Predicting the NCAA Men's Basketball Tournament with Machine Learning

Andrew Levandoski and Jonathan Lobo
CS 2750: Machine Learning
Dr. Kovashka

25 April 2017

Abstract

As the popularity of the NCAA Men's Basketball Tournament grows, so have bracket competitions, with nearly 19 million people attempting to predict the game outcomes of the 2017 tournament. With billions of dollars at stake to reward correct predictions, many people are turning to platforms based on machine learning models to construct their brackets. To develop an understanding of which attributes and techniques are essential for developing the most accurate predictions for each game played during the tournament, this paper analyzes Adaptive Boosting, K-Nearest Neighbors, Naïve Bayes, Neural Network, logistic regression, Support Vector Machine, and Random Forest learning models. In training the models on both classification and win probabilities, each method could be evaluated based on accuracy, bracket score, and log loss; based on these metrics, the regression, neural net, and random forest models performed exceptionally well. The neural net predicted outcomes with 80% accuracy while regression scored the lowest log loss. The random forest model constructed the highest-scoring bracket, earning 900 points for the 2017 tournament, well above the average score of human participants, 715.4.

1 Introduction

Every year, millions of college basketball fans attempt to predict the outcome of the National Collegiate Athletic Association (NCAA) Men's Basketball Tournament, also known as "March Madness." In 2017, nearly 19 million brackets were entered in ESPN's Tournament Challenge competition, which challenges participants to predict the result of each game of the post-season tournament. The tournament consists of six rounds of single-elimination basketball between the 68 teams judged to be the best during the regular season. Before the first round begins, the eight lowest qualifying teams are matched up in the "First Four" games to determine the last four teams in the field of 64. Ignoring these four play-in games, the tournament is divided into four regions, each with 16 teams ranked from 1 to 16. Each team's ranking is determined by an NCAA committee based on an evaluation of each team's regular season performance. The bracket is structured so that the highest seed in a region plays the lowest seed, the second highest plays the second lowest, and so on.

Nobody has ever correctly predicted the outcome of all 67 games held during the tournament, and with good reason. There are 2^{63}, or 9.2 quintillion, possible brackets. These microscopic odds prompted Warren Buffett to famously offer \$1 million to anyone who filled out a perfect bracket in 2014, and nobody came close. The disparity in quality between different teams means that many of these brackets are highly improbable, but historical results and the single-elimination nature of the tournament tell us to expect the unexpected. One major hindrance to a human's ability to make accurate predictions is the presence of bias. Everyone is biased, given the reality is that there is no clear-cut answer to the question of what factors, or features, contribute to the result of a game. Through the use of data, however, machine learning algorithms can mathematically and algorithmically and attempt to *learn* which statistics correlate the most with the result of a game. The emergence of more accurate machine learning techniques has led to increased prediction accuracy using algorithms powered by historical data. These algorithms may even demonstrate that what our intuition tells is us improbable may not actually be improbable at all.

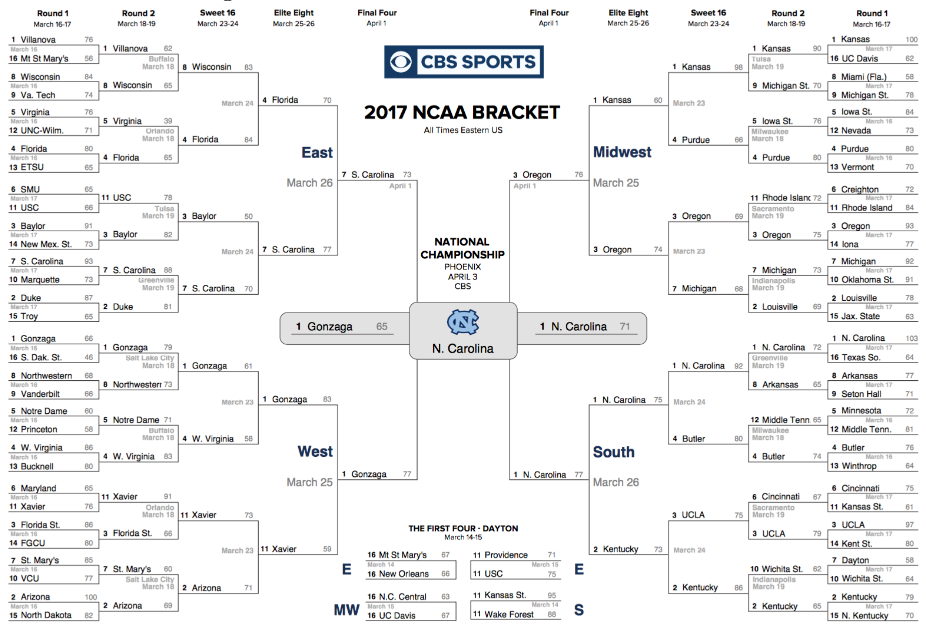
The primary inspiration for this project stems from Nate Silver's FiveThirtyEight.com prediction platform, an extremely popular resource for bracket construction. The platform offers both predictions and likelihoods of all potential matchup outcomes. Furthermore, `Kaggle.com`'s annual March Machine Learning Mania competition [1], which accepts model submissions and determines a winner based on the prediction accuracy, offers several examples of successful techniques for bracket prediction. The winner of Kaggle's 2016 competition employed Random Forests, a supervised learning method in which different decision trees are trained on different random subsets of features at training time and the output class is the mean prediction of the individual trees. In addition to Random Forests, the other models we will consider are k-Nearest Neighbors, Logistic Regression, Neural Network, Naive Bayes, Support Vector Machines, and Adaptive Boosting (AdaBoost).

