The problem of pitch recognition in Major League Baseball

A project for the course of "Data Mining"

***Abstract*—Proper pitch classification/recognition is an important aspect in baseball, both for tv broadcasting purposes and team desicion making. We approach the problem using the Statcast data.**

***Index Terms*—data-mining, baseball, pitch classification**

I. Introduction

The use of statistics in baseball has a long and fascinating history. During the past decades, baseball research has become an independent field and has given birth to ideas and tools that influenced and shaped the whole field of sports analytics. Especially in the recent years, the technological advances of other fields have provided us with in-game data that were unimaginable until recently.

Perhaps the most important breakthrough was the introduction of PITCHf\x in 2006, a system of cameras able to record data from any pitch thrown or hit during a game. In this project we will use data from Statcast, which was introduced in 2015 as the successor to the PITCHf\x, to approach the pitch recognition/classification problem, namely to provide an automated way to classify each pitch into its correct type.

In Appendix A we briefly describe the basic concepts of baseball. We focus on the duel between the pitcher and the hitter as it is the only part of the game that we examine in our project. Although we included it as an appendix, the unfamiliar reader might consider starting his study from it. In Paragraph II we describe the problem of pitch classification and review the relevant literature. Paragraph III is devoted to our implementation of the pitch type classification based on the Statcast data.

II. The pitch classification problem

During a baseball broadcast, after each pitch has been thrown, its velocity and pitch type usually appear on the screen automatically. Recognizing the pitch type with the naked eye is often a challenging task even for baseball veterans. Surely, it is relatively easy to distinguish between a fastball and a curveball based on the ball's trajectory and speed, however distinguishing between each subtype is not that straightforward.

Major League Baseball (MLB) first began to automatically classify pitches in 2006, by taking into account features of the ball thrown, like its speed, break, spin rate, acceleration/velocities on each axis, etc., as provided by the PITCHf\x software. Each pitch type was then presented in real-time during the broadcast, among with some qualitative characteristics of it, such as its speed or its exact trajectory.

In the beginning [SS20], MLB only used two neural networks to perform the classification, one for left and the other for right handed pitchers. This resulted in poor accuracy since pitch characteristics vary significantly from one pitcher to another. After constant tweaks and improvements, they have now patented [PS10] a uniquely customized neural network for each pitcher.

Closely related to our work is [ADG13], where various classifiers are applied and assessed for classifying pitch type. In addition, dimension reduction methods were used as a means to identify the prevalent features that influence pitch type. Their results are not directly comparable to ours, as they only focused on a relatively small number of pitchers (10) during the 2011 season.

MLB's classification system is geared towards broadcasting. As such, the pitcher-centered classification is vital, but it can also be flawed: Two different pitchers may throw a similar pitch but assign two distinct pitch types to them when they describe them. The MLB algorithm will respect their choice of names and will classify them into two different categories, despite them being almost identical.

Approaches which bypass the broadcast-centered model have been developed. In [Moo19] k-means clustering is proposed to detect each pitcher's repertoire, partially ignoring the Statcast labels. In [UYN20] and [PVS13] a similar procedure is followed, based on the Variational Bayesian GMMs and Multivariate GMMs respectively. A different perspective is considered in [WDS20], where video footage is used in order to reconstruct the pitcher's full body motion and then pitch classification is performed using only this information. They conclude that although pitch type could be partially deduced by the pitcher's motion, the accuracy of the prediction was significantly lower (50-60%) than the one which would take into account the kinematic characteristics of the pitch itself.

Perhaps of greater importance is the problem of pitch prediction, namely the ability to predict what type of pitch the pitcher is about to throw, deploying only information which is available prior to, and not during, the throw. In [ST18] they use PITCHf\x data from seasons 2013-2015 to achieve a 60% success rate when predicting the pitch to come. In [Boc15] they break down pitch prediction for various in-game situations, resulting to an overall success rate of 74.5% for the seasons 2011-2013.

III. Pitch Classification Implementation

Our initial dataset was the Sportradar's MLB 2016 Pitch-by-Pitch which is publicly available by the Google Cloud Platform. A preliminary analysis on it resulted to relatively poor classification results which was to be expected as the features it contained lacked any kinematic information and were related to pitch classification only indirectly. We enhanced our dataset using pybaseball [Led17], a python library which scrapes data from the three major baseball stat sites, Baseball Reference, Baseball Savant, and FanGraphs.

A. Feature Selection

The main body of our work was based on the 2016 MLB season dataset, as drawn by pybaseball using a basic query. Apart from general information concerning the game situation before each pitch, it also contained kinematic information for each thrown pitch, information which we expect to be vital for the goal of pitch classification.

