

---

# ISyE 6740 – Spring 2025

## Final Report

---

Team (224) Member Names: **Vishnu Bindiganavile**

Project Title: **Predicting Men's College Basketball NCAA Tournament Winners**

### **1. Problem Statement**

Every year, the NCAA College Basketball Tournament, lovingly termed "March Madness", excites college basketball fans, casual sports watchers, and sports gamblers alike. This tournament, serving as the thrilling conclusion of the college basketball season, hosts 64 of the best performing college basketball teams to compete in a single-elimination playoff. Each tournament, the entire country flocks to create their own brackets, predicting the tournament outcome from their own expertise, whether for fun or to compete in high stakes bracket competitions. Many base their predictions based on gut feel, expert sports commentator predictions, or other heuristics such as team color or favorite players. The main reason why so many brackets end up "busted" after even the first or second rounds is because of the wildly chaotic nature of the tournament. Upsets, injuries, buzzer beaters, and other seemingly random events turn the tables on many teams. For example, the 2024 final match was between two number 1 seeds. However, the 2023 final match was between a number 4 and a number 5 seed.

I am interested in predicting the annual NCAA Tournament winners based on proven data and statistics, using objective, rather than subjective, means. Specifically, I focused my efforts on the Men's NCAA Tournament. I intended to quantify the chaotic nature of the tournament, in addition to using team, conference, and star player statistics, in order to predict the winner and the probability of various teams winning an NCAA Tournament. There is no inherent feature or group of features that is well known as being a great predictor in determining who the winner is. So, my analysis was also to be able to identify those features. In the end, I will show by final model's ability to predict the winner of the upcoming 2025 NCAA Tournament and the probability of each team winning to aid in my ability to create a promising data-driven tournament bracket.

### **2. Methodology**

The goal of this project was to identify the best classification and class probability prediction model to predict a tournament winner and the probability of various teams winning the tournament. Within my data, I selected whether or not a team won the NCAA Tournament as the response variable. However, I used my models to predict both whether or not a team won the NCAA Tournament and the probability of all teams winning the tournament. The reason that I predicted two responses is because of the inherent large class imbalance between winners and non-winners in tournaments. I obviously wanted to find out who would be the sole winner of any given tournament. However, because of the class imbalance, the more interesting goal was predicting the probability of each team winning. This provided me with a greater outlook when considering the chaos of the NCAA Tournament. I implemented my solution in the following manner:

First, I implemented feature selection using correlation and L1 regularization.

Next, I built five different sklearn models in order to evaluate which model would perform the best: KNN (K nearest neighbors), logistic regression, random forest, NBC (naive bayes classifier), and simple neural network (multi layer perceptron). All of these models were able to classify points, as well as provide probability for each class. I also use grid search and K fold cross validation in order to determine the best hyperparameters for each model.

Next, I evaluated my models using the following evaluation metrics for classification: accuracy and confusion matrices. I also evaluated my models using the following evaluation metrics for probability: ROC AUC score, log loss, and Brier score.

Finally, I used my selected final model to predict the winner for the 2025 NCAA tournament and the probability that many top teams had to win the tournament.

### 3. Data Collection and Preparation

The data source for this project is the College Basketball Reference<sup>1</sup>, which is a reference website containing college basketball team, player, tournament, and historic statistics for all Division I college basketball conferences and the NCAA tournament. The dataset overall consisted of four parts: (1) the tournament team information for each year, (2) the season statistics for each team each season, (3) information about statistics leaders for each season (points leader, assists per game leader, etc.), and (4) information about the AP Player of the Year for each season. These four parts were merged into one overall table for feature selection, model creation, and model evaluation.

The specific statistics that I considered for player statistics leaders were: points, points per game, total rebounds, total rebounds per game, assists, assists per game, steals, steals per game, blocks, blocks per game.

I collected all of this data for all seasons from 1985 to 2025 (not including the 2020 NCAA tournament, which was cancelled due to COVID-19) to create the dataset, by web scraping the website in an ethical and safe manner. I used the `requests` and `BeautifulSoup` packages to collect the necessary HTML files and parse them to place the information into `pandas` data frames.

