NBA Predictions

Lee Richardson (lrichard)
Daren Wang (darenw)
Chi Zhang (chiz2)
Xiaofeng Yu (xiaofen1)

December 7, 2014

Abstract

Last season, the defending champion Miami Heat lost back to back games against the lowly Detroit Pistons and Milwaukee Bucks. While a two game stretch like this is certainly unlikely, it exemplifies the impact randomness has on basketball games. In this paper, we look at various predictions made regarding National Basketball Association (NBA) basketball games. Specifically, we see that we can predict the outcome of single games with a maximum of around 70% using various Machine Learing algorithms. We use these predictions to simulate the outcome of the 2012 and 2013 season, and our predictions landed us in the top 15 of all publically available NBA projection systems in both years [9]. Finally, we saw that the recently developed Regularized Adjusted Plus Minus (RAPM) statistic was a better predictor than any other statistic in our database.

1 Introduction

In recent years, there has been lots of talk in NBA circles regarding the rise of statistical analysis[10]. This has been in the works for years, most notably starting with Dean Oliver's Basketball on Paper, which was the cornerstone of basketball statistics for years. More recently, new data has become available, most notably from play by play events as opposed to box score statistics. There is also data from camera's which track player's motion, but this is mainly proprietary, with few exceptions.

In this paper, we look to harness all of the data publically available to predict the outcome of NBA basketball games. In section two, we will look at previous attempts at this problem as well as more recent data we could use for predictions. Next, we will describe how we obtained and put together our dataset, and give information on how one could access it. In section four, we describe our efforts of predicting games using different algorithms, features, and model selection techniques. Then we will discuss how we used our game predictions to obtain a distribution of for each teams record, and compare this with other projection systems. Finally, we will end with a discussion of what we learned and ways to improve our predictions.

2 Related Works

We looked into the literature to see if anyone had worked on the same or similar problems. We found a few papers, specifically [1] and [2], which also attempted to predict the results of NBA games. These papers used linear regression, logistic regression, naive bayes, and SVM's to predict the outcome of single games. They also used the same loss function that we are proposing, which gives us a prediction rate to shoot for when implementing our algorithms. Specifically, [1] achieved the highest single season classification rate of 73% in the 1996 season using linear regression. All of the other seasons/algorithm combinations had error rates from the mid-high 60's to low 70's.

One advantage we believe we have compared with these groups is that we have a more robust feature set, most notably we have regularized adjusted plus minus (RAPM). RAPM has been anointed by many as the next big thing [4] in the basketball statistics community, and we hope that using it as a feature can help differentiate our attempts at game classification. Taylor et al. have a great explanation of the statistic [3], but the basic idea is to split each game into miniature games, each one occurring in time periods when there's no substitutions. These ten players play a certain amount of possessions on offense and defense, and we can estimate their overall effect on both ends of the floor. RAPM adjusts for flaws in original plus minus, by correcting for the fact that player totals are heavily influenced by the play of his opponents and teammates, and by pooling information from previous seasons to reduce the margin of error.

3 Dataset

3.1 Challenges for dataset preparation

Several challenges have to be handled for the dataset preparation phase.

- There are currently no ready-made data sources for our experiments. What we can find is several websites on NBA games. Since we are interested in statistics of thousands of games with hundreds of players, it's not realistic to acquire those data manually from websites.
- Data sources from different websites have to be merged using a uniformed format. Different websites may treat player names and season matching differently. For example Roger Mason Jr. and Roger Mason are in fact same person from different websites, while uta and utah are different abbreviation for Utah Jazz.
- Data needs to be stored in a proper manner so that our future experiments can be implemented based on a uniformed and easily accessible API.

To meet those challenges, we are writing our own web crawlers to get data from websites, then made lots efforts on ETL on ensure different data sources were properly matched and merged. Finally we stored all data into a SQLite database. It's worth mentioning that we are making our dataset public online at
`https://github.com/leerichardson/game_simulation/blob/master/nba_rRegression_chi/nba.db`
Hopefully this will save some effort for anyone interested in research on NBA statistics recent years.

