NCAA March Madness: Exploring the Mathematics Behind Cinderella Teams

Andrew Huang and Vinai Rachakonda

Abstract. This paper seeks to use regular season performance and tournament seeding to predict the performance of low-seeded teams in the annual March Madness tournament. Using regression, classification, and ordinal regression algorithms, we seek to develop a method of outperforming expert analysts' picks for the NCAA tournament.

Keywords and phrases: March Madness, Cinderella, regression, classification, ordinal regression

1. Introduction

In March of every year, millions of basketball fans around the globe fill out predictions on the NCAA basketball tournament in hopes of being the first person in history to complete a perfect bracket, one where the outcomes of all 67 games are guessed correctly. The best bracket in 2017 made it to 39 games before making its first mistake[1]. The biggest reason that this feat is practically impossible is due to the nature of cinderella teams - low-ranked teams that unexpectedly win far more games than projected. This paper seeks to explore the correlation between low-ranked basketball teams' regular season performance and their March Madness outcomes. Through mathematical modeling and machine learning, this paper hopes to discover the mathematics behind cinderella teams - if such a thing exists.

2. Methodology

Our team aims to use machine learning to predict the performance of low-seeded teams in NCAA March Madness. In specific, we want to use regression, classification, and ordinal regression algorithms to predict the number of games a team will win if they are seeded between #9-16 regionally.

To accomplish this, we took data from Sports Reference, which included basic stats (i.e. total points scored, total wins, etc.) along with advanced stats (i.e. TS%, TRB%, ORtg, etc.). In addition, we manually added each team's tournament seed and total tournament games won. We used the basic stats, advanced stats, and tournament seed as the algorithms' inputs to predict the number of tournament games won for each team.

As aforementioned, we ran three main types of algorithms – regression, classification, and ordinal regression. The exact algorithms we used were: regression - Linear, Ridge, and Lasso Regression; classification - Linear SVM, Linear SGD, Naïve Bayes; ordinal regression - Ordinal Ridge Regression and LAD.

For regression and classification, we used the packages available on scikit-learn, while for ordinal regression, we used the package called mord [2]. Ordinal regression is a form of regression that attempts to predict a discrete and ordered variable. Below, in Figure 1, we outline a brief outline of our methodology. After calculating the Mock Bracket Score (see Section 3), we analyzed our results and used feature analysis to edit our input.

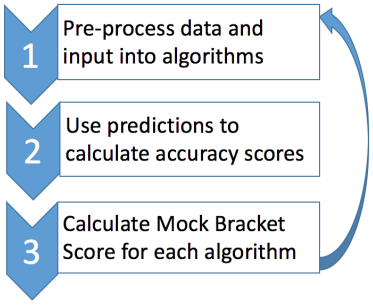

Figure 1: Process Flowchart

Through this experiment, our expectations were to outperform expert analysts', as well as a simple "baseline" of not picking any upsets for the tournament. For the expert analysts', we picked 4 ESPN analysts that published their brackets

(Michelle Beadle, Stephen A. Smith, Jalen Rose, and Roman Shelbourne), as well as the People's Bracket, a bracket created by selecting the majority pick.

For the experiment, we trained on the years 1992 to 2017, and predicted on the 2018 tournament data. The reasoning for this is that to get the best predictions for the current year, we should train on as much data as possible. During the progress of this project, the 2018 tournament was not over yet, and it was thus fitting to test on the final results of the 2018 data.

3. Results

Both the regression and the classification algorithms outputted different results. For regression (regular and ordinal regression), we were able to obtain R^2 and RMSE values, while for classification, we obtained accuracy scores on the test data.

To be able to compare the outputs of these algorithms, along with the expert brackets, we developed the Mock Bracket Score (MBS). This is a scoring function based on the point totals from the ESPN Bracket Challenge; it rewards correctly predicted games, but penalizes under or over predictions (see Table 1 below).

Table 1: MBS Score Function (true games won vs. predicted games won)

		Predicted games won				
		0	1	2	3	4
True games won	0	10	-10	-30	-70	-150
	1	-10	30	-20	-60	-140
	2	-30	-20	70	-40	-120
	3	-70	-60	-40	150	-80
	4	-150	-140	-120	-80	310

Note that this table ranges from 0 to 4 predicted games won, since in the history of the tournament, a low-seeded team has never won more than 4 games.

Below, we have the RMSE and MBS of the top 5 regression algorithms (Table 2), and the accuracy and MBS of the top 5 classification algorithms (Table 3). Note that "mean" signifies we filled in missing data values with the average value, while for "drop" we dropped features completely.

