NBA Predictions

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

December 12, 2014

Abstract

Last year, the defending champion Miami Heat lost consecutive games against the lowly Detroit Pistons and Milwaukee Bucks. While this two game stretch was certainly unlikely, it exemplifies the inherent randomness present in NBA basketball games. In this paper, we examine how well we can predict the outcomes of individual games using various machine learning algorithms. Specifically, we see that we can accurately predict around 70% of individual games, and that simple, interpretable classifiers such as Logistic Regression and Naive Bayes work the best. We then use these predictions to simulate full NBA seasons, which land us in the top 20 of all publicly available NBA projection systems [9]. Finally, we see that the recently developed Regularized Adjusted Plus Minus (RAPM) statistic is a better predictor than the more commonly used box score statistics.

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, starting with Dean Oliver's Basketball on Paper [12], which was for a long time the cornerstone of basketball statistics. Recently, new data has become publicly available, most notably play by play data which provides a record of each event, when they happened, and which were on the floor when they happened. This is in contrast to the more commonly known Box Score statistics, which are useful, but have less information than play by play data. In fact, every box score statistic can be inferred from play by play data. Within the last few seasons, camera's were installed in every NBA arena [11], which track the location of the ball and each player 10 times per second. However, this camera data is proprietary, and it will be awhile before it's released and people are able to make sense of it.

In this paper, we harness all publicly available data sources to predict the outcomes of NBA basketball games. In section two, we will look at previous attempts at the problem and provide a more detailed description of RAPM. Next, we will describe how we compiled our database, and give information on how one could access it. In section four, we describe our efforts in predicting games using various algorithms, features, and model selection techniques. Then we discuss how we used our game predictions to simulate full seasons, and compare our results with other projection systems. We conclude with a discussion of what we learned and ways we can improve our predictions.

2 Related Works

We looked into the literature to see if anyone had worked on the same or similar problems. From this, we found a few papers, specifically [1] and [2], which also attempted to predict the outcomes of NBA games. These papers used linear and logistic regression, naive Bayes, neural networks, and support vector machine's as their classifiers, and used mainly box score statistics as features. They also used the same loss function (zero one loss) that we are proposing, which gave us a target accuracy 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 season/algorithms combinations had error rates from the mid 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 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 will 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. The final dataset consists of the score margin of each miniature game, and indicator variables for each player involved (+1 for offense, -1 for defense). Then a regression is ran (people are now using ridge regression, which is why the R in RAPM stands for Regularization), and the coefficients represent each players individual contribution per 100 possessions on offense and defense. 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

We encountered several challenges in 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 unified format. Different websites may treat player names and season labels differently. For example Roger Mason Jr. and Roger Mason are in fact same person from different websites, while uta and utah are different abbreviations for the Utah Jazz.

- Data needs to be stored in a proper manner so that our future experiments can be implemented based on an 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 combined. Finally we stored all data into a SQLite database, shown in 1 We are making our dataset publically available 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 this type of NBA statistics in the coming years.

3.2 Data Sources

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. 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. They also use different, more advanced statistics, such as the widely used Player Efficiency Rating (PER), compared with just box score statistics provided by ESPN.

