MLB Team Rank Prediction

Soobum Jun
University of Michigan

Introduction

Baseball is not just a sport, but a huge business that the total revenue of the season this year amounted to 10 billion dollars and boasted a combined attendance of nearly 110 million fans across Major League Baseball and Minor League Baseball[1]. It is natural that a large stake lies in the match outcomes for both professional league teams and baseball fans.

It is also an interesting sport that allows numerous combinations with diverse positions and tactics, which gives a large room for statistical analysis and data analytics. The result of baseball games is determined by more scores and less lost points. Each team in the game takes turns to attack and defend so that the factors determining scores and lost points can be clearly identified (scores depend on hits and base running and lost points rely on pitch and defense), which is great for statistical analysis. Also, although a play can be influenced by luck or bad luck, a result of a match that is an aggregation of many plays, or a rank after a season that is the sum of 162 games are more likely to be decided by a team's objective capability.

Lastly, baseball has the longest history as the first professional sports of more than 130 years and abundant publicly available data due to its number of games per season and well-accumulated data sources such as Lahman baseball data[2], which we used for our machine learning analysis.

We acquired the data that contains the individual players' performance for the history of MLB. We hypothesize that machine learning algorithms will enable to predict the team's rank while maintaining individual players' values in the prediction process and the outcome from this will be more accurate than forecasting teams' ranks based on the team's aggregated performance, which is a mean value of the performance of players in a team.

1 https://www.mlb.com/news/mlb-increased-viewership-attendance-in-2019
2 http://www.seanlahman.com/

Therefore, we aim to build a model that forecasts the ranks of the team, which accept invidiaul players' performance as an input using neural-network. Our model approach uses the past 30 years of Lahman baseball data when making a prediction.

Previous works

There have been many works to explain the results of matches with variables and forecast the outcomes of baseball matches statistically. It became The most famous efforts that represent these works is the Pythagorean expectation, a sports analytics formula devised by Bill James to estimate the percentage of games a baseball team should have won based on the number of runs they scored and allowed. Comparing a team's actual and Pythagorean winning percentage can be used to make predictions and evaluate which teams are over-performing and under-performing.

$$\text{Win Ratio} = \frac{\text{runs scored}^2}{\text{runs scored}^2 + \text{runs allowed}^2} = \frac{1}{1 + (\text{runs allowed}/\text{runs scored})^2}$$

Exhibit 1. Pythagorean expectation formula

The expected number of wins would be the expected winning ratio multiplied by the number of games played.

After Machine algorithms gained popularity in sports analytics fields, new attempts were made to find the most important features to determine the results and build the model to actually predict it. Recent studies concentrated on using and generating new statistics called sabermetrics in order to rank teams and players according to their perceived strengths and consequently applying these rankings to forecast specific games. One of the examples is César Soto-Valero's work[3], which used a dataset with accumulative sabermetrics statistics for each MLB team. The paper reveals that the classification predictive scheme forecasts game outcomes better than regression scheme, and of the four data mining methods used, SVMs produce the best predictive results with a mean of nearly 60% prediction accuracy for each team.

3 https://www.researchgate.net/publication/311862823_Predicting_Win-Loss_outcomes_in_MLB_regular_season_games_-_A_comparative_study_using_data_mining_methods

Methods and Process

1. Data Cleaning

The original data included various information for each team and each player of the American League. The dataset had more than 30 features for each batting and fielding information, including unnecessary variables for modeling such as players' names, salaries, and birth years. We first should extract the variables that we would need for modeling. The below chart is the data cleaning process we followed.

