**How to not Blow a 3-1 Lead in this Year’s Bracket Predictions**

**Damir M. ZhaksilikovJason Frazier**

Department of Computer Science Department of Computer Science

University of Washington University of Washington

Seattle, WA 98105 Seattle, WA 98105

*damir@cs.washington.edu* *jasonf56@cs.washington.edu*

**Abstract (Damir)**

As of recent, the NCAA March Madness tournament has piqued the interests of the computing and machine learning community. The tournament’s seemingly unpredictable nature questions whether tournament brackets and the associated upsets can be accurately predicted. In this document, we will explore the existence of upset indicators, success of various models in predicting brackets, and impact of mixing raw team statistics with various measures of team rankings. First, we describe the results of regular season data manipulation and model performance of Logistic Regression, Random Forest, Multilayer Perceptron, and K-Nearest Neighbors models, of which Random Forest consistently outperformed others. Next, we outline data and model improvement to cater towards the nature of March Madness Tournament play. Our model’s success in prediction is then quantified – we estimate that it has an accuracy of about 70 and 60 percent accuracy in predicting regular and tournament games respectively. Our techniques in tuning our model provides a 25% and 20% increase in correctness from tournament game predictions made by non-experts increase from experts respectively [1].

**1 Project Outline (Damir)**

**1.1 NCAA March Madness Tournament**

The NCAA Division I Men’s Basketball (nicknamed March Madness) is a single elimination basketball tournament played by 68 Division I collegiate basketball teams. The tournament consists of 32 teams who win Division I conferences and 36 teams awarded positions via committee choice [2]. It has become tradition for fans to predict the tournament bracket given initial team slots. Due to the nature of single elimination, game outcomes are highly unpredictable. Additionally, there are 9.2 quintillion valid bracket combinations. Thus, the prediction of the correct bracket combination is highly improbable.

**1.2 Workflow**

Our first goal was to determine the best model for predicting NCAA basketball game results. Although there are differences between tournament and regular season play, we operated under the assumption that model success would reflect across all forms of NCAA basketball games. Using regular season data from 2003–2016 we tested the success of Logistic Regression, Random Forest, Multilayer Perceptron, and K Nearest Neighbors models. Once Random Forest was determined as the most successful model, we adjusted our tournament data set to reflect teams’ yearly performance, tournament seeding, and Las Vegas point spreads for each game and weighted upset games more heavily. Our models were then run on tournament data on a March Madness tournament simulator which provided the most probable bracket based on our models’ predictions.

**1.3 Key focuses**

The following questions were central to formulating our workflow and model/data manipulation techniques:

* Do predictable indicators of an upset exist?
* Is there a correlation between tournament seeding and the Las Vegas Point Spread?
* Is it possible to combine team statistics with seeding and the Las Vegas point spread to predict more accurately than the spread/seeding on their own?
* How does performance during the regular season transfer to tournament performance?

**1.4 Hypothesis**

As discussed by Lopez and Matthews, the “rules of efficient gambling markets” state that it is impossible to beat the spread of the Las Vegas sports books in the long run [3]. We believe that it is possible to train our model to predict upsets more accurately than the spread through weighting upsets more heavily. The negative aspect of this, though, is that our ability to predict “straightforward” results will suffer. Additionally, we expect that our model prediction performance will drop from regular season to tournament games as tournament games are considered to be more unpredictable due to a variety of extraneous factors.

**2 Data Overview (Damir)**

**2.1 Original dataset**

Kaggle’s March Machine Learning Mania 2017 challenge provided data for all NCAA basketball regular season and tournament games from 2003 – 2016 [4]. Each data point contained a game’s statistics pertaining to game information such as the year, location, and participating team IDs. Additionally, each point provided game statistics for both participating teams (e.g. points scored, field goals made, steals, etc.). Teams for each game were marked as team1 and team2 where team1 was the winner of each matchup.

The Kaggle challenge also provided a table mapping team IDs to the school name. Other data was provided, but not used. We used Python’s SQLite library to format and modify data as necessary.

