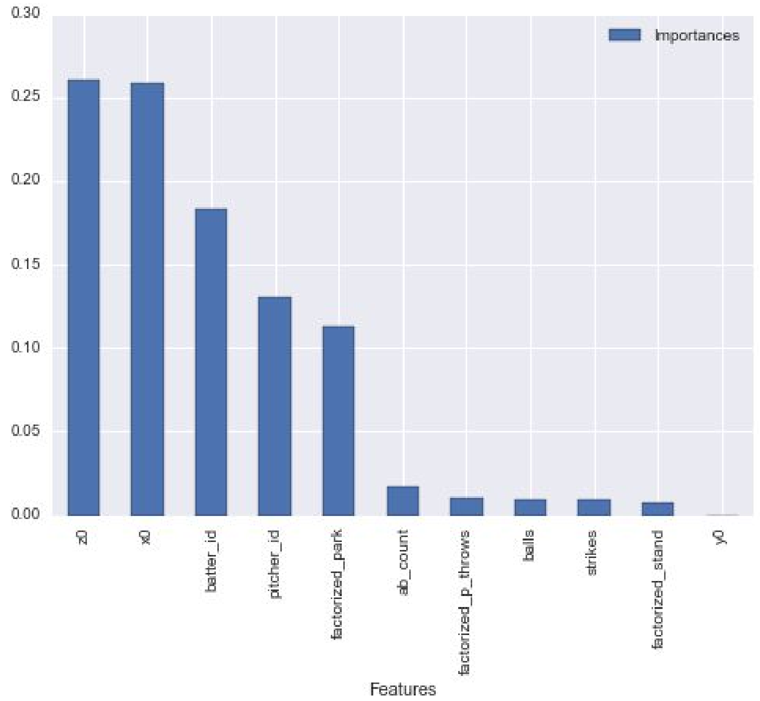
Note the underclassification of strikes–it's a persistent theme.



Analysis & Results – Model I

Features importances aren't too informative given our lousy F1 score, however:

- `batter_id` is more informative than `pitcher_id`
- `z0` (initial vertical position) is more informative than `x0` (initial horizontal position)



Analysis & Results

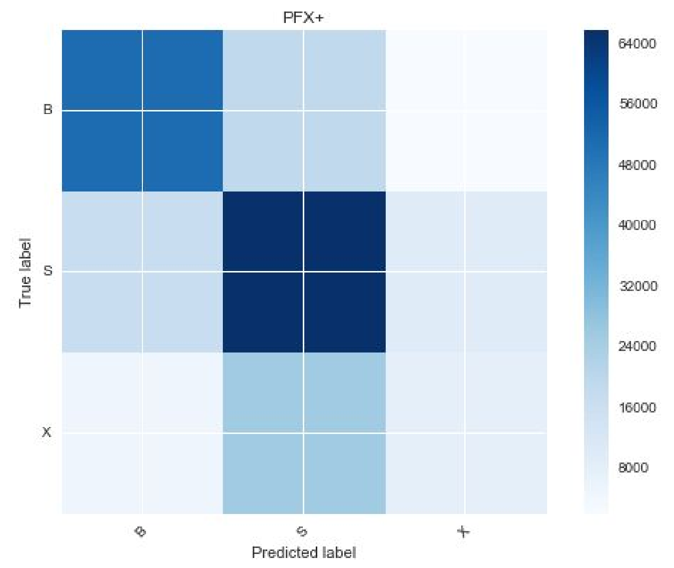
Model IIa: PITCHf/x + Initial

Columns: Columns from Model I, plus `sz_top, sz_bot, spin, pfx_x, pfx_z, start_speed, vx0, vz0, vy0, ax, ay, az`

Analysis & Results – Model IIa

Results: Better!

