Baseball Pitch Prediction Model

Matthew Spitz
MS, Data Science Institute
Columbia University
mrs2299@columbia.com

Abstract—**This study investigates the effectiveness of predicting the pitch type of pitches thrown in Major League Baseball (MLB) competition. Leveraging Google's BigQuery data warehouse, event-level data from the 2016 MLB season provided by Sportradar LLC was accessed to extract the main features and target label of the model. Apache Spark's machine learning library is at the core of the research design with a gradient boosted tree binary classifier utilized to predict whether a pitch is a fastball or breaking ball. Lack of training data from subsequent seasons and depth of pitch-specific advanced metrics leaves room for improvement in the current 62% accuracy of the classifier. Nevertheless, results of the model are in line with other relevant works and are encouraging that accurate pitch prediction is feasible with high probability; however, real-world application will prove challenging due to latency and player training issues.**

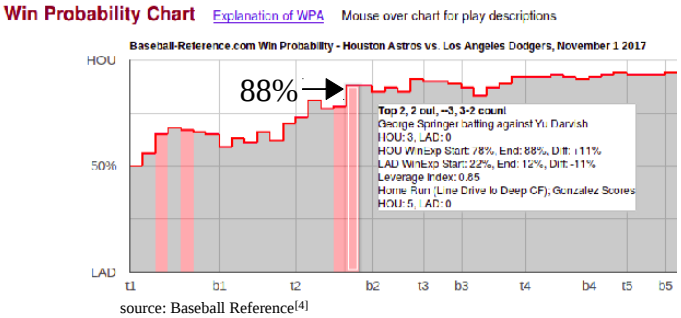
I. Introduction

The motivation behind predicting pitches in MLB competition is the strategic advantage gained by the player who is batting if this information were available to the player before the pitch is thrown. Baseball is a sport that comprises a series of encounters between a pitcher and a hitter. Inherently the pitcher has the advantage in these match-ups as they know the speed, location, and spin rate of the pitch they are planning on throwing. Whereas the batter has only 0.4 seconds once the pitch leaves the pitcher's hands and before it crosses the plate to make a decision with 0.25 of these seconds required just to see the ball and react[1]. This leaves the batter with minimal time to compute all the aforementioned factors needed to determine if and how to swing the bat. As a result, one would hypothesize that a batter would benefit from any reliable information about the pitch before it is thrown.

An indicator that supports the merits of how accurate results from this research could benefit the batter is seen when pitchers are caught "tipping" their pitches. Tipping pitches occurs when the pitcher through some inadvertent repeated action is indicating what type of pitch he or she is intending to throw. To avoid this, it is incredibly important for the pitcher to repeat the same motions in the same manner for the delivery of every pitch while they are on the mound pitching. If the batting team is able to decode an abnormal action in the pitcher's delivery that consistently appears, the pitcher may be "tipping" their pitches. This swings the strategic advantage lever back into the batter's favor. The pitcher in this situation loses its inherent advantage because now the batter has data on the upcoming speed, location, and spin rate of the pitch.

The idea of "tipping" pitches came into play on the game's biggest stage, the World Series, this past season in 2017. Los Angeles Dodgers pitcher Yu Darvish was caught "tipping" his pitches in game 3 of the series and as result the opposing team, the Houston Astros, "won" 86% of the at-bats while he was on the mound. Of the pitches Mr. Darvish threw, 86% of them were either balls or the batter made contact with the pitch. This resulted in 4 earned runs conceded on 6 hits in only 1 and 2/3 innings pitched. In game 7, Yu Darvish again started the game and once again could not make it out of the second inning due to the competitive advantage gained by the Astros. For the series, the Astros hit 0.556% against Darvish's off-speed pitches, which represents an extremely poor performance when compared to his typical benchmarks[2]. For example, Darvish's batting average against for the entire season was only 0.288% across all pitch types, ranking Darvish 13th best in the league in this statistic[3].