Table 2: Top 5 Regression Algorithms, by Mock Bracket Score

Regression	RMSE	Bracket Score
Linear, mean	40227.70	-140
Lasso, drop	2.40	-1360
OrdinalRidge, drop	3.62	-4120
OrdinalRidge, mean	3.62	-4120
LAD, drop	3.62	-4120

Table 3: Top 5 Classification Algorithms, by Mock Bracket Score

Classification	Accuracy	Bracket Score
Naïve Bayes, mean	0.71875	-140
Naïve Bayes, drop	0.625	-190
Linear SVM, mean	0.15625	-360
Linear SGD, mean	0.15625	-360
Linear SVM, drop	0.15625	-360

Immediately upon analysis of the results, we noted that regressions were a poor choice of algorithm type. This is because the predictions from the regressions would often be very high or low. This led to a very poor RMSE score for certain algorithms, or a very poor MBS for others (note that if the algorithm predicted negative games won, we would use 0 as the predicted games won, while if it exceeded 4, we would cap it at 4). This is the reason why "Linear, mean" had

extremely high RMSE values but a decent bracket score.

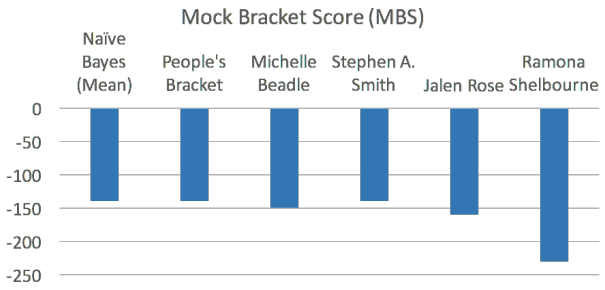

Figure 2: Bar Graph of MBS

To put this analysis in context, we compared the MBS of our best algorithm (Naive Bayes, mean) with the 4 ESPN analysts and the People's Bracket. Surprisingly, our algorithm outperformed all of them (tied with two, and beat the other three). For reference, a baseline MBS score would be -140, as this is the score obtained for predicting 0 games won for the entire tournament.

Upon close analysis of our data, we noticed that Naive Bayes was very conservative in its predictions. It essentially predicted only 0 or 1's; below, we show a sample prediction for 2016.

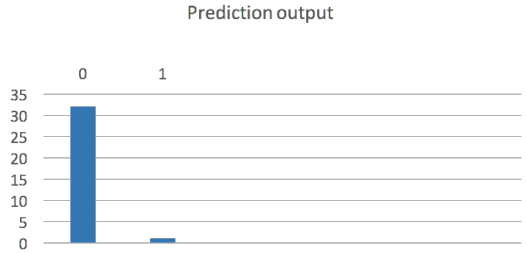

Figure 3: 2016 Prediction Output

To visualize this performance on different years, below in Table 4 is a table of Naive Bayes MBS for 2013 to 2017. Note that some scores are very positive while some are very negative; note that positive scores typically indicate a year with few upsets, while negative scores typically indicate a year with many upsets.

	2013	2014	2015	2016	2017
Naïve Bayes MBS	60	200	-100	180	-140

Table 4: Naive Bayes MBS for 2013-2017

3a. Feature Analysis

To get a better understanding of what features played a larger role in our results we performed several feature analysis tests. The first test comprised of comparing basic stats to advanced stats. We trained our Naive Bayes model on all the data up to a certain year using just basic features (stats) or just advanced features (stats) and predicted on that year's tournament bracket. Our results indicate that the basic features led to far more variance in terms of prediction as the following tables illustrate.

	2014	2015	2016	2017
Predictions	2: 3 1: 9 0: 20	2: 9 1: 9 0: 14	2: 11 1: 8 0: 13	2: 7 1: 11 0: 14
Accuracy	0.6875	0.5	0.406	0.5
Score	190	-70	-240	-150

Table 5: Outputs of Naive Bayes with Basic Features

	2014	2015	2016	2017
Predictions	2: 1 1: 3 0: 29	2: 1 1: 2 0: 29	2: 3 1: 5 0: 24	0: 32
Accuracy	0.8437	0.593	0.718	0.718
Score	220	-110	70	-140

Table 6: Outputs of Naive Bayes with Advanced Features

The results from above indicate that just on pure basic features alone the Naive Bayes model was far less conservative with its guesses. It would routinely predict a cinderella team winning 2 games, which is an extremely unlikely event. The advanced stats on the other hand were very conservative, picking more than 20 teams to lose in the first round each year. Overall, the advanced features performed better than the basic features with the exception of 2015. However, neither

feature set alone was enough to beat the results of the combined feature set. Hence, we decided look at feature sets that combine some basic features and some advanced features.

