**Visesh Kumar Jaiswal (01866194)**

**MTH 540-01**

**Professors: Alfa Heryudono, Scott Field**

**University of Massachusetts Dartmouth**

**vjaiswal1@umassd.edu**

BASEBALL PITCH PREDICTION

ABSTRACT

In this paper, I am predicting the pitch type of a baseball pitch by using the data produced by the Flight Scope Strike. During the process, few pre-visualizations were performed which can be used to evaluate the performance of the players as well. The radar which is being used in our college is producing a lot of data about the pitch but was not able to map the correct pitch type. That work was being performed manually. I analyzed the data which was provided and used few Machine learning algorithms which is classifying the correct pitch type with a good accuracy. Using this, I can then extend this classification task to prediction by utilizing features only known before a pitch is thrown. I am keeping this task as my future work. For now, I have classified the pitch type which can overcome the problem of mapping it manually. There are several pitch types but, in our classification, we will be only using four of them that are fastball, curveball, changeup and slider.

We then extend the classification task to prediction of the pitch type before a pitch is thrown. It can be implemented through a probabilistic classifier such as Naïve Bayes which will be giving different probabilities of the next pitch type.

Cleaning and Processing of data has taken a lot of time but has performed thoroughly. Different Machine learning based models are being trained on the data set by using different methodologies. LDA, kNN, Naïve Bayes, Random Forest are some of the algorithms which are modelled as a part of this project.

1.Introduction

Baseball is one of the most popular sports in North America. In 2012, Major League Baseball (MLB) had the highest season attendance of any American sports league (MLB, 2012). Partially due to this popularity and the discrete nature of game play (allowing easy recording of game statistics between plays) and the long history of baseball data collection, baseball has become the target of significant mathematical and statistical analysis. Player performance, for example, is often analyzed so baseball teams can modify their roster (by drafting and trading players) to achieve the best possible team configuration. One area of statistical analysis of baseball that has gained attention in the last decade is pitch analysis.

To aid this study, baseball pitch data produced by the “Flight Scope Strike” which is available in our college for use in home games and practice settings. This equipment can produce the latest baseball data instantly and produces a .csv file which contains Information about the pitching and hitting data. This data contains useful information about each pitch. Several characteristics such as pitch speed, break angle, spin rate etc. In this paper we provide a machine learning approach to pitch prediction, using classification methods to predict pitch types.

I was able to improve performance by examining

Different types of classification methods and by taking a pitcher adaptive approach to feature set selection.

Every baseball game feature hundreds of pitch from 60 feet to 6 inches. Out of all, I have considered 4 of them based on the dataset provided, which are

Fastball:

Essentially, the old-fashioned fastball. This is the “heater,” a pitcher’s straight pitch that gets to the plate in (typically) the 90 mph-plus speed range.  The best fastball pitchers can reach closer to 100 mph.

Curveball:

This is a frequently used “breaking ball” pitch, which simply means it has movement. In the case of the curve, this is usually a “12 to 6” movement, as in on the face of a clock. It comes in slower than a fastball, then suddenly drops down as it reaches the plate.

Slider:

Like the curveball, a slider is about deception. It looks more like a fastball than a curve and has higher velocity. But when it reaches the plate, it breaks down and away from the hitter. A “slurve” is a breaking pitch that operates somewhere between a curveball and a slider, generally with “11 to 5” movement.

Changeup:

Perhaps the most deceptive pitch in a hurler’s arsenal. When throwing a changeup, a pitcher releases the ball exactly like a four-seam fastball. However, it’s thrown in the 80-mph range, causing batters (who have only a split second to read a pitch) to swing “ahead” of the pitch. This is one that causes many strikeout victims to bang their bat in frustration.

All these pitches varies based on the speed, movement and break

2. DATA PREPROCESSING AND CLEANING

A screenshot of a cell phone

Description automatically generatedDataset which has been provided was of different players of different teams. This data is of 8 weeks of League. Each week dataset has different team’s data in multiple csv files.

A screenshot of a cell phone

Description automatically generated

Fig 1: 8 weeks of data Fig 2: Each week data

It takes lot of time and effort to load multiple csv files. In order to avoid these multiple times of loading files, all these csv files had been merged into one csv file which has 56 variables of 8370 observations.

