2019 NCAA Tournament Summary

1. Introduction

This was the second year of my project to develop a model that can predict NCAA Tournament games as accurately as possible. Last year, I was able to create a full pipeline to get from input data to tournament predictions and the results were promising. My bracket correctly predicted the tournament champion, but there was plenty of room for improvement in predicting upsets more accurately and with a higher rate. I also found that my model too often picked the favored team to win games. This year, I focused my efforts on trying to improve the ability of my model to more accurately predict upsets more often. I think I was successful in this goal, but there are still some ways I can improve this aspect of the model.

2. Changes From 2018 Version

For this year's bracket, I have added some improvements on top of the codebase from last year. The key changes for this season are extracting more data and updating the model selection process, including visualizations to evaluate the various machine learning models. I also made some smaller changes that are described in the miscellaneous section.

2.1 Data Retrieval

Unfortunately, the first major changes made for this version did not end up making it into the final model for this NCAA Tournament due to time restrictions. However, I still think it is worth describing these additions to the project because I will be sure to include them for next year.

In last year's summary of the project, I included some future goals I had in mind to improve the model. The first item on that list was to include more data sources for training and making predictions. The thought process here was that the Kenpom data may have been biased towards certain aspects of a team's performance or maybe did not paint a complete picture of a team's abilities. To help alleviate these potential problems, I thought that bringing in more data sources would help to fill in the gaps of information about each team that the Kenpom data was unable to provide. I looked to two different data sources to help with this issue. First, Bart Torvik's college basketball statistics could provide more advanced statistics and also more traditional advanced team statistics, such as effective field goal percentage and rebounding rates. T-Rank stats are somewhat similar to Kenpom, but due to their slightly different methodologies in computation, I think they could be used in tandem to reduce any biases in the data. Secondly, I extracted basic team statistics because I think they could be helpful to fill in any gaps in knowledge of each team left by only using the advanced tempo based statistics. Basically, the more data sources included, the more likely it is that my model has all of the necessary information about each team to make accurate predictions.

2.2 Model Selection

I also made some changes to my model selection process. I found during last year's work that the probability scores outputted by the model were much more important than I originally had thought because I discovered that a predictive model would almost never predict upsets. This makes sense because upsets are, by definition, unexpected and less likely to happen than the predictable favorite winning. My model selection technique last year was to select the model with the highest F1 score after a round of cross validation on my training data. While this promoted models that actually predicted upsets, due to the recall component of the F1 score, I realized while using the model to create my bracket, I needed to utilize the probability scores to find the most likely upsets in each round. With this in mind, I altered my model selection process to include a component of analyzing the probability scores to see if they actually reflected the chances that a game had of ending in an upset. Instead of just looking at precision, recall and F1 scores, I also tried to visualize how close the probability scores were to the chances that a game ended in an upset. These visualizations are described in more detail in section 2.3 below.

Another change to the model selection process was to make an alteration to the cross validation technique I was using to compare models. I wrote a function to perform, as I call it, "leave one march out" cross validation. This function takes a set of training feature vectors as input, where this data contains game feature vectors over several years including regular season and postseason data. Each fold of the cross validation leaves out one year's postseason data, trains an the rest of the data and finally evaluates the model on the previously left out postseason year of data. The hope is that this is a more accurate representation of how I make the final predictions because the test set will always only include actual NCAA Tournament game data.

2.3 Data Visualization

As part of the model selection process described above, I wanted to better evaluate how well the probabilities generated by the model actually represented the chance of a given game resulting in an upset. I created a bar chart visualization to help me see this relationship. The predictions made by the model are partitioned into equal sized bins based on the probability associated with the prediction. Then, each bar in the chart represents the accuracy of the predictions within its associated bin. The user can select any accuracy metric out of accuracy, precision, recall or F1 score.

I used this visualization to see how close the accuracy of predictions in each bin was to the range of probability values in that bin. For example, in a bin containing predictions with a probability score between 80 and 85 percent, a good model would correctly predict an upset close to 82 percent of the time. As I evaluated the best models after cross validation, I found that the probability scores of the logistic regression model were the most accurate and the best predictor of the chance of an upset.

