Predicting the Outcome of the

NBA Championship Finals

Using MapReduce and Machine Learning

Henry Bogardus

Turner Strayhorn

**Abstract**

The overall goal of this project was to attempt to predict the outcome of the 2015-2016 season NBA championship based on a variety of Team data and Player data that we have scraped from the websites Basketball-Reference.com, stats.nba.com, and ESPN.com. We used Map Reduce to transform our data as described later in the paper and the python scikit-learn library to make predictions based on the transformed data.

**Introduction**

Each year, the NBA Finals produces one winner and, each year, there is an incredible amount of data collected by the NBA and other private parties and put online, available for our viewing, collecting, and processing. Throughout the class, we have learned and made use of data processing assets like mapreduce and machine learning libraries like scikit learn. There are, obviously, unpredictable reasons that a team may not win or be predicted to win, then lose such as random player injury (i.e. Derrick Rose in 2011). However, we want to know if there is 1) an obvious way, and 2) a less obvious way to predict the outcome of the NBA championships using transformations on this data we have collected and machine learning with psikit learn.

**Data Collection**

**1. NBA Stats**

To collect our team season statistics and game logs, we scraped from the NBA stats website, which makes it very easy to collect data as data can be downloaded in JSON form and easily stored into a Pandas DataFrame, which is then easily stored to a CSV files with the to\_csv() attribute. Each JSON for each season was read into its own DataFrame, which was then stored in a list. When all data was collected the DataFrames were all concatenated and stored in the CSV files.

**2. Basketball-Reference**

We collected team rosters and player data from Basketball-Reference. This website was much trickier to scrape than NBA Stats and has stated that they prefer not to be scraped. They have been known to ban IP addresses if they think scraping is occurring from the address.

We chose to use selenium browser automation instead of python requests as we used with NBA Stats for this reason exactly. The website did, luckily, provide an option for exporting player data as a csv which allowed us to automate all of the downloads. The URLs were all uniform and described as

http://www.basketball-reference.com/teams/<team-abbreviation>/<year>.html

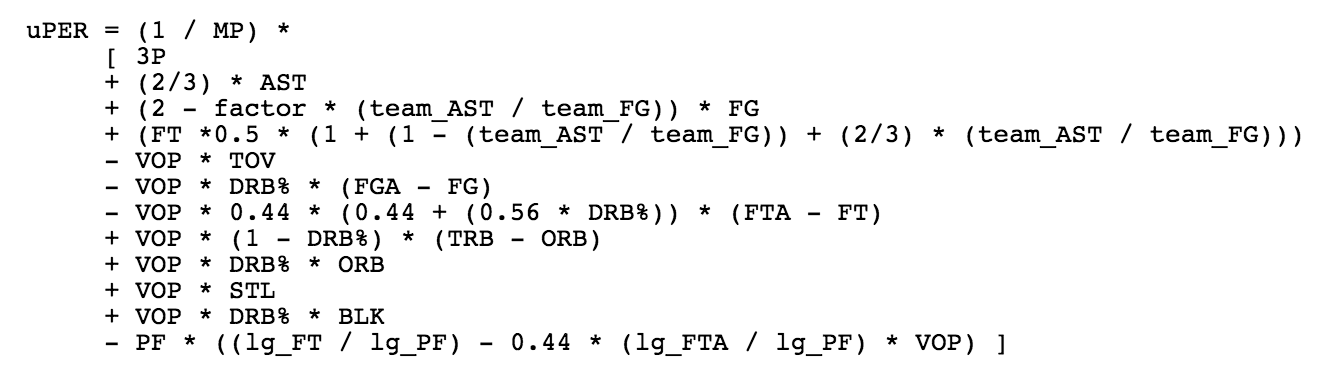
So we were able to make a python list of each team abbreviation and open the url in Google Chrome with selenium web browser. To export the player information, we used a JavaScript call that is made when the export button was pressed. Initially, we put a timer on the script to make sure it didn't seem to suspicious. However, we quickly realized how slow it was to collect the data. There are 30 NBA teams and 2 CSVs downloaded per team, the roster and the player totals for the season. The site is slower and would take about 10-20 seconds to load leading to about 20 minutes of data collection per season with the timers. Even after we removed the timer, the collection was still quite slow and would take 10-15 minutes per season. We decided the past 10 years of data was more than plenty for our collection. The rosters were all compiled into one big csv and the season and team abbreviation was appended.