This paper will detail the success of various machine learning techniques on predicting the results of the 2017 tournament as well as discuss the implementation of the random forest algorithm used in our model. Section 2 contains a discussion on previous work on the problem of predicting the

NCAA basketball tournament. Section 3 will develop the problem in further detail. In Section 4, we will report our findings and finally, the paper concludes in section 5.

2 Background

Figure 1: The results of the 2017 tournament.



Most previous work analyzing outcomes of NCAA Basketball games has been developed in the context of statistical modeling, rather than machine learning to predict future games. Further, we have found that existing work which employs machine learning techniques often suffers from poor feature selection. Thus, we intend for the work in this paper to fill this void and to offer a accurate model using learning techniques without overfitting or introducing bias through the inclusion of inappropriate features.

Lopez and Matthews [2] develop a model for submission to the Kaggle competition in which they predict the tournament outcome using logistic regression based on efficiency metrics and the Las Vegas point spread. The point spread, or betting line, for a given game is set by sports betting agencies to a value that indicates which team is the favorite to win the game, and by how many points. Bettors can win money if they wager money on the correct side of this threshold. Contrary to popular belief, the point spread is not set to a number that Las Vegas believes to be the most likely margin of victory. Rather, since sports betting agencies are motivated by making profits, the margin of victory is set to a number that they feel will cause an equal number of people to bet on the underdog as on the favorite. The authors conducted a simulation study of their model, and found that it provided less than a 50% chance of being in the top 10 of the Kaggle competition and less than 20% chance of providing the smallest log-loss in the competition. The inaccuracy in their model is driven at least in part by their inclusion of the Las Vegas point spread as a key factor in the model; the spread is determined subjectively and is not necessarily a good indicator of tournament game outcomes.

Yuan et al. [3] present the results of a team of modelers working together to forecast the 2014 tournament. In particular, they highlight the difficulty of predicting the tournament accurately, noting that most of their models failed to outperform the baseline prediction of 0.5 win probability for all games. The group of modelers produced more that 30 different models including both team- and player-level data, the majority of which were variants of logistic regression, decision trees, and neural networks. The models were trained on 10 years of historical data from a variety of public sources, including various offensive and defensive efficiency statistics, algorithmic ranking systems for the tournament teams, and raw season statistics. The researchers report that logistic regression, with significant regularization to prevent overfitting, demonstrated the best results, but that feed-forward neural networks and stochastic gradient boosting also saw success. Finally, they discuss the interesting finding that ensemble methods did not outperform the individual models, perhaps due to overfitting to the training data.

Zimmermann, Moorthy, and Shi [4] use machine learning techniques to train classification learners that can predict the outcome of games. They note that college basketball teams only play 30 games per season, can choose their own regular season opponents, and have little consistency in team composition from one year to the next due to the graduation of players. Furthermore, there is far more of a disparity in team quality than there is in professional athletics, which can distort perceptions of team quality. The researchers evaluated decision trees, rule learners, multilayer perceptron (neural network), Naive Bayes, and random forest. For their team attributes, rather than raw average statistics, they used adjusted efficiency statistics, which were calculated relative to all other teams. They claim that explicitly modeling the difference between teams' attributes did *not* improve prediction accuracy. They ultimately arrive at the interesting conclusion that feature selection seems to be much more important than model selection, and that there seems to be an "glass ceiling" on the predictive quality of their models at around 75%.

