In Pursuit of Perfection: An Ensemble Method for Predicting March Madness Match-Up Probabilities

Sara Stoudt* Loren Santana† Ben Baumer‡

Abstract

In pursuit of the perfect March Madness bracket we aimed to determine the most effective model and the most relevant data to predict match-up probabilities. We present an ensemble method of machine learning techniques. The predictions made by a support vector machine, a Naive Bayes classifier, a k nearest neighbors method, a decision tree, random forests, and an artificial neural network are combined using logistic regression to determine final matchup probabilities. These machine learning techniques are trained on historical team and tournament data. We discuss the performance of our method in the two stages of Kaggle's March Machine Learning Madness competition and compare it to the performance of the winning team, One Shining MGF (OSMGF). We also assess our performance using a generalizable simulation-based technique and in this way compare our performance to that of OSMGF and the performance of predictions based on seed.

Key Words: ensemble method, NCAAB, machine learning

1. Introduction

Every March, as the NCAA Men's basketball tournament approaches, millions of fans fill out a bracket predicting which teams they think have what it takes to be champions. Along with bragging rights, competitions for correct brackets can lead to monetary rewards (Dizikes, 2014). This year the stakes were raised by two parties: Warren Buffett, who offered a billion dollars for a perfect bracket; and Kaggle (a website that hosts predictive modeling competitions), who teamed up with Intel to give a monetary award to the top team in its March Machine Learning Mania competition. Most fans choose their bracket based on their intuition, but what kind of statistical models can be used to predict basketball game wins, and how accurate can predictions based on these models get?

In this paper we present an ensemble method of machine learning techniques for predicting NCAA basketball games using external team data. We discuss our approach's performance in the Kaggle competition and compare it with the performance of the winning team, *One Shining MGF*. We evaluate our predictions using a generalizable simulation-based technique [1].

1.1 Previous Work

Many have attempted to predict the outcome of NCAA games using different combinations of data inputs and modeling techniques, but no consensus has been reached on a standard or "best" technique. Smith and Schwertman (1999) used seed to predict margin of victory, and Carlin (1996) assessed the predictive performance of team strength and regular season point spread.

In general, predicting the outcomes of college basketball games is more challenging than for professional games due to the greater variability in team performance from season

*Smith College
†University of Utah
‡Smith College

[1] The idea for this technique came from *One Shining MGF*.

to season. New players are introduced on a yearly basis, and team dynamics change much more frequently. Measuring strength in team over various seasons will also depend on the strength of schedule in each season.

These challenges placed on NCAA basketball modeling can partially be addressed by machine learning, since these algorithms can identify recurring trends that remain obscure to a sports fan or even expert (Schumaker, Solieman, and Chen, 2010). Dean Oliver identified the "Four Factors"—shooting, turnovers, rebounding, and free throws—that best quantify success in college basketball and optimize predictive modeling accuracy (Oliver, 2004). Zimmermann, Moorthy, and Shi (2013) used the Four Factors along with adjusted efficiencies to predict match outcomes for the 2008–2009 through 2012–2013 seasons. They focused primarily on decision trees, artificial neural networks, Naive Bayes, rule learners, and random forests. They identified neural networks and Naive Bayes as the best performers in their experiment. Further, they concluded that there is an upper threshold on machine learning predictive accuracy of 75%.

It is clear that an effective prediction strategy for NCAA games requires a blend of informative data inputs and machine learning techniques that perform best given the training data. What may be considered a favorite machine learning technique may have poor performance if trained on irrelevant data. Prediction becomes even more difficult as one must both optimize the variables to train the model on and the model itself.

1.2 Objectives

Our goal was to identify key factors in predicting NCAA tournament wins and to find a model that would perform well in the Kaggle competition.
The Kaggle competition had two stages:

1. Predict outcomes of the past five NCAA Tournaments
2. Predict outcomes of the 2014 NCAA Tournament

For each stage we submitted a list $\hat{y}$ of probabilities (values between 0 and 1) that each team in the tournament would defeat every other team in the tournament, regardless of whether this match-up actually occurs. For this year, this was $m = 2278$ predictions [2]. We were judged based on the log-loss $L(y|\hat{y})$, or the predictive binomial deviance, of the games that actually occurred:

$$L(y|\hat{y}) = -\frac{1}{n}\sum_{i=1}^{n}[y_i \cdot \log(\hat{y}_i) + (1-y_i)\cdot \log(1-\hat{y}_i)],$$

where n is the actual number of games played in the tournament (67), y_i is the actual binary outcome of each game, and $\hat{y}_i$ is the corresponding submitted probability [3]. If the opponents in game i are teams A and B, then $\hat{y}_i(A,B)$ is the predicted probability that team A beats team B, and $\hat{y}_i(B,A) = 1 - \hat{y}_i(A,B)$.

The goal is to find a set of predictions $\hat{y}$ that minimizes $L(y|\hat{y})$ for the unknown outcome vector y. This scoring method heavily penalizes being simultaneously confident and wrong. If we say Team A will beat Team B with probability 1 and Team B wins, then our log-loss is infinite (although Kaggle bounded the log-loss function to avoid infinite scores). At the same time, it is important to balance between being too conservative and too confident. If we are too conservative (e.g. $\hat{y} \approx 0.5$), then we will never accrue points, but if we are too confident, we make ourselves vulnerable to huge losses.

[2] Note that in addition to the canonical 64 team tournament, there are now 4 play-in games. This brings the total number of potential matchups to $\binom{68}{2} = \binom{n+1}{2} = 2278$.

