Predicting the outcomes of MLB games with a machine learning approach

Business Analytics Research Paper

Author: Tim Elfrink

Supervisor: Prof. dr. Sandjai Bhulai

June 13, 2018

Abstract

Baseball is an unpredictable sport where teams are really competitive with each other. Each team can win from the others and it takes 162 games each season to decide which teams will go to the playoffs. In this paper we will investigated if it can be predicted which team will win individual MLB games. This is done by using historical data of games and using different machine learning algorithms such as random forests and XGBoost.When using the XGBoost algorithm it shows the best results with an accuracy of 0.5552. This result can be improved when using more data, more computing power and better feature engineering.

Contents

1 Introduction

In the past baseball was a conservative sport just like all other sports. Coaches and managers based their decisions in the games based on their experience and instincts. But in baseball, there were always statistics involved. All events in the game were measured and players got valued if they were good in certain statistics. For example, if a player hit the most home runs he was considered the valuable player. This view of the game changed after an approach by the general manager of the Oakland Oaks in 2002, Billy 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 [4]. With the already available data and now the people applying different models, things are getting interesting. Is it for example possible to predict which team will win a game based on previous results?

Baseball is known to be really competitive, the best team of the season wins only 60% of all the games and the worst team 40%. In practice, this means that every team can win and loose from every other one. This seems a bit random who will win, but each team plays at least 162 games each season which makes it more significant.

In a paper [2] published in a machine learning course at Stanford University, a team tried to beat the betting companies by using data to predict which team will win. They simulated games and used Monte Carlo simulations to generate a distribution of possible outcomes. Their goal was to predict if the sum of the score would be higher than a certain value set by betting companies. In this paper they showed that they could be profitable when using their algorithm and bet on a selection of games. They said the following about a different approach in their paper:

> "Baseball games are sufficiently random and complex that it is not reasonable to model a game as a single vector of features and expect that any machine-learning algorithm will be able to give accurate predictions."

I would like to challenge this quote and will look if this is possible to do. I will use historical data and machine learning algorithms to predict who will win a baseball game.

2 Data preparation

In this research, Retrosheet data [1] is used to calculate all the statistics. The data consist of all events in games from 1930 till 2016. One event is defined as one at-bat in a game. The data in each event make sure that all necessary statistics can be extracted from it. It describes the current situation and what happened in the event. Each event has over 164 different features and each year has around 200,000 different events.

Per game, the goal is to predict who won the game. All events must first be aggregated by a unique game and the final score should be presented. This will be the start of the new dataset which now consists of the game identifier and the result. From this point, the data can be extended by more information provided by the events dataset which will contribute by predicting the score. To create a fair result, all statistics of data used should be based on data previous to the game date. This will make sure that all predictions will use data that happened before the actual game. In the data, there are four main categories that can give information about a matchup; offense, defense, pitching and general statistics. Per group, one can look at different factors that can play an important part in predicting the score. We can look at general statistics per category, but we can also look at groups such described down below and a combination of them. Maybe some teams play differently on certain days or ballparks. In this way, we can add information through the following features:

- Morning, day or night game
- Home or road game
- Ballpark
- Opponent team
- Day of the week

This all can be done per year or an accumulate of n games. n should be chosen carefully as it can represent how good a team is in general when chosen big, but also can discover the form of a team when it is small. Looking at multiple years might be not optimal as team rosters change over time which influences performance.

To generate these features manipulation of the data is performed. First the events are ordered by time and then grouped by the different groups. The cumulative sum is taken of the value and this new feature represents the total times a certain event happened until that time. From here the data will be summarized per date and the maximum value of the new features are taken. This represents the value of the different statistics per day. Now we have generated for example the number of hits and the number of at-bats for all teams at every date. Based on those features the real useful statistics and features can be generated.

Statistics which should be used for the model should be chosen carefully. In Appendix A some of the basic baseball terms and statistics are explained. It is important to use the right statistics as just adding data will generate features which are just noise. It is for example not useful to look how many hits a team hit, but it can be helpful to look at the batting average as this gives insights in the hitting capabilities with an equal number of opportunities of different teams. The generated statistics can be compared with the averages and the statistics of all other teams. Next to, the generated statistics of the appendix, team statistics are generated. This includes winning percentage, losing percentage, the fraction of runs scored by total runs and fraction of runs against by total runs.