**2.2 Regular Season Games**

To experiment with various models on the success of predicting regular season games we needed to reformat team statistics to reflect the strength of participating teams rather than their actual performance during that game. Obviously, in its original format the model would simply choose the winner of the game as whoever scored more points, this was not the function of our intended model.

To reformat data, we replaced each team’s statistics for a game with the average of their statistics for an entire year. This way, the averages reflected the team’s strength in that statistic and the model could use this as a factor in impacting game evaluation. Additionally, we added an indicator value to classify the games, where 1 meant team1 won and 0 meant otherwise. Since team1 won each game in the original dataset, each classifier value was 1 for the original data. In this given format, the model would simply predict that team1 for any game and feature values would be irrelevant. Thus, for each data point, we created a copy with a team1 and team2 flipped and an indicator value of 0. We had hoped, as well, that this would completely remove any bias towards weights in team1 over team2 and that those weights would ultimately have an inverse relationship.

**2.3 March Madness Games**

The March Madness games were reformatted to contain the same averages found in the regular season for each team during a given year. This made sense to us, as a team’s performance during the regular season is presumably an accurate reflection of the same team playing in the tournament. In addition, we added the tournament seed of each team and the Las Vegas point spread for the game.

**2.3.1 Seeding**

Tournament seeds are preliminary rankings. For March Madness purposes, these values are determined by the NCAA tournament committee [2]. For each region, there are seeds given 1-16, but the NCAA also releases an all-inclusive 1-68 ranking of all participants. For each game in the March Madness dataset we added the participating teams’ tournament seed. The intuition is that the highest seeded teams consistently make it to the final rounds of the tournament. Thus, the seed is an accurate representation of more qualified teams. Additionally, although raw statistics provide insight on individual strength of each team, the seed provides an all-encompassing team ranking which we believed would holistically strengthen our model. These values were entered manually and due to time constraints, we entered values for 2012-2016, all other values were 0.

**2.3.2 Las Vegas Point Spread**

The Las Vegas Point Spread “provides the predicted difference in total points scored between the visiting and the home team; a spread of – 5.5, for example, implies that the home team is expected to win by 5.5 points” [3]. In our case, team1 was considered the home team as all games were in neutral locations. Our intuition for adding these values comes from Lopez and Matthews who argue that it is impossible to beat the Las Vegas Point Spread in the long run due to normalization associated with the large number of bets made [3]. Lopez and Matthews cite that “should act as the standards on which to judge any pre-game predictions” [3]. These values were entered by hand from Odds Shark [5].

We ran into numerous issues with Odds Shark. Firstly, we simply did not have the resources to gather data for every game (naming conventions between our name-to-ID and Odds Shark were different therefore data scraping was ruled out). Additionally, Vegas odds did not exist for all games between 2003-2016 and no odds exist for this year as the bracket has not been finalized. Therefore, we a built an L2 Normalized Linear Regression model to find a relationship between tournament seeding and Las Vegas odds. For games with seeds, but no Las Vegas Odds we used the relationship to guess the Las Vegas odds. Our intuition was that both values are based on overall perceived team strength and thus must be at least loosely correlated. Our model found the following weights:

0.29599983 for the higher team seed

-0.37982249 for the lower team seed

These values are consistent as the Las Vegas odds are negative towards the better (lower seeded) team. But, since the higher seed values are larger than the lower seed values, the weight must be lower in magnitude for higher seeds.

**2.4 Feature selection results**

The tournament data originally used contained 35 features including both raw statistics as well as seeding and Las Vegas odds. We ran L1 Regularized Linear Regression feature selection to determine the most impactful features. After running feature selection, our model contained between 17 and 20 features depending on the run. The results are as follows:

Figure 1.2: Scores for Features after Feature Selection on Regular Season Data

Figure 1.1: Scores for Features after Feature Selection on Regular Season Data

Figure 1.2: Scores for Features after Feature Selection on Tournament Data

Regular Season Data

* Location was the most impactful feature for determining the winner of regular season games. All March Madness games are played from a neutral location and could account for some of the unpredictability in games results.
* As expected, average score and score difference were the most impactful raw team statistics.

