

# **Baseball Pitch Result Prediction**

**Ugur Oguz**

2982620

Submitted in partial fulfillment for the degree of

Master of Science in Big Data Management & Analytics

Griffith College Dublin

Month, 2019

Under the supervision of Osama Abushama

**Disclaimer**

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the Degree of Master of Science in Big Data Management & Analytics at Griffith College Dublin, is entirely my own work and has not been submitted for assessment for an academic purpose at this or any other academic institution other than in partial fulfillment of the requirements of that stated above.

**Signed:** \_\_\_\_\_**Date:** \_\_\_\_\_

## **Acknowledgements**

I would like to thank my thesis advisor Osama Abushama for the continuous support of my thesis study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my thesis.

I must express my profound gratitude to my parents for providing me unfailing support and continuous encouragement throughout my years of study and the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.



# Table of Contents

|  |    |
|--|----|
| List of Equations .....                          | 6  |
| List of Figures .....                            | 6  |
| Abstract .....                                   | 7  |
| Chapter 1. Introduction .....                    | 8  |
| 1.1    Baseball Pitch Result Prediction .....    | 8  |
| 1.2    Goals.....                                | 9  |
| 1.3    Overview of Approach .....                | 9  |
| 1.4    Document Structure.....                   | 9  |
| Chapter 2. Background.....                       | 10 |
| 2.1    Literature Review .....                   | 10 |
| 2.2    Related Work.....                         | 11 |
| Chapter 3. Methodology .....                     | 13 |
| Chapter 4. System Design and Specifications..... | 15 |
| 4.1    Technologies & Techniques Used .....      | 15 |
| 4.1.1  Programming Language .....                | 15 |
| 4.1.2  Classification Algorithms.....            | 15 |
| 4.2    The Model Structure .....                 | 18 |
| Chapter 5. Implementation .....                  | 23 |
| Chapter 6. Testing And Evaluation .....          | 26 |
| 6.1    The Data .....                            | 26 |
| 6.2    Data Filtering.....                       | 26 |
| 6.3    Data Preprocessing .....                  | 27 |
| 6.4    Result For The Model .....                | 29 |
| Chapter 7. Conclusions and Future Work .....     | 32 |
| References .....                                 | 33 |
| Appendix I.....                                  | 36 |

## List of Equations

|   |    |
|---|----|
| Equation 4.1 Decision Tree Information Gain .....                             | 16 |
| Equation 4.2 Huang's Equation To Find Minimal Node Number.....                | 18 |
| Equation 4.3.1 Huang's 2 Layer Equation To Find First Layer Node Number.....  | 19 |
| Equation 4.3.2 Huang's 2 Layer Equation To Find Second Layer Node Number..... | 19 |
| Equation 4.4 Sigmoid Activation Function.....                                 | 21 |
| Equation 4.5 Softmax Activation Function.....                                 | 21 |
| Equation 4.6 Cross-Entropy Loss.....  | 22 |

## List of Figures

|  |    |
|--|----|
| Figure 3.1 WorkFlow .....                      | 13 |
| Figure 4.1 Decision Tree .....                 | 16 |
| Figure 4.2 MLP Model With 2 Hidden Layer.....  | 18 |
| Figure 4.3 Activation Function On Neuron ..... | 20 |
| Figure 5.1 The Index Page. ....                | 23 |
| Figure 5.2 The Rule Page .....                 | 24 |
| Figure 5.3.1 Predict Page Forms .....          | 24 |
| Figure 5.3.2 Predict Page Test-Table .....     | 25 |
| Figure 5.4 The Result Page .....               | 25 |
| Figure 6.1 Example Of Cleaning Pitch Data..... | 27 |
| Figure 6.2 Example Of Merging Datasets.....    | 28 |
| Figure 6.3 Random Forest.....                  | 29 |
| Figure 6.4 Decision Tree Classifier.....       | 29 |
| Figure 6.5 kNN Classifier .....                | 30 |
| Figure 6.6 OneVsRest .....                     | 30 |
| Figure 6.7 MLP model .....                     | 31 |

# Abstract

---

People interests in sports made athletes take their work more serious and increased technologies in the sports field. Baseball is one of the sports that holds a vast amount of data. These data's are used in improving athletes and betting companies, but baseball is a game that holds too many noises that causes a less accurate result. Similar projects made in this field and most of them are using binary classifications. Also, due to noises in baseball data, they exclude anomalies that occur and uses a limited amount of data.

In this project, our aim was to predict baseball pitch results without excluding anomalies. To achieve this goal we have used R and Python programming languages. Various algorithms and features tested, based on these evaluations, we have created MLP model that achieves 46% accuracy.

# Chapter 1. Introduction

---

## 1.1 Baseball Pitch Result Prediction

Most of the people have an interest in sports at some stage in their lives. Among these sports in the USA, baseball is one of the most popular sport. In baseball, coaches and managers based their decisions in-games from their experience and instincts.

All events in the game were measured, and players got evaluated based on certain statistics such as the number of home runs, earned run average (ERA), how many runs they allowed (R), innings pitched (IP), etc. This view of the game changed after the approach presented by the general manager of Oakland Oaks in 2002, Bill Bean. He did not care about the number of home runs, but he cared about the contribution of total runs and how many runs he could save the team. This approach used big data and got famous when it was written down in the book Moneyball [1]. In 2006, MLB installed PITCHf/x system to all stadiums that measures ball trajectory with details of breaking angle, starting and ending speed, spin rate, etc. The system drastically increased the data amount in games allowed to improve umpires evaluation and judgment. Major League Baseball (MLB) games lend themselves well to machine learning due to the vast amount of data that is recorded for each game. The downside is that baseball itself includes various variables and thus, difficult to predict with high accuracy. A player that is having a good day can hit a home run, and this action alone can increase morale and momentum of the weak team to beat stronger teams. The models are important to improve team performance and selecting players to win the games. By winning games, teams can attract fans and most importantly increase their revenue.

Major League Baseball alone is a multi-billionaire industry. In 2015, the four largest professional sports leagues generated almost \$28 billion in revenues with \$7.9 billion. MLB is the second-largest sports league [2]. Also, MLB offers more bet opportunities than other sports, regular season alone offers bettors 2430 games to wager still it is under bet compared to other sports which makes Baseball prediction models highly valuable.



## **1.2 Goals**

In this project, our aim is to predict what pitches will yield with the highest accuracy. This project mostly benefits for teams. It allows coaches and battery (pitcher and catcher) to simulate in-game situation which pitch will give the most satisfying result. On the other hand, it is beneficial for betting companies as well. While baseball has very active sessions, the betting amount is lower than other sports. This project can help betting companies to increase their in-game betting options that results with increasing bettor's interest in-game.