The final source we used was from a website put together by Jeremias Engleman [5]. This site has RAPM statistics described above, 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.
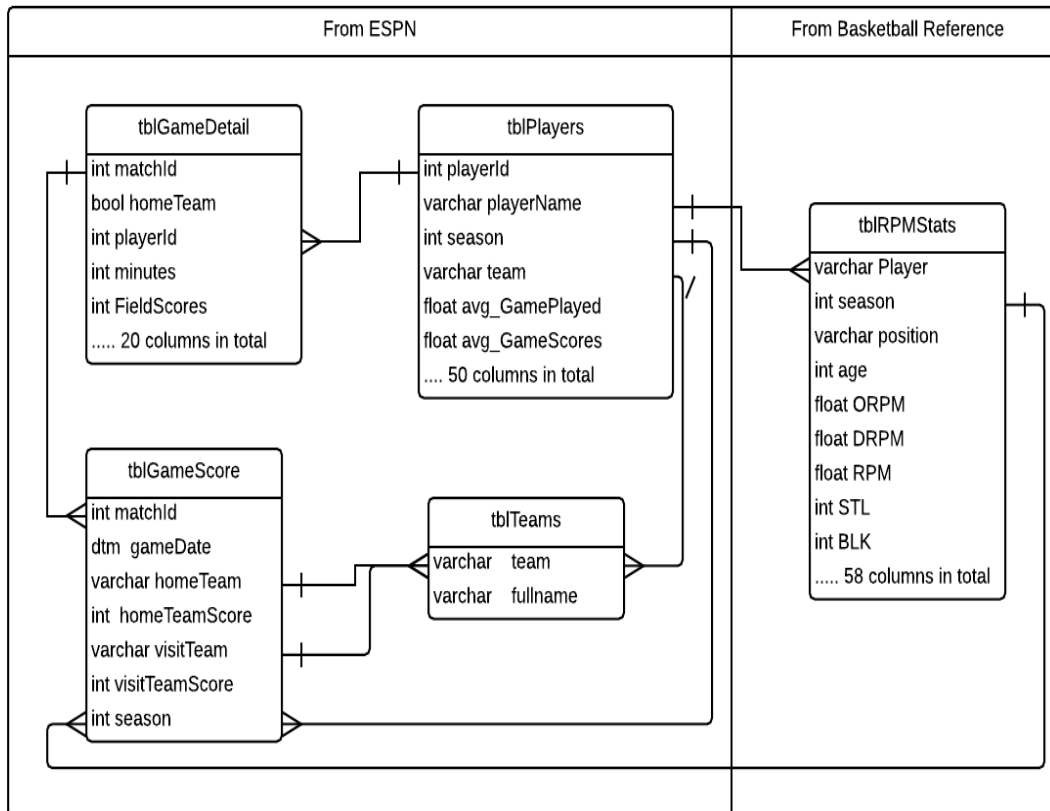
Figure 1: Entity - Relationship graph for SQLite database. The database follows the third normal form to ensure there is no redundancy. Two main data sources of the dataset are ESPN and Basketball References websites. Lines between tables indicate the primary key - foreign key relationship between dataset entities.

3.3 Web Crawlers

To obtain all of these datasets, we used web-crawlers 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.

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 idiosyncrasies between the three datasets, which is important because it gives more interesting features than just box score statistics. That being said, we don't doubt that there's still be further cleaning to be done, as it's hard to catch all of the idiosyncrasies in these three separate datasets.

4 Experiments

4.1 Training and test datasets

We devoted a substantial amount of time to construct features that can be used to train the model. The features are constructed using all three data sources (recall that the ESPN data provide basic statistics such as the average number of points per game or the average number of rebounds per game for a given season). To create the features, we considered each match and listed the players on each team, then we merged the players' statistics from the previous season with the results in the matches of the current season. We then add up the players' statistics to form statistics for the home team and away teams. In the computation of the offensive and defensive RAPM statistics of each team, the players' average minutes per game from the previous season were used as weights. To illustrate, suppose that there are n players in each team, with the i-th player playing m_i minutes per game and having RAPM score r_i in the previous season. Then, the weight for the i-th player and the team offensive RAPM are determined as:

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

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

Home Offensive RAPM	Home Defensive RAPM	Away Offensive RAPM	Away Defensive RAPM	Home Team Wins
-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 tried to understand how the features that we constructed related to the game results with exploratory data analysis. In Figure 2, the left plot has the home team RAPM on the x axis and the away team RAPM on the y axis. The plot suggests that the game results are not linearly separable using RAPM. However, the

green dots (corresponding to the losses for the home team) are close to a Gaussian distribution centered above the diagonal line of the plot, while the red dots are centered below the diagonal line. If one takes the diagonal line as the decision boundary, the majority of the data points are correctly classified. The right plot displays the teams' RAPM in wins compared with the teams' RAPM in losses. The horizontal line at the middle of each box corresponds to the sample mean. The top and bottom ends of the boxes are the empirical first and third quartiles, respectively. From the plot, one observes that RAPM is higher in wins than losses, suggesting that RAPM is a relevant feature.
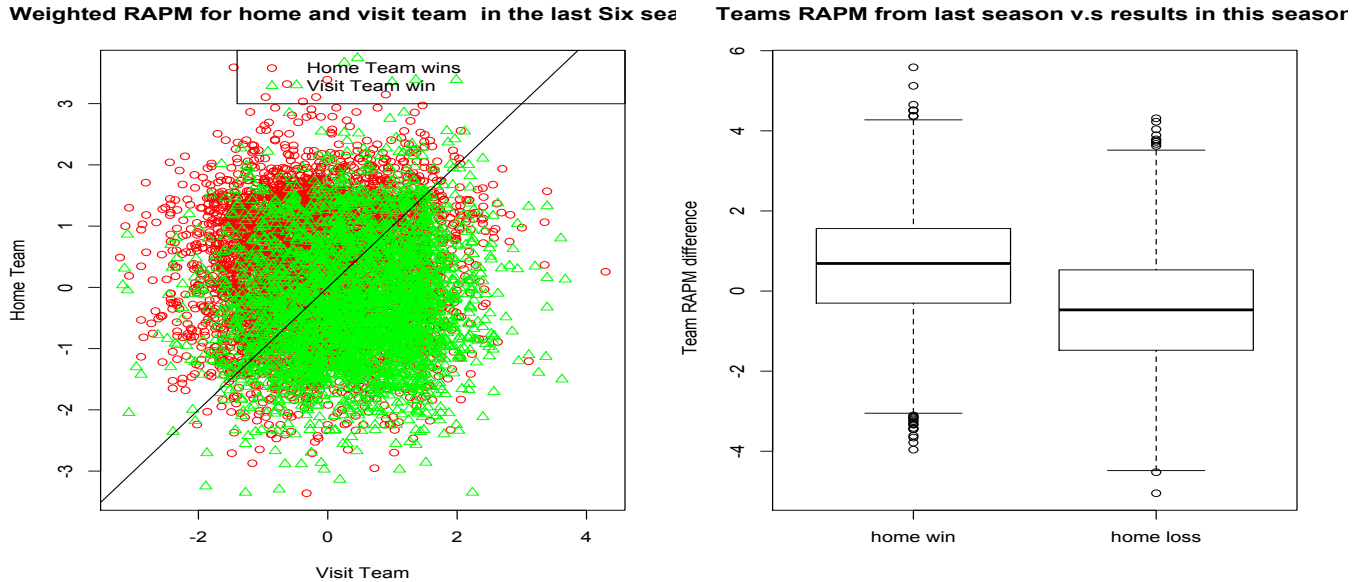

Figure 2: Left: Visiting team overall weighted RAPM against the same statistic for the home team; the green dots indicate that the vistiting team wins while the red dots indicate home team wins. Right: box plot of RAPM; the y axis represents the difference in RAPM between the home team and the visit team.

4.2 Algorithms and feature selection

We tried different algorithms based on the feature matrix we constructed to predict the game results. For each model, we split our dataset into a training set and a test set. We trained our model using available data from the previous seasons and we used the model to predict the game results in the 2009-2014 seasons. For example, to predict the game results for the 2010 season, we trained the model using data from seasons 2008 and 2009. Figure **??**dif

4.3 A discussion of the algorithms

To predict the game results, we used algorithms such as linear regression, logistic regression, naive Bayes, support vector machines, and decision trees.