Tournament Data

* Average score difference was the most important feature for prediction.
* Vegas odds were the second most important feature and this supports our claim that Vegas odds should be a standard for pre-game indication.
* Seeding plays very little role in the result of the tournament games. This may be because we did not have seeds for all games, but if this was the case our Vegas odds would have a similar score (as we have the same number of Vegas odds and seeds statistics). Thus, this implies that our assumption that Las Vegas odds and tournament seeds are correlated is incorrect.

**3 Model Selection (Jason)**

**3.1 Original Model**

At first we figured using logistic regression would be the best model to use with our dataset. Shortly after building a logistic regression model we realized that due our dataset having over 100,000 rows of data that Stochastic Gradient descent would be a better option. We used our SGD code from homework 2 as a starting point, editing it to work for this specific problem.

**3.1.1 L2 Normalization**

After running SGD on our dataset, we noticed that many of our resulting weights were well over 100,000 with some weights being in the millions, as well as our prediction rates consistently being around 50%. In class, we learned that extremely high learned weights could be the result of overfitting, and due to our data having 35 features we thought that was exactly what was happening. To stop our model from overfitting we decided to add L2 normalization. Once L2 was adding to the model all resulting weights were within the range of [2, -2] and our prediction rates rose to around 60%.

**3.2 Additional Models**

We thought it would be interesting to add additional models to see which model would give the best prediction rates. We decided to use the scikit-learn python library to add these models in. We decided on adding 3 additional models; K-Nearest Neighbors, Random Forest, and Multi-Layer Perceptron. K-Nearest Neighbors seemed like a great model to add as teams with similar stats usually tend to play similarly. After reading documentation on many different classifiers, Random Forest and Multi-Layer perceptron learners seemed like a good fit for this problem. Random Forest fits multiple decision tree stumps on sub-samples of the dataset and averages those results to try and control overfitting and increase prediction rates. This sounded like a good idea to us as overfitting was an issue with Logistic Regression so we wanted to make sure that all other models didn’t suffer from the same problem. We chose to use the Multi-Layer Perceptron classifier as in class we just started to learn about Bayes Nets and Neural Networks so thought it would be a good idea to test it out on our final project. A MLP also has the ability to try and fit data that may not have a linear relationship.

**3.3 Choosing Model Parameters**

All 4 of our models have many different parameters that can be finely tunes to increase prediction rates, such as lambda values, step sizes, and number of estimators. To try and maximize our prediction rates, we ran every model multiple times with different parameters to try and choose which gave us the best results For SGD we plotted the number of iterations over the dataset and learned that 15 iterations over the dataset gave us the best prediction results. For K-Nearest Neighbors, we found that using uniform weights over distance consistently gave us about 3% better prediction rates. We also plotted the number of neighbors, using multiples of 10 from 0 to 400 and found a neighbor number of 330 to consistently give us the best rates. With the Random Forest classifier, we iterated over the number of decision tree estimators our model should use, coming to the conclusion that 22 estimators was ideal. With the Multi-Layer Perception classifier, we decided to learn based on what lambda values to use, coming up with a lambda value of .1 giving the best prediction rates. The graphs for choosing these values are as follows:

