

A mathematical framework for improving March Madness models

by

Dennison Jackson

B.S., Kansas State University, 2024

A THESIS

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2025

Approved by:

Major Professor
Michael Higgins

Copyright

© Dennison Jackson 2025.

Abstract

Every year, millions of people attempt to predict the results of the NCAA Division I Men’s Basketball Tournament (“March Madness”), often turning to models or team rankings to help inform their decisions. We propose a mathematical framework capable of taking any model that can return the probability that some team i beats some opponent j and returning round-specific win probabilities and expected wins. We illustrate our framework by fitting models to 24 years of historical tournament data using the differences in the teams’ stats, and find that they all outperform a traditional seed-based “chalk” model. Two of our best models include an XGBoost model with a median accuracy and total points of 66.67% and 116.5 respectively and an Elastic-Net model with a median accuracy and total points of 73.02% and 115 respectively. We also rank teams using expected wins from the XGBoost model and find that 22 of the past 25 champions ranked within our top 4, with 11 ranked first. In addition, we apply our framework to a knapsack bracket competition, which involves optimizing team selection under a cost constraint. The solutions of an Elastic-Net model average nearly 23 wins and capture almost 70% of the maximum achievable win total.

Table of Contents

List of Figures	v
List of Tables	vii
1 Introduction	1
2 A Framework to Estimate Probabilities and Expected Wins	5
2.1 Generating a Probability Matrix $\mathbf{P}$	5
2.2 Calculating Probabilities and Expected Wins	7
2.3 Making Game Predictions Using Stat Differences	9
2.3.1 The Dataset	10
2.3.2 Model Fitting and Tuning	11
2.3.3 Model Evaluation	15
2.3.4 Results in the 2025 Tournament	21
2.3.5 Model Results as Ranking Methods	22
3 Knapsack Bracket Competitions for March Madness	29
3.1 Introduction	29
3.2 Solving the 0-1 Knapsack Problem	31
3.3 Knapsack Model Results	34
4 Discussion and Future Work	38
Bibliography	41
A Supporting Materials for Model Fitting, Tuning, and Evaluation	44

List of Figures

2.1	Left: Results of feature selection for our Random Forest model when Seed is included as a predictor. The model with all 18 predictors yields the largest AUC (labeled as ROC in the plot). Right: Tuning results for our Random Forest model. Each line represents a value for <code>ntrees</code> across the tested <code>mtry</code> values. The largest AUC (labeled as ROC in the plot) is where <code>ntrees</code> = 100 and <code>mtry</code> = 5.	13
2.2	Boxplots of champion ranks, grouped by model. “Avg” refers to the average rank, calculated by averaging the ranks of the other five models. Left: Teams are ranked by their expected wins. Right: Teams are ranked by their probability of winning the championship.	23
3.1	Instructions for the 2015 Knapsack Bracket Competition held between Kansas State University and Michigan State University.	30
3.2	Left: Boxplots of total wins grouped by model. Right: Boxplots of proportions of total wins captured grouped by model. The Elastic Net model’s results look to be slightly higher than all of the other models.	36
A.1	Top: Median total points for each cutoff value grouped by model type. Bottom: Median overall accuracy for each cutoff value grouped by model. . . .	47
A.2	Distributions of total points from 2010 to 2024 grouped by model and decision method. Note that the high seed model is identical for all decision methods, as it really has it’s own decision method: pick the higher ranked team every time.	48

A.3	Distributions of overall accuracies from 2010 to 2024 grouped by model and decision method. The high seed model is identical for the same reason as Fig. A.2.	48
A.4	Left: Distribution of total points from 2010 to 2024, grouped by model. Right: Distribution of overall accuracies from 2010 to 2024, grouped by model.	49
A.5	High Seed is included as its own decision method in this plot. Left: Distribution of total points from 2010 to 2024, grouped by decision method. Right: Distribution of overall accuracies from 2010 to 2024, grouped by decision method.	49

List of Tables

2.1	The optimal values for each parameter of the XGBoost model, determined using grid search and LOYO-CV.	14
2.2	Aggregated model results over every tournament from 2010 to 2024. “exp_wins” refers to using expected wins to make predictions, “prob” refers to using round probabilities, and “h2h” refers to using head-to-head probabilities. Med_Pts and Med_Acc refer to Median Total Points and Median Overall Accuracy respectively. SD_Pts and SD_Acc stand for the standard deviation of total points and overall accuracies respectively. HS stands for “High Seed” and refers to the baseline model that only picks the higher ranked team for each matchup. Models are ranked by Med_Pts, then Avg_Pts, then Med_Acc, then Avg_Acc, all in descending order. For a visualization of these results, see Fig. A.2 and Fig. A.3	18
2.3	Model results from 2010 to 2024 aggregated by model type. HS stands for “High Seed” and refers to the baseline model. The results are in the same order as the ones in Table 2.2 . For a visualization of these results see Fig. A.4 .	20
2.4	Model results from 2010 to 2024 aggregated by decision method (round probabilities, head-to-head probabilities, or expected win). HS stands for “High Seed” and refers to the method of picking the higher ranked team in every game. The results are in the same order as the ones in Table 2.2 . For a visualization of these results see Fig. A.5	21
2.5	Model results for 2025, ordered by points first, then overall accuracy (both in decreasing order). HS refers to the “high seed” baseline model.	22

2.6	2025 team rankings by expected wins, using an XGBoost model.	27
3.1	Knapsack results by model over 2010 to 2024. “Best” refers to the optimal solution, using the teams’ actual wins as values. The “Prop” columns refer to the proportion of the maximum possible wins a model captures. The models are ordered by median cumulative wins.	35
A.1	A description of all the variables used in the training data. All differences are calculated by subtracting the lower ranked team’s stat from the higher ranked team’s stat.	45
A.2	The parameters and their values used in the grid search for tuning the Random Forest model.	45
A.3	The parameters and their values used in the grid search for tuning the XG-Boost model.	46

Chapter 1

Introduction

The NCAA Division I Men's Basketball Tournament, commonly referred to as "March Madness", is a competition where 64 of the best men's college basketball teams compete against each other at the end of each season. Teams are split into four regions, and the 16 teams in each region are assigned a seed. The four top teams in each region are given a 1-seed, the next best get 2-seeds, and so on until the four 16-seeds. The tournament is single-elimination, meaning teams compete in successive rounds where only the winners of a single game in each round advance until there is only one team remaining.

This tournament has become a major event that millions of people participate in by filling out brackets before the start of the tournament, picking which teams will win in every round. People often compete in groups to see who has the best bracket, and in many of these groups there is money being wagered. The American Gaming Association has estimated that in 2024 there were 2.72 billion dollars bet on March Madness, while in 2023 there were a whopping 15.5 billion dollars wagered¹.

With money on the line, many players may turn to outside sources to help them make their decisions. Some may turn to the Associated Press (AP) Poll², the Basketball Power Index (BPI)³, or the Pomeroy College Basketball Rankings (more commonly referred to as KenPom)⁴, which are all polls and ranking systems that provide insight into which teams

may be dominant in the tournament. Some of these methods, such as the AP Poll, are purely based on the opinions of broadcasters, sportswriters, or coaches. Ranking systems like BPI or KenPom are more mathematical and based on a team's data throughout the season.

Predicting the outcomes of sporting events is a common subject of research. As March Madness is one of the most popular contests involving the prediction of game outcomes, there are many papers revolving around predicting games in this tournament specifically. Kvam and Sokol (2006)⁵ propose using logistic regression to obtain transition probabilities for a Markov Chain model (referred to as an LRMC model). They collected data on which teams played in each game, who was the home team, and the margin of victory over four seasons of games. The LRMC model was found to outperform all other commonly used methods for ranking teams / determining winners such as the aforementioned polls or mathematical models. In Brown and Sokol (2010)⁶ a significant improvement was made to the model by using an empirical Bayes method to calculate the transition probabilities.

There have also been recent efforts to apply deep learning and machine learning methods to predict game outcomes. Kim et al. (2023)⁷ use algorithms such as an artificial neural network (ANN) and random forest to classify games primarily based on non-box score statistics such as a team's proximity to game location and average home game fan attendance. Their ANN model performed the best, correctly predicting 67% of game outcomes. Alfatemi et al. (2024)⁸ also train deep learning algorithms, but use combinatorial fusion analysis to build ensemble models, which was found to increase model precision in some cases.

Kaplan and Garstka (2001)⁹ were among the first to examine March Madness results in the context of "office pools" with a focus on points over accuracy for brackets. In a point-based scoring system, players are rewarded more for making correct predictions in later rounds of a tournament, and Kaplan and Garstka sought to try and maximize a bracket's score. They also propose a similar method to us for calculating expected wins for teams, though they use results from existing ranking systems to make their predictions. Their best model accuracies were around 58%, but in a point-based scoring system they were able to

outperform a traditional seed-based model.

In this paper, we propose a mathematical framework that is able to derive predictive measures such as the probabilities of teams winning in each round of March Madness and their expected number of wins in the tournament. These measures can also be used as a ranking system that can be used to support decisions in the bracket-filling process. In addition, we have developed models based on historical data from past tournaments that can be plugged into this framework. These models not only achieve high predictive performance on past tournaments, but can also provide ranks for teams that are consistently able to rank national champions highly and determine what teams may be over or under-ranked relative to their seeds. Finally, we also examine a knapsack bracket competition and how our framework can be used to solve an 0-1 knapsack problem to maximize team wins under a cost constraint.

The primary contribution of this thesis is our mathematical framework, which is able to take any predictive model’s win probabilities and convert them into round-specific probabilities of victory and expected wins. This is most impactful for those who have preexisting models or are interested in creating their own models to predict game outcomes, as it provides more flexibility when it comes to how a model makes game-by-game predictions or how it ranks teams in the tournament. Additionally, there is no reason this framework must be limited to college basketball; the general idea can be applied to any single-elimination tournament (e.g., the NFL playoffs or the FIFA World Cup).

A secondary contribution is our own model results, which all demonstrated superior performances over a conventional “high seed” model where all the higher ranked teams are selected in every matchup. This performance extends beyond just straight game-by-game predictions; it also extends to team rankings. We found that our models can also consistently rank past champions very highly, such as one model that has ranked 22 of the past 25 champions in its top four. These models can act as useful resources for anybody filling out a bracket, similar to other established ranking systems like KenPom.

Additionally, we believe that this paper provides the first exploration of knapsack bracket competitions (KBCs), a unique style of tournament where players are given a budget and purchase teams with the goal of achieving the most cumulative wins among their purchased teams. Our analysis can potentially open the door for future research into KBCs or other alternative competition formats that are not as widely studied.

Chapter 2

A Framework to Estimate Probabilities and Expected Wins

In this chapter, we will cover a framework by which the probabilities of winning in any given round as well as the expected wins for each team can be calculated. This methodology relies on the prior computation of a probability matrix $\mathbf{P}$. This matrix can be filled with probabilities from any predictive method that can return the probability that some team i will beat some team j . In Section 2.1, we will discuss how to create $\mathbf{P}$. In Section 2.2, we will discuss how to use $\mathbf{P}$ to calculate the probabilities of any team winning in any given round, as well the number of expected wins for each team. In Section 2.3 we will showcase our own models to demonstrate how this framework can be applied to directly predict game outcomes and/or rank the teams in the tournament.

2.1 Generating a Probability Matrix $\mathbf{P}$

To estimate round-specific win probabilities and expected number of wins for each team, we must first compute a probability matrix $\mathbf{P}$, shown in Eq 2.1.