1. **Linear Regression.** In order to fit the linear regression model, the game results are denoted as 0 and 1 (with 1's denoting home team wins). The fitted model is described in terms of a vector of estimated coefficients (one for each feature) which we denote $\hat{\beta}$. To predict new games results using this model, one evaluates $\hat{\beta}^T X_{\text{test}}$, where X_{test} corresponds to the vector of features for the new games. We predict that the home team wins if $\hat{\beta}^T X_{\text{test}} \geq 0.5$, and that the home team loses otherwise.

2. **Logistic regression.** Logistic regression is particularly useful when the outcome is binary. As before, we use 0 and 1 to denote wins and losses, and we predict that the home team wins a new game if the predicted value for that game is greater than or equal to 0.5. Predicted values from a logistic regression are estimates of the probability that the home team wins. Thus, predictions from the logistic regression model are easily interpretable.
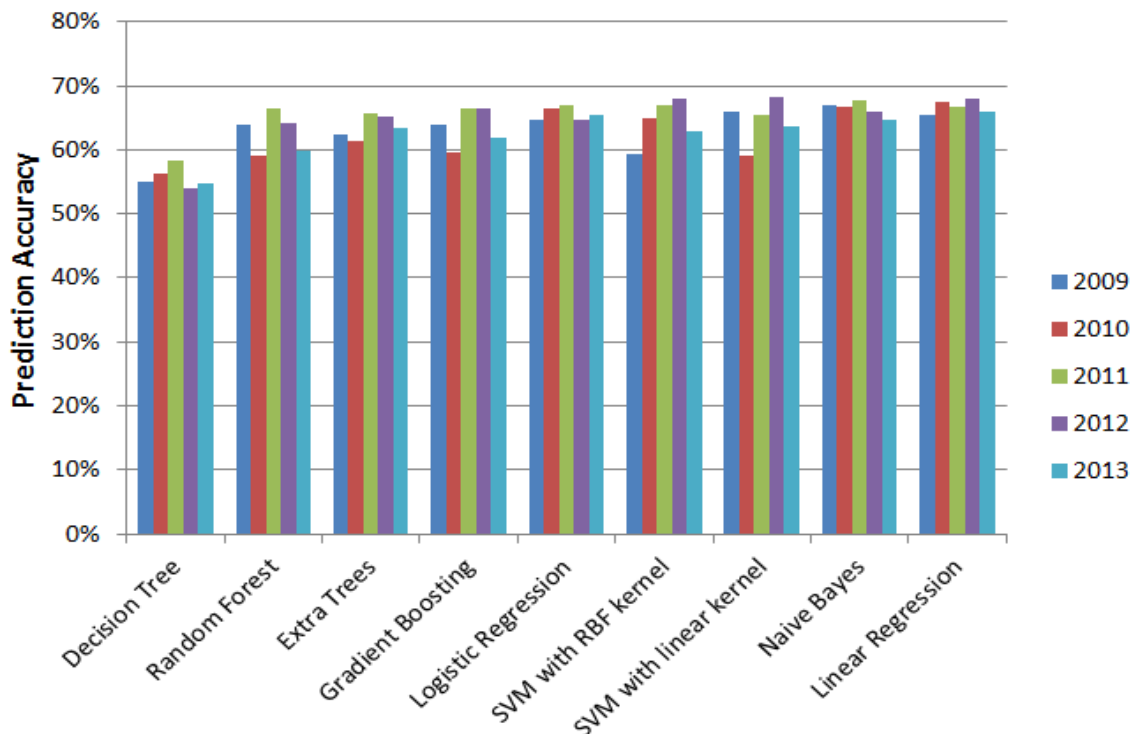
Figure 3: Test Accuracy for various different algorithms classifying the outcomes of NBA games in seasons 2009-2014.

3. **Naive Bayes.** The naive Bayes classifier is a simple probabilistic classifier that selects the hypothesis that maximizes a posterior probability. The underlining assumption for naive Bayes classifier is that all the features are conditionally independent. Although in our case the features are not independent, the naive Bayes classifier performs relatively well compared with the other models.