[3] Note here that i ranges over the $n = 67$ values of y_i, not the $m = 2278$ values of $\hat{y}_i$.

1.2.1 Benchmarks

Kaggle provided benchmarks to surpass for each stage of the competition. These relatively simple, public predictive models were informative.

Stage One

- Seed Benchmark: uses only the teams' tournaments seeds (ranging from 1 (best) to 16 (worst)). For each game i, suppose that team A is stronger than team B, and let $s_i(A), s_i(B)$ be the corresponding seeds. Then,

$$\hat{y}_i(A,B) = 0.50 + 0.03 \cdot (s_i(B) - s_i(A)).$$

  The seed benchmark can be derived by fitting an OLS regression model to the difference in seeds in the past tournaments.

- Rating Percentage Index (RPI) Benchmark: transforms the RPI, a measurement of team strength $rpi(A)$ with a higher number being better, into a rating $R(A)$ and then uses the difference in two teams' ratings $R_i(A) - R_i(B)$ to determine the winning percentage.

$$R(A) = 100 - 4 \cdot \log(rpi(A) + 1) - rpi(A)/22$$
$$\hat{y}_i(A,B) = 1/(1 + 10^{-(R_i(A) - R_i(B))/15})$$

- Chess Metric:

  1. Assign each team the same rating of 50.
  2. $GameScore = 1/(1 + 10^{-PointDifferential/15})$
  3. $RatingDiff = -15 \cdot \log_{10}(1/GameScore - 1)$
  4. Add $RatingDiff$ to the team's rating.
  5. Recalibrate after each iteration so that the average rating is 50.
  6. Iterate to converge to a stable solution.

Stage Two At the beginning of Stage Two, all teams submitted their lists of predicted probabilities $\hat{y}$. As the 2014 Tournament unfolded, entries in the outcome vector y were revealed. Let $y^* \in \{0,1\}^n$ be the outcome vector corresponding to the actual 2014 Tournament.

- All Zeros Benchmark: This is the benchmark where one chooses a zero probability for each game ($\hat{y}_0 = 0$). One will not lose infinite points as Kaggle bounds the log-loss function away from infinity, but one will lose the maximum number of points for each wrong prediction, and gain the maximum number of points for each correct prediction. This number turned out to be $L(y^*, \hat{y}_0) = 19.19$—this is the worst score that you could get in the competition.

- All 0.5 Benchmark: This is the "coin flip" benchmark where one chooses $\hat{y}_{0.5} = 0.5$ for each game. One will lose the same amount of points on each game ($\ln(0.5) \approx 0.693$). Thus, $L(y^*, \hat{y}_{0.5}) = 0.693$.

j	Model	$\hat{\beta}_j$	95% CI for β_j	$\exp(\hat{\beta}_j)$	95% CI for $\exp(\beta_j)$
0	Intercept	-4.41	[-5.37, -3.45]	0.01	[0.00, 0.03]
1	SVM	0.45	[-0.39, 1.29]	1.57	[0.68, 3.63]
2	Naive Bayes	0.54	[0.01, 1.07]	1.72	[1.01, 2.92]
3	KNN	1.35	[0.38, 2.32]	3.86	[1.46, 10.18]
4	Decision Tree	5.91	[5.27, 6.55]	368.71	[194.42, 699.24]
5	Random Forest	1.40	[0.55, 2.25]	4.06	[1.73, 9.49]
6	ANN	-1.24	[-1.81, -0.67]	0.29	[0.16, 0.051]

Table 1: Coefficients from Ensemble Logistic Regression Model. Note the large coefficient for the decision tree model.

- Seed Benchmark: Denoting the seed benchmark predictions by $\hat{y}_s$, we had $L(y^*, \hat{y}_s) = 0.600$.

Additionally, the mean score $L(y^*, \hat{y})$ over all entries $\hat{y}$ was 0.576, while the median score was 0.566. *One Shining MGF*'s winning score was 0.529.

2. Model

Our prediction model was an ensemble method of machine learning predictions based on team data retrieved from the NCAA website. In the following sections we outline our approach.

2.1 Ensemble Method

Ensemble methods are commonly used in classification problems. These methods allow predictions based on different models to be combined, often using a type of voting schema, to determine one final prediction. The benefit to using an ensemble method is that we can leverage the information gleaned from different methods, combining the benefits from each method while hopefully negating or minimizing the flaws in each. Ensemble methods have been used to combine predictions from different machine learning techniques in order to increase stability and to more accurately classify test cases (Dietterich, 2000). The combination of methods can also reduce the variance of the predictions as they are less sensitive to irregularities in a training set (Valentini, 2003).

The most basic ensemble method is to average the results from a set of predictions. In this way, an ensemble method can avoid extreme predictions. Ideally, we want to assign optimal weights to each group of predictions.

In order to ensure that our predictions were in $(0,1)$, we fit a logistic regression model where each explanatory variable is a prediction from one machine learning technique, and the response is the binary outcome.

Our model takes the form:

$$y_i(A,B) = \frac{1}{1+e^{-\left(\beta_0+\beta_1\cdot\hat{y}_i^{(1)}(A,B)+\cdots+\beta_6\cdot\hat{y}_i^{(6)}(A,B)+\varepsilon_i\right)}},$$

where each $\hat{y}_i^{(j)}(A,B)$ is a prediction from machine learning technique j for the probability that team A beats team B in game i. The fitted coefficients $\hat{\beta}_j$'s are the optimal weights yielded by the logistic regression model. These estimates and 95% confidence intervals for them are shown in Table 1.

