Analysis of Pitch Type Prediction using Individualized Models

STA4241 Written Report

Zachary Allen

December 8, 2023

Introduction

In baseball, pitchers employ various grips on the ball, along with different arm and hand motions upon release, to generate pitches with distinct velocities, spins, and horizontal or vertical movements. The primary goal is to deceive a hitter's timing, vision, and swing. For instance, at the professional level, a Four-seam Fastball typically exceeds 90 miles per hour, while a Curveball may be slower but possess more vertical movement and spin. If a batter had perfect knowledge of the upcoming pitch type, their likelihood of making contact with the ball would significantly increase. This is because the reaction time required for the batter to identify the ball's movement would be nearly eliminated. In 2017, the Houston Astros illicitly used cameras to steal codes that catchers would convey to their pitchers, indicating the type of pitch to throw. They deciphered these codes in real-time, signaling whether an impending pitch would be a breaking ball (curveball, slider, sinker, etc.) by banging on a trash can, and achieved a 93% success rate in predicting these pitch types.

Project Objective

The project aims to develop a binary classifier capable of predicting whether an upcoming pitch will be a fastball or not. The project objectives are as follows: (1) To outperform both random guessing and a baseline model, (2) To be individualized, with models tailored to data associated with specific pitchers, and (3) To utilize publicly available data and apply techniques acquired throughout STA4241.

Data Collection

To fulfill the individualized model objective of this project, the `baseballr` package will be employed to retrieve pitch data for Justin Verlander, Zach Eflin, Clayton Kershaw, Blake Snell, and Shohei Ohtani from `https://baseballsavant.mlb.com/` from 2015 (when StatCast data was first collected) through the 2023 season. Each pitcher's data will be treated as a separate dataset for training models, instead of using a consolidated dataset with an additional pitcher covariate. This approach aims to enhance clarity in determining the effectiveness of models for individual pitchers and identifying overall model performance.

Methodology

Feature Selection

For every pitched ball, the MLB compiles a set of 85 features that describe the pre-pitch game state, the movement of the pitch itself, and the outcome of the pitch. In the context of this project, the focus is solely on the pre-pitch game state features. Among these, the most pertinent include the count, which denotes the number of balls and strikes before a pitch is thrown (12 factors); information about runners on each base, with 2 factors per base; the number of outs, encompassing 3 factors; the hitter's stance, represented by 2 factors indicating whether the hitter is in a left or right stance; the current inning; and the current year (factor).

Furthermore, an upcoming pitch is rarely independent of the preceding pitches. To address this, two additional features have been introduced to capture relevant information. Firstly, `FBStreak` represents the number of pitches thrown since the last breaking ball per batter. Secondly, `isRecentFB` indicates whether the most recent pitch type was a fastball per batter (categorized as yes, no, or none if it is the first pitch of the at-bat). Considering that a pitcher may engage with the same batter multiple times in a game, adjusting their approach each time, the number of times a batter has faced a pitcher per game was also computed as `abNum`.

Model Selection

The following four models were fit for each of the five pitchers. An important characteristic of these models is that for a given test prediction, class probabilities are computed.

Baseline

The baseline model simply always guesses whichever class is most common in the training set. For instance, Clayton Kershaw threw 55.41% fastballs in his training set, so every prediction in the test set under this model would also be fastball.

Naive Bayes

Although not directly covered in this course, Naive Bayes takes a similar approach to LDA and QDA. Bayes' theorem provides an expression for the posterior probability $p_k(x) = P(Y = k \mid X = x)$ in terms of $\pi_1, \ldots, \pi_K$ and $f_1(x), \ldots, f_K(x)$. The first of which is generally straightforward to calculate, however, rather than assuming $f_k(x)$ belongs to a multivariate normal distribution like in LDA and QDA, Naive Bayes makes a single assumption that within the kth class, the p predictors are independent. Mathematically this looks like

$$f_k(x) = f_{k1}(x_1) \times f_{k2}(x_2) \times \ldots \times f_{kp}(x_p)$$

where f_{kj} is the density function of the jth predictor for observations in class k. This new assumption makes estimating a p-dimensional density function easier since the joint distribution of the predictors does not be considered, ultimately introducing bias and reducing variance that typically works well in practice even if the primary assumption is not met.