For the 2025 player statistic leaders, I had to manually enter the data.

Tournament data was structured as follows:

	seed	school	year	champion
0	1	Georgetown	1985	False
1	16	Lehigh	1985	False
2	8	Temple	1985	False
3	9	Virginia Tech	1985	False
4	5	SMU	1985	False

Season data was structured as follows:

	year	conference	rank	school	overall_wins	overall_losses	overall_win_loss_percentage	conference_wins	conference_losses	conference_win_loss_percentage
0	1985	ACC	1.0	Georgia Tech	27.0	8.0	0.771	9.0	5.0	0.643
1	1985	ACC	2.0	UNC	27.0	9.0	0.750	9.0	5.0	0.643
2	1985	ACC	3.0	NC State	23.0	10.0	0.697	9.0	5.0	0.643
3	1985	ACC	4.0	Duke	23.0	8.0	0.742	8.0	6.0	0.571
4	1985	ACC	5.0	Maryland	25.0	12.0	0.676	8.0	6.0	0.571

Player of the Year data was structured as follows:

	year	player	school
0	2025	Cooper Flagg	Duke
1	2024	Zach Edey	Purdue
2	2023	Zach Edey	Purdue
3	2022	Oscar Tshiebwe	Kentucky
4	2021	Luka Garza	Iowa

Statistics leaders were structured the same as Player of the Year data.

I merged tournament and season data on the year and school for each respective row. Of course, I only included (year, school) pairs that participated in the tournament. I encoded information about Player of the Year and statistics leaders in the data frame by creating respective columns and setting them to 0 (if that school did not have the Player of the Year/stat leader that year) or 1 (if they did). Furthermore, each row contained a column that indicated whether or not that school won the tournament that year. That column was used as the response variable.

After full data preparation, the data frame used for feature selection was structured as follows:

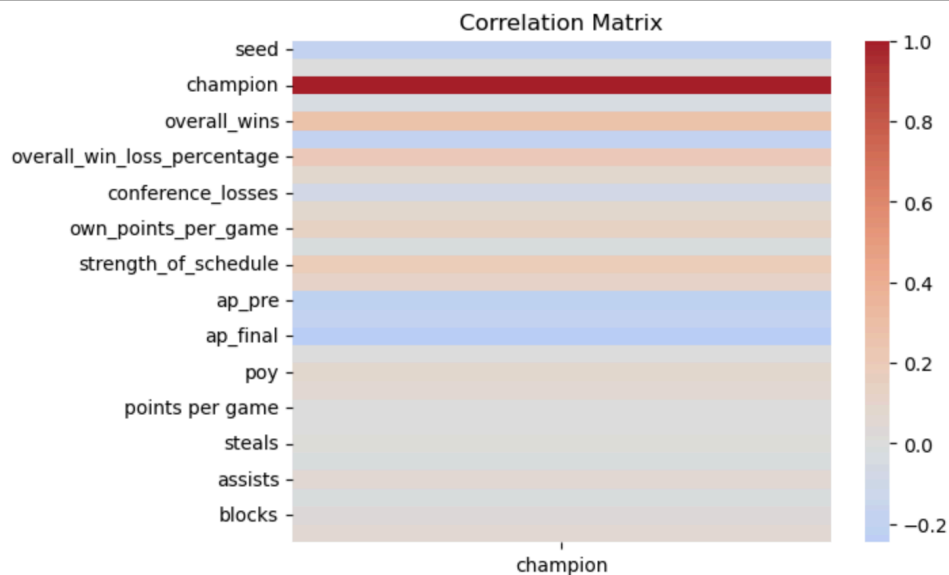
	seed	school	year	champion	conference	rank	overall_wins	overall_losses	overall_win_loss_percentage	conference_wins	...	points	points per game	rebounds
0	1	Georgetown	1985	0	Big East	2	35	3	0.921	14	...	0	0	0
1	16	Lehigh	1985	0	ECC	6	12	19	0.387	6	...	0	0	0
2	8	Temple	1985	0	A-10	2	25	6	0.806	15	...	0	0	0
3	9	Virginia Tech	1985	0	Metro	2	20	9	0.690	10	...	0	0	0
4	5	SMU	1985	0	SWC	2	23	10	0.697	10	...	0	0	0