Coleman et al. [5] discuss the flaws in NCAA tournament seeding, driven primarily by human bias held by members of the NCAA tournament selection committee. The researchers find substantial evidence of seeding bias in favor of virtually all major and mid-major conferences in selection and/or seeding, as well as evidence of bias toward majors over mid-majors. They also find substantial evidence of bias toward teams with some type of representation on the selection committee. Verifying these conclusions ourselves, we eliminated the use of seeding from our models.

3 Problem Statement

The models discussed in this paper serve two purposes: to construct brackets based on classification and to develop outcome probabilities for each game. Because of the sequential nature of the games played in the tournament, there are multiple ways to evaluate the models. The first and most naive method of evaluation is to compute the outcome of every potential tournament game and compute the accuracy on the results of the games actually played. Because there are 68 teams in the tournament, there are 2278 games that could possibly occur in the tournament, but only 67 of them will actually occur, meaning that most of these predictions go untested. Next, we can use our models to generate an actual bracket and evaluate its accuracy based on a commonly used points system (10 points each for correct first round predictions, 20 for second round, 40 for third round, etc.). In this system, the value of predictions is not weighted equally between the games. Most websites offering competitions to fill out the "best" bracket use this scoring system, but the system suffers from its somewhat arbitrary nature and a lack of statistical significance. The final evaluation method we can use is a log-loss based on outcome probabilities provided by our models:

$$\text{LogLoss}= \frac{-1}{n}\sum_{i=1}^{n} = [y_i log(\hat{y_i}) + (1-y_i) log(1-\hat{yi})]$$

where n is the number of games played, $\hat{y_i}$ is the predicted probability of team 1 beating team 2, and y_i is 1 if team 1 wins, 0 if team 2 wins.

A smaller log loss is better. For our purposes, games which are not played are ignored in the scoring. Play-in games are also ignored, so only the games among the final 64 teams are scored. The use of the logarithm provides extreme punishments for being both confident and wrong. In the worst possible case, a prediction that something is true when it is actually false will add an infinite value to the error score; to prevent this, predictions must be bounded away from the extremes by a small value. We can compare the LogLoss score of one algorithm to the score of another algorithms to evaluate relative performance.

Section 3.1 discusses the nature of the data used to train our models, and Section 3.2 deals with feature selection. In Section 3.3 we discuss the advantages to using particular techniques on for this problem, and section 3.4 details our own model based on random forests.

3.1 Data Source

Below are descriptions of all of the data files and statistics used in the models. The data is from Kaggle's Machine Learning Mania dataset.

3.1.1 Teams

This file identifies the different college teams present in the dataset. Each team has a 4 digit id number.

3.1.2 Seasons

This file identifies the different seasons included in the historical data, along with certain season-level properties. The file contains the following fields:

- "season" - the year in which the tournament was played
- "dayzero" - the date corresponding to daynum=0 during that season.
- "regionW/X/Y/Z" - the region which a team was assigned to

3.1.3 RegularSeasonCompactResults

This file identifies the game-by-game results for 32 seasons of historical data, from 1985 to 2015. Each year, it includes all games played from daynum 0 through 132 (which by definition is "Selection Sunday," the day that tournament pairings are announced). Each row in the file represents a single game played. The file contains the following fields:

- "season"
- "daynum"
- "wteam" - the id number of the team that won the game
- "wscore" - the number of points scored by the winning team
- "lteam" - the id number of the team that lost the game

- “lscore” - the number of points scored by the losing team
- “numot” - the number of overtime periods in the game, an integer 0 or higher
- “wloc” - the ”location” of the winning team

3.1.4 RegularSeasonDetailedResults

This file is a more detailed set of game results, covering seasons 2003-2016. This includes team-level total statistics for each game (total field goals attempted, offensive rebounds, etc.) The column names should be self-explanatory to basketball fans (as above, “w” or “l” refers to the winning or losing team):

- “wfgm” - field goals made
- “wfga” - field goals attempted
- “wfgm3” - three pointers made
- “wfga3” - three pointers attempted
- “wftm” - free throws made
- “wfta” - free throws attempted
- “wor” - offensive rebounds
- “wdr” - defensive rebounds
- “wast” - assists
- “wto” - turnovers
- “wstl” - steals
- “wblk” - blocks
- “wpf” - personal fouls

3.1.5 TourneyCompactResults

This file identifies the game-by-game NCAA tournament results for all seasons of historical data. The fields contained in this file are identical to those in RegularSeasonCompactResults.

