Simulating the NBA playoffs using Logistic Regression and Random Forests

A Senior Project
Presented to
The Faculty of the Statistics Department
California Polytechnic University, San Luis Obispo

In Partial Fulfillment
Of the Requirements for the Degree
Bachelor of Science

By
Boudewijn Aasman

Senior Project Advisor:
Gail Potter, Ph.D

June 2015

Table of Contents

This project is split into nine different sections:

Coursera Data Mining Courses

The senior project I partook in consists of two separate parts. One part includes going through the data science specialization courses offered by Johns Hopkins through Coursera. The second part consists of a small project of my own choosing. The main premise of this senior project is to apply at least three concepts learned from the data science specialization courses to the project I choose to do.

Before describing what this individual project consists of, and which concepts I choose to apply in my project, I will present a small summary on what the Coursera courses consisted of. All nine classes were available freely on Coursera, a website devoted to offering a wide array of classes. Each course is designed to last exactly one month, and often has a quiz at the end of each week to test your knowledge of that week's subject. One of the most useful aspects of these classes is the requirement of a small project. These projects often involve real life data sets, and require you to apply the knowledge learned to these data sets. Upon completion of this project, the student is forced to peer review the work of four other students. This may have been the most useful aspect of each of these classes, as the student is exposed to completely different mindsets and programming techniques, which has the capability of enhancing our own. The following list displays the topic names of all nine courses, and a small description on what they cover.

- The Data Science Toolbox: Identifying data science problems, creating a Github account, and pushing files to repositories on Github.
- R Programming: General overview of R, it's functions, loops, and the use of simulation.
- Getting and Cleaning Data: Obtaining data from numerous different sources, and turning messy data into tidy data.
- Exploratory Data Analysis: Create visual representations with the base, lattice, ggplot2 packages in R, construct summary analysis, and explore multivariate statistical techniques.
- Reproducible Research: Learn how to use R Markdown, knitr, and how to make code reproducible.
- Statistical Inference: General overview of assumptions, tests, probability, and models.
- Regression Models: How to fit models, interpret coefficients, and investigate assumptions. Also covers other types of regression models, such as Poisson and logistic.
- Practical Machine Learning: How to apply several different machine learning tools and test predictions against the actual results.
- Developing Data Products: How to communicate statistical products, use various interactive graphic-making tools, and create R packages.

The goal of this project is to apply at least three concepts learned to a data set of my own choosing. Of course, many of the skills learned in these classes

carry over to all facets of statistics, but the following concepts include those that I was specifically interested in and wanted to apply to my project.

Feature Analysis (from Applied Machine Learning)
Feature analysis can be incredibly important in building the appropriate model. In the applied machine-learning course on Coursera, a section is dedicated to this concept. Using the caret package in R, we can use several different graphs to assess what variables seem to be associated with the response, get specific correlations for all predictors with the response, and determine the correlations between predictors. Similarly, the course discusses functions such as varImp(), which helps us determine what predictors contribute most to the error rate of a model.

Random Forests (from Applied Machine Learning)
Along with boosting, random forests are one of the more popular models used in machine learning related problems today. Random forests are an extension of bagging on classification and regression trees. In the class, we learned several ideas related to understanding and creating random forests. It covered the processing steps, such as taking bootstrap samples (random samples with replacement) from training data, growing trees, and ultimately combining those trees to create the final random forest model.

Github (from Data Scientist Toolbox)
One of the main concepts the courses made use of is Github. Github is a hosting service that allows for version control, and allows for collaboration of many different parties on one project. In order to submit many of the projects completed in the Coursera courses, they forced you to submit it to the proper repository, and often perform specific tasks, such as forking and cloning other repositories.

Background of project

Since the recent explosion of the use of statistics in the real world, people have been attempting to apply statistical techniques to nearly all-possible quantifiable events. Ranging from healthcare to traffic patterns, many statisticians have been developing models to predict and analyze the likelihood of specific patterns and outcomes. One of the most intriguing real life events to model is professional sport. Professional sports are a multi-billon dollar industry that has an incredible influence on people and communities all around the world. Therefore, the need to predict outcomes in the sports world comes as no surprise.

This project is divided into two separate parts. The first part includes the creation of a logistic regression model and a random forest, where both attempt to classify the probability of a home team victory in the NBA playoffs. Similarly, they take as input the statistics for any two teams, and will output the probability that the team playing at home is the victor. The second part of this project uses simulation to estimate the probability a certain team will win a best of 7 series in the NBA playoffs. To do this, an algorithm loops through a seven game series, where each individual game is assigned a probability of the home team winning based on the model created in the first stage of this project. Whoever gets to four games first is projected to win the series. The estimated probability of the home team winning is the number of simulated wins divided by the total number of simulations.

Data structure and collection

One of the main challenges of this project was coming up with the appropriate data frame that allows us to create analytical and predictive models. It is an inherent feature of sports that makes the performance of any team dependent on the opposing team. This presents a key problem. Let A be the event that team A wins, and let B be the event that team B wins. Then

$$1 - pr(A) \neq pr(B)$$

To explain this in context, take a look at the following two logistic regression models based on the number of field goals of any two teams:

$$Prob(A) = expit(\beta_0+\beta_1 FG + \beta_2 opponentFG)$$

$$Prob(B) = expit(\beta_0+\beta_1 opponentFG + \beta_2 FG)$$

To compute the following probability, we assign the following variables:

- $\beta_0 = 0$
- $\beta_1 = 1$
- $\beta_2 = 2$
- $FG = .5$
- $opponentFG = 1.5$

We have team A, who's number of field goals is represented by FG. Then there is team B, whose number of field goals is represented by opponentFG. Using the given numbers above, we can evaluate the probability of event A, which is expit(0 + (1*.5)+(2*1.5)) = .9706. This implies that the probability team B wins must be 1 - .9706, or .0294. However, evaluating the above model for Prob(B) returns the following value, expit(0 + (1*1.5)+(2*.5)) = .9241. Since

$$1 - .9706 \neq .9241,$$

we violate one of the major assumption of probability.