How does this poor performance impact the difference between winning and losing? Look no further than the win probability chart featured below of the aforementioned World Series game, the one in which Yu Darvish was caught "tipping" pitches[4]. At the start of the 1st inning each team had a 50% chance of winning the game, after the 1 and 2/3 innings pitched by Yu Darvish the Astros win probability skyrocketed to 88%. Thus, illustrating the significant potential benefits of creating an accurate, reliable pitch prediction model.



source: Baseball Reference[4]

II. Related Works

In the private sector, it is assumed that professional teams in MLB are working on this research and have created their own pitch prediction models. However, since there is no competitive advantage of sharing this work, it is not available to the public for analysis.

Meanwhile in the public sector, there have been a few studies published regarding pitch prediction. One such study was conducted by Gartheeban Ganeshapillai and John Guttag of MIT in 2012[5]. This study used a linear support vector machine classifier with soft-margin as the model to predict pitches, building a unique model for each pitcher. For features, the model utilizes information about the current situation of the game as well as data about the pitcher's prior tendency to throw a given pitch in a given situation. The most important feature noted in the study was the previous pitch history between a specific pitcher and hitter. Each pitch type was analyzed independently where the model performs a binary classification, predicting yes if the pitch is of the type being analyzed or no if it's not that specific pitch type. This study received results of approximately 70% accuracy on average across pitchers when predicting if the pitch was a fastball or not. These results performed optimally when compared to the approach of predicting pitches simply based on the tendency of a pitcher to throw that pitch type historically.

Another pitch prediction study produced by Noah Woodward used a decision tree machine learning model to make pitch type predictions[6]. The features of the model were broken into three different categories, game situation, historical information, and data on the batter. Similar to the previous study, a different model was created for each pitcher. However, the model did not perform binary classification, but instead predicted pitches across multiple pitch types. The generated results produced an accuracy rate of approximately 50%. An important feature uncovered by the study was the average amount of pitches seen per game by a hitter. For at least one pitcher in the study, Justin Verlander, this feature was a critical component of its decision tree and thus in predicting pitch type.

III. System Overview

The system design of this study consists of extracting features from two data sources that feed into the machine learning model implemented in Spark, which is used to make pitch type predictions. In order to utilize two different data sources in the construction of the feature set, the design required matching the datasets on player names. To overcome potential inconsistencies in player name spellings, the python package, difflib, was used to match players whose names are spelled slightly differently between the two datasets. This was a critical step to maintain data integrity of the final feature set used by the model.

The primary data source of this research study comprises event-level Major League Baseball data provided by Sportradar LLC[7]. This dataset contains every pitch from every game of the 2016 Major League Baseball season. The data is accessible via Google's BigQuery data warehouse in a structured format across two tables, one for the regular season and another for the postseason. The schema of some of the elements of the regular season table is featured below. Combined, the size of this data is approximately 1.8 GB, consisting of 145 pieces of information per pitch. The data was queried and extracted using Google's BigQuery UI front end, but the data is available via a command-line tool and through the BigQuery Rest API as well. Once the data was extracted, it was stored in Google Cloud Platform's storage application. The data was then saved locally as a CSV file. At this point, a python script was created to extract the features for the machine learning model.