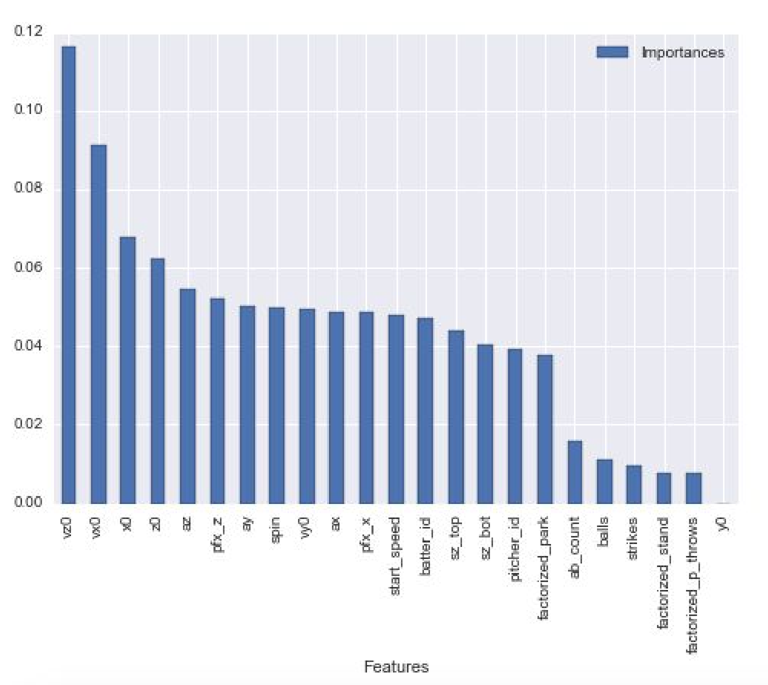
F1: 0.616



Analysis & Results – Model IIa

There's been some shuffling–`vz0` (initial vertical velocity) outranks `vx0`, but `x0` beats out `z0`, in contrast to our previous model.

Notice also that the Pfx data is on balance more informative.



Analysis & Results

Model IIa – PITCHf/x *Only*

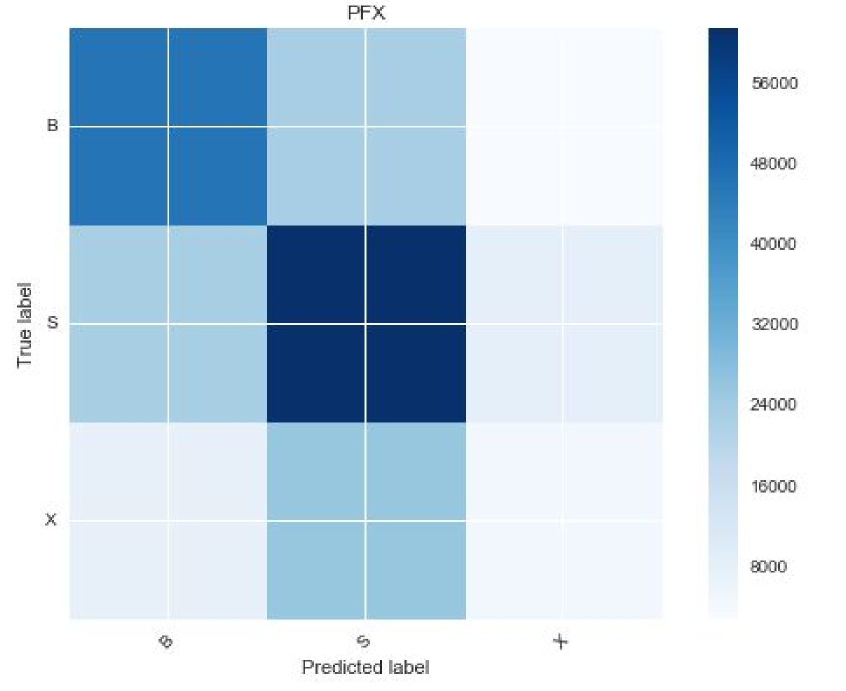
Columns: *Only* the PITCHf/x columns, without the initial data.

Analysis & Results – Model IIb

Results: Only slightly worse!

F1: 0.553

Weights essentially unchanged, reinforcing our suspicions about the relative importance of PITCHf/x data vs our initial data.