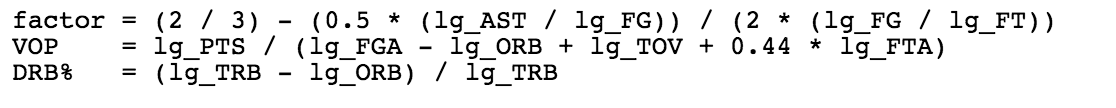
**Data Transformations**

**1. Game Logs**

Most data transformations were done using yelp's MRJob module for python. We first did transformations on game-logs to extract, per season, for how many games did the winning team rely on freethrows to win. The mapper step mapped the log data by the GAME\_ID. The combiner step then read in that data, determined the winning team, and calculated the number of points the team got from free throws. The combiner step also calculated the difference in the number of points the winning team won by and sees if the number of free throws is greater than that value. Finally, the reduce step sums the number of games won because of free throws. We discussed and decided not to take into account the losing teams' free throws when coming up with this metric because we determined it was irrelevant for what we wanted to determine.

**2. Player Stats**

For player logs, we wanted to see if average player weight, height, and number of colleges had any correlation with NBA championships. We wrote a map-reduce job for each of these metrics that read in the player rosters for each team and each season and reduced them with the key as the season and the team name. Finally, we decided to use the player efficiency rating PER1. We totaled the PERs of each team's players and added that to our list. The equation is described as the following. 

where

../../../../Desktop/Screen%20Shot%202016-04-28%20at%2010.25.21%20PM.png

../../../../Desktop/Screen%20Shot%202016-04-28%20at%2010.22.19%20PM.png../../../../Desktop/Screen%20Shot%202016-04-28%20at%2010.22.23%20PM.png

This metric, as ridiculous as it looks, does a good job "summing up the positive accomplishments of a player, subtracts the negative accomplishments of a player, and returns a per-minute rating of a player's performance. This was an integral part of our assessment made for the Player Statistics that we scraped.

**Machine Learning**

To solve our high level problem of which team exactly is the most likely to win the 2016 NBA championship, we ran 1000 simulations of our machine learning algorithm to determine the percent chance of every team to win for the 2016 season. Starting off, we based a lot of our functionality off of Dr. Fabbri’s machine learning example file on GitHub and then tailored it to our needs and the differences in how the data was stored. The classifier that produced the set of results that was the most easily understood was the SGD linear model. We achieved giving each team a percent chance of winning by running a thousand simulations, keeping a running counter of how many times each team got a simulated win, then dropping teams not in the playoffs that year and normalizing to 100% shared between the sixteen playoff teams. We then can see which teams have the highest percentage chances of winning. To train the classifier we had a number of features (over 30) to choose from to find the right combination to most accurately pick a champion. There were a number of complications involved with keeping the NBA data consistent over the past 20 years, as there have been a number of changes to team names, team cities, and the number of teams in the league. These problems showed up when using Sci-Kit Learn’s Predefined Split and defining exactly which rows to train on and which to test on. To fix this, I added a quick couple lines before the main method that standardized the process of which lines were for testing versus which lines were for testing based on the year for which we were looking for results. In addition to using the SGD classifier, we also use a Gaussian Naïve-Bayes classifier and a Random Forest Calculator to test the Area Under the Curve score.

**Outcomes**

Henry 1 page **Conclusion**

Turner 1 page