Team A	Team B	$\hat{y}_i$	$\hat{y}_i^{(1)}$	$\hat{y}_i^{(2)}$	$\hat{y}_i^{(3)}$	$\hat{y}_i^{(4)}$	$\hat{y}_i^{(5)}$	$\hat{y}_i^{(6)}$
Dayton	Ohio State	0.73	0.51	0.13	0.71	0.71	0.19	0.27
Stephen F. Austin	VCU	0.96	0.94	0.56	1.00	0.87	0.54	0.27

Table 2: Example of Model Predictions for Dayton vs. Ohio State and Stephen F. Austin vs. Virginia Commonwealth University

For example, consider the games between Dayton and Ohio State and also Stephen F. Austin and VCU (which actually occurred in the Round of 64). The final and component predictions generated by our model are shown in Table 2. We can see that we were more confident that Stephen F. Austin was going to beat VCU than we were about Dayton beating Ohio State. This is due to the difference in the decision tree prediction ($\hat{y}_i^{(j)}$) and the very high confidence in the SVM and KNN predictions.

2.2 Data

Each machine learning technique was trained on historic team data from the past five seasons of NCAA tournament games. Let $X = (x_1, x_2, \ldots, x_p)$ be the matrix of data inputs, where each row corresponds to one game and each column records the difference in a single statistic x_ℓ, for $\ell = 1, \ldots, p$, over the course of the season for the two teams in that game. Then each of the $\hat{y}^{(j)}$ was trained on input X.

Kaggle provided regular season wins, losses, point differences, dates, and game locations (home/away) as well as tournament wins, losses, point differences, seeds, and dates of games. We collected additional team data from the NCAA (2014) and from a team rankings website (Greenfield, 2014). Extra data for seasons earlier than the past five years was not accessible via our web scraping methods, and some teams had missing data for some of the variables. This limited our training set, which required complete cases, and our use of all of the data provided by Kaggle, as we could only use the subset of the historical data provided that corresponded to the time period of data that we collected.

Additional Data Collected

Rebound Margin	Assists Per Game
Blocked Shots Per Game	Difference in Steals Per Game
Turnover Margin	Assist-Turnover Ratio
Field Goal Percentage	Field Goal Percentage Defense
Three Point Field Goals Per Game	Three Point Field Goal Percentage
Free-Throw Percentage	Team Ratings Power Index (RPI)
Difference in Conference RPI	

2.2.1 Ranking

We included two other factors in our model, team and conference rankings, using the PageRank algorithm. PageRank was initially used to improve the efficiency of search queries on the Internet (Page, Brin, Motwani, and Winograd, 1999).

Briefly, webpages are represented by nodes in a graph with directed edges connecting nodes if a link to one website exists on another. Thus, if webpage A has a web link that directs the user to webpage B, the associated graph has a directed edge from node A to node B. The edges are weighted with respect to how many incoming edges a node possesses. These ideas can be further extended to other fields, in particular, to sports.

Ranking basketball teams is analogous to ranking webpages. The associated graph would have nodes representing basketball teams and directed edges between teams that have played each other with the edge directed towards the winning team (Govan, Meyer, and Albright, 2008). The edges are weighted based on point differentials during regular season games.

The weighted adjacency matrix is then adapted by dividing each value in the matrix by the sum of every element in the matrix (sum of point differentials for every game) to make a transition matrix for a Markov process. By multiplying the transition matrix and a starting vector, with the number of rows equal to the number of columns in the transition matrix and with each entry being the reciprocal of the number of columns, one can repeat the multiplication by the transition matrix to produce a stationary result that represents the steady-state value of each row (team). These values can then be ranked to create a team ranking.

In this way the teams are ranked based on the total weights on incoming edges. A dominant team will have many edges entering its node, with few edges leaving. The edge weights will contribute to the magnitude of their dominance. The resulting team and conference rankings were used as inputs in our model.

2.3 Machine Learning

For reasons illustrated above, we wanted to use many different machine learning techniques and leverage them against one another so that we would gain the benefits from each and hopefully mitigate the weaknesses of each. We used Support Vector Machines, Naive Bayes, k-Nearest Neighbor, Decision Trees, Random Forests, and Artificial Neural Networks. In what follows we give a brief overview of these techniques. Readers seeking further clarity should consult Tan, Steinbach, and Kumar (2006) or Hastie, Tibshirani, and Friedman (2009).

2.3.1 Support Vector Machines

Each game in the training set X has a series of characteristics based on the difference in team statistics and a label y_i (win or loss). These can be plotted in a p-dimensional Euclidean space. The support vector machine then finds the hyperplanes $\mathbf{w} \cdot X - b = 1$ and $\mathbf{w} \cdot X = 0$ with the greatest distance between them that partitions the space such that wins and losses are separated (where $\mathbf{w}$ is the normal vector to the plane and $\frac{||\mathbf{w}||}{b}$ is the offset of the hyperplane from the origin along $\mathbf{w}$ (Burges, 1998)). To obtain probabilities, a logistic model is fit to the decision values for each binary classifier (Platt, 1999).

Let the output of an SVM be $f(X)$. Then

$$\hat{y}_i^{(1)}(A,B) = \Pr(y_i = 1 | f(X)) = \frac{1}{1 + \exp(Q \cdot f(X) + V)},$$

where Q and V are parameters chosen through maximum likelihood estimation from a training set made up of coordinates $(f(X)_i, y_i)$.