## **1.3 Overview of Approach**

This paper offers different view of predicting baseball in-game situation. In our project, we have used multi-class classification to predict pitch results such as strike, ball, in-game (when a batter hits the ball). While most of the papers are using a dataset from MLB, there is not any published paper that predicts pitch results. The most similar project predicts the first pitch type in inning being a fastball or not. The other one predicts pitch type of the next inning, which they use small and specific data based on the pitch amount of pitchers and certain pitchers to reduce the noises of the dataset. To preserve the realisticness of the games and players, we have not inserted conditions for prediction in this paper.

## **1.4 Document Structure**

The rest of this document is as follows. Chapter Two provides a literature review of Baseball Pitch Result Prediction, and the sources consulted in accomplishing the project, in addition to related work. Chapter Three describes the methodology and high-level design of the project structure. In Chapter Four, discusses the technologies, techniques, approaches that are used and explained in detail. It also includes model structure. Chapter Five provides the implementation details of the project and details are shared about demonstration part of the project. In Chapter Six, details of the data, data filtering, preprocessing of the project, including evaluation of classifiers interpreted in detail and decisions made are explained. Finally, Chapter Seven presents conclusions and future work.

## Chapter 2. Background

### 2.1 Literature Review

As interest in the sports field increased, people desire to participate increased as supporting their team or betting against other teams. With the increased data of the games, it became easier to predict the result of the games and the weaknesses of the players that can be improved.

As the field of machine learning (ML) and its application improves, it became possible to classify and predict beyond handling binary problems. While binary predictions are still valuable, such as yes, no or win, lose answers. In the world of increasing complexity, machine learning methods have high demand in multi-class problems.

Major League Baseball provides precise and detailed data set with PITCHf/x system that tracks pitches trajectory in every game and stadium. Pitch prediction has been a topic of previous researches. They have mostly focused on binary classifications between first ball in inning, fastball and other pitches [11].

In [6], Gutttag and Ganeshapillai's, binary classification model was able to correctly predict next pitch 70% of the time compared to naïve classifier accuracy of 59%. They have achieved an 18% improvement at predicting for pitchers who overwhelmingly utilized one pitch, such as Mariano Rivera. Rather, their SVM approach holds more promise when a pitcher has a more diverse pitch repertoire.

In [10], Hoang also studied the problem of classifying pitches into fastball and non-fastball categories. They have used four different binary classification algorithms included k-nearest neighbours (kNN), Support Vector Machine with linear kernel (SVM-L), Support Vector Machine with Gaussian kernel (SVM-G) and Linear Discriminant Analysis (LDA). While SVM approach resulted 75% with the worst results, LDA achieved 78% accuracy as the best. Another project focused their work on improving feature selection and achieved a modest improvement in accuracy by using dominant features only.

In [8], Bock has developed statistical models of pitcher behaviour using pitch sequences thrown during three MLB seasons (2011-2013). The purpose of these models was to predict the next pitch type, for each pitcher, based on data available at the immediate moment, in each at-bat. Independent models were developed for each player's most frequent four pitches. The results were obtained using non-linear kernel support vector machine (SVM) s versus multinomial logistic regression. Overall predictability of next pitch type is 74.5% while they examined the ERA and FIP of each pitcher, eventually determined there was “no significant relationship between predictability and performance”. The true accuracy for blind prediction was listed as less than 61%.

## **2.2 Related Work**

The baseball predictions can be made by using binary classifications but using multi-class classification we can achieve more complex result rather than win-lose rate. While there is not any previous project made on our topic, there are similar projects what we wanted to achieve. In [15] project analysis the pitch data with decision tree models, random forest models and logistic regression to study what factors would affect the umpire's decision. The training models achieved 88%-91% accuracies among these models Random forest holds the highest accuracy with tree model second place and logistic regression in the last place.

In [12] Sidle's uses multi-class classification, classifying non-fastball pitches as curveballs, changeups or sliders. They applied three different methods to classify pitches to seven different types, for models they have used Latent Dirichlet Allocation (LDA), Support Vector Machine (SVM) and bagged random forest of Classification Trees to classify. The result showed that forest of Classification Trees is superior with LDA second and SVM last. On the other hand, LDA is both more efficient and more consistent than the others.

In [13], Attarian used to data from PITCHf/x system to classify pitches thrown by certain pitchers into different pitch types. The types were based on trajectory measures like spin rate and velocity of the pitch. In this paper kNN algorithm and naïve Bayes

classifier compared as a result kNN algorithm performed 4% higher than naïve Bayes classifier.

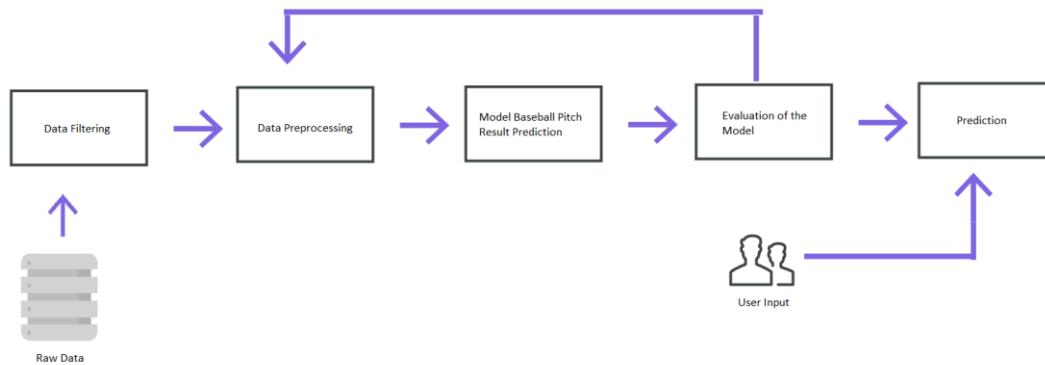
In MLB's official website<sup>14</sup>, they used "hang-time" that referred as the time that a batted ball spends in the air, and ball travel distance to categorize if the ball is catchable in-game. The classification is based on the probability of a field player catching the ball. For example, 23%-50% catch probability class is low, due to short hang time or short travel distance. These two features determine the categorization of the ball.

The background search shows that different data filtering and preprocessing decisions, the model structure and selection of the classification algorithm affects the performance of the model. In most of the similar projects have a limited amount of data while having high success rates than shows us machine learning (ML) methods can be used to improve in-game decisions despite high noise values. Therefore, this project proposes using classifications with minimum amount of limitation, thus being more realistic and accurate evaluation to predict the outcome of the pitch result in-game. The side effect of lowering these limitations causes lower accuracy in the model. The details of this project are given in the following sections.

## Chapter 3. Methodology

---

As mentioned before, this project mainly includes data mining and classification tasks. Therefore, to reach the project goal, the typical data mining approach was fitted to the project requirements. You can see the steps followed in Figure 3.1.