Among 56 variables, I have considered only 16 quantitative features including horizontal movement, vertical movement, spin rate, spin direction, break angle, start speed etc. Dataset consists of more than 3000 missing values, which needs to be filled. If the any of the 16 quantitative variables are missing, then those are filled with mean of the respective columns. But, if the predicted (pitch type) value was missing then those rows are removed completely.

A picture containing text, receipt

Description automatically generatedA screenshot of a cell phone

Description automatically generated

Fig 3: Before cleaning up the data Fig 4: After cleaning up the data

Given data has been partitioned into 75% and 25%, to train the model and to validate the result respectively.

A screenshot of a cell phone

Description automatically generated

Fig 5: Partitioning of Data

3.METHODOLOGY TO TRAIN THE MODEL

a) Linear Discriminant Analysis (LDA):

LDA is a well-established machine learning technique and classification method for predicting categories. Its main advantages, compared to other classification algorithms such as neural networks and random forests, are that the model is interpretable, and that prediction is easy. Linear Discriminant Analysis is frequently used as a dimensionality reduction technique for pattern recognition or classification and machine learning.[1]

Think of each case as a point in N-dimensional space, where N is the number of predictor variables. Every point is labeled by its category. (Although it focuses on [t-SNE](https://www.displayr.com/using-t-sne-to-visualize-data-before-prediction/), this [video](https://experiments.withgoogle.com/ai/visualizing-high-dimensional-space) neatly illustrates what we mean by *dimensional space*)[1]

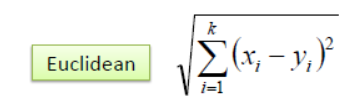
The LDA algorithm uses this data to divide the space of predictor variables into *regions*. The regions are labeled by categories and have linear boundaries, hence the "L" in LDA. The model predicts the category of a new unseen case according to which region it lies in. The model predicts that all cases within a region belong to the same category [1]

b) K- Nearest Neighbour (kNN):

K-Nearest Neighbors is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.[2]

It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data.[2]

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbour.[2]



Training of the model by considering features like horizontal movement, vertical movement and start-speed are taken two at a time. Firstly, removed all those rows which has outliers for features mentioned above. Secondly, in order to get the better results, I have normalized/scaled down the values between 0 and 1.

c)Principal Component Analysis (PCA):

PCA is a type of linear transformation on a given data set that has values for a certain number of variables (coordinates) for a certain amount of spaces. This linear transformation fits this dataset to a new coordinate system in such a way that the most significant variance is found on the first coordinate, and each subsequent coordinate is orthogonal to the last and has a lesser variance. In this way, you transform a set of x correlated variables over y samples to a set of p uncorrelated principal components over the same samples.

Where many variables correlate with one another, they will all contribute strongly to the same principal component. Each principal component sums up a certain percentage of the total variation in the dataset. Where your initial variables are strongly correlated with one another, you will be able to approximate most of the complexity in your dataset with just a few principal components. As you add more principal components, you summarize more and more of the original dataset. Adding additional components makes your estimate of the total dataset more accurate, but also more unwieldy.[3]

PCA has been applied to our dataset as there were 16 columns, 16 principal components have been made. But we will be considering only first 10 as more than 85% variance has been explained by them

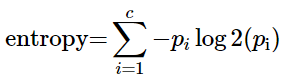
A picture containing text

Description automatically generated

Fig 6: First 10 principal components showing above 85% variance

d) Decision Tree:

Decision tree learner is a technique of machine learning. As its name implies, the prediction or classification of outcomes is made going from root to leaves. The tree is made up of decision nodes, branches and leaf nodes. The tree is placed upside down, so the root is at the top and leaves indicating an outcome category is put at the bottom. At the root, all classifications are mixed, representing the original dataset. Then the tree grows to the first node where a certain feature variable is used to split the population into categories.  Because the parent population can be split into in numerous patterns, we are interested in the one with the greatest purity. In technical terminology, purity can be described by entropy.