The statistics also will be generated based on the last n events. This will generate the same features as before, but gives some information about the shape of the team they currently in. So instead of looking at the batting average for the season, maybe the batting average of the last 10 games can give more insights. As the team is clearly performing better than usual.

For every statistic, we also look at how much it differs from the current opponent. With this difference, we can see how much better or worse the team is in that area. This data is used fully as a simple algorithm can split on whether this value is positive or negative without combining two different features.

If we have generated all the statistics for each of the groups we can start with the prediction part. There are two approaches we can take. One of them is to predict who won: did the home team win (0/1). This would be a binary classification problem. But maybe we can predict better if we try to predict the difference of the actual score: $home\ score - visitor\ score$ and then transform it to a binary value. This will be a regression problem. The last approach might give more insights as the bigger the absolute value of the prediction value, the better the winning team apparently was. In this research, we will look at both approaches.

3 Methods

In this section, different methods are described to predict the objective. As described in the data preparation we have two ways to predict who won. We can treat it as a binary classification problem when we only look at the team that won or we can treat it as a regression problem which will determine the difference in the score.

First, we look at different machine learning models and after we compare the performance of them. We use different sets of models for the different objective functions. To compare results we optimize on the root mean square error (RMSE) for both objectives and look at the accuracy to compare the actual results.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(Prediction_i - Actual_i)^2}{n}}$$

$$Accuracy = \frac{True\ positives + True\ negatives}{All\ games}$$

We can use the accuracy as a method to compare results for the second objective if we map the difference to a binary classifier again. This way we can compare all results on the same metric.

The features that are generated might not all help in predicting our end goal, we will use different parts of the features and compare the results. We split on features that are based on the whole season and on features that are based on the last 50, 100, 200 and 500 events.

After we have the full dataset, we will perform some last preparations. We apply independent component analysis (ICA) [3] which adds new variables that are linear combinations of the original set such that the components are independent. A total of 100 new features are added. This will add some extra information about the original features. Not all features will help the model and even make it worse. That is why highly correlated features ($p>0.99$) are removed as well as linear combinations of each other.

We first split the prepared dataset into a train and test set. This split is done based on the date of the game. To obtain a meaning full result the test set contains the last $2,428$ games which are all games from the season 2016. The train set will be the complement of it consisting a total of $38,870$ games. This way we can simulate a situation that will look like the reality if we want to use this the final approach in practice. Because in practice you will also have a model that knows what happened in the past. 5-fold cross-validation is used to find the optimal parameters for the different models based on the RMSE. When the optimal parameters are found the accuracy is calculated based on the test set. Seeds are used to make the results reproducible. To speed up the hyperparameter optimization, only 10 % is used for this validation. The training is then done by training the whole training set again.

4 Results

To define a baseline we first look at an intuitive approach. We predict that all home teams will win all the games. We use the season 2016 as the validation set. The games in this season give an accuracy of 0.5301 in a total of 2,428 games. Which is already more than 6% better than a random approach.

We will try 4 different models, a generalized linear model, random forest, XGBoost and a boosted logistic regression for the binary problem only. We will produce results for the 2 different data sets ("season data" and "season data + recent events") and we will predict the target value in the two different ways described before. The first 3 models are chosen because they both work for classification problems as for regression problems. This way it is easy to compare if there will be differences in the different approaches. Also, the models are known to be good for handling numerical only datasets. For the binary classification problem, we also test another model, the boosted logistic regression model.

Both the random forest model and the XGBoost model are using trees. With the generated features the splits can be made on obvious variables such as the difference. In Figure 1 the top features are shown of the best XGBoost model. The most frequent tree split is the WHIP against a particular opponent. This makes sense from a logical perspective. The higher this value, the higher the chances of runs against as there are more opportunities for scoring for the opponent.

