NCAA Men's Basketball "March Madness" Prediction Using Random Forests

Ryan Egbert

December 15, 2020

---

1 Introduction

1.1 Summary

For Project 2, I decided to build a predictive model for the NCAA Men's Basketball tournament, otherwise referred to as "March Madness". Since early in my life, March Madness has been something that my family and friends have looked forward to with a huge amount of excitement. As soon as the first week of April ends, we found ourselves getting ready for the following March. To me, this event is three weeks of pure basketball.

As one of the most popular events during the entire year, many people decide to participate in a "Bracket Buster" challenge, where individuals try to predict the result of every matchup in the tournament. With 68 teams participating, this becomes quite the task.. Most of the time, the first round is the easiest to predict. After that, it becomes more difficult for a few reasons: the teams are better, some teams you predicted may not have advanced, and there could be a lot of emotional reasoning for choosing a specific team over another. These issues could be resolved by using a model that could choose your predictions for you.

1.2 Implementation

My model takes 20 years of NCAA basketball data - from the regular season and March Madness tournament - to train, validate, and test random forests which predict each individual game in a March Madness tournament. Using my model, I was also able to simulate the 2020 March Madness tournament that, unfortunately, was cancelled due to COVID-19.

2 Resources

The data used in this project came from a couple of sources. Since the scope of this projects exists deeper than tournament data, I needed to pull data from the NCAA regular seasons from the past 20 years as well. The data acquired from these sources is stored in the `data/` directory.

2.1 Source 1 - `data.world`

The tournament data was pulled from a `data.world` dataset of March Madness tournament results since 1985. Because of the sparsity of regular season data, I was forced to reduce this data from nearly 40 years to 20 years. This data provided me with the results of individual tournament games.

2.2 Source 2 - NCAA

The regular season data came from the NCAA website. This website included well organized regular season data (starting in 2000) in PDF format. I used the Python library `tabular` to parse through the PDFs and pull out all relevant information.

3 Process

My process consisted of 5 steps: data acquisition, data processing/cleaning, data manipulation, model creation, and model prediction.

3.1 Data Acquisition:

I acquired data from the sources listed in Section 2, `data.world` and the NCAA website. The research involved in this step was not too difficult, even though it did take a good amount of time to find the correct regular season data. One issue I ran into was that the NCAA only began using a standard organization pattern for their data in 2000. This then created the issue of not having as much data as I would have hoped.

3.2 Data Processing/Cleaning:

This step took the longest in my process. In order to have usable data, I had to parse through PDF files using a Pythong library, `tabular`. This created CSV files of information stored in tables, but it still needed some touching up. I used regular expressions and a few manual adjustments to correct some irregularities in my data after pulling it from the PDFs.

3.3 Data Manipulation:

Manipulating the data so that I had one source to pull from took a long time as well. I had to take data from the March Madness tournament history (individual games) and combine that with regular season results (record, RPI, ranking, etc.). By doing this, I was making it much easier to create a dataframe using Python's `sklearn` package.

3.4 Model Creation:

Finally it was time to create the model. Using `sklearn` let me streamline the process because of their built in `RandomForestClassifier`. I was able to read in my custom CSV file (`/src/csv/all_data_combined.csv`) and create a dataframe. I then manipulated the data by using one-hot encoding and created testing and training sets randomly. I used the `StandardScalar` to transform the data correctly and finally fitted it on my tesing data.

3.5 Model Prediction:

After all this work, I was finally able to predict results using my testing data. I began with one random forest, and tweaked the hyperparameters to find the best accuracy. With a single random forest, I was able to achieve a 75% accuracy. This forest consisted of 500 trees and used a 95/5 split on training and testing data.

However, that made me wonder if multiple forests could be used to achieve a higher accuracy. After this, I used a voting classifier consiting of 5 RFs with the following number of trees: 150, 250, 500, 750, 900. This classifier was able to achieve an accuracy of 77%, a slight increase, but not enough to make a huge difference.

4 Deliverables

In my submission, I have included the original data (`data/`), the source code, (`src/`), the manipulated data (`src/csv/`), pickled classifiers (`src/pck/`, the voting classifier is compressed in a zip file because GitHub would not let me upload the file due to its size ($\tilde{3}$00 MB)), and a few result visualizations (`results/`). The `results/` directory contains my simulation of the 2020 March Madness tournament. I used the original tournament seedings found on FiveThirtyEight's tournament predictions. Due to the nature of the tournament, I had to run my prediction round by round, while updating the CSV files as I went.
In the `src/` directory, there are 6 Python files.

- `get_data.py`: Used in acquiring data from PDFs
- `adjust_data.py`: Used in data processing. Parsed through all the data and saved the most important features. Saved all the data in a Python dictionary which was then pickled to be easily accessible.
- `combine_data.py` Used in data manipulation. Combined data from March Madness history and NCAA regular season history.
- `rewrite_data.py` Used in data cleaning. Used regular expressions to remove some unwanted characters.
- `predict.py` MODELS. Created, fitted, and predicted using testing and training data combined from `combine_data.py`.
- `predict_2020.py` Used to simulate the 2020 March Madness tournament round by round.

5 Running the Code

In `src/predict.py`, there exists all the code that is necessary to run the model. The user simply has to unzip the `top_clasifier_vc.pck.zip` file and then run `src/predict.py` from the root directory. The console should show results from the classification. There is a lot of commented out code in `src/predict.py`; all of it was necessary in creating the original classifiers.

6 Conclusion

Overall, I was content with how my model performed. I do believe that there are ways I could have improved it, however, due to the nature of sporting events, there is never a way to truly predict who will win a certain game.

I am glad that I chose this topic. I found myself constantly thinking about ways I could improve the model and its accuracy. I was able to implement a few of my own ideas which, I was pleased to find out, actually improved its performance. It was also interesting to see which features were most important when making a decision.

This project made me a better programmer, because it forced me to build something from the ground up. Instead of relying on starter code to point me in the right direction, I was forced to think deeper about how to implement certain things. I was also forced to find my own data, which proved to be one of the more difficult parts of this project. It pushed me to become better, something that I was grateful for.