Predicting A World Series Champion: A Machine Learning Approach Using MLB Statistical Data

Ashton Larkin, Austin Hwang, Claire Gammon, Paul Johnston
CS 478, Winter 2019
Department of Computer Science
Brigham Young University

Abstract

Many people like to predict which team will win a major sporting event. This is especially common for the NBA Finals, NFL Super Bowl, NCAA College Basketball March Madness, and the MLB World Series. There are lots of factors that determine which team will win, including statistical data gathered for competing teams throughout a given season. We wanted to see how reliable statistics are in predicting the winner of the MLB World Series. We collected regular-season data from each World Series team starting at the 1903 season, and then tested our data on various machine learning models to see how accurate of a prediction could be made as to who would win the World Series based on each team's statistics. The final results show that several machine learning models such as Random Forest, MLP with Backpropagation and KNN do a surprisingly good job of predicting the World Series winner for a given season, resulting in accuracy as high as 87.14% with Random Forest.

1 Introduction

The World Series is the final event of the MLB (Major League Baseball) season. Two teams compete in a multi-game series to determine who will be crowned as the MLB champion for a given season. Statistical data plays a large role in professional baseball, with a major shift occurring within the industry to hire statisticians and data analysts to create the best optimized baseball team. In the case for our experiment, the use of machine learning can also be used when it comes to predicting which team will win a game. Similar metrics and predicting models are used these days using machine learning to make these predictions (both before and in-game).[1] Using gathered statistics for machine learning has also become popular for evaluating how effective a team's training techniques carry over to regular and postseason in-game performance. People sometimes argue that statistics aren't enough to get a true prediction since it doesn't factor in things like home field advantage, weather, or momentum that a team has coming into a game. In addition, the World Series takes place in a best of 7 game series, which is a small sample size in comparison to the regular season where 162 games are played. This small sample size might give a large variance and not always have the best team win (statistically speaking). We wanted to see if making predictions using only statistics through something like machine learning would result in a reliable prediction framework for World Series winners. In order to determine this, we collected regular-season data from each team participating in the World Series for a given year starting at the 1903 MLB season. The collected data was then modified to remove unnecessary statistics so that it would enhance the performance of the machine learning models we ended up testing.

2 Methods

The completion of this project consisted of gathering the data, using it on machine learning models, analyzing the model results, modifying the data based on the results, and then experimenting with machine learning models again using the modified data. This was an iterative process that required a few changes in order to improve our initial results.

2.1 Data Sources

Our data was collected from the MLB website (https://www.mlb.com/). We decided to collect data from this website because it had statistics for every team starting at the 1876 season. There were a few issues we encountered when collecting the data that are described below.

Although statistics were available beginning in 1876, the World Series did not begin until 1903, which meant that we had less data available to us than we had originally thought. The World Series was also not held in 1904 or 1994. This left us with 114 seasons of statistics (1903, 1905-1993, 1995-2018) instead of the originally planned 142 seasons (1876-2018).

We also faced some issues when determining which features (statistics) to include for the World Series teams. We were originally planning on including home field advantage and defensive efficiency ratio as part of the data. Once we

[1] See Silver, Nate

started collecting data, we found that home field advantage was determined in many different ways over the years, making it difficult to figure out which team had home field advantage for a given season. We also learned that the defensive efficiency ratio statistic wasn't recorded until the 1999 season, which meant that we couldn't use this statistic since we were using statistics starting from the 1903 season.

After determining which seasons we would take data from and figuring out which statistics were available for these seasons, we decided to use the following statistics from each World Series team for a given season: team league, win percentage, batting average (BA), on-base percentage plus slugging (OPS), earned runs average (ERA), walks and hits by a pitcher (WHIP), passed balls (PB), and fielding percentage (FPCT). We decided on these statistics because they were available for each season dating back to 1903 and because we wanted to consider a team's hitting (BA, OPS), pitching (ERA, WHIP), and fielding performance (PB, FPCT) equally.

2.2 Data Sets

We used the statistics mentioned in the previous section to create four different data sets. Our first data set, the side-by-side data set, used the statistics for both teams of a given year as the features of an instance, with the label for that instance being the team that won. The side-by-side data set is nice because it includes all of the data we collected but may be problematic since there are a total of 16 attributes (8 attributes for a team, and 2 teams per instance), and only 114 instances in the data set. We tried to avoid this issue (curse of dimensionality) by making 3 other data sets: the differences data set which used the differences between each team's statistics for a given year, the ratios data set which used the ratio of each team's statistics for a given year and the individual teams dataset which used each team's statistics for a given year as a separate instance with the labels being 0 for a loss and 1 for a win. We thought that doing things like a ratio or difference would still keep some information regarding the relationship between the two team's statistics while reducing the number of attributes in the data set.