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} \\ \vdots & \vdots & \ddots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} \end{bmatrix} \quad (2.1)$$

where $p_{i,j}$ is the probability that team i wins against team j . $\mathbf{P}$ will be an $n \times n$ square matrix where n is the number of teams in the tournament. For March Madness, $n = 64$ so $\mathbf{P}$ is a 64×64 matrix, and will account for the probabilities of each team winning against every other team. These probabilities can be determined by any methods that can generate a probability of victory for a team in any given matchup of two teams; we will discuss our method in Section 2.3.

It is important to note that there are several assumptions that must hold with these results so that our framework in Section 2.2 functions as intended and gives accurate results.

- All the 64×64 values in the matrix must be filled.
- All probabilities must be between 0 and 1.
- For all pairs of probabilities $p_{i,j}$ and $p_{j,i}$:

$$p_{i,j} + p_{j,i} = 1. \quad (2.2)$$

What this last point says is that the probability that team i beats team j plus the probability that team j beats team i must always equal 1. This makes sense logically, as if two teams play against each other, one must win; there are no ties. Depending on the method used to calculate probabilities, this assumption is likely to already be true, but it is important to check as the framework will break if it is not.

2.2 Calculating Probabilities and Expected Wins

To estimate the probabilities of winning in a round as well as expected wins, we take a mathematical approach that seeks to derive results using the probabilities in $\mathbf{P}$ and basic properties of probabilities. The general idea is that for any given round r , the probability that team i wins in round r will be the probability that team i made it round r times the sum of the probabilities that team i beats all possible opponents j they could face in round r given that opponent j also made it to round r . Let us define this probability that team i wins in round r as $\pi_{i,r}$. We can express $\pi_{i,r}$ mathematically as follows:

$$\pi_{i,r} = \pi_{i,(r-1)} \sum_{j=1}^{n_r} p_{i,j} \pi_{j,(r-1)} \quad (2.3)$$

where n_r is the number of potential opponents team i can face in round r , j is an index for each of the n_r potential opponents ($j = 1, \dots, n_r$), $\pi_{i,(r-1)}$ is the probability that team i won the previous round, and $\pi_{j,(r-1)}$ is the probability that team j won the previous round.

For any round r , we need to know the number of opponents n_r that any team can face. Due to the nature of the tournament, this number doubles every round. Mathematically, we can express this as

$$n_r = 2^{r-1} \quad (2.4)$$

Accordingly for any team i , there is one possible opponent in the Round of 64, two possible opponents in the Round of 32, four in the Sweet 16, eight in the Elite Eight, sixteen in the Final Four, and thirty-two in the Championship.

To illustrate Equation 2.3, let us look at what the first two rounds of the tournament could look like for some team i . In the first round, there is only one possible opponent for team i , so $n_r = 1$. We also know that both π probabilities will be 1, as the probability of “winning” Round 0 and making it to the first round is 1 for all teams, as they are all already in the first round to start the tournament. This means that the probability that team i wins

the first round is:

$$\pi_{i,1} = \pi_{i,0} \sum_{j=1}^1 p_{i,j} \pi_{j,0} = 1 \sum_{j=1}^1 p_{i,j} 1 = \sum_{j=1}^1 p_{i,j} = p_{i,j} \quad (2.5)$$

This makes sense logically as team i only has one opponent to face in the first round, so the probability they win the first round is just the probability that they win against that opponent.

Now, let us examine the probability that team i wins in the second round, $\pi_{i,2}$. In the second round, there are $n_r = 2$ potential opponents team i can face. This gives us the formula:

$$\pi_{i,2} = \pi_{i,1} \sum_{j=1}^2 p_{i,j} \pi_{j,1} \quad (2.6)$$

Let us express this probability in words. For clarity, say that team i is Team A, they won against Team B in the first round, and they are going to face either Team C or Team D in the second round. The above equation can then be written as:

$$\begin{aligned} P(\text{Team A wins Round 2}) &= P(\text{Team A wins Round 1}) \times [P(\text{Team A beats Team C}) \times \\ &\quad P(\text{Team C wins Round 1}) + P(\text{Team A beats Team D}) \times \\ &\quad P(\text{Team D wins Round 1})] \end{aligned} \quad (2.7)$$

This is then expanded upon for each future round.

We are also interested in calculating the expected number of wins for each team in the tournament. Say that W_i represents the number of wins for team i across multiple realizations of the tournament based on our round-specific win probabilities. We could then label a team's expected win as $E(W_i)$. We can calculate this by adding all the $\pi_{i,r}$ probabilities for any team i . This works due to the Linearity of Expectation, which states that the expected value of a sum of a random variable is the same as sum of the individual variables' expected

values. $\pi_{i,r}$ is the expectation for the individual round r , so the expected number of wins $E(W_i)$ for any team i can be found using Eq 2.8:

$$E(W_i) = \sum_{r=1}^6 \pi_{i,r} \quad (2.8)$$

This framework provides a way to rank teams as well; one could rank the teams by their expected wins or by the highest probability to win the championship or make it to the final four, for example. This is also an easy way to see what teams may be under- or overrated relative to how they are ranked in the tournament; for instance, in the 2025 tournament, one of our models had Memphis (CA) ranked 36th overall in terms of expected wins. Memphis was a 5-seed in the tournament, but our ranking placed them below several 9-, 10-, and 11-seeds. Sure enough, they ended up losing in the first round to the 12-seed Colorado State (ranked 37th, above some 9-, 10-, and 11-seeds). Even though Memphis may have been ranked higher than their opponent within our own rankings, it is still telling that they are far lower than we would expect from a 5-seed and that we should maybe be cautious about taking them too far in the tournament.

Along with the mathematical framework, we also developed a custom R function that can simulate any specified number of March Madness tournaments using **P**. When simulating a large number of tournaments and then combining all the results, we see almost identical results to the mathematical approach, which is what we would expect. The mathematical function is much faster and has no element of random chance, so we prefer it over simulation.

2.3 Making Game Predictions Using Stat Differences

In this section, we will demonstrate how the above framework can be applied to models we have created. In total, we tested five different models and three different methods of selecting game winners for each model using data from past tournaments.

2.3.1 The Dataset

The data we used was manually collected from the Sports Reference website¹⁰ and contains data from every game in March Madness tournaments from 2000 to 2025. For each team in each game, seventeen variables revolving around their season statistics are collected, including FTpct (free throw percentage), TRB (total rebounds), Drtg (defensive rating), and more. After comparing models that use a team’s seed as a predictor versus those that do not, it appeared that including it caused model performances to increase, so seed was also used as a predictor.

In any given matchup, we label the higher ranked team as Team A and the other as Team B. This means that for each game, our original dataset has 36 variables of interest where each “A” variable has a corresponding “B” (e.g., there will be A_FTpct and B_FTpct for Team A and B’s free throw percentages respectively).

To obtain our final variables, we take the difference of each pair of “A” and “B” variables. This causes our final data to drop back to 18 variables, now all being differences between each pair (e.g., $\text{diffFTpct} = \text{A_FTpct} - \text{B_FTpct}$). The reasoning behind this is that looking at variable differences gives us a direct way to compare the two teams’ stats to each other; if diffFTpct takes on the value 3 we could say that Team A’s free throw percentage is 3 percent higher than Team B’s. If the value is negative, that means Team A had a lower value. To keep the meaning of the values consistent, we always subtract the lower ranked team’s (Team B’s) values from the higher ranked team’s (Team A’s) values. Table A.1 in Appendix A contains all the variables along with their descriptions.

Our final training dataset consists of 1,511 rows with 18 explanatory variables. Each row represents one tournament game played between 2000 and 2024, and all the predictors are differences between the two teams’ stats. To train the models, one year at a time was excluded from the dataset and made the test data, and the models were trained on the remaining years of data. Then, predictions were made on the year that was left out and results were recorded. An additional 63 rows of data for the 2025 tournament were also

collected, but were saved for testing and were not used in any part of the training process.

2.3.2 Model Fitting and Tuning

Using this data, we trained five different types of models using three different methods of choosing winners. Additional models were trained to compare performance with and without including seed as a predictor, as well as comparing results when variables were scaled versus unchanged. We found that including seed led to an overall increase in performance across the models, while scaling the variables did not have a noticeable effect. The final models were compared against a baseline model that only picks the highe ranked teams in every match.

The five different models included three logistic regression models fit using the `glmnet` package in R¹¹, a Random Forest model fit using the `randomForest` package in R¹², and an XGBoost model fit using the `xgboost` package in R¹³. For the three logistic models, one was fit using ridge estimators (L2 Regularization)¹⁴, one was fit with Lasso estimators (L1 Regularization)¹⁵, and the other was fit using Elastic Net regularization¹⁶.

The three different methods that we used to select the winner of any given matchup are:

1. Calculate expected wins for each team, rank them from most to least, and pick the highest ranked team to advance in each matchup.
2. Calculate the probability that each team wins in each round. For each matchup, advance the team that has the highest chance of winning that specific round.
3. In each matchup, use the direct head-to-head probabilities from the probability matrix $\mathbf{P}$ to choose the winners.

Section 2.2 went into more detail about how expected wins and the probabilities of each team winning in any given round can be calculated from $\mathbf{P}$. There are many other alternative ways to select winners beyond just the three aforementioned methods; for example, one could rank the teams by their probability of winning the Championship and then pick winners based

on those rankings. For the purposes of this paper, we only looked at the three mentioned above.

In order to improve the model results, additional tuning was performed for our models. For Ridge, Lasso, and Elastic Net, the regularization parameter λ was automatically tuned using the `cv.glmnet` function from the `glmnet` library. λ controls the strength of the regularization, with $\lambda = 0$ meaning that no regularization is applied. `cv.glmnet` takes a sequence of these lambdas and uses cross-validation to determine an optimal value. For the Elastic Net model, we also wanted to tune the mixing parameter α . This is used in the Elastic Net penalty which is defined as

$$(1 - \alpha)/2\|\beta\|_2^2 + \alpha\|\beta\|_1 \tag{2.9}$$

where $0 \leq \alpha \leq 1$. When $\alpha = 0$ this becomes the ridge penalty, and when $\alpha = 1$ this becomes the lasso penalty. For tuning, we fit models with the values $\alpha = 0.1, 0.2, \dots, 0.9$. To evaluate these models, we used their average and median points and overall accuracies in past tournaments. We will discuss these metrics and our overall evaluation process in more detail in Section 2.3.3. This tuning was done separately for all three of our decision methods and we found that over the past fourteen tournaments, $\alpha = 0.2$ was the best value for Method 2 (using probabilities of winning in each round), and $\alpha = 0.3$ was optimal when using expected wins or head-to-head probabilities.