Our target feature is the type of pitch thrown:

pitch_type: The pitch type can be either a 4-Seam Fastball (FF), a 2-Seam Fastball (FT), a Cutter (FC), a Split-Finger Fastball (FS), a Curveball (CU), a Slider (SL), a Knuckleball (KN), a Knuckle Curveball (KC), a Changeup (CH), a Forkball (FO) or a Sinker (SI). Some additional pitch types were excluded from our dataset due to their scarcity, or due to the existence of more convenient[1] classification methods.

The goal is to deduce the pitch_type given a series of features which are related to the pitch thrown, such as its velocity, its spin etc. Among the features considered, we briefly describe the most important ones [BS]:

release_speed: The ball's velocity (MPH) the instance it leaves the pitcher's arm.

effective_speed: The ball's velocity (MPH) as perceived by the batter [Auc19].

release_pos_x: The horizontal release position of the ball measured in feet from the catcher's perspective.

release_pos_z: The vertical release position of the ball measured in feet from the catcher's perspective.

player_name: Pitcher's full name.

p_throws: Hand pitcher throws with (L or R).

stand: Side of the plate the batter is standing (L or R).

pfx_x: Horizontal movement[2] of the ball in feet from the catcher's perspective.

pfx_z: Vertical movement[2] of the ball in feet from the catcher's perspective.

plate_x: Horizontal position of the ball when it crosses home plate from the catcher's perspective.

[1]For example *pitch outs* and *intentional balls* are pitches thrown far away of the strike zone, intentionally surrendering a ball for purposes unrelated to the current at-bat. Such pitches are easily recognizable with the naked eye.

[2]The ball movement value measures the distance between the end point of the actual pitch and the end point of a hypothetical pitch corresponding to the trajectory (or path) that the ball would have taken if it wasn't affected by its rotation.

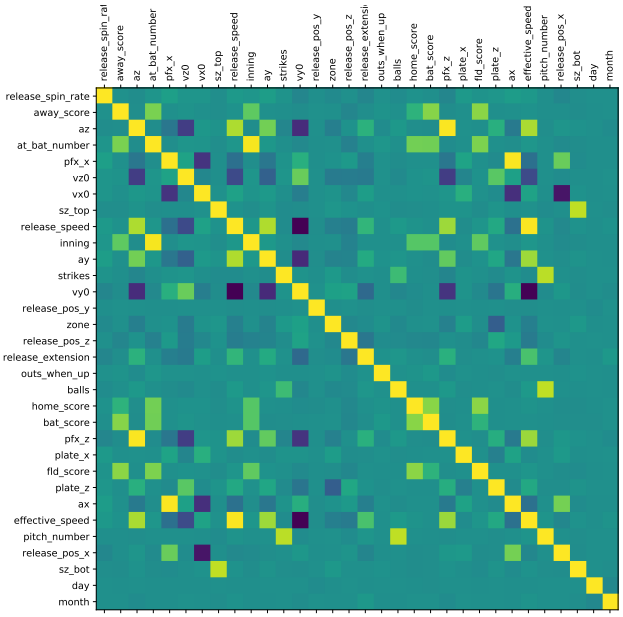

Figure 1. Correlations between our numeric features.

plate_z: Vertical position of the ball when it crosses home plate from the catcher's perspective.

release_spin_rate: Ball's spin rate (in RPM) when it leaves the pitcher's hand.

release_extension: How far off the mound (ft) a pitcher releases the pitch. Pitchers with long wingspans tend to have longer extensions, since they can hold on to the ball for a greater distance before releasing it, during their pitching motion.

vx0, vy0, vz0: The velocity of the pitch (ft/sec) in the x, y and z-dimensions respectively, determined at y=50 feet.

ax, ay, az: The acceleration of the pitch (ft/sec^2) in the x, y and z-dimensions respectively, determined at y=50 feet.

To decide which features to work with, we performed an initial selection using the variance criterion. Continuous features with small variance were excluded. For the categorical variables we did a similar procedure, using as a criterion the entropy of the feature, $H(\tilde{p}) = -\sum_{i=1}^{N} p_i \log p_i$, where p_i denotes the relative frequency of the i-th level of the feature in question and $\tilde{p} = (p_1, \ldots, p_N)$. Features with extremely high or low entropies were discarded.

Additional features were dropped using expert knowledge as they contained information either unrelated to the problem, or unknown to us prior to each pitch. After the initial selection we examined the correlations between our features (Figure 1) and between the features and the pitch type (Figures 2 and 3).

