Making March Less Mad - Predicting the NCAA Men's Basketball Tournament

Sreya Banerjee
University of Notre Dame
Notre Dame, Indiana
Sreya.Banerjee.9@nd.edu

Gabriel Wright
University of Notre Dame
Notre Dame, Indiana
Gabriel.S.Wright.142@nd.edu

Abstract

March madness is an annual collegiate basketball tournament where the top 64 teams from across America participate to win the national championship. For basketball enthusiasts and computer scientists alike, in essence, this becomes a binary classification task where, with sufficient historical data, we can try to predict the outcome of tournament. In this work, we implement and evaluate some common supervised machine learning approaches, such as both linear and kernel SVM, Gaussian Naive Bayes, and logistic regression, to predict the result of the 63 games in the tournament. Our data set includes results for 47,598 regular season games dating back to 2010, as while as team statistics for every team which played in those games in that year. Through a variety of methods we were able to consistently attain an accuracy of .76 on the testing set, which corresponded to an accuracy between .59 and .64 on the 2018 tournament.

1. Introduction

Given the TV ratings and the amount of money spent on gambling or put up as prizes every year for March Madness brackets, the NCAA men's basketball tournament is undoubtedly one of the biggest events in America and has generated interests of scientists worldwide. It has been an active area of research in fields as diverse as social psychology to statistics to computer science [4, 6, 8, 9, 11].

In this work, we implement and evaluate some common supervised machine learning approaches to predict the result of the 63 games within the tournament, based on almost a decade worth of basketball results. Formally, our problem can be defined as, given two data sets: one being a data set of the outcome of NCAA college men's basketball games from previous years, and the other containing individual statistics for each team, is it possible for machine learning algorithms to predict the results of the games in the tournament? Essentially, the task can be formulated as a binary classification problem with two possible outcomes of win or lose for each game. A "successful" model would be one that outperforms the null model where each game is picked based on tournament seeding.

For the classification task, we used common machine learning algorithms such as naive Bayes, support vector machines (SVM) [3] and logistic regression. We chose the data set [1] where the results of basketball matches each year is tabulated as csv file. Additionally, the data set [2] provide advanced statistics for each team in each year. We compiled data dating back to the 2010 season, which included results from almost 48,000 regular season basketball games.

For the tasks, we divide the data set into training and testing components. We tried a couple different ways of splitting the data, and it proved robust across those methods. Additionally, we applied our models to the 2018 tournament and analyzed those results.

2. Related Work

The idea of using machine learning for decision making in sports is not an uncommon phenomenon [2, 4, 10]. Previous work in this regard includes detecting goals through support vector machines [1]. Quite recently, Brooks *et al.* [2] developed a data-driven strategy to rank players in soccer based solely on the pass origins and destinations during a possession with the probability of a shot. Their method correctly guesses whether a possession ends in a shot from the pass locations alone. They used the 2012-2013 La Liga season as a dataset for their model. When it comes to volleyball, Van Haaren *et al.* [10] proposed a relational-learning based pattern discovery method based on spatial and temporal cues of games. For their data set they used both the men's and women's final match from the 2014 FIVB Volleyball World Championships and were able to discover interesting strategies for the game.

March madness is no exception. Earlier work in this regard primarily concentrated on building logistic regression based probability models using seed positions [8, 11],

[1] https://www.kaggle.com/c/march-machine-learning-mania-2017/data.

[2] https://www.Sports-Reference.com/

nearest-neighbour based approach [4] or specialized logistic regression/Markov chain models[5] for predicting the outcomes of matches. Schwertman *et al*. in their article, used the NCAA regional basketball tournament data to develop 11 simple linear regression and logistic regression models using seed position for predicting the probability of each of the 16 seeds winning the regional tournament and compared the empirical probabilities not only to the predicted probabilities of winning the regional tournament but also the predicted probabilities of each seed winning each contest. Their data consisted of only 600 basketball games played between 1985-1994. Kvam *et al*. uses the data from 1999 2000 through 2004 2005 seasons (378 games in total). Additionally, they compared their model with predictions from AP, ESPN, RPI, Seed,Sagarin, KG, and Sheridan at the 0.05 level of signicance using one-tailed version of McNemars test [7]. However, none of the methods discussed above use classification accuracy for the tournament data as an evaluation metric.

Our work primarily differs from them in the following aspects:

1. A large dataset comprising of $47,598$ matches from 2010-2018.

2. More than one attributes or features that are most predictive of the outcomes of game from knowledge/experience.

We believe having access to a large dataset and predictive attributes are crucial in designing any machine learning based model.