Schema of Regular Season Data Table

Table Details: games_wide

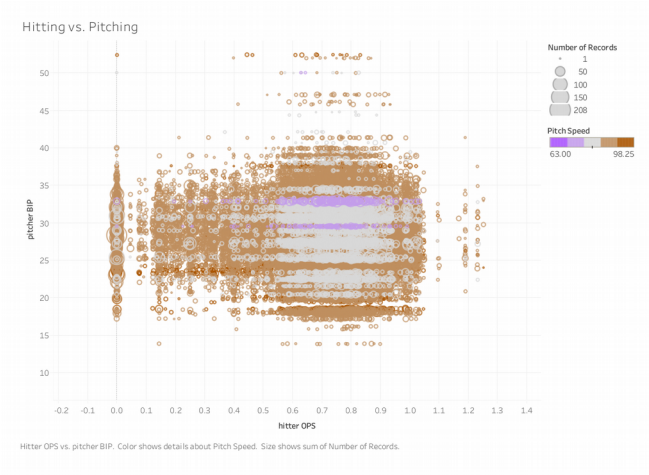
Schema | Details | Preview

gameId	STRING	NULLABLE	Describe this field...
seasonId	STRING	NULLABLE	Describe this field...
seasonType	STRING	NULLABLE	Describe this field...
year	INTEGER	NULLABLE	Describe this field...
startTime	TIMESTAMP	NULLABLE	Describe this field...
gameStatus	STRING	NULLABLE	Describe this field...
attendance	INTEGER	NULLABLE	Describe this field...
dayNight	STRING	NULLABLE	Describe this field...
duration	STRING	NULLABLE	Describe this field...
durationMinutes	INTEGER	NULLABLE	Describe this field...
awayTeamId	STRING	NULLABLE	Describe this field...
awayTeamName	STRING	NULLABLE	Describe this field...
homeTeamId	STRING	NULLABLE	Describe this field...
homeTeamName	STRING	NULLABLE	Describe this field...

source: Data provided by Sportradar LLC via Google BigQuery data warehouse[7]

A complementary dataset was introduced to enrich the feature set of the model. Baseball Reference was the source of this data, providing the model with statistics measuring the quality of both the batter[8] and the pitcher[9]. If one were to only use the Sportradar event data there would be no features that provide context into the quality of the players. As a result, in order to give the model this much-needed context, pulling a secondary data source like Baseball reference was necessary. For example, to better understand the batter, the player's 2016 average on-base

plus slugging % (OPS) was included in the feature set. This metric measures both the ability for a batter to get on base as well as quantify the number of bases the batter reaches per hit. According to MLB.com, "OPS is widely considered one of the best evaluative tools for hitters"[10]. The potential impact of this feature on the model can be seen in the chart below that plots this statistic against another feature of the model that quantifies pitcher's performance based on the percentage of balls hit in play against them compared to the amount of strikes they throw. Since pitch types correlate with pitch speed, the variation in pitch speed illustrated in the chart amongst different combinations of these two metrics support the value of features that capture the quality of the player. A scrapping function was not needed to pull this data as Baseball Reference has a convenient way to save a data table as a CSV file locally. A few python scripts were created to clean this data to correct for name syntax issues as well as to eliminate duplicate names for the times when players who played for multiple teams in 2016 were listed more than once in the dataset.



The feature set is broken down into 3 distinct categories, the current game situation, information concerning the pitcher, and information concerning the batter. In the following tables, you can find the 20 selected features included in this study divided into these categories. The research is not assuming that this feature set is exhaustive, but given the information available in the two datasets these features were sensible to include in order to cover the aforementioned 3 categories, which represent the critical components that go into the decision-making process for a pitcher when deciding the type of pitch to throw. Modifying this feature set is on the agenda for future work to improve the model. The first column of these tables indicates the data source used to extract the feature with 'SR' representing Sportradar and 'BR' representing Baseball Reference and the second column indicates the feature itself. The features with an (*) marked next to its name are all metrics that are computed as an average over the entire 2016 season. As a result, these metrics will not change for a specific batter or pitcher throughout the dataset. Ideally, these metrics should differ per specific batter or pitcher throughout the season and instead be the average over the games played up until the point in the season corresponding to the time in which the pitch event occurred. However, due to limitations of the datasets this was not possible. Thus, the study is assuming that the season long averages serve as good approximations for what the current average would have been at the specific point in the season when the event occurred.