3.1.6 TourneyDetailedResults

This file contains the more detailed results for tournament games from 2003 onward. The fields contained in this file are identical to those in RegularSeasonDetailedResults.

3.2 Data Construction

The dataset from Kaggle.com provided game-by-game statistics along with the results of each game. Due to the sequential nature of a basketball season, however, it is not possible to train a model using individual game statistics and game outcomes and then use this model to predict the outcome of future games. Before a game occurs, we have no game statistics with which to make a prediction, and after the game finishes and the stats are available, there is no use of making a prediction. Likewise, we cannot use average season statistics at the end of the year to train the outcomes of games that occurred earlier in the year.

Instead, the models are trained based on features that are rolling averages of each statistic up to, *but not including*, the current game, that is recomputed after each game. One inherent limitation of this approach is that earlier in the season, the rolling average of a statistics is based on a smaller sample size, and may therefore have less predictive power. Another key decision concerns the relevance of early-season games. As a team evolves over the course of 30 season games, how much applicability do early games really hold? There is no easy answer to this question. For our purposes, we limited the rolling average of features to only include the 15 most recent games.

The averages of 14 chosen statistics were calculated both for a team and for all the opponents they played up to that point in the season. Thus, in each matchup, we use Team A's averages, Team A's opponents' averages, Team B's averages, and Team B's opponents' averages, plus one additional feature for game location, for a total of 57 features per example. The statistics we ended up feeding into the model were the following:

- Points Per Game
- Field Goals Made Per Game
- Field Goals Attempted Per Game
- 3-Pointers Made Per Game
- 3-Pointers Attempted Per Game
- Free Throws Made Per Game
- Free Throws Attempted Per Game
- Offensive Rebounds Per Game
- Defensive Rebounds Per Game
- Assists Per Game
- Turnovers Per Game
- Steals Per Game

- Blocks Per Game
- Personal Fouls Per Game
- Game Location: Home or Away/Neutral (binary flag)

Note that the final feature, Game Location, is the only one that *is* known before a game occurs, and thus we can use the actual value. Also note that win-loss record (or win percentage), as well a team's seed, are not used at all. We verified experimentally that including wins and/or seeding negatively impacted the classification accuracy of our models. Thus, these statistics were deliberately excluded. Seeding is inherently biased because it is holistically determined by a human committee. Likewise, the wins statistic may be misleading because it does not take into account how close the games were, treating all games equally. Therefore, data on win totals can often be noisy. The worse results when including win percentage and seed in our model fit with our intuition that win percentage and seed would be biased, but also highly correlated with the other statistics listed above, and thus unnecessary to include in our model.

3.3 Techniques

Prior to implementing our own prediction method surrounding the random forests algorithm, we tested models based on several learning methods to both understand their performance and set a benchmark for our implementation. Using each technique, we trained our models on the classification problem of deciding a winner and loser for each matchup as well as the problem of determining the likelihood of each outcome.

3.3.1 Adaptive Boosting

The AdaBoost model was trained using 100 weak learners, each a decision tree based on selecting the features and classifier at each stage that provided the lowest error.

3.3.2 K-Nearest Neighbors

The K-Nearest Neighbors model was trained with K = 10 and with each neighbor's contribution to the prediction weighted by distance.

3.3.3 Naive Bayes

The Naive Bayes classifier was trained with the assumption that the likelihood of the features was Gaussian. The parameters of the model were estimated using maximum likelihood.

3.3.4 Neural Network

The Neural Network was trained on a network with two hidden layers, each with 30 nodes. The input layer had dimensionality equal to the number of features, and the output layer had dimensionality 1, outputting 0 or 1 as the class prediction.

3.3.5 Logistic Regression

Logistic regression was used in a binomial setting, with 0 or 1 output to predict the result of a game.

3.3.6 Support Vector Machine

The SVM classifier was trained using an RBF kernel, which gave better performance than linear, polynomial, and sigmoid kernels.

3.4 Random Forest

The Random Forest Classifier was trained using 300 decision trees, each using a randomly selected subset of the features, equal to the square root of the input dimensionality. For this problem, each tree used 8 random features from the 57 available.

A random forest is an ensemble learning method that operates by constructing a multitude of decision tress at training time, outputting the class that is the mode of results of the trees. While decision trees alone suffer from overfitting, random forests correct overfitting through bagging. Figure 2 shows an example of a basic decision tree for our model where f_{11} represents feature 1 of the first sample.