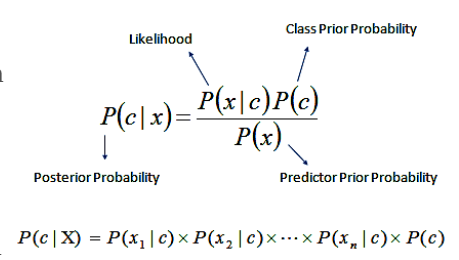


where c is the number of different classifications. For the research of mortality outcome, c equals to two. *pi* is the proportion of observations falling into class *i*. From the formula, we can see that entropy is zero when the population is completely homogenous; and 1 indicates the maximum degree of impurity. Other statistics such as Gini index, Chi-square statistics and gain ratio are also employed to describe the purity and to decide the best splitting.

The tree will stop growing by the following three criteria: (I) all leaf nodes are pure with entropy of zero; (II) a prespecified minimum change in purity cannot be made with any splitting methods; and (III) the number of observations in the leaf node reaches the prespecified minimum one ([2](https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4980381/#r2)). There are algorithms that allow the tree to grow and then the tree is pruned. Pruning technique aims to address the problem of overfitting as can occur in other parametric regression models.[4]

e) Naïve Bayes:

The *Naïve Bayes classifier* is a simple probabilistic classifier which is based on Bayes theorem but with strong assumptions regarding independence. Historically, this technique became popular with applications in email filtering, spam detection, and document categorization. Although it is often outperformed by other techniques, and despite the naïve design and oversimplified assumptions, this classifier can perform well in many complex real-world problems. And since it is a resource efficient algorithm that is fast and scales well[5]



f) Random forest:

Random forest is a way of averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of overcoming over-fitting problem of individual decision tree.  
  
In other words, random forests are an ensemble learning method for classification and regression that operate by constructing a lot of decision trees at training time and outputting the class that is the mode of the classes output by individual trees. Random forest comes at the expense of some loss of interpretability, but generally greatly boosts the performance of the final model. Random Forest is one of the most widely used machine learning algorithm for classification. It can also be used for regression model (i.e. continuous target variable) but it mainly performs well on classification model (i.e. categorical target variable). It has become a lethal weapon of modern data scientists to refine the predictive model. The best part of the algorithm is that there are a very few assumptions attached to it, so data preparation is less challenging and results to time saving.[6]

g) SMOTE:

Before discussing about the smote algorithm lets talk about Data Imbalancement. It happens when one class outnumbers other class by a substantial proportion. The predictive model developed using machine learning algorithms could be biased and inaccurate.

A close up of a logo

Description automatically generated

Fig 7: Data Imbalancement

For our dataset, most of the proportion of the data are fastball as shown below.

A close up of a screen

Description automatically generated

Fig 8: Proportion of Pitch Type in our data

In regard to synthetic data generation, synthetic minority oversampling technique (SMOTE) is a powerful and widely used method. SMOTE algorithm creates artificial data based on feature space (rather than data space) similarities from minority samples. We can also say, it generates a random set of minority class observations to shift the classifier learning bias towards minority class.

To generate artificial data, it uses bootstrapping and k-nearest neighbors. Precisely, it works this way:

Take the difference between the feature vector (sample) under consideration and its nearest neighbor.

Multiply this difference by a random number between 0 and 1

Add it to the feature vector under consideration

This causes the selection of a random point along the line segment between two specific features.[7]

A close up of a map

Description automatically generated

Fig 9: SMOTE algorithm

After applying the SMOTE algorithm in our dataset, the pitch type was equally balanced as shown.

A screenshot of a cell phone

Description automatically generated

Fig 10: Balanced Proportion of Pitch Types

4. RESULTS

LDA Output:

A screenshot of a cell phone

Description automatically generated

Fig 11: Confusion Matrix & Accuracy of LDA

In the above figure there is a table which is the confusion matrix[8]. If we sum up the diagonals and divide it by total sum of the table, we will get the accuracy. It’s even mentions in the above figure.

KNN Output:

A screenshot of a cell phone

Description automatically generated

Fig 12: Confusion Matrix of kNN

A close up of a map

Description automatically generated

Fig 13: Variation of Accuracy with the k value.

Decision Tree Output:

A screenshot of a cell phone

Description automatically generated

Fig 14: Confusion Matrix & Accuracy of Decision Tree

A close up of a map

Description automatically generated

Fig 15: The Decision tree

In the above figure, it is showing up the probability of different pitches in the leaf node. First is the changeup followed by Curveball, Fastball and slider.