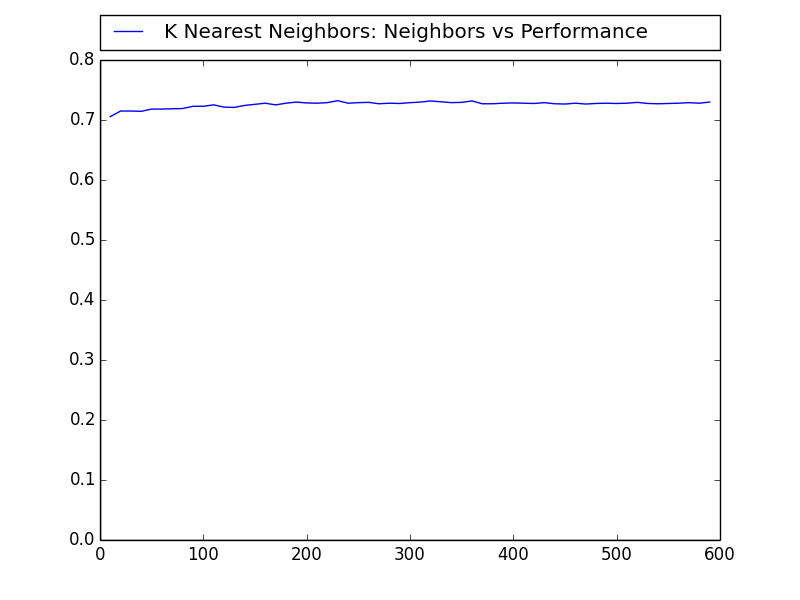
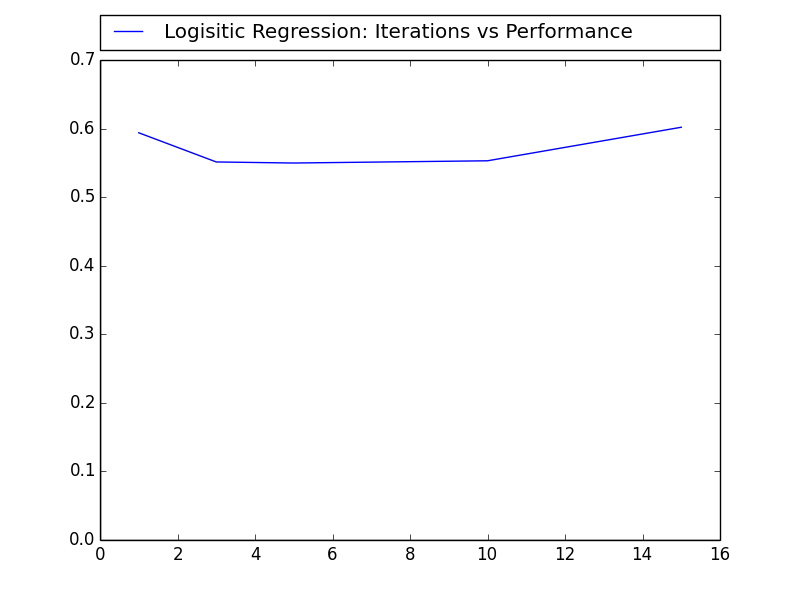


Figure 1.1: Number of dataset iterations vs Percent Right Figure 1.2: Number of Neighbors vs Percent Right