Analysis & Results

Model III: Pitch type

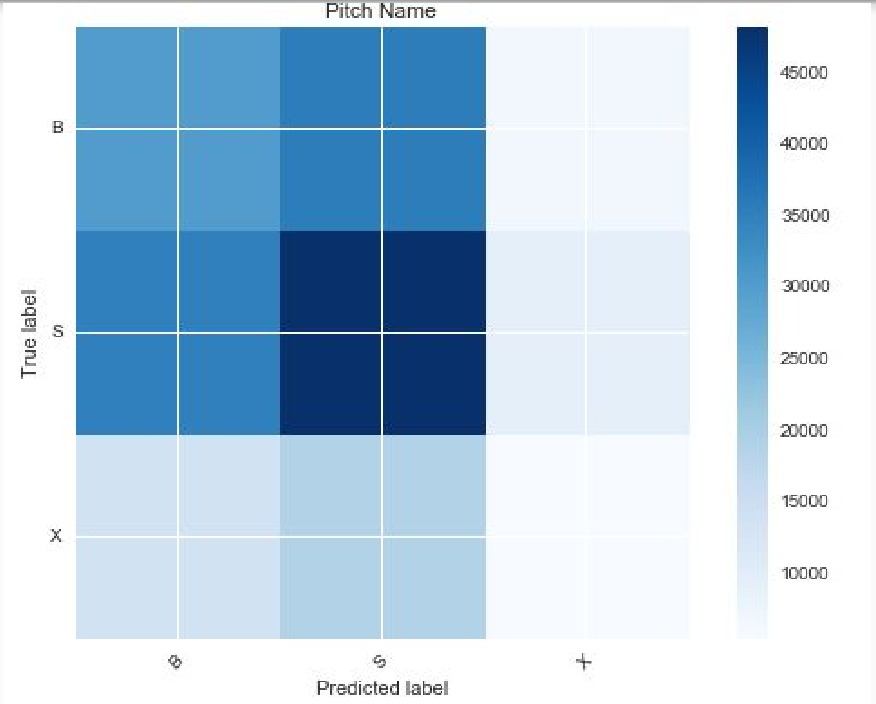
Columns: Initial data + `factorized_mlbam_pitch_name`

MLBAM labels every pitch with a RNN; Brooks hand-corrects it when necessary. Theoretically, this captures most of our PITCHf/x data–but does it?

Analysis & Results – Model III

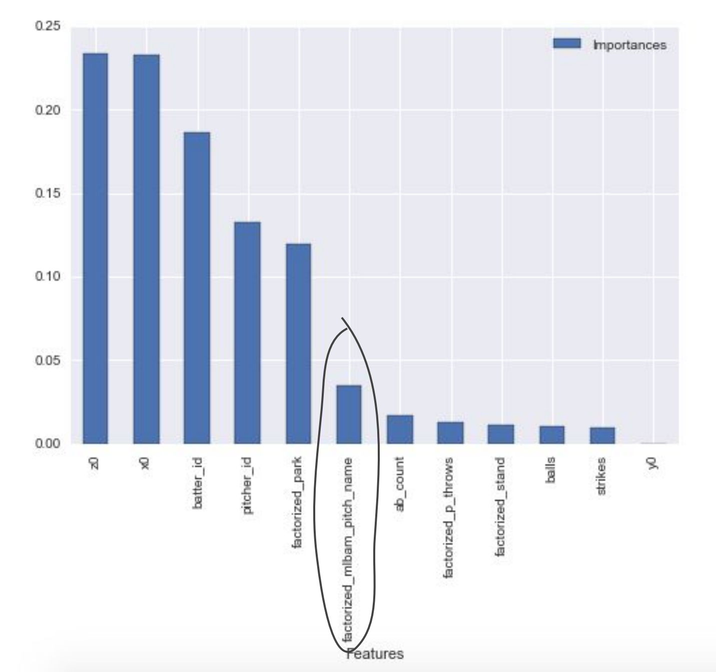
Results: Not really?

F1: 0.414



Analysis & Results – Model III

Even more damning: pitch type is less informative, according to our model, than several of our previously-used features.



Analysis & Results

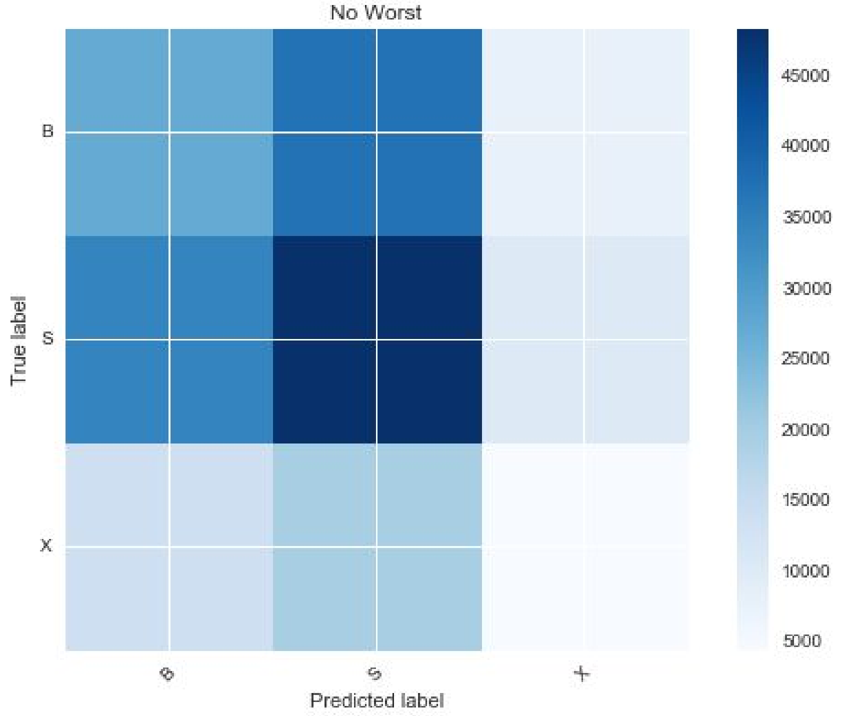
Certain factors are consistently low-weighted, across models. We can train some models without these to measure their impact.