The original data contained 710.623 rows, some of which were referring to non pitching events. After removing them, the final data frame contained a total of 654.383 pitches and 41 features, 8 of which were categorical and should be converted to numerical. The most important of the categorical features was the pitcher's name. As we mentioned before, MLB's official classification improved dramatically when they used

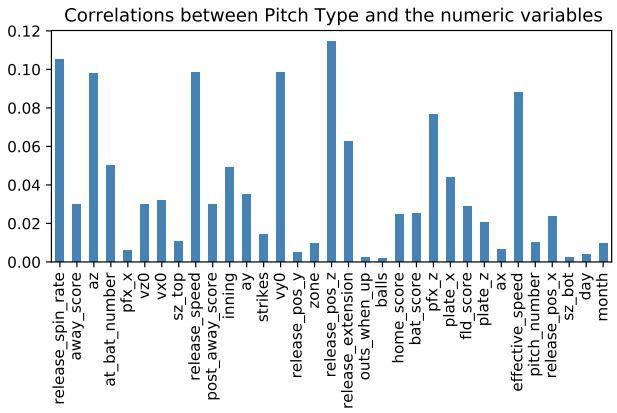

Figure 2. Correlations between pitch type and the numeric features.

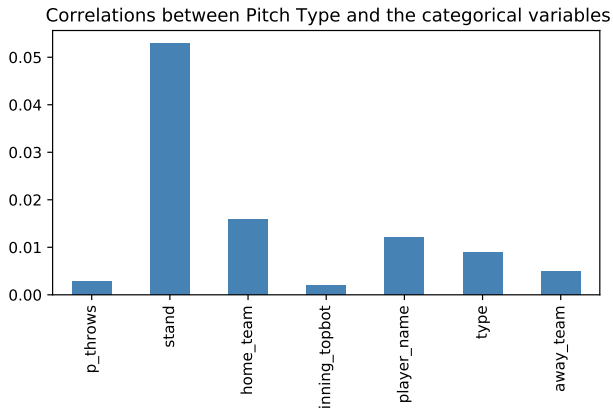

Figure 3. Correlations between pitch type and the categorical features.

personalized classifiers, so we expect that proper handling of the player_name variable may greatly affect our results as well.

A total of 742 pitchers threw at least one pitch during the 2016 MLB season. We used the binary encoder, provided by the category encoders package, as it required only 11 extra variables ($2^{11} = 2048 > 742$) to cover every pitcher, as opposed to one-hot-encoding, which despite seeming more natural, it would require the addition of 742 variables. After the binary encoding was completed, the total number of features increased to 69.

B. Classifiers

Our classification problem can be mathematically described as follows: We are given d-dimensional datapoints (features) $x = (x_1, \dots, x_d)$, and for each such point we try to classify it into its respective pitch type. Lets call the pitch categories $\omega_1, \dots, \omega_k$ for convenience.

The *Naive Bayes classifier* is a very simple but useful classifier which relies on the assumption that the feature vectors $x = (x_1, \dots, x_d)$ are independent, so $p(x \mid \omega_i) = \prod_{j=1}^{d} p(x_j \mid \omega_i)$ for each $i = 1, \dots, k$. It then picks the category ω_i that maximizes this expression. Of course, the independence assumption does not hold for our data, however the Naive Bayes Classifier has been reported to perform well even when this assumption is violated [KT08, Paragraph 2.5.7]. For our data, we consider the Gaussian Naive Bayes classifier, where we assume that each feature follows a normal distribution, with no covariance between itself and the rest of the features.

The *Nearest Neighbor Classifier* is also a popular choice. For each data point x that we need to classify, the algorithm finds the point of the training data set that is closer to it and classifies it to the same category. In a similar spirit, one may not only compare x to its nearest point, but to its k-nearest ones and classify it to the category that is represented the most among those k points. This is called the $k-NN$ classifier.

Support Vector Machines are another very common category of classifiers. They basically aim at separating the different categories by a line, or a hyperplane. When the data are not linearly separable, as is in our case, one can add additional dimensions to the data set, making it linearly separated in this newly created space. One can also strengthen this approach even more, by replacing the linear classifiers with non linear ones. This is done with the introduction of kernel functions. In our analysis we applied SVM for various kernel functions and compared the results.

Decision Tree Classifiers are perhaps the most popular rule-based classifiers. In order to classify a data point, they sequentially check each feature's value and assign it to a category accordingly. Their main advantage is their interpretability, as it is usually very easy even for non-experts, to understand the logic behind each classification.