3.2 Data Sources

We did not have a processed dataset for this project, so we created our own database. The three main sources we used were ESPN's NBA website [7], basketball reference [6], and a new website from Jeremias Engleman [5]. We used the ESPN data to get information about all NBA games from 2009-2014. Specifically, this includes the game score, the home and away teams, the players involved and their individual statistics. Also from ESPN, we have a player database, which has 50 individual statistics for each player in each season. There are 7139 games in this dataset.

The next data source we used was basketball reference [6]. The main reason we used this site is because they have a larger individual player database, with information dating back to the 1950's and more advanced statistics, such as the widely used Player Efficiency Rating (PER), compared with the box score stats provided by ESPN.

The final source we used was from a website put together by Jeremias Engleman [5]. This site has RAPM statistics dating back to the 1980's. This statistic has been widely adopted in the NBA statistics community, and it's one of the few trustworthy stats which provides an individual assessment of defense. As we see below, RAPM is a very useful feature in predicting game outcomes.

3.3 Web Crawlers

To obtain all of these datasets, we used webcrawlers to pull them off their websites. All of these scripts can be found in our Github repository [8]. For the ESPN data, we used the BeautifulSoup package in the Python language. For the other two datasets, we used the XML package in R.

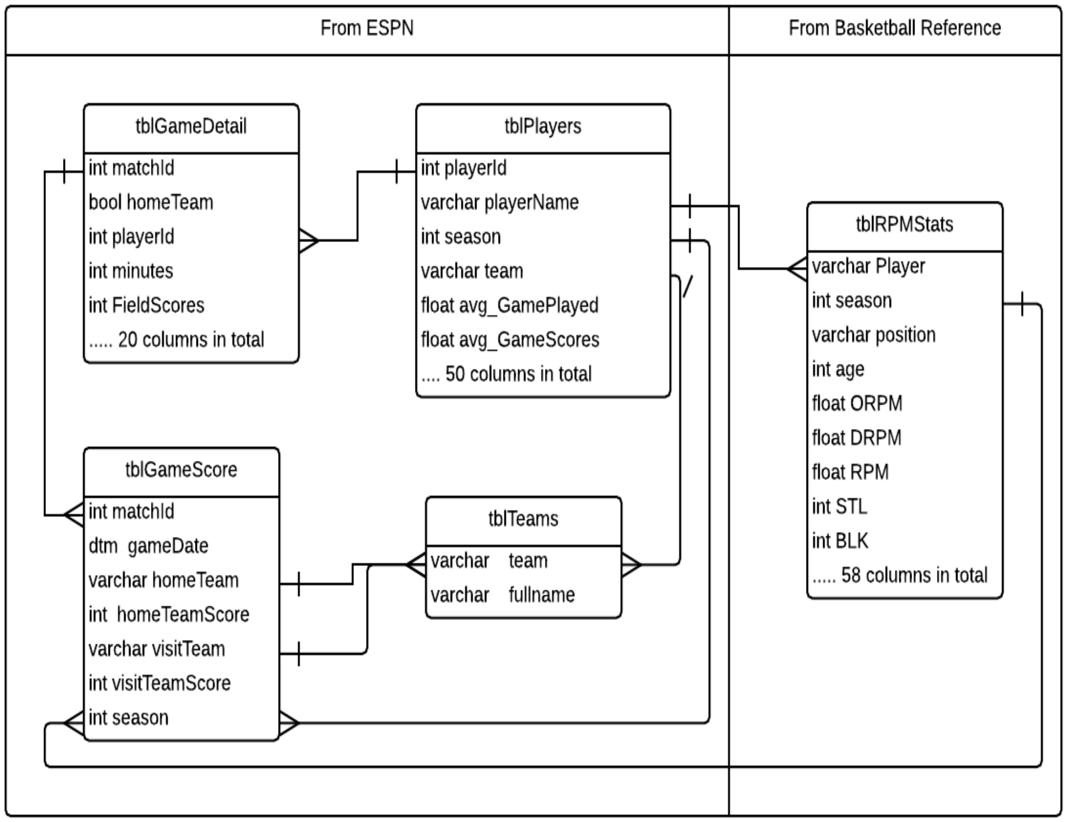

Figure 1: Entity - Relationship graph for SQLite database. The database follows the third normal form to ensure there is no redundancy. Two main data sources of the dataset are ESPN and Basketball References websites. Lines between tables indicate the primary key - foreign key relationship between dataset entities.

