Predicting the 2015 NCAA Tournament

Dan Loman

March 19, 2015

1 Introduction/Motivation

I decided to take on the kaggle competition "March Machine Learning Mania 2015", where the goal is to create an algorithm to predict the 2015 NCAA Tournament. As a huge sports and college basketball fan who's studying analytics, I immediately jumped at this opportunity.

Over the course of my graduate program at the University of San Francisco, I have been exposed to many topics in statistics and data science. In addition to my interest in the NCAA Tournament, I am using this project to hone the skills that I have learned and even pick up a few things. I hit on many of these topics in this project, including Machine Learning, SQL, Data Acquisition, and Exploratory Data Analysis.

2 The Data

I used two sources for my data: the kaggle competition and kenpom.com. Data I used from kaggle came in 3 separate tables: the first table mapped team name to an ID number, and the next two consisted of game results from every regular season and tournament game for the past 20 years.

The data I got from kenpom is extremely rich and contains a plethora of advanced statistics. It cost me $20 to attain for a year long membership, which is a small price to pay for arguably the premium college basketball data around. I used 26 statistics for each team, explained below (Most of these statistics are measure for the team and their opponents, i.e. (Offensive, Defensive)):

- *Pythagorean Win Percentage*: The expected win percentage a team would have against an average opponent.
- *Adjusted Offensive Efficiency*: Points per 100 possessions, adjusted for opponents.
- *Adjusted Defensive Efficiency*: Points allowed per 100 possessions, adjusted for opponents.
- *Adjusted Tempo*: Possessions per 40 minutes, adjusted for opponents.

- *Effective Field Goal Percentage (Offensive, Defensive)*: Field Goal Percentage weighted for 3 point shots.
- *Offensive Rebound Rate (Offensive, Defensive)*: Percent of available Offensive Rebounds grabbed.
- *Turnover Rate (Offensive, Defensive)*: Percent of Possessions that end in a turnover.
- *Free Throw Rate (Offensive, Defensive)*: Measures how often teams get to the line and the percentage they hit free throws at.
- *3 Point Percentage (Offensive, Defensive)*
- *2 Point Percentage (Offensive, Defensive)*
- *Free Throw Percentage (Offensive, Defensive)*
- *Block Percentage (Offensive, Defensive)*
- *Steal Percentage (Offensive, Defensive)*
- *Assist Percentage (Offensive, Defensive)*
- *3 Point Attempt Percentage (Offensive, Defensive)*: The percentage of a team's shots that are 3 pointers

A potentially major problem with this data, however, is that most of it is from both the regular season and the tournament, so when I went to test my model I had some of my testing data contained in my training data.

I'll explain in the next section how I was able to merge these two data sources into one.

3 Data Preprocessing

The data preprocessing stage was the one I spent the most time on during this project. When I started the project I had a lot of data tables that I needed to combine into one usable dataframe for machine learning. I decided to use mySQL (through python using mysql.connector) and pandas for my data preprocessing.

I encountered several annoying data issues during this stage. The first was that my data from kenpom was not consistent on a year to year basis. For example, some years contained column headers while others didn't. Most issues were minor and just required me to manually change a few things in the .csv file. In other cases it forced me to make my SQL code less generalizable. The second annoying issue was that team names that were inconsistent from table to table within kenpom tables and between kenpom and kaggle, which made it hard to join tables. For example, I might see 'USF' in one table and 'U San Francisco' in the other. I solved this issue by manually inspecting teams that did not match and creating a mapping from the "incorrect" team name to the correct one.

The data cleaning thus far was only for my kenpom data, so that I could join it to my kaggle data. I still had a lot of work to do with kaggle. First, I only considered data from 2003 onward, since that was all that I had from kenpom. Next, I wanted to adjust for home court advantage. I used BeautifulSoup and urllib2 to scrape data from http://www.predictionmachine.com/college-basketball-homecourt-advantage, which contains the estimated home court advantage for each team over the past 15 years. I then adjusted the scores of each game in the kaggle data to account for home court advantage.

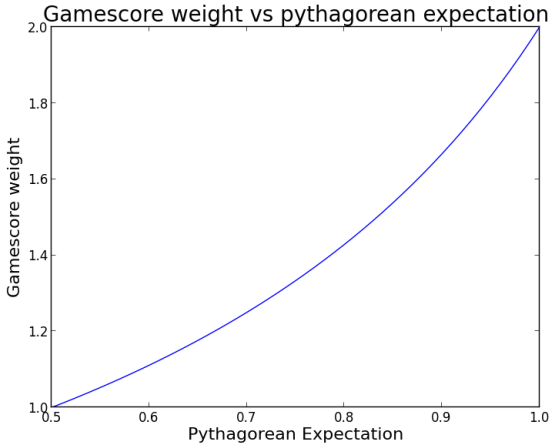
Next, I wanted to put a heavier weight on certain games for inclusion in my machine learning model. First, I wanted to give a higher weight to games with a high point differential. The thinking here is that you can be more confident that the better team won when they win by a lot vs when they win by a little. I used Daryl Morey's Pythagorean expectation to extract the win percentage of each winning team based on the final score:

$$Win = \frac{pointsfor^{13.91}}{pointsfor^{13.91} + pointsagainst^{13.91}} \tag{1}$$