Game Situation Features

SR	Inning
SR	Number of Batters Faced in Inning
SR	Number of Pitches Thrown to Current Batter
SR	Number of Pitches Thrown in Game
SR	Number of Outs in Inning
SR	Number of Balls in Count
SR	Number of Strikes in Count
SR	Run Differential
SR	Venue

Pitcher Features

SR	Pitcher ID
SR	Pitcher Handedness
SR	Average Speed of Pitcher's Pitch*
SR	Average Location of Pitcher's Pitch*
BR	Average Ball in Play Percentage of Pitcher*

Batter Features

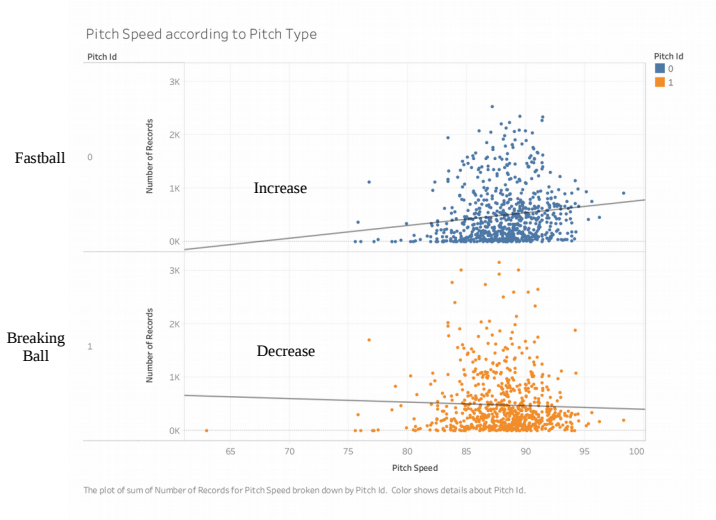
SR	Batter ID
SR	Batter Plate Side
BR	Average OPS of Batter*
BR	Average OPS of Team Batting*
SR	Batter Height
SR	Batter Weight

IV. ALGORITHM

To make the pitch type predictions using the previously described feature set, the study uses a gradient boosted tree classifier as its machine learning model. This machine learning method groups decision trees that are trained iteratively to minimize a loss function[11]. Starting with a weak model, the algorithm chooses a "smarter" model over each iteration, using an optimization approach by minimizing the cost of inaccurately predicting pitch types[12][13]. In this case, supervised learning is performed

as the target value, the pitch type, is available in the training set, which consists of 80% of the data. Meanwhile, the testing hold-out set includes the remaining 20% of the data. This machine learning model was implemented in Spark's MLlib library, specifically coded in PySpark, using the default parameters.

Specifically binary classification was performed by the model, predicting whether each pitch was a fastball or a breaking ball. This type of classification was chosen for the study based on the less than one second response time that the batter has to make a decision on the pitch. Thus, the batter typically only has time to discern if the pitch is a fastball or breaking ball. The difference between the speed of these two pitches is the primary reason for this, which can be seen in the below plot. This plot illustrates the number of fastballs vs. breaking pitches that were observed in the study based on pitch speed. As you can see the trend line for the fastball chart is increasing, while the trend line for the breaking ball chart is decreasing. This highlights the fact that fastballs on average are faster pitches when compared to breaking balls and thus illustrates the different calculus batters need to compute in their heads depending on pitch type.

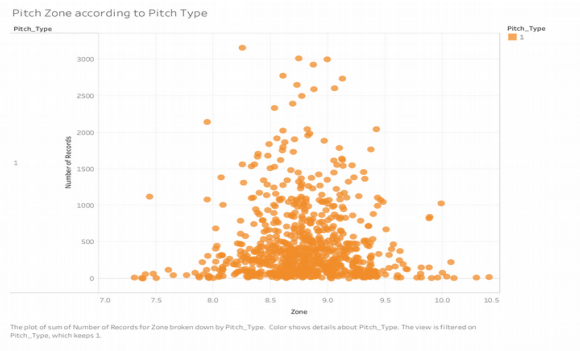


After training the model, the value of the selected 20 features could be observed. Using Spark's built in 'featureImportances' method, each feature's importance was measured by its average importance across all trees in the model with the resulting importance vector normalized to sum to 1[14]. The resulting scores for each feature can be found in the following table. The importance of the features naturally broke into three tiers that corresponded fairly well with the 3 different categories of features extracted in the study. The features quantifying pitching performance are most important to the model, followed by the game situation features, and the least important features are the data points capturing information on the batters. These results on the surface jibe with conventional wisdom in the sense that location and speed of pitches correlate well with pitch type.