**Figure 3.1** workflow

To reach the goal, the steps shown in the diagram were followed. When the data was acquired, firstly it was necessary to filter the data to get rid of unrelated features. Plus it was essential to select proper features to achieve best results. In data preprocessing step, we have got rid of corrupted data and added new features. After executing the data on the model, based on evaluations we have added-removed features on the dataset. When the data became ready, it was time to build a model. As you can see above, we have configured and evaluated our models, feature selection multiple times to find the best model with best-fitted classification algorithm. After evaluation of each classifier, the most efficient model was picked. Finally, some tests were run on the final baseball pitch prediction model and performance was evaluated.

In this project, we have selected R to perform our data filtering and preprocessing tasks. The reason is that R is more powerful and efficient to read and process data than other tools and do not require to use extra libraries to preprocess our data. In this project we also used python 3.6 for modelling and application part. The reason is that python offers powerful libraries for classifying model tasks. For instance, it provides Keras, Tensorflow, Sci-kit Learn for building efficient models, etc. For application we have used Flask that provides developers tools and libraries to build scalable web applications easily.

During the selection of the best model for each classifier, it was aimed to find out the best algorithm that will return the highest accuracy. Therefore, we have used Random Forest, Decision Tree, kNN, OnevsRest and MLP models. Based on previous papers and these algorithms are simple yet efficient to achieve our goal. The performance of Multilayer perceptron model was evaluated because neural networks are good at identifying complex relationships between the data points.

In the implementation part, a simple web application created that shows how the model works and allows user to make predictions by their inputs and based on the model. To develop such a web application, we have used a framework called Flask. Flask provides libraries, tools and technologies to develop simple web applications, and it is compatible with our python version.

## Chapter 4. System Design and Specifications

---

In chapter 3. We had mentioned about different technologies and approaches that we used during this project. In this chapter, we explained these technologies and model structures more detailed.

### 4.1 Technologies & Techniques Used

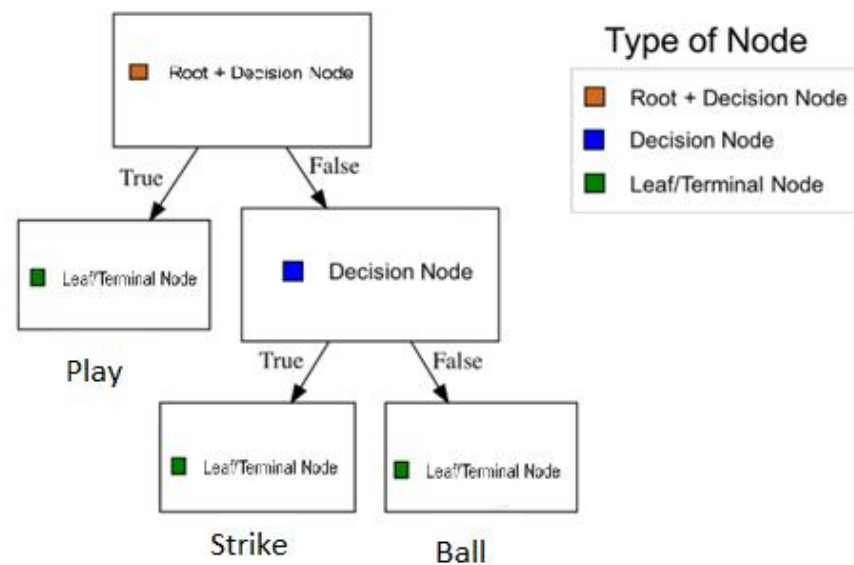
#### 4.1.1 Programming Language

In order to handle the tasks effectively, we have split project into parts as data filtering, preprocessing, model creation and application. R and python programming languages are used. Both of these languages excel in data mining. We have handled the preprocessing stage in R that allowed visibility of the chosen variables and tracking data defiance more clear. We have used python for the rest of our project. The packages and libraries in python allow developing project easily, especially for data analytics such as pandas is BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools<sup>16</sup>. Scikit-learn provides machine learning algorithms such as k-nearest neighbors (kNN), Decision Tree, SVM, etc. Numpy is the fundamental package for scientific computing. It contains powerful N-dimensional array object and efficient multi-dimensional container of generic data also arbitrary data-types can be defined<sup>17</sup>. Keras is an open-source python library that allows developer to build neural network models. These were the packages that have been used during this project

#### 4.1.2 Classification Algorithms

Decision Tree Classifier, Random Forest, KNN, OneVsRestClassifier and MLP algorithms were compared in order to find the best-suited algorithm to our project. These algorithms have proven themselves in similar project and they are widely used.

Decision Trees are popular supervised learning method. It can be used in regression and classification problems. Decision Tree Classifier is a class capable of performing multi-class classification on a dataset and supports multi-output strategies. The root node is just the topmost decision node. In other words, it is the starting point to Travers classification tree. The leaf node, also called terminal node, is nodes that do not split into more nodes. Leaf nodes are here classes are assigned by majority vote.



**Figure 4.1** Decision Tree

Below you can see the equation of the Decision Trees information gain:

$$IG(D_p, f) = I(D_p) - \sum_{j=1}^m \frac{N_j}{N} I(D_j)$$

f: feature split on  
 $D_p$ : dataset of the parent node  
 $D_j$ : dataset of the jth child node  
 $I$ : impurity criterion  
 $N$ : total number of samples  
 $N_j$ : number of samples at jth child node

**Equation 4.1** Decision Tree Information gain

Random Forests are combination of tree predictors such that each tree depends on the values of random vector sampled independently and with the same distribution for all



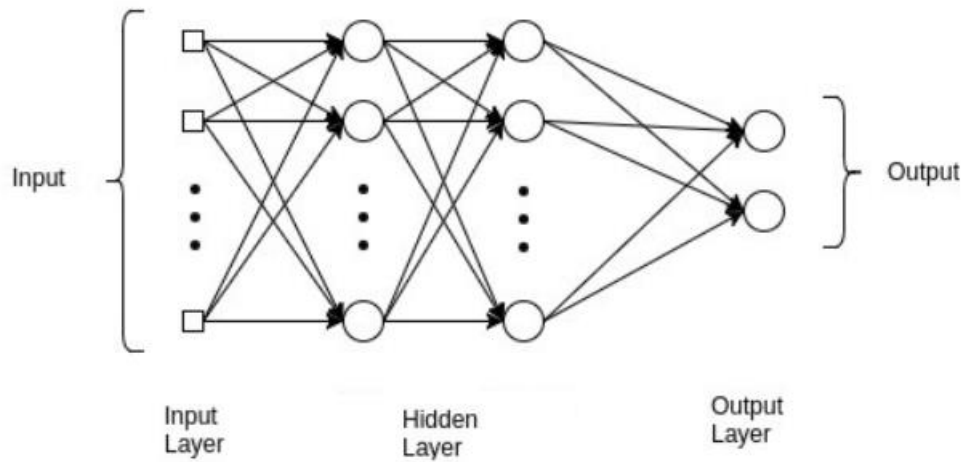
trees in the forest. The generalization error of a forest of tree classifiers depends on the strength of the individual trees in the forest and the correlation between them, but they are robust with respect to noise [20].