3.4 Merging Sources into One Database

One of the major obstacles in our project has been combining these three data sources into one single database. The ESPN dataset had both match and playerID's for each game, so merging the game statistics with the ESPN player database wasn't very difficult. However, the basketball reference and RAPM dataset didn't have these identifiers, so it was more challenging to put them together. We ended up using the player's name, team, and season to join both of these datasets together with ESPN. Some common problems we had were inconsistent spelling of names in different datasets, inconsistent team names, teams have changed cities, etc.. In the end, we were able to sync most of the idiosyncracies between the three datasets, which is important because we will have more interesting features than just box score statistics. That being said, we don't doubt that there will still be further cleaning to do, and we will deal with these situations as they arise.

4 Experiments

4.1 Training and test datasets

We have put a substantial amount of time into constructing our training and test datasets. The dataset consists of both RAPM data and ESPN's player basic statistics(i.e, average points or rebounds of a season). To create the dataset, we went through each match to find the players on each team, and merged these players statistics from the previous season with the match results in the current season. We then sum up the players statistics to form home team and away team statistics. To form the RAPM features, we used each players average minutes pergame from the season before as weights to compute weighted offensive and defensive RAPM statistic for each team. Say there's n players on each team, each playing m minutes per game with and RAPM score r in the previous season. Then the formula for the players weights and team offensive RAPM is:

$$w_i = \frac{m_i}{\frac{1}{n}\sum_{i=1}^{n} m_i}$$

$$\text{Team Offensive RAPM} = \frac{1}{n}\sum_{i=1}^{n} w_i \times r_i$$

ORPM_home	DRPM_home	ORPM_away	DRPM_away	homeWin
-0.28	0.89	0.65	0.18	1
-0.28	0.89	1.15	1.05	1
-0.29	0.99	-1.44	0.12	1
0.03	-0.66	0.04	1.09	1
0.28	1.26	-0.81	-0.20	0
-0.75	-0.46	0.51	1.02	1

Table 1: A look at what our training and test datasets look like. The first four columns are features and the 5th column (indicating whether or not the home team won the game) is our label.

Before training the model, we would like to get a general feeling about how are the features related to the game results In figure 2, the left is a plot showing the correlation between home team RPM and visit team RPM in our dataset. This shows that our data is not linearly separable for these two features, which is common among this dataset. However, the green dots(corresponding to home team loss) are more likely to accumulate on the lower right and the red ones are on the top left corner of the plot. A classification boundary like the diagonal line in the plots shows that it will classified a majority of the data set and making fewer mistakes. Also, we do see that there is some signal between wins and losses for these two features, as a higher score indicated a higher chance of winning the game. On the right, we look at team RAPM in wins compared with losses. As we can see, RAPM is clearly higher in wins than losses, so their is clear signals coming from this feature, as expected.

