2018 NCAA Tournament Summary

1. Introduction

This was the first year of my project to develop a model that can predict NCAA Tournament games as accurately as possible. As an Alumnus of Marquette University, a school with a feverish devotion to college basketball, I felt compelled to apply the skills I developed there to predict the results of college basketball games.

To begin with, a bit about me. This project has brought two sides of my life together. The first of those sides began when I decided that I would be attending Marquette University. During my time there (2013 - 2017) we were never very good, only making the tournament once, but I had season tickets all four years and the games were one of my favorite parts of a great experience at Marquette.

On the other side, I am interested in data science and I knew that a project like this would be the perfect marriage between my work and one of my favorite hobbies. While at Marquette, I took classes in the field including databases, artificial intelligence, data mining, probability, regression and intro to data science. I decided to continue my education by attending University of Wisconsin - Madison for a graduate degree in computer science. Since moving west to Madison, I have continued to take classes such as database systems, machine learning, and data science. I also work with professor AnHai Doan in the field of entity matching where we have used machine learning to help match tuples in data tables that refer to the same real world entities. My experiences, both at Marquette and Wisconsin, have been a great learning experience and in my time at these institutions, I learned all of the skills that made this project possible. As I continue my education and move on to industry in the future, I hope to improve my skills and further develop this project.

2. Methods

I generated my predictions by training a machine learning model on advanced statistical data and the results of games going back to 2001. I cleaned and joined the data sets, reduced the training set by removing obvious results, selected a machine learning model from a set of several different algorithms, and then finally was able to make predictions for each game in the 2018 NCAA Tournament.

2.1 Data Retrieval

The first task was to retrieve some data to analyze as a means to make educated predictions later on. The data came from two sources. The first, sports-reference, is a website that posts statistics for several sports for current and past seasons. I retrieved all the scores of division one men's college basketball games from the 2001-2002 season up to the current 2017-2018 regular season.

The other source of data for this project is Ken Pomeroy's college basketball ratings. Pomeroy has published his rating for each season dating back to the 2001-2002 season and that is the reason I have chosen to collect data going back as far as this season. Pomeroy's rating are designed to evaluate a team's efficiency instead of looking at absolute numbers. Since many basketball statistics are greatly affected by the team's tempo, or the number of possessions they have each game, the ratings are adjusted to compare how efficient a team is on a per possession basis.

2.2 Data Cleaning and Feature Generation

The only real data cleaning challenge for this project was matching the school names across the two different data sets. In order to match them, I compared all the school names present in each data set looking for matches. After this step, I had a list of the remaining names that did not match up between the two datasets. Since the Kenpom dataset is the shorter of the two, I created a function that would edit the school names to match the Sport-Reference data.

After the school names were matched between the two datasets, I was ready to join the them and generate features. Each row in the Sport-Reference contains the two schools playing in the given game as well as the score for both the home and away team. Considering that there are so many neutral site games throughout the regular season and the NCAA tournament games are also neutral site games, I wanted to de-emphasize the factor of which team was playing at home. Instead, I framed each game as a possible upset where the team with the higher Kenpom rating was considered the favorite and the team with the lower Kenpom score was considered the underdog. With this in mind, I generated three features for each attribute from the Kenpom data set: favorite score, underdog score, and the difference between the two. This way, I hoped to allow the model to consider the scores for each team as well as a direct comparison of the two teams in that particular attribute. Given how the features are framed, a game tuple is labeled as a 0 if the favorite won and a 1 if the game resulted in a upset.

2.3 Blocking

Before training and testing a machine learning model, I implemented a blocking scheme to try and improve the accuracy of the model. In this context, blocking refers to the process of removing obvious or irrelevant results from the training and test sets before training or running the model. The idea here is that if we can remove obvious results that are easy to predict, such as highest ranked team in the country playing a team ranked in the three hundreds, we can develop a more accurate model for the closer and more difficult to predict games that are often included in the tournament.

In particular, my blocking scheme contained two rules, that if triggered, remove the game from the given input set. The first rule checked if the difference between the AdjEm statistic from the Kenpom set for the two schools was greater than 15. This rule signifies that the teams are rated far enough part that we can be confident that the favored team will win. Of course, this is not always the case in practice, but just about any model will have difficulty with accurately predicting an upset in this type of game. Instead of focusing on these fringe predictions, I chose to just assume the favored team would win in this case to focus on accurately predicting closer matchups. The second rule checks if the Kenpom AdjEM of either team is below -10. Since the end goal of this project is to predict tournament games, I needed to make sure the training set only consisted of games between tournament quality teams. A model that can predict games between teams that are not good enough for the tournament is not useful for my goal. Therefore, I decided that it would be best to have the model focus only on games that would be closer to those that I would be attempting to predict.