I then used the following equation to weigh each game accordingly, based on the winning team's Pythagorean expectation:

$$weight = \frac{1}{1.5 - pyth} \tag{2}$$

This formula gives no additional weight to a toss up game (overtime, pyth = .5) and doubles the weight of a blowout (pyth = 1). The function looks as follows:

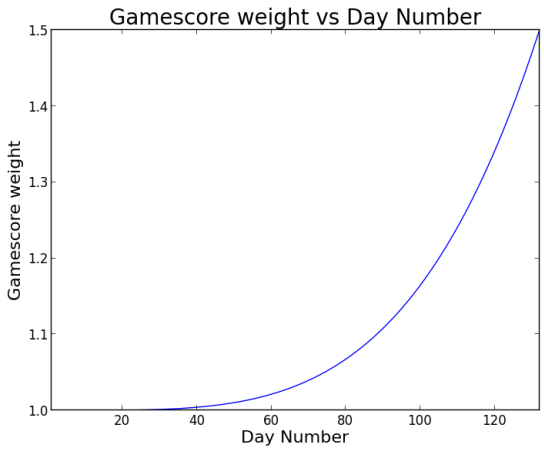


When people make their brackets, one thing they may base their predictions on is the recent success of a team. I've taken that into account as well, by giving higher weight to late season games. This wouldn't reward teams that play well late in the season, but it may reward the types of teams that play well come

tourney time. I used the following equation to weigh each game according to the day it was played on (day number spans from 1 to 132):

$$weight = .5 * (\frac{daynumber}{132})^4 + 1 \tag{3}$$

This formula gives no additional weight to games played earlier in the season and gives weight up to 1.5 for games closer to the tournament. The function looks as follows:



Since I am including all past games since 2003 in my training data, there will be an unbalanced ratio between regular season and tournament games. I countered this by weighing tournament games such that they would be counted equally to regular season games. The idea behind this is to identify any statistics that may be more indicative of tournament success than regular season success.

Finally, I had to transform the data into a usable dataframe for machine learning. For each game I randomly assigned a team as "Team 1" and the other as "Team 2". I then created a target variable called "result", which indicates whether or not Team 1 won the game (1 for win, 0 for loss). I merged my kenpom data with my kaggle data based on team name and year. My indicator variables were the kenpom data variables for each team 1 and team 2 for every game.

4 Machine Learning

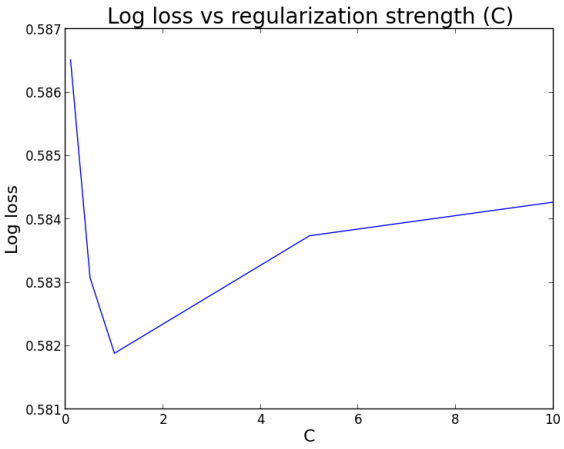
The goal of the kaggle competition was to minimize log loss for all NCAA tournament predictions from 2011-2014. Log loss is defined below:

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} log(p_{ij}) \tag{4}$$

Once I finally had the data in a usable format, I could begin the machine learning part of this project. My first intuition was to use a random forest classifier, since random forests had usually worked well for me in the past. Instead, I found through trial and error that a bagging classifier with logistic regressions gave me the lowest log loss. To improve on this model, did feature reduction and parameter tuning.

In feature reduction, I first sorted my features by their importance (extracted from random forest - I wouldn't expect them to be too different) and found a threshold where adding more features was harmful to my model. I reached this point at around the 40th feature out of 52 total. I continued to trim my features using an interative 'leave one out' approach, where I went through each feature, left it out of the model and determined which feature exclusion helped my model the most. I did this around 4 times before finding diminishing returns.

