Brian McCabe

Data Science 4449 – Capstone

Denver University

Midterm

10/9/2022

Professor: Sada Narayanappa

Baseball Classification

Baseball is Americas past time, while it is mainly played in the United States baseball has many things to offer as far as statistics go. Sports analyses are constantly creating new formulas to measure players' performance. The goal of this project is to use the more well-known statistics of baseball to see if we can train an algorithm to classify a player's position on the field with respect to the players basic batting and fielding stats.

Coaches, scouts and even players are always tracking their stats and seeing where they can improve both on and off the field. This project can be used by baseball players to see how their stats define the positional player that they are. When a team is looking to fill a role instead of looking at a player's position, teams will try to mold players into new positions in order to fill their team.

My data set used in this analysis is the Lahman Baseball Database, which was officially released on January 24, 2015, and is updated at the end of every baseball season. The data includes multiple files for many different aspects of baseball. The data ranges from 1871 through 2021. My analysis will focus on two of the major parts of baseball, batting and fielding. With the goal of predicting the position that the player plays while on the field. The reason I am incorporating batting statistics in order to predict a player's fielding position is because I wanted to see if any batting statistic was prevalent to a certain fielding position.

The batting data set is a csv file with over 110k records. Players had records for multiple years, so I condensed the data set into unique players by combining records on playerID. Features included games played, at bats, runs, hits, walks, steals, strikeouts, and so on. All the batting stats are numerical integers or floating-point numbers.

The fielding data set is a csv file with over 147k records. Same as with the batting data set, I condensed the data on playerID and removed some fielding statistics that I had deemed

unnecessary to my analysis. Features included, games started, Inn Outs, Put Outs, Assists, Errors, and so on. All the fielding stats are numerical integers or floating-point numbers.

In order to create one data frame with all the information that I needed I combined the two data sets that I had cleaned on playerID and added 0 for all the NAN values as a value of zero represents the data correctly. With all the data merged and cleaning done I was left with 19964 records and 25 features including my target feature, player position.

Exploratory data analyses included looking at the distributions and outliers for each of my features. I used a library called dtale to help me visualize all my data. Dtale is an interactive data frame that can produce histograms, count plots, and summary statistics for each variable. I looked at the distribution of player positions and concluded that an algorithm could perform at 50% accuracy just by predicting each player as a pitcher. So, I had to take that into consideration later. Looking at a heatmap to determine correlations between variables I realized that all the features are positively correlated which is what I expected.

Other exploratory analyses included looking at feature selection with chi^2 and mutual info classification techniques. This helped me to identify which features would likely be most important in my analysis. So, I took the top features from my feature selection and looked at them in a pair plot with a hue overlay defining the player's position. This resulted in a better view of how some of the statistics were dispersed between player positions. When I saw this, I had a strong feeling that my classification would be able to work well.

Before starting my analysis, the merged data frame was split into a 70% training set and 30% test set. I ran these training and test sets through five different algorithms doing minimal parameter tuning. I used Gaussian Naïve bayes, Gradient Decent, K neighbors, Decision trees, and Random Forest Classifiers to see which would give me the best results. As predicted the

Gaussian and Gradient Decent classifiers did not perform well as they mostly predicted players as pitchers thus resulting in about 50% accuracy for both the training and test sets. Looking at the confusion matrix the Gaussian classifier often predicted players as pitchers. Whereas the Gradient Decent classifier correctly predicted most of the pitchers but then predicted most of the other players as outfielders.

For the K-Neighbors Classification, I constructed an elbow plot to find the optimal value for the n_neighbors. So based on the elbow plot I chose to use 4 n_neighbors, resulting in a training accuracy of 78% and test accuracy of 67%. It was getting better but not the results I was looking for. The confusion matrix showed me that the K-Neighbors classifier was doing a better job of learning the data but had trouble predicting short stops and 3rd basemen as well as having a slight tendency to default to predicting pitcher.

That is where the Decision Tree and Random Forest Classifiers come in. Without doing any tuning the results of these classifiers were much higher. Both had training accuracy of 99.9%. Decision Tree had a test accuracy of 73% and Random Forest had a test accuracy of 79%. The confusion matrix showed that the decision tree classifier had better results predicting short stops and 3rd basemen than the Random Forest classifier. But overall, Random Forest outperformed the Decision tree by getting more of the other positions predicted correctly.

With these results I felt comfortable choosing the Random Forest Classifier and hyper-tune it to increase accuracy. After hyper-tuning all the different parameters over countless Randomized Cross validations I was only able to decrease the training accuracy by about 1% and increase the test accuracy by about .2% so all in all the hyper-tuning did not do much to yield better results.

Since my hyper-tuning did not yield better results, I decided to try principal component analysis to see if that would make a difference in the outcome of the predictions. I had high hopes for this approach and boy did it fail me. The models were not even able to learn that they could predict pitcher every time and still get 50% accuracy. Two of my PCA models dipped below 50% accuracy and the highest one had 56% accuracy. Thus, resulting in me scraping the idea to do PCA as it did not help the results whatsoever.

A good amount of this project went as I expected. I knew some of the algorithms would learn to predict pitcher and perform at 50% as well as it being hard to predict short stops 2nd and 3rd basemen as they have very similar fielding stats. What was shocking to me was the poor performance of PCA and the feature importance. I assumed PO and Assists would be the most important which was correct, but I thought that with pitchers not often batting it would be easier to predict pitchers based on their batting statistics. That was not the case, assists and inn outs were most important in predicting pitchers. This was a surprising and unexpected result.

All in all, this project was a success. I was able to train an algorithm that was able to predict with about 80% accuracy the position a baseball player takes when fielding. This algorithm does particularly well with predicting pitchers, outfielders, catchers, and 1st basemen.

I learned a lot from this project. About baseball and how much thought goes into player statistics. About all the fun and interesting libraries that python has to offer to make data analysis visually appealing and more accessible. As well as all the intricacies of machine learning and hyper parameter tunning.