We derived three different features sets that combined both basic features and advanced features. The first feature type was denoted as the "Pure Offensive" feature set. It comprised of solely offense based features such as the number of offensive rebounds, three point percentages, true shooting performance, etc. The results of training on this feature set are in Table 7. For comparison purposes we are predicting on the 2014, 2015, 2016, and 2017 tournaments. Furthermore, when dealing with missing data we took the larger score of the two methods we described in Section 2.

	2014	2015	2016	2017
Accuracy	0.7187	0.345	0.156	0.625
Score	70	-80	-440	-180

Table 7: Outputs of Naive Bayes with "Pure Offensive" Features

These results seemed to indicate that just offensive stats were not a good predictor of a good bracket score. In fact, the Naive Bayes classifier predicted very sporadically with some teams predicted to win 3 games. While in 2014, this feature set outperformed the overall feature set, from 2015-2017 this feature set did not perform well.

The second feature type was denoted as the "Pure Defensive" feature set. It comprised of solely defense based features such as the number of steals, blocks, block percentage, etc. The results of training on this feature set are in Table 8. Furthermore, when dealing with missing data we took the larger score of the two methods we described in Section 2.

	2014	2015	2016	2017
Accuracy	0.874	0.593	0.781	0.687
Score	240	270	110	-150

Table 8: Outputs of Naive Bayes with "Pure Defensive" Features

The results from this feature were very strong. The results in 2014-2016 from this feature set were much better than the overall feature set. The outputs from this feature set were very conservative with the vast majority of teams predicted to not win at all. However, for each year the classifier seemed to have predicted some teams to win 1 or 2 games correctly. This created the really high scores shown in Table 8.

Seeing the result above we decided to keep all the defensive features and add some offensive features to get better results. We called this feature set the "Offense Defense" feature set. The offensive features we chose included true shooting percentage, assists and some others. The results of training on this feature set are in Table 9. Furthermore, when dealing with missing data we took the larger score of the two methods we described in Section 2.

	2014	2015	2016	2017
Accuracy	0.75	0.312	0.375	0.593
Score	180	-220	-240	-200

Table 8: Outputs of Naive Bayes with "Offensive Defensive" Features

Almost in all cases did this new feature set fail to outperform the overall feature set. It seemed to predict extremely sporadically as shown by the high negative scores displayed in Figure 7. Furthermore, the results of this feature set indicate the offense based features really confuse the Naive Bayes Classifier. They also indicate that the defense based stats tend to be more reliable when predicting team performance.

4. Related Work

Relevant literature to our projects includes Gumm, Barrett and Hu's paper titled "A Machine Learning Strategy for Predicting March Madness Winners.[3]" Gumm et al. primarily used regression to for predicting matchup victories on the NCAA tournament bracket. Their work also used many of the features that we had such as assists, three-pointers, etc. However, they explored different "advanced statistics", such as RPI rank difference and BPI rank difference, that may offer better insights than the advanced statistics we used. Our work improves on such results by using far more training data as well as exploring classification methods.

5. Future Work

From the results of our work, it is clear that it is very difficult to predict tournament performance simply from regular season statistics. In the future, with more time and data, a better approach may be to attempt to predict head to head results. The most important factor to a Cinderella team may just be the opponents that they face; perhaps analyzing specific matchups will yield a more definitive reason for unexpected tournament success.

6. Conclusion

Using regular season basic statistics and advanced statistics, as well as the tournament seed, we successfully used regression and classification algorithms to predict low-seeded teams' performance in the NCAA March Madness tournament. A Naive Bayes approach, coupled with using the mean to fill in missing features, yielded the best Mock Bracket Score (MBS), a scoring function we developed to emulate the ESPN Bracket Challenge scores. This algorithm tied the best performance of 4 ESPN sports analysts that published their brackets for the 2018 tournament, and matched our baseline standard.

Upon analysis of the Naive Bayes classifier predictions, we discovered that it took a very conservative approach - predicting 0 or 1 games won for almost every team in the tournament. It is evident that this method, compared to attempting to predict upset performances, is actually fairly successful.

Further analysis from our feature analysis indicated that the advanced stats tended to produce much more conservative results. We also discovered that the basic stats tended to introduce significant variance in our results. Our feature analysis also indicated that defense based stats tend to be a better indicator for cinderella team performance. These results fit our intuition as in college basketball the top 64 teams all tend to be fairly strong offensively. However, the best teams tend to be longer and more athletic leading to a better defensive rating.

7. Works Cited

[1] https://www.ncaa.com/news/basketball-men/bracket-beat/2017-03-14/march-madness-longest-perfect-bracket-streak-we-know

[2] https://pythonhosted.org/mord/

[3] https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7176206