Naïve Bayes Classifier Output:

A close up of a piece of paper

Description automatically generated

Fig 16: Confusion Matrix & Accuracy of Naïve Bayes

Random Forest Output:

A screenshot of a cell phone

Description automatically generated

Fig 17: Confusion Matrix & Accuracy of Random forest

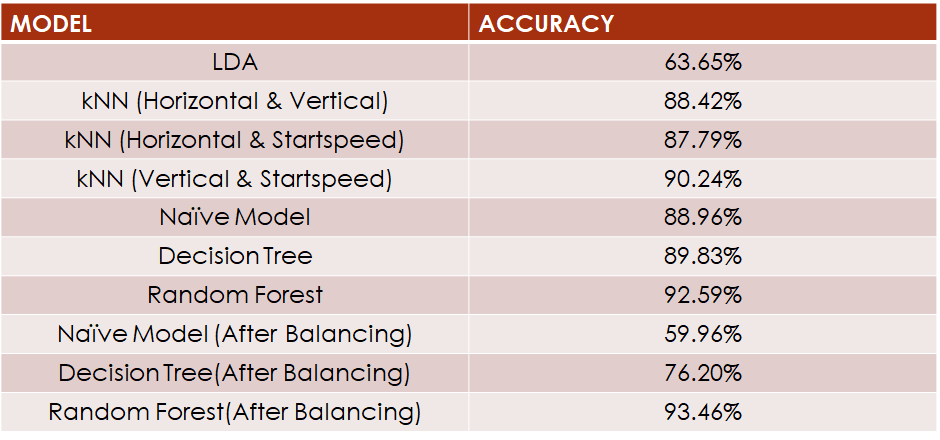


Fig 18: Comparison of Models accuracy

Few Visualizations for Performance:

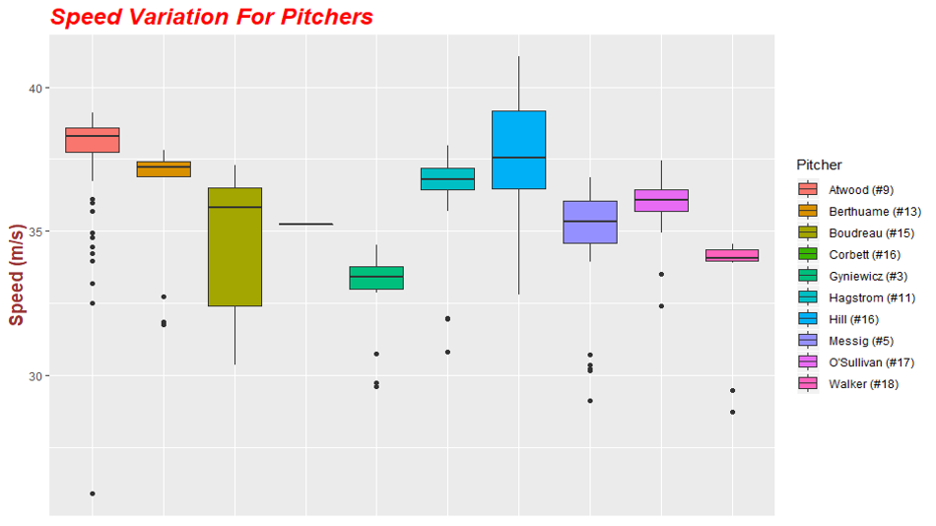


Fig 19: The figure above is a box plot[9] which shows the variation of speed of ball for different pitchers.

We can get few insights about the pitchers here. For example, the pitcher Hagstrom has the highest speed from all the other pitchers.

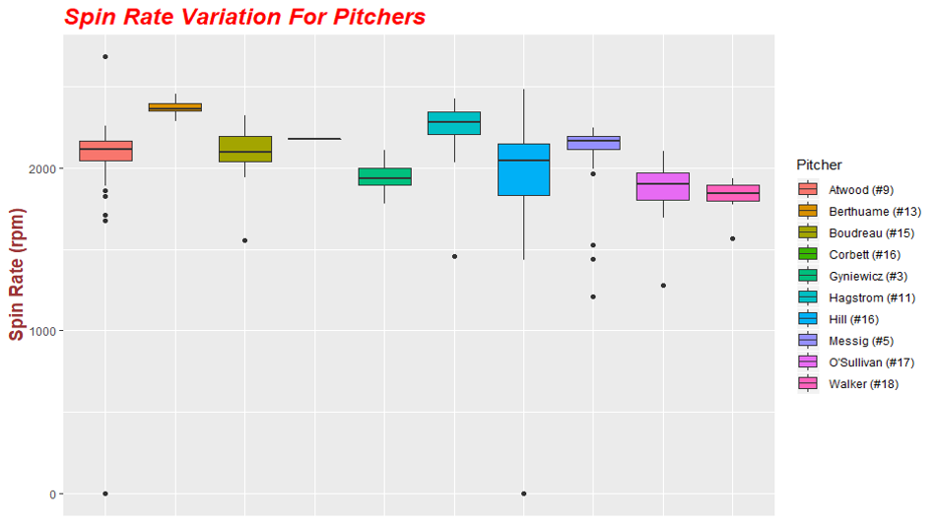


Fig 20: The figure above is a box plot[9] which shows the variation of spin rate of ball for different pitchers.

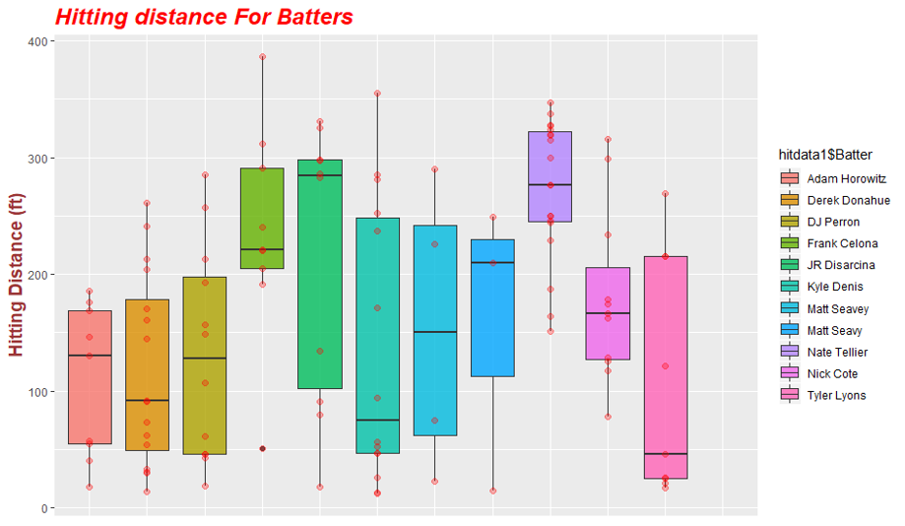


Fig 21: The figure above is a box plot[9] which shows the variation of Hitting distance for different hitters .

5.CONCLUSION & FUTURE WORK

There is much that can be done to improve the model. First, new features would be helpful. There is much game information that we did not include in our model, such as batting average, count of previous ball results, and others, which could help improve the prediction accuracy of the model. Implementing a model which will guess the type of next pitch before it’s pitched is the next step in the project. It can be sorted with the help of a Naïve Bayes Classifier with some extra feature added as the predictor. It will be interesting to see how the model performs as that would be very helpful for the both pitcher and hitter.

6.REFERENCES

[1] <https://www.displayr.com/linear-discriminant-analysis-in-r-an-introduction/>

[2] <https://www.saedsayad.com/k_nearest_neighbors.htm>

[3] <https://www.datacamp.com/community/tutorials/pca-analysis-r>

[4] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4980381/>

[5]<https://uc-r.github.io/naive_bayes>

[6] <https://www.listendata.com/2014/11/random-forest-with-r.html>

[7] <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-deal-imbalanced-classification-problems/>

[8] <https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>

[9] <https://www.tutorialspoint.com/r/r_boxplots.htm/>

Paper Reference:

<https://drive.google.com/open?id=18t-XAMUBEXAzqYNnoWANqIUBupHmpu-O>

7.VARIABLE STRUCTURE & SUMMARY:

A screenshot of a computer

Description automatically generated

Fig 22: Structure of Dataset

A screen shot of a social media post

Description automatically generated

Fig 23: Summary of Dataset