Support Vector Machine (SVM)'s are set of related supervised learning methods used for classification and regression. SVM map input vector to higher dimensional space where maximal separating hyperplane (it is a line in 2D space) is constructed and uses parallel hyperplanes to define both linear and non-linear boundaries that give good results in complex problems [21].

k-nearest neighbours (kNN) is the simplest classification algorithm. The classification algorithm doesn't depend on the structure of the data. When new example is encountered, its kNN from the training data are examined. Distance between two examples can be Euclidean distance between their feature vectors. The majority class among the kNN is taken to be the class for encountered example.

OneVsRestClassifier also known as one-vs-all, this strategy consists in fitting one binary classifier per class. We associate a set of positive examples for a given class and a set of negative examples which represents all other classes for each classifier. In addition to its computational efficiency, one advantage of this approach is its interpretability.

Multilayer Perceptron (MLP) is a perceptron that teams up with additional perceptron's, stacked in several layers to solve complex problems. The combined form of perceptron is a form of artificial neural network. A neural network can theoretically answer any question if it's given enough training data and computing power. MLP contains at least three layers as input layer, hidden layers and output layer. It is essential for MLP to have backpropagation, hyperparameters and Advanced structures to become deep neural network.



**Figure 4.2** MLP model with 2 Hidden Layer

## 4.2 The Model Structure

The structure of the model for baseball pitch result prediction is the key part of this project. The similar projects and their success rates were the inspiration sources to build the model. Also we have tested various approaches till finalizing model to use MLP algorithm.

The internal parameters of the model play an important role in efficiently and effectively training the model and produce accurate results. [22] Huang proved that in the two-hidden-layer case, with  $m$  output neurons, the number of hidden nodes that are enough to learn  $N$  samples with negligibly small error is given by

$$2\sqrt{(m+2)N}.$$

**Equation 4.2** Huang's equation to find  
Required minimal node number

Specifically, he suggests that the sufficient number of hidden nodes in the first layer is

$$\sqrt{(m+2)N} + 2\sqrt{N/(m+2)},$$

**Equation 4.3.1** Huang's 2 layer equation to find

Number of first layer node number

And the second layer is

$$m\sqrt{N/(m+2)}.$$

**Equation 4.3.2** Huang's 2 layer equation to find

Number of second layer node number

As the optimal topology is judged by the capacity to generalize on unseen data, the most accurate structure will have fewer nodes than that suggested by equations.

Based on Huang topology, we have calculated and tested best-suited node amount.

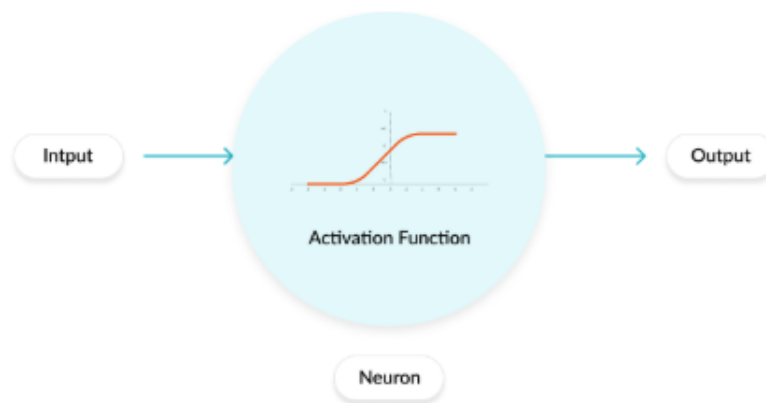
Optimizer algorithms help us to minimize an Error function which simply a mathematical function dependent on the Model's internal learnable parameters which are used in computing the target value (Y) from set of predictors (X) used in the model.

adaDelta is an extension of AdaGrad which tends to remove the decaying learning Rate (model stops learning entirely because learning rate gets smaller decreases model to learn fast which gives very slow convergence and takes long time to train model) problem of it [23].

Adam stands for Adaptive Moment Estimation is another method that computes adaptive learning rates for each parameter. Additionally storing an exponentially decaying average of past squared gradients like AdaDelta, Adam also keeps an exponentially decaying average of past gradients [24].

Root Mean Square Propagation (RMSProp) maintains a per-parameter learning rate that is adapted based on the average of recent magnitudes of the gradient for the weight. This means the algorithm does well on online and non-stationary problem (like noise) [25].

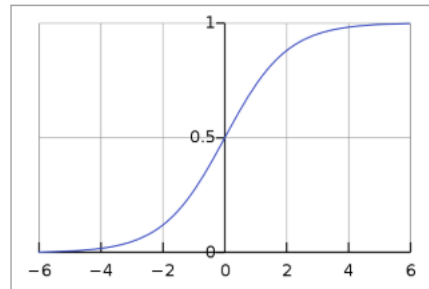
Activation functions are mathematical equations determine the output of a neural network. The function is attached to each neuron in the network and determines whether it should be activated based on each neuron's input is relevant for the model's prediction it also helps to normalize the output of each neuron to a range between 1 and 0 or -1 to 1. Backpropagation techniques to train models increase computational strain on activation functions. Because of the speed, non-linear functions emerged such as ReLU, Sigmoid etc.



**Figure 4.3** Activation function interaction with neuron

There are three types of activation functions. Binary step function, if input value is above a certain threshold, it sends exactly the same signal to the next layer. While it does not allows multi-value outputs. Secondly Linear Activation Function, the given input multiplied by the weights for each neuron and it also allow multiple outputs. However it is not possible to use backpropagation and all layer of the neural network collapse into one. Finally, Non-Linear Activation Function allows the model to create complex mapping between the network and it is essential for learning high dimensional or non-linear datasets<sup>26</sup>.

There are seven common Non-linear Activation Functions. The advantages of the sigmoid are it has smooth gradient, output values bound and clear predictions while it has the vanishing gradient problem.



$$f(s_i) = \frac{1}{1 + e^{-s_i}}$$

**Equation 4.4** Sigmoid activation function

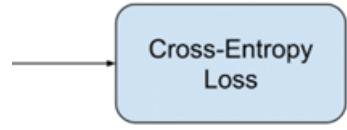
ReLU (Rectified Linear Unit)'s is computationally efficient and non-linear although it has the Dying ReLU problem that is when input approaches to zero or input is negative, the gradient function becomes zero and network cannot learn.

Softmax is typically used only for the output layer. It can handle multiple classes, normalizes outputs for each class and divides by their sum, giving the probability of the input value being in specific class. Softmax can be computed as

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}}$$

**Equation 4.5** Softmax activation function

Loss functions are one of the two parameters required to compile a model. In our model, we have used Categorical Cross-Entropy loss that supports label classes.