In order to account for this violation, rather than having each observation represent the statistics for one team, we subtract the season statistics of the away team from the home team. This accounts for the dependency of the other team, and allows us to model the probability the home team wins a game in the playoffs. So, rather than creating a model that estimates the probability any random team wins, it has become a generalized model that merely looks at the chance of a home team victory.

Since the advent of the Internet, a vast amount of data has been collected and exposed to the public. One of the most comprehensive databases of information on basketball is basketball-reference.com. This website contains

team and individual player statistics for any team since 1946, and has provided me with all the necessary data. Since the game of basketball is constantly changing, I only used playoffs data from the last five years in order to stay as relevant as possible. This has resulted in a data frame that includes 382 different games. I collected two separate pieces of information. One of these pieces measures data regarding specific matchups in an NBA series. These variables are listed in Table 1. The other piece of the data includes measures on a teams' season statistics, which are shown in Table 2.

Variable	Description
Home Team	Which team played at home
Away Team	Which team played away
Win	Did the home team win or not (1 denotes a win)
Win Previous	Whether the home team won the previous game or not
Home Seed	Seed of the home team
Away Seed	Seed of the away team

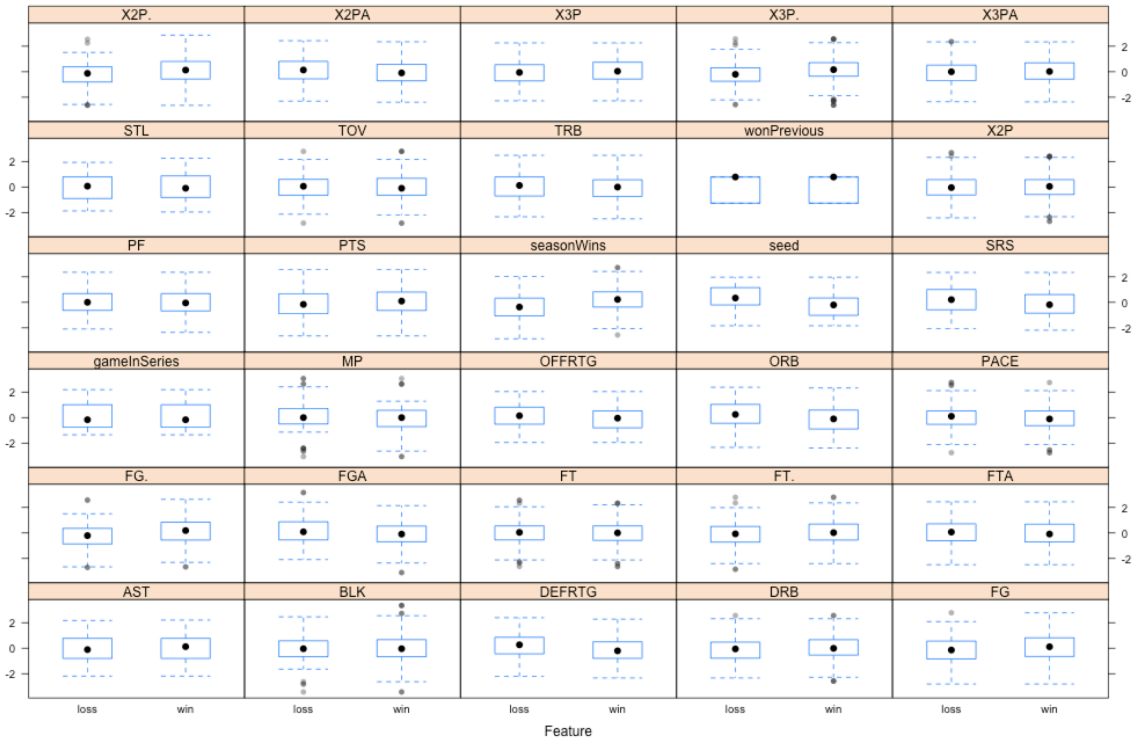
Table 1. Game data – contains information for specific games

Variable	Description
Team	What team the observation is for
FG	Field goals made per game
FG%	Field goal percentage
FGA	Field goals attempted per game
3P	Three pointers made per game
3PA	Three pointers attempted per game
3P%	Three pointer percentage
2P	Two pointers made per game
2PA	2 Pointers attempted per game
2P%	2 Pointer percentage
FT	Free throws made per game
FTA	Free throws attempted per game
FT%	Free throw percentage
ORB	Offensive rebounds per game
DRB	Defensive rebounds per game
TRB	Total rebounds per game
AST	Assists per game
STL	Steals per game
BLK	Blocks per game
TOV	Turnovers per game
PF	Personal fouls per game
PTS	Points per game
SRS	Points differential that accounts for strength of schedule
PACE	Number of possession per 48 minutes
Offrtg	Points produced per 100 possessions
Defrtg	Points allowed per 100 possessions

Table 2. Season statistics

The statistics for table 1 and 2 are then merged together using the home team as the frame of reference. Now that both data frames are merged, each observation contains the statistics for a home team, the statistics of an away team, and the data based on that specific matchup. I then took the difference of the home and away team statistics to come up with one simple and unified data frame that is always in perspective of the home team. This ultimately allows us to create a predictive model.

In order to get an overall idea of how our variables are related to the outcome of winning a single game, I created a feature plot. This will help us towards deciding what variables seem to be important, and which one's will likely not contribute much towards the classification of a win or a loss. Since we are dealing with both general numbers and percentages, I standardized each variable so that each facet of the graph is on the same scale. Remember, this plot uses the differences of predictors compared to the binary response of winning. This means that a positive value implies that the home team had a higher value in that category, and is thus superior team in that area of the game, while a negative value implies that the home team had a worse value in that category, and is thus inferior in that category of the game. Of course, it is different for statistics such as seeding, as a lower seed is in favor of the better team, and thus a negative difference implies the home team is superior in that category.