2.4 Miscellaneous

In addition to the changes mentioned above, I made some smaller changes to other parts of the codebase. For data extraction, I began using the python package Beautiful

Soup to help with parsing html files. This will hopeful make the extraction functions more robust, especially since the code for extracting from Sports Reference was broken between the time last year's code was written and when I attempted to use it again this year.

Another change I made this year was to update the blocking scheme. Since I extracted data for all games in each season, I wanted a way to remove games from the training set that included teams that are not tournament quality. Last year I removed games where either team was too poorly rated by Kenpom's AdjEM ranking or where the difference in that stat between the two teams was too large. After taking a close look at the data, I found that I should use a different threshold to judge the quality of the favored team than I use for the underdog. This makes sense because the worst teams in the tournament are rarely, if ever, actually the favorite and therefore there is a much lower floor for the AdjEM statistic for favored teams than there is for the underdog. By finding two different appropriate minimum values for the favorite and the underdog, I was able to reduce the training set to a more accurate representation of tournament games.

The next change made this year was to improve the way I made sure each data set used the same name for each school. This step is important because they school names need to be the same before I can join different data tables together to create feature vectors for each game. Previously, I just created a dictionary from two lists of school names that could be used to change the Kenpom data to reflect the game data. However, with the inclusion of the T-Rank and basic statistics, I found it was better to keep the various school names in a CSV file where each row represents a school and each column represents the name for that school in a given data set if happens to be different from the Sports Reference game data. This file is just a cleaner way to organize the various school names than hardcoding lists in a python file.

3. Results

Overall, I am happy with the performance of the model this year, but it is inherently difficult to judge the quality of the model due to the relatively small number of games to predict in each tournament as well as the unpredictable nature of a tournament appropriately called March Madness. However, I think it is fair to say that the improvements made to the model this year have resulted in more accurate predictions.

3.1 ESPN Results

Like last year, I submitted this bracket to the ESPN Tournament Challenge to see how my model performed compared to other brackets. ESPN awards each 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 1370 points, was ranked about 121 thousandth, which left me in the 99th percentile. As I found last year, this is a flawed way to judge success for this project because the later rounds count for a significantly more even though they are much harder to judge and based more on luck than predictive ability. Since this bracket correctly picked the champion again this year, it gained a very large boost to its

score and rank and this makes it difficult to judge how well the bracket was actually put together. However, even before the later rounds, this bracket was regularly placed in the 97th percentile or higher and its relatively high accuracy through the early rounds further supports the claim that the bracket was able to make predictions better than most ESPN users.

3.2 Accuracy, Precision and Recall

While submitting the bracket to ESPN was a fun experiment, I found that looking at the accuracy of the model by round is a better way to evaluate the model. The table below shows the proportion of games my model predicted the correct winner (Total Accuracy) as well as the proportion of games where my predicted winner was actually playing that the model correctly predicted the Winn (Adjusted Accuracy). I think this adjusted accuracy is worth examining because in the later rounds, the team I predicted to win may not have even played and it does not really represent the model's predictive ability to count these games against it. Though, it is important to note that this metric does include games where my predicted winner was playing a different team than I believed they would be playing, so it is not a perfect representation. Ultimately, I think that the first round numbers are the most useful because the model is guaranteed to know exactly which two teams are playing and can make an informed prediction.

Prediction Accuracy By Round

Round	Games	Correct Predictions	Total Accuracy	Adjusted Accuracy
Round of 64	32	27	0.844	0.844
Round of 32	16	13	0.813	0.867
Sweet Sixteen	8	5	0.625	0.714
Elite Eight	4	2	0.500	0.667
Final Four	2	1	0.500	1.000
Championship	1	1	1.000	1.000
Total	63	49	0.778	0.831

The table shows that the model correctly predicted 84.4% of the games in the first round, 77.8% of the total games and had an overall adjusted accuracy of 83.1%. Given the unpredictability of the NCAA Tournament, I think this shows that the model is performing very well. However, I must note that this year's tournament had relatively few upsets compared to previous years, making it easier for predictive models to make correct predictions.