A diagram showing a light blue rounded rectangle labeled "Cross-Entropy Loss". An arrow points from the left into the rectangle.

$$CE = - \sum_i^C t_i \log(f(s)_i)$$

**Equation 4.6** Cross Entropy Loss

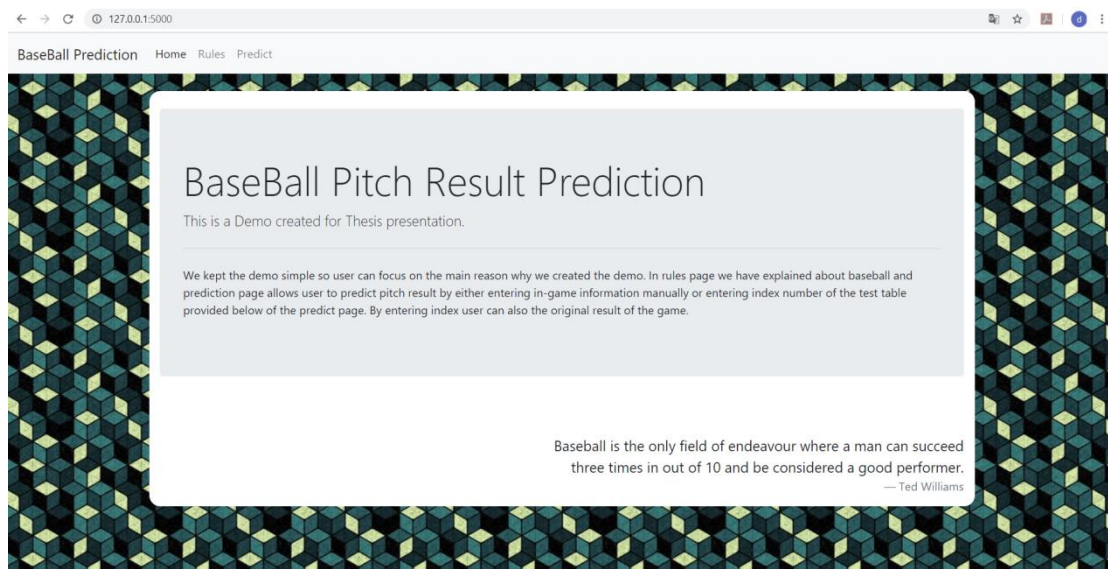
In this project after model training finishes, the trained model and its weights are saved to be called in application. Thus it would not require learning the model every time in the application. Also separating these processes helps to make changes much easier.

## Chapter 5. Implementation

---

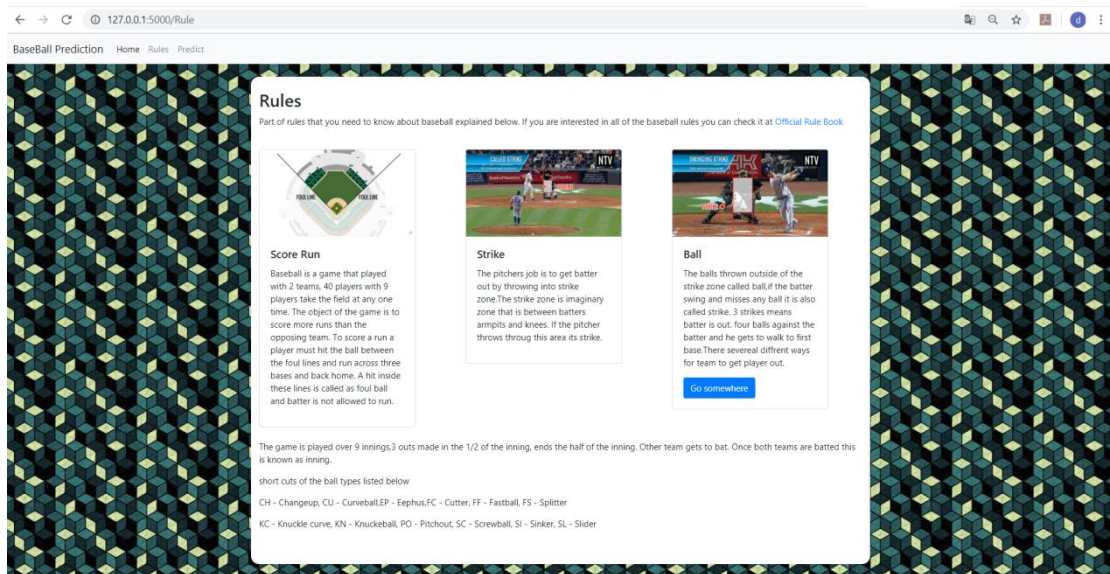
In order to demonstrate how the model works, a simple web application was designed using flask framework. Flask is a python web application that simply works as a wrapper around Werkzeug and Jinja2 and provides the developer's tools and libraries to build scalable web applications easily.

Our applications start with loading pre-trained model with weights that help to lower the server wake up time. The web application consists of 4 web pages such as the "Index", "rule", "prediction" and "result" pages.



**Figure 5.1** The Index Page

The index page is the first page that user interacts, it is static page contains explanation about application and quota.



**Figure 5.2** The Rule Page

The Rules page shows small information about baseball basics of how it is played, calls are made and have a link to the official baseball rule book.

**Predict what will happen in given situation**

Table contains 10 test set in form below you can enter either index number, or enter random values to make prediction

Enter Index number from sample test-set to see the prediction result

Index Number  [Submit](#)

**Predict game situation**

Inning:

Pitcher Team Score:

Batter Team Score:

Batter Outs:

Pitch number against the batter (current inning):

Loaded Bases:  
☐ Base 1  
☐ Base 2  
☐ Base 3

spin rate:

spin direction:

start speed:

end speed:

Pitch Count in Game:

Strike Count in Game:

Pitch Type:

total CH:

total CU:

total EP:

total FC:

total FF:

total FS:

total KC:

total KN:

total PO:

total SC:

total SI:

total SL:

[Predict](#)