A main issue with them is that they often exhibit overfitting, due to the complete exhaustion of the feature space after a long sequence of rules. To avoid that, we often use *Random Forest Ensembles*, which consist of a set of randomized decision trees, each of which may overfit the problem, but when combined together they tend to eliminate this trait.

Adaboost is an example of a meta-classifier. It consists of a base classifier which one fits to his data set and then applies the same process repeatedly, each time assigning different weights to the data points, giving emphasis to the misclassified examples.

Multi-layered Perceprtons (MPLs) learn functions $f : \mathbb{R}^d \to \mathbb{R}$ of the form $f(x) = Tx$, where

$$T = A_{m+1} S_m A_m \cdots A_2 S_1 A_1$$

is an operator consisting of sequential compositions of affine transformations A_i and activation functions S_i. In the special case of a single-layered network, f has the form $f(x) = \sum_{j=1}^{N} a_j \sigma(w_j^\intercal x + \theta_j)$ where $N \in \mathbb{N}, a_j \in \mathbb{R}, w_j \in \mathbb{R}^d$ and $\theta_j \in \mathbb{R}$. They can be used for classification problems by learning a function $f : \mathbb{R}^d \to \{\omega_1, \dots, \omega_k\}$, which assigns each point to its respective category.

C. Implementation

Due to the pitch types being extremely imbalanced, we increased our dataset artificially using oversampling with the help of imbalanced-learn's RandomOverSampler function.

Classifier	Accuracy	Training Set
Gaussian Naive Bayes	0.60	full
Decision Trees	0.91	full
Random Forests	0.92	full
kNN	0.79	sampled
Adaboost	0.52	sampled
MLP	0.95	sampled
SVM (linear)	0.58	sampled
SVM (RBF)	0.92	sampled

Table I
Accuracy summary for the classifiers used.

Accuracy	SVM Linear	SVM rbf	Random Forest
mean	0.964	0.956	0.971
median	0.972	0.960	0.976
std	0.034	0.032	0.028
max	1.000	1.000	1.000
min	0.783	0.825	0.781

Table II
Accuracy aggregates on the test set for three different types of personalized classifiers.

This way, all pitch types where equally represented in our training set.

The Naive Bayes Classifier was implemented using the Scikit-learn [PVG11] GaussianNB module and achieved an overall acurracy of 60%. The Decision Tree classifier was performed using the scikit learn DecisionTreeClassifier function. The optimal parameters, obtained using 10-fold cross validation and grid-search, resulted in an accuracy of 91%. The Random Forest Ensemble, implemented through the RandomForestClassifier, achieved a slightly better accuracy of 92%.

Due to the large number of observations, especially after taking into account the oversampling, the rest of the classifiers required a much slower training procedure. To reduce the training times, we reduced our initial data set using weighted random sampling and then we continued with the same procedure as before.

The optimal parameter for the Nearest Neighbors Classifier was found to be $k = 11$, after performing a 10-fold cross validation on the sampled dataset, and achieved a 79% accuracy. The SVM classifier with an rbf kernel achieved a 92% accuracy with the optimal parameter found equal to $C = 10$. The highest accuracy (95%) was achieved by the Multi Layer Perceptron Classifier having the $f(x) = \tanh(x)$ as the activation function and with two hidden layers of 400 neurons each. Adaboost and LinearSVC performed much worse than the rest of the classifiers (Table I).

D. Classifiers personalized to each pitcher

In our previous analysis, the pitcher name was included as a feature in our dataset, so we have been training one model that should be able to classify effectively any pitcher. To improve our classification results, we examined whether it would make a difference if we trained a distinct classifier for each pitcher. This is the standard followed by the official MLB classification and has been proven to give significantly better results.

1) Procedure: Initially we trained all the previous classifiers without cross-validation and examined which were the most promising ones. These turned out to be the random forest classifier and the SVM's for rbf and linear kernels. We then partitioned the full data set into smaller ones, one for each pitcher, and repeated the same pipeline as used in the original data set (normalization, oversampling and cross-validation).

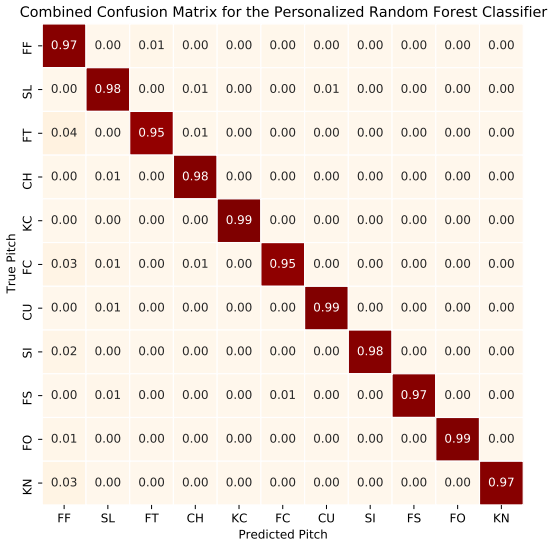

Figure 4. Combined confusion matrix for the pitcher specialized random forest classifiers.