Model IVa: Initial - worst-performing columns

Analysis & Results – Model IVa

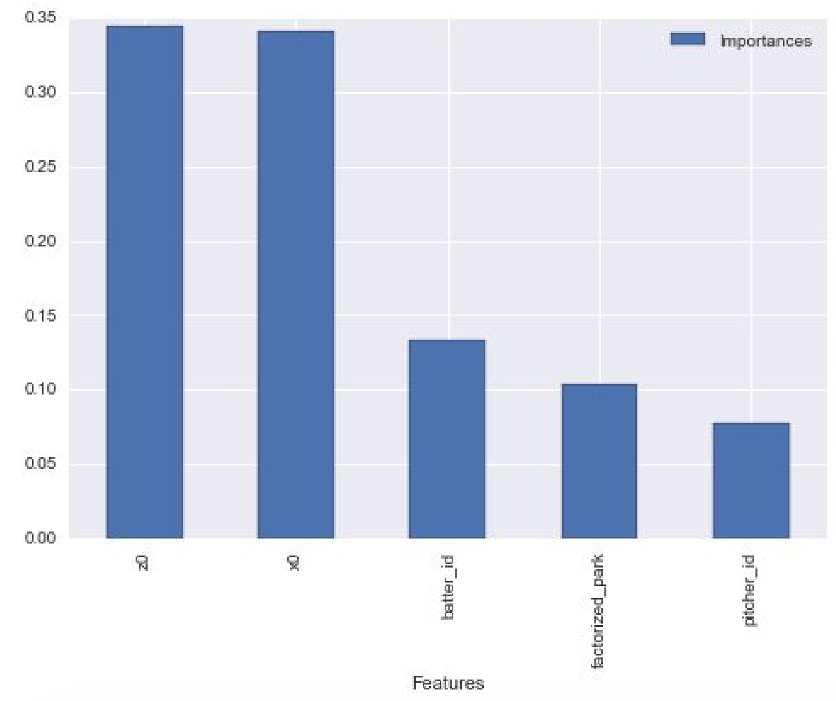
Results: Not great, but that's to be expected with only 5 factors to consider.

F1: 0.395



Analysis & Results – Model IVa

No real surprises here.



Analysis & Results

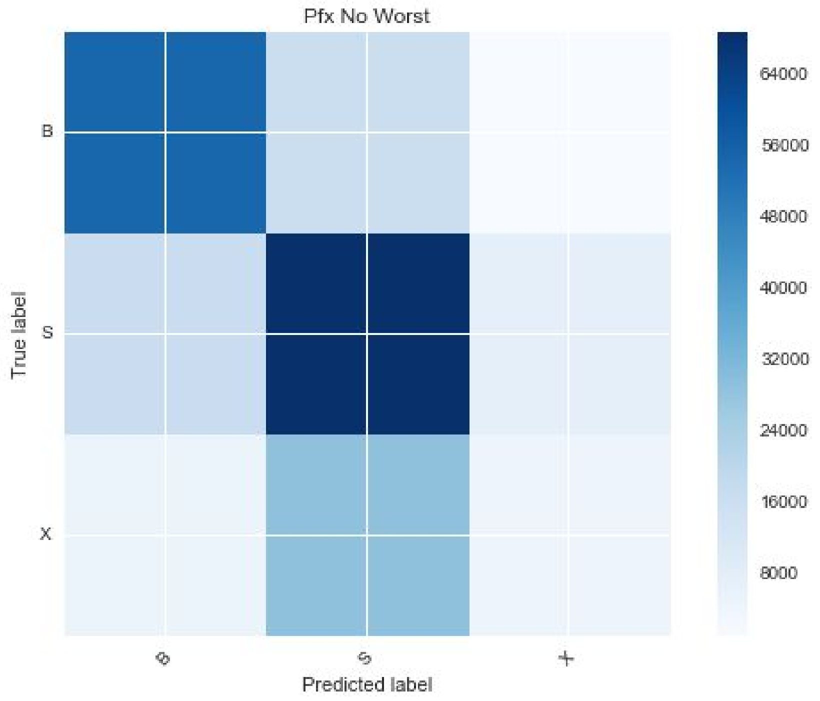
Model IVb: Pfx + ‘best’