Figure 2: A single decision tree classifier

$$S = \begin{bmatrix} f_{11} & f_{21} & f_{31} & C_1 \\ \vdots & \vdots & \vdots & \vdots \\ f_{1N} & f_{2N} & f_{3N} & C_N \end{bmatrix}$$

To construct the ensemble of trees used in the random forest classifier, features are drawn randomly, with replacement, and added to a new tree. Figure 3 shows an example construction of the random forest. By decorrelating the individual trees, the random forest mitigates the impact of overfitting by any one tree.

Figure 3: An ensemble of random decision trees

$$S_1 = \begin{bmatrix} f_{112} & f_{212} & f_{312} & C_{12} \\ f_{115} & f_{215} & f_{315} & C_{15} \\ \vdots & \vdots & \vdots & \vdots \\ f_{135} & f_{235} & f_{335} & C_{35} \end{bmatrix} \qquad S_2 = \begin{bmatrix} f_{12} & f_{22} & f_{32} & C_{2} \\ f_{16} & f_{26} & f_{36} & C_{6} \\ \vdots & \vdots & \vdots & \vdots \\ f_{120} & f_{220} & f_{320} & C_{20} \end{bmatrix}$$

$$S_M = \begin{bmatrix} f_{14} & f_{24} & f_{34} & C_{4} \\ f_{19} & f_{29} & f_{39} & C_{9} \\ \vdots & \vdots & \vdots & \vdots \\ f_{112} & f_{212} & f_{312} & C_{12} \end{bmatrix}$$

The final step in the algorithm is to compute a prediction using each tree and to return the mode to get the forest's prediction.

4 Results

Upon training and testing each of our chosen learning methods, we found some techniques to be more successful than others at predicting tournament outcomes while our random forest implementation performed very well relative to the other algorithms. Section 4.1 contains the results for classification of winners and losers and section 4.2 evaluates our models based on their predicted win probabilities.

4.1 Classification

Our accuracy metric was computed by predicting the results of every potential game in the tournament and checking these predictions based on the outcomes of the actual matchups that occurred. Thus, the consequences of incorrectly predicting an early game are not carried through the rest of the tournament. The neural net performed the best in this category, achieving 79.4% accuracy on its predictions. The regression (76.2%), random forest (69.8%), and Bayes (69.8%) models also performed very well. SVM predicted with 68.3% accuracy and adaptive boosting predicted with

66.7% accuracy. K-nearest neighbors performed the worst, predicting 61.9% of the games correctly (Figure 4).

Figure 4:

Algorithm	Accuracy
AdaBoost	.667
KNN	.619
Bayes	.698
Neural Net	.794
Regression	.762
SVM	.683
Random Forest	.698

Next, based on the predictions made by each model, we constructed brackets to measure the performance of each algorithm in the setting of a bracket pool. Using this metric, incorrect predictions at the beginning of the tournament can be propagated through the entire tournament, resulting in poor scores - thus, we expected the models with high accuracy to perform similarly well by this measure. Note that while a decent measure of performance for a model (their intended function, after all), bracket scores are sensitive and heavily subject to the tumult of March Madness. That said, the random forest model performed the best, earning 900 points on its 2017 bracket predictions. The regression (670) and neural net (650) models also performed well. Bayes earned 610 points, K-nearest neighbors earned 600 points, SVM earned 570 points, and adaptive boosting earned 400 points (Figure 5).

Figure 5:

Algorithm	Bracket Score
AdaBoost	400
KNN	600
Bayes	610
Neural Net	650
Regression	670
SVM	570
Random Forest	900

4.2 Probabilities

To assess the probability predictions made by each model in an appropriate manner, we used the log loss metric so that we could account for confidence in each prediction e.g. a confident prediction that is wrong will be penalized more heavily. This metric is also the one we believe has the best indication of a model's power since it will be more consistent year-to-year because it is more resistant to unlikely results in individual matchups. The regression model performed the best with a log loss of .529. The neural net (.545), random forest (.578), and Bayes (.579) models also performed very well. The SVM model scored a log loss of .657 and K-nearest neighbors scored a log loss of .687. Adaptive boosting performed very poorly with a log loss of 1.261 (Figure 6).