3. Data Acquisition and Pre-processing

In order to build our models to predict the NCAA tournament we took data from two sources: Kaggle and Sports-Reference.com.

At Sports-Reference.com, statistical data is provided for each Division 1 basketball team for a given season. We chose to use team statistics dating back to 2010. The motivation for using more recent data was both to save time compiling the data and because the game of basketball is constantly evolving, resulting in more recent statistical data being more predictive of current results. For example, the recent trend in professional basketball has moved away from dominant centers and has put a premium on dominant three point shooters. Trends like this may show up in team statistics for successful teams.

Kaggle, the self proclaimed "Home of Data Science & Machine Learning", is an online host for data science competitions on a wide variety of topics. As part of these competitions they freely provide a large amount of data for users to work with. For the past several years Kaggle has hosted a March Madness competition, and this year is no different. Along with the competition, they have published a large amount of college basketball data for public use. Of the files they have published, we only concerned ourselves with four - "RegularSeasonCompactResults" (RSCR), "TeamSpellings" (TS), "NCAATourneySlots" (Slots) and "NCAATourneySeeds" (Seeds).

In the RSCR file, results are provided for every Division 1 basketball game dating back to 1985 (a total of 150k games). One quirk of the Kaggle data is that teams have been mapped to ID numbers (e.g. Notre Dame is team number 1323), making it incompatible with the data files from Basketabll-Reference, leading us to need the TS file. In TS, multiple possible spellings of each team are given with the corresponding team ID, with the intent of allowing external data to be mapped to the provided Kaggle data. We used this file to map the Kaggle data to the Sports-Reference data, resulting in a file containing results for $47,598$ games (dating back to the 2010 season) with statistics for each team that played in each game.

Sports-Reference provides numerous attributes (>50) for each team for a given year. Our final data set pruned these down to 28 attributes (14 for each team in a given game). These attributes tried to cover a variety of aspects for each team, and contained broad measures such as Wins and Simple Rating System, as well as more detailed stats such as steal percentage or assist percentage. Some of the attributes, such as Offensive Rating, we defined for both the team in the game as well as the opponents of the team in the game. For example, if Notre Dame is playing in a game we are trying to predict, we considered the season wide Offensive Rating as well as the Offensive Rating of Notre Dame's opponents in games against Notre Dame. This can be viewed as a proxy for "Defensive Rating" for Notre Dame. A full list of the attributes considered in our models can be seen in Appendix A.

One final data pre-processing step that was used in some of our models (mainly the logistic regression models) was attribute combination by merging attributes that were related so each other. For example, if team A and team B are playing in a game, an attribute which is defined for both were merged into one (i.e. 'Wins for A' and 'Wins for B' were merged into 'Win Difference Between A and B'). In cases (such as Offensive Rating described above) where an attribute was defined for team A, opponents of A, team B, and opponents of B, all four attributes were combined into one for some runs. This was done to make our model parameters more digestable to a user of the model. While we did not exhaust all possible options of attribute combination/separation, the results we do have seem to indicate combinations do not have an effect on the model results.

The final two data sources we used were the Seeds and the Slots data sets provided by Kaggle. These enabled us to generate the dynamic bracket structure of the 2018 NCAA

Tournament, allowing the teams which won an early round game to be placed in the correct later round game.

4. Methodology

4.1. Models/Results

We used a divide and conquer approach to modeling, each taking our final data set and creating a model from it. As per our initial plan, we have successfully implemented SVM, Logistic Regression and naive Bayes.

4.1.1 Support Vector Machine

Since our data is numeric and high-dimensional, SVM [3] was a natural choice as it has been found to be extremely efficient in high-dimensional spaces for large-scale classification problems. SVMs use a subset of training points in the decision function (called support vectors). As a consequence, it has been found to be memory efficient and has faster execution times if the data is normalized. For analysis, we assumed our data to be linearly separable and used a linear SVM kernel. We also trained our data using a non-linear SVM kernel to find out whether it affects classification accuracy. We normalized our data using min-max normalization.

A SVM model, with a set of labelled training data tries to find an optimal hyperplane for classifying new samples based on some constraints.

Given a training dataset, D = (x_i,y_i) of size m with x_i = (x_1, x_2,.., x_n), an n-dimensional feature/attribute vector and label, y_i= -1 or +1, formally the SVM classifier can be defined as a quadratic optimization problem solving the following equation:

$$min \|w\|^2 \text{ s.t } y_i(w^T x_i + b) \geq 1 \text{ for all } i \quad (1)$$

where w = (w_1, w_2, ...,w_n) is a weight vector and b is bias.