Analysis & Results – Model IVb

Results: our best model yet!

F1: 0.632

Suggests more conclusively that our ‘worst-performing’ columns are useless at best, slightly harmful at worst.



Analysis & Results

Model Va: “Cheating”

Columns: `px`, `pz`, `norm_ht`, `factorized_zone_location`

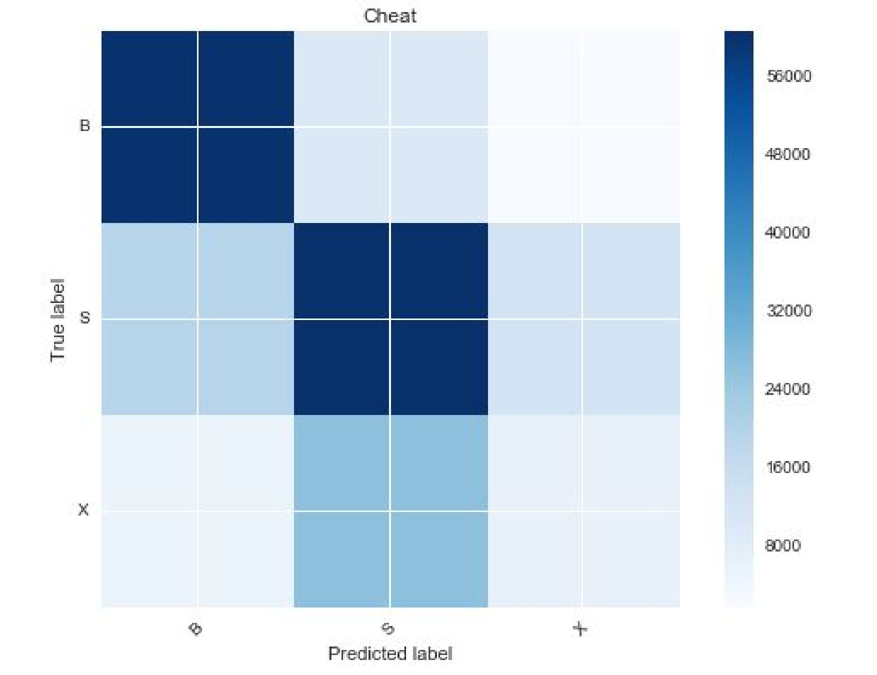
`px` and `pz` are the horizontal and vertical position of the ball as it crosses home plate; `norm_ht` is another vertical measure, `factorized_zone_location` abstracts over both.

Analysis & Results – Model Va

Results: Expectedly good!

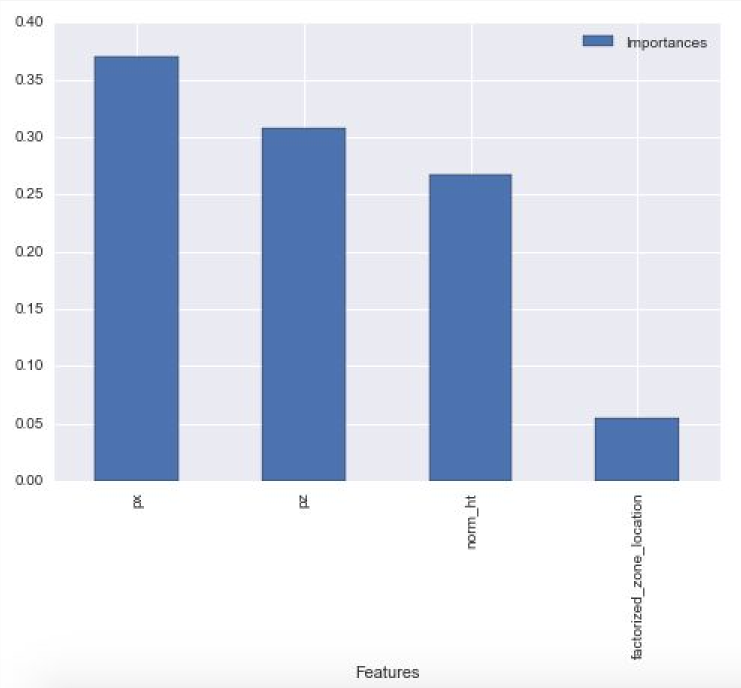
F1: 0.628

These 4 columns predict roughly as well as all our previous columns.