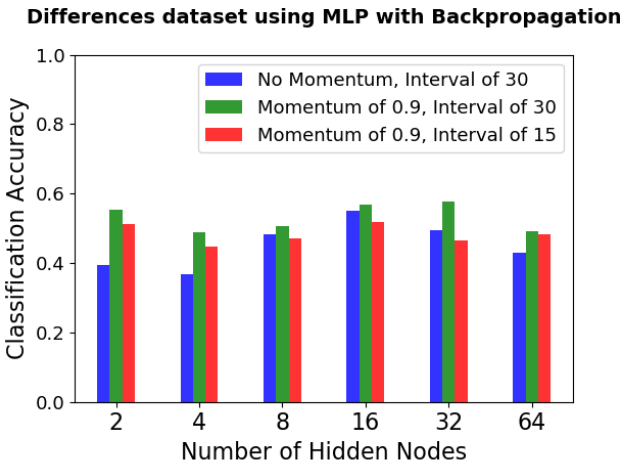
The data sets described above were put into the ARFF file format. The reason why we chose this format is because it allowed for us to try these data sets on algorithms that we implemented ourselves while also allowing us to use Weka (a machine learning software that also supports ARFF files) to try other machine learning algorithms.

2.3 Selected Models

We wanted to run as many models as possible to determine which would be the most effective for our data set. Initially we determined that we would use the models that we created throughout the semester to test the data. This included our perceptron, back-propagation, and k nearest neighbor models. We did not use our decision tree learners because they did not support the continuous data in our created data sets. We knew that perceptron may not have the best results, but it is simple and easy to understand and could potentially give us a good baseline.

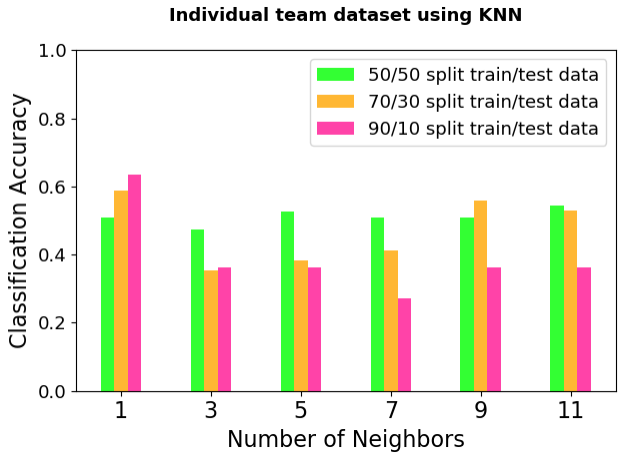
After testing our data sets on these models that we were familiar with, we wanted to experiment with other models. We discovered Weka[2], a software which provides a graphical user interface for testing a variety of machine learning algorithms. Using Weka, we were able to test other algorithms on our data sets. We tried to pick a variety of different algorithms and ended up selecting and testing ZeroR (baseline), OneR, J48 Trees, Random Forest, Bayes Net, Naive Bayes, jRip, Decision Table, PART, Simple Logistic, Lazy K Star, and SGD. Learning on a variety of models allowed us to find more effective algorithms.

3 Initial Results

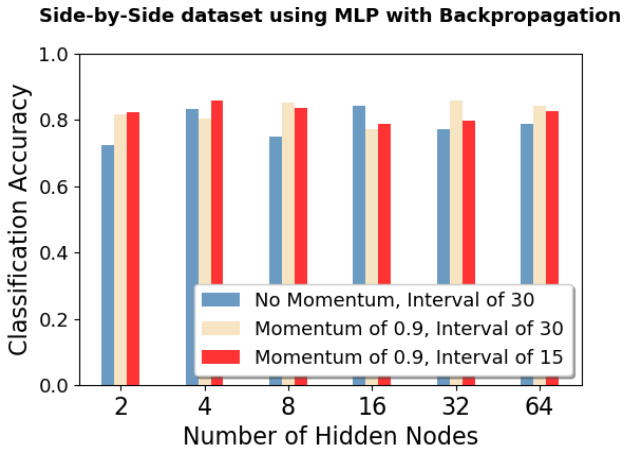


The graph above shows the initial results obtained when training a MLP with Backprop on the dataset that contained the differences in statistics of the two teams which competed in a given World Series. The highest accuracy was 57.8 percent which was achieved with one layer of 32 hidden nodes, a learning rate of 0.1, a momentum term of 0.9 and an interval size of 30 epochs. Interval size refers to the number of epochs ran with no improvement on the validation set before stopping the model.