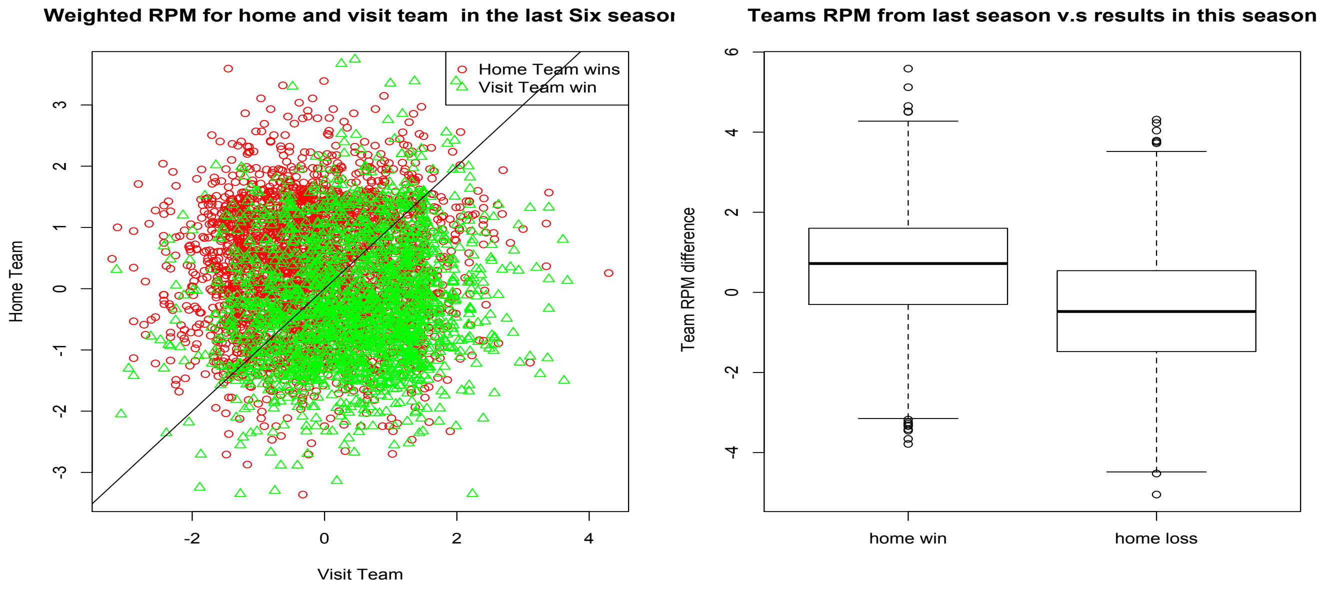

Figure 2: Left: visit team RPM against home team RPM with green dots meaning visit team wins and red dot meaning home team win. Right: A box plot of RPM. Y-axis represent the the RPM difference between home team and visit team.

4.2 Training the model and the test error

The data set is now ready. We have in total 6 season data. We first would like to try different model on our data set. For each model we try, we will split our data set into training part and testing part. For example if we try to predict the results in season 2010, we will train the model using season 2008 and 2009 data.We then use our trained model to predict the results of season 2010.

Figure 3 demonstrates the test error of different model. It turns out that the Naive Bays and linear model have the best testing accuracy. Note that based on the observation we make in the last section, we believe that simple model (with linear classification boundary) will have good performance. The test accuracy plot agrees with our intuition.

Notice so far we have 44 features in each of our model. Thus we believe that feature selection will help us to improve the prediction accuracy. Here we will illustrate our feature selection using linear regression. For other models the idea would be similar. For linear model, we try lasso and AIC to accomplish the feature selection. For lasso, we choose λ , the degrees of freedom, to minimize the CV, and then fit the model using that λ. For AIC, we use set our minimal and full model, and then perform stepwise AIC feature selection.

	Full model (linear)	Lasso	AIC
2009	0.6544715	0.6658537	0.6617886
2010	0.6739837	0.6853659	0.6788618
2011	0.6676768	0.6626263	0.6717172
2012	0.6786005	0.6712775	0.6818552
2013	0.6601626	0.6634146	0.6601626

Table 2: A look at how feature selection change the prediction accuracy.

It turns out that AIC in general helps to find a more predictive model. Note that the absolute increment is not significant in this case. However, as discussed later in the paper, the best prediction accuracy in the market is around 70%, while if one always predict home team win, the accuracy is around 60%. Take this into consideration, the improvement from AIC is not negligible.