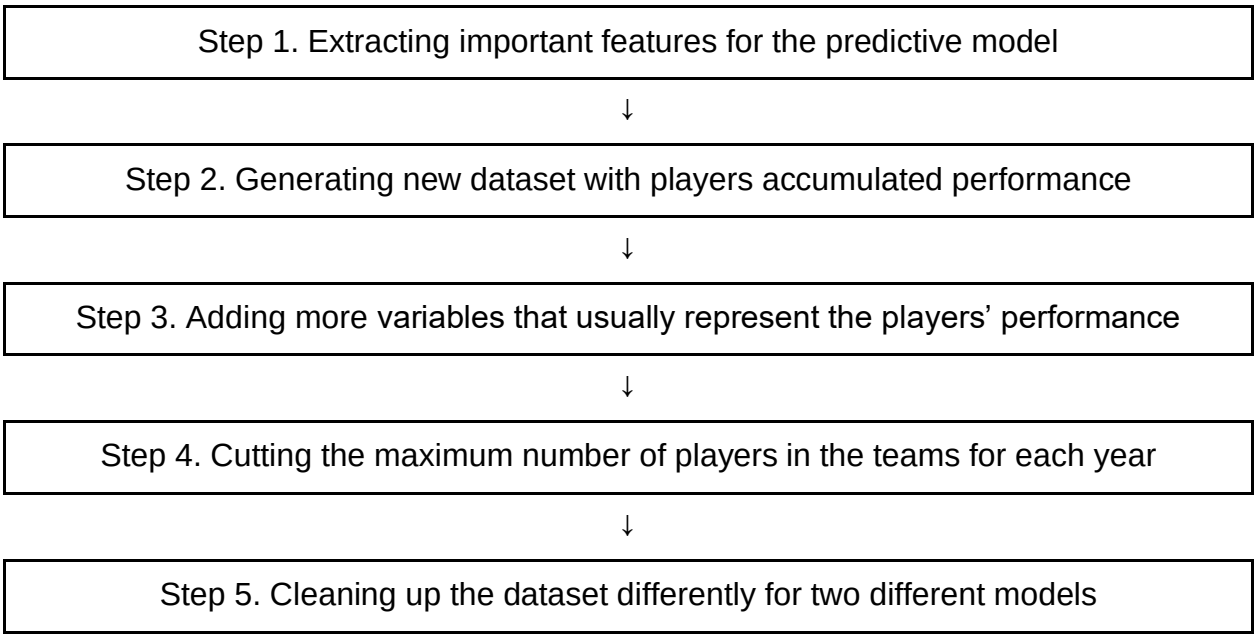


Exhibit 2. Data Cleaning Process Summary

1) Extracting important features for the predictive model

First of all, we should identify which information we would use for the ranking predictive model so 12 variables out of about 30 features were extracted from the original dataset, assuming that the rest variables were not significant to represent the players' performance. The final dataset included the number of games (*'G'*), total bases (*'TB'*), home runs (*'HR'*), and so on, which would influence to measure the team's rank eventually.

2) Generating new dataset with players accumulated performance

In order to make the model predict the team's rank based on players' performance, the dataset should have the players cumulated performance score year by year. Thus, we

got those accumulated scores with the 12 variables we got at the first step by using two for-loop from Python (see Appendix x).

3) Adding more variables that usually represent the players' performance

Besides the 12 features that were extracted from the first step, we calculated more variables that were commonly used for assessment of the player's performance such as hit rate, homerun rate, etc. For instance, we were given the number of hits (*'H'*) for each player and the number of the player's turn at batting (*'AB'*), and then the batting average became *H/AB.* Eventually, another 5 variables were added: Batting Average, On Base Percent, Slug Percent, On Base Plus Slugging and Batting Average on Balls In Play.

4) Cutting the maximum number of players in the teams for each year

The cleaned dataset from step 3 includes all the different number of players in each team for each player. To build the model, the independent variables shapes should be the same, so we needed to cut the number of players to make all the team include the same number of players for each year. Although the minimum number of players was 31 for the team ATL in the 1995 game, most teams indicated similar numbers for the major players who played the game the most, which was 15 players. In other words, about 15 players dominated games each year the most. Thus, we decided on the number 15 to cut the numbers for independent variables.

5) Cleaning up the dataset differently for two different models

Since we modeled two different approaches, a regression model and a neural network, and then compare the performance of both, two different cleaned datasets were required. For the neural network model, all the teams have 15 major players for each year so the dataset seemed fine as predictors. However, for the regression modeling, the independent variables could not be a matrix format but should be a vector, so we aggregate all the 15 players' numbers to one for each feature. We finally got the 839 rows which represented all players' aggregated year performance in each team and one number for its rank.