[2] See Brownlee, Jason



The graph above shows the results of training a KNN model on the individual teams data set where each team is treated as a separate instance, and the highest score of two competing teams determines the predicted winner. A weighted distance metric was used when computing the nearest neighbor distance. Different splits of training and testing data were experimented. The highest accuracy was 63.6 percent which was obtained using a neighbor size of 1 with a 90/10 split between training and testing data.



When using the side-by-side data set on a MLP with Backprop, the model was able to get 85.8 percent accuracy by using one layer of 32 hidden nodes with a learning rate of 0.1, momentum term of 0.9 and epoch interval of 30. Interval size again refers to the number of epochs ran with no improvement on the validation set before stopping the model.

4 Data and Feature Improvements

As stated earlier, as we progressed through the project we discovered changes to the data that had to be made. We found that home field advantage was determined in a variety of ways throughout the years which made it difficult to know which team had home field advantage for a given season. For this reason, we decided to exclude the home field advantage as a feature. As we collected data, we also learned that the defensive efficiency ratio statistic is only recorded in seasons after 1998. Because we used statistics from 1903 - 2008, we decided to exclude this statistic because not enough instances of data would have a value for that feature.

After taking the data from the MLB website, most of the instances in our data set had a label of 0 due to the way World Series data was presented online (websites online typically listed the team that won the World Series first for a given year). We ran the data with most of the labels being 0 and found that our accuracy was pretty poor. We believed that this was because the learning model learned to almost always output 0 since most of the labels were 0. Fearing that our model had this bias, we changed the data so that half of the data's instances had a label of 1 and half of the instances had a label of 0 so that there would be less of a chance of overfit and bias in the learning algorithms.

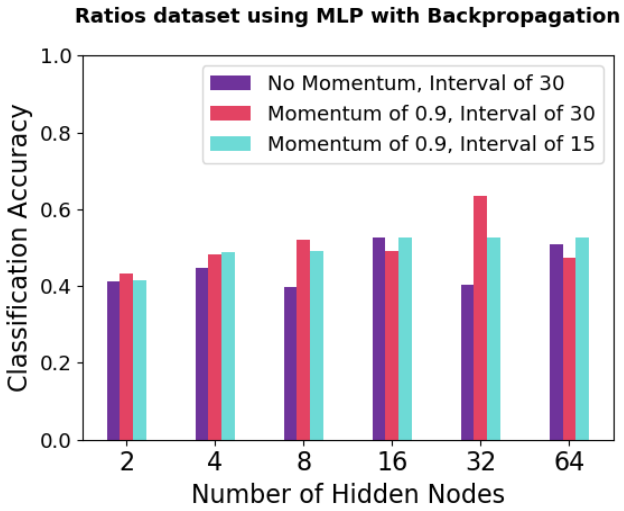
After changing the data so that half of the instances had a label of 0 and the other half had a label of 1, we ran this new data on machine learning models again. We found an immediate improvement in accuracies of the models. This generalization improvement validated our assumption that having many labels set to 0 was causing bias in the training.

We experimented with variations of the data sets. In the beginning of the project we initially tested the models on only three data sets: the side-by-side data set, the differences data set, and the individual teams data set. We thought an improvement to the data set containing the differences between the two teams' statistics would be to instead take the ratio of the two teams' statistics. We made a new ARFF file containing the ratios of the two teams' statistics and tested that on our learning models.

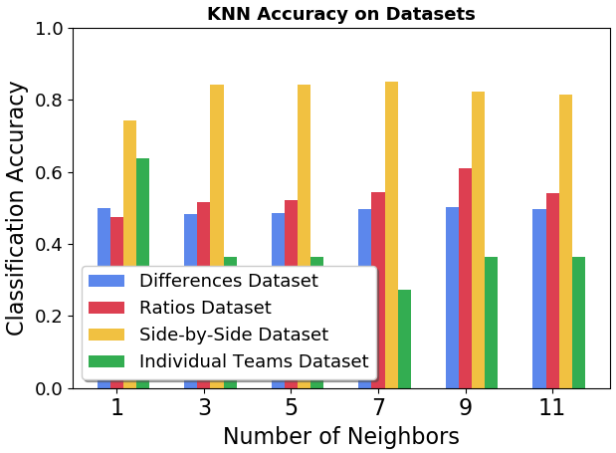
We found that there were significant differences in accuracies depending on which learning algorithm we used. We also wanted to see if our most successful data set, the side-by-side data set, could see even more accurate results when using other learning models. Because of this we used Weka to test our data on a variety of different learning algorithms. Weka was easy to use and provided a wide range of algorithms that added diversity to our learning models.