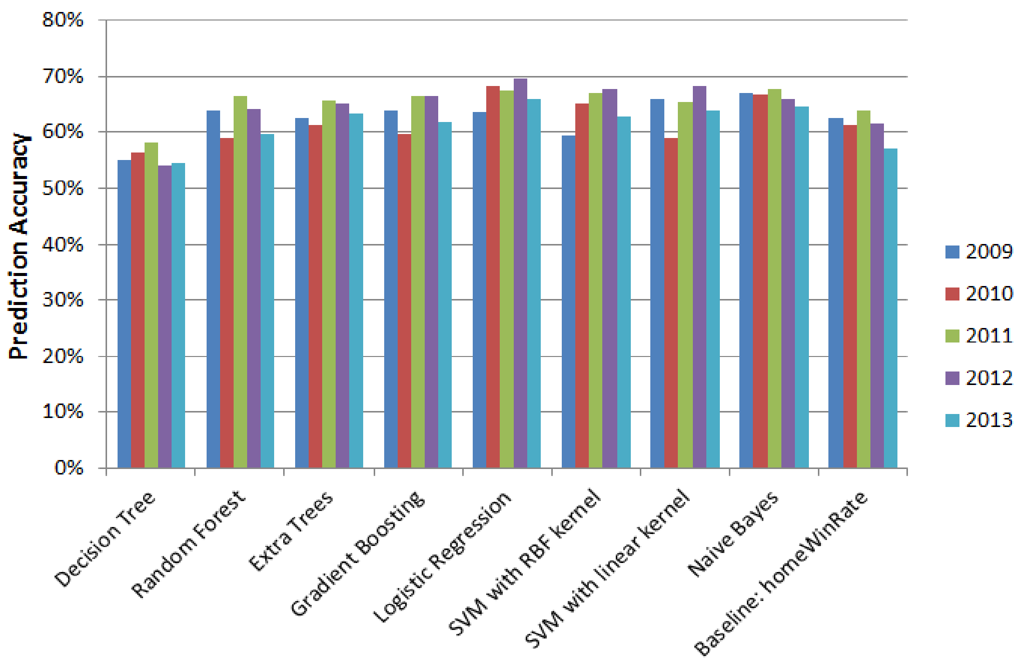

Figure 3: This plot shows the test error for our different algorithms in all of our years. From this, we see that Naive Bayes and Logistic Regression gave us the best overall test error. For each testing year, we used all the previous years as training data.

5 Simulation

In order for the various classifier's to predict which team would win each game, they generate probabilities for each team, and predict the maximum. Another natural way to utilize these probabilities is to use them to simulation the entire season. Specifically, by generating Uniform(0,1) random variables, and scaling the probabilities to sum to one (when necesary), we were able to stochastically determine the outcome of each game, and simulation each season to see the distribution predicted win totals for each team. Figure 4 shows the win distributions for each team, using logistic regression for the 2013-14 season. Out of the two best classifiers (Naive Bayes and Logistic Regression), logistic regression performed much better in the two seasons we simulation, using both the Root Mean Squared Error (RMSE) and Median Absolute Deviation (MAE) loss functions. This is because the Naive Bayes prediction outcomes were much more extreme than logistic regression. For example, the standard deviations of the number of wins predicted for each team using Naive Bayes were 16.12 and 17.07, compared with 9.66 and 9.68 for logistic regression. Due to computing and time restrictions, we weren't able to run as many simulations as originally intended. However, we did run 1000 2013 simulations various times, and the win distribution was nearly identical, meaning that 1000 is a safe range for convergence.

Predicting the number of wins for each team preceding each season is a much more popular

6 Conclusion

References

[1] Matthew Beckler, Hongfei Wang, Michael Papamichael *NBA Oracle* 2009.

[2] Dragan Miljkovic, Ljubia Gajic, Aleksandar Kovacevic, Zora Konjovic *The Use of Data Mining for Basketball Matches Outcomes Prediction* 2010: SISY 2010