2.3.2 Naive Bayes

This method is based on Bayes Theorem:

$$\Pr(X|Y) = \frac{\Pr(Y|X) \cdot \Pr(X)}{\Pr(Y)}$$

Given a matrix of attributes X we want to predict the label or class y for each row in X. This method works to maximize $\Pr(y|X)$ by choosing $y = \hat{y}$ that maximizes $\Pr(X|y)$. To determine this, we assume that the attributes are independent (a dubious assumption) so that we can use $\Pr(X|\hat{y}) = \Pr(x_1|\hat{y}) \cdot \Pr(x_2|\hat{y}) \cdots \Pr(x_p|\hat{y})$ (where x_ℓ is the ℓth column of X) which can be determined from the data (Manning, Raghavan, and Schütze, 2008).

$$\begin{aligned}\hat{y}_i^{(2)}(A,B) &= \Pr(y_i = 1|X_1, X_2, ...X_p) \\ &= \frac{1}{\Pr(X_1, ..., X_p)} \cdot \Pr(y_i = 1) \cdot \prod_\ell \Pr(X_\ell = x_\ell | y_i = 1)\end{aligned}$$

2.3.3 k-Nearest Neighbor

This method uses the same beginning framework as the support vector machine. We still have each training set game labeled according to the difference in team statistics. For each new game lacking a label, we can obtain all of the same difference statistics. Then, the labeled points closest in Euclidean space to the new point, "vote" to determine the label of the new game. For example, if 3 out of 5 nearest neighbors are winners, then the new point will be assigned to be a winner, or given a 0.6 probability of winning.

Given a candidate game $x_i \in \mathbb{R}^p$, find the k observations in X that are closest to x_i in Euclidean distance. Let $(z_1, \ldots, z_k)$ be this set of observations, and $y_1, \ldots, y_k$ the corresponding outcomes. Then the k-Nearest Neighbor prediction is:

$$\hat{y}_i^{(3)}(A,B) = \frac{1}{k} \cdot \sum_{j=1}^{k} y_j \,.$$

2.3.4 Decision Tree and Random Forest

Decision trees split a data set using a set of binary rules that make up the branches of the tree. Sets further down in the tree contain increasingly similar class labels. The objectives are to produce the purest sets possible and to be able to classify any object based on comparison of features to the rules in the tree.

A random forest is a bootstrap of the decision tree. Like in the standard bootstrap method, the available training data can be resampled to build many different decision trees (Efron, 1979). Additionally, the random forest method takes a random subset of available predictors to build its new decision tree. By resampling both the cases and the input variables, many decision trees are grown. Final classification decisions are made based on a majority rule of the many decision trees.

Given a decision tree with height k, let $\Pr(r_d)$ be the probability associated with node d on branch path r. Then the probability of success for a new case is:

$$\hat{y}_i^{(4)}(A,B) = \prod_d \Pr(r_d).$$

For the random forest of m trees where $\Pr(r_{(d,j)})$ is the probability associated with node d on branch path r on decision tree j, the probability of success for a new case is:

$$\hat{y}_i^{(5)}(A,B) = \frac{1}{m} \sum_j \prod_d \Pr(r_{(d,j)}).$$

2.3.5 *Artificial Neural Networks*

The artificial neural networks model was inspired by biological neural networks. An ANN model contains nodes connected with directed links as neurons in the brain are connected by axons. The algorithm has input nodes representing the attributes (dataset variables) and output nodes, which is the model output. In between, there are intermediate nodes and links which define the strength of the connection between the input and output nodes. The algorithm determines the optimal weights needed to characterize each strength.

Intermediate nodes $Z = (Z_1, ...Z_m)$ are derived from linear combinations of the inputs:

$$Z_m = \sigma(\alpha_{0m} + \alpha_m^T \cdot X),$$

where the α's are the optimal weights for the strength of each node and $\sigma(v)$ is the sigmoid function: $\frac{1}{1+e^{-v}}$.

The inputs then pass through these intermediate nodes via the following function:

$$T = \beta_0 + \beta_1^T \cdot Z$$

where the β's are another set of optimal weights.

Finally the probability of a success $\Pr(y = 1)$ in a binomial setting is (Hastie et al., 2009):

$$\hat{y}_i^{(6)}(A,B) = f(X) = T.$$

3. Results

3.1 Results on Historical Data

We were able to produce predictions that had a smaller log-loss than the three benchmarks set by Kaggle in the first stage. We were in the top 10 of the Kaggle competition for a portion of the competition, and we finished in the top 25 in this stage. Our $L(y,\hat{y})$ was 0.377, where here the outcome vector y was based on known historical tournament results.

3.2 Results on the 2014 NCAA Championship Tournament

We were less successful in the second stage. Disappointingly, we were not able to surpass the coin flip or seed difference benchmarks. We finished in 216th place with a log loss score of $L(y^*,\hat{y}) = 0.865$. However, we are able to tweak our model to surpass the coin flip and seed benchmarks after the fact (see Discussion) which would have placed us in the top 125.

To illustrate which teams had the best chance of winning the tournament based on our predictions, we simulated 1000 random tournaments using $\hat{y}$ as probabilities. Table 4 shows the top ten most likely champions and the number of Final Four appearances of these teams. We note that in this "Year of Upsets", many top-seeded teams which we predicted to fare well were eliminated in the early rounds of the tournament. A complete bracket showing how our predictions fared in the actual tournament is shown in Figure 1. Note that the eventual champion UConn had very poor odds of winning the tournament under our predictions, since we had 95% confidence that Villanova would beat them in the Round of 32.