Feature Importance Scores

Category	Feature [vector index]	Score
P	Avg Location of Pitcher's Pitch [14]	.2636
P	Avg Ball in Play % of Pitcher [16]	.1826
P	Avg Speed of Pitcher's Pitch [13]	.1664
P	Pitcher ID [15]	.1384
P	Pitcher Handedness [12]	.0536
G	# of Strikes in Count [18]	.0416
G	Inning [8]	.0390
G	# of Pitches Thrown in Game [11]	.0280
G	# of Pitches Thrown to Batter [4]	.0262
G	# of Balls in Count [0]	.0203
B	Avg OPS of Batter [3]	.0195
G	# of Batters Faced in Inning [9]	.0088
B	Batter Height [5]	.0035
B	Batter ID [2]	.0027
B	Avg OPS of Team Batting [1]	.0020
G	Run Differential [17]	.0016
G	Venue [19]	.0013
B	Batter Plate Side [6]	.0008
B	Batter Weight [7]	.0003
G	# of Outs in Inning [10]	NA

For example, the average location of a pitcher's pitch, the most important feature of the model, will vary based on pitch type where breaking pitches tend to cross the plate at a lower point when compared to fastballs, as breaking pitches on average break away from pitchers in a downward direction. This is illustrated in the below plot where the vast majority of breaking pitches are thrown by pitchers whose average pitch location is between 8.5 and 9.5, which corresponds to the bottom of the strike zone.



On the other side of the spectrum, information on the batter is not critical to the model. One reason why this feature analysis may be devaluing batter data simply may be the result of a lack of training data. For example, if a certain batter only faces a specific pitcher once in the data set, it will be difficult for these features to impact the model's pitch type prediction. As a result, to get more relevant batting features, historical data on specific batting-pitching relationships would likely be helpful.

V. Software Package Description

The software package produced by this research allows any user to input their own set of feature data and in turn have the model predict the pitch type that the pitcher will throw given the user inputted data.

To run this software, the user has to first run a Python pre-processing script called main_pre-processing.py to clean the data from the two aforementioned sources and extract the necessary features to train the model. However, feature extraction only needs to be performed once and then the resulting CSV file can be used to train the model in future software runs when making predictions. As a result, the program will prompt the user if they would like to extract features; if the user answers no then they can move past this step and start entering their own feature data to predict on without wasting time re-acquiring the features.

```
matthew@datavizsc:~/Dropbox/Documents/Columbia/2017-18_Fall/2-BigData/HW/Project$
python main_pre-processing.py
Would you like to extract the features of the model? Answer Yes or No:
```

After pre-processing, the user will input the feature data that they would like to predict on. The terminal will prompt the user with questions that will correspond to each feature of the model. For example, to predict pitch type the first question asks the user to enter the full name of the pitcher. If a user enters an invalid player name, the program will ask for a new name, until it gets a valid one. Below is a screen shot of all the questions the user needs to answer.

```
Would you like to extract the features of the model? Answer Yes or No: no
Would you like to predict pitch type for a specific event? Answer Yes or No: yes
What pitcher would you like to predict pitch type for? (Enter Full Name): Justin Verlander
What hand does the pitcher throw with? (Enter Right, Left, or Both): Right
How many pitches has the pitcher thrown thus far in the game? (Enter integer number): 10
Who is the batter that the pitcher is facing? (Enter Full Name): David Wright
What side of the plate does the batter hit from? (Enter Right, Left, or Both): Right
How many pitches has the batter faced thus far in the at bat? (Enter integer number): 3
What is the weight of the batter? (Enter integer number): 180
What is the height of the batter? (Enter integer number): 70
What inning is the game in? (Enter integer number): 3
How many batters has the pitcher faced in the inning, including the current batter? (Enter integ
er number): 2
How many balls has the pitcher thrown thus far in the at bat? (Enter integer number): 0
How many strikes has the pitch thrown thus far in the at bat? (Enter integer number): 1
How many outs are there in the inning currently? (Enter integer number): 0
How many runs is the pitching team winning or losing by? (Enter 0 for tie and negative integer i
f losing and postive integer if winning): 2
What is the average OPS of the batting team? (Enter float number): .800
What is the average OPS of the batter? (Enter float number): .840
What is the average Balls in Play % of the pitcher? (Enter float number): 29.1
What is the average speed of the pitcher's pitches? (Enter float number): 92.3
What is the average location of the pitcher's pitches? (Enter float number): 6.3
What stadium is the game being played in? (Enter full venue name): Citi Field
```