Figure 6:

Algorithm	Log Loss
AdaBoost	1.261
KNN	.687
Bayes	.579
Neural Net	.545
Regression	.529
SVM	.657
Random Forest	.578

5 Conclusion

In this paper, we describe models for predicting NCAA tournament outcomes using a variety of machine learning techniques by using aggregated season data. Based on the results presented above, the regression (76.2% accuracy, 670 points, .529 log loss), random forest (69.8% accuracy, 900 points, .578 log loss), and neural net (79.4% accuracy, 650 points, .545 log loss) models performed the most successfully in predicting the 2017 tournament. AdaBoost and KNN performed rather poorly in terms of all three metrics, suffering from many predictions that were both wrong and highly confident. Interestingly, SVM performed decently in terms of accuracy but rather poorly in terms of log-loss. Upon further examination, SVM suffered from the opposite problem of AdaBoost and KNN; namely, the model had low confidence in its predictions in comparison with the other models, outputting win probabilities within a much narrower band closer to 0.5. Ultimately, machine learning models demonstrate greater overall accuracy in predicting NCAA tournament outcomes than the average human. However, it appears as though there is an upper limit to the success that can be achieved using the attributes that we used for training, and that luck plays a large part in outcome of a tournament in any given season.

6 Future Work

Because most of the models considered seemed to demonstrate similar levels of success, particularly with respect to log-loss, we believe that experimenting with different feature selection techniques rather than models in the future may lead to more significant improvement. There are many potential explanations as to why our prediction accuracy was not higher.

One potential flaw with the current features selected for our model is that there is no clear measure for strength of schedule (SOS) or consistency. Frequently used in sports, SOS is a measure of the average strength of a team's opponents. Although the average statistics of a team's opponent are available, it is unclear if they are influenced more by the team's skill level or the skill level of their opponents. Sophisticated and well-regarded Pythagorean team rankings available at Ken Pomeroy's `KenPom.com` could be used to construct a SOS metric. Consistency could potentially be quantified by season-long variance in individual statistics. Future work on our model could focus integrating SOS and consistency metrics to determine if test accuracy is increased.

In addition, no existing data directly takes into account intangible factors that may impact the outcome of basketball games, including the following: experience, leadership, "clutchness" (performance in high-pressure situations), luck, injuries, and the benefit of a superstar player. With respect to some of these intangibles, Pomeroy has developed metrics that attempt to quantify the luck and the experience level of a team. Incorporating player-level data (such as the season average statistics of individual players) in addition to team-level data could also help to predict the outcome of a game.

Another completely different approach to the basketball game prediction problem could be to view it as a time-series problem, using a recurrent neural network to model a team's progression throughout the season. Such a model could be able to quantify the impact of recent performance versus early-season performance on the outcome of a game.

Finally, predicting the outcome of basketball games presents an interesting dilemma of how to treat incorrectly predicted games. When classifying Iris species, for instance, a flower will always be of the same species no matter how many times it is examined. In the case of a basketball game, however, the result of a game in no way guarantees that the outcome will be the same if the matchup occurs again. Identifying similarities (or differences) between misclassified games could help determine if there are identifying characteristics to these examples, or if the outcome was simply highly unlikely. Constructing a metric to quantify the "unlikeliness" of the outcome of a game could help avoid overfitting to an unlikely training example. One way to quantify how unlikely the outcome of a game was could be to determine how much the statistics of the teams in a given game differed from their season averages. Perhaps, a model is not misguided and should not be penalized for incorrectly predicting to win a team that *should* have won.

References

[1] Kaggle, "March machine learning mania 2017." `https://www.kaggle.com/c/march-machine-learning-mania-2017`, 2017. Online; Accessed: 2017-04-04.

[2] M. J. Lopez and G. J. Matthews, "Building an ncaa men's basketball predictive model and quantifying its success," *Journal of Quantitative Analysis in Sports*, vol. 11, Jan 2015.

[3] L.-H. Yuan, A. Liu, A. Yeh, *et al.*, "A mixture-of-modelers approach to forecasting ncaa tournament outcomes," *Journal of Quantitative Analysis in Sports*, vol. 11, Jan 2015.

[4] A. Zimmermann, S. Moorthy, and Z. Shi, "Predicting college basketball match outcomes using machine learning techniques: some results and lessons learned," *CoRR*, vol. abs/1310.3607, 2013.

[5] B. J. Coleman, J. M. Dumond, and A. K. Lynch, "Evidence of bias in ncaa tournament selection and seeding," *Managerial and Decision Economics*, vol. 31, no. 7, p. 431–452, 2010.