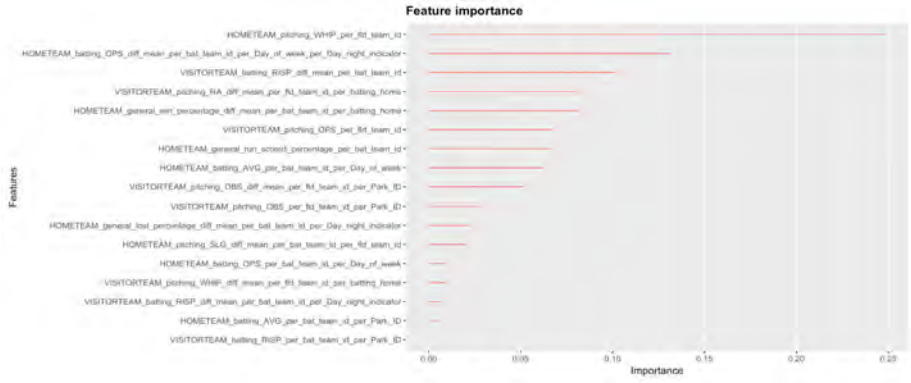

Figure 1: Feature importance best model XGBoost.

For the generalized linear model and the boosted logistic regression model, there are no parameters that can be tuned. For the other two there are a lot. In this research we limit it to the most important features. These are the learning rate and the number of trees. The other parameters in the models are kept default.

We used a grid search to find the optimal parameters, this is just trying all combination of set values. With cross-validation the optimal learning rate and

the number of trees are determined.

Table 1: Results of the RMSE of different models viewed as a regression problem

	Season (RMSE)	Season + Recent (RMSE)
Linear model	5.421368	247.838
Random forest	4.302865	4.31006
XGBoost	4.294094	4.293452

Table 2: Results of the accuracy of different models viewed as a regression problem

	Season (Acc.)	Season + Recent (Acc.)
Linear model	0.5078	0.5181
Random forest	0.5334	0.521
XGBoost	0.5502	0.5502

Table 3: Results of the accuracy of different models viewed as a binary classification problem

	Season (Acc.)	Season + Recent (Acc.)
GLM	0.5375	0.4749
Random forest	0.5408	0.5354
XGBoost	0.5552	0.5527
Boosted Logistic Regression	0.5478	0.5449

The results of the models are presented in Table 1, 2 and 3. We see that the XGBoost model is performing the best in all 3 different situations. The best accuracy is 0.5552 with only season data when seeing it as a binary classification problem. The result is 11% better than a random guess and 5% better than the home advantage. The optimal parameters used for this were 2 for the maximal depth of the trees and 0.005 for the learning rate.

Notable is that there is only one situation where the recent data gives a better result for the same model than without it. It probably gives too much noise with the data that it will not help after all. Also, we see that the there is not a big difference between the different objective functions and that the accuracy is even the highest at the binary classification problem.

5 Conclusion

Although the games can be very random, generated results are better than a random approach. With the right feature generation, there is indeed information which is known upfront of the game that can be an indicator of the outcome of the game. Interesting to see is that the difference in the outcome (regression problem) does not give better results than the classification problem. Also, we have seen that the more features added about recent statistics the performance decreases. This might have everything to do with the number of features added that are just noise and the algorithms just cannot handle this. Still the results are not that good that it can beat betting companies. In the discussion section possible improvements are discussed which might improve the performance to a result which can beat the betting companies.

6 Discussion

We can zoom into the best results of the regression problem. We take a look if there might be a difference in prediction the accuracy if we only take a look at predictions of the regression model which show how absolute values. We do this to check if there is a fraction of games we can predict with a higher certainty. We could divide the results into different groups with different confidence levels. This way we identify games that we could predict with a better confidence than others. In Figure 2, there is a plot of the regression problem and how it is mapped to the classification problem. We can look for an absolute value on the y-axis which can improve the accuracy of all "extreme" values. The practical use of this can be when using the research for betting purposes. By only betting on a select fraction of games with a high accuracy might be profitable.

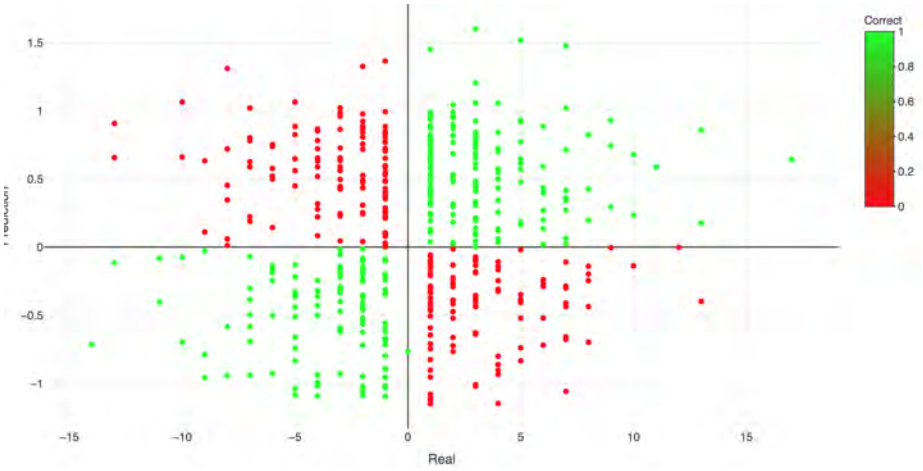

Figure 2: Plot of the regression problem and their errors.

A way to enrich the dataset with more quality is to look at the starting lineups of the teams. If the star player has a day off, the team might suffer from it. Also, players can have good momentum and that can be detected by the algorithms if it is in the data. It will be interesting to see if the prediction will be better or worse if the number of seasons increases. It will give more information, but it might generate data which will be too noisy for the latest prediction year. Also, one can look at different baseball statistics. Every statistic can give a little bit different information about the teams.

Now we only looked at statistics over only one season and the different number of events. It will be interesting to look at statistics over multiple seasons. Normally teams seem to perform quite stable over years and we might spot trends. Now all seasons were independent in the data preparation.

We generated a total of 6 different models. Maybe there are other machine learning models that can generate better results which are not explored yet.

Also combining all the models with ensemble techniques can boost the results. We can use the approach of the Monte Carlo Simulation [2], reproduce it and combine it with our results. Another way to increase the results can be to optimize different parameters as now only two parameters were optimized in the best model.

We can use boosting methods to combine multiple weak learners to obtain different results. We can use the different models with the different models for that. Bagging methods can also be used to get better results, generate different models which are trained on different subsets of the data and combine them. In this research there were restrictions on the computer power and time. When those limitations are dealt with, one can look at doing the 5-fold cross-validation on the whole dataset instead of only a part.

References

[1] *The information used here was obtained free of charge from and is copyrighted by Retrosheet. Interested parties may contact Retrosheet at "www.retrosheet.org".*

[2] Nico Cserepy and Robbie Ostrow. Predicting the final score of major league baseball games. 2015.

[3] A. Hyvärinen. *Independent Component Analysis: A Neural Network Approach.* Acta polytechnica Scandinavica: Ma. Finnish Academy of Technology, 1997.

[4] Michael Lewis. *Moneyball: The Art of Winning an Unfair Game.* W. W. Norton Company, 2003.

Appendices

A Baseball stats

- At bats (AB): Number of plate appearances (PA) - (Base on balls (BB) + hit by pitch (HPB) + sacrifice fly (SF) + sacrifice hit (SH))
- Batting average (AVG): Number of hits (H) / AB
- On-base percentage (OBP): (H + BB + HBP) / PA
- Slugging percentage (SLG): (H + 2 * number of doubles (2B) + 3 * number of triples (3B) + 4 * number of home runs (HR)) / AB
- On-base plus slugging (OPS): OBP + SLG
- Batting average with runners in scoring position (BA/RISP): AVG if runners are on second base and/or third base
- Walk and hits per inning (WHIP) = (H+BB)/IP
- Pitching Runs against (RA) = (9*R)/IP