For the Random Forest model, two parameters were tuned: `mtry`, the number of predictors sampled for each node split, and `ntrees`, the number of decision trees to grow in the forest. Feature selection was also performed beforehand using recursive feature selection using the `rfe()` function from the `caret` library in R¹⁷. The optimal number of features was decided by maximizing AUC, the area under the ROC curve. We chose this metric instead of accuracy as there is a class imbalance in the data (the higher ranked teams win about 71% of the time), so a model that always predicts the higher ranked team could have a relatively high accuracy but be terrible at picking upsets correctly. AUC is less sensitive to class imbalances as it considers both true-positive and false-positive rates, meaning it is a

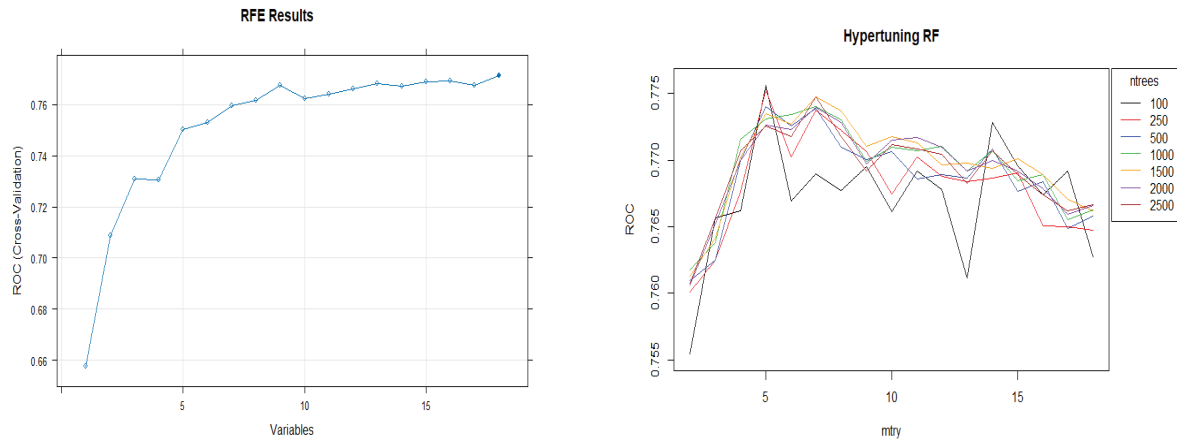


Figure 2.1: **Left:** Results of feature selection for our Random Forest model when Seed is included as a predictor. The model with all 18 predictors yields the largest AUC (labeled as ROC in the plot). **Right:** Tuning results for our Random Forest model. Each line represents a value for `ntrees` across the tested `mtry` values. The largest AUC (labeled as ROC in the plot) is where `ntrees` = 100 and `mtry` = 5.

better judge of the model’s classification ability.

The AUC scores were calculated by cross-validating over manually specified folds in a method we call Leave-One-Year-Out Cross-Validation (LOYO-CV). We have 24 years of training data so 24 folds were created, where in each fold one year of data was excluded and the model was trained on the remaining years. Using LOYO-CV, we found that the best model had an AUC of 0.7713 and used all eighteen predictors. The five most important variables were `diffSRS`, `diffWLpct`, `diffSeed`, `diffORTg`, and `diffDRtg`. The RFE results are displayed in Fig. 2.1.

After feature selection, we performed LOYO-CV again to find the optimal values for `mtry` and `ntrees`. We performed a grid search using `ntrees` values in increments from 100 to 2500 and `mtry` values from 2 to 18. The largest AUC ended up being 0.7756 with `ntrees` = 100 and `mtry` = 5. The results of this process are displayed in Fig. 2.1, and the full list of values tested can be found in Table A.2.

The XGBoost model underwent similar tuning using the `train()` function from the `caret` library. A grid search was performed over seven parameters, including `eta` (the model’s learning rate), `max_depth` (the maximum depth of the trees), and `subsample` (the ratio of

training data to collect to grow trees) among others. The same LOYO-CV process mentioned previously was used for this tuning process as well. For a complete list of parameters and the values tested for each of them, see Table A.3. The optimal values from the LOYO-CV grid search are displayed in Table 2.1. The model fit using these values ended up with an AUC of 0.7744.

Explicit feature selection was not performed on any of the models other than Random Forest. The primary reason for this is that Ridge, Lasso, Elastic Net, and XGBoost all have built-in regularization techniques to help them be less sensitive to irrelevant variables. More specifically, they all include some variation of the L1 and L2 regularization terms, which shrink the effects of the less relevant variables. Future work could include feature selection for these models and explore how it may affect their performances.

Parameter	Value
eta	0.01
max_depth	10
subsample	0.5
colsample_bytree	1
nrounds	250
gamma	0.1
min_child_weight	1

Table 2.1: The optimal values for each parameter of the XGBoost model, determined using grid search and LOYO-CV.

In addition to the model-specific tuning above, in all cases where a head-to-head decision method was used, we also tuned the probability cutoff value used to decide whether to say the higher ranked team wins in a matchup or not. By default, this cutoff is 0.5 meaning if the probability the higher ranked team wins is above 0.5, we say they win. We noticed that our models were consistently predicting fewer upsets than truly happened, and raising the cutoff to a larger value could potentially lead to more upsets being picked and generally better results, especially over the baseline which picks no upsets. Our goal with the tuning is to maximize both total points and overall accuracy, but with more of an emphasis being placed on points. For a more detailed breakdown of why we value points more, see Section

2.3.3. We tested the cutoff values 0.5 to 0.65, increasing by 0.1.

For XGBoost, we decided on a cutoff of 0.59, as while it does lose 0.79% accuracy compared to the baseline cutoff (0.5), we gain 22.5 median points. The Ridge model is also clear-cut; choosing the cutoff of 0.57 also loses 0.79% accuracy, but gains 9.5 median points. The Lasso model is more debatable, as a cutoff of 0.6 yields a model with the highest median points and third highest accuracy, but relative to the baseline it is a decrease of 1.59% accuracy (equivalent to about 1 more incorrect prediction) and an increase of 2.5 median points. We decided to go ahead with the 0.6 cutoff as losing one game but still gaining points means that we are likely only losing some early games that are not worth as much, but are getting more valuable predictions in later rounds correct. The Elastic Net model is also debatable, as a cutoff of 0.61 leads to by far the most points relative to other values, but also is tied for third worst accuracy. The accuracy decrease from the baseline is 3.98% (about 2.5 more wrong predictions), but the median point increase is 11.5. With such a large point increase, we decided to use 0.61, as getting more points is more valuable than getting more games correct. For Random Forest, the default cutoff of 0.5 yielded the best results. Fig A.1 in Appendix A shows the points and accuracies for each model over each cutoff value.

2.3.3 Model Evaluation

Our models can be evaluated with many different metrics; which one you choose depends on what you value most from a model. The two primary metrics we will use in this paper are overall accuracy and total points. Overall accuracy is simply the percentage of games in a bracket that a model correctly predicted the winners. The issue with judging models solely on accuracy is that predicting more games correctly is not a guaranteed way to win in a bracket setting; generally, points are awarded for each correct prediction, with more points being awarded for correct predictions made in later tournament rounds. Judging on overall accuracy equates correctly selecting the champion and correctly selecting the winner of any game in the first round, which is not reflective of most bracket competitions.

The second method we used to evaluate models assigns point values to correct predictions in each round. We decided on using the same scoring system as most major websites that run tournaments: a bracket gets one point for every correct prediction in the Round of 64, two points in the Round of 32, four in the Sweet 16, eight in the Elite Eight, sixteen in the Final Four, and 32 for picking the champion correctly¹⁸. We believe that total points is the most relevant metric, as it balances having a large number of correct predictions with making sure that a higher value is being placed on correct predictions in later rounds. One could also look at other evaluation metrics such as a model's accuracy of picking past champions, the Final Four, etc. if so desired, but for this study we stick with overall accuracy and total points.

With these metrics, we tend to value medians more than means. The main reasoning for this is that there are often outliers in our results; years where our models did very poorly or exceptionally well. The mean is not a resistant statistic, which means it is easily influenced by outliers and is pulled away from the center of the data. When outliers are present, the median provides a more accurate representation of the true center. Thus, in most of our results, we primarily prioritize median values, typically sorting our results by medians first and using means to break ties.

In order to obtain accuracies and points for our models, we created several custom R functions. The primary function takes a year of data we wish to test our models on and creates the training and testing data, where the training is data from every year but the specified one. Every time a new year is specified, this process is repeated, ensuring that the models are never trained on the same year they are predicting.

The function then runs through all five model types, using three separate custom helper functions per model. The first of these functions builds our probability matrix $\mathbf{P}$ from Section 2.1 by making predictions on every possible team matchup and reshaping the results into a 64x64 matrix. The next function calculates round victory probabilities and expected wins for each team based on Section 2.2. It accomplishes this by taking $\mathbf{P}$ as an input along

with a vector containing the names of the teams in the first round in order (order meaning names from top to bottom on the left side of the bracket followed by top to bottom on the right side) and uses Eq. 2.3 and Eq. 2.8 to calculate round-specific win probabilities and expected wins respectively.

Finally, the last helper function runs a full tournament simulation that makes predictions one round at a time based on the desired decision method, specified as an argument in the main function. The winners are selected one round at a time, stored, and used to filter the data for the next round. This process is repeated until there is only one winner remaining. If a vector of the true tournament results is supplied, the function then checks the predictions in each round against the true results, calculating the accuracies for each round and the total points. What is returned is a list that contains the model's predictions for each round in the form of team names and their probability of victory if applicable alongside a table that includes overall accuracy, accuracy in each individual round, and point totals.

The main function binds the results of all the models alongside a baseline model that always chooses the higher ranked team (determined by seed, using SRS as a tiebreaker in the case the seeds are tied). These are returned along with predictions for a knapsack-style tournament, which will be discussed further in Chapter 3. The results are also returned with indicators corresponding to several optional arguments in the main function, such as if Seed was included as a predictor or if the variables should be standardized. This allowed us to compare our findings not only between models and decision methods (expected wins, round probabilities, or head-to-head), but also between these other factors.

Evaluation was performed over fourteen years of tournaments from 2010 to 2024 (no tournament was played in 2020). The results were then aggregated by model type and decision method. The mean, median, and standard deviation of the total points as well as overall accuracy were computed for each group. Originally, the models were also grouped by whether seed was included as a predictor and if the data was standardized. We found that standardization did not have a significant impact on performance; more often than not,

Model	Decision	Med_Pts	Avg_Pts	SD_Pts	Med_Acc	Avg_Acc	SD_Acc
XGB	exp_wins	116.5	105.14	33.24	0.6667	0.6678	0.0706
XGB	prob	116.5	105.14	34.69	0.6667	0.6633	0.0755
XGB	h2h	115.5	106.29	36.44	0.6667	0.6533	0.0864
El-Net	prob	115.0	105.43	36.43	0.7302	0.6803	0.0944
Lasso	prob	114.5	104.36	37.38	0.6984	0.6757	0.0919
El-Net	h2h	109.0	103.79	39.29	0.6825	0.6587	0.0884
RF	h2h	108.0	102.64	36.56	0.6746	0.6542	0.0824
El-Net	exp_wins	108.0	100.86	31.62	0.6904	0.6723	0.0859
Lasso	exp_wins	107.5	101.29	32.22	0.6984	0.6723	0.0897
RF	prob	105.5	100.00	34.16	0.6587	0.6531	0.0802
Ridge	exp_wins	105.5	94.57	29.84	0.6508	0.6497	0.0748
Ridge	h2h	103.0	100.93	33.75	0.6667	0.6519	0.0820
Ridge	prob	103.0	94.86	30.04	0.6587	0.6542	0.0713
RF	exp_wins	101.5	99.07	31.16	0.6746	0.6542	0.0788
Lasso	h2h	99.5	101.07	36.56	0.6984	0.6644	0.0915
HS	HS	96.0	91.14	24.79	0.6032	0.6202	0.0515

Table 2.2: Aggregated model results over every tournament from 2010 to 2024. “exp_wins” refers to using expected wins to make predictions, “prob” refers to using round probabilities, and “h2h” refers to using head-to-head probabilities. Med_Pts and Med_Acc refer to Median Total Points and Median Overall Accuracy respectively. SD_Pts and SD_Acc stand for the standard deviation of total points and overall accuracies respectively. HS stands for “High Seed” and refers to the baseline model that only picks the higher ranked team for each matchup. Models are ranked by Med_Pts, then Avg_Pts, then Med_Acc, then Avg_Acc, all in descending order. For a visualization of these results, see Fig. A.2 and Fig. A.3.

a model that used standardizing and one that did not would have identical outcomes to each other. With seed, we found that including it as a predictor improved the outcomes for every metric aside from the standard deviations which slightly increased. Thus, for all the results discussed below, assume that the data is not standardized and that seed is used as a predictor.