In a logistic regression, there are 2 primary parameters to check: regularization type and regularization strength (C). The default regularization strength is 1, so I tried multiple models with c spanning from .1 to 10. I found that the default regularization strength of 1 gave me optimum results:



I also found that the default regularization type of 'L2' regularization worked better than 'L1' regularization.

Finally, I found I could further minimize my log loss by raising my probabilities to the .88 power and then renormalizing them to sum to one. What this is essentially doing is slightly skewing my predictions toward 50 percent, making my predictions more liberal towards upsets. I suspect it worked well on my training data because there have been numerous upsets in the past 4 years, so predicting less definitive probabilities minimizes error in cases where upsets are abundant. The obvious concern here is that I am overfitting to my training data.

5 Results and 2015 predictions

My model confirms the following variables to be of most importance when predicting NCAA Tournament success: Pythagorean Win Percentage, Offensive Efficiency, Defensive Efficiency, Effective Field Goal Percentage and Defensive Effective Field Goal Percentage. These results are not at all surprising - it says that the most successful teams in the tournament are the best teams in the regular season (Pythagorean Win Percentage), with the best offenses, best defenses, that make the most shots and prevent their opponents from making their shots. I was hoping to find one strong predictor that would surprise me, but found no such luck (after the above variables, everything else is more or less the same).

I didn't submit my results to kaggle, but I was achieving a log loss of around .582 for my training data, which would put me in the top 1/3rd of the competition. For context, a purely seed-based log loss is around .69, while Vegas odds will be around .57. So all things considered, I was pretty happy with my results.

Next came the tricky part of the assignment: using this model to forecast the 2015 tournament. After going through the annoying data preprocessing stage I decided to use all games prior to the tournament and from 2003 onward as my training data, and my test data would be every possible matchup in the 2015 tournament. My weighting criteria would be the same. After running the data through my model, I extracted the win probabilities for every team against every other team. I then manually inserted the first round matchups, and used a Markov chain to extract the probabilities of each team making each round. I won't go through my predictions here because I will be keeping track of the tournament at *greenandgoldanalytics.wordpress.com*.

6 Future Considerations

This project was done under roughly a 3 week time constraint in the midst of a busy semester. I'm happy that I was able to achieve results but my goal is to make this a multi-year endeavor. Here are some things I wish I could have tried with more time, and may look to improve on with this project in the future:

In data collection, I would like to try different data sources. As mentioned above, my data from kenpom is slightly contaminated which gives me pause about the validity of my results. I would like to see what other data is available to download or scrape that may be useful. I might be interested in using indicator variables, such as if there are NBA prospects on a team, coaching tournament track record, if the team comes from a BCS conference, or if the team won their conference tournament.

In data preprocessing, I considered making my target a continuous variable based on the pythagorean win expectation of team 1, but instead went with the binary classification of win/loss. My reasoning is that my ultimate goal is to predict whether or not a team won or lost in the tournament, regardless

of final score, and that my weights would take point differential into account when fitting my model. In the future it would be worth investigating if I can improve my result by treating this as a regression problem rather than a binary classification. I would at least like to go back and test my model with different weighting functions, which were chosen intuitively but arbitrarily and have not been backtested. In hindsight I feel I probably didn't put enough weight on games with a high point differential.

The machine learning stage of this project was the one hit hardest by my time constraint. I didn't have enough time to try out everything I would have liked, and settled for a good but fairly basic bagged logistic regression model. One approach I tried but was unable to see improvements with was with a pseudo boosting algorithm. This algorithm would cluster games by their similarity, then iteratively improve the weights of misclassified games in my test set. If this doesn't work, I am definitely interested in using clustering in the future, if not to improve results then to see if I can find anything interesting in different clusters of tournament (or regular season) games. I would also like to spend more time on cross validation.

Finally, there is one important thing that I believe is overlooked by professional bracketologists that I want to include in future models, and that's the economic benefit of each pick. The goal of making a bracket isn't necessarily to find the most likely outcome, but to be more right than your opponents. In a large pool, and one where only the top few entries cash out, you should look to stand out by making unique picks. In my future models my goal is to predict teams that should win, while also projecting their global prediction rate to identify good value picks. So for example, if my model picks Kentucky to win, but it also picks Kentucky as the most popular pick, I might find better value in picking a slight underdog like Wisconsin or Arizona. Or if my model gives a 12 seed a 30 percent chance to win, but only 15 percent of my opposition is picking them to win, it would be behoove me to pick the underdog.

7 Conclusion

The NCAA Tournament is (objectively) the greatest sporting event of the year, and it has been a lifelong dream of mine to be a bracketologist. I had a lot of fun doing this project and was able to apply a lot of tools I've been learning in grad school to it. I can't wait to gain more knowledge and experience in data science over the years and improve my model as best I can!