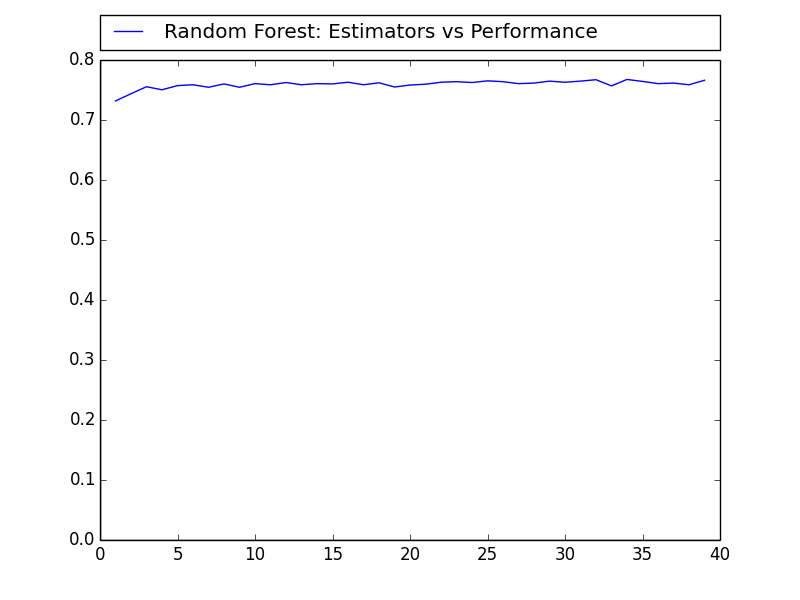
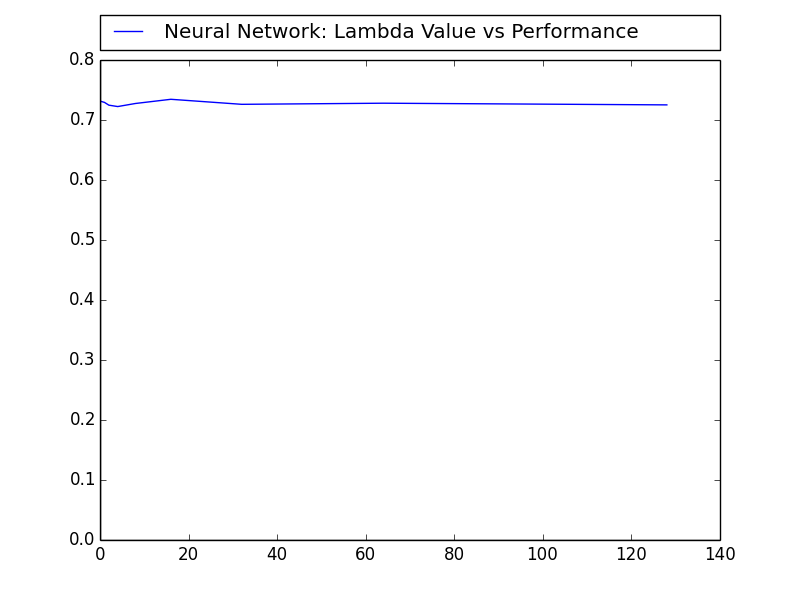


Figure 1.3: Lambda Values vs Percent Right Figure 1.4: Number of Estimators vs Percent Right

**4 Tournament Adjustments (Damir)**

As mentioned above, tournament seeds and Las Vegas odds were added to our dataset to provide a holistic indicator of team strength. Additional data manipulation and simulation of tournament play was implemented to address differences between regular season and tournament play.

**4.1 Upset Weight**

The March Madness tournament is popularly known for tournament “upsets” where a team perceived to be considerably worse than their competitor wins a matchup. Upsets of various degree have occurred every year and pose the greatest challenge for prediction models. A key interest we had during these experiments was to find any indicators of a possible upset if they exist. The intuition for our focus on upsets is due to the rules of efficient gambling markets. Since it is impossible to beat Las Vegas odds in the long run, our model would simply converge towards Las Vegas predictions if provided enough raw team statistics and tournament prediction data.

Our solution to predicting upsets was to weight upset data points more heavily. Using news reports from previous years, we found for each year and repeated the corresponding data points in our set 3-5 times in our dataset. We had hoped that this would influence our model to search for a correlation between upset games. Our measure for the success of this method was to determine how many (if any) tournament upsets our model predicted. This method obviously has the potential for overfitting our model towards making upset predictions. But, if provided more time we would be interesting in experimenting with various upset weights and other models of data manipulation to make our model to predict upsets accurately (rather than over fit). Additionally, there are unquantifiable factors which contribute to upsets such as how team operates under tournament pressure. As these factors are unquantifiable, it would be impossible to train our model to account for them.

**4.2 Simulated Tournament Play**

**4.2.1 Determining winning teams**

Simulating tournament play caused us to make many changes to our original program we had written to test our models on regular season data. As we shuffled our dataset on every call to our program, it was possible for us to get different results on each run of our program. Our solution to try and minimize this randomness was to take the team who had the highest probability of winning a game to move on. We did this by running each game 100 times, a result of 1 meant that team 1 had won the game, and a result of 0 meant that team 2 had won the game. We summed up the 0 and 1 results and divided them by 100, giving which teams won a majority of the simulated games. The team who had won the majority of the 100 games was then picked to move on to the next round. Due to the nature of how tournament matchups work, editing our dataset for future games ended being fairly simple. As the number of teams was always cut in half each round, simply each game was determined just by taking pairs of 2 adjacent teams in our dataset. Once a team lost, a new dataset was built with them excluded and the rest of the teams shifted up.