Through the blocking stage I was able to reduce the number of games in the training set from 85,683 to 45,863. Ideally, this process would refine the training set to specifically cater to somewhat close matchups between tournament quality teams in order to have a model better suited to predicting tournament games.

2.4 Selecting and Training the Machine Learning Model

After the blocking stage I had a refined training set and could start working on selecting a machine learning model. I compared five classifiers, including KNN, decision tree, random forest, logistic regression and AdaBoost. I trained each model on all of my regular season data and then evaluated them on the march madness data acquired from Sports-Reference. In the end, I settled on using the AdaBoost classifier for my predictions because it had the best F1 score and the second best precision score. When predicting upsets, it is important to choose a classifier with high precision because I want to be a sure as possible that I am correct when the model predicts an upset. This is the case because any prediction made will affect the predictions in the later rounds. If the model makes too many upset predictions, it will most likely fail to predict the correct teams later in the bracket.

Once I had selected the AdaBoost model, I trained it on all of the data that survived the blocking step, including data from both the regular season and march madness datasets. I was then ready to use the model to make my predictions for the upcoming NCAA Tournament.

2.5 Tournament Predictions

At first, I intended to use the straightforward predicted labels from the classifier to make my predictions for the tournament, but I soon realized that the resulting bracket almost always just picked the higher seeded team to win the game. From my experience of watching the tournament every year, I knew that I needed to ensure that the model would pick some more upsets. To solve this issue, I needed to consider the most likely upsets, as given by the probability score outputted by the model for each prediction, as upset predictions in order to improve the bracket. While the model actually predicted a win for the favored team in these games, there was a high enough chance of an upset that I considered these games as an upset prediction. Specifically, whenever there was at least a 0.4985 probability score of an upset from the model, I considered the prediction to be an upset. I settled on this particular cut off by trying various values until I arrived at a bracket that I felt had a good balance between upsets without doing anything too out of the ordinary. Using the model with this update, I was able to create a bracket for the 2018 tournament that I hoped would be able to predict at least some of the madness that was sure to happen in March.

3. Results

Properly evaluating the success of the model proved to be difficult due to the limited number of games available to test its predictive ability. In this section, I comment on my bracket's ESPN results, the various interpretations of accuracy in this domain, and others evaluation metrics.

3.1 ESPN Results

I decided to submit this bracket to the ESPN Tournament Challenge as a further way to judge the success of my model. ESPN awards a bracket points for each correctly predicted win, with the number of points increasing each round from 10, to 20, 30, 40, 80, 160, and finally 320 points for correctly predicting the tournament winner. My bracket scored 1030 points, was ranked 2.0 millionth, which left me in the 89th percentile. My bracket gained a large boost from correctly predicting that Villanova would win and that largely left this as a flawed way to judge success, but overall I found it a fun exercise to examine how my bracket compared to the millions of other people who submitted a bracket to ESPN.

3.2 Accuracy

I also evaluated the proportion of games that I was able to correctly predict the winner, as seen in the table below.

Prediction Accuracy By Round

Round	Games	Correct Predictions	Total Accuracy	Adjusted Accuracy
Round of 64	32	21	0.656	0.656
Round of 32	16	7	0.438	0.583
Sweet Sixteen	8	3	0.5	0.75
Elite Eight	4	1	0.25	1.0
Final Four	2	1	0.5	1.0
Championship	1	1	1.0	1.0
Total	63	34	0.540	0.667

I found that in 34 of the 63 games in the entire tournament, or 54% of the time, my model correctly predicted the winner. However, this method of evaluation is flawed for games in later rounds because of the bracket format used to make predictions. In many of the games, my predicted winner did not even play in the game because they had already lost and therefore I had no chance to make a correct prediction. To help deal with this issue, I added another column to the table, called adjusted accuracy, to show the proportion of games that actually included the team I predicted would win. Using this metric, I was able to correctly predict the winner of games that included my predicted winner about 66.7% of the time. Unfortunately, even this metric is not perfect because some of these games pitted my predicted winner against a different team than

I had them playing against because my predicted opponent had lost in a previous round. As a final evaluation metric, I suggest looking at the performance of my bracket in the first round because my model is making a prediction knowing both teams playing in the game. Looking at the first round only, the model correctly predicted the winner in about 65.6% of games. This method still only provides some insight because first round games often include some of the most lopsided and easy to predict matchups, even with the incredible upset UMBC manages to pull off against number one seeded Virginia this year.