The most important thing we learned from our initial testing was the importance of the representation of the data. We found that the side-by-side data set produced significantly higher results than any of the other data sets that we tested. For this reason, it was the main focus of our testing moving forward.

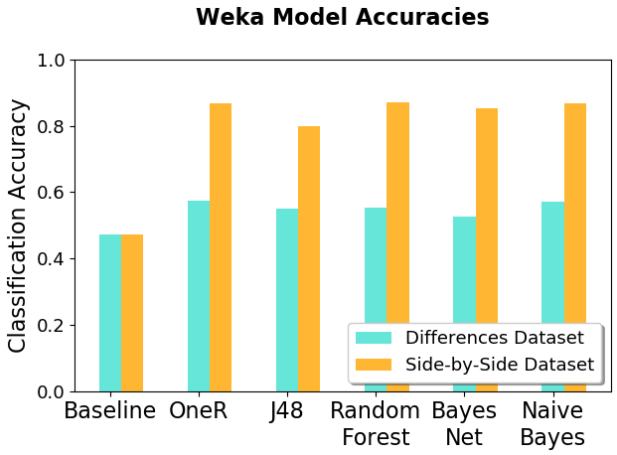
5 Final Results

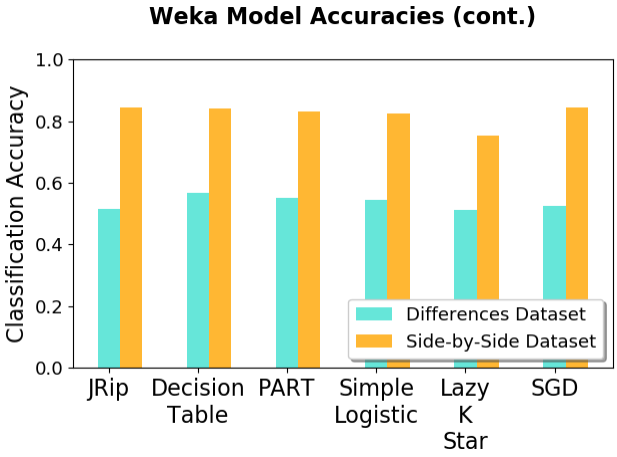


The graph above shows the accuracy on the Ratios data set using MLP with Backpropagation. The best accuracy was 63.5 percent which was obtained when using one layer of 32 hidden nodes a learning rate of 0.1, momentum term of 0.9 and an epoch interval of 30. Interval size again refers to the number of epochs ran with no improvement on the validation set before stopping the model. This was a slight improvement (5.7 percent) over the differences data set.



The best KNN accuracy was 85.2 percent from the side-by-side data set using seven nearest neighbors and weighted voting. Not quite as good as the 85.8 percent with backprop but pretty close. KNN was a convenient model to use because it allowed for regression and nominal outputs which allowed us to use all our data sets on the KNN model. It is clear that the side-by-side dataset does the best among the data sets no matter the number of neighbors used. The individual teams dataset decreases in accuracy as the number of neighbors increase but it had the second highest accuracy of the data sets.





The Weka models consistently performed best on the side-by-side dataset, with the best accuracy being 87.14 percent using Random Forest. This was an increase of 1.34 percent from using a MLP with Backpropagation on the same side-by-side dataset.

Overall the accuracy improved according to the data set more than the model used. Across multiple models from WEKA as well as the models we used in class, the side-by-side dataset outperformed the differences, ratios and individual data sets. The ratios data set, which was an improvement on the differences data set, increased the accuracy of the MLP with backpropagation by 5.7 percent. This shows the importance of data collection and how much the same data represented in a different way can affect results in machine learning.

6 Conclusions

After making the necessary improvements and obtaining the final results above, running Random Forest on the side-by-side data set obtained the highest classification accuracy (87.14%) in determining the World Series Champion.