As March Madness actually starts with 68 teams instead of 64, we couldn’t start with this method for determining winners. To get around the unevenness, we put placeholders in our data where the winner of the first four games would be placed, and created a separate dataset for just the first four. Using the same method above, we determined the highest probable winner for the first four, then inserted the winning team’s id into the bracket dataset, so it could be ran with the above method.

**4.2.1 Disabling Seeding for the Elite Eight**

With prior knowledge to how March Madness works, as a group we decided that once teams got into the Elite Eight that most games are usually a 50/50. Because of this, we decided it would be best if we turned off the idea of seeding and Vegas odds once teams got into the Elite Eight. With seeding still turned on, most of our models showed that all 4 number 1 seeds consistently made it into the Final Four, which is very rarely the case. By turning the idea of seeding off, our tournament simulation would then pick the winners of these games based of team stats alone, not by their perceived strength as a team. This changed substantially improved our models and we started getting results that better represented tournament play.

**4.3 Bracket Scoring**

Traditionally, brackets are scored using a point system that increases the score of a correctly predicted game per round. This is done to give each round the same amount of available points. Since this is a standard for evaluating brackets, we the NCAA scoring system to evaluate our 2016 generated bracket [6].

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Round | 1 | 2 | 3 | 4 | 5 | 6 |
| Points per Game | 1 | 2 | 4 | 8 | 16 | 32 |

Table 1: Scoring system for brackets

**5 Results (Jason)**

Comparing all 4 of our models against one another, the Random Forest classifier consistently gave us the best prediction results. Learning on just regular season data, Random Forest averaged a prediction rate of 76% whereas the other classifiers averaged around 73%. However, as tournament play is much more unpredictable than regular season play, all models lost about 10% accuracy on their predictions when switched to trying to predict tournament outcomes. We believe that this is due to the neutrality of the location of each game. As mentioned above, the location of a game was one of the highest weighted feature in our dataset for determining the winner, which leads us to believe that a neutral location makes the games much more unpredictable.

For creating brackets based on our models, we decided to make 5 brackets in total. We created a bracket for each individual model, as well as create one additional bracket that averages the results of every model. We then ran our program using both the starting brackets from 2016 and 2017. We generated brackets based off of 2016 so we could see how our programs compared to the actual results of the tournament. Comparing our generated 2016 brackets to the actual results, our project did fairly well. As mentioned earlier, we thought upsets would be a big problem due to the randomness of their occurrence, but some of our generated brackets did predict some of them to happen. One of our brackets even correctly predicted the upset Middle Tennessee had over Michigan St. in the first round. Most of our generated brackets also correctly guessed at least two teams in the Final Four, with one of our brackets correctly guessing three out of four of the final teams. To our surprise, the model that gave us the best scoring bracket for 2016 ended up being the Multi-Layer Perceptron model. While the other models averaged a bracket score between 60 and 75, the MLP generated a bracket that scored 104. While we don’t have the 2017 tournament results yet, this does give us confidence in our model for fairly accurately generating brackets.