We focused on pitchers with at least 60 thrown pitches, so that we had enough data to build each classifier.

Due to the large number of pitchers, thus the large numbers of distinct classifiers, we had to compute some aggregate scores to assess the results. In Table II we present a summary of the accuracy for each of the classifiers. The random forest outperforms the rest in terms of mean accuracy. We added all the classification results from every pitcher to create a single confusion matrix (Figure 4).

2) Interpretation: In Figure 5 we present the most important features for the random forest classifier. In descending order of importance, they can be grouped together into the following categories: 1) The pitch velocity (represented by three highly correlated variables vy0, release_speed, effective_speed, 2) the pitch acceleration (ax, az, ay), 3) the pitch break (pfx_x, pfx_z) and 4) the pitch spin rate.

Neither of these is particularly surprising as they are all highly important features when analysing the kinematic characteristics of the ball. What is rather interesting is the extremely low importance of balls and strikes, as even the month or the day of the game seem to be more important to the classifier. Balls and strikes largely influence pitch selection. As it is shown in [ST18], they are the second and third most important features when it comes to pitch prediction, namely the correct prediction of the pitch to come using only information available before the throw. However, in the

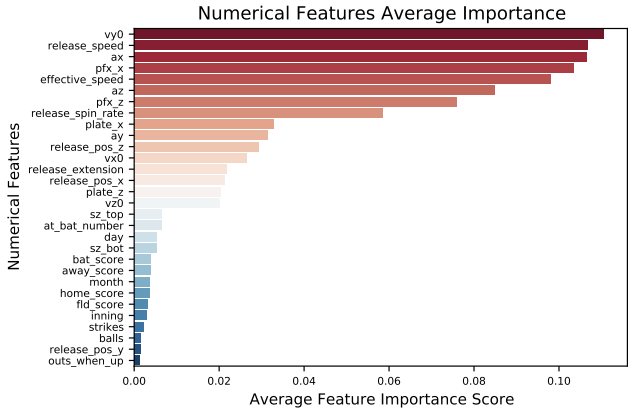

Figure 5. Average feature importance for the numerical features in the personalized pitcher random forest classifiers.

present of the kinematic information of the pitch, they no longer seem particularly useful.

Although the random forest classifier achieved a high mean accuracy, further investigation is required to identify the specific pitchers on which it may have failed to give satisfactory results, and the reasons why this might have happened. In Figure 8 we see the dependence of the random forest classifier on the number of pitches and the pitcher's hand. Evidently, the fewer the number of pitches thrown, the smaller the sample, so the comparatively smaller accuracy of some pitchers can be partially attributed to the sample size on which they were trained.

On the other hand, pitcher's handedness does not seem to affect our classification capabilities as both categories score an average of 0.971 accuracy (based on a sample of 183 LHP pitchers and 488 RHP). Pitcher heights[3] also did not seem to affect the accuracy, as the mean accuracy varied between 0.97 and 0.99 among all categories.

The class of pitchers that achieved relatively lower accuracy despite being trained on a sufficiently large dataset requires some special attention. These were Shane Greene, Nick Martinez, Vance Worley, Michael Lorenzen, Eric Surkamp, Jake McGee, Jerry Blevins, Jesse Chavez, Mat Latos and Raisel Iglesias.

Among them, Vance Worley had the highest pitch count (1209) and the third worse accuracy (90.6%). A graphical representation of his pitches reveals significant overlaps between the different pitch types (Figures 6 and 7). A look at his confusion matrix confirms the two sources of errors: Firstly, his slider was often misclassified as a cutter and vice-versa. Secondly, his four-seam fastballs were often misclassified into any other of the pitch types he threw, except for his curveball.

We will let Worley himself provide an explanation, taken from a 2013 fangraphs interview, when he was asked to describe his best secondary pitch:

[3]We added the pitchers heights in our data frame using another pybaseball query, this time from the Lahman database.