Table 2.2 shows the results of all our models, ranked from best to worst. Immediately there are two things that jump out at us. First, the top three models are all XGBoost, which is a strong indicator that it is the best model for this problem regardless of how we decide on how to pick winners. Second, every single model outperforms the baseline model in every category aside from standard deviations for both total points and overall accuracy. This

indicates that our model performances are less consistent than picking the high seed is, but despite this they all tend to perform much better on average. In other words, the baseline may be more consistent, but it is more consistent around significantly worse results than the other models.

The first two models are almost identical, but the XGBoost model that uses expected wins to determine winners ends up pulling ahead as it has a slightly higher average accuracy and lower standard deviations, making it a little more accurate and less variable than the XGBoost using round probabilities. The median points of 116.5 compared to the average of 105.14 indicates that the total points across the tested years is left skewed. This means that while most of the values are around 116.5, there are some years with very low point totals that are bringing the average down. The median and mean accuracies are almost identical however, and their accuracies' standard deviation is the lowest of every model other than the baseline, meaning their values are fairly consistent around a little less than 67% accuracy.

Other models of note include the head-to-head XGBoost model which has the highest average total points and the Elastic Net model using round probabilities which had the highest median and mean accuracies. The XGBoost model has the highest mean points by about one point, but its median is lower by one point, and despite having the same median accuracy as the other XGBoost models, its mean accuracy is lower. It also has larger standard deviations, meaning that it is slightly less consistent than the other XGBoost models. We rank it lower than the others because as mentioned previously we tend to value median over mean.

The Elastic Net model with round probabilities seems like a very solid contender for best model; despite placing third in terms of median points (our most important metric), it places second in terms of mean points and first in terms of both median and mean accuracies. While it is only 1.5 median points away from tying for first place under that metric, its median accuracy is 6.35% higher than the XGBoost models, 3.18% higher than the next closest model, and 12.7% higher than the baseline.

Model	Med_Pts	Avg_Pts	SD_Pts	Med_Acc	Avg_Acc	SD_Acc
XGB	116.5	105.52	33.96	0.6667	0.6621	0.0760
El-Net	112.5	103.36	35.09	0.6904	0.6704	0.0879
Lasso	105.5	102.24	34.62	0.6984	0.6708	0.0889
RF	105.0	100.57	33.23	0.6667	0.6538	0.0785
Ridge	103.0	96.79	30.63	0.6587	0.6519	0.0743
HS	96.0	91.14	24.79	0.6032	0.6202	0.0515

Table 2.3: Model results from 2010 to 2024 aggregated by model type. HS stands for “High Seed” and refers to the baseline model. The results are in the same order as the ones in Table 2.2. For a visualization of these results see Fig. A.4.

While these numbers are impressive, it is important to keep in mind that the Elastic Net model also has the highest standard deviation of accuracies, meaning that though it does do the best on average, it can be more inconsistent than the other models. As the average accuracy is almost 5% below the median, we can tell the data is left-skewed, meaning that there were some years with exceptionally bad accuracies that brought the mean down (and the standard deviation up). Despite these flaws, it still remains one of the top models and one of the best alternatives to XGBoost.

Alongside discovering what the top performing individual models are, we are also interested in if specific model types or decision methods performed better than others. Table 2.3 shows the results when aggregating between model types. In terms of points, the XGBoost models appear to be the best; this makes sense since the top three models in Table 2.2 are all XGBoost. They are beat by the Lasso and Elastic Net models when it comes to accuracy, as those models have the highest and second-highest accuracies respectively. Most bracket competitions value points over accuracy, but if one is participating in a tournament where accuracy becomes more important, these models may be better options than XGBoost.

Table 2.4 shows our findings when grouping all the models by their decision method. Using round-specific probabilities appears to be the best method in terms of median points, although it does have a worse average points, median accuracy, and overall accuracy compared to head-to-head probabilities. Expected wins scores the lowest of our three methods,

Decision	Med_Pts	Avg_Pts	SD_Pts	Med_Acc	Avg_Acc	SD_Acc
prob	112	101.96	33.86	0.6667	0.6653	0.0815
h2h	106	102.94	35.54	0.6746	0.6569	0.0838
exp_wins	106	100.19	30.90	0.6667	0.6633	0.0785
HS	96	91.14	24.79	0.6032	0.6202	0.0515

Table 2.4: Model results from 2010 to 2024 aggregated by decision method (round probabilities, head-to-head probabilities, or expected win). HS stands for “High Seed” and refers to the method of picking the higher ranked team in every game. The results are in the same order as the ones in Table 2.2. For a visualization of these results see Fig. A.5.

but not by much. It does have the lowest standard deviations, meaning its results are less variable than the other models. We do hesitate to say any one method is obviously better than another, as they all have very similar performances.

2.3.4 Results in the 2025 Tournament

The tournament held in 2025 was not used for any part of the training process. The total points and overall accuracies for each of our models in this tournament are shown in Table 2.5. This was a rather unusual year as there were very few upsets; high seeds won almost every matchup. The Final Four ended up consisting of all the 1-seeds, which is only the second time that this has occurred since the tournament was expanded to 64 teams. Every model ended up predicting this Final Four correctly, which is partly why all the point values are so high.

One could argue that the models only appear to perform well because there were very few upsets in the tournament. If our models always end up picking high seeds, then they would appear to do very well this year but would struggle in most other years. To counter this, we point to Table 2.5, specifically the position of the baseline model that only chooses higher ranked teams. The baseline model ranks tenth out of all the models; this indicates that not only are the majority of our models picking upsets (if they were not, they would be at the same level), but they are getting those upsets right.

Two of the top three performing models, H2H Elastic Net and H2H XGBoost, had their

Model	Decision	Points	Accuracy
Lasso	h2h	142	0.8095
El-Net	h2h	141	0.7937
XGB	h2h	140	0.7778
Ridge	h2h	137	0.7460
El-Net	prob	124	0.7778
XGB	exp_wins	123	0.7619
RF	h2h	123	0.7619
RF	exp_wins	118	0.7302
El-Net	exp_wins	109	0.7778
HS	HS	109	0.7619
Lasso	prob	108	0.7619
Lasso	exp_wins	108	0.7619
Ridge	prob	107	0.7460
RF	prob	107	0.7460
XGB	prob	107	0.7460
Ridge	exp_wins	107	0.7460

Table 2.5: Model results for 2025, ordered by points first, then overall accuracy (both in decreasing order). HS refers to the "high seed" baseline model.

brackets submitted to ESPN’s Tournament Challenge. ESPN is one of the most popular websites for people to submit brackets to—in 2025 a record-setting 24.4 million brackets were submitted¹⁹. At the conclusion of the tournament, our submitted models ranked as the 91.9th and the 91.4th percentiles respectively, while the baseline model placed as the 71.1st percentile. This shows that our models performed exceptionally well not only compared to the baseline but also relative to all brackets submitted to ESPN in 2025.

2.3.5 Model Results as Ranking Methods

In the previous sections, we used our models to directly predict the outcomes of games. This is not the only way the models can be used though; we can also use their results to rank the teams, and use those rankings to help decide which teams to advance in the bracket. This is similar to using ranking systems such as KenPom⁴ or LRMCMC⁶ to help advise your decisions rather than picking every matchup for you. Using our proposed framework, one can calculate the expected wins for any team as well as the probability they win in any round, and use

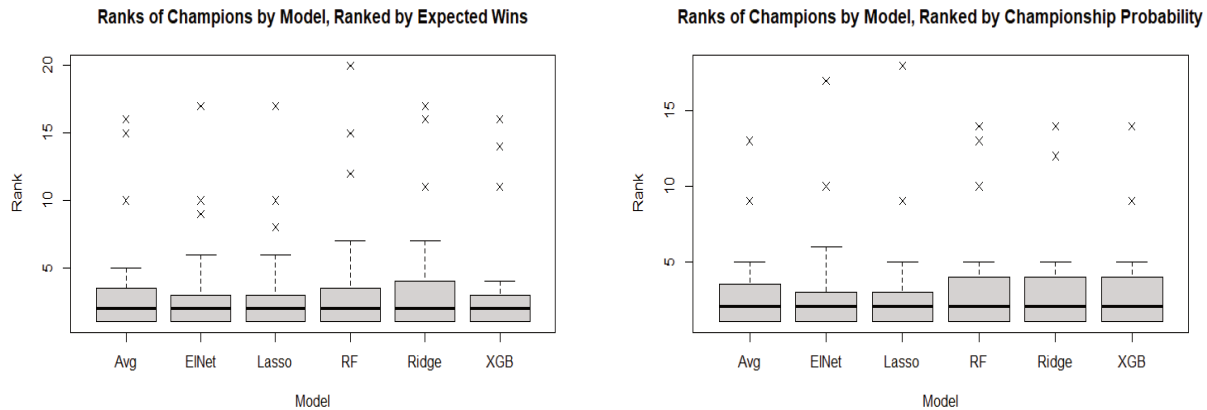


Figure 2.2: Boxplots of champion ranks, grouped by model. “Avg” refers to the average rank, calculated by averaging the ranks of the other five models. **Left:** Teams are ranked by their expected wins. **Right:** Teams are ranked by their probability of winning the championship.

those values to rank the teams. In this section, we will demonstrate this system with an example using our previous models.

Looking back at the past 25 years of tournaments, we compare two forms of ranking: ranking based on expected wins and ranking based on the probability of a team winning the championship. If one so desired, they could also rank with other metrics such as probability of making it to the Final Four, but we will focus on these two methods. Looking at the rankings, we are interested in seeing how our models perform when it comes to ranking past champions.

All the rankings end up performing similarly, as can be seen in Figure 2.2; they all have a median rank of 2, for example. For all these models, we counted how many past champions fell within the top rankings. Looking back at all 25 tournaments from 2000 to 2025, there are two models that both perform exceptionally well.

First, we examine an XGBoost model that ranks teams according to their expected wins. XGBoost also happened to have our best model for a traditional bracket competition, specifically the model that used expected wins to make its predictions—it makes sense that its expected-wins-based rankings would perform well. When we look at the rankings for this model, we make the following observations:

- 22 of the past 25 champions were ranked in top 4.
- 20 of the past 25 champions were ranked in the top 3.
- 18 of the past 25 champions were ranked in the top 2.
- 11 of the past 25 champions were ranked #1.

The only three champions who did not rank fourth or above were the 3-seeded UConn team in 2011 (ranked 11th by XGBoost), the 7-seeded UConn team in 2014 (ranked 14th), and the 4-seeded UConn team in 2023 (ranked 16th). Granted, in 17 of the past 24 tournaments a 1-seed has won and we would expect 1-seeds to be ranked in the top 4. However, looking at the model results, there are often lower seeds that make it into the top four as well; most notably, XGBoost ranked 3-seeded Florida first overall in 2006, the year they won the championship. When Syracuse, also a 3-seed, won in 2003, they were ranked third overall.

A second model that comes very close in performance to XGBoost with ranking by expected wins is the Lasso model using probability of winning the championship to rank teams. This is interesting as the Lasso model, while not bad, has yet to be discussed among the “best” models in Section 2.3.3. However, when it comes to ranking teams by probability of winning the championship, it does quite well. Looking at these rankings we observe that:

- 23 of the past 25 champions were ranked in top 5.
- 19 of the past 25 champions were ranked in the top 3.
- 14 of the past 25 champions were ranked in the top 2.
- 8 of the past 25 champions were ranked #1.

Comparing this to the XGBoost model, it would seem that XGBoost does have an edge and is the best model for ranking. That being said, there is one area that this model did better in than XGBoost, and that is ranking the three UConn championships. UConn’s 2011 team was ranked 9th overall, their 2023 team ranked 18th, and most notably their 2014 team was

ranked 5th as a 7-seed. The model also ranked Syracuse and Florida third overall in 2003 and 2006 respectively.