**Figure 5.3.1** Predict Page Forms



|   | inning | p_score | b_score | outs | pitch_num | on_1b | on_2b | on_3b | spin_rate | spin_dir | start_speed | end_speed | total_C |
|---|--------|---------|---------|------|-----------|-------|-------|-------|-----------|----------|-------------|-----------|---------|
| 0 | 7.0    | 5.0     | 3.0     | 0.0  | 8.0       | 1.0   | 0.0   | 0.0   | 1438.340  | 232.320  | 84.9        | 78.9      | 4.0     |
| 1 | 6.0    | 3.0     | 5.0     | 2.0  | 2.0       | 0.0   | 0.0   | 0.0   | 1479.409  | 42.785   | 79.1        | 72.9      | 11.0    |
| 2 | 6.0    | 1.0     | 8.0     | 0.0  | 1.0       | 0.0   | 0.0   | 0.0   | 549.512   | 172.351  | 89.2        | 82.3      | 4.0     |
| 3 | 2.0    | 0.0     | 5.0     | 0.0  | 5.0       | 0.0   | 1.0   | 0.0   | 1138.520  | 83.693   | 83.9        | 77.8      | 0.0     |
| 4 | 6.0    | 1.0     | 0.0     | 0.0  | 1.0       | 0.0   | 0.0   | 0.0   | 1387.353  | 161.831  | 85.7        | 80.3      | 5.0     |
| 5 | 7.0    | 1.0     | 1.0     | 0.0  | 1.0       | 0.0   | 0.0   | 0.0   | 1032.217  | 164.300  | 81.4        | 75.4      | 6.0     |
| 6 | 6.0    | 8.0     | 4.0     | 1.0  | 1.0       | 0.0   | 0.0   | 0.0   | 1591.873  | 219.537  | 87.7        | 80.4      | 8.0     |
| 7 | 4.0    | 3.0     | 0.0     | 1.0  | 2.0       | 0.0   | 0.0   | 0.0   | 1934.080  | 239.997  | 87.0        | 80.4      | 15.0    |
| 8 | 7.0    | 9.0     | 3.0     | 1.0  | 2.0       | 0.0   | 1.0   | 0.0   | 2340.430  | 241.827  | 90.7        | 83.5      | 5.0     |
| 9 | 1.0    | 0.0     | 0.0     | 0.0  | 2.0       | 0.0   | 0.0   | 0.0   | 1540.562  | 243.069  | 82.7        | 76.4      | 23.0    |

**Figure 5.3.2** Predict Page Test-Table

Predict page contains 2 forms and 1 table. The table shows a small portion of the test set, and one of the forms allows user to enter the index number of the test-set and view predicted result with original result of the chosen test. Final form predicts depending on user input.

|                  | Ball    | Strike  | Play    |
|------------------|---------|---------|---------|
| Predicted result | 0.28418 | 0.39249 | 0.32333 |
| original result  | 1       | 0       | 0       |

**Figure 5.4** The Result Page

Result page displays the result of requested prediction.

## Chapter 6. Testing and Evaluation

---

### 6.1 Data

Major League Baseball (MLB) is a professional baseball organization and the oldest of the major professional sports league in the USA and Canada. After 2006 PITCHf/x system installed in every MLB stadium [1], with this system 23 attributes recorded in pitch element, the data is used in broadcasting strike-zone, determining the type of pitch, grade and provide feedback to umpires. The MLB has data from 1900 till present, allowing users to download and scrapped their data. “Pitch-level data for every pitch thrown during the 2015-2018 MLB regular seasons” data that we use in this project scrapped from MLB’s official site by Paul Schale for research purpose.

The scrapped data was divided into five CSV files:

1. At\_bat: information about the batter and contains game information that is related to the batter like inning number, number of outs etc.
2. Ejections: actions taken by umpires. Which removes a player, coach or manager from the game.
3. Games: general information about game and umpires name with wich base they cover.
4. Pitches: contains PITCHf/x system measurements about the trajectory of the pitch and matches up with at\_bat and player\_names.
5. player\_names: matches player names with player Ids.

### 6.2 Data Filtering

The baseball dataset is a very large dataset that includes more than 2867150 pitch event and 331 unique pitchers with total 9711 games. However, not all baseball information is necessary for this project purpose. Therefore, we have removed unnecessary features in filtering step. Below there is detailed information about features.

We didn't use ejection.csv file because it contains detailed information about ejections which do not affect our pitch prediction.

ab\_id: Foreign key for atbats.csv, ejection happened before, after during atbat

des: description, natural language

event\_num event number for ejection

g\_id: foreign key for game.csv

player\_id: foreign key for players.

date: Directly from game.csv

BS: 'Y' if ejection was for arguing balls and strikes, empty otherwise

CORRECT: Whether the ejection was correct.

team: Team for player ejected

Is\_home\_team: whether that team is the home team

As baseball in-game pitch prediction it requires in-game information like number of outs, pitch count, bases, ball information and team score etc. The rest of the data will only cause noise while we would not include player names due to uniqueness but to find out the potential amount of the pitch type and pitch of the pitcher. It has been excluded later on data preprocessing.

### 6.3 Data Preprocessing

Before doing the model test, we applied data preprocessing to acquire better performance from the test. In this step we have removed missing values and pitch\_types that do not exist in PITCHf/x system definition. Plus we have corrected pitch\_types that are categories wrongly and excluded pitch\_types that have only one possible result like Intentional ball which ball is thrown intended to walk a batter as the name suggest, the result of this pitch is always ball.

```
#removing unknown data/wrong type pitch // empty rows
clean.dataset <- (na.omit(pitches.dataset))
clean.dataset <- clean.dataset[clean.dataset$pitch_type != "UN" ,]
clean.dataset <- clean.dataset[clean.dataset$pitch_type != "FA" ,]
clean.dataset <- clean.dataset[clean.dataset$pitch_type != "AB" ,]
clean.dataset <- clean.dataset[clean.dataset$pitch_type != "IN" ,]
#change FO to PO (pitchout)
clean.dataset[clean.dataset$pitch_type == "Fo", "pitch_type"]<-"Po"
#changed FT to SI(sinker)
clean.dataset[clean.dataset$pitch_type == "FT", "pitch_type"]<-"SI"
```

**Figure 6.1** Example Of Cleaning Pitch Data

After preparation of pitch features, we have merged the dataset with other features. Plus we have added some features that display total pitch amount a pitcher can throw depending on its type, total pitch and strikes that pitcher made.

```
total<- merge(clean.dataset, atbat.dataset, by="ab_id")
total<- merge(total,games.dataset, by="g_id")
total<-merge(total , player.dataset, by.x = "batter_id",by.y = "id")
colnames(total)[colnames(total) == "first_name"]<- "batter_Fname"
colnames(total)[colnames(total) == "last_name"]<- "batter_Lname"
total<-merge(total , player.dataset, by.x = "pitcher_id",by.y = "id")
colnames(total)[colnames(total) == "first_name"]<- "pitcher_Fname"
colnames(total)[colnames(total) == "last_name"]<- "pitcher_Lname"

total<-total[,c("g_id", "pitcher_id", "inning", "pitch_type", "p_score", "b_score", "outs", "pitch_num", "on_1b", "on_2b", "on_3b", "spin_rate", "spin_dir", "start_speed", "end_speed", "type")]
total<-cbind(total, dummy(total$pitch_type, sep = "_"))
```

**Figure 6.2** Example Of Merging Dataset's

Before we became certain of our feature selection, we have tried our feature selection and created different variations that might suit better and we evaluated on multiple models and decided to use our current features. In the model, we are using total 41 features. Three of them are the classes that we want to predict (strike, ball, play). Eight of them are current game situation (Inning, pitcher team score, batter team score, outs, current pitch number against batter, on\_1b , on\_2b , on\_3b). As ball information four features (spin\_rate, spin\_dir, start\_speed, end\_speed), plus twelve features that shows which pitch type thrown in other words, we have used onehotencode method to binaries the pitch type. Also twelve features that interpreted as total types of pitches that pitcher can throw. Finally, two features that we created shows potential pitch amount that pitcher can throw and how many of them will result with a strike.

We also wanted to include some other features. We have tried to use pressure factor as a variable but we noticed there is no certain way to evaluate pressure conditions and affects because it changes person to person [28]. In our project we did not exclude the anomalies because the goal of this project is to predict real game situation pitch results, rather than finding pitch results in ideal game flow. Another feature that we wanted to have is a feature that showed current pitch count on the event so we could measure when pitchers pitch starting to deform, but the dataset that we are using does not hold precise information about pitching order.

## 6.4 Results For The Model

This section explains the models that have been tested and accuracy achieved.

In our project, we have tried various models such as Random Forest, Decision Tree Classifier, KNN, OnevsRest with linear SVC and SVC and MLP algorithms with various optimizer. During our experiment process, we have used 50.000 game data.

The Random Forest model yielded 0.23% Accuracy, which is low compared to other tested models.

```
#####RANDOM FOREST
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier
#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Random Forest Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

**Figure 6.3** Random Forest

In Decision Tree Classifier accuracy is slightly better than Random forest, 38% accuracy.

```
##DescisionTreeClassifier
# training a DescisionTreeClassifier
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
dtree_model = DecisionTreeClassifier(max_depth = 500).fit(X_train, y_train)
dtree_predictions = dtree_model.predict(X_test)
print("Descision Tree Classifier Accuracy:",dtree_model.score(X_test, y_test))
tree.plot_tree(dtree_model)
##
```

**Figure 6.4** Decision Tree Classifier

kNN Classifier returned 33% accuracy

```
##KNN classifier
# training a KNN classifier
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3).fit(X_train, y_train)
# accuracy on X_test
accuracy = knn.score(X_test, y_test)
print("kNN Accuracy:",accuracy)
# creating a confusion matrix
knn_predictions = knn.predict(X_test)
##
```

**Figure 6.5 KNN**

The OneVsRest Classifier yielded 44% accuracy by using linear-SVC, which is relatively high compared to other classifiers. While using SVC returned 3% accuracy as the lowest accuracy among the models.

```
##OneVsRest
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import LinearSVC
from sklearn.svm import SVC
clf = OneVsRestClassifier(LinearSVC(random_state=0)).fit(X_train, y_train)
print("OneVsRest LinearSVC Accuracy:",clf.score(X_test,y_test))

clf = OneVsRestClassifier(SVC()).fit(X_train, y_train)
print("OneVsRest SVC Accuracy:",clf.score(X_test,y_test))
##
```

**Figure 6.6 OnevsRest Linear SVC- SVC**

The current model that we are using yielded 46% accuracy. It is slightly better than OneVsRest classifier, which resulted in 44% accuracy. During our MLP model, we have tested various features. Among these features rmsprop optimizer yielded 45% accuracy, and adadelta returned 45% accuracy.

```
#Final Model
model = Sequential()
model.add(Dense(300, activation = 'relu', input_dim=38))#1100
model.add(Dense(150, activation = 'relu' ))#relu
model.add(Dense(3, activation = 'softmax'))

# For a multi-class classification problem
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
print(model.summary())
pk =model.fit(X_train,y_train,batch_size = 100,epochs=450,verbose = 0,validation_split=0.2)
score,acc = model.evaluate(X_test,y_test,batch_size = 100,verbose=2)
print("Final model Accuracy: ",acc)
##

#MLP test optimizer
model.compile(optimizer='rmsprop',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
pk =model.fit(X_train,y_train,batch_size = 100,epochs=450,verbose = 0,validation_split=0.2)
score,acc = model.evaluate(X_test,y_test,batch_size = 100,verbose=2)
print("rmsprop accuracy: ",acc)
#
model.compile(optimizer='adadelta',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
pk =model.fit(X_train,y_train,batch_size = 100,epochs=450,verbose = 0,validation_split=0.2)
score,acc = model.evaluate(X_test,y_test,batch_size = 100,verbose=2)
print("adaDelta accuracy: ",acc)
##END- test
```

Figure 6.7 MLP algorithms And Tested Optimizer

## Chapter 7. Conclusions and Future Work

---

This paper presents a model using scrapped MLB data. In the beginning, we have removed unrelated features from data. After we have optimized candidate models and created sample datasets. Evaluated all datasets on candidate models and used the most successful model.

In general, it was aimed to find the best model to predict baseball pitch result. In order to find the best model configuration, we have used variations of MLP functions, kNN, Random Forest, Decision Tree, OnevsRestClassifier. For our model we have chosen MLP model with ReLu activation function, adam optimizer and categorical-crossentropy for loss function. In our project we have achieved 46% accuracy.

Future work concerns different versions of model and feature based approaches can be made. Two model approaches can be used like one of the models can predict the anomalies and the other one can predict the ideal game situation. Or multiple models can be used on groups of baseball events like pitcher related model (all ball measurements), environment-related model (played field, spectators, weather) and game-related model (team score). The accomplishments of these models do not exclude the anomalies; thus the predictions are more realistic.



## References

---

- [1] Michael Lewis. Moneyball: The Art of Winning an Unfair Game. W. W. Norton Company, 2003.
- [2] Gerald T. Healy III, Jing Ru Tan, Peter Orazem: Measuring Market Power in Professional Baseball, Basketball, Football and Hockey
- [3] Tim Elfrink: Predicting the outcomes of MLB games with a machine learning approach. June 13, 2018
- [4] Nico Cserepy and Robbie Ostrow. Predicting the final score of major league baseball games. 2015.
- [5] Randy Jia, Chris Wong, and David Zeng. Predicting the Major League Baseball Season. , AUTUMN 2013
- [6] Gartheeban Ganeshapillai, John Guttag. A Data-driven Method for In-game Decision Making in MLB. Massachusetts Institute of Technology Cambridge.
- [7] Johnathon Tyler Clark. Regression Analysis of Success in Major League Baseball. Spring 5-5-2016.
- [8]Bock, J. R. “Pitch Sequence Complexity and Long-Term Pitcher Performance”. Sports (3 2015), pp. 40–55.
- [9] Hoang, P. “Supervised Learning in Baseball Pitch Prediction and Hepatitis C Diagnosis”. NC State University, Ph.D. Thesis (2015).
- [10] Michael Hamilton, Phuong Hoang, Lori Layne, Joseph Murray, David Padgett, Corey Stafford and Hien Tran. Applying Machine Learning Techniques to Baseball Pitch Prediction.
- [11] Kaan Koseler, Matthew Stephan: Machine Learning Applications in Baseball: A Systematic Literature Review.
- [12] Sidle, Glenn Daniel. Using Multi-Class Machine Learning Methods to Predict Major League Baseball Pitches. PhD thesis, North Carolina State University. 2017.

- [13] Attarian. A comparison of feature selection and classification algorithms in identifying baseball pitches. In International MultiConference of Engineers and Computer Scientists, 263–268. 2013.
- [14] Baseball Savant. Statcast Catch Rates. (Accessed January 15, 2020)  
[https://baseballsavant.mlb.com/statcast\\_catch\\_probability](https://baseballsavant.mlb.com/statcast_catch_probability)
- [15] Daniel Feltey, Spencer Florence and Shu-Hung You: Predicting Baseball Pitching Outcome EECS 349 Machine Learning, Northwestern University
- [16] <https://pandas.pydata.org/> (Accessed January 15, 2020)
- [17] <https://numpy.org/> (Accessed January 15, 2020)
- [18] Mart'ın Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng: TensorFlow: A System for Large-Scale Machine Learning .
- [19] <https://keras.io/> (Accessed December 20, 2019)
- [20] Leo Breiman. Random Forests. Statistics Department University of California Berkeley January 2001
- [21] Durgesh K. Srivastava, Lekha Bhambhu: data classification using support vector machine.
- [22] D. STATHAKIS. How many hidden layers and nodes? Joint Research Centre (JRC) of the European Commission, Institute for the Protection and Security of the Citizen (IPSC), Monitoring Agriculture with Remote Sensing (MARS) Unit, Food Security Action, TP266, 21020 (VA), Ispra, Italy.
- [23] Matthew D. Zeiler. ADADELTA: ANADAPTIVELEARNINGRATEMETHOD, New York University, USA.
- [24] Diederik P. Kingma, Jimmy Ba. Adam: A Method for Stochastic Optimization.
- [25] Geoffrey Hinton, Nitish Srivastava, Kevin Swersky. Lecture 6e rmsprop: Divide the gradient by a running average of its recent magnitude.
- [26] <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/> (Accessed December 16, 2019)
- [27] Marius-Constantin Popescu, Valentina E. Balas, Liliana Perescu-Popescu, Nikos Mastorakis. Multilayer Perceptron and Neural Networks.

- [28] Paul Gamble. Implications and Applications of Training Specificity for Coaches and Athletes. June 2006.
- [29] Kaan Koseler, Matthew Stephan. A Survey of Baseball Machine Learning: A Technical Report. Miami University. 2018.
- [30] T. L. Doolittle. Human Factors in Sport: An Overview. September 1, 1986.
- [31] Aurelien Geron. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts.
- [32] Matt E. Ryana, Marshall Grammb and Nicholas McKinney. Information effects in major league baseball betting markets.

## Appendix I

---

The Appendix section contains shortcut description about PITCHf/x system and shortcuts about ball types. Allowing readers to understand the importance of the system and the data it holds.

|                |                    |
|----------------|--------------------|
| CH - Changeup  | KC - Knuckle curve |
| CU - Curveball | KN - Knuckeball    |
| EP - Eephus    | PO - Pitchout      |
| FC - Cutter    | SC - Screwball     |
| FF - Fastball  | SI - Sinker        |
| FS - Splitter  | SL - Slider        |

- **start\_speed**: the pitch speed, in miles per hour and in three dimensions, measured at the initial point, y0. Of the two speeds, this one is closer to the speed measured by a radar gun and what we are familiar with for a pitcher's "velocity".
- **end\_speed**: the pitch speed measured as it crossed the front of home plate.
- **sz\_top**: the distance in feet from the ground to the top of the current batter's rulebook strike zone as measured from the video by the PITCHf/x operator. The operator sets a line at the batter's belt as he settles into the hitting position, and the PITCHf/x software adds four inches up for the top of the zone.
- **sz\_bot**: the distance in feet from the ground to the bottom of the current batter's rulebook strike zone. The PITCHf/x operator sets a line at the hollow of the knee for the bottom of the zone.
- **px\_x**: the horizontal movement, in inches, of the pitch between the release point and home plate, as compared to a theoretical pitch thrown at the same speed with no spin-induced movement. This parameter is measured at y=40 feet regardless of the y0 value.
- **px\_z**: the vertical movement, in inches, of the pitch between the release point and home plate, as compared to a theoretical pitch thrown at the same speed with no spin-induced movement. This parameter is measured at y=40 feet regardless of the y0 value.

- **px**: the left/right distance, in feet, of the pitch from the middle of the plate as it crossed home plate. The PITCHf/x coordinate system is oriented to the catcher's/umpire's perspective, with distances to the right being positive and to the left being negative.
- **pz**: the height of the pitch in feet as it crossed the front of home plate.
- **x0**: the left/right distance, in feet, of the pitch, measured at the initial point.
- **y0**: the distance in feet from home plate where the PITCHf/x system is set to measure the initial parameters. This parameter has been variously set at 40, 50, or 55 feet (and in a few instances 45 feet) from the plate at different times throughout the 2007 season as Sportvision experiments with optimal settings for the PITCHf/x measurements. Sportvision settled on 50 feet in the second half of 2007, and this value of  $y_0=50$  feet has been used since. Changes in this parameter impact the values of all other parameters measured at the release point, such as `start_speed`.
- **z0**: the height, in feet, of the pitch, measured at the initial point.
- **vx0, vy0, vz0**: the velocity of the pitch, in feet per second, in three dimensions, measured at the initial point.
- **ax, ay, az**: the acceleration of the pitch, in feet per second per second, in three dimensions, measured at the initial point.
- **break\_y**: the distance in feet from home plate to the point in the pitch trajectory where the pitch achieved its greatest deviation from the straight line path between the release point and the front of home plate.
- **break\_angle**: the angle, in degrees, from vertical to the straight line path from the release point to where the pitch crossed the front of home plate, as seen from the catcher's/umpire's perspective.
- **break\_length**: the measurement of the greatest distance, in inches, between the trajectory of the pitch at any point between the release point and the front of home plate, and the straight line path from the release point and the front of home plate.
- **des**: a brief text description of the result of the pitch: Ball; Ball In Dirt; Called Strike; Foul; Foul (Runner Going); Foul Tip; Hit by Pitch; In play, no out; In play, out(s); In play, run(s); Intent Ball; Pitchout; Swinging Strike; Swinging Strike (Blocked).
- **id**: a unique identification number per pitch within a game. The numbers increment by one for each pitch but are not consecutive between at bats.
- **type**: a one-letter abbreviation for the result of the pitch: B, ball; S, strike (including fouls); X, in play.

- **x, y:** the horizontal and vertical location of the pitch as it crossed home plate as input by the Gameday stringer using the old Gameday coordinate system. I'm not sure what units are used or where the origin is located. Note that the y dimension in the old coordinate system is now called the z dimension in the new PITCHf/x coordinate system detailed below.
- **sv\_id:** a date/time stamp of when the PITCHf/x tracking system first detected the pitch in the air, it is in the format YYMMDD\_hhmmss.
- **pitch\_type:** the most probable pitch type according to a neural net classification algorithm developed by Ross Paul of MLBAM.
- **type\_confidence:** the value of the weight at the classification algorithm's output node corresponding to the most probable pitch type, this value is multiplied by a factor of 1.5 if the pitch is known by MLBAM to be part of the pitcher's repertoire.