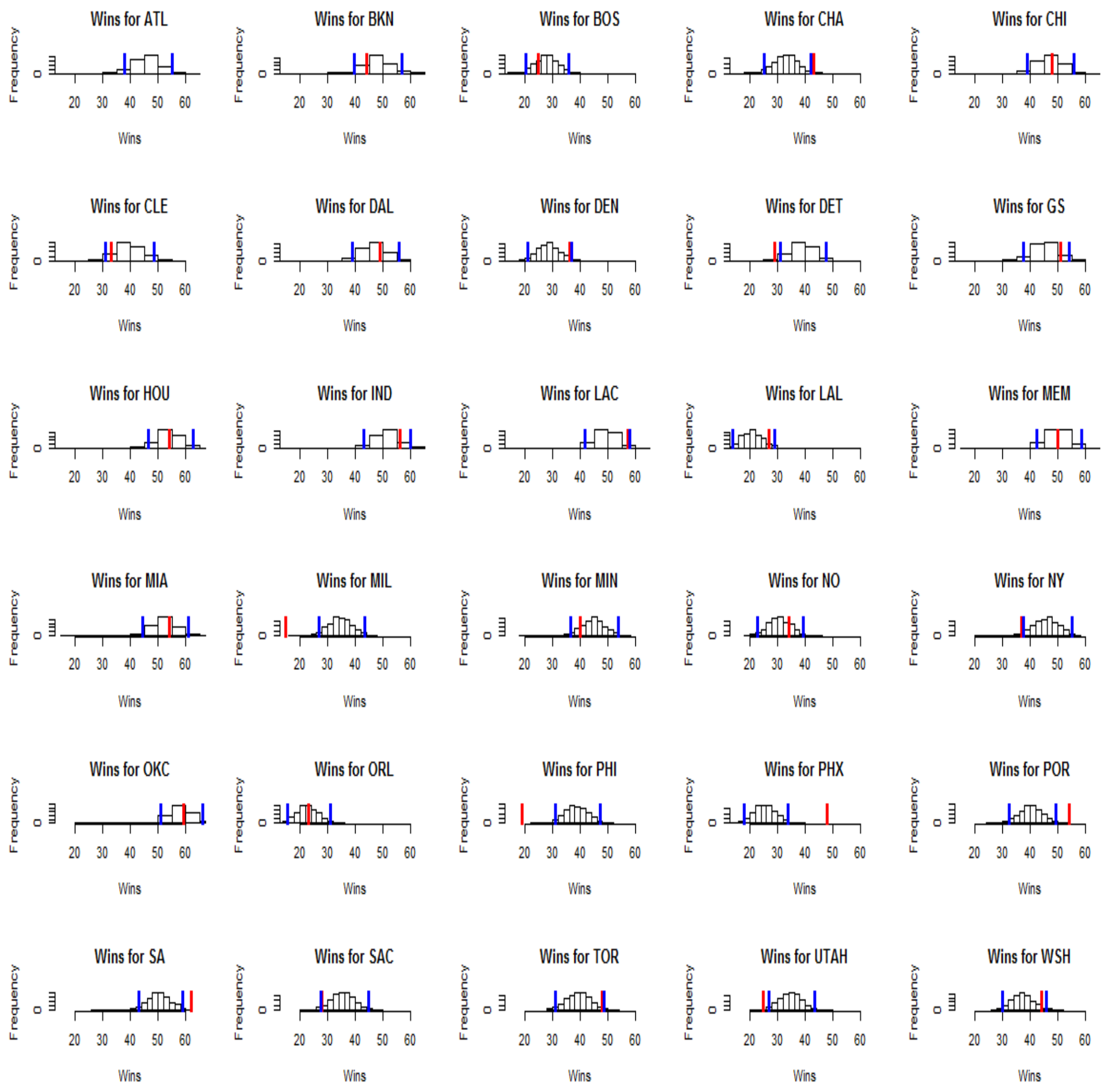

Figure 4: Results from simulating the 2013 NBA season 1000 times, using probabilities from our best Logistic regression model. The blue lines represent our confidence intervals whereas the Red lines represent the actual number of wins for each team. Our simulations trapped the true number of wins in 70 % of our intervals.

[3] Paul Fearnhead, Benjamin M. Taylor *On Estimating the Ability of NBA Players*. 2010: http://arxiv.org/pdf/1008.0705.pdf.

[4] Steve Illardi. *The next big thing: real plus minus*. 2014. ESPN.com

[5] Jeremias Engleman. http://stats-for-the-nba.appspot.com/

[6] Basketball Reference. http://www.basketball-reference.com/

[7] ESPN. http://espn.go.com/nba/.

[8] Repository for Game Simulation. https://github.com/leerichardson/game_simulation.

[9] Weak Side Awareness Blog. http://weaksideawareness.wordpress.com/2013/04/23/checking-2012-13-nba-win-predictions-projections/

[10] Sam Hinkie and the Analytics Revolution in Basketball. NILKANTH PATEL http://www.newyorker.com/news/sporting-scene/sam-hinkie-and-the-analytics-revolution-in-basketball