Graph 1. Feature plot of all predictors

None of the variables truly jump out at us, though we can see some slight patterns. To the naked eye, the variables that seem to lead to the largest discrepancy between winning and losing a playoff game are season wins, the seed of the playoff team, and SRS (point differential that takes into account strength of schedule). These are three variables we would certainly expect to be associated with winning, and to play an important role in the creation of the model that will be introduced later on in this paper. In order to measure this correlation quantitatively, I calculated the biserial correlation between each predictor and the outcome. This correlation is interpreted in much the same way a regular Pearson correlation coefficient is interpreted, yet accounts for the situation in which there is a dichotomous response and quantitative predictor. Again, I am not calculating the correlations based on the variable itself, like season wins, but on the difference between the home and away team values. Table 3 displays the 5 strongest associations.

Variable versus win	Correlation
Season wins	.34
Seed	-.32
SRS	-.29
Field Goal %	.25
Offensive rebounds	-.23

Table 3. Five strongest correlations

Like I stated in the background, it is my intention to use these variables to create a model that predicts the outcome of a given game. However, one thing I was interested in is how the event of winning the previous game in the series affects the probability of winning the next game. Therefore, I decided to split the data into two different subsets: One that contains only the first game of each series, and one that includes all other games, and has an extra column indicating whether or not the home team won the previous game. This will allow us to incorporate this element into our model.

Logistic Regression

Logistic regression is often the standard that is used when needing to classify a binary response. It can often be incredibly useful, as it actually gives the user an idea of how each predictor contributes to the response (log odds in the logistic regression case). By examining the contribution of each predictor to the log odds of the response, we can obtain the odds ratios, but more importantly, the probabilities of a certain outcome based on a particular input. The following equation displays the general model of logistic regression:

$$\text{logit}(p_i) = \ln\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_{1,i} + \cdots + \beta_m x_{m,i}$$

Though it's in log form, by exponentiating both sides of the equation we can convert the response to the more interpretable odds. Then, these odds can be used to convert the response to a probability, as will be shown in the next section.

Game 1 model:

This model is purely based on the subset of all game 1's from the total data. Since the total data set is quite small (382 observations), this subset happens to be especially small at 68 observations, leading to a model that might perform poorly when extrapolating to data outside of the original data set. Similarly, since it is so small to begin with, I opted to not use training and test data set. Using stepwise regression to choose a model with the lowest AIC, I found a model that was grossly overfitting the data. One of the variables correlated with a win so strongly that it always returned a probability of 1 or 0, depending on the input. Through sequentially removing the variables that caused the model to overfit, I came up with the following model:

	Coefficient	P value
Intercept	1.0010	.00139
Total Rebounds	-.171	.09931
Personal Fouls	-.4281	.01503

Table 4. Coefficient for the game 1 logistic regression model

Example: The home team has 4 less rebounds per game throughout the season, on average,, but fouls 3 times more per game in the season than the away team. This would return a log odds of:

$$1.0010 + (-4 * -.171) + (3 * .4281) = .4007$$

Log odds are not quite as useful and interpretable as odds ratios. Therefore, to convert a log odds to an odds ratio, we merely exponentiate the log odds. In our case, exp(.4007) = 1.49. This implies that a team with the above statistics is 1.49 times more likely to win a given game, when at home, than the opposing team. However, since we are interested in classification, we can convert this number to the probability of the event, which in our case is winning a playoff game at home in the NBA playoffs.

$$\frac{\exp(4.007)}{1 + \exp(4.007)} = .5589$$

This means that, for a home team that on average has 4 less rebounds per game throughout the season compared to the other team, but fouled 3 times more, we would estimate the probability of them winning to be .5589. Since this probability is greater than .5, we would classify this situation as a win, which is a concept that will become important in the final simulation.

Although both predictors are significant at the .1 alpha level, this model lacks any real practicality. Though simple, it lacks sophistication by only including two variables, and is based on such a small sample size that extrapolation will be difficult.

Game 2 Model:

Similar to the previous model, this logistic regression model is based on the subset of all game 2 through 7's. The primary difference in this subset compared to the previous is that it includes a new variable indicating whether or not the team won the previous game. This will allow us to account for who has won the previous game, which will increase the sophistication and practicality of the simulation later on. Additionally, since the sample size is significantly larger, I am able to bypass several problems that the previous model presented.

Again, I came up with the final model through a stepwise regression that is aimed at minimizing the AIC. In order to be able to test the accuracy of this model, I separated the data into a training and test data set, where the training data set contains approximately 85% of the data. This model is then created based on the training data set. The following table shows the results of that model.

	Estimate	P value
Intercept	.506	.000506
Pace	.058	.01454
Field Goal Attempts	.084	.1167
Three Point Attempts	-1.24	.0819
Three Pointers per game	8.504	.00456
Two Pointers per game	3.209	.04005
Two Pointer %	117.73	.0581
Free throw attempts	1.83	.001551
Free throw %	83.32	.000004
Defensive rebounds per game	.479	.000002
Steals per game	.573	.005636
Points per game	-2.464	.000965

Table 5. Logistic regression estimates for model 2

Compared to model 1, this model is quite a bit more useful. As we can see, the variable that provides the strongest influence on the output is two point percentage, with free throw percentage closely behind. Similar to the game 1 model, we can compute both log odds, odds, and probabilities of a home team winning in the NBA playoffs.

Since it's my goal to have an accurate classification model, I next tested how well my model performs against the test data set. Since each observation is a game I'd like to predict, I applied the logistic regression model to every observation within the test data set, and observed what percent of the time it was correct. The following table shows a confusion matrix, which compares the predicted values to the actual values.