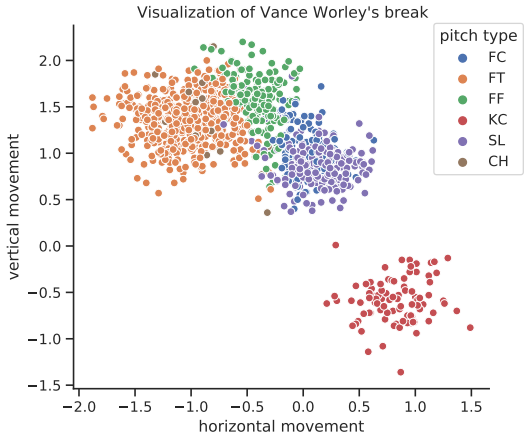

Figure 6. Vance Worley's pitch break. His cutters (FC) and sliders (SL) break almost identically. Notice also the overlap between his four-seamer (FF) and his two-seam fastball (FT).

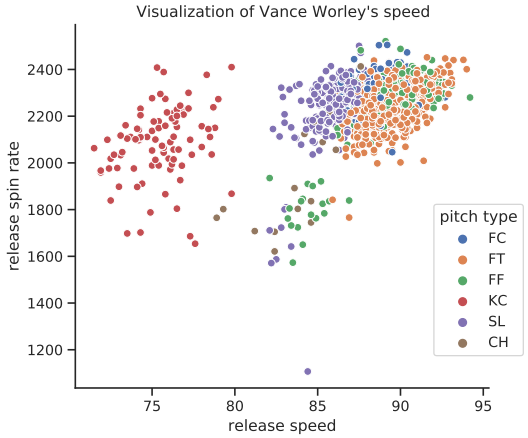

Figure 7. Vance Worley's pitch speed. His sliders are generally slower than his cutters, but there is still a significant overlap, not only between them, but also between his four-seam and two-seam fastballs.

> Probably my cutter. [...] You can call it whatever you'd like. The pitch has two-plane action. It slides and it falls out. It is basically a cutter that morphed into a slider. I throw it the way you're supposed to throw a cutter, but it moves like a slider. Everybody is different. Everybody has different arm slots when they pitch, and that's how mine breaks.

During the same interview he also mentions how he deliberately throws his fastballs slower at times, to mess with hitters' timing. Given the importance of release speed and break in the random forest classifier, it is not a surprise it had a hard time distinguishing between some of these pitches: They look almost identical and their official classification is based on the pitcher's grip, on which we have no information.

In [WDS20] and [PR18], pitch classification was based solely on the pitcher's body motion, but with poor results. The ball's grip is often obscured in the videos, as the pitcher hides the ball in his glove for as long as possible to avoid exposing the incoming pitch to the hitter. Between this and the actual delivery, the ball is visible only for a split second

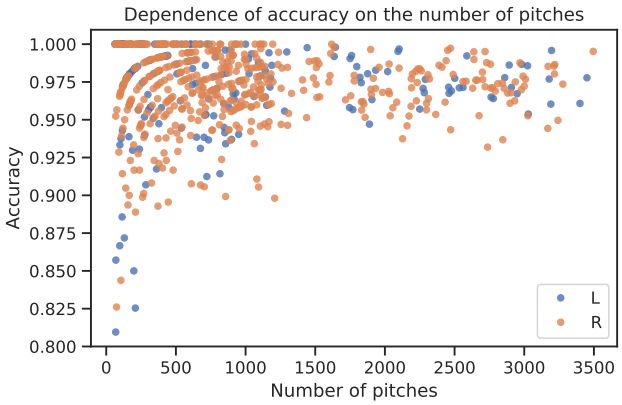

Figure 8. Dependence of the accuracy of the personalized random forest classifier on the number of pitches. Right handed pitchers are depicted with orange, whereas left handed pitchers with blue.

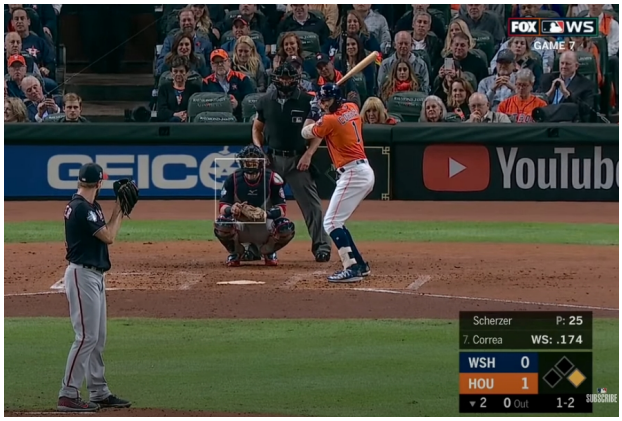

Figure 9. A typical instance from a baseball broadcast as the pitcher prepares to throw the ball to the catcher. The strike zone extends between the batters knees and the midpoint between his shoulders and his belt.

during the pitching motion, but even then the pitcher's arm moves so quickly that distinguishing the actual grip and the ball seams is challenging to say the least. The body motion can also be deceiving as the pitcher tries to deliver his pitches with a similar motion regardless of the pitch type.