Multiple Logistic Regression

Another model that is typically well-suited for binary classification and computes probabilities for each class is Multiple Logistic Regression (MLR) which has the following model definition:

$$\log\left(\frac{P(Y=1 \mid X=x)}{1-P(Y=1 \mid X=x)}\right)=\beta_0+\sum_{j=1}^{p}\beta_j X_j$$

Stepwise Regression was also performed for each pitcher's model to determine the most significant predictors in terms of AIC. However, doing resulted in minimal fluctuations to the test error rate, AUC, and AIC values, so these reduced models were ignored for the final results.

Boosting

The final model that will be tested is Boosting. Boosting is more complex and less interpretable compared to the previous models, but has several advantages including the ability to handle complex relationships, robustness against overfitting, and good generalization performance. Boosting works by iterating and growing decision trees sequentially, that is, each tree is grown using information from previously grown trees and does not involve bootstrap sampling. Compared to Random Forests, which builds multiple large decision trees independently in parallel, Boosting learns slowly over time and can emphasize instances that are misclassified in subsequent iterations. Boosting has several tuning parameters which were tuned using grid search and 5-fold Cross Validation

- Number of Trees: 100 to 2500 in 100 tree increments.
- Interaction Depth: 1, 2, and 3.
- Shrinkage: $5\times10^{-4}, 5\times10^{-3}$, and 5×10^{-2}.
- Minimum Observations in Node: 1, 5, and 10.

The resulting parameters for each pitcher were as such

Pitcher	Trees	Depth	Shrinkage	Min Obs In Node
Justin Verlander	1800	2	0.05	5
Zach Eflin	1800	3	0.05	5
Clayton Kershaw	900	3	0.05	1
Blake Snell	1500	2	0.05	5
Shohei Ohtani	2200	3	0.005	10

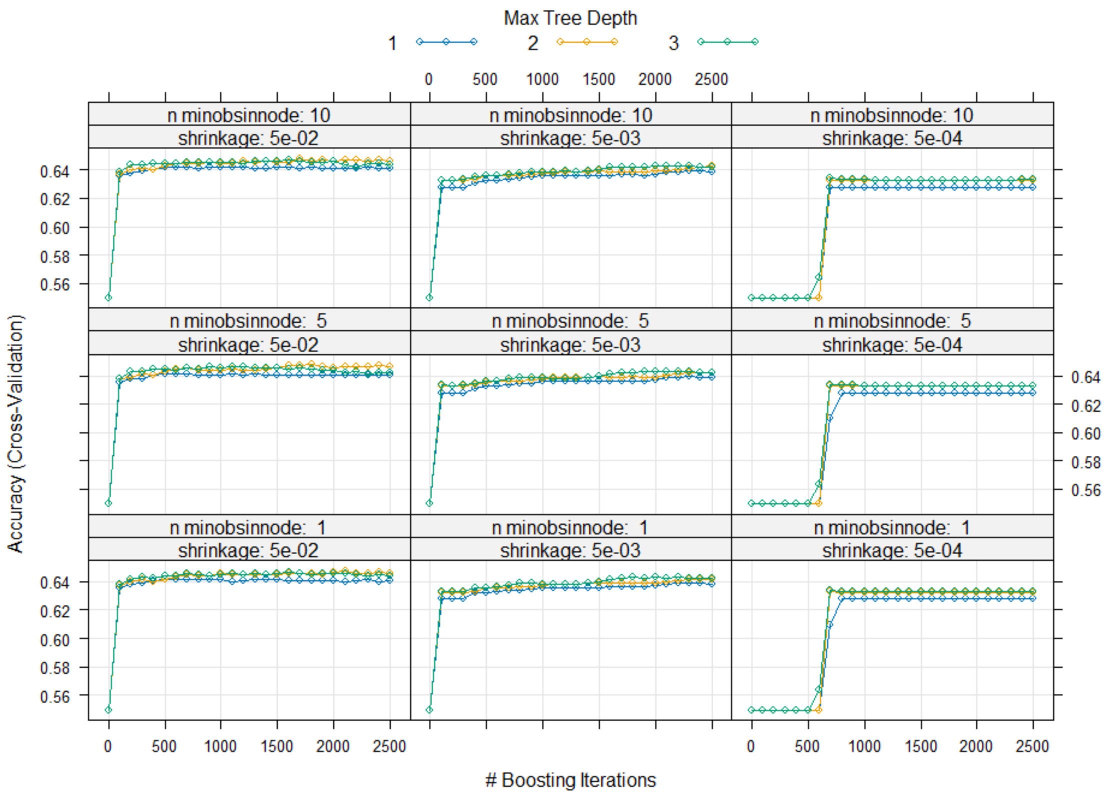

Figure 1: Justin Verlander Learning Curves for Boosting Parameters

Learning curves were also plotted for each of the pitcher's such as Justin Verlander's above. For the brevity of this report, all charts, tables, and data generated for each pitcher can be found here: https://github.com/ZackAllen1/pitch-prediction/.