Analysis & Results – Model Va

Notice how unimportant `factorized_zone_location` is–like `mlbam_pitch_name`, finer-grained factors are clearly more informative.



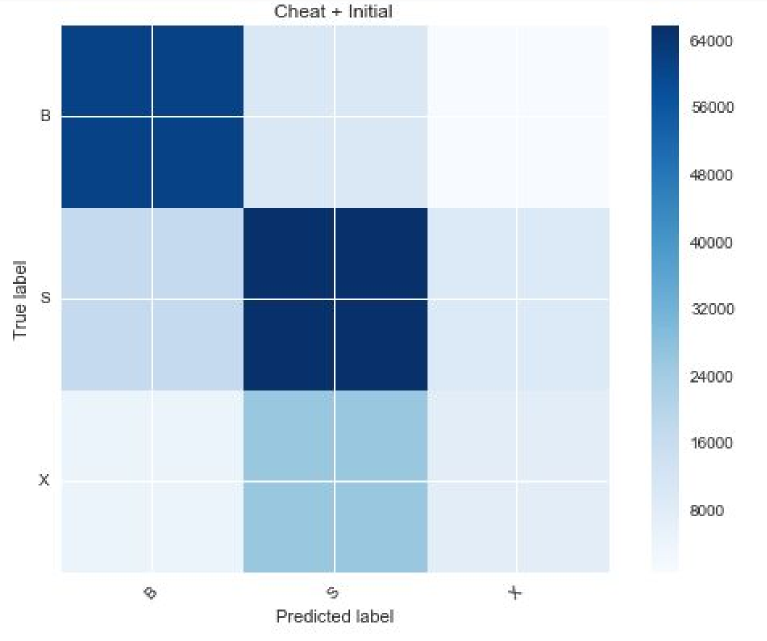
Analysis & Results

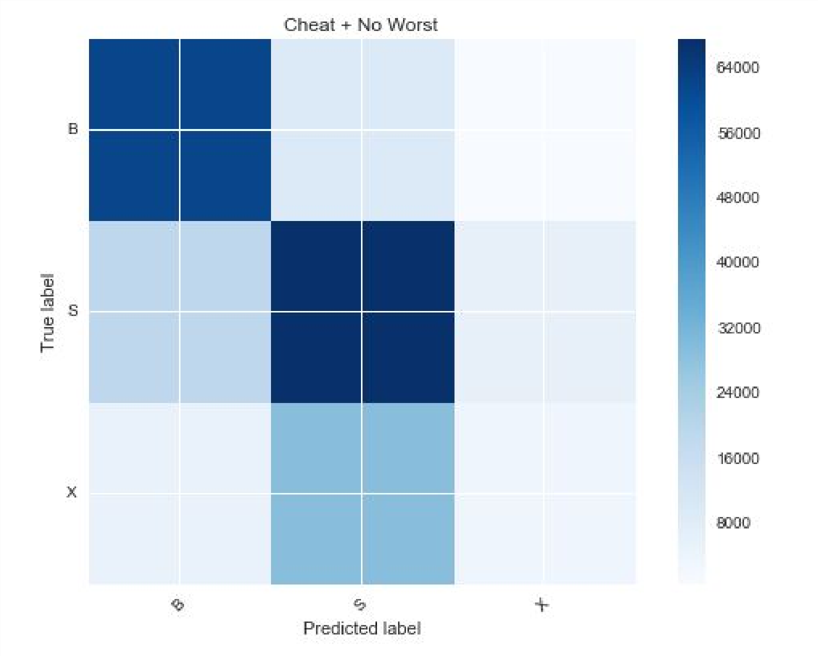
Model Vb, Model Vc: “Cheat”+

Vb: “Cheat” + Initial

Vc: “Cheat” + “best”