		Actual	
		Loss	Win
Predicted	Loss	10	10
	Win	7	20

Table 6. Confusion matrix

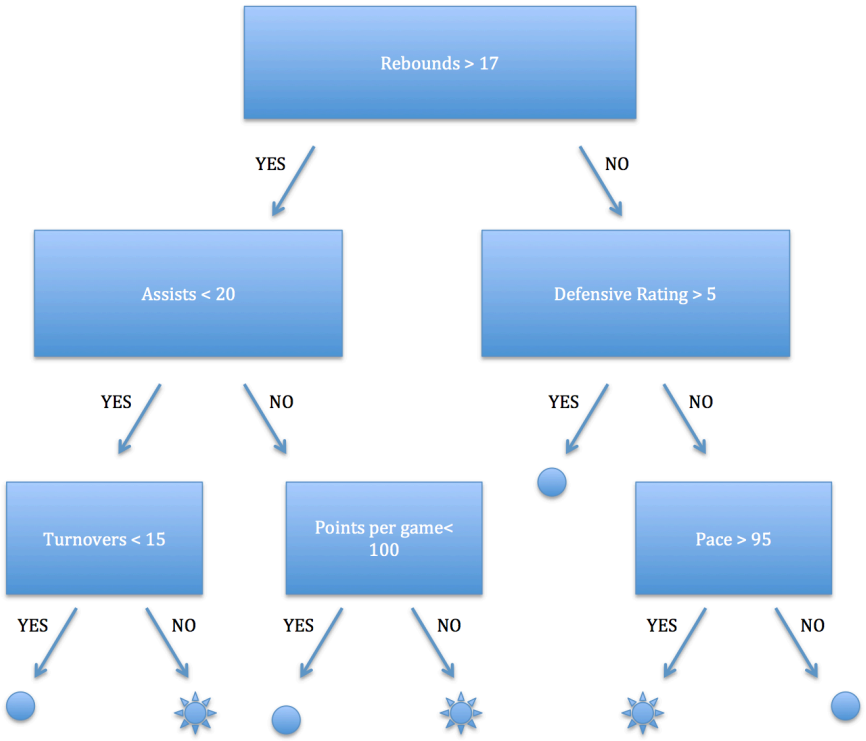
We can find the classification rate by dividing the total amount of times we correctly predicted the response by the total amount of observations.

$$\frac{10 + 20}{10 + 10 + 7 + 20} = .638$$

Thus, we are able to correctly classify a home team win, other than game 1's, and in the last 5 years, approximately 64% of the time. When accounting for the inherent variability in sports, 64% percent is not bad.

Random Forests

After learning about random forests in the practical machine-learning course on Coursera, I knew I wanted to apply and focus on building a model using this technique. Similar to logistic regression, the ultimate goal of random forests is to classify a response, which in our case is a win or a loss. Since random forests are less dependent on small sample sizes, and on the whole more sophisticated, we may be able to see improvement in our classification rates. For those initiated to random forests, they can be described as follows. At its core, a random forest is a collection of a large number of trees. Of course, that statement is not entirely useful without defining what a decision tree actually is. The following graph displays an example decision tree that I conjured up. Each box and decision is arbitrary for the sake of an example, in order to simplify the explanation.



Graph 2. Example of a decision tree

To create a decision tree, we essentially segment the data into several different pieces that allows us to classify an observation. Starting at the top with the total data set, each segment is determined by a rule. Rules are often determined by how to optimally split the segment of data left in that node into two segments that are each as homogeneous as possible. Each box is known as a node, and each node takes in an input, and creates a split based on the rule. For example, all teams with greater than 17 rebounds move to the left, and all inputs

with less than 17 rebounds a game moves to the right. This is a process that continues down, until we arrive at a “leaf”. In order to get to this leaf, we go through a unique path that ultimately leaves us with a classification, or decision. The algorithm used by R to create this decision uses a concept known as entropy. Entropy is a measure of homogeneity, as we want to divide the sample into different segments that each has similar elements. If the value entropy is high, we need to create more rules to split the data further and receive an even more homogeneous sample that ultimately allows us to create an accurate prediction. Additionally, there are several different methods to optimize the accuracy of the tree and thus minimize the error rate through processes such as pruning. At the same time, individual decision trees have certain drawbacks. One of those drawbacks is an over complexity, as a decision tree can present so much information based on the data set it becomes unpractical to use in real life situations. Random forests can address this problem.

But how exactly do decision trees relate to random forests? Well, a random “forest” is seen as an ensemble of decision trees. We specify the number of trees we want to build, grow each one, and ultimately combine them to create a strong learning model. However, not each tree is created in the same way that an individual tree is built. The random forest algorithm introduces elements of randomness to each through, which is primarily done through bootstrap sampling, which can be explained by many repeated samples from the original sample, where each sample is the same size of the original set, and each element is chosen with replacement. Similarly, to avoid an overly complex model, each small tree is limited in how deep it can go.

This process creates n weak learning trees, but is combined to create one strong learning random forest that minimizes error. There are two separate methods for combining all these created weak learners. One of the methods involves averaging the predictions of each of the weak decision trees, which often leads to a much better classification rate due to the reduced variance. However, there is a more sophisticated method, which is known as boosting. The general method of boosting consists of sequential model building, where the new model improves slightly upon the previous. It does this by giving greater weight to rules that led to correct classifications, and giving less weight to the rules within the learners that classified incorrectly. Through this sequential operation of building models, a final strong learner is formed based on all weak learners before it. These are two of the main methods of creating random forests from a large amount of decision trees.