Nevertheless, we predicted 41 out of 66 games correctly, and we were particularly strong in the South and East where we predicted 25 out of 33 games correctly. We predicted several upsets correctly, including Dayton's run to the Elite Eight and Steven F. Austin over

Team 1 Name	Team 1 Seed	Team 2 Name	Team 2 Seed	Prediction	Point Difference	Log-Loss
New Mexico	7	Stanford	10	0.98	-5	3.85
Massachusetts	6	Tennessee	11	0.97	-19	3.59
Kansas	2	Stanford	10	0.97	-3	3.48
Connecticut	7	Villanova	2	0.05	12	3.09
New Mexico State	13	San Diego State	4	0.95	-4	2.95
NC State	12	Xavier	12	0.07	15	2.72
Mercer	14	Tennessee	11	0.90	-20	2.35
NC State	12	Saint Louis	5	0.90	-3	2.31
Arizona State	10	Texas	7	0.86	-2	1.97
BYU	10	Oregon	7	0.84	-19	1.81
Iowa	12	Tennessee	12	0.82	-13	1.73
Connecticut	7	Florida	1	0.18	10	1.71
Duke	3	Mercer	14	0.80	-7	1.61
N. Dakota State	12	San Diego State	4	0.80	-19	1.60
Kentucky	8	Michigan	2	0.21	3	1.54
Kentucky	8	Wichita State	1	0.23	2	1.48
Kentucky	8	Louisville	4	0.23	5	1.46
Baylor	6	Wisconsin	2	0.76	-17	1.43
Cincinnati	5	Harvard	12	0.75	-4	1.40
N. Dakota State	12	Oklahoma	5	0.25	5	1.39
Albany	16	Mt. St. Mary's	16	0.25	7	1.38
Memphis	8	Virginia	1	0.72	-18	1.28
Oregon	7	Wisconsin	2	0.72	-8	1.27
Connecticut	7	Michigan State	4	0.38	6	0.97
Colorado	8	Pittsburgh	9	0.50	-29	0.70
Tulsa	13	UCLA	4	0.45	-17	0.60
Kentucky	8	Wisconsin	2	0.56	1	0.58
Michigan	2	Wofford	15	0.56	17	0.57
Arizona	1	Weber State	16	0.62	9	0.47
Arizona	1	Wisconsin	2	0.38	-1	0.47
Iowa State	3	North Carolina	6	0.63	2	0.46
Baylor	6	Nebraska	11	0.67	14	0.40
Harvard	12	Michigan State	4	0.32	-7	0.39
Stephen F. Austin	12	UCLA	4	0.27	-17	0.32
Baylor	6	Creighton	3	0.27	30	0.31
Dayton	11	Ohio State	5	0.73	1	0.31
Iowa State	3	NC Central	14	0.73	18	0.31
George Washington	9	Memphis	8	0.24	-5	0.28
Dayton	11	Syracuse	3	0.76	2	0.27
Kansas State	9	Kentucky	8	0.22	-7	0.25
Louisville	4	Manhattan	13	0.78	7	0.25
Delaware	13	Michigan State	4	0.19	-15	0.21
Arizona	1	Gonzaga	8	0.83	23	0.18
Connecticut	7	Iowa State	3	0.86	5	0.15
Michigan State	4	Virginia	1	0.87	2	0.14
Cal Poly	16	Wichita State	1	0.11	-27	0.11
Albany	16	Florida	1	0.11	-12	0.11
Dayton	11	Stanford	10	0.91	10	0.09
North Carolina	5	Providence	11	0.93	2	0.07
Louisville	4	Saint Louis	5	0.94	15	0.07
Eastern Kentucky	15	Kansas	2	0.06	-11	0.06
American	15	Wisconsin	2	0.06	-40	0.06
Florida	1	UCLA	4	0.95	11	0.05
Cal Poly	16	Texas Southern	16	0.95	12	0.05
Coastal Carolina	16	Virginia	1	0.05	-11	0.05
Dayton	11	Florida	1	0.04	-10	0.04
Villanova	2	Milwaukee	15	0.96	20	0.04
Stephen F. Austin	12	VCU	5	0.96	2	0.04
Arizona	1	San Diego State	4	0.96	6	0.04
Connecticut	7	St. Joseph's	10	0.97	8	0.03
Syracuse	3	Western Michigan	14	0.97	24	0.03
Michigan	2	Tennessee	11	0.97	2	0.03
Creighton	3	La. Lafayette	14	0.97	10	0.03
Gonzaga	8	Oklahoma State	9	0.97	8	0.03
Florida	1	Pittsburgh	9	0.97	16	0.03
Michigan	2	Texas	7	0.98	14	0.02

Table 3: Complete Results for our submission. The Prediction column is our proposed probability that Team 1 would beat Team 2 ($\hat{y}$), and the point differential is given from Team 1's perspective. Note the coin flip benchmark was 0.693.