Analysis & Results – Model Vb, Vc





Analysis & Results – Model Vb, Vc

Vb F1: 0.665

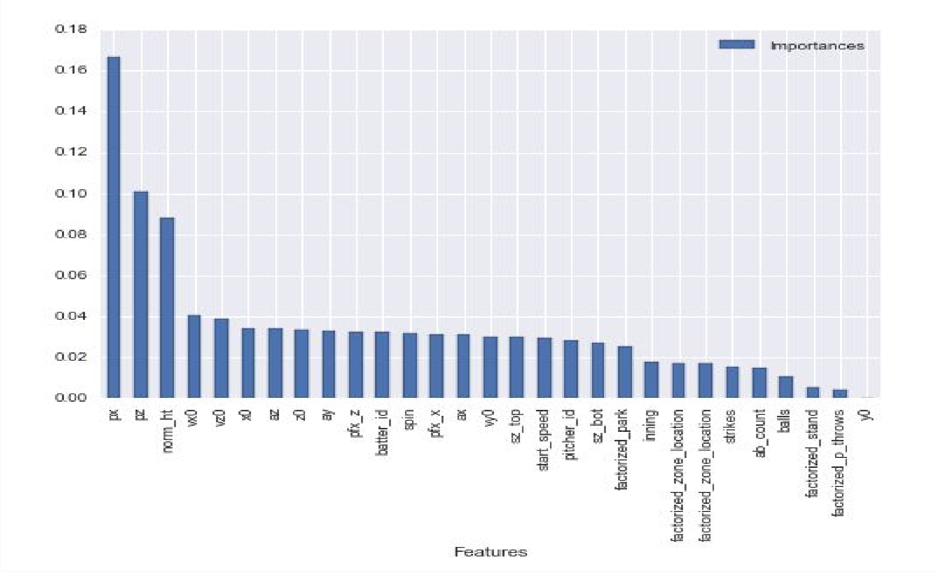
Vc F1: 0.656

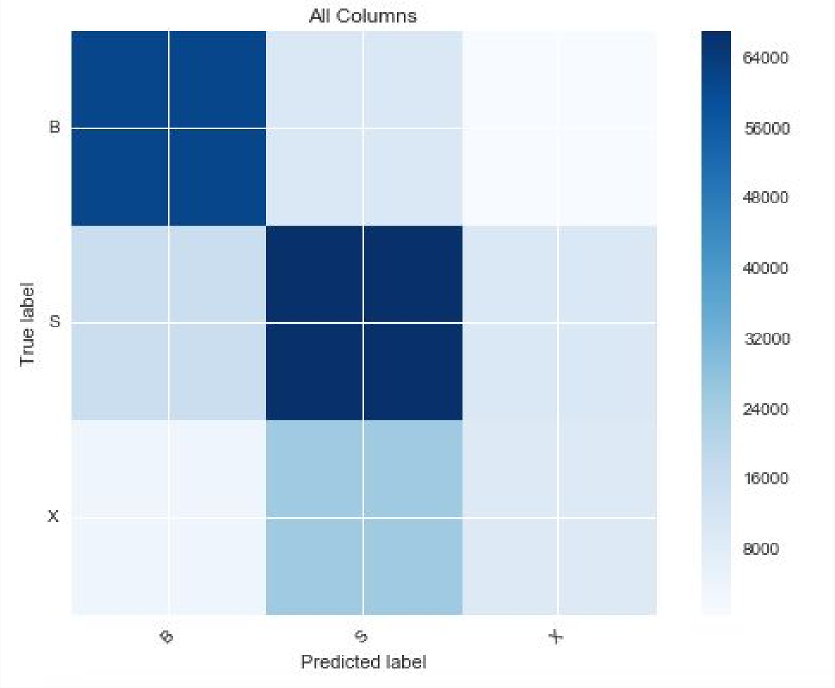
Analysis & Results

Model VI: “All” columns

Analysis & Results – Model VI

F1 Score: 0.678





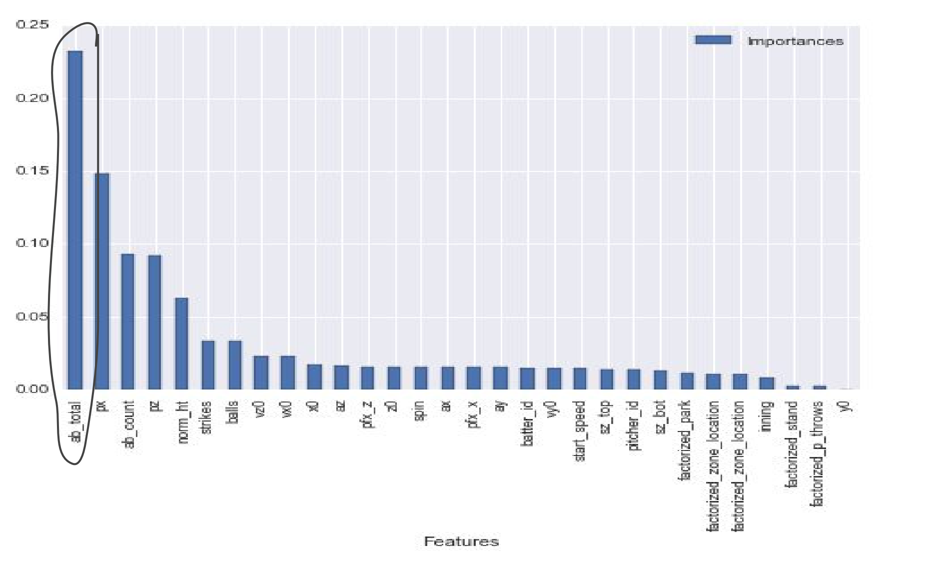
Analysis & Results

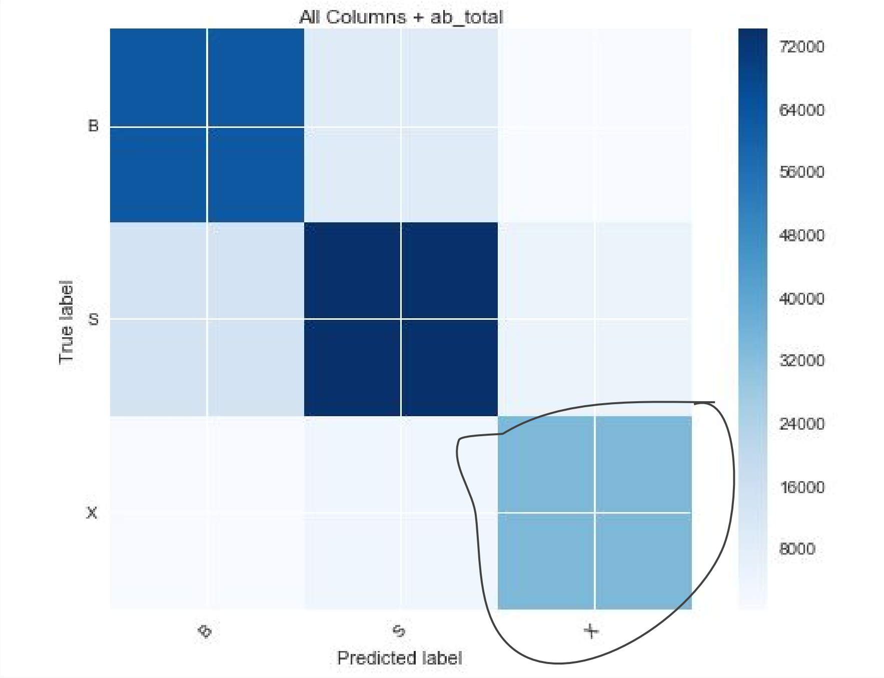
Model VII: Cheat Harder

`ab_total` represents the *total number of events in the at-bat*. It hasn't been included because it's not strictly "about" the individual pitches, but rather the at-bat *as a whole*. It's also *incredibly helpful*.

Analysis & Results – Model VII

F1: 0.843 (!)





Analysis & Results

```
In [71]: train[train.type == 'X'].ab_total.describe()

Out[71]: count    558217.000000
         mean          3.484437
         std           1.856281
         min           1.000000
         25%           2.000000
         50%           3.000000
         75%           5.000000
         max          16.000000
         Name: ab_total, dtype: float64
```

At-bats that end in a hit are significantly shorter than those that don't!

```
In [72]: train[train.type == 'S'].ab_total.describe()

Out[72]: count    558217.000000
         mean          5.173180
         std           1.850429
         min           1.000000
         25%           4.000000
         50%           5.000000
         75%           6.000000
         max          16.000000
         Name: ab_total, dtype: float64
```

```
In [74]: train[train.type == 'B'].ab_total.describe()

Out[74]: count    558217.000000
         mean          5.163071
         std           1.635035
         min           1.000000
         25%           4.000000
         50%           5.000000
         75%           6.000000
         max          16.000000
         Name: ab_total, dtype: float64
```