Good rankings are not limited to champions either. Take, for example, 2021, when the 11-seed UCLA made it to the Final Four. Both models ended up ranking them 8th overall, surpassing half of the 2-seeds, as well as every three, four, and five-seed. An 11-seed “should” place (meaning, where they would place if we ranked all the teams by seed) 41st to 44th, so them being placed 8th is an indicator they may have been massively under-seeded. Another notable example is the 4-seeded Michigan team in 2013 that ended up placing second in the tournament. Despite being a 4-seed, both models ranked them 2nd overall; the only team ranked higher was Louisville, the 1-seed that won the tournament.

The above results were specifically selected as they show some of the most impressive rankings these models have made, but we do not wish to push the idea that these models are always excellent; they have their fair share of very poor rankings as well. For instance, in 2011 the 8-seeded Butler team placed second in the tournament. XGBoost ranked them 42nd overall, and Lasso placed them at 38th. We would expect a typical 8-seed to fall somewhere between 29th and 32nd, so both these models were essentially claiming that Butler was over-seeded, which ended up clearly not being the case. The same year, the 11-seeded VCU made it to the Final Four; XGBoost ranked them even lower at 49th, and Lasso placed them at 43rd. For a case where a higher seed was mishandled, one can look at Michigan State’s 5-seeded Final Four team in 2010. XGBoost ranked them decently at 14th, slightly higher than a typical 5-seed, but the Lasso model placed them at 22nd, which is lower than a typical 5-seed.

The reasoning for covering these more negative examples is twofold: first, we want to make sure that readers are aware that there are limitations to the rankings. Despite making some excellent team placements, they can also make poor ones. We believe that they tend to do better more often than not, however, and are thus still useful as a ranking method. The second reason is to provide more evidence that there is no data leakage, meaning that we

do not believe that any of the models “cheated” and were trained on past games they then predict outcomes for. When we saw outcomes like UCLA’s 2021 team being ranked 8th we grew concerned that this might be an issue and checked our training and tuning processes again, but were not able to find any issues that indicate the models may be training on the games they are predicting. On top of this code-checking, we noticed that despite making some excellent rankings, none of the models consistently made excellent rankings (i.e. always putting Final Four teams in top ten or anything along those lines). For UCLA specifically, we also went and checked the KenPom⁴ ratings that year, and found them ranked 13th, also incredibly high for an 11-seed.

Let us look at how XGBoost ranked the teams in the 2025 tournament with respect to expected wins. Table 2.6 contains the team ranks, names, and seeds. The teams are ranked by their calculated expected wins in decreasing order. There are many insights we can make from these rankings—for instance, Gonzaga (ranked 18th) and UC San Diego (ranked 24th) are 8 and 12-seeds respectively, but are ranked much higher than other 8 and 12-seeds. UC San Diego ended up losing to Michigan by three in the first round, and Gonzaga lost to Houston in the second round, also by three. Despite these being losses, these outcomes were still closer than most 5 vs. 12 or 1 vs. 8-seed match-ups, which indicates that despite losing, they were still potentially under-seeded; their opponents just happened to be slightly better.

The most egregious cases of teams being “over-seeded” from these 2025 rankings are the two five-seeds Michigan (ranked 35th) and Memphis (ranked 36th). These rankings place them far below where we would “expect” a five-seed to be, as they are very distant from the other five or even six-seeds and are ranked under several much lower seeded teams. Michigan’s opponent, the 12-seed UC San Diego, ranks 24th—much higher than both teams. As stated previously, Michigan did end up winning by three and ended up going on to win against Texas A&M (ranked 17th, slightly low for a four-seed) before losing to 1-seed Auburn (ranked 3rd). Memphis, however, did end up losing to its 12-seed opponent Colorado State by eight points. Colorado State is ranked 37th, which is directly after Memphis and also

Rank	Team	Seed	Rank	Team	Seed
1	Duke	1	33	Xavier	11
2	Florida	1	34	Utah State	10
3	Auburn	1	35	Michigan	5
4	Houston	1	36	Memphis	5
5	Tennessee	2	37	Colorado State	12
6	St. John's	2	38	Creighton	9
7	Texas Tech	3	39	Arkansas	10
8	Michigan State	2	40	New Mexico	10
9	Maryland	4	41	Oklahoma	9
10	Alabama	2	42	UNC	11
11	Iowa State	3	43	Liberty	12
12	Wisconsin	3	44	Mississippi State	8
13	Kentucky	3	45	Georgia	9
14	Arizona	4	46	Vanderbilt	10
15	Clemson	5	47	Drake	11
16	Saint Mary's	7	48	High Point	13
17	Texas A&M	4	49	McNeese State	12
18	Gonzaga	8	50	Yale	13
19	Purdue	4	51	Troy	14
20	Marquette	7	52	Robert Morris	15
21	Ole Miss	6	53	Akron	13
22	Missouri	6	54	Lipscomb	14
23	Oregon	5	55	Bryant	15
24	UC San Diego	12	56	Montana	14
25	Kansas	7	57	Omaha	15
26	Illinois	6	58	Grand Canyon	13
27	BYU	6	59	UNC Wilmington	14
28	Louisville	8	60	Alabama State	16
29	Baylor	9	61	Norfolk State	16
30	VCU	11	62	Mount St. Mary's	16
31	UConn	8	63	Wofford	15
32	UCLA	7	64	SIU Edwardsville	16

Table 2.6: 2025 team rankings by expected wins, using an XGBoost model.

above several 8-, 9-, and 10-seeds; a sign that they are a potentially under-seeded team, and a good choice to pick as an upset.

When it comes to the rankings of teams that made it deep into the tournament, XGBoost does very well. All teams that made it to the Elite Eight (Auburn, Michigan State, Florida, Texas Tech, Duke, Alabama, Houston, Tennessee) were ranked within the top 10. The teams

that ended up in the Final Four (Auburn, Florida, Duke, Houston) were ranked as the top 4. Houston, the runner-up, was ranked 4th and Florida, the champion, was ranked 2nd.

In summary, while the models we created can be used for direct game-by-game predictions, we believe they can also provide more value as a ranking system that can advise a player filling out a bracket. Two of our ranking systems, the XGBoost ranking by expected wins and the Lasso ranking by probability of winning the championship, demonstrate excellent performance and can frequently identify potential champion teams as well as highlight other teams that may be over- or under-seeded. Although there are occasional poor rankings, their overall performance suggests that our rankings could be valuable decision-making resources for filling out a bracket.

Chapter 3

Knapsack Bracket Competitions for March Madness

3.1 Introduction

March Madness bracket competitions have become a tradition for many college basketball enthusiasts, and tens of millions of people participate in such contests each year. For example, in 2025 24.4 million brackets were submitted for ESPN’s annual competition¹⁹ with a collective 3.1 billion dollars being placed on the outcomes of games¹. The majority of contests have users select who they believe will win in each round of the tournament, awarding points for correct predictions. The further along the tournament progresses, the more points each correct choice is worth, with the championship game being worth the most points. The winner is determined by who has the largest total after summing all points earned by each player.

In this chapter, we consider an alternative form for a bracket competition—a knapsack bracket competition (KBC). In these contests, each team “costs” a certain amount of points, and given a budget constraint, participants are challenged to choose a subset of teams to maximize the number of wins earned by the selected teams.

24th ANNUAL NCAA BASKETBALL TOURNAMENT CONTEST

The NCAA basketball tournament begins this coming Thursday. If you are a basketball fan, then a contest is being held to demonstrate your basketball knowledge. If you are not a basketball fan, then a contest is being held to allow you to demonstrate how little the self-proclaimed "basketball fans" know about basketball. The rules are simple, but most importantly **IT'S FREE!**

CONTEST INSTRUCTIONS

1. Select as many teams as you wish from the following list, but the total cost must be less than or equal to 100. For example, you might pick all the #1 seeds.
2. Write your choices on the accompanying entry form and return it to my mailbox in the Statistics Department office or you can e-mail me your picks, listing your teams (sorted by region, please) and their costs. This must be accomplished by 11:00 a.m. on Thursday, March 19th.
3. Your score for the tournament is the number of games won by the teams you choose.
4. There are 4 PLAY-IN games this year. Those games are all listed in italics. You do not get anything for picking any of those winners. If you submit your entry before those games are played, you can put *both teams' names* in the space and you will then get credit for any games the winner of that game gets AFTER the play-in. You do *not* get a point for any team's play-in victory.
5. **PRIZES:** \$10.00 for 1st, \$5.00 for 2nd, 25 cents for last. There is **NO ENTRY FEE**.

Seed	Cost	MIDWEST	WEST	EAST	SOUTH
1	25	Kentucky	Wisconsin	Villanova	Duke
2	19	Kansas	Arizona	Virginia	Gonzaga
3	13	Notre Dame	Baylor	Oklahoma	Iowa State
4	12	Maryland	North Carolina	Louisville	Georgetown
5	11	West Virginia	Arkansas	Northern Iowa	Utah
6	10	Butler	Xavier	Providence	Southern Methodist
7	8	Wichita State	VCU	Michigan State	Iowa
8	5	Cincinnati	Oregon	North Carolina State	San Diego State
9	5	Purdue	Oklahoma State	Louisiana State	St. John's
10	4	Indiana	Ohio State	Georgia	Davidson
11	4	Texas	<i>BYU/Ole Miss</i>	<i>Boise State/Dayton</i>	UCLA
12	3	Buffalo	Wofford	Wyoming	Stephen F. Austin
13	2	Valparaiso	Harvard	Cal Irvine	Eastern Washington
14	2	Northeastern	Georgia State	Albany	UAB
15	1	New Mexico State	Texas Southern	Belmont	North Dakota State
16	1	<i>Hampton/Manhattan</i>	Coastal Carolina	Lafayette	<i>North Florida/Robert Morris</i>

Figure 3.1: Instructions for the 2015 Knapsack Bracket Competition held between Kansas State University and Michigan State University.

To the best of our knowledge, the origin of KBCs can be traced back to Dr. James Stapleton of Michigan State University in 1986. Their creation led to annual competitions hosted by the Statistics Departments at Michigan State University and Kansas State University. Figure 3.1 contains the instructions for the contest held in 2015.

In this particular competition, participants were given a budget of 100 points with which to select teams, with the goal of maximizing the number of wins by their selected teams. Looking at the results after the tournament concluded, the maximum possible number of wins was 28. The optimal teams and their costs were Duke (25), Notre Dame (13), Louisville

(12), Utah (11), Michigan State (8), Oregon (5), NC State (5), Cincinnati (5), UCLA (4), Ohio State (4), Dayton (4), UAB (2), and Georgia State (2). Duke won the championship in 2015, and Michigan State made it to the Final 4. The other two Final 4 members, Kentucky and Wisconsin, were both 1 seeds, and Wisconsin ended up going to the championship. Despite this, neither of them are included in the optimal solution.

3.2 Solving the 0-1 Knapsack Problem

A knapsack problem is an optimization problem that seeks to answer the question “If you have n items that each weigh some amount and have some value and can only carry up to a certain weight W , what is the optimal subset of items to take to maximize value?” An 0-1 knapsack problem is the most common form of the problem; it limits you to only being able to take either zero or one copy of an item, which is reflective of most real-world cases. The 0-1 knapsack can be applied to any situation where you have items that all have some sort of cost/weight, all have some value you are wanting to maximize, and there is a limit on the total cost/weight. One such application of this is our knapsack bracket competition, as each team has a cost and a value we want to maximize (number of wins), and there is a cost constraint we must meet.