Results

Pitcher	Total Pitches	Baseline Test Error	Naive Bayes Test Error	MLR Test Error	MLR AUC	Boosting Test Error
Justin Verlander	20986	0.4297	0.3484	0.3515	0.6894	0.3506
Zach Eflin	12900	0.3880	0.3783	0.3839	0.6420	0.3584
Clayton Kershaw	19254	0.4424	0.3556	0.3471	0.6935	0.3292
Blake Snell	17419	0.4756	0.4322	0.4177	0.6180	0.4002
Shohei Ohtani	7612	0.3755	0.3688	0.3620	0.6415	0.3544
Average	**15634.2**	**0.4222**	**.3767**	**0.3724**	**0.6569**	**0.3586**

Table 1: All Model Results for Each Pitcher

Based on the table above, for all pitchers except for Verlander, the Boosting model resulted in the best test error rates. Additionally, some pitchers like Eflin and Ohtani had all three models show only a 1 to 3% test error improvement over the Baseline whereas others like Kershaw showed a 9 to 12% improvement. Averaging over all pitchers, Naive Bayes, MLR, and Boosting resulted in a 4.55%, 4.96%, and 6.36% test error improvement respectively over the baseline. The average AUC for all five pitchers is 0.6569 which is subpar compared to what is typically considered acceptable (0.70+) or good (0.80+).

Varying Thresholds

In addition to analyzing the raw test errors, since all three models output the probability of a given test pitch belonging to a given pitch type, we can calculate what percent of the test set could be theoretically related to a hitter. For instance, Justin Verlander's Naive Bayes model results in approximately $0.75 \times 925 = 694$ test pitches that could have been correctly relayed to a hitter using a threshold of 0.75. Assuming this model could have been used against him since 2015, this would have resulted in 14690 pitches with no information relayed, 4655 correct relays, and 1551 incorrect relays. Similar results occur across all models and pitchers.

Threshold	Accuracy	Num. Test Pitches	% of Total Test Pitches
0.50	0.6515866	3183	100.000000
0.60	0.6773150	2473	77.525968
0.75	0.7427027	925	29.713566
0.80	0.7690840	524	16.997167
0.90	0.7920000	125	3.966006
0.95	0.8392857	56	1.857098

Table 2: Justin Verlander, Naive Bayes, Test Accuracy By Threshold

Conclusion and Limitations

In conclusion, this study demonstrates the feasibility of predicting whether an upcoming pitch will be a fastball solely based on the pre-pitch game state. Strong evidence supports the notion that fitting models based on individual pitcher data is advantageous, as evidenced by Clayton Kershaw's 12% improvement over guessing the most frequently thrown pitch type using a Boosting Tree model. Moreover, the study reveals the potential for relaying upcoming pitch types to a batter only when the model exhibits a certain confidence level, thus providing a strategic advantage. While this approach may yield a smaller percentage of correctly predicted pitches compared to the 2017 Houston Astros, the absolute number of potentially correct pitch types that could be relayed is not insignificant, potentially leading to noticeable improvements in hitter batting averages or on-base percentages.

However, the results of this analysis must be interpreted in light of certain limitations. Firstly, there are missing features that would enhance the models but cannot be measured. For example, the distinction between the pitcher's intended location for a pitch and its actual outcome plays a role in determining the next pitch type, but this information is not recorded by the MLB. Additionally, measuring pitching coach tendencies is not feasible, as they often guide pitchers and catchers on pitch selection, considering hitter weaknesses and pitcher strengths.

References

Astros Cheating Analysis by Jake Mailhot:
1: Most Important Bangs of Astros' Scheme
2: How Much Did the Astros Really Benefit from Sign-Stealing?

BaseballR Library: `https://billpetti.github.io/baseballr/index.html`

StatCast Column Descriptions: `https://baseballsavant.mlb.com/csv-docs`

Project GitHub Repository: `https://github.com/ZackAllen1/pitch-prediction`