After the user inputs their data, then a new Python main function called main_model.py needs to be run in Spark. To run this script in terminal, a user can utilize the 'spark-submit' executable file with two arguments that need to be fed into the command. For each argument the user must enter either 'yes' or 'no'. The first argument should be 'yes' if the user has not previously trained the model, since this script converts the pre-processed data into a new vector form compatible with Spark. If this step has already been completed then the user can input 'no' for the first argument to save time. Meanwhile, the user should always input 'yes' for the second argument, if they are indeed testing the unique feature data they previously inputted in the prior step. Below is an example of how to run this code.

```
matthew@datavizsc:~/.linuxbrew/Cellar/spark-2.2.0-bin-hadoop2.7/bin$ ./spark-submit --master local /home/matthew/Dropbox/Documents/Columbia/2017-18_Fall/2-BigData/HW/Project/main_model.py no yes
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
17/12/19 18:57:49 INFO SparkContext: Running Spark version 2.2.0
17/12/19 18:57:49 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
17/12/19 18:57:50 INFO SparkContext: Submitted application: BigData
17/12/19 18:57:50 INFO SecurityManager: Changing view acls to: matthew
17/12/19 18:57:50 INFO SecurityManager: Changing modify acls to: matthew
17/12/19 18:57:50 INFO SecurityManager: Changing view acls groups to:
17/12/19 18:57:50 INFO SecurityManager: Changing modify acls groups to:
17/12/19 18:57:50 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users  with view permissions: Set(matthew); groups with view permissions: Set(); users  with modify permissions: Set(matthew); groups with modify permissions: Set()
17/12/19 18:57:50 INFO Utils: Successfully started service 'sparkDriver' on port 33091.
```

This program will first create a dataframe in Spark with the features inputted by the user, as seen below.

```
+-----------+--------------+------+----------+-----------+----------+-----------+
----------+------+-------------------+----------+-----------+----------+--------+-
----------+----------+-------+-----------+--------+-------------+-----+
|balls_count|batting_tm_OPS|hitter|hitter_OPS|hitter_count|hitter_hgt|hitter_side|
hitter_wgt|inning|inning_batter_count|outs_count|pitch_count|pitch_hand|pitch_id|p
itch_speed|pitch_zone|pitcher|pitcher_BIP|run_diff|strikes_count|venue|
+-----------+--------------+------+----------+-----------+----------+-----------+
----------+------+-------------------+----------+-----------+----------+--------+-
----------+----------+-------+-----------+--------+-------------+-----+
|          0|           0.8|   458|      0.84|          3|        70|          1|
        180|     3|                  2|         0|         10|         1|     100|
       92.3|       6.3|    490|       29.1|          2|         1|    1|
+-----------+--------------+------+----------+-----------+----------+-----------+
----------+------+-------------------+----------+-----------+----------+--------+-
----------+----------+-------+-----------+--------+-------------+-----+
```