One approach to solving the knapsack problem is to use dynamic programming²⁰, also known as the “tabulation” or “bottom-up” approach. Say we have n teams and each team has a cost c_i (specified in Figure 3.1) and a value v_i . The value in our problem is the number of wins, which is not something we know before a tournament begins. Instead, we will estimate true wins with the expected wins of the teams, which are calculated using the framework in Section 2.2. Let us also say that C is the cost limit, or the maximum points

we can spend on teams. We can write a model for this problem as follows:

$$\begin{aligned}
& \text{maximize: } \sum_{i=1}^n v_i x_i \\
& \text{given that: } \sum_{i=1}^n c_i x_i \leq C \\
& \text{where each } x_i \in \{0, 1\}
\end{aligned} \tag{3.1}$$

where x_i is an indicator representing if any given team i was chosen or not. To summarize Eq. 3.1, we want to maximize our value (wins) while not exceeding our cost limit C (which is typically 100).

Dynamic programming approaches this method by filling T , an $(n + 1) \times (C + 1)$ table where the rows can be represented as $j = 0, 1, \dots, n$ and the columns as $k = 0, 1, \dots, C$. Note that j ends up representing the teams, and k is all possible weights from 0 to C . The indices of the table $T_{j,k}$ are filled out according to the following rules, as long as $c_j \leq k$:

$$T_{j,k} = \begin{cases} 0 & \text{if } j = 0 \text{ or } k = 0 \\ \max(v_j + T_{j-1,k-c_j}, T_{j-1,k}) & \text{if } j > 0 \end{cases} \tag{3.2}$$

What these rules do is first set the first row and first column to be zeroes. Then, if we are in any row aside from the first, we fill the cell we are in with whatever is larger: the total value if we keep team j , which is $v_j + T_{j-1,k-c_j}$, or the value if we do not keep team j , which is $T_{j-1,k}$. The constraint $c_j \leq k$ checks to make sure that team j can be acquired at a given cost k . If this does not hold, then we cannot afford the team and we fill the current index with the value above it. That is, if $c_j > k$ then:

$$T_{j,k} = T_{j-1,k} \tag{3.3}$$

As we move through the table from top to bottom starting in the top left, this ends up

progressively increasing our value until we get to $T_{n,C}$, the bottom right corner of the table and the maximum possible value for the optimal solution.

In summary, what we are doing is going through each team one at a time and determining if it gives us more value to include it or not. To figure out how much value a team adds, we look at its value v_j added to the best previous value, specifically the one that's cost plus the cost of the team being added equals k . We compare this to the value if we did not add the team, which is just the value in the previous row but the same column, and pick whichever is bigger.

For example, say that we are at $k = 100$ and we are considering adding a 1-seed that has cost $c_j = 25$ and value $v_j = 3$. Say the value in the previous row and the same column is 12; this represents the maximum value we have found thus far in the problem. We compare this number with the value if we added the 1-seed. To get that, we will take $v_j = 3$ and add it to the value in the previous row, but in column $k = 75$. This works because this is a “bottom-up” approach; the previous row will always have the best possible values for each value of k up to whatever point we are at in the process. The value at $k = 75$ in the previous row will be the maximum value we could get from spending 75 points. Combining this with the value of our 1-seed, which costs 25, would yield the maximum possible value for $k = 100$ given we are including our 1-seed team.

Let us say that the value in the previous row at $k = 75$ is 10. If we add our team's value, we get 13, which is larger than the previous value of 12. That indicates that we should keep the 1-seed, along with whatever teams made up the other 75 points. However, if our value of k is less than the cost of our team (say we were looking at $k = 10$ for our 1-seed instead of $k = 100$), then we cannot include our 1-seed. The best value would be to keep whatever the previous best value was, which is the value in the previous row but the same column, $T_{j-1,k}$. This general process is repeated for every spot in the table and will always yield the maximum possible value in the last cell.

Using the steps described above, we can figure out what the maximum achievable wins

is. However, to figure out what teams to select to obtain this value, we will need to work backward. To do this, we follow the steps below starting at the maximum value $T_{n,C}$:

1. If $T_{j,k} > T_{j-1,k}$ (if the current value is greater than the previous one), then team j was included in the optimal results. Store team j 's name and move to the value at $T_{j-1,k-c_j}$.
2. If $T_{j,k} = T_{j-1,k}$, team j was not included in the optimal results. Move to the value at $T_{j-1,k}$.
3. Repeat steps 1 and 2 until a value of 0 is reached.

This process returns the names of all selected teams to reach the maximum value in $T_{n,C}$. Recall that c_j is the weight of the j th team, so when we accept a team and move to the value at $T_{j-1,k-c_j}$, we effectively take the cost of the selected team out of the total remaining cost k .

This is the process that we used to get our knapsack results. We created a custom R function to run through this algorithm when given vectors of the team names, their expected wins, their costs, and optionally their “true” values (the actual number of times they won in a given tournament). It returns three things if used on past data: the names of the teams selected for the optimal solution, the predicted maximum value (i.e. the predicted number of cumulative wins the selected teams will get), and the actual number of wins those teams got. Optionally, it can also return the optimal results for the tournament by using the teams’ true number of wins instead of expected win as values. The only difference when running on a year where the true results are unknown is that the actual number of wins is not returned.

3.3 Knapsack Model Results

In this section, we discuss the results of our models for a knapsack bracket competition (KBC). In Section 2.3.3 we evaluated our five models (Ridge, Lasso, Elastic-Net, Random

Model	Med_Wins	Avg_Wins	SD_Wins	Med_Prop	Avg_Prop	SD_Prop
Best	33.5	32.7	2.84			
El-Net	22	22.6	4.47	0.688	0.694	0.141
Ridge	21.5	21.4	4.57	0.656	0.655	0.140
XGB	21	20.8	3.79	0.645	0.638	0.115
Lasso	20.5	21.9	4.62	0.672	0.674	0.147
RF	20.5	20.9	4.54	0.639	0.643	0.150

Table 3.1: Knapsack results by model over 2010 to 2024. “Best” refers to the optimal solution, using the teams’ actual wins as values. The “Prop” columns refer to the proportion of the maximum possible wins a model captures. The models are ordered by median cumulative wins.

Forest, and XGBoost) on tournaments from 2010 to 2024, judging their performance on a more “traditional” tournament setting (namely by using total points and overall accuracy as evaluation metrics). We also grouped the models by decision type to see if any specific methods of selecting teams in a matchup work better than others. Here, we will look at the same five models over the same years of data, but the decision method will not be a factor. We are no longer making predictions on a game-by-game basis; our goal now is to maximize expected wins while staying within our cost constraint, and the expected value calculations do not change when using different decision methods.

To judge our models, we look at two main metrics: the first is the number of cumulative wins a model’s teams get, and the second is the proportion of the maximum possible wins a model captures. As we know the true number of wins for each team, we know what the optimal solution is for the tournament. We want our models to not only maximize number of wins, but we also want to maximize this proportion of wins captured, since the larger it is, the closer our model is to the true optimal solution.

The mean, median, and standard deviations for the cumulative wins and the proportion of total wins captured are displayed in Table 3.1. The first row represents the optimal solution, using the true wins as values to solve the knapsack problem. We can see that all our models have very similar performances, but the Elastic-Net model does end up as our top performer. Not only does it have the highest mean and median for cumulative wins, it

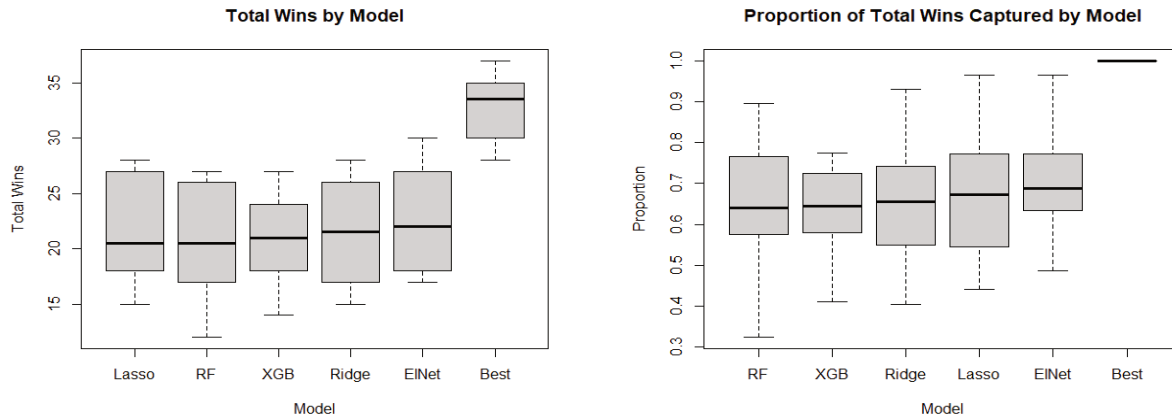


Figure 3.2: **Left:** Boxplots of total wins grouped by model. **Right:** Boxplots of proportions of total wins captured grouped by model. The Elastic Net model’s results look to be slightly higher than all of the other models.

also usually captures almost 70% of the maximum achievable wins.

Our XGBoost model, which we consider to be the best model for a “traditional” bracket competition in Section 2.3.3, ends up in the middle of our models in terms of KBC performance. It does have one noteworthy feature though; its standard deviations for both wins and proportion of wins captured are considerably lower than the other models, meaning it tends to have more consistent results. Although this is an advantage, we can see in Fig. 3.2 that it never performs to nearly the same level as the other models. In its best year (2016), the XGBoost model captured only 77.42% of the possible wins, while Elastic-Net captured 96.55% in its best year (2019).

In years where there are very little upsets, it is hard to get a large number of wins as the teams that are making it deep into the tournament all cost a lot. The tournament in 2025 was one such year, with 23 wins being the maximum achievable number of wins. It was actually the worst year we tested in terms of maximum possible wins; the previous lowest maximum was 28 wins in 2015.

Our Elastic Net model selected Auburn (1), Texas Tech (3), Iowa State (3), Maryland (4), Louisville (8), Gonzaga (8), UConn (8), Utah State (10), VCU (11), Drake (11), UNC (11), Colorado State (12), and UC San Diego (12) for the 2025 tournament. These teams

collectively achieved a total of 14 wins, which is almost 61% of the maximum possible wins. The only overlap between the teams selected by Elastic Net and the optimal teams was Gonzaga and Colorado State.

In this chapter, we explored the idea of knapsack bracket competitions, in which we can turn the tournament into an 0-1 knapsack problem with the goal of maximizing team wins while staying within a cost constraint. We established an algorithm that can solve this problem, demonstrating how dynamic programming can be used to calculate the maximum number of wins in a given year and the optimal teams to buy to achieve those results. Our analysis shows that when using our models to calculate expected wins for this algorithm, our Elastic Net model achieves the best performance, on average capturing a little under 70% of the maximum possible wins in a tournament and averaging 22 to 23 wins a year.

Chapter 4

Discussion and Future Work

In this paper, we have proposed a mathematical framework that can take any model that returns a probability of some team i winning against some team j and calculate round-specific probabilities of victory as well as expected wins. This structure can provide additional supporting metrics for existing models and methods as well as convert them into actionable rankings, giving researchers even more ways to analyze and improve upon their work. This foundation is not limited solely to basketball either; it can be applied to single-elimination tournaments in any sport.

We have shown this framework in action by using logistic regression models fit using Ridge, Lasso, and Elastic-Net regularization, as well as Random Forest and XGBoost models. These models were fit using historical tournament data based on the differences in two teams' stats for each game. We then showed three different ways of using our framework to make game predictions: ranking teams by expected wins and choosing the higher ranked team, choosing the team that has the highest probability of winning in the current round, and using direct head-to-head probabilities that team i will beat their opponent team j .