The R package caret is an all-purpose machine learning package that contains all facets of learning models, including options for boosting, bagging, and tuning existing models. Before running the algorithm, it is necessary to classify the method with which random sampling from the training data set will take place. The function trainControl in the caret package lets us do just that. Within this function, we describe through what method we want to use to resample, how many iterations we would like to do of this resampling method, what we would like the train function to return, and several options on how to allocate space during the creation of the model. If no training control is specified,

the algorithm will automatically use bootstrapping. However, for this case I achieved better results with the resampling method called repeatedCV. Repeated CV, or repeated cross validation consists of randomly dividing the original data into k equal sized samples, where k – 1 samples create predictions for the kth sample. This process is repeated for all k samples.

Even though I ran the random forest algorithm using the caret package with all possible variables in the data set, the final model automatically drops the unused variables. In our case, it dropped eight covariates, leaving us with a model that makes use of 20 variables. One important thing to make note of is that the variable that indicated the result of the previous game was dropped. This means that although I had initially planned to introduce that element to the simulation, it will not play a role. The following table shows the five most important ones, as given by the varImp() function. In order to see the process through which varImp() chooses the most important covariates, please refer to the documentation for the Caret package. A reference is provided at the end of this project.

Variable
Season wins
Free throw percentage
Offensive rebounds
Seed
Field goal percentage

Table 7. Five most important variables used in classification

Similar to the logistic regression model, we can input a vector that contains two pieces of information, the home team statistics and the statistics of the away team, to output the probability the home team will win. For example, take the 2015 Golden State Warriors and the 2015 Cleveland Cavaliers, which are both from the current year, and will be battling it out in the 2015 NBA finals. One data frame will contain all necessary information for the Warriors, and one data frame will have all the necessary information for the Cavaliers. Then depending on what team we want to receive the proper home game probability of victory for, we subtract the away team statistics from the home team statistics. Using the predict function, we can use the final random forest model to output a vector containing the probability of both winning and losing. In the case of the Golden State Warriors versus the Cavaliers, the following table displays all possible events.

Team at home	Probability of victory	Probability of loss
Warriors	74.6%	25.4%
Cavaliers	29.2%	70.8%

Table 8. Estimated probabilities of winning at home for each team

Though these probabilities seem rather extreme, keep in mind that the Warriors in 2015 had an all-time great season, and thus it is of no surprise that we see such a high chance of winning (they had 67 wins in the season compared to the

cavaliers' 53). One interesting thing to note, the chance of the cavaliers winning at home compared to on the road is associated with a mere 4% increase.

Simulation

Now that the best possible model has been established, I will apply it to a simulation of an entire series. For those uninitiated to the NBA playoffs, it works in the following way: two teams have a best of seven series to determine who advances. This means that the first team to four victories wins the series. However, what makes the simulation interesting is how they rotate between home and away games. The team with the better record starts out with two games at home. Then, the team with the worse record gets to play two games at home. If neither team has won four games by this point, they rotate back and forth until a team reaches their fourth win. This information will allow us to set up the simulation in the proper way.

In order to obtain the necessary information to go through the simulation, data from two opposing teams is needed. Based on the information, we can subtract the statistics of the away team from the home team for both teams, as each team plays at least two games at home in a NBA playoffs series. Based on these resulting two data frames, we can use the random forest model to calculate the probabilities of home victory for both teams, and we're set with information to use in the simulation.

The actual simulation is quite simple. This simulation is always from the viewpoint of the team with the better record in that season, as that team is always the first one to play at home. Each game is decided by one of the following:

- Sampling from a Bernoulli distribution with probability parameter equal to the probability of that the team with the better records wins while playing at home
- Sampling from a Bernoulli distribution with probability parameter equal to the probability of that the team with the worse records wins while playing at home

This is done 7 times, with respect to the format of when a team plays at home or away. Each pick made by the random sample is put into a vector of 7 values, and the sum is computed. Since the event of sampling a 1 from the Bernoulli distribution always implies the team with the better record won, no matter if that team is at home or not, we know that If the sum of the 7 outcomes is greater than or equal to 4, the team with the better record is classified as winning the series. To estimate the final probability of that team winning, we simulate the 7 game series a thousand times, and count the number of times the team with the better record wins out of a thousand.

Several weeks ago, Nate Silver, a famous statistician known for his fivethirtyeight blog (http://fivethirtyeight.com/), posted his own simulation of 10000 trials to estimate the probability of each team in the playoffs of 2015 advancing through the first round. In other words, his model and simulation sets out to do the same thing as mine, thus I thought it would be interesting to compare our results, and test it against what actually happened. Therefore, the

following table shows each series I simulated, my prediction, the prediction from Nate Silver, and the actual result.

First round series	Prob. of advancing	Nate Silver's prob. of advancing	Who wins?
Warriors vs. Pelicans	93.6% (Warriors)	92% (Warriors)	Warriors
Rockets vs. Mavericks	49% (Rockets)	66% (Rockets)	Rockets
Clippers vs. Spurs	48% (Clippers)	49% (Clippers)	Clippers
Blazers vs. Grizzlies	59.8% (Grizzlies)	54% (Grizzlies)	Grizzlies
Hawks vs. Nets	97.5% (Hawks)	90% (Hawks)	Hawks
Cavaliers vs. Celtics	86.1% (Cavaliers)	87% (Cavaliers)	Cavaliers
Bulls vs. Bucks	57.5% (Bulls)	70% (Bulls)	Bulls
Raptors vs. Wizards	64.3% (Raptors)	60% (Raptors)	Wizards

Table 9. Results of simulation in 1st round and comparison to Nate Silver model

Our predictions were within 10% of each other on 6 out of the 8 of the first round series, and the only substantial discrepancies were between the Bulls vs. Bucks and the Rockets vs. Mavericks series. My model gave the Bulls a much lower probability of winning, similar with the Rockets vs. Mavericks series, where I actually gave the Rockets a 49% percent chance of winning. This is likely due to the fact that Rockets have been over-achieving all season, and doing much better despite not having incredible statistics. This presents a slight problem because my model is primarily focused on regular season data. Overall, the model seems to be performing quite well. If we were to classify a series win when the probability is greater than 50%, then my model would be correct 6 out of 8 times, and Silver's model correct 7 out of 8 times.