4. **SVM.** A Support Vector Machine is a non-probabilistic classifier. We tried different kernels, and figure 3 displays the highest model accuracy achieved by the SVM among the kernels that we considered. Note that even though different kernels generate different decision boundaries, none allows to exactly classify all of the data points. For this reason, we use soft margin to train the model.

5. **Decision tree.** A decision tree is useful when one needs a good way to split a dataset with many features. In our case, there are 44 features overall. In order to avoid overfitting, we choose the subtree that minimized the regularized training error, defined as the sum of the training error and $\lambda|T|$. Here, $|T|$ denotes the size of the tree while λ is a penalty parameter which is selected using cross-validation.

Figure 3 displays the test error of different algorithms, from seasons 2009-2013. The linear model appears to have the best accuracy, which is the same result obtained by [1]. Note that, on the basis of the discussion above, the linear decision boundary was expected to perform well. The test accuracy plot confirms our intuition.

4.4   Feature selection

There are 44 features for each model. Feature selection is used to prevent overfitting and thus improve prediction accuracy. We discuss feature selection fin terms of the linear regression model, since it gave us the best test accuracy. For the linear regression model, we perform feature selection by using lasso regularization and a stepwise AIC procedure. In the lasso regression, the effective degrees of freedom λ are chosen to minimize the generalized cross validation error. In the stepwise AIC procedure, we set the minimum model to be the model including only the

6

intercept term (which represents the probability that the home team wins) and the maximum model to be the model including all of the 44 features. At each step of the stepwise AIC procedure, a feature is included or dropped from the current model in such a way to give the highest decrease in the AIC score until no further changes in the AIC score occur.

	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 improves the prediction accuracy.

According to table 2, stepwise AIC found a model with higher accuracy. The absolute difference in accuracy between the original model model and the selected model are not significant. However, the best prediction accuracy is known to be around 70%, and when one always predicts home team win, the accuracy is around 60%. From this, we note that improvement in predictive accuracy from feaure selection is not negligible.

We also note that both the lasso and the stepwise AIC procedure identify the overall team weighted RAPM as the most important predictive feature. This is confirmed by the fact that the stepwise AIC procedure starting with the null model selects home and away RAPM as the first two features that should be included in the model. Moreover, the coefficient of the Weighted RAPM didn't shrink to 0 after the lasso, whereas all the others did, and this is another indication that these feature is more relevant than the others.

5   Simulation

In order for the various classifier's to predict which team would win each game, they generate probabilities for each team (class), 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, we were able to stochastically determine the outcome of each game, and simulation each game in a season. Next, we simulated the season 1000 times to obtain the distribution of 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 classifiers with easy to use probabilities (Naive Bayes and Logistic Regression), logistic regression performed much better in these simulations, using both the Root Mean Squared Error (RMSE) and Median Absolute Deviation (MAE) loss functions. This is because the game probabilities generated by Naive Bayes were much more extreme than logistic regression. While this doesn't effect classification as much since we're only predicting the maximum, the exact probabilities matter in a more complicated dstribution, such as simulating a complete season. To see how sensitive the simulated seasons are to these probabilities, consider the standard deviations of the number of predicted wins for each team using Naive Bayes and Logistic Regression. For Naive Bayes, they were 16.12 and 17.07, compared with 9.66 and 9.68 for logistic regression. This shows that the win totals predicted by Naive Bayes were much more extreme than logistic regression.