These models were then evaluated in past tournaments from 2010 to 2024 based on their predictive abilities, as well as how they perform as ranking systems. When it comes to game-by-game predictions, two of our best models were XGBoost selecting based on expected wins

and Elastic-Net selecting based on round-specific win probabilities. The XGBoost model was the best performer in terms of median points, with a median of 116.5 points and a median accuracy of 66.67%. The Elastic-Net model was the most accurate model, with a median points of 115 and a median accuracy of 73.02%. Both these models as well as every other model we tested performed above the baseline model of picking the higher ranked team in every matchup. This “High Seed” model has a median of 96 points and a median accuracy of 60.32%.

Although our models do well on a game-by-game basis, we believe that they may offer more value as ranking methods to aid players in filling out a bracket, as opposed to purely making the decisions for them. Our best model in terms of ranking past champions ended up also being an XGBoost model, where the rankings were based on teams’ expected wins. Using these, we found that 22 of the past 25 champions have been in our model’s top four ranks, and the actual champion has been ranked first eleven times. These successes are not limited solely to champions either, with cases like 11-seed UCLA being ranked eighth overall in 2021, the year they made the Final Four. Considering our overall findings, we believe that our models’ performances are strong enough that they can be used as a good alternative to established ranking systems or at the very least work alongside them to provide additional insights into the team-selection process.

We believe that our framework’s value for models comes more from its ability to rank teams in a tournament as opposed to generating game-by-game predictions for two major reasons. Firstly, the insight gained from the rankings can immediately highlight certain teams as being ranked above or below where we would expect them to be. This can help a player determine which teams may be over- or under-valued. Secondly, using rankings keeps the spirit and excitement of the tournament by preserving the player’s agency in the decision-making process. Instead of picking games for the player, which is not quite as engaging, it can be thought of as an additional resource to help guide the player’s decisions. In this regard, it achieves a better balance of giving the player control while still providing

them support.

Finally, we discussed a knapsack bracket competition (KBC)—an alternative competition where players have a limited budget with which to select teams, and the goal of maximizing wins of those selected teams. We provide an algorithm that can use our models to solve the knapsack problem and return an optimal subset of teams to select to maximize total wins. We discovered that our Elastic-Net model tends to return the best performance in a KBC tournament, averaging slightly less than 23 wins and capturing a little under 70% of the maximum possible wins.

For future work, there are many things that can be tested and tweaked to potentially get better results. One simple place to start would be to keep the same models and data, but change our decision methods or team ranking methods. In this paper, we only considered three decision methods (expected wins, round probabilities, head-to-head probabilities) and two ways of ranking the teams (expected wins and probability of winning the championship). Using our framework from Chapter 2, we could easily change these and use the probability of winning in the championship, in the Final Four, etc. to make our game-by-game predictions and see if there is an improvement there. We could also rank our teams based on a metric like probability to win in the Final Four and use that to see how our models rank past Final Four teams.

We are also not limited to the specific models we chose for this paper. The point of our framework is that *any* model can work if it returns probabilities. We are not even limited to our data; any dataset and any model can be used to make predictions or rankings. Future research could look at trying different models, including different variables, or even training on different types of datasets. We took the difference of teams' stats as that allows us to directly compare two teams to each other, but that is not a requirement for our process to work. Theoretically, as long as a model can output a probability of some team i winning against an opponent j , the framework will work, which leaves the door open for many different future models and potentially even better results.

Bibliography

- [1] American Gaming Association. Estimated amount of money bet on march madness in the united states in 2019 and 2024 (in billion u.s. dollars). Statistica, 2024. URL <https://www.statista.com/statistics/1296462/total-amount-bet-march-madness/>.
- [2] Ncaa college basketball rankings: Ap top 25 basketball poll, 2025. URL <https://apnews.com/hub/ap-top-25-college-basketball-poll>.
- [3] 2024-25 men’s college basketball power index - espn, 2025. URL <https://www.espn.com/mens-college-basketball/bpi>.
- [4] Ken Pomeroy. 2025 pomeroy college basketball ratings, Mar 2025. URL <https://kenpom.com/>.
- [5] Paul Kvam and Joel S. Sokol. A logistic regression/markov chain model for ncaa basketball. *Naval Research Logistics (NRL)*, 53(8):788–803, Jul 2006. doi: 10.1002/nav.20170. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.20170>.
- [6] Mark Brown and Joel Sokol. An improved lrmc method for ncaa basketball prediction. *Journal of Quantitative Analysis in Sports*, 6(3), 2010. doi: 10.2202/1559-0410.1202. URL <https://doi.org/10.2202/1559-0410.1202>.
- [7] Jun Woo Kim, Mar Magnusen, and Seunghoon Jeong. March madness prediction: Different machine learning approaches with non-box score statistics. *Managerial and Decision Economics*, 44(4):2223–2236, 2023. doi: <https://doi.org/10.1002/mde.3814>. URL <https://onlinelibrary.wiley.com/doi/abs/10.1002/mde.3814>.
- [8] Ali Alfatemi, Mohamed Rahouti, D. Frank Hsu, and Christina Schweikert. Advancing

- ncaa march madness forecasts through deep learning and combinatorial fusion analysis. In Kohei Arai, editor, *Intelligent Systems and Applications*, pages 539–560, Cham, 2024. Springer Nature Switzerland. ISBN 978-3-031-66431-1. URL https://link.springer.com/chapter/10.1007/978-3-031-66431-1_38.
- [9] Edward Kaplan and Stan Garstka. March madness and the office pool. *Management Science*, 47:369–382, 03 2001. doi: 10.1287/mnsc.47.3.369.9769. URL <https://pubsonline.informs.org/doi/epdf/10.1287/mnsc.47.3.369.9769>.
- [10] Sports Reference. Men’s and women’s college basketball statistics and history, 2024. URL <https://www.sports-reference.com/cbb/>.
- [11] Jerome Friedman, Robert Tibshirani, and Trevor Hastie. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1): 1–22, 2010. doi: 10.18637/jss.v033.i01. URL <https://pubmed.ncbi.nlm.nih.gov/20808728/>.
- [12] Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL <https://CRAN.R-project.org/doc/Rnews/>.
- [13] Tianqi Chen, Tong He, Michael Benesty, Vadim Khotilovich, Yuan Tang, Hyunsu Cho, Kailong Chen, Rory Mitchell, Ignacio Cano, Tianyi Zhou, Mu Li, Junyuan Xie, Min Lin, Yifeng Geng, Yutian Li, and Jiaming Yuan. *xgboost: Extreme Gradient Boosting*, 2024. URL <https://CRAN.R-project.org/package=xgboost>. R package version 1.7.7.1.
- [14] S. Cessie and J. C. Houwelingen. Ridge estimators in logistic regression. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 41(1):191–201, 3 1992. ISSN 0035-9254. doi: 10.2307/2347628. URL <https://doi.org/10.2307/2347628>.
- [15] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1 1996. doi: 10.111

- 1/j.2517-6161.1996.tb02080.x. URL <https://doi.org/10.1111/j.2517-6161.1996.tb02080.x>.
- [16] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 67(2):301–320, 03 2005. ISSN 1369-7412. doi: 10.1111/j.1467-9868.2005.00503.x. URL <https://doi.org/10.1111/j.1467-9868.2005.00503.x>.
- [17] Kuhn and Max. Building predictive models in r using the caret package. *Journal of Statistical Software*, 28(5):1–26, 2008. doi: 10.18637/jss.v028.i05. URL <https://www.jstatsoft.org/index.php/jss/article/view/v028i05>.
- [18] PrintYourBrackets. Ncaa bracket scoring systems. PrintYourBrackets, 2012. <https://www.printyourbrackets.com/bracket-scoring.html>.
- [19] Kevin Ota. 24.4 million brackets: Espn tournament challenge sets new record for third consecutive year - espn press room u.s., 3 2025. URL <https://espnpressroom.com/us/press-releases/2025/03/24-4-million-brackets-espn-tournament-challenge-sets-new-record-for-third-consecutive-year/>.
- [20] R. Andonov, V. Poirriez, and S. Rajopadhye. Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research*, 123(2):394–407, 2000. ISSN 0377-2217. doi: [https://doi.org/10.1016/S0377-2217\(99\)00265-9](https://doi.org/10.1016/S0377-2217(99)00265-9). URL <https://www.sciencedirect.com/science/article/pii/S0377221799002659>.

Appendix A

Supporting Materials for Model Fitting, Tuning, and Evaluation

Variable	Definition
diff3Ppct	Difference in the teams' 3-point percentages
diffFTpct	Difference in the teams' free throw percentages
diffTRB	Difference in the teams' total rebounds per game
diffAST	Difference in the teams' assists per game
diffSTL	Difference in the teams' steals per game
diffBLK	Difference in the teams' blocks per game
diffTOV	Difference in the teams' turnovers per game
diffPF	Difference in the teams' personal fouls per game
diffPPG	Difference in the teams' points per game
diffOppFGpct	Difference in the teams' opponents' field goal percentages.
diffOppPPG	Difference in the teams' opponents' points per game
diffDRtg	Difference in the teams' defensive ratings
diffWLPct	Difference in the teams' win/loss percentages

diffSRS	Difference in the teams' SRS (Simple Rating System). SRS is a team rating based on point differentials and strength of schedule, where 0 represents average.
diffPace	Difference in the teams' paces (estimated number of possessions per 40 minutes).
diffORtg	Difference in the teams' offensive ratings
diffFGpct	Difference in the teams' effective field goal percentages (similar to field goal percentage but adjusted to account for 3-point shots being worth more than a 2-point shot).
diffSeed	Difference in the teams' seeds.

Table A.1: A description of all the variables used in the training data. All differences are calculated by subtracting the lower ranked team's stat from the higher ranked team's stat.

Parameter	Values Explored
ntrees	100, 250, 500, 1000, 1500, 2000, 2500
mtry	2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18

Table A.2: The parameters and their values used in the grid search for tuning the Random Forest model.

Parameter	Values Explored
eta	0.01, 0.1, 0.2
max_depth	4, 6, 8, 10
subsample	0.5, 0.75, 1
colsample_bytree	0.5, 0.75, 1
nrounds	50, 100, 200, 250
gamma	0.01, 0.1, 0.5, 1
min_child_weight	1, 10, 30

Table A.3: The parameters and their values used in the grid search for tuning the XGBoost model.