Considering the unpredictability of a sporting event called March Madness, I think the model is able to make predictions with a decent accuracy. While it only correctly predicted the winner of just over half of the total tournament games, it correctly was able to make predictions in the games actually including my predicted winner closer to 65% of the time. I think that this is a respectable start to this project because tournament games between college athletes can be so difficult to predict, but I think this is only a baseline to improve in future years.

3.3 Precision and Recall

As my final method of evaluation, I wanted to look at the precision and recall numbers for specifically my upset predictions. I think this is important to look closer at because, often, the different between a decent bracket and a great bracket is how many upsets it is able to correctly predict. Precision provides a good way to see how well the bracket makes each upset prediction and recall shows how many of the actual upsets the model correctly predicted.

Precision and Recall By Round

Round	Games	Number of Actual Upsets	Upset Precision	Upset Recall
Round of 64	32	9	0.333	0.222
Round of 32	16	8	0.5	0.125
Sweet Sixteen	8	4	0.0	0.0
Elite Eight	4	1	0.0	0.0
Final Four/Championship	3	0	0.0	0.0
Total	63	22	0.272	0.136

The results above show that there were 22 upsets in the 2018 NCAA Tournament where an upset is defined as when the team with the lower Kenpom AdjEM metric wins. The model was correct 27.2% of the time it made an upset prediction and it was able to correctly predict 13.6% of the total upsets in the tournament. Again, these numbers suffer from the fact that, in later rounds, my bracket did not even contain the upset winner in that round, but overall it did not make sense to create an adjusted version of these metrics because then there would not have been enough possible upset predictions in the later rounds. Overall, these number provide a good place to work on improving in future years and, despite the difficulty of predicting upsets in March Madness, I need to focus my efforts on finding a better way to predict these upsets.

4. Future Improvements

Being the first year of this project, I have only begun to implement my ideas for predicting the NCAA Tournament and I plan to improve my methods each year as I learn more machine learning and data science techniques. In particular, there are three main areas that I see as good places to improve my model: including more data, feature selection, combining multiple models and improving how I pick upsets.

4.1 Adding More Data Sources

For the upcoming season, I have plenty of potential improvements that I could make to my model that I hope to implement. To start with, I would like to gather more data to help improve the model. I currently only really use KenPom rankings as input to my machine learning models, but there are plenty of other sources of valuable data that could be used to improve the model. The obvious addition would be to get some team statistics on the year like shooting percentage or points per game. However, sometimes these simple stats can be a little misleading on their own. For example, shooting percentage does not always provide the full story because a team could have a lower shooting percentage because they shoot more threes.

These kinds of problems are addressed by more advanced statistics like effective field goal percentage as well as the models provided by outlets like KenPom. To combat these types of problems, the model could include more of these advanced statistics from sources other than KenPom. Ratings like T-Rank, Sagarin, Sonny Moore’s ratings, LRMC ratings, ESPN’s BPI, and FiveThirtyEight’s Elo could provide more than enough additional information. Unfortunately, these types of rating may not be enough because they are all based on team performance throughout the season.

Sometimes teams are able to win games with things that do not necessarily show up on the stat sheet. Things like talent level and the “eye test” are subjective and difficult to measure, but the model can get an insight into this type of information by considering recruiting rankings and pre-season polls. These two rankings can help the model consider how people judge the teams playing and these results in tandem with the purely statistical approach given by the advanced stats will hopefully help the models make more accurate predictions.

4.2 Feature Selection and Ensemble Methods

On the other hand, more data sources means more features and more complex models. I have two future additions that will help address these issues. Fist, feature selection is an excellent way to help reduce the number of features considered by each model. Sometimes a smaller set of better features will produce a more accurate model than one trained on every feature. Feature selection techniques will help me select only the best features for my models. The other improvement to address this problem will be to create an ensemble of models that each take into account different input data and features. I plan to experiment with different ways to accomplish this goal. My thinking at the moment would be to train a model with the features associated with each data source. I would have a model for the KenPom data, a model for the recruiting rankings, and so on. This would also address the fact that each data source varies on the amount of data available and how many years back the data goes. Then, I could use a weighted combination of the results from each model to get a final prediction.

4.3 Improve Upset Prediction Methods

The last improvement I hope to make is to find a better way to choose the most likely upsets. Since the models will generally predict the favored team to win, I need a way to make sure that the model predicts enough upsets. So far, I think the ensemble method could help in this regard because I could choose upsets based on how controversial a pick is, or how much the individual models disagree on who will win the game.