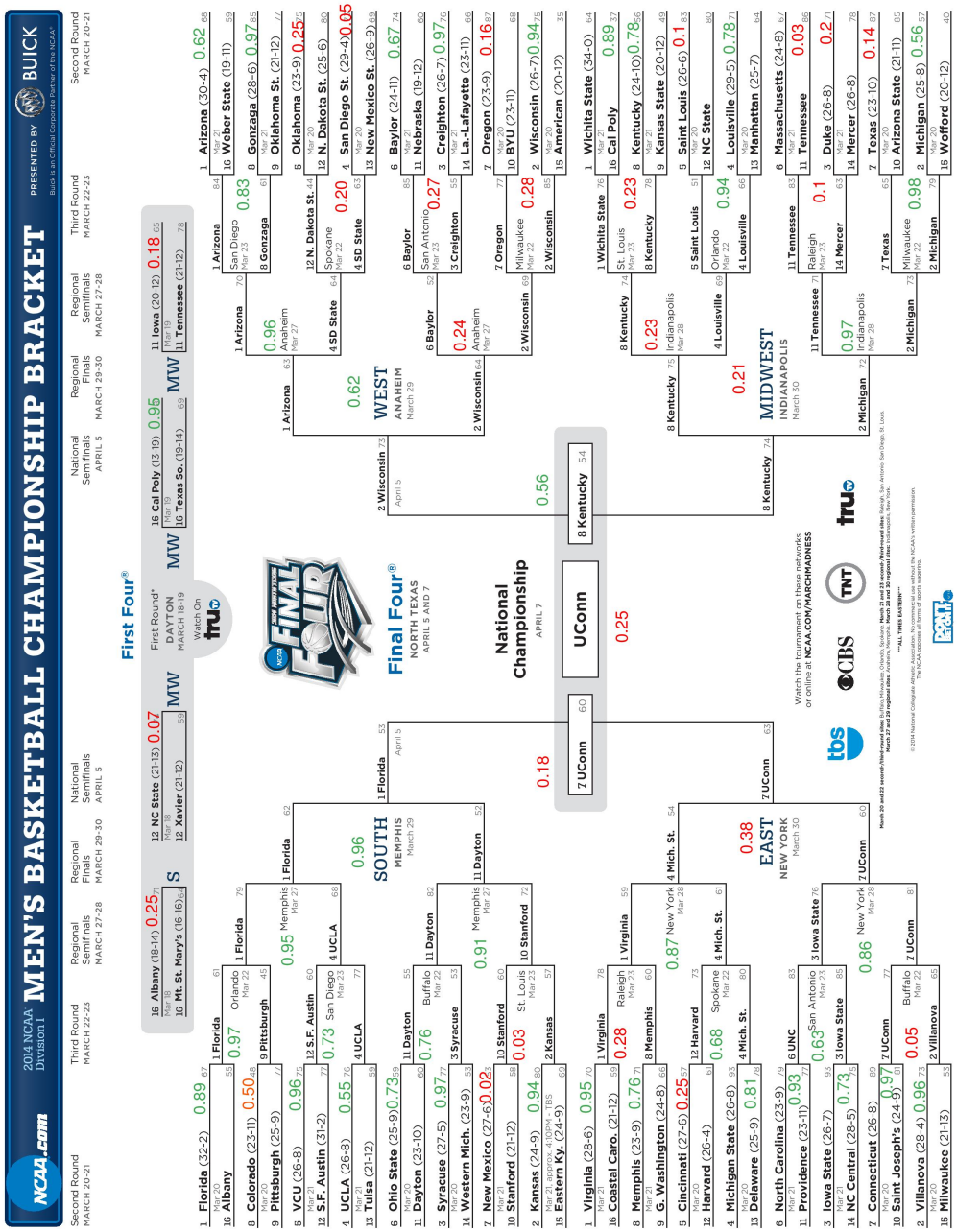

Figure 1: 2014 Bracket Predictions: The probabilities shown are the probability that we gave to the winner. A number in red indicates where we were wrong, and a number in green indicates where we were right. Note that UConn's path is blocked by Villanova, which we had as heavy favorites to beat them in the Round of 32.

Seed	Region	Team	Champions	Final Four	Actual Finish
1	South	Florida	312	499	Final Four
2	South	Kansas	176	383	Round of 32
2	East	Villanova	105	239	Round of 32
1	Midwest	Wichita State	98	349	Round of 32
1	West	Arizona	79	360	Elite Eight
3	Midwest	Duke	45	248	Round of 64
3	East	Iowa State	39	282	Sweet Sixteen
4	Midwest	Louisville	31	170	Sweet Sixteen
3	West	Creighton	18	337	Round of 32
4	East	Michigan State	14	200	Elite Eight

Table 4: Top 10 Teams by Championship Wins and Final Four appearances in 1000 simulated tournaments using our predictions.

Round	N	L	Cumulative L	$\max L_i$	$\min L_i$
1	4	1.47	1.47	2.72	0.05
2	32	0.81	0.88	3.85	0.03
3	16	1.04	0.93	3.48	0.02
4	8	0.42	0.86	1.46	0.03
5	4	0.76	0.86	1.54	0.04
6	2	1.14	0.87	1.71	0.58
7	1	1.40	0.87	1.40	1.40

Table 5: Log-Loss Score by Round. Note that we did quite well in Round 4, but were otherwise bested by the coin flip benchmark in all other rounds.

VCU (Felt, 2014). We also placed a more conservative probability that Duke would beat Mercer at only 0.8. However, we had 23 games on which we lost more than one point (of L), and on 8 of these games we lost over 2 points. We can see in Table 3 that we were not expecting Stanford, Kentucky, or Connecticut to perform as well as they did, and our high hopes for UMass, Kansas, and NC State were dashed.

In Table 5 we show our progress throughout the tournament. Our best round was the fourth, and our worst was the initial play-in round.

4. Discussion

Although our model predicted 61% of the 2014 NCAA Championship games correctly, it made many damaging mistakes. Our performance could have been hindered by overfitting, an overly complex model, or an overlooked trend, among other things.