#### 4. Feature Selection

I implemented feature selection using correlation and L1 regularization. In my project proposal, I also stated that I would use chi-square tests for this goal. However, once I realized that all of my features would be continuous, rather than categorical, I realized that chi-square tests would be inappropriate for this specific problem. This was extremely important because my initial dataset contained a large amount of features, many irrelevant. Although I collected statistics describing a variety of information that I thought may be relevant, feature selection allowed me to understand what was truly important when predicting NCAA Tournament winners.

I first calculated the correlation of each feature to the indicator of a tournament winner, which can be seen below.

	champion
seed	-0.179994
year	-0.001026
champion	1.000000
rank	-0.049754
overall_wins	0.271552
overall_losses	-0.176920
overall_win_loss_percentage	0.200456
conference_wins	0.074411
conference_losses	-0.074594
conference_win_loss_percentage	0.078920
own_points_per_game	0.134121
opp_points_per_game	-0.011550
strength_of_schedule	0.195906
simple_rating_system	0.117946
ap_pre	-0.220685
ap_high	-0.190824
ap_final	-0.245277
conference_tournament_champion	0.001068
poy	0.085634
points	0.067850
points per game	-0.006140
rebounds	-0.005012
steals	0.012210
steals per game	-0.010358
assists	0.045658
assists per game	-0.014027
blocks	0.031719
blocks per game	0.042450