Forecasting how many wins each time will get during a seasons turns out to be a much more popular task than computing test errors, so we were able to compare our projections to lots of other systems using our RMSE loss function (which is commonly used by in the basketball analytics community [9]). Our RMSE for 2012 was 7.54 and 9.7278 in 2013. This would be in the top 20 for the past two seasons. Had we used Naive Bayes as opposed to Logistic regression, our system would have consistently performed as one of the worst.
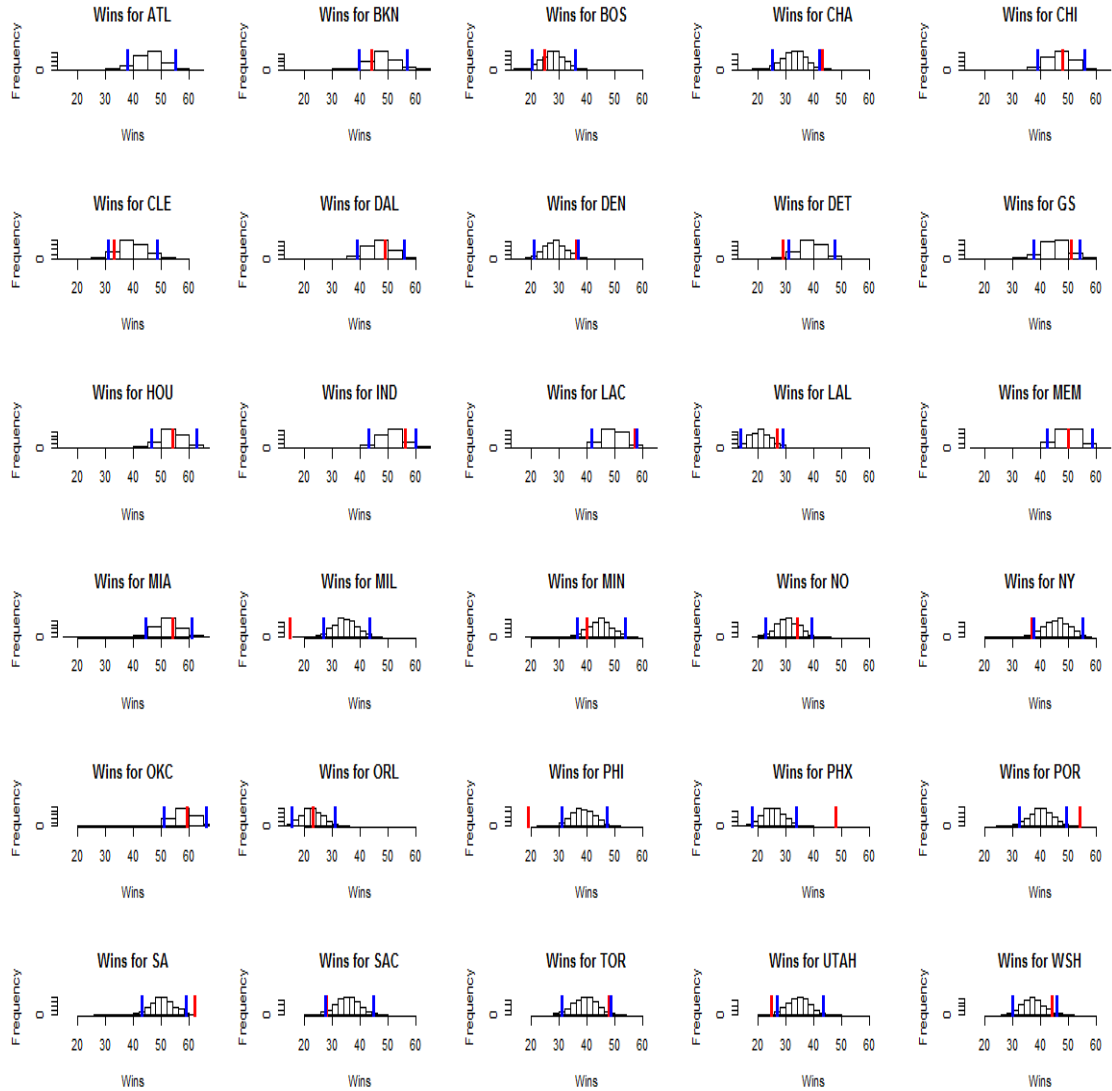
Figure 4: Distribution of wins for each team from simulating the 2013 NBA season 1000 times, using probabilities from our best Logistic regression model. The blue lines represent our confidence intervals whereas the Red lines represent the actual number of wins for each team. Our simulations trapped the true number of wins in 70 % of our intervals.