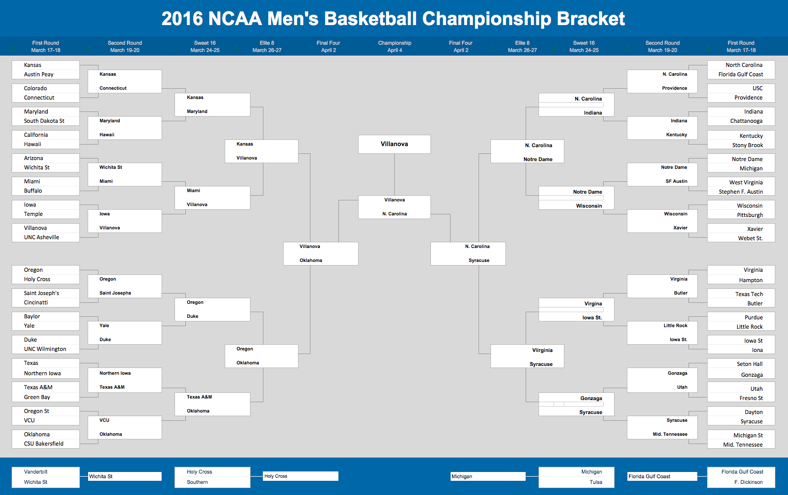
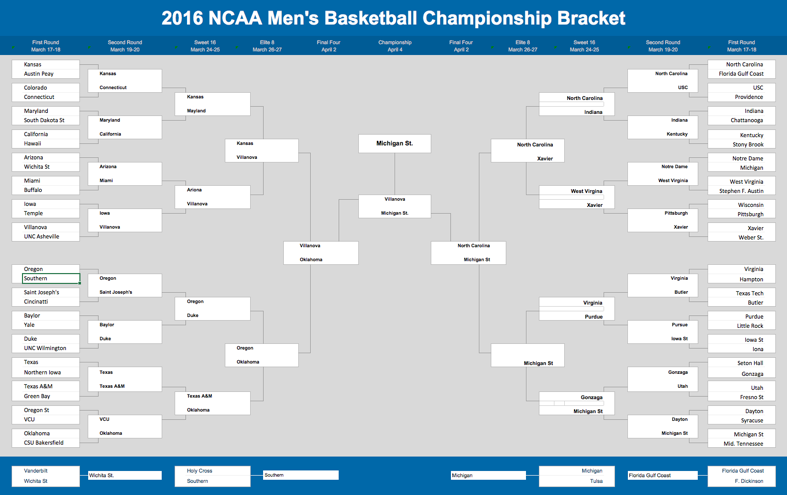


Figure 1.1: Actual 2016 March Madness Results Figure 1.2: Our best scoring generated bracket

**6 Conclusions (Jason)**

As stated in our hypothesis, “rules of efficient gambling markets” state that it is impossible to beat the spread of the Las Vegas sports books in the long run. Throughout the course of working on this project we do believe that this does hold true to some extent. Because of this we decided to supplement out data with seeding numbers and predicted Vegas odds for each game. To our surprise, seeding was considered unimportant when it came to determining a winner whereas Vegas odds still maintained as one of the most important along with location of games. Even with weighting upset games more heavily in our dataset, our model did still prefer standard predictions when it came to matchups, simply because higher seeded teams tended to have better overall statistics. Even with extreme knowledge of every team, it is almost impossible to perfectly guess March Madness brackets due to the inherit unpredictable nature of the tournament itself. In the end, to our surprise, our models actually predicted multiple upsets to occur, as well as concluding that a number 6 seed would win the entire tournament.

