Joe Coyne

1/5/2024

Modeling the NCAA College Basketball Tournament

Introduction

The goal of this project is to answer the typical question you might hear leading up to the NCAA Tournament in March: Can we create a perfect bracket? According to NCAA.com, the odds of a perfect bracket are extremely slim, and no one has even come close to predicting all 67 games correctly. If you are choosing matchups by the flip of a coin, for instance, using 50/50 odds, your odds of a perfect bracket are about 1 in 9 quintillion. With a little bit of basketball knowledge and slightly more informed decisions, your odds are still about 1 in 120 billion. The aim of this project will not necessarily be to create a perfect bracket but to rather best model the outcomes in a way that is replicable from year to year. This is for two reasons, one being that even with robust predictive capabilities, we still most likely won't get close to a perfect bracket using new data. The other is that if we do predict a previous year's tournament perfectly, we will have overfit the model and made it less useful in different years. A second question this project will aim to answer is if the 2019-2020 season were to have run uninterrupted, how would the tournament have unfolded and who would have been the eventual champion?

Before delving into the technical details of this project, it is important to have some background on the NCAA Tournament in general. Every year around March, the NCAA Selection Committee chooses the 68 best teams in Division I basketball to be invited to play in the Tournament. However, each conference within Division I has a spot reserved for its conference tournament winner. Before the first full round of the tournament (the Round of 64), there are 4 games played in Dayton, Ohio called the First Four, where teams that just barely

made the tournament play each other. There are a total of 6 rounds in the tournament, spanning from the Round of 64 up until the Final Four and Championship, adding up to 67 total games.

The data used for this project consists of three separate datasets from two different sources. The first comes from an open competition on Kaggle hosted by Playground Prediction Competition called "March Machine Learning Mania 2021 – NCAAM – Spread", in which the goal was to predict the winner and the margin of victory of each NCAA Tournament game. Within this dataset, there are 3 files being used. The first simply links each NCAA Division I team with its Team ID. The other two are the main crux of the data, separated by regular season data and NCAA Tournament data. This data consists of more basic statistics seen in box scores, such as field goals, free throws, rebounds, assists, turnovers, steals, blocks, personal fouls, and a few others. Another feature of this data is that it is collected on a game-by-game basis, so in order to complete analyses on teams, we will have to group by year and then Team ID, summarizing each team's year-end totals for each statistic.

The other datasets are from Sports Reference. The first data set is comprised of more advanced metrics, summarized into season statistics for each team. Some of these variables include Strength of Schedule (SOS), Offensive Rating (Ortg), Pace, and True Shooting Percentage (TS%), which are all defined and further explained in Appendix B. The other Sports Reference data set has some basic information and data on each conference, like overall win percentage, Strength of Schedule, and Conference Tournament champion.

All the data for this project will span from the 2002-2003 season through the 2019-2020 season. However, since there was no NCAA Tournament for the 2019-2020 season, we will only use the NCAA Tournament data through the 2018-2019 season.

To briefly preview what this project will look like, an initial model will be built using the 2002-2003 through 2018-2019 regular season data in order to predict each year's tournament outcomes and champion. This will be done using the actual bracket provided by the NCAA Selection Committee. Once the bracket is created, the next step will be to find the best model building technique to best predict the actual bracket progression and these head-to-head matchups (for instance, Random Forest models, Jackknife and Leave One Out Cross Validation models, Neural Networks, etc). Finally, once the model creation is final, it will be implemented on the 2019-2020 regular season in order to predict the 2020 NCAA Tournament and its eventual champion.

Key Variables and Concepts

Explanations of all variables from the data can be found in Appendix B. However, some of the advanced variables are quite complex, so here are some further descriptions and formulas. The first variable discussed will be Strength of Schedule (SOS). Below is the formula used to calculate SOS, with OW% indicating Overall Opponent Winning Percentage, OppW% indicating opponent winning percentage, OOW% indicating Overall Opponent Winning Percentage of the opponents, and TmGP indicating the number of games played by the team for which you are calculating SOS.

$$\mathrm{SoS} = \frac{2\mathrm{OW\%} + \mathrm{OOW\%}}{3} \qquad \mathrm{OW\%} = \frac{\sum_{i=1}^{n} \mathrm{OppW\%_i}}{\mathrm{TmGP}} \qquad \mathrm{OOW\%} = \frac{\sum_{i=1}^{n} \mathrm{OW\%_i}}{\mathrm{TmGP}}$$

Another important advanced variable is the Simple Rating System (SRS), which is calculated by adding margin of victory (MoV) with strength of schedule (SOS).

$$SRS = MoV + SOS$$

One of the more popular advanced metrics in basketball right now is True Shooting Percentage (TS%). True shooting percentage equals half the points scored (PTS) divided by the sum of the field goals attempted (FGA) and 0.475 times the free throws attempted (FTA).

$$TS\% = \frac{0.5\, pts}{FGA + 0.475\, FTA}$$

True Shooting Percentage calculates the efficiency of a player's scoring by taking into account all types of shots attempted. More simply put, it represents the percentage of points a player scores per scoring opportunity, considering the different point values of different shots.

There is also the variable Effective field goal percentage (eFG%), which is a statistic that adjusts field goal percentage to account for the fact that 3 point field goals count for 1 more point than 2 point field goals. Below is the formula used for this calculation.

$$eFG\% = \frac{FG + (0.5 * 3P)}{FGA}$$

Finally, a variable that will be manually added to the dataset will be one that quantifies the number of NCAA Tournament wins a team attains for the respective season's tournament, called PostW. This variable will range from 0 to 6 and will be the target variable for the initial regression models. Since it is an ordinal factor variable, the analyses performed will be of the multi-class variety. This variable is calculated simply by counting the number of times each team name appears in the March Madness data set as the winning team.

Initial Data Input and Manipulation

First, the three data files from the Kaggle data source, found in Appendix B.1, were loaded into R, those being "MRegularSeasonDetailedResults.csv", "MNCAATourneyDetailedResults.csv", and "MTeams.csv". Both the Regular Season and March Madness data have the same format and variables. Each row listed represents a game played between two teams (whether that be in the regular season or in the NCAA Tournament). The variables in this data are the Season the game occurred in, the winning and losing team id, and then some basic game statistics for each team, like score, field goals made, rebounds, assists, turnovers, etc. The teams data had 4 variables: TeamID, TeamName, FirstD1Season, and LastD1Season. To use this dataset more fully in the analyses, two new variables were created, "WTeamID" and "LTeamID". Then, the teams data was merged with the regular season data, first by WTeamID and then by LTeamID so that we had the winning and losing team for each game. Next, this merged data was separated into a win dataset and a loss dataset, choosing only statistics that applied either to only the winning or losing team (depending on the dataset) or to both teams. A similar merging was done with the March Madness data, matching up the winning and losing teams with their id's. Both the win and loss data along with the March Madness data were split into different datasets by filtering for season.

Now moving into some of the initial data manipulation, we first start with the win and loss data. Two new variables are created, "Wins" and "Losses". In the win data, we set each game to have a Wins value of 1 and a Losses value of 0. In the loss data, we set each game to have a Wins value of 0 and a Losses value of 1. Next, we group both the win and loss data by TeamID and Team_Name and summarize each variable by taking each variable's sum, so that each row is one team's total statistics for when they are the winning team or for when they are

the losing team. Finally, we combine the win and loss data by completing a full join by TeamID and Team_Name, summarizing each variable again to get the overall season totals for each team.

The next step was to create a variable to quantify the number of wins a team gets in the NCAA Tournament, which will be denoted as PostW. We will start by creating this variable with the 2003 March Madness data and then apply it to the following years, only selecting teams that made the tournament for our initial designs. First, the data is grouped by the Winning Team ID. Then, the PostW variable is created by summing the number of times a team appears in this data as the winning team. The march madness data is then right joined with itself so we can also select Season, WTeamID, and LTeamID along with the Team Name and PostW. However, this lists each team as many times as they have postseason wins, so the distinct function is used to keep each team name only once. This successfully calculates the amount of postseason wins for the teams with at least one win in the tournament, but not for teams that made the tournament but lost in the first round. To handle this, we right join only the LTeamID and LTeam_Name from the march madness data and filter by complete cases. This gets rid of any rows where the losing team already appeared as a winning team because the winning team will be listed as NA, along with Season, WTeamID, and PostW. Now the goal is to create a new row for all of the losing teams so their PostW can be calculated too. To do this, the bind_rows function is used along with mutating Team Name and TeamID to select the team name and id if it doesn't already appear in the WTeam_Name and WTeam_ID variable. The PostW variable for these losing teams are denoted with NA for the time being. Next, we select only the Season, Team_Name, WTeamID, LTeamID, and PostW variables. Because some of the losing teams also appeared as winning teams in previous rounds, duplicates must be removed, so we keep only unique team names or

rows with a value in PostW. Finally, we select only the variables Season, TeamID, Team_Name, and PostW and replace any NA values in PostW with 0.

Finally, the Regular Season data is merged with the March Madness data so that the regular season statistics can be shown next to the team's total postseason wins. A left join of the march madness data for each year into the regular season data for that year by TeamID and Team_Name is conducted. We mutate the PostW variable to be listed as 0 if the current value is NA, as well as convert it into an ordered factor variable with levels ranging from 0 to 6. 0 is labeled as "First.Round.Exit", 1 is labeled as "Round.of.Thirty.Two", 2 is labeled as "Sweet.Sixteen", 3 is labeled as "Elite.Eight", 4 is labeled as "Final.Four", 5 is labeled as "Runner.Up", and 6 is labeled as "Champion". At this point, the Sports Reference data is merged in as well, by Team Name, to create the full data sets that will be explored.

Initial Data Exploration

Before any initial models are built, the variables must be looked at and analyzed to see some general trends. Looking at the correlogram describing the Kaggle variables from the 2002-2003 Regular Season, found in Appendix C.1, the first thing that pops out is that Wins and Losses have the strongest correlation, reaching a correlation of almost -1. This makes sense because as a team wins more games, their opportunities to lose games shrink, and vice versa. Another observation is that a lot of the shot-making variables, like field goals made (FGM) and field goals attempted (FGA), among others, are highly correlated with each other. This signals that some VIF analysis made be needed in order to deal with some of the multicollinearity. Finally, looking at our target variable PostW, we see that Wins have the strongest positive

correlation with PostW and that Losses have the strongest negative correlation. This is also shown in the Variable Importance Table in Appendix C.2, which is ranked in descending order based on the absolute value of correlation with PostW, as, after PostW which has a correlation of 1.0 (since it is the same variable), Wins and Losses have the strongest correlations, with total field goals made (total_FGM) having the third strongest correlation.

Next, we explore the variables in the 2018-2019 Regular Season Data. Another correlogram displaying all of the Kaggle variables is listed in Appendix C.3. Once again, the strong negative correlation between Wins and Losses is noted. There is also similar multicollinearity that will need to be dealt with in the shot-making variables. This will have to be dealt with in each year's data because of the structure and definitions of these variables. In terms of variables correlated with PostW, we see that Wins and Losses have the strongest correlations again, but now total blocks (total_Blk) has the third strongest correlation. This is confirmed using the Variable Importance Table in Appendix C.4.

Next, we look at the change in variable importance from the 2002-2003 Regular Season to the 2018-2019 Regular Season. The difference in variable importance is calculated in the Variable Importance table in Appendix C.5. The variables that grew in correlation with PostW the most were total 3-point field goals made (total_FGM3) and total 3-point field goals attempted (total_FGA3), which makes sense because basketball in general has become a lot more reliant on the 3-point shot now than it had in the past. Conversely, the variables with the largest decrease in correlation with PostW were total free throws attempted (total_FTA) and total free throws made (total_FTM).

When we look at the distributions of our initial variables, almost all of them have normal looking bell-shaped curves with no real outliers. The only variable where this is not the case is

PostW. Only the variables in the 2003 and 2019 tournaments were analyzed but since the structure of the tournament hasn't changed (specifically, there are the same number of rounds and pretty much the same number of teams that can achieve each postseason win threshold), we expect the PostW distribution to remain the same. The histograms for PostW in the 2003 NCAA Tournament and the 2019 NCAA Tournament can be found in Appendix C.6 and Appendix C.7, respectively. There is a clear and obvious right skew, with around 300 teams (out of 327 in 2003 and 353 in 2019) achieving 0 postseason wins. This indicates that there is a class imbalance that will need to be dealt with, whether through downsampling, scaling, or some other method in order for the rarer cases of postseason wins to be equally represented and predicted.

Initial Modeling

Before diving into any models, the data needs to be prepared and cleaned a little more to better fit the models. Since each season can only have 1 team win 6 postseason games and 1 team win 5 postseason games, our first initial models will be conducted using data from the 2002-2003 season through the 2005-2006 season so that each value of PostW has more than one observation. In order to reduce some of the class imbalanced mentioned at the end of the Initial Data Input and Manipulation section, we only select teams that appeared in the Tournament. This narrows the number of teams that have 0 postseason wins to around 32, which is a lot less than the 300 we had before.

Another step taken before the actual models was to check for multicollinearity among variables. Both a correlation matrix and a vif calculation were done on all of the numeric variables to determine which variables were strongly correlated with each other. Some of these

correlations can be viewed in the correlograms in Appendices C.1 and C.3, particularly looking at the larger circles indicating greater correlations. After this analysis, the following variables were removed from the initial model due to multicollinearity: TeamID, Team_Name, Losses, total points (total_pts), total field goals attempted (total_FGA), total 3-point field goals attempted (total_FGA3), and total free throws attempted (total_FTA).

Now that the variables for the initial models have been selected, we can move onto the actual modeling. The first model created was based on the Jackknife and Leave-One-Out Cross Validation (LOOCV). Like K-fold cross validation, the jackknife and leave-one-out cross validation resamples by systematically leaving one observation out of the dataset at a time, and then calculating the estimate of interest for each subset. So, in a dataset with N observations, you would create N subsets, each containing N-1 observations. Running the leave-one-out method on the initial model for the 2003-2006 seasons and setting our method to "polr" for ordinal logistic regression, we achieve the Confusion Matrix in Appendix D.1. Calculating the accuracy of this confusion matrix, with the correct predictions along the diagonal, we get a value of 0.6923. Looking at the Variable Importance table in Appendix D.2, we see that Wins is still especially important to PostW, which makes sense. Something interesting to note is that total turnovers (total_TO) appears rather important in this model but was not reported as highly correlated with PostW in the initial data exploration.

The next model ran on the initial model data was a Decision Tree. Before creating the tree, the initial model data was partitioned with a 50/50 split into training and testing data. Using the tree function found in the "tree" package in R, we create a decision tree predicting PostW and plot it in Appendix D.3. An interpretation of this decision tree goes as follows: If a team is in the NCAA Tournament but has less than 20.5 wins, the model predicts that you will be a First Round

Exit. But if that team has more than 20.5 wins, it moves to the right branch of the tree. Then, if it has more than 25.5 wins as well, you move to the right branch of that part of the tree. If the team also has more than 160 total blocks on the season, the decision tree model predicts that the team will make the Elite Eight but lose in that round. The confusion matrix and accuracy are then analyzed, as seen in Appendix D.4. We calculate the accuracy to be 0.6531, which is about .04 less than the jackknife and leave-one-out cross validation, indicating that this decision tree model is slightly less accurate in predicting PostW. As visible in the confusion matrix, this model doesn't predict any team to even get past the Elite Eight, so some further investigation will have to be done in order to get the necessary number of predictions for each value.

Then, a stepwise selection was run on the data, using the same splits for the training and testing data. Instead of using the "lm" function for typical regressions, the "polr" function, which is used in ordered logistic or probit regression, was used, since the target variable PostW is an ordinal variable. Based on the confusion matrix in Appendix D.5, we get an accuracy of 0.6939, which is ever-so greater than the accuracy of the jackknife and leave-one-out cross validation model.

The next model used was a random forest model, which is essentially an aggregate model based on the average of n decision tree models. Each random forest model takes a subset of variables to predict and filters through which ones enter and leave the model. The same training and testing data splits were used in this model. One issue that was encountered during this model building process was that there was a class imbalance issue where there were a lot more observations for PostW being equal to 0 or 1 as opposed to 5 or 6, so downsampling was necessary. This is when you intentionally reduce the training dataset in size because the variable of interest is so underrepresented. Using the downsampled data and setting the number of

decision trees in the random forest to 1000, we run the model and create the confusion matrix in Appendix D.6. We also see that the OOB Error rate is equal to 54.17%. OOB Error, which stands for Out of Bag Error, basically acts as a test or validation data set, representing the variables that were left out of the specific iteration of the random forest model. In Appendix D.7, we see a plot of the misclassification rates for each value of PostW by the number of trees in the random forest model. The red line represents the misclassification rate of the First Round Exits, the green line represents the Round of 32 misclassification rate, the black line is the Out of Bag Error, the blue line is the Sweet 16 misclassification rate. The Elite Eight, Final Four, Runner Up, and Champion values all have a misclassification rate of 1 since the model didn't predict any of these values. Finally, we apply the random forest model to the test data and print the confusion matrix and statistics, found in Appendix D.8. This leads to an accuracy of 0.7217, which is the highest accuracy of the initial models. This indicates that, based on the initial models of the 2003-2006 data, a random forest model may be the best modeling technique to predict PostW for each year.

One issue that kept appearing in these initial models was that none of them predicted teams to have high PostW values like Champion, Runner Up, Final Four, etc. And even though the initial models had high accuracy, these numbers inflate the actual predictive power of the models because most of the predictions fell into the First Round Exit category. To combat this, two steps were taken. The first step was to combine all 17 seasons together to ensure a larger sample size of each PostW value. The second, and more crucial step was to split the modeling process into a binary model predicting if a team would amass 1 postseason win and then a more robust model predicting how many postseason wins they would have if they got the first.

Secondary Modeling

Similar to the initial models, the first step taken was to filter the dataset to only contain teams that made the NCAA Tournament in that particular year. Unlike the previous models though, all 17 seasons were merged together into one dataset to better ensure that each PostW value was selected when the data was later split into train and test data. Next, the full dataset was mutated to create a PostW_binary factor variable, where if a team were a First Round Exit, they would be denoted as 0, and every other team would be denoted as 1. The same variables were removed due to multicollinearity as in the initial models, and finally, the levels of PostW_binary were changed from "0" and "1" to "level_0" and "level_1", as some of the models were running into errors with the 0 and 1 denotations.

After this data cleaning, the first binary model run was the Jackknife and Leave One Out Cross Validation (LOOCV) method. The parameters and arguments were relatively similar to the ones in the initial model, including running with 20 folds. The main differences included the fact that the summary function of the training control for cross validation is now a two-class summary instead of a multi-class summary, and the method of the model was now "glm" with family being set to binomial, whereas before the method was "polr". This model yielded an accuracy of 0.6528, and the accompanying confusion matrix can be found in Appendix E.1.

The next model run was a Decision Tree model. But before this model could be run, the data had to again be split into training and testing data, and a 50/50 split was used once again. Using the rpart function with a control parameter of cp = 0.001, where "cp" is known as the complexity parameter, we create a decision tree predicting PostW and plot it in Appendix E.2. An interpretation of this decision tree is as follows: If your team has more than 24.5 wins, you move to the right side of the tree. Then, if they had less than 205.5 total steals that season, you would

move to the next left branch. Finally, if your team had greater than or equal to 849.5 total defensive rebounds, this binary model would predict the team to win at least one postseason game (level_1). But if the team had less than 849.5 total defensive rebounds along with the previous parameters, the model would predict the team to lose its first postseason game (level_0). The confusion matrix and accuracy are then analyzed, as seen in Appendix E.3. We calculate the accuracy to be 0.6449, which is about 0.01 less than the jackknife and leave-one-out cross validation.

Next, a stepwise selection was run using a generalized linear model on the binary data, with the same splits for the training and testing data. This model selects the following variables for the PostW_binary formula: Wins, TOV%, total_NumOT (total number of overtime periods), total_Blk, TRB%, and total_DR. Based on the confusion matrix in Appendix E.4, we calculate an accuracy of 0.6572, which is just slightly better than that of the jackknife and LOOCV method.

Finally, a random forest model was run using the same training and testing splits. Using the binary training data and setting the number of decision trees in the random forest to 1000, we run the model and create the confusion matrix in Appendix E.5. We also see that the OOB Error rate is equal to 36.75%. In Appendix E.6, we see a plot of the misclassification rates for each value of PostW_binary by the number of trees in the random forest model. Finally, we apply the random forest model to the binary test data and print the confusion matrix and statistics, found in Appendix E.7. This leads to an accuracy of 0.6590, which is the highest accuracy of the initial models. This indicates that, based on the binary models of the data, a random forest model may be the best modeling technique to predict PostW_binary for each year. We then apply these random forest model predictions using the predict function to the regular data (not the binary

data), and select only when the predictions are equal to level_1, meaning that those teams won at least one postseason game. This will be referred to as the multiclass data, which is the data being used to make the final predictions of how many total postseason wins each team will ultimately achieve. As a quick way to see how well these binary predictions performed, only 15.5% of all of the multiclass data contained teams with actual PostW values of First Round Exit.

Final Models

Now that the final multiclass dataset is filtered and cleaned, the final models and predictions can be run and computed. The same variables that were removed in the first two modeling stages are removed again, along with total_FTM, total_Blk, AST%, and TS%, while total_Ast is added to the variable pool. The final data is once again split 50/50 into training and testing data. The first model run will be the Jackknife and Leave One Out Cross Validation. Similar to the first multiclass models, the parameters remain the same, using a multi-class summary for the summary function and "polr" for the training method. This only yields an accuracy of 0.4086, which is a lot worse than our initial models. However, accuracy is not the best performance metric to use, since there are a lot more instances of Round of Thirty-Two values for PostW, and thus, these values will heavily sway the results. Rather, the area under the ROC Curve is the better metric to use. Using the multiclass.roc function, we achieve a multiclass AUC of 0.6318. This result can also be found in Appendix F.1.

The next modeling technique used on this final data is a decision tree. The important parameters that differ from the binary model are that the control parameters for the train data use a 10-fold cross validation and that the train function uses the random forest method. A visual

representation of this decision tree can be found in Appendix F.2, with a multiclass AUC of 0.625, which is slightly lower than that of the Jackknife and LOOCV method. This result can be found in Appendix F.3.

Then, we move onto the stepwise selection method. Using the "polr" function, which fits a probit regression model on an ordered factor response, and going in both directions, starting with the null formula, we an ordered logistic regression. We found a multiclass AUC of 0.6265, which is slightly worse than the jackknife model as well. These results can also be found in Appendix F.4.

Finally, a random forest model is run on the final data. We set the number of decision trees in the random forest to 1000 and run the model. A confusion matrix from the model is produced in Appendix F.5, with an OOB Error rate equal to 62.50%. In Appendix F.6, we see a plot of the misclassification rates for each value of PostW by the number of trees in the random forest model. Finally, we apply the random forest model to the test data and print the multiclass roc, found in Appendix F.7. This leads to a multiclass AUC of 0.6265, which is slightly lower than the jackknife model. These final results indicate that the jackknife and LOOCV method, with an AUC of 0.6318, is the best predicting model.

The final step of this project is to apply these jackknife predictions to each year subset of the data. The first season we will complete this comparison for is the 2003 season, the first in the dataset. Looking at the multiclass data filtered to select only when Season is equal to 2003, we use the predict function to apply the jackknife model to this data, with type being set to probability, creating a predictions_jk variable. These values of predictions_jk will be a data frame of 7 columns, containing the probability each team will have in each of the 7 possible values of PostW. Next, an empty variable, predicted_PostW, is created to later store our predicted

PostW values. To select our predictions, we first look at the Champion column in the predictions_jk data frame and select the team with the highest probability. In the 2003 Season, this would be Kansas. Then, we move to the Runner Up column and select the team (excluding teams already with a predicted PostW value) with the highest probability. In the 2003 Season, this would be Syracuse. Then, we move to the Final Four column and select the two teams (excluding those already with a predicted PostW value) with the highest probabilities. This process is repeated until we find 4 teams for the Elite Eight value, 8 teams for the Sweet Sixteen value, 16 teams for the Round of Thirty-Two value, and any remaining teams to First Round Exit. Below are the predictions for the 2003 Season:

	PostW <ord>	predicted_PostW <ord>
Kansas	Runner.Up	Champion
Syracuse	Champion	Runner Up
Arizona	Elite.Eight	Final Four
Kentucky	Elite.Eight	Final Four
Butler	Sweet.Sixteen	Elite Eight
Duke	Sweet.Sixteen	Elite Eight
Pittsburgh	Sweet.Sixteen	Elite Eight
Florida	Round.of.Thirty.Two	Elite Eight
Texas	Final.Four	Sweet Sixteen
Oklahoma	Elite.Eight	Sweet Sixteen
Notre Dame	Sweet.Sixteen	Sweet Sixteen
Wisconsin	Sweet.Sixteen	Sweet Sixteen
Gonzaga	Round.of.Thirty.Two	Sweet Sixteen
Louisville	Round.of.Thirty.Two	Sweet Sixteen
Tulsa	Round.of.Thirty.Two	Sweet Sixteen
Xavier	Round.of.Thirty.Two	Sweet Sixteen
Connecticut	Sweet.Sixteen	Round of Thirty Two
California	Round.of.Thirty.Two	Round of Thirty Two
Indiana	Round.of.Thirty.Two	Round of Thirty Two
Missouri	Round.of.Thirty.Two	Round of Thirty Two
Stanford	Round.of.Thirty.Two	Round of Thirty Two
UNC Asheville	Round.of.Thirty.Two	Round of Thirty Two
Utah	Round.of.