4.1 Overfitting and Smoothing

The heavy influence of the decision tree on our model, coupled with the good results on the past tournaments, but poor results on the actual tournament, suggests that our model was overfit to the historical data. Another possibility is that the decision tree produced only a small number of unique values of $\hat{y}^{(4)}$ due to sparsity at the leaves.

Many of the $\hat{y}^{(4)}$ values are very extreme, and due to the size of the decision tree (maximum height of 8), the sample size at each leaf is small. Our training set X has 329 rows representing the 329 complete-case games, so with these spread out among about 20 leaves,

the average leaf contains relatively few observations. Chawla (2006) recommends smoothing and bagging the leaves in this event, and presents two approaches:

- Laplacian smoothing: At each leaf, set the prediction equal to:

$$\frac{TP+1}{TP+FP+2},$$

  where TP is the number of true positives that end up here from the training set, FP is the number of false positives that end up here from the training set.

- m-estimate: Here the prediction becomes

$$\frac{TP+b\cdot m}{TP+FP+m},$$

  where b is the prior estimate of the positive class (here: $b=0.5$) and m is a parameter to control the shift towards b. Chawla (2006) recommends $b\cdot m=10$.

Using these modifications, we were able to reduce our log-loss score to 0.596, which was just above the seed benchmark (0.6), and represented a a large jump from our actual Kaggle score of 0.854. We obtained similar results using bagging.

4.2 Residual Analysis

We were badly hurt by a few games in which we had false confidence in the losing team. Some of these games were close (e.g. Arizona State lost to Texas by two points on a buzzer-beater) and may be attributed to bad luck, but others were not (e.g. we thought UMass had a 97% chance to beat Tennessee, but they lost by 19 points). Unfortunately, we were not able to detect any obvious trends in these large deviance games.

4.3 The Role of Chance

To assess the extent of our bad luck vs. bad predictions, we generated a sampling distribution for $L(y|\hat{y})$ by simulating 1000 random tournaments. If we operate under the null hypothesis that our predictions are the true probabilities (i.e. $H_0 : y=\hat{y}$), then we can determine how likely our log-loss score of 0.854 from the actual tournament was. Unfortunately, this value falls in the far right tail of the distribution of the log-loss obtained from the simulation (see Figure 2). Therefore we must reject the null hypothesis that our predictions were accurate.

We repeated this procedure for *One Shining MGF*'s predictions $\hat{y}_{mgf}$ (i.e. $H_0 : y=\hat{y}_{mgf}$). When we do this, we find that their log-loss score of 0.529 falls nicely in the middle of the distribution, providing no evidence against the null hypothesis that their predictions are correct.

Note, however, that had we been correct, our log-loss would have been smaller than that of *One Shining MGF*. This reflects our success in Stage One of the competition.

Under this same procedure, the log-loss score of the seed benchmark (0.600) remains within the acceptance region. We fail to reject the null hypothesis that the seed benchmark probabilities are correct. However, it makes sense that the log-loss score falls to the right of the center in light of the large number of upsets that occurred during this year's tournament.