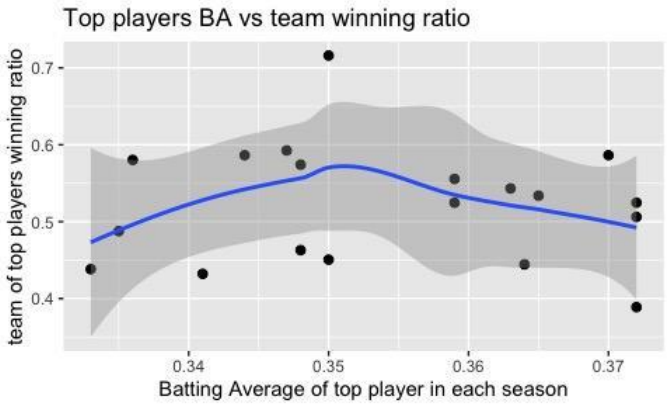
2. Model Building

1) Why neural network?

There are different types of regressors and classifiers for a prediction related problem. In this project, we focused on Neural Network using Keras deep learning library.

Individual sports, such as tennis and golf, can predict the outcome of a game only with the individual's stats. In the case of team sports, however, it is easy to find that the team's performance does not always correlate with an individual player, even though the player is highly capable and competent. The reason is that in the case of team sports such as soccer and baseball, the organized combination of players is as important as individual players' abilities.

For example, In a baseball game, no matter how well a start pitch was perfect, the team can't win if the closer show bad performances and lose the score. Conversely, even if a starter doesn't throw, he can win the game if the hitters play well. Therefore, these organic characteristics of a baseball game are not taken into account with simple reform or conventional logistic method.




```
        Pearson's product-moment correlation

data:  winpct$BA.x and winpct$WinPct
t = -0.21001, df = 17, p-value = 0.8362
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 -0.4936726  0.4128792
sample estimates:
       cor
-0.05086985
```

Exhibit 3. Correlations between top hitters' performance with the team's winning ratio

The graph above shows the players with the highest batting average for every year from 1985 to 2017 and compares them to their team's winning ratio. As shown in the scatter plot, we cannot observe any trends. Also, the correlation test result shows the number -0.05, which indicates that there is no or less correlation between top hitters.

Therefore, we decided to use an algorithm that can combine and understand an organic combination of players to predict team performance, such as neural networks.

2) Neural network model

We used a feed-forward neural network, which is the basic neural network models. We have to predict the team ranking they belong to based on all the features of each player. In other words, when the individual characteristics of the players were input in the input layer, the model was designed to derive the result value to the last output side through four hidden layers.

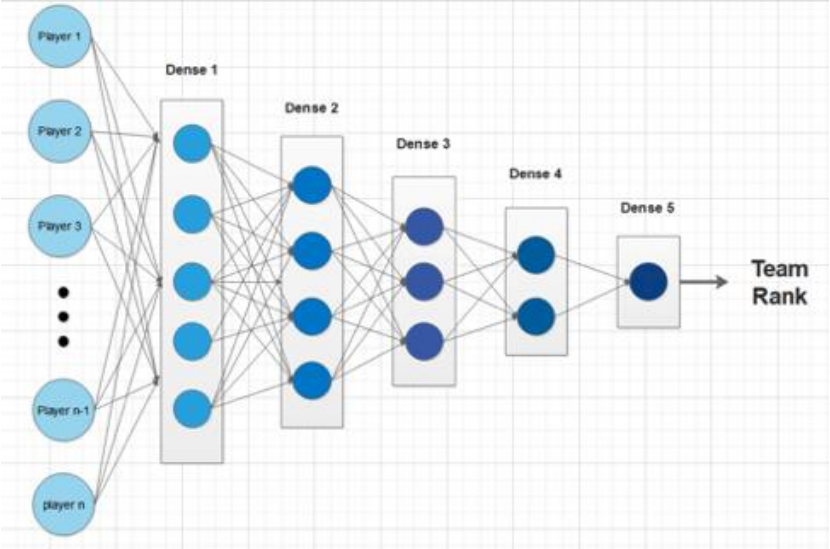


Exhibit 4. Our neural network model architecture

Train/Test split: First, a train set and validation set were divided into 8: 2. Missing values were removed during pre-processing, and each athlete's values were scaled using a minmax scaler.

