Application of Machine Learning Techniques on Baseball Statistics to Predict Future Winners

Kyle Bushick, Austin Kim, and Will Molter
EECS 349, Northwestern University
Spring 2016
Contact us: kylebushick2018@u.northwestern.edu, austinkim2018@u.northwestern.edu, will.molter@u.northwestern.edu

Introduction

The goal of the project was to be able to create a classification model that can successfully predict the outcomes of professional baseball games based on the aggregation of statistics of the players who are going to play in the game for each team. More specifically, we desired to create a model that can determine who will win the matchup of two teams, given the individual player statistics for those playing for each team. Baseball is increasingly becoming a sport that is being analyzed from the perspective of not only feel and instinct, but also sabermetrics and statistical analytics. Many people, especially team general managers, are now infatuated with using analytics to justify any decision they make.

We thought it would be interesting to test whether this infatuation is justified, and if so, which statistics turn out to be the best attributes/predictors. Figuring this out was probably one of the most interesting parts of the project. Beyond our team's own curiosity, we thought this problem to be relevant because it would draw interest among sports betting enthusiasts; of course, initial betting lines these days are determined from methods like machine learning, but after betting opens, the line movement is based entirely on people's bets. Then, if the machine-learned classifier were accurate, then it could determine when betting on one side or the other would be statistically prudent.

Dataset

The datasets used for this project were based on the box scores of the every MLB game in the last 15 years, available on *baseball-reference.com*. We partitioned our data into three sets (five years each) to reduce training time. This leaves over 11,000 examples in each set. Since we are interested in predicting future games, we wanted to extract only data that would be available before a game begins, and we also purposely discarded using the teams' records as attributes because we wanted to see if prior player performance could predict the games.

For each box score, we approximated the pre-game statistics with the post-game ones available on the box score page. Specifically, earned run average (ERA), batting average (BA), on-base percentage (OBP), on-base percentage plus slugging (OPS), and slugging percentage (SLG) for each player on each team were included. Additionally, to access other attributes of interest such as salary, we used full-season data for each team in each game, extracted from a different set of web pages. These pages also included many metrics not available on the box score, such as walks and hits per inning pitched (WHIP) for all the pitchers on the team, which can be used to assess "strength of bullpen", often talked about in baseball. Essentially, we formatted

the data of individual player statistics of every single MLB game within the last 15 years to create a training example for each game.

Unfortunately, we did not have access to the database underlying *baseball-reference.com*, and so in order to get our data we had to programmatically generate the URLs for the box scores of all the games in the last 15 years, retrieve the pages at these URLs, and then parse through the resulting .html files for the relevant statistics. We did the same for the season data for each team for all 15 seasons. To achieve this, we used Python's requests and BeautifulSoup libraries and ample time to retrieve and parse web pages.

Original approach and results

Prior to examining our initial results, the intention of our project was to find a model that would hopefully develop into a good enough classifier to be able to predict the outcomes of MLB games based on the statistics of players who will be playing up to that point in the season.

After creating the initial datasets, we first passed them into several different learners on Weka to gauge the accuracy. Figure 1 shows the results. We began using the default settings and then manipulated the settings to see if there would be a change in accuracy of greater than 1%. However, since there did not seem to be much improvement of the accuracy compared to that of the default settings, we decided to use the default features for each learner in future iterations.

Figure 1: Cross-validation accuracy on the datasets for team average statistics across different learners.

Algorithm	2015-2011	2010-2006	2005-2000
AdaBoostM1	**62.84**	**63.17**	**60.85**
BayesNet	62.16	62.51	59.5
DecisionTable	62.37	63.13	60.41
IBk	54.29	54.95	54.29
J48	**62.83**	**61.86**	**61.41**
Logistic	63.95	63.89	63.89
Multilayer Perceptron	63.85	62.68	61.37
RandomForest	62.16	61.92	60.55
SimpleLogistic	**63.89**	**63.86**	**63.7**
VotedPerceptron	**63.32**	**63.56**	**63**
ZeroR	53.49	55.16	53.13

Despite variation, the performance across the three datasets was similar. Because of relatively long training times, we decided to focus on just the most recent dataset, 2011-2015, referred to just as "the dataset" following.

New approach and results

After passing all our data with all the attributes into the learners, we were disappointed to see that the accuracy was consistently in the 60 percentile. At this point in the project, we decided that it would be more plausible to move on from trying to create an accurate predictor to seeing which attributes are the most valuable in terms of contributing to a good predictor. Therefore, we narrowed the learners we were using to four: ADABoostM1, J48, SimpleLogistic, and VotedPerceptron. These were among the top performers on the first pass, and also had relatively quick training times compared to their counterparts (i.e. VotedPerceptron is much faster than MultilayerPerceptron) so that we could more easily assess their performance on many different sets of attributes.

Since our initial datasets were constructed directly from the .html files and only contained the averages of each player's statistics, we needed to reconstruct our training data. Since parsing through the .html files was on the timescale of hours, we decided to take all of the data and create a json file which could be used to quickly generate new datasets. These new datasets did not average the team's stats, but instead took each batter individually. Thus we expanded from 10 attributes to 74 (9 batters* 4 stats * 2 teams + 2 pitching). We then created a number of datasets from this full data set, omitting either statistics, batters later in the lineup, or both.

The results from all of these attribute sets are tabulated in Figure 2. The figure indicates that excluding pitching stats from the dataset greatly decreases the accuracy, by almost 10%, regardless of the learner. From this, we learned that some statistics clearly have more value towards developing a good enough classifier to predict the winner of a game than others. Therefore, we changed the intention of our problem to be one where we look for attributes that are important to developing a predicting model.

With that in mind, we noticed that the dataset with just OBP performed best out of all the batting statistics. This is interesting because OPS is literally a sum of OBP and SLG and keeps track of how many bases a player produces rather than simply how often he gets on base. However, when we use a pruned tree to run our dataset on, OPS becomes a better attribute to split on than OBP. This might be because OBP and OPS are redundant, meaning the decision tree will not gain from splitting on both. Thus in this case, possibly just by chance, OPS was more informative than OBP, and so the tree picked it. It should also be recognized that in this discussion of accuracy, the majority of the values are within 1 or 2 percentage points of each other.

Figure 2: Cross-validation accuracy on the dataset for different attribute selection across four different learners. "First five" and "ff" refer to attribute sets constructed by considering the first five batters in the batting order and discarding the rest. A batting statistic in the name of an attribute set means that statistic was used. If no batting statistic is listed, all four are used. Unless otherwise stated, pitching is included.

Attribute set	AdaBoost	J48	SimpleLogistic	VotedPerceptron	Best
full	61.19	58.3	62.88	**63.14**	VP
first five	61.1	59.92	62.69	62.74	VP
ffBA	60.95	60.28	62.43	62.01	SL
ffBA/OBP	60.89	60.68	62.58	62.18	SL
ffOBP	60.98	60.82	62.76	62.32	SL
ffOPS	61.08	60.51	62.64	62.35	SL
ffSLG	61.1	60.82	62.59	62.54	SL
BA	60.79	59.77	62.92	62.31	SL
BA/OBP	61.17	59.17	62.81	62.56	SL
OBP	**61.34**	60.27	**63.22**	62.52	SL
OPS	61.02	59.91	62.94	62.87	SL
SLG	60.82	60.42	62.84	62.58	SL
Just Pitching	61.01	**60.89**	61.91	62.15	
No Pitching	54.76	53.57	56.42	56.37	SL
Best	**61.34**	**60.89**	**63.22**	**63.14**	SL

While considering the raw statistics of each batter is important, we also thought that in predicting the winner between two teams in a head to head matchup, considering the difference between the statistics would also be significant. For instance, if one team had really high stats, but played against a team with even better stats, we would expect the difference to be a better indicator than the raw numbers. Thus we generated another set of attributes that included these factors. The first set took the difference of each statistic for each batting slot, for instance the difference between Home and Away Batter 1 BA, SLG, OBP, and OPS. This gave us a set of 36 attributes, plus the two ERAs. Since we found OBP to be the most important batting statistic, we also made a set whose attributes were each batter's OBP and the difference between each. The third set looked to determine if including the difference resulted in the first half of the lineup mattered more. These accuracies are found in Figure 3. From these results we determined that while a difference in statistics may make intuitive sense, it does not seem to aid in creating a prediction model, which is a somewhat unexpected finding.

Figure 3: Accuracies on sets including the difference between statistics across four different learners.

Attribute set	AdaBoost	J48	SimpleLogistic	VotedPerceptron	Best Algorithm
OBP_diffandstat	61.35	60.19	63.25	62.83	SL
ffOBP_diffandstat	60.98	60.43	62.76	62.48	SL
justdiff	60.8	59.75	62.91	62.87	SL

Figure 4 shows performance on attribute sets using team season data. Strangely, adding these attributes greatly hurt the performance of the voted perceptron learner, and somewhat the decision tree, while the other two seemed relatively unchanged. It is likely this is because the team attributes will be the same for any two games played with the same two teams, and as baseball fans know, the outcome is not always the same.

Figure 4: Cross-validation accuracy on the dataset using team season attributes and box score attributes across four different learners. "Full" indicates all the data was used. A batting statistic in the name of an attribute set means only that statistic was used.

Attribute set	AdaBoost	J48	SimpleLogistic	VotedPerceptron	Best
team full	61.76	55.64	63.1	54.77	SL
team OBP	61.8	56.43	62.97	54.77	SL
only team	56.29	55.09	57.09	54.77	SL
salary	54.11	53.84	54.32	54.53	VP
pitching	61.01	59.24	62.57	54.41	SL
fielding	53.97	53.78	52.64	53.5	AB

Similarly, while the other two learners did not perform significantly worse, they did not perform better either. However, note that using the team season data for pitching and the starting pitcher ERAs gets 62.6% accuracy, only a percentage point below the best performance overall, which reinforces that pitching is extremely important in baseball. That alone can make or break a team. Fielding statistics, on the other hand, predict essentially no better than simply choosing the home team to win. This suggests that fielding ability either does not affect the outcome of baseball games or the fielding ability of teams is roughly equal, or both. Salary, often complained about because of the lack of a salary cap, does beat predicting the home team to win by about a percentage point, but clearly it is not very indicative of a winning team.

Conclusions

Perhaps as we could have expected, the prediction accuracy we achieved is not groundbreaking. Since baseball winner prediction has been studied many times, and accuracy has not seemed to improve over time, these many experiments could be used to establish a bound on the ability of machine learning to predict baseball game outcomes and inform a baseline statistical variance in baseball.

But as fans, we have found that pitching ERA is the single most important statistic and that getting on base matters more than getting hits. Machine learning can continue to provide interesting, if not profitable, information like this about baseball to fans and teams alike. The next step is learning and predicting specific player performance, pitches, pitching substitutions, and more in-game events, some of which already have been studied. Combining all of these, games or even entire seasons could be simulated, and perhaps an uninformed observer would think the stats were from a real baseball season. Maybe people would become fans of simulated baseball, just like the real game. Whatever the reason for one's interest in baseball, machine learning can provide results to care about.