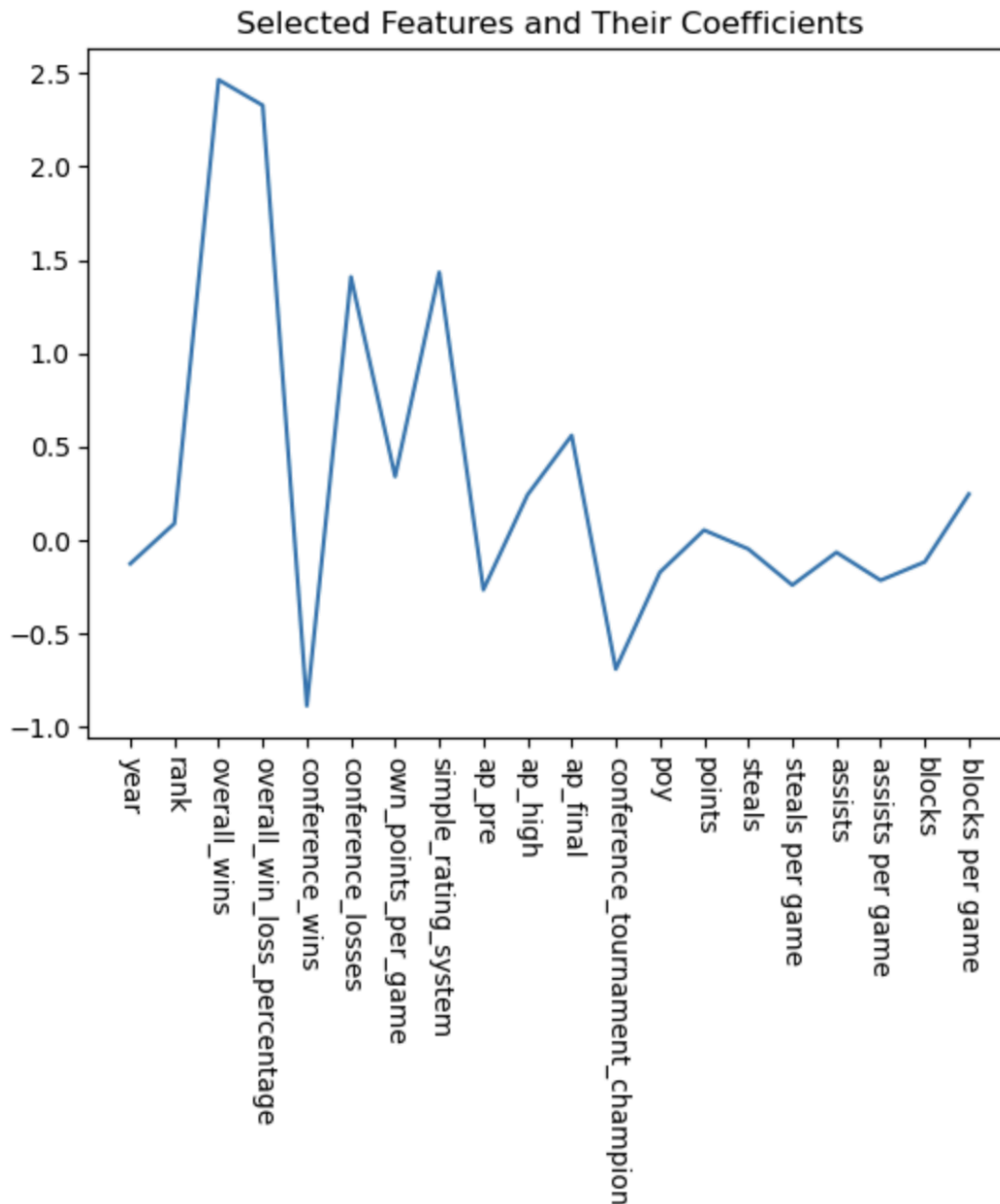


I then set a correlation threshold of 0.1 for feature selection to select the following features (0.1 was used instead of a more common threshold of 0.3 or 0.5 because of the comparatively low threshold for all features):

	champion
seed	-0.179994
champion	1.000000
overall_wins	0.271552
overall_losses	-0.176920
overall_win_loss_percentage	0.200456
own_points_per_game	0.134121
strength_of_schedule	0.195906
simple_rating_system	0.117946
ap_pre	-0.220685
ap_high	-0.190824
ap_final	-0.245277

Next, I computed L1 regularization to select features. I used logistic regression (w/ L1 regularization) instead of LASSO regression due to the fact that my response variable is categorical and not continuous. As a result the following features were selected.

```
array(['year', 'rank', 'overall_wins', 'overall_win_loss_percentage',
      'conference_wins', 'conference_losses', 'own_points_per_game',
      'simple_rating_system', 'ap_pre', 'ap_high', 'ap_final',
      'conference_tournament_champion', 'poy', 'points', 'steals',
      'steals per game', 'assists', 'assists per game', 'blocks',
      'blocks per game'], dtype=object)
```



One major difference that I saw, from a conceptual point of view, was that L1 regularization removed the seed feature, while correlation kept it. When I brainstormed what features would be required for prediction, the first one that came to mind was the seed feature. It makes sense that a 1 seed in the tournament has a far greater chance of becoming a champion than a 16 seed. However, L1 regularization determined that the other features described a team's seed based on their ranks and that there was no need to specify it separately.

L1 regularization also determined that most of the statistical leader indicators were important, while correlation determined that they were not. What was surprising was that the steals per game and assists per game indicators were slightly negative. It would make more sense that a team with any statistical leader has a higher likelihood of winning the tournament. However, as

L1 regularization seems to indicate, teams with statistical leaders do not have a historically high likelihood of winning the tournament.

Ultimately, I determined that the feature selection from L1 regularization would take into account both the relationship between the features and the response, and the relationship in between the features. Thus, L1 regularization feature selection was used.

## 5. Model Creation

For all five models, I tuned the model hyperparameters using grid search and k fold (5 folds) cross validation. Specifically, I used stratified k fold cross validation because it ensures that each fold has a proportionate number of labels in each fold. This is especially important when the data has a large class imbalance, such as in my case.

For each model, the hyperparameters I tuned and the choices I used are as follows (the red choice indicates the hyperparameter value in the best model).