In addition to accuracy, I examined the precision and recall of my upset picks and the results are included in the table below. Interestingly, the second round contained no upsets and therefore the model obviously did not correctly predict any upsets in that round. Overall, the model had 87.5% precision and 63.6% recall in the first round and

60% precision and 52.9% recall in the whole tournament. I think these are decent numbers, especially in the first round where the model is guaranteed to know which teams are playing. As of now, the model does not predict many lower seeds (seeds smaller than 6) to be upsets in the early rounds, so I think finding a way to predict these upsets could be a good way to attempt to improve the recall. Again, the lack of as many upsets as usual may have contributed to higher precision and recall this year.

Precision and Recall By Round

Round	Games	Total Upsets	Upset Precision	Upset Recall
Round of 64	32	11	0.875	0.636
Round of 32	16	0	0.000	0.000
Sweet Sixteen	8	3	0.500	0.333
Elite Eight	4	2	0.500	0.500
Final Four	2	1	0.000	0.000
Championship	1	0	0.000	0.000
Total	63	17	0.600	0.529

3.3 Comparison to Last Year

Compared to last year, the model has improved significantly in all areas. The bracket performed better on ESPN's Tournament Challenge and had higher accuracy, precision and recall. While it is difficult to compare the model across different tournaments, especially considering how there were fewer upsets in this years tournament, I think it is fair to say that the improvements I made to the model this year have actually made a difference and improved the model's overall predictive ability. Even with this consideration, all metrics are significantly improved with this year's model.

4. Future Improvements

Due to the high level of difficulty in predicting the results of college athletics, there are always ways to improve the model. I have several concrete additions that I hope to be able to make to the model for next year's NCAA Tournament, including incorporating the new data I've already extracted, feature selection and ensemble methods.

4.1 Incorporating New Data Sources

As I stated in section 2.1, I have already written the code to extract data from T-Rank and Sports Reference's basic team statistics. However, I have not yet included this data into the model and this is clearly the first place to start next year. I believe that this data will be useful to help reduce any bias that might be present in the Kenpom data. While T-Rank provides an advanced statistic that is generally similar to Kenpom's metrics, the variations between the two could potentially be a useful prediction tool. The T-Rank data also includes more traditional advanced statistics such as effective field goal percentage that can provide more details than than Kenpom's overall

offensive and defensive ratings. I have also gathered basic team statistics such as points per game from Sports Reference. While there are clear issues with these types of per game statistics due to differences in team tempo, I think they could potentially provide a new perspective on teams in a way that Kenpom's per possession data cannot. For example, from my years of experience watching college basketball, I have the sense that Kenpom tends to slightly overrate teams with slower tempos and I believe that these more basic stats could help offset that bias in the Kenpom data. Of course, I could be (or rather probably am) wrong, but there is no harm in gathering more data for analysis to find out.

4.2 Feature Selection and Dimension Reduction

As more data is included in the model, I will need to find ways to select the best features and/or reduce the total number of features using methods like PCA. I think that feature selection could be a good place to start because it is more intuitive and easier to understand. I probably will have some redundant features or some features that are poor predictors of game results that could be removed to create a more robust and simpler model. Feature selection methods could be a good way to reduce the number of features and evaluate what kind of data is most useful.

On the other hand, I think that dimension reduction might be a more effective way to combine similar features together. Take for example the defensive rating provided by both Kenpom and T-Rank. These two metrics are both slightly different ways to evaluate the same aspects of a team. It might be more effective to project these two features onto a single dimension that will be used by the model than to have two mostly redundant dimensions. I think that it would be very interesting to experiment with PCA and see how it might be able to reduce the number of dimensions without sacrificing much information in the data.

4.3 Ensemble Methods

The other improvement to address the amount of new data 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 T-Rank data, 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.