At the conclusion of the program, the model will be trained and the pitch type will be predicted based on the user-inputted data. The following screen shot shows the predicted pitch type based on the user inputted features. Of note, the '100.0' label is a dummy label, as these unique features inputted by the user don't correspond to an actual pitch type that has occurred in the past, but instead a new future scenario. As seen in the following screen shot, the pitch type predicted based on this set of feature data is a fastball, which corresponds to the '0' label.

```
17/12/19 18:58:57 INFO DAGScheduler: Job 86 finished: showString at NativeMethodAccessorImpl.java:0, took 0.033759 s
17/12/19 18:58:57 INFO CodeGenerator: Code generated in 20.007253 ms
+----------+-----+--------------------+
|prediction|label|            features|
+----------+-----+--------------------+
|       0.0|100.0|(20,[1,2,3,4,5,6,...|
+----------+-----+--------------------+

17/12/19 18:58:57 INFO FileSourceStrategy: Pruning directories with:
17/12/19 18:58:57 INFO FileSourceStrategy: Post-Scan Filters:
17/12/19 18:58:57 INFO FileSourceStrategy: Output Data Schema: struct<label: double, features: vector>
17/12/19 18:58:57 INFO FileSourceScanExec: Pushed Filters:
17/12/19 18:58:58 INFO CodeGenerator: Code generated in 59.078937 ms
17/12/19 18:58:58 INFO MemoryStore: Block broadcast_202 stored as values in memory (estimated size 277.5 KB, free 316.2 MB)
17/12/19 18:58:58 INFO MemoryStore: Block broadcast_202_piece0 stored as bytes in memory (estimated size 23.4 KB, free 316.2 MB)
```

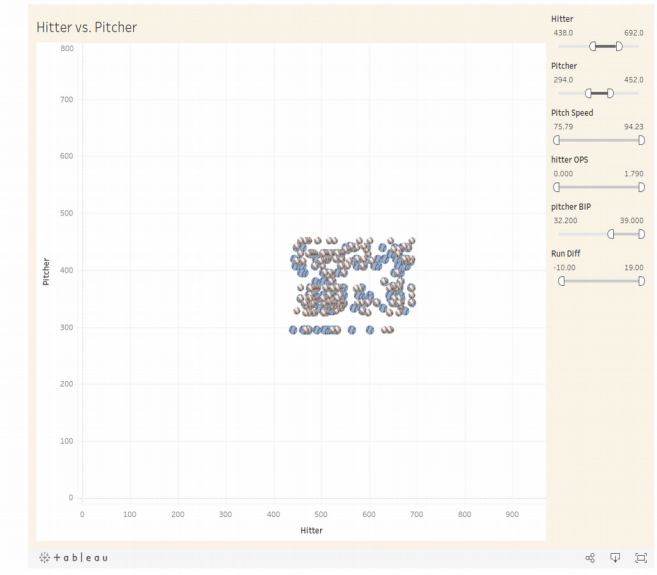
Currently, all of these steps are done in terminal. In the future, a front-end can be built out to make the program more user-friendly and more applicable for implementation by Major League Baseball teams.

VI. Experiment Results

After training the model on 80% of the data and using the remaining 20% of the data for testing, the overall accuracy recorded on predicting pitches is approximately 62%. After adding the top 3 features to the model, a 5% improvement was recorded from a previous 57% accuracy rate. Given the statistics available in the datasets used in the study, these results are fairly comparable to previous works in this field. However, there is plenty of room for improvement by utilizing more historical data to train the model against as well as acquire more advanced and specific metrics to add as features.

Performance of the algorithm was measured against previous studies conducted. In the future, the research could extend to create a separate model for each pitcher and in this way predicted results can be compared to a predictor that simply uses a uniformly independent distribution of a pitcher's historical tendency to throw a given pitch type regardless of any other factors to predict pitches. This was done effectively in related works described earlier in this research paper.

After computing all the predicted pitch types of the testing data, the resulting prediction vectors were cleaned and fed into a Tableau dashboard, a screen shot of the dashboard is below. This allows a user to manipulate a subset of the features and see how the model's predictions change based on different scenarios. In this dashboard, the white baseballs represent predicted fastballs whereas the blue baseballs represent predicted breaking balls.