Conclusion

Ever since I started watching sports, I have been interested in how companies like ESPN arrived at their predictions. After going through the process myself, I finally have an idea of what goes on behind the scenes. I was glad to have the opportunity to create random forests and get a much better understanding of them than I had before. Though I had a loose understanding of the random forest's purpose, and how they are created, I now have a much deeper understanding that I hope to apply to the workforce soon. Similarly, it was quite interesting to compare the results to logistic regression, which is a topic I've been familiar with for a while now.

Though I tried to make this project as comprehensive as possible, there are certainly things I'd like to add. First off, more data may greatly benefit the classification capability of my models. It would be worthwhile to develop a method to scrape large amounts of basketball data from www.basketball-reference.com. This may especially affect the logistic regression model created for game 1's of a series, as I had only 68 instances of that event. Second of all, I think it would be interesting to analyze the point spread of a game, rather than focusing on the binary response of winning and losing. This would still lead to the same simulation results, but would provide more information regarding the outcome of each game. Similarly, random forests do not only focus on classification, but regression as well, so they still could have been used to predict the point spread in a game, since that is a continuous response. Thirdly, It would have been interesting to include more complex conditionals within the simulation, such as momentum, the point differential of the previous game, how much the previous series was won by each team, and how much rest time was in between each game. Even if these turned out to be insignificant in correctly classifying a win or loss, it would have been an interesting conclusion to make.

Acknowledgements

1. Dr. Potter for assisting me with this project, and coming up with an interesting premise.
2. Johns Hopkins, for offering interesting courses on data science and making them freely available.
3. R software and information sites such as StackOverflow, Stats.StackExchange, and Quora.
4. Basketball-Reference.com for providing me with historical data.

References

"NBA & ABA Playoffs Series History." Basketball Reference. Ed. David Corby. Sports Reference, n.d. Web. 11 June 2015

Beyamin, Dan. "A Gentle Introduction to Random Forests, Ensembles, and Performance Metrics in a Commercial System | Blog & Press." *Blog Press*. Citizen Net, 09 Nov. 2012. Web. 11 June 2015.

L. Breiman, "Random Forests", Machine Learning, 45(1), 5-32, 2001.

"Cross-validation and the Bootstrap." *Regression and Time Series Model Selection* (1998): n. pag. *Lagunita Stanford*. Web. 11 June 2015.

Kuhn, Max. "The Caret Package." *The Caret Package*. N.p., 15 May 2015. Web. 11 June 2015.