6 Conclusion

In the end, it seems that sticking with a simple classification rule using the best features yields the best accuracy in forecasting NBA games. It's clear that the two factors have the largest impact on each game are home court advantage and the quality of each team. Imagine that there's some perfect measure of the quality of each team, and that two teams are identical. In this situation, you should always pick the home team to win. The question then becomes, how much better does the away team have to be in order to choose them to win? Figuring out where to draw this line seems like the way to get the best performance out of classification algorithms. Obviously, since there's no way to know the exact quality of each team, this is estimated from data and is not an exact science.

In terms of simulating seasons, it seems that more conservative estimates, such as those yielded by logistic regression, work better than more extreme predictions. For this endeavor, there were more public predictions available to compare our results with, and we see that while our system performs well, it did not approach some of the best publicly available systems. For instance, John Hollinger, who was recently hired by the Memphis Grizzlies, had the best projection system in 2012. He has been doing this for around 5 years, and written various books on the subject, so it makes sense his system would have better performance than ours, just based on experience. As we discuss below, there's many nuances that could be built into a projection system to make it more marginally accurate, and since we only spent a semester on this project, there wasn't enough time to incorporate everything we hoped to include.

One interesting finding that came out of our report was how much lower our test error was in the final season, 2013. This is counter to what one would expect in a typical machine learning situation, since we had the most amount of training data for this season. However, we think the reason for this is that there was just more variability in the 2013 compared to other seasons. This shows that no matter what algorithm/features you use, upsets happen in the NBA, and some seasons have more unexpected results than others. Other systems also showed much lower accuracy in 2013 compared with 2012 [9].

Another interesting result was how much predictive the RAPM statistic was than box score statistics. We think the main reason for this is simply that it simply contains more information. RAPM uses possession level events to construct offensive and defensive ratings for each player, and all box score statistics can be computed with the same possession level data that calculates RAPM. What does this mean for the future of statistics in the NBA? We think it indicates that soon enough, statistics like RAPM might soon become less state of the art, as even higher resultion camera data is now being collected by a company called STATS LLC [11]. The data from these cameras tracks the location of each player 10 times each second, and is sure to produce some novel insights into the game of basketball.

There are many ways to improve upon the results displayed in this paper. One of the main flaws in our system is that we only used individual player data from the previous season in order to predict the current season. This means that if a player misses the previous season due to injury (IE: Derick Rose, Rajon Rondo, etc...), then a significant aspect of a team may be left out of our systems. We are also treating rookies as league average players, which is a very favorable assumption, since most rookies are worse then the league average in their first season. A couple ways we could combat this is by using projected season statistics as features to predict the current season. Some of the most accurate RAPM based systems used in 2013 used this strategy [9]. That way, we could account for things like player age, injury history, and expected changes due to team composition. We could also consider not just the previous season, but a weighted average of the past 3-5 seasons to get an estimate for the quality of each player.

We also wanted to include data from the current season into our projections. To do this, we could divide each season into K different chunks, and use the teams winning percentage in all chunks of the season before each game to predict game outcomes. As seen in figure 4, some of the estimates which represented how good we thought each team was before the season were very off. Incorporating current season data into the predictions would allow us to save ourselves when it becomes clear a team is much better than expected. We can update our valuation of how good each team is as we get more information on them throughout the season.

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

[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

[11] Zach Lowe. http://grantland.com/features/the-toronto-raptors-sportvu-cameras-nba-analytical-revolution/

[12] Basketball on Paper: Rules and Tools for Performance Analysis. Dean Oliver