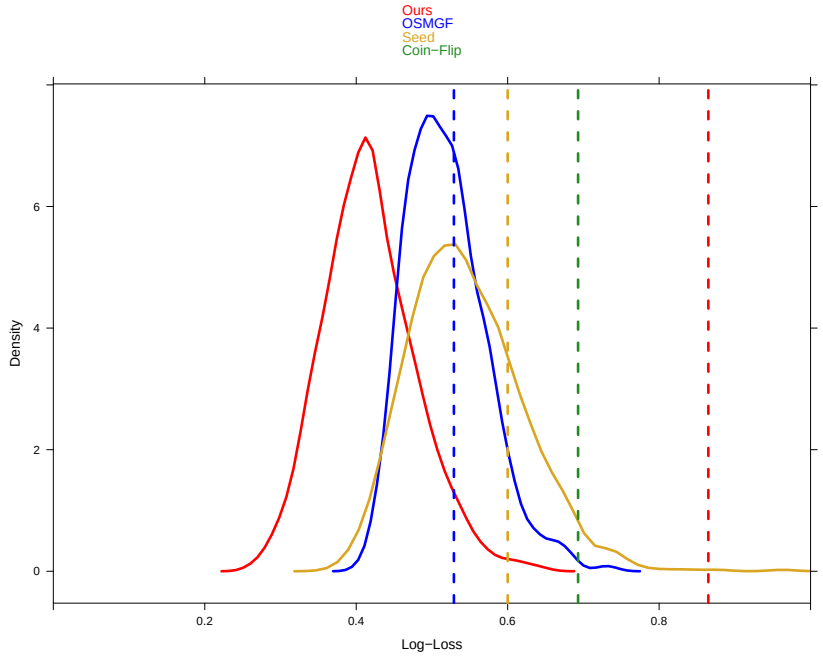

Figure 2: Simulations Under Different Null Hypotheses

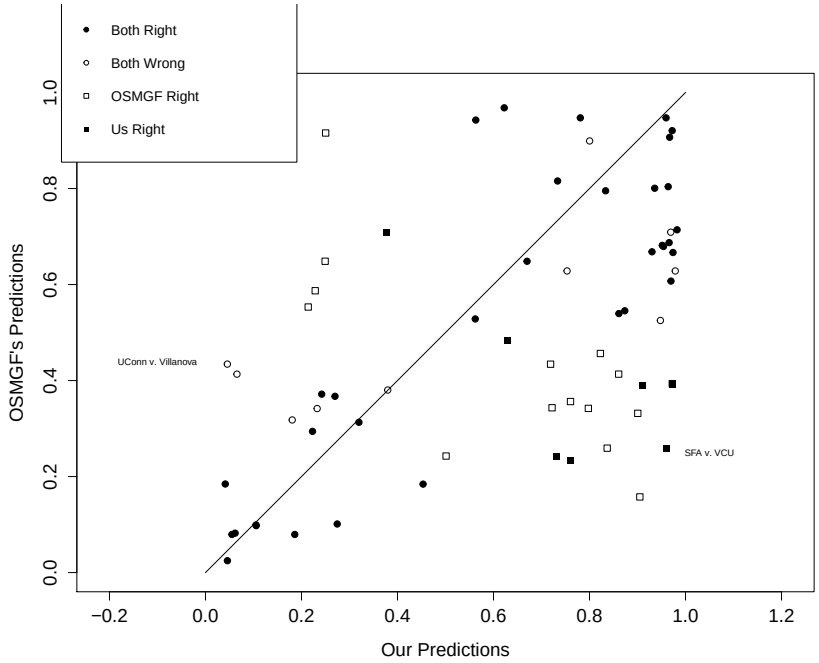

Figure 3: Comparison of Our Predictions to *One Shining MGF*'s. Games in which we agreed are shown are circles, and games for which we disagreed are shown as squares. If our prediction was accurate, then the mark is filled in.

4.4 Comparison to the Winner's Predictions

In Figure 3, we compare our predictions to those of *One Shining MGF* and see significant differences. Many of our predictions were higher than the respective predictions of the winning team (more points to the right of the 45 degree line than the left). There were 10 games that we both got wrong and 38 that we both got right. There were 14 games that the winning team predicted right and we predicted wrong while there were 6 games that we predicted right that the winning team did not. The average log-loss per category can be found in Table 6. We note that when we were correct, our Log-Loss scores were better than *One Shining MGF*'s, but when we were incorrect, our scores were much worse than theirs when they were incorrect.

	Our Average Log-Loss	OSMGF's Average Log-Loss
Both Correct	0.19	0.26
Both Wrong	2.32	1.12
OSMGF Right	1.72	0.42
Us Right	0.24	1.15

Table 6: Average Log-Loss Score by Category. Note that we did better on average in the games where we were both right.

5. Future Work

In future work, inclusion of recent data could improve our results and lead to fewer games with a large log-loss. We did have access to the 2013–2014 regular season data, however, we chose not to include it in the model. Our perspective was that tournament games, which are played on a neutral court on a regular schedule, are importantly different than regular-season games, where factors like home-court advantage have a strong influence. Furthermore, information from regular-season games appears in the training set through the seeds, RPI, and other seasonal aggregate measures. In light of our performance, this perspective many need revision. Access to other statistics that are not averaged over the season might also provide more accuracy.

References

C. J. Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

B. P. Carlin. Improved ncaa basketball tournament modeling via point spread and team strength information. *The American Statistician*, 50(1):39–43, 1996. doi: 10.1080/00031305.1996.10473540. http://amstat.tandfonline.com/doi/abs/10.1080/00031305.1996.10473540.

N. V. Chawla. Many are better than one: Improving probabilistic estimates from decision trees. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 41–55. Springer, 2006.

T. G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin Heidelberg, 2000. ISBN 978-3-540-67704-8. http://dx.doi.org/10.1007/3-540-45014-9_1.

P. Dizikes. Into the pool: Ncaa tourney betting booms. *ABC News*, March 2014. http://abcnews.go.com/Business/story?id=88479.

B. Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

H. Felt. Dayton upsets Syracuse and UConn eliminate Villanova in March Madness, March 2014. Accessed: 2014-03-23, http://www.theguardian.com/sport/2014/mar/23/dayton-upsets-syracuse-uconn-eliminate-villanova-march-madness.

A. Y. Govan, C. D. Meyer, and R. Albright. Generalizing google's pagerank to rank national football league teams. In *Proceedings of the SAS Global Forum*, volume 151-2008, 2008.

M. Greenfield. Team Rankings, 2014. Accessed: 2014-03-01, http://www.teamrankings.com.

T. Hastie, R. Tibshirani, and J. J. H. Friedman. *The elements of statistical learning*. Springer New York, 2nd edition, 2009. http://www-stat.stanford.edu/~tibs/ElemStatLearn/.

C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

NCAA. National Rankings, 2014. Accessed: 2014-03-01, http://stats.ncaa.org.

D. Oliver. *Basketball on Paper: Rules and Tools for Performance Analysis*. Brassey's, Incorporated, 2004. ISBN 9781574886870. http://books.google.com/books?id=Xh2iSGCqJJYC.

L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

J. C. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *ADVANCES IN LARGE MARGIN CLASSIFIERS*, pages 61–74. MIT Press, 1999.

R. Schumaker, O. Solieman, and H. Chen. Predictive modeling for sports and gaming. In *Sports Data Mining*, volume 26 of *Integrated Series in Information Systems*, pages 55–63. Springer US, 2010. ISBN 978-1-4419-6729-9. http://dx.doi.org/10.1007/978-1-4419-6730-5_6.

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. doi: 10.1080/00031305.1999.10474438. http://amstat.tandfonline.com/doi/abs/10.1080/00031305.1999.10474438.

P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Pearson Addison-Wesley, 1st edition, 2006.

G. Valentini. Ensemble methods based on bias-variance analysis, 2003.

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