Further investigation of how Random Forest works produced the following definition: "To say it in simple words: Random Forest builds multiple decision trees and merges them together to get a more accurate and stable prediction."[3] Random Forest is a relatively simple algorithm that usually produces accurate classification results because of its use of an ensemble of random decision trees.

In addition, the side-by-side data set produced the best results for most of the algorithms (Random Forest included) because it allowed a side-by-side comparison of the teams with each instance. The statistics of both teams were included and could be compared together, which explains why decision trees and backpropagation produced the best results from the algorithms that were tested.

7 Future Work

We were surprised by the results that we obtained, specifically in how accurate side-by-side data set was. There are further potential improvements that could be made in terms of the features that are selected, the range of chosen years to obtain statistical data, and additional algorithms that might verify or improve the results that were obtained. Overall, we believe these are the greatest factors in perhaps exploring potential improvements for our experiment of predicting a World Series Champion.

7.1 Future Refinements

Chosen features used for testing in these experiments were selected based on MLB statistical data that were most likely to have an effect on determining a better team based on three categories: batting, pitching, and fielding. These feature selections contained a level of bias based on what we perceived as the "best" telling statistical data that was provided in the MLB database. Some of the features were not selected (i.e wins above replacement (WAR)) based on modern day metrics that did not exist in prior years. Thus, statistics before 1903 were also not used as part of the conducted experiment since a significant amount of data was missing or not recorded for the features that were selected.

Baseball has evolved over time, and different statistical categories might reflect different things. For example, 1968 is famously dubbed as "the Year of the Pitcher", to where the MLB lowered the height of the pitcher's mound and shrunk the strike zone after "the resulting lack of offense had thrown baseball in a crisis."[4] That year, pitching statistics were far better historically, and hitting statistics were subsequently worse. The following year would return these statistics closer to the averages. The lesson learned is that the game is different during different time periods, and statistics today might tell a different story than 100 years ago. Therefore, another future refinement would be to concentrate on more recent data (i.e the past 10-20 years) and use other metrics to compensate for the reduction of data. Individual stats within the teams could be a possible consideration.

Other statistics other than those solely confined within the game of baseball could also be considered. Betting odds are a prime example of what people may consider the chances of a particular event happening. Observing what these odds were before the season or World Series starts could be another indicator in helping achieve more accurate results. This may also reflect a team's momentum heading into the World Series, weather conditions, or other factors that are not evaluated in pure statistical baseball data. These factors may also be used as potential features.

7.2 Other Models

The Weka tool provided by the University of Waikato was a helpful tool in exploring other models aside from the primary algorithms that were discussed in the report. In fact, ARFF files originated from the "Machine Learning Project at the Department of Computer Science of The University of Waikato for use with the Weka machine learning software."[5] Thus, it was easy to transfer the ARFF files that were created for this experiment and run tests with other algorithms using this software. All of these algorithms that were used using Weka are listed above. Many other algorithms are included in the Weka tool aside from the ones that were selected for our experiment. These could be used to verify the results that were obtained by our models, or perhaps obtain additional useful information to improve the predictions above.

Other machine learning tools that already contain built-in algorithms such as tensorflow, scikit, and pytorch can also be used for deep learning to obtain further analysis for better predicting a World Series champion. Granted, the format of the files and the process of running the tools may differ than our models or Weka, but these other tools could yield promising results that could verify and improve the results obtained in this report.

References

Attribute-Relation File Format (ARFF), 1 Nov. 2008, www.cs.waikato.ac.nz/ml/weka/ARFF.html.

Bogage, Jacob. "After 1968's 'Year of the Pitcher,' MLB Lowered the Mound. Now, the League Could Do It Again." *The Washington Post*, WP Company, 7 Feb. 2019, www.washingtonpost.com/sports/2019/02/07/after-s-year-pitcher-mlb-lowered-mound-now-league-could-do-it-again/?utm_term=.6b4581eba7e5

Brownlee, Jason. "Design and Run Your First Experiment in Weka." *Machine Learning Mastery*, 22 June 2016, machinelearningmastery.com/design-and-run-your-first-experiment-in-weka/.

[3] See References

[4] See Bogage, Jacob

[5] See *Attribute-Relation File Format*

Donges, Niklas. “The Random Forest Algorithm.” *Towards Data Science*, 22 Feb. 2018, towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd.

Silver, Nate. “2019 MLB Predictions.” *FiveThirtyEight*, 8 Apr. 2019, projects.fivethirtyeight.com/2019-mlb-predictions/.