Input layer: we set the 50 dimensions of output for the input layer with ‘relu’ activation function.

Hidden layer: we have 5 hidden layers with ‘relu’ activation, and its dimension is reduced from 50 to 1 to get one prediction value for each player.

Output layer: For the output layer, we use sigmoid and compile it with ‘Adam’ optimizer.

Loss function: We used Cross Entropy Error as a loss function due to prevent that the drawbacks that MSE would incur. When we used the MSE, the slope value in the weight calculation includes an adjustment factor of (output) * (1-output). As the calculated output approaches are values between 0.0 and 1.0, the value of (output) * (1-output) gets smaller as the values go through multiple layers. It will eventually make the weight insignificant and the neural network may stop learning and improving the accuracy.

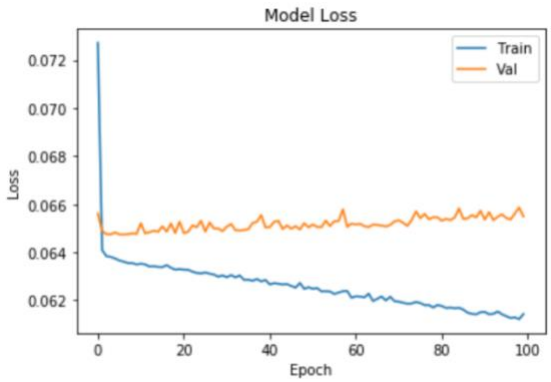
Insights and Conclusion

As outcomes of the two models, we got the rankings of teams from each model, the number of teams that matched in MLB for the last 30 years.

	ranking
0	3.362815
1	3.430460
2	3.506131
3	2.623464
4	3.152306
5	3.299520
6	2.962544
7	2.491156
8	3.071918
9	3.084839
10	3.439873

Exhibit 5. Predicted ranking

To prove our original hypothesis that considering individual players' performance into the machine learning algorithm will predict more accurately than the model based on team's aggregated mean performance values, we built the linear regression model as a base benchmark to compare with our neural network model. While Mean Absolute Value (MAE) of the regression model was 1.183, the MAE of the neural network model was 0.2. It demonstrates that predicting ranks that reflect individual players' capabilities in a team would provide more accurate forecast than predicting ranks with a single value that represents the team's capability.



Throughout the 100 epochs, our model showed the overfitting issue with dramatically decreasing loss for train data but gradually increasing loss for validation data. We thought about adding dropout layers between dense layers but had to leave that for the future improvement due to the time constraint issue.

Exhibit 6. Model loss for train and test

While the range of predicted ranks for each team from the neural network model is between 2.24 to 4.00, the linear regression generated the wider range of predictions from 1.94 to 4.39. Considering that each league consists of three divisions and each division has five teams, the results can be applied to the real applications of rank prediction.

Because predicted ranks are mostly values between 2 to 4, it would be difficult to apply those values directly as a rank of a team in a certain division in a league. Given that the predicted ranks are based on team members' performance, it is possible that several teams in a specific division have relatively similar competition powers (For instance, if team A have players whose performance are relatively similar to players in team B, the model will predict both their rankings as 3.07 and 3.24). Therefore, we believe the predicted rankings should be considered 'absolute rankings' that reflect only the performance of players who consist of a team rather than 'relative rankings' to the other teams in a team. To make the model applied to the real case, we need to set up another process that translates absolute ranks into relative ranks based on the size of the absolute rank values. (For example, if team A's rank is 3.07 and team B's is 3.24, the relative rank of team A is higher than team B).

A shortcoming that our model inherently contains is its inability to consider situational factors. As every sports game does, the team's rank can't perfectly coincide with the players' performance. The win or lose of baseball games are also affected by the situational factors. It is not clear what makes more favorable situations to a team. It can be pure luck, players' focus on the game, team synergies or coaches' abilities that make different situations with the same team members. These situational factors are what we cannot put into our considerations when building quantitative prediction models and calculating the ranks.

There are several tasks that we would like to do if with no time-constraint issue as a next step. First, we would adjust our neural network model with additional dropout layers to prevent overfitting and observe the improvement of model loss. Also, we would add a program that transforms the absolute ranking into the relative ranking information for the real-life application.