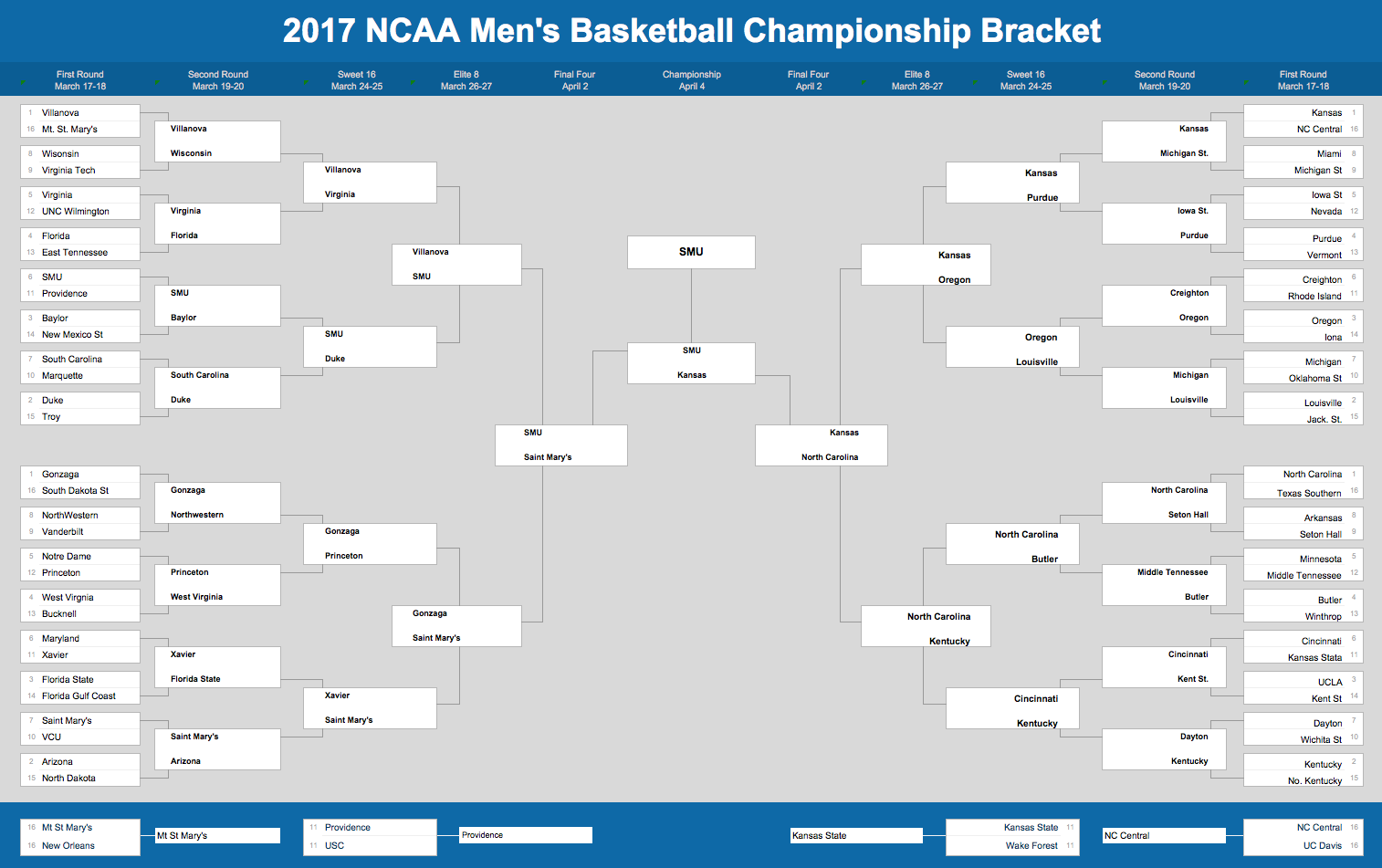


Figure 1.1: Our final 2017 March Madness Bracket Prediction

**References**

[1] Pettigrew, S. (2014, April 10). Proof That America Fills Out March Madness Brackets Like Idiots. Retrieved March 09, 2017, from http://stephenpettigrew.kinja.com/11-million-brackets-vs-espn-cbs-and-fox-experts-who-1561354312/1562354592

[2] N. (n.d.). March Madness bracket: How the 68 teams are selected for the Division I Men's Basketball Tournament. Retrieved March 09, 2017, from http://www.ncaa.com/news/basketball-men/article/2017-03-12/march-madness-bracket-how-68-teams-are-selected-division-i

[3] Lopez, M. J., & Matthews, G. J. (2015). Building an NCAA men’s basketball predictive model and quantifying 131 its success. Journal of Quantitative Analysis in Sports. Retrieved February 20, 2017

[4] March Machine Learning Mania 2017 | Kaggle. (n.d.). Retrieved February 20, 2017, from https://www.kaggle.com/c/march-machine-learning-mania-2017

[5] NCAA Basketball. (n.d.). Retrieved March 01, 2017, from http://www.oddsshark.com/ncaab