An important consideration when designing a SVM model is the parameter C that dictates the trade-off between having a wide margin and correctly classifying training data.

$$min \|w\|^2 + C \sum_{1}^{m} \xi_i \text{ s.t } y_i(w^T x_i + b) \geq (1 - \xi_i) \text{ for all } i \quad (2)$$

Needless to say, a higher value of C implies a lesser number of mis-classified training samples and is prone to overfitting.

Experimental settings: We performed three quantitative experiments to measure the goodness of fit for our SVM model.

The first two experiments consisted of finding an optimal C value based on a fixed training and testing set; such that the model has higher prediction accuracy on unseen data without losing its generalization capability. This can be achieved in two ways: manually setting up the value of C for linear kernel and using grid search for parameter optimization for non-linear kernel.

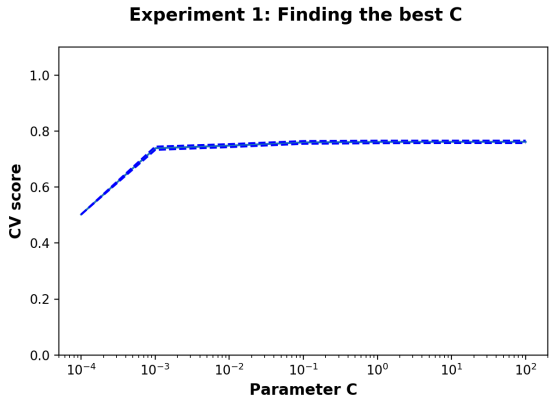

Figure 1. Results of tuning C parameter on SVM model with linear kernel. The accuracy on changing the value of parameter C plateaus almost at C =.001. For our experiments we use C =1.0.

For the second method, we used k-fold cross-validation technique as a method to improve our model. Ideally, cross-validation should improve classification accuracy as it divides the data into number of folds specified (suppose, k) and trains on (k-1) sets while validating or testing the model on the other set, k times. For our experiments, we divided our data into 3 to 10 folds randomly.

Evaluation results: Figure.1 details the result of tuning the parameter C on linear SVM keeping the training and testing data fixed. The result was quite surprising. The classification accuracy seems to plateau somewhere around 0.75 for all values of C >= 0.001. The accuracy never reaches 0.80. For our experiments, we used the value of C = 1.0. With this, the accuracy on 67 games of this year's tournament was 0.6417910447761194

Figure.2 details the result of tuning the parameters C and γ keeping the training and testing data fixed for the non-linear kernel.

With grid search, the optimal parameters for non-linear SVM was found to be C =10.0 and γ = 0.1 and the non-linear kernel was found to be radial basis function. With this, the accuracy on 67 games of this year's tournament was 0.6119402985074627.

The results of third experiment are consolidated in Table 1. None of the cross validation setup with randomly partitioned data performed relatively better than the others by a wide margin.

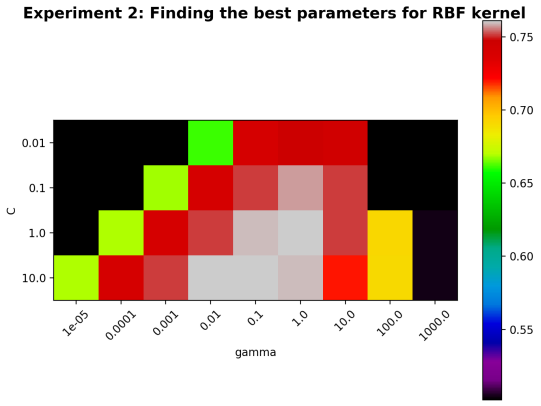

Figure 2. Results of tuning C and γ parameter on non-linear SVM model via grid search. The best estimator was found to be C =10.0 and γ = 0.1.

Method	Classification Accuracy
with 20% hold-out training data	0.76
with 3-fold cross validation	0.76(+/- 0.01)
with 4-fold cross validation	0.76(+/- 0.01)
with 5-fold cross validation	0.76(+/- 0.02)
with 6-fold cross validation	0.76(+/- 0.01)
with 7-fold cross validation	0.76(+/- 0.01)
with 8-fold cross validation	0.76(+/- 0.01)
with 9-fold cross validation	0.76(+/- 0.01)
with 10-fold cross validation	0.76(+/- 0.02)

Table 1. Results of cross-validation on SVM model. Classification accuracy for the basketball data does not vary with cross validation.