It may not be an easy task, but enhancing the kinematic features of the ball with additional information concerning the grip could potentially improve our model even further.

IV. Conclusions

We approached the pitch recognition problem using several classifiers. Among them, the decision trees (91%), the random forest (92%), the SVM with RBF kernel (92%) and the MLP (95%) classifiers achieved the greater accuracy. To increase the performance even more, we then trained different classifiers for each pitcher. This indeed increased the accuracy significantly, reaching up to 97% for the random forest classifier.

Focusing our attention on the random forest, we managed to extract the features which are most relevant to our problem and we explored the reasons why some pitchers were more difficult to recognize than others. Although we achieved an extremely high accuracy, we strongly suspect that we could do even better by considering information additional to the kinematic characteristics of the pitch thrown, in particular an automated way to recognize the pitcher's grip on the ball.

Appendix

The game of baseball is dictated by a relatively large number of rules and may often seem intimidating to watch for the casual fan. However, only knowledge of a handful of them is required in order to be able to watch a game and completely understand what is going on.

Most of the action is evolving over the so-called pitcher vs batter matchup/duel (see Figure 9). The player on the left is called the *pitcher* and his goal is to throw the ball to his teammate, called the *catcher*. Between them interferes a player from the opposing team, called the *batter*, whose goal is to hit the ball with his bat, thus not allowing the catcher to obtain possession of the ball.

The mini game of the pitcher-batter duel is more commonly called an *at-bat*. The pitcher's main goal is to send the ball through an imaginary rectangle, called the *strike zone*. Its width is fixed and equal to 17 inches, however its height extends between the batter's knees and torso in height, and may vary between batters. Batters with a more closed stance have smaller strike zones.

During the at-bat, points are awarded to each side for every successful effort: Each time the pitcher successfully pitches the ball to the catcher through the strike zone, he is awarded one point (strike). If he misses the strike zone and the batter does not swing his bat, then the batter is awarded a point (ball). If the pitcher misses the strike zone but the batter swings his bat unsuccessfully (swing and a miss), then the pitcher is awarded with a point (strike).

There are two main ways for the duel to end. The first one is for the batter to hit the ball into play. In this case, the strike-ball score is irrelevant and the result of the duel is determined by where the ball ends up. The second way is when the pitcher or the batter runs out of balls or strikes respectively. A batter is allowed to make at most 2 strikes, so a third strike by the pitcher immediately ends the at-bat in his favor. The pitcher is allowed at most 3 balls, so a fourth ball ends the at-bat in favor of the batter.

In general there are several different types of pitches a pitcher can throw (see Figure 10). Despite the large number of pitch types and subtypes, they can all be roughly broken down into three major groups: Fastballs, breaking balls and off-speed pitches.

- *Fastballs* are the most commonly used pitch types and usually every pitcher knows how to throw at least one type of them. A Four-Seam Fastball is characterized by a high velocity (80-100mph) and little to no movement. It is usually thrown inside the strike zone with the intention of challenging the batter's reflexes. Slightly slower fastballs with small downwards or side movement are called Two-Seam Fastballs and Cutters respectively.
- *Breaking balls* are typically much slower than fastballs,

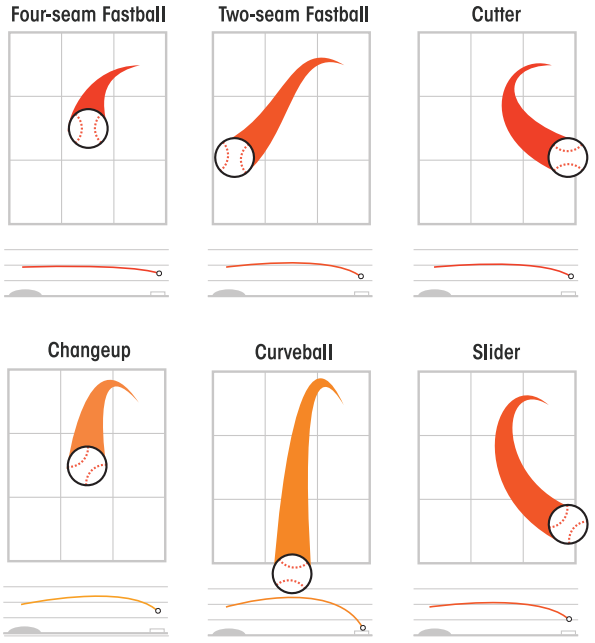

Figure 10. Trajectories [Dha] for six of the most common pitch types thrown by a right handed pitcher, as seen from the catcher's perspective.