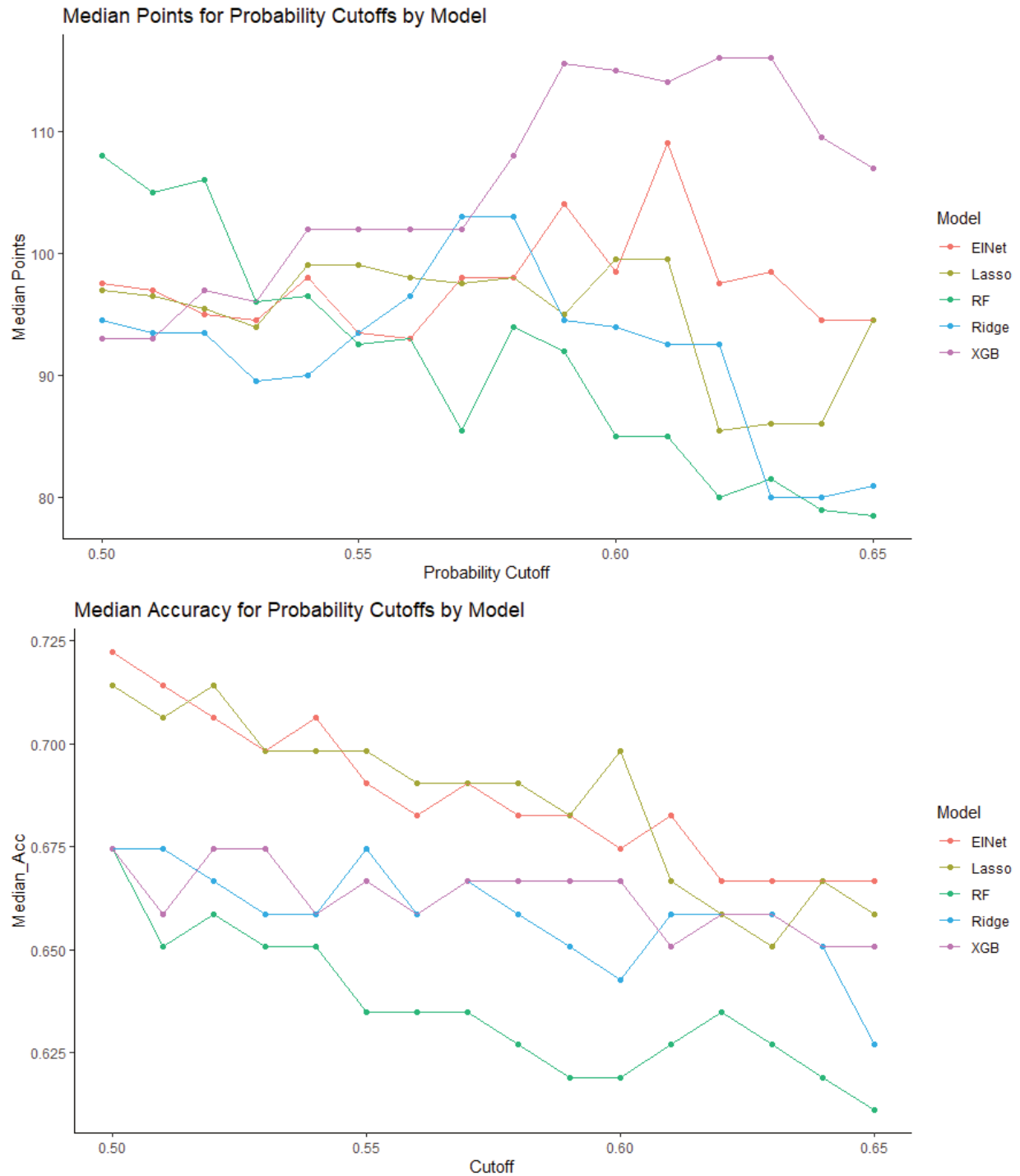


Figure A.1: **Top:** Median total points for each cutoff value grouped by model type. **Bottom:** Median overall accuracy for each cutoff value grouped by model.

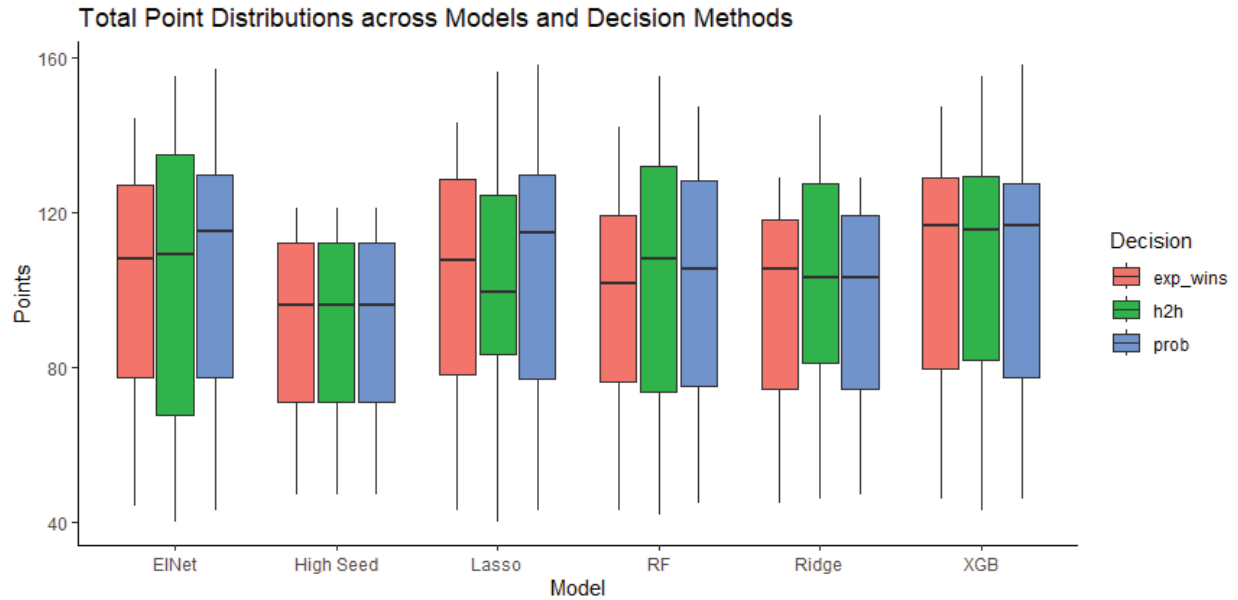


Figure A.2: Distributions of total points from 2010 to 2024 grouped by model and decision method. Note that the high seed model is identical for all decision methods, as it really has it's own decision method: pick the higher ranked team every time.

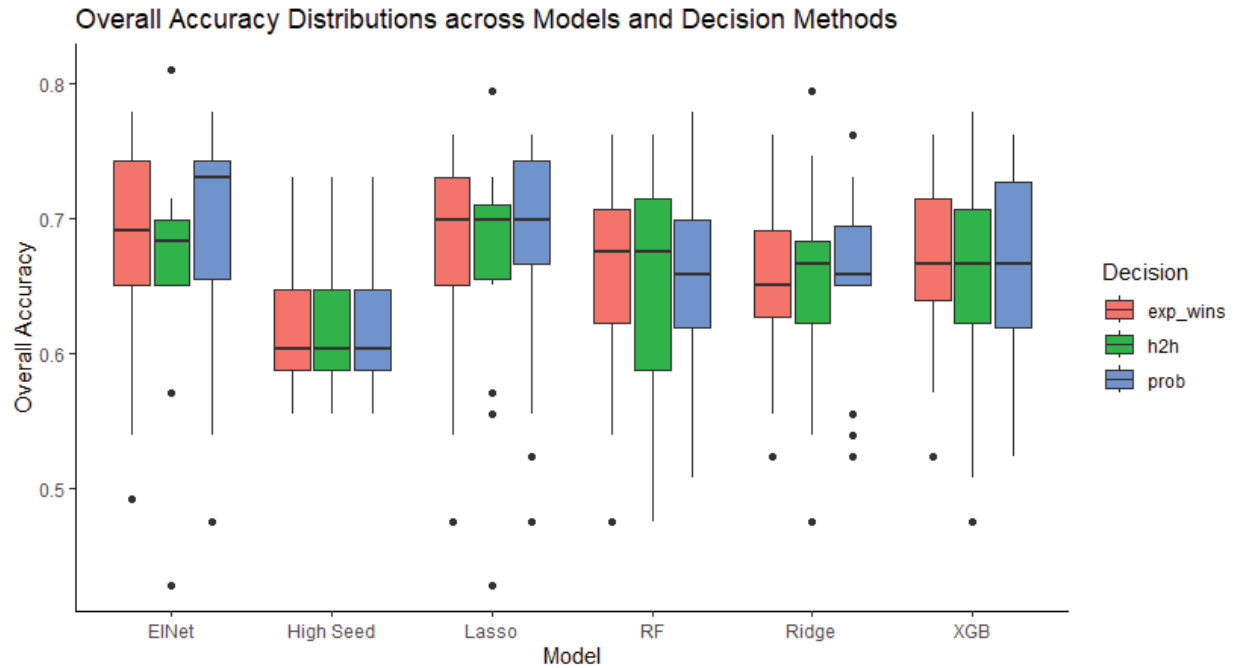


Figure A.3: Distributions of overall accuracies from 2010 to 2024 grouped by model and decision method. The high seed model is identical for the same reason as Fig. A.2.

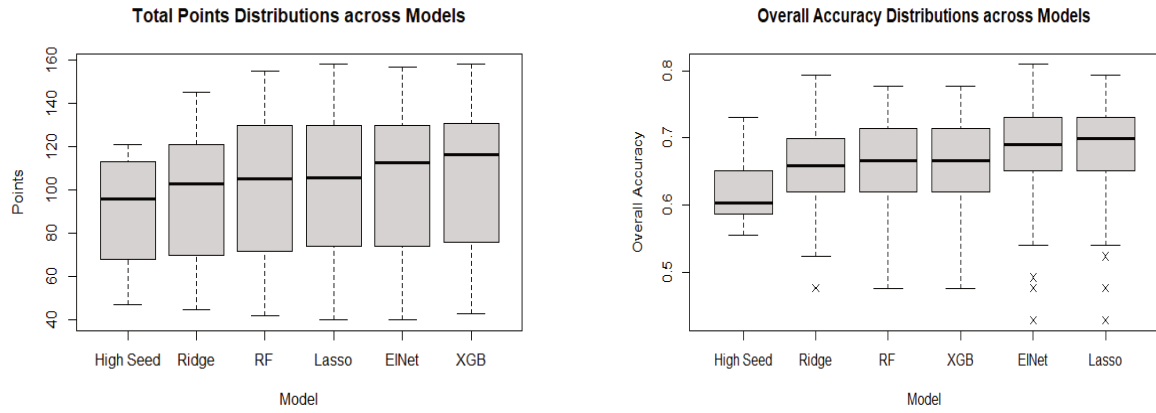


Figure A.4: **Left:** Distribution of total points from 2010 to 2024, grouped by model. **Right:** Distribution of overall accuracies from 2010 to 2024, grouped by model.

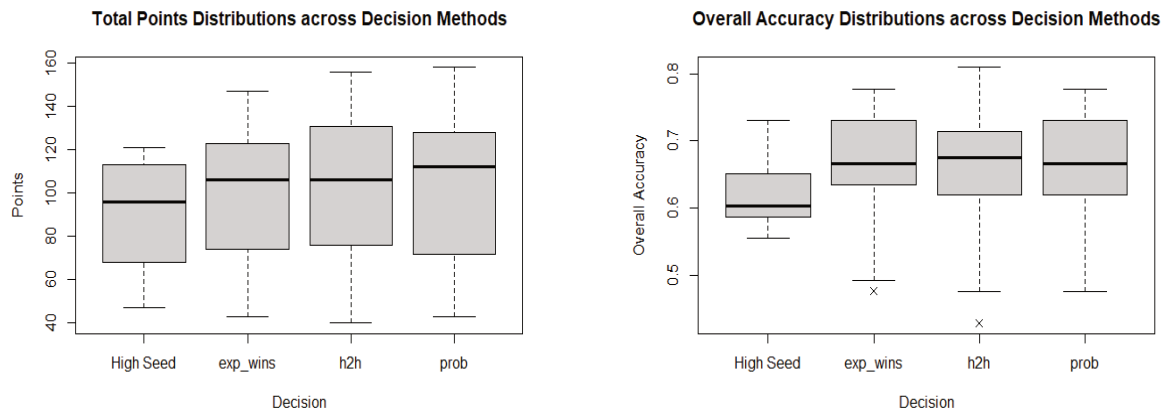


Figure A.5: High Seed is included as its own decision method in this plot. **Left:** Distribution of total points from 2010 to 2024, grouped by decision method. **Right:** Distribution of overall accuracies from 2010 to 2024, grouped by decision method.