4.1.2 Gaussian Naive Bayes

Another common method that is often used for supervised classification tasks is naive Bayes. Naive Bayes is a very simple probabilistic classification algorithm that makes strong assumptions about the independence of each attributes of data and is based on the classical Bayes theorem. Nevertheless, it has been shown to be effective in a large number of problem domains.

Naive Bayes calculates the posterior probability of an observation v belonging to a class C_k, $p(C_k|v)$ by combining the likelihood of observation v belonging to the class C_k, $p(v|C_k)$ and prior probabilities of C_k, $p(C_k)$.

$$p(C_k|v) \propto p(v|C_k)p(C_k) \tag{3}$$

Gaussian naive Bayes extends the concepts of simple naive Bayes to that of real-valued continuous attributes. For example, suppose x is a real-valued attribute, μ_k represents the mean and σ_k^2 the variance of x associated with the the class, C_k. Then, for Gaussian Naive Bayes, the likelihood or probability of some observation, v given the class, C_k, $p(x = v|C_k)$ can be computed by plugging v in the equation for Gaussian distribution parameterized by μ_k and σ_k^2:

$$p(x = v|C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{\frac{-(x-\mu_k)^2}{2\sigma_k^2}} \tag{4}$$

Now if x = (x_1,x_2,..,x_n), is a vector representing n independent features, the likelihood would be given by $\prod_{i=1}^{n} p(x_i|C_k)$ and the Bayes classifier for class, $\hat{y} = C_k$ for some k is defined as:

$$\hat{y} = argmax_{k\in\{1,2,3...\}} p(C_k) \prod_{i=1}^{n} p(x_i|C_k) \tag{5}$$

Experimental settings & Results: For the naive Bayes, we ran our model 10 times and took an average of the classification accuracy score. For our randomly chosen set, the accuracy came out to be 0.74. The classification accuracy was 0.72 for allthe basketball matches for 2018. The accuracy on the test tournament data was found out to be 0.597.

4.1.3 Logistic Regression

Predicting the outcome of games can be formatted as a binary classification problem. For example, given team statistics for Team 1 and Team 2 (order arbitrarily decided), does Team 1 win the game? A very common approach to solving such a problem is a logistic regression model. Logistic regression assigns weights to each input attribute and outputs a value between 0 and 1. This output can be viewed as a probability of success relative to the target variable (in this case Team 1 winning), with any probability >0.5 being considered a "success". This is how we formatted the classification problem.

Experimental settings: For this model we did some attribute combination to make the results more intuitive. For each game in our data set, a number of our attributes (SRS, eFG%, TS%, and ORtg) are defined four times:

1. How Team 1 ranks in these categories
2. How teams who have played against Team 1 did against Team 1 in these categories
3. How Team 2 ranks in these categories
4. How teams who have played against Team 2 did against Team 2 in these categories

Items 2 and 4 in the list above can be viewed as defensive statistics for Team 1 and Team 2 respectively. Therefore, it makes sense to combine the four versions of each attribute into one attribute. This was done as follows:

2018 Tournament Results for A Logistic Regression Model

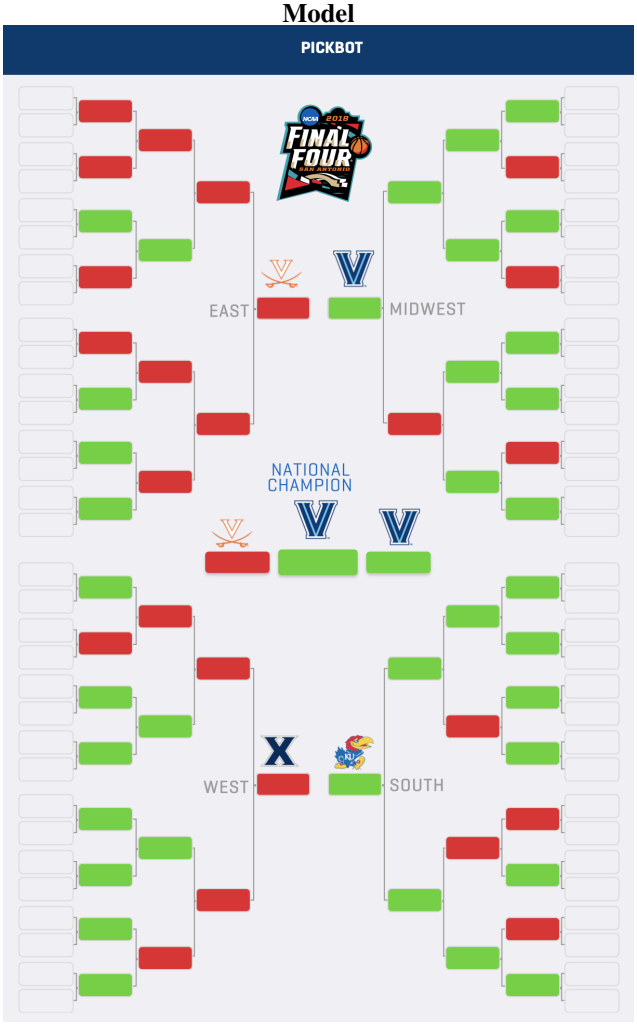

Figure 3. The results from an instance of Model 1 for the 2018 tournament.

$$(A_1 - A_2) - (A_3 - A_4)$$

Where the first set of parentheses represents how Team 1 does overall for a given statistic, and the second set does the same for Team 2.

A more minor attribute combination was also done. We combined attributes that were defined for both teams so as to not weight an attribute for a given team higher than that for another. Explicitly, if attribute X is defined for both team A and team B, X_A and X_B were combined into one attribute to be sure that they were weighted equally. This is a logical choice, as which team was "team A" or "team B" was defined at random, and should contain no information about which is more important.

Evaluation results: Each combination of parameters for the logistic regression model was run ten times with approximately 70% of the games randomly chosen to be in the training set for each iteration, and the remaining 30% placed in the testing set. The average accuracy on testing set across the 10 iterations was about 0.76 regardless of what attributes were input into the model. This result conforms to the classification accuracy achieved using SVM.

Attribute choice seemed to have very little effect on the model. Numerous feature combinations were input into the model with a negligible effect on the model's testing set accuracy. A few of the specific choices are outlined below:

- **Model 1:** One of the first models run with logistic regression contained the following attributes: $Win_A - Win_B$, $SRS_A - SRS_B$, $TS_A - OppTS_A$, $TS_B - OppTS_B$, $ORtg_A - OppORtg_A$, $ORtg_B - OppORtg_B$, $eFG_A - OppeFG_A$, $eFG_B - OppeFG_B$. This model had an average accuracy of .762 for ten iterations on the randomly chosen testing set, as well as .612 for ten iterations on the 2018 tournament. Visual results for this model can be seen in Figure 3.

- **Model 2:** The attributes from above were then pruned down to only include $Win_A - Win_B$, $SRS_A - SRS_B$, and $(ORtg_A - OppORtg_A)$ - $(ORtg_B - OppORtg_B)$. This model had an average accuracy of .761 for ten iterations on the randomly chosen testing set, as well as .610 for ten iterations on the 2018 tournament.

- **Model 3:** A third model included all attributes listed in Appendix A. Attributes that were combined with their related features include wins and SRS. Offensive Rating, True Shooting, and Effective Field Goal were combined with their opponent's counterpart for each team individually. All of the rest of the attributes were input individually for each team. This model had an average accuracy of .760 for ten iterations on the randomly chosen testing set, as well as .619 for ten iterations on the 2018 tournament.

One final model we implemented aimed at capturing the "Madness" within March Madness. In instances where the model predicted the games would be close (when no team had a probability of winning greater than a certain threshold), the model used weighted random chance to determine the winner. The features used in this model are the same as those in the first bullet point above, and results for different thresholds are given below:

- **1**: When all games are subjected to weighted random chance, the model attains an average accuracy of .53, with a maximum accuracy of .64 and a minimum of .42 over 10 iterations.

- **.75**: When weighted random change is applied to games where the probability of one team winning are

less than .75, the model attains an average accuracy of .50, with a maximum accuracy of .61 and a minimum of .40 over 10 iterations.

- **.6**: When weighted random change is applied to games where the probability of one team winning are less than .6, the model attains an average accuracy of .52, with a maximum accuracy of .61 and a minimum of .42 over 10 iterations.

- **.55**: When weighted random change is applied to games where the probability of one team winning are less than .55, the model attains an average accuracy of .52, with a maximum accuracy of .58 and a minimum of .40 over 10 iterations.

It is clear from our results that inserting this randomness into the model is not a beneficial addition.