but with a much higher movement. A Curveball (70-80mph) breaks with a direction from top to bottom. A Slider is typically faster (80-90mph) and will break both downwards and sideways towards the direction of the inside of the pitchers hand. The main intention when throwing a breaking ball is not to find the strike zone, rather than to lure the opposing hitter into chasing a ball which will eventually turn away from him, inducing a swing and a miss.

- *Off-speed* pitches (Changeup) are pitches which are designed to look exactly like fastballs, but thrown at a much lower velocity (70-85 mph). They aim to trick the batter by making him expect a fastball, thus swinging the bat earlier than he should, resulting to a swing and a miss. They may, or may not have any movement, depending on the pitcher's technique.

Most pitchers have a rather limited but specialized arsenal which they can reliably use at the highest level. Instead of learning to throw every pitch type adequately, they focus on only a few specific pitch types which they systematically try to master and polish. Depending on their role in the team's rotation, they may occasionally add pitch subtypes to their repertoire to become more diverse.

References

[ADG13] A. Attarian, G. Danis, J. Gronsbell, G. Iervolino, H. Tran, A comparison of feature selection and classification algorithms in identifying baseball pitches, *Proceedings of the International MultiConference of Engineers and Computer Scientists* **1**, 2013. URL: researchgate.net/publication/237044046, cited on p. 1

[Auc19] D. Aucoin, Calling the right pitch: Investigating effective velocity at the MLB level, URL: www.drivelinebaseball.com, cited on p. 2

[BS] Baseball Savant, Statcast search CSV documentation, URL: baseballsavant.mlb.com/csv-docs, cited on p. 2

[Boc15] J. Bock, Pitch Sequence Complexity and Long-Term Pitcher Performance, *Sports* **3**(1), pp. 40-55, 2015. DOI: 10.3390/sports3010040, cited on p. 1

[Dha] L. Dhakar, Baseball pitches illustrated, URL: lokeshdhakar.com/baseball-pitches-illustrated, cited on p. 7

[KT08] K. Koutroumbas, S. Theodoridis, *Pattern Recognition*, Academic Press, 2008. ISBN: 978-1-59749-272-0 Cited on p. 3

[Led17] J. LeDoux, *Introducing pybaseball: an open source package for baseball data analysis*, 2017. URL: jamesrledoux.com/projects/open-source/introducing-pybaseball GITHUB: github.com/jldbc/pybaseball Cited on p. 2

[Moo19] E. Moore, A potential alternative for public pitch classification, *Baseball Prospectus*, 2019 URL: baseballprospectus.com, cited on p. 1

[PVS13] M. Pane, S. Ventura, R. Steorts, A. Thomas, Trouble with the curve: Improving MLB pitch classification, 2013. PREPRINT: arXiv:1304.1756, cited on p. 1

[PS10] R. Paul, C. Schwartz, Real time pitch classification, *U.S. Patent 12/696,577*, 2010 URL: US8876638, cited on p. 1

[PVG11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine Learning in Python, *Journal of Machine Learning Research*, 2011. URL: https://www.jmlr.org/papers/, cited on p. 4

[PR18] A. Piergiovanni, M. Ryoo, Fine-Grained Activity Recognition in Baseball Videos, *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018. DOI: 10.1109/CVPRW.2018.00226, cited on p. 5

[SS20] C. Schwartz, S. Sharpe MLB pitch classification, *MLB Technology Blog*, 2020. URL: technology.mlblogs.com/mlb-pitch-classification-64a1e32ee079, cited on p. 1

[ST18] G. Sidle, H. Tran, Using multi-class classification methods to predict baseball pitch types, *Journal of Sports Analytics*, **4**(1), pp. 85-93, 2018. DOI: 10.3233/JSA-170171, cited on p. 1, 4

[UYN20] K. Umemura, T. Yanai, Y. Nagata, Application of VBGMM for pitch type classification: Analysis of TrackMan's pitch tracking data, *Japanese Journal of Statistics and Data Science*, 2020. DOI: 10.1007/s42081-020-00079-8, cited on p. 1

[WDS20] N. Wiedemann, C. Dietrich, C. Silva, A tracking system for baseball game reconstruction, 2020. PREPRINT: arXiv:2003.03856, cited on p. 1, 5