### *KNN:*

- n\_neighbors: 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15
  - I wanted to determine if the model performed better if it was more simple or complex. Of course, I didn't want the model to be too complex, so I started the grid search with 5 neighbors.
- weights: uniform, distance
  - I wanted to determine if using equally weighted neighbors or distance-weighted neighbors would lead to better performance. Unsurprisingly, distance-weighted neighbors performed better. This makes sense because the farther away teams are away from each other, statistically, the less similar they should be. Your result should match teams that are statistically similar to you.

### *Logistic Regression:*

- C: 0.1, 1, 10
  - C represents the inverse of the regularization strength, so I used the default (1) and two values that are even jumps from the default.
- solver: liblinear
  - I simply used liblinear as the solver because the dataset is relatively small and the sklearn library described that option as the best for my scenario.

### *Random Forest:*

- n\_estimators: 50, 100, 200
  - I simply used the default value of 100 and two reasonable other choices (one less and one more). Since this is a complex problem, it makes sense that more estimators performed better.
- max\_depth: None, 5, 15
  - I wanted to determine if it was useful to have a max depth and if so, how deep each tree should be allowed to go. Again, this complex problem requires deeper trees in the forest.
- min\_samples\_leaf: 1, 2, 5
  - I wanted to determine how many teams should be at each leaf node.

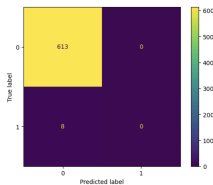
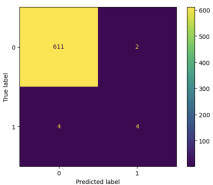
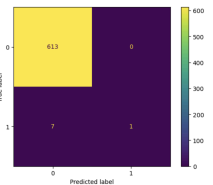
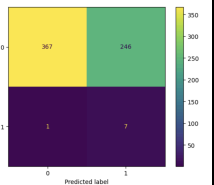
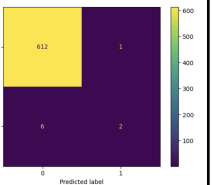
### Naive Bayes:

- There were no hyperparameters to tune, so I did not use grid search of K fold cross validation for this model.

### Neural Network:

- hidden\_layer\_sizes: (50,), (100,)
  - I wanted to test the default of (100,) and also see if using slightly lower hidden layer sizes would work better.
- activation: **logistic**, tanh, relu
  - I wanted to try all of the options.
- solve: lbfgs, sgd, **adam**
  - I wanted to try all of the options

## 6. Evaluation

	KNN	Logistic Regression	Random Forest	Naive Bayes	Neural Network
<b>Accuracy</b>	0.987	0.990	0.989	0.602	0.989
<b>Confusion Matrix</b>					
<b>ROC AUC Score</b>	0.981	0.939	0.997	0.851	0.962
<b>Log Loss</b>	0.033	0.034	0.027	8.043	0.034
<b>Brier Score</b>	0.009	0.007	0.007	0.389	0.008

On all fronts, the Naive Bayes classifier performed miles worse than the other four models.

From a classification standpoint, logistic regression had the highest accuracy and classified the most true tournament winners correctly. From a class probability standpoint, the random forest model provides the best performances with the highest ROC AUC score and the lowest log loss and Brier score. Given that my data set has a large amount of class imbalance, it makes sense to place more weight on the class probability performance than the classification performance. It helps that random forest provides a comparable classification performance to logistic regression.

Therefore, my final selected model is: Random Forest (n\_estimators=200, max\_depth=None, min\_samples\_leaf=2).

## 7. Predicting the 2025 NCAA Tournament Winner



After using the 2025 NCAA Tournament data, my model correctly predicted that the team with the highest probability of winning (at 44.1%) would be the: Florida Gators.

## **8. Future Work**

The model has a lot of room for improvement. There are more features than can be included/considered and other gut-based statistics to include (i.e., blue bloods). They may perform better than statistical features since the NCAA Tournament is a largely chaotic and surprising event. Oftentimes, the logical outcome does not occur.

I would like to modify this model to predict the outcome of each game in the tournament, instead of just the winner. This would require a much more streamlined and accurate model that is able to predict upsets.

## **10. Reference**

1. <https://www.sports-reference.com/cbb/>