5. Discussion and Conclusion

For both logistic regression and SVM we were able to consistently get an average accuracy of 0.76 on our test set and just over 0.6 on the 2018 tournament data. Gaussian Naive Bayes had a slightly lower accuracy, attaining 0.74 on the training data with 0.597 on the tournament. Compared to prediction based on seeding where the accuracy was 0.556, our models performed relatively better. This consistency in accuracy persisted regardless of which parameters were input into the SVM and logistic regression models. This suggests that the predictive information gained from the parameter additions between Model 2 and Model 3 in the logistic regression is minimal. Additionally, the model accuracy proved robust to changes in how the data was split into training and testing sets.

At first glance the difference in accuracy between the tournament data and the regular season testing data may seem alarming, however, the drop is to be expected. This is clearly illustrated in Figure 3. In the "East" section of the bracket (the top left portion), our model predicted the University of Virginia to advance all the way to the national championship game. However, because of an unprecedented upset by the University of Maryland, Baltimore County, in the first round of the tournament, the next four games were by default incorrect. Because of this dependent nature of the games in a bracket-style tournament, an early mistake can result in a string of incorrect picks, resulting in the expected accuracy to be lower than in a series of independent games. Additionally, it may stand out that the tournament accuracy in general varied between .59 and .64 when the testing accuracy stayed so consistent. This can be attributed to the large difference in sample sizes between the two sets (67 games vs 14,000 games). Also, the random selection of the training set in the logistic regression model allows the model coefficients to vary slightly between runs, which alters the predictions on the small tournament set enough to see a change in accuracy.

6. Future Work

Future work into this problem could go a number of different ways. We believe one of the more promising routes to pursue would be to train the model on past tournament games. The nature of a "win or go home" game may result in certain features being more important in these scenarios than others. Because our model is trained on regular season data, we might be missing or undervaluing these features. However, this route would also create a data scarcity problem. If the model were to be built on the results of tournaments from 2010 to 2017 (8 seasons), the model would only have 504 games (63*8) to train and test on, as opposed to the 47,598 we used for the same time period.

Other future work could be implementing any one of the variety of available models we did not. Two commonly used possibilities include Decision Trees or Neural Networks.

References

[1] N. Ancona, G. Cicirelli, A. Branca, and A. Distante. Goal detection in football by using support vector machines for classification. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 1, pages 611–616. IEEE, 2001.

[2] J. Brooks, M. Kerr, and J. Guttag. Developing a data-driven player ranking in soccer using predictive model weights. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 49–55. ACM, 2016.

[3] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[4] A. Hoegh, M. Carzolio, I. Crandell, X. Hu, L. Roberts, Y. Song, and S. C. Leman. Nearest-neighbor matchup effects: accounting for team matchups for predicting march madness. *Journal of Quantitative Analysis in Sports*, 11(1):29–37, 2015.

[5] P. Kvam and J. S. Sokol. A logistic regression/markov chain model for ncaa basketball. *Naval Research Logistics (NrL)*, 53(8):788–803, 2006.

[6] S. M. McCrea and E. R. Hirt. Match madness: Probability matching in prediction of the ncaa basketball tournament. *Journal of Applied Social Psychology*, 39(12):2809–2839, 2009.

[7] Q. McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[8] N. C. Schwertman, K. L. Schenk, and B. C. Holbrook. More probability models for the ncaa regional basketball tournaments. *The American Statistician*, 50(1):34–38, 1996.

[9] T. Smith and N. C. Schwertman. Can the ncaa basketball tournament seeding be used to predict margin of victory? *The American Statistician*, 53(2):94–98, 1999.

[10] J. Van Haaren, H. Ben Shitrit, J. Davis, and P. Fua. Analyzing volleyball match data from the 2014 world championships using machine learning techniques. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 627–634. ACM, 2016.

[11] B. T. West. A simple and flexible rating method for predicting success in the ncaa basketball tournament. *Journal of Quantitative Analysis in Sports*, 2(3), 2006.

A. Full List of Attributes

The following 14 attributes were defined for each team in each of our 47,598 games in our data set.

1. Wins
2. Simple Rating System
3. Offensive Rating
4. True Shooting Percentage
5. Effective Field Goal Percentage
6. Opponents Offensive Rating
7. Opponents True Shooting Percentage
8. Opponents Effective Field Goal Percentage
9. Pace
10. True Rebound Percentage
11. Assist Percentage
12. Turn Over Percentage
13. Steal Percentage
14. Block Percentage