Appendix

The following R code is the cleaned up version of all R code used to create this project. If you are interested in the data set I used, please email me at boudeyz@gmail.com and I can provide you the necessary data.

```
rm(list = ls())
setwd("~/Documents/SeniorProject")

#necessary packages to run code
library(caret)
library(MASS)
library(randomForest)
library(ggplot2)

#### DATA PREPARATION#######################################

season = read.csv("seasonStats.csv",header = TRUE)
game = read.csv("gameStats.csv",header = TRUE)

data = merge(season,game,by = "home")

#example of a model that wouldnt make sense
fit = glm(as.factor(win) ~ FG + awayFG,data = data,family = "binomial")
d2 = data
colnames(d2)[3] = "awayFG"
colnames(d2)[37] = "FG"
predict(fit,d2[1,],type = "response")

final =
data.frame(data$win,data$home,data$away,data$gameInSeries,data$wonPrevious)
names = c("win","home","away","gameInSeries","wonPrevious")
colnames(final) = names

#create variables
final$seed = data$homeSeed - data$awaySeed
final$SRS = data$SRS - data$awaySRS
final$PACE = data$Pace - data$awayPace
final$Offrtg = data$Offrtg - data$awayoffrtg
final$Defrtg = data$Defrtg - data$awaydefrtg
final$seasonWins = data$homeSWins - data$awaySWins
final$MP = data$MP - data$awayMP
final$FG = data$FG - data$awayFG
final$FGA = data$FGA - data$awayFGA
```

```
final$FG. = data$FG. - data$awayFG.
final$X3P = data$X3P - data$away3P
final$X3PA = data$X3PA - data$away3PA
final$X3P. = data$X3P. - data$away3P.
final$X2P = data$X2P - data$away2P
final$X2PA = data$X2PA - data$away2PA
final$X2P. = data$X2P. - data$away2P.
final$FT = data$FT - data$awayFT
final$FTA = data$FTA - data$awayFTA
final$FT. = data$FT. - data$awayFT.
final$ORB = data$ORB - data$awayORB
final$DRB = data$DRB - data$awayDRB
final$TRB = data$TRB - data$awayTRB
final$AST = data$AST - data$awayAST
final$STL = data$STL - data$awaySTL
final$BLK = data$BLK - data$awayBLK
final$TOV = data$TOV - data$awayTOV
final$PF = data$PF - data$awayPF
final$PTS = data$PTS - data$awayPTS



#feature plot (standardized)
nd = final
nd$win = ifelse(nd$win ==1,"win","loss")
theme1 <- trellis.par.get()
theme1$plot.symbol$col = rgb(.2, .2, .2, .4)
theme1$plot.symbol$pch = 16
theme1$plot.line$col = rgb(1, 0, 0, .7)
theme1$plot.line$lwd <- 2
trellis.par.set(theme1)
featurePlot(x = scale(nd[, 4:33]),
            y = as.factor(nd$win),
            plot = "box",
            layout = c(5, 6))

#calculate correlation between response and predictors
x = data.frame(biserial(final[,4:33],final$win))
x = t(x)


#subset game 1 data frame
game1 = final[final$gameInSeries == 1,]
game1 = game1[,-5]
```

```
#subset all other games data frame
gameOther = final[final$gameInSeries != 1,]

prop.table(table(game1$win))
#game 1 home team wins 72.05% of the time

prop.table(table(gameOther$win))
#all other games home team wins 59.8% of the time



#### LOGISTIC REGRESSION GAME 1 ######################################

#get proper data frame
mod1 = game1[,-c(2,3)]

#stepwise regression that eliminates perfectly correlated variables
temp = glm(as.factor(win) ~ . - X2P - X2P.,data = mod1, family = "binomial")
s = stepAIC(temp,direction = "both")
s$anova

#final model
game1Fit = glm(as.factor(win) ~ TRB + PF,data = mod1, family = "binomial")
summary(game1Fit)

#prediction example
pred = predict(game1Fit,data.frame(TRB = -4, PF = 3))



#### LOGISTIC REGRESSION MODEL ALL OTHER GAMES
##################################

#get proper data frame used for model
modO = gameOther[,-c(2,3)]

#create training and test data set
t = createDataPartition(y = modO$win,p = .85,times = 1,list = FALSE)
training = modO[t,]
test = modO[-t,]

#stepwise regression
fit = glm(as.factor(win)~.,data = training,family = "binomial")
step = stepAIC(fit,direction = "both")
step$anova
```

```
otherGamesMod2 = glm(as.factor(win) ~ PACE + FGA + X3P + X3PA + X2P + X2P. +
FTA +
                    FT. + DRB + STL + PTS,family = "binomial",data = training)

summary(otherGamesMod2)



#####CLASSIFICATION FOR LOGISTIC REGRESSION
################################

#accuracy check
p = predict(otherGamesMod2,test,type = "response")
right = c()
m = cbind(p,test)
m$resp = 0
for(i in 1:nrow(m)){
  if(m$p[i] > .5){
    m$resp[i] = 1
  } else{
    m$resp[i] = 0
  }
  if(m$resp[i] == m$win[i]){
    right = c(right,"right")
  } else{
    right = c(right,"wrong")
  }
}
prop.table(table(right))
table(m$resp,m$win)




### SIMULATION  WITH LOGISTIC
REGRESSION#########################################

#NOTE: not included in report

#Game 1 Stats (without wonPrevious)

g1team1stats = data.frame(seed = 1, SRS = 1 ,DEFRTG = 3 ,seasonWins = 62, FGA =
83.5, X3PA = 21.4,
                      X2P = 32,X2P. = .517 ,FT = 15.7,ORB = 9.3,
```

```
                    DRB = 34,TRB = 43.3,PF = 18.2)
g1team2stats = data.frame(seed = 2, SRS = 7, DEFRTG = 11,seasonWins = 54, FGA =
76.5, X3PA = 22.3,
                    X2P = 32,X2P. = .558,FT = 17.5,ORB = 7.6,
                    DRB = 29.2,TRB = 36.9,PF = 19.5)
teamAg1 = g1team1stats - g1team2stats
teamBg1 = g1team2stats - g1team1stats


#Game Other stats(with wonPrevious),
#if won
goteam1statsW = data.frame(seasonWins = 62,FG. = .486,X3P = 8.5, X3PA = 21.4,
                    X2P = 32,X2P. = .517,FT = 15.7,FT. = .785,ORB = 9.3,
                    DRB = 34,TRB = 43.3,BLK = 5.1,PF = 18.2,PTS = 105.2,wonPrevious
= 1)
goteam2statsL = data.frame(seasonWins = 54,FG. = .501,X3P = 8.1, X3PA = 22.3,
                    X2P = 30.2,X2P. = .558,FT = 17.5,FT. = .76,ORB = 7.6,
                    DRB = 29.2,TRB = 36.9,BLK = 4.5,PF = 19.5,PTS =
102.2,wonPrevious = 0)
teamAgoW = goteam1statsW - goteam2statsL
teamBgoL = goteam2statsL - goteam1statsW

#if lost
goteam1statsL = data.frame(seasonWins = 62,FG. = .486,X3P = 8.5, X3PA = 21.4,
                    X2P = 32,X2P. = .517,FT = 15.7,FT. = .785,ORB = 9.3,
                    DRB = 34,TRB = 43.3,BLK = 5.1,PF = 18.2,PTS = 105.2,wonPrevious
= 0)
goteam2statsW = data.frame(seasonWins = 54,FG. = .501,X3P = 8.1, X3PA = 22.3,
                    X2P = 30.2,X2P. = .558,FT = 17.5,FT. = .76,ORB = 7.6,