VII. Conclusion

In conclusion, this study provides a solid foundation for developing a pitch prediction product. Given the datasets and features available, the accuracy results are competitive but there is significant room for improvement before the model can be brought to market. For example, when performing the feature analysis, the data on batters was proven to be unimportant in predicting pitches. However, this could change with more detailed metrics gathered based on historical data between specific batter-pitcher relationships. Furthermore, the feature set can be improved by utilizing average based statistics that capture the performance of the batter or pitcher more accurately when the event occurs, instead of simply using the entire season average statistics of players. More advanced data on pitches such as initial velocity and spin rates of previous pitches will also likely improve the model, as seen in the feature importance analysis that heavily weighted the importance of the pitching data.

In the future, the research would like to make these feature improvements to improve the overall accuracy of the model. Additionally, building out a front-end app that can be used in real-time for teams to make decisions would be the next step once the model's error rate is tolerable to Major League Baseball teams. Latency issues would also have to be solved to perform real-time analysis, especially if this involved re-training the model to make predictions during a game. However, training the model before each game could solve a bulk of these potential problems.

Acknowledgment

The author would like to thank Professor Dr. Ching-Yung Lin as well as the teaching assistants of Big Data Analytics (EECS E6893) for a great semester and providing the education that serves as the basis of this study. Furthermore, the author would like to thank Google, Sportradar, and Baseball Reference for making the data used in the research publicly available. Additionally, thank you to the open-source community of Python and Apache Spark for making the tools accessible to perform the research. Finally, thank you to Tableau for providing the software to create the visualizations seen in this report.

Appendix

Individual Contribution:
Matthew Spitz conducted 100% of this research study.

REFERENCES

[1] Lauren Sommer, “How Can Anyone Hit a 90 mph Fastball? Science Explains!” <https://ww2.kqed.org/news/2013/05/08/how-can-anyone-hit-a-90-mph-fastball-science-explains/>, KQED Science, May 8, 2013.

[2] Dan Gartland, “This Is How Yu Darvish Was Tipping His Pitches in the World Series” <https://www.si.com/mlb/2017/12/11/dodgers-yu-darvish-tipped-pitches-world-series-astros>, Sports Illustrated, December 11, 2017.

[3] “MLB Player Pitching Stats – 2017”

<http://www.espn.com/mlb/stats/pitching/_/sort/opponentAvg/type/opponent-batting/order/false>, ESPN .

[4] “Win Probability Chart”

<https://www.baseball-reference.com/boxes/LAN/LAN201711010.shtml>,

Baseball Reference.

[5] Gartheeban Ganeshapillai and John Guttag, “Predicting the Next Pitch”

<http://www.sloansportsconference.com/wp-content/uploads/2012/02/98-Predicting-the-Next-Pitch_updated.pdf>, Massachusetts Institute of Technology Sloane Sports Analytics Conference, March 2-3, 2012.

[6] Noah Woodward, “A Decision Tree Approach to Pitch Prediction” <https://www.fangraphs.com/tht/a-decision-tree-approach-to-pitch-prediction/>, FanGraphs, March 17, 2014.

[7] “Major League Baseball Data”

<https://cloud.google.com/bigquery/public-data/baseball>,

Google and Sportradar.

[8] “2016 MLB Standard Batting”

<https://www.baseball-reference.com/leagues/MLB/2016-standard-batting.shtml>, Baseball Reference.

[9] “2016 MLB Pitching Pitches”

<https://www.baseball-reference.com/leagues/MLB/2016-pitches-pitching.shtml>, Baseball Reference.

[10] “On-base Plus Slugging (OPS)”

<http://m.mlb.com/glossary/standard-stats/on-base-plus-slugging>, MLB.

[11] "Gradient-boosted tree classifier”

<https://spark.apache.org/docs/2.2.0/ml-classification-regression.html#gradient-boosted-trees-gbts>, Apache Spark.

[12] "Gradient Boosting”

<https://en.wikipedia.org/wiki/Gradient_boosting>, Wikipedia.

[13] "Loss functions for classification”

<https://en.wikipedia.org/wiki/Loss_functions_for_classification>, Wikipedia.

[14] "PySpark master documentation”

<https://spark.apache.org/docs/preview/api/python/pyspark.ml.html?highlight=gbt#pyspark.ml.classification.GBTClassifier>,

Apache Spark.