                    DRB = 29.2,TRB = 36.9,BLK = 4.5,PF = 19.5,PTS =
102.2,wonPrevious = 1)
teamAgoL = goteam1statsL - goteam2statsW
teamBgoW = goteam2statsW - goteam1statsL



### GAME 1 PREDICTIONS

predHomeAg1 = predict(game1Fit,teamAg1,type = "response")
predAwayAg1 = 1 - predHomeAg1

predHomeBg1 = predict(game1Fit,teamBg1,type = "response")
predAwayBg1 = 1 - predHomeBg1

#only run if using rf for round 1
p1 = predict(tree,teamAg1,type = "prob")
```

```
predHomeAg1 = p[2]
predAwayAg1 = 1 - predHomeAg1

p2 = predict(tree,teamBg1,type = "prob")
predHomeBg1 = p2[2]
predAwayBg1 = 1 - predHomeBg1


### ALL OTHER GAMES PREDICTIONS

#if team 1 wins prev game
predHomeAgoW = predict(otherGamesMod,teamAgoW,type = "response")
predAwayAgoW = 1 - predHomeAgoW

#if team 1 loses prev game
predHomeAgoL = predict(otherGamesMod,teamAgoL,type = "response")
predAwayAgoL = 1 - predHomeAgoL


#if team 2 wins prev game
predHomeBgoW = predict(otherGamesMod,teamBgoW,type = "response")
predAwayBgoW = 1 - predHomeBgoW

#if team 2 loses prev game
predHomeBgoL = predict(otherGamesMod,teamBgoL,type = "response")
predAwayBgoL = 1 - predHomeBgoL


#find probability of winning 7 game series that takes into account previous game

teamHCwins = c()
games7 = c()
for(i in 1:1000){
 #game 1
 games7[1] = sample(c(1,0),1,replace = TRUE,c(predHomeAg1,predAwayAg1))
 #game 2
 if(games7[1] == 1){
  games7[2] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoW,predAwayAgoW))
 } else{
  games7[2] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoL,predAwayAgoL))
 }
 #game 3
 if(games7[2] == 1){
  games7[3] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoW,predAwayBgoW))
 } else{
  games7[3] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoL,predAwayBgoL))
```

```
  }
  #game 4
  if(games7[3] == 1){
    games7[4] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoW,predAwayBgoW))
  } else{
    games7[4] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoL,predAwayBgoL))
  }
  #game 5
  if(games7[4] == 1){
    games7[5] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoW,predAwayAgoW))
  } else{
    games7[5] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoL,predAwayAgoL))
  }
  #game 6
  if(games7[5] == 1){
    games7[6] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoW,predAwayBgoW))
  } else{
    games7[6] = sample(c(0,1),1,replace = TRUE,c(predHomeBgoL,predAwayBgoL))
  }
  #game 7
  if(games7[6] == 1){
    games7[7] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoW,predAwayAgoW))
  } else{
    games7[7] = sample(c(1,0),1,replace = TRUE,c(predHomeAgoL,predAwayAgoL))
  }

  #determine who wins
  if(sum(games7) >= 4){
    teamHCwins=c(teamHCwins,1)
  } else{
    teamHCwins = c(teamHCwins,0)
  }
}

prop.table(table(teamHCwins))

hist(teamHCwins)


##########RANDOM FORESTS GAME 1 AND
IMPORTANCE###################################

#Game 1 Model

#create training/test
t = createDataPartition(y = mod1$win,p = .85,times = 1,list = FALSE)
```

```
training = modO[t,]
test = modO[-t,]


#game 1
tree = randomForest(as.factor(win) ~ seed + SRS + DEFRTG + seasonWins + FGA +
X3PA +
                X2P + X2P. + FT  + DRB + TRB + PF,data = mod1,importance = TRUE)
tree = randomForest(as.factor(win) ~ seed + SRS + DEFRTG + seasonWins + FGA +
X3PA +
                X2P + X2P. + FT  + DRB + TRB + PF,data = training,importance =
TRUE)


#find importance for game 1 model and graph it
imp = data.frame(importance(tree))
rowNames = row.names(imp)
imp = cbind(rowNames,imp)
newimp = imp[order(imp$MeanDecreaseAccuracy),]
newimp$rowNames = as.character(newimp$rowNames)
newimp$rowNames = factor(newimp$rowNames,levels = unique(rowNames))
ggplot(data = newimp,aes(y = reorder(rowNames,MeanDecreaseAccuracy),x =
MeanDecreaseAccuracy)) +
  geom_point() + ggtitle("Mean Decrease Accuracy per predictor") + ylab("variable") +
theme_set(theme_grey(base_size = 17))

tree = randomForest(as.factor(win) ~ .,data = mod1,importance = TRUE)
varImpPlot(tree,main = "Variable Importance")
tree

#######FULL RANDOM FOREST MODEL ############

mod = final[,-c(2,3)]

#divide into test and training data set
t = createDataPartition(y = mod$win,p = .75,times = 1,list = FALSE)
training = mod[t,]
test = mod[-t,]

#create model and resampling method
mod$win = ifelse(mod$win == 1,"win","loss")
mod$win = as.factor(mod$win)
ctrl <- trainControl(method = "repeatedcv",
                number = 10, repeats = 10)
rf = train(win~. - gameInSeries - wonPrevious,data = mod,method = "rf",trControl = ctrl)
```

```
#check importance
varImp(rf)
plot(rf)

#define final model to be used in simulation
fm = rf$finalModel
fm



#########SIMULATION USING FULL GAME
MODEL#################################

#this data frame contains statistics for the playoff teams in the last season. can
#be used to predict with
predictD = read.csv("predictDF.csv",header = TRUE)

teamAstats = predictD[1,-1]
teamBstats = predictD[2,-1]
teamAhome = teamAstats - teamBstats
teamBhome = teamBstats - teamAstats

#if team with better record is home
probAHome = predict(fm,teamAhome,type = "prob")


#if other team is at home
probBHome = predict(fm,teamBhome,type = "prob")


#SIMULATION: find probability of winning 7 game series

teamHCwins = c()
games7 = c()
for(i in 1:1000){

  #game 1
  games7[1] = sample(c(1,0),1,replace = TRUE,c(probAHome[2],probAHome[1]))

  #game 2
  games7[2] = sample(c(1,0),1,replace = TRUE,c(probAHome[2],probAHome[1]))

  #game 3
  games7[3] = sample(c(0,1),1,replace = TRUE,c(probBHome[2],probBHome[1]))

  #game 4
```

```
  games7[4] = sample(c(0,1),1,replace = TRUE,c(probBHome[2],probBHome[1]))

  #game 5
  games7[5] = sample(c(1,0),1,replace = TRUE,c(probAHome[2],probAHome[1]))

  #game 6
  games7[6] = sample(c(0,1),1,replace = TRUE,c(probBHome[2],probBHome[1]))

  #game 7
  games7[7] = sample(c(1,0),1,replace = TRUE,c(probAHome[2],probAHome[1]))

  #determine who wins
  if(sum(games7) >= 4){
    teamHCwins=c(teamHCwins,1)
  } else{
    teamHCwins = c(teamHCwins,0)
  }
}

#get final proportion of times the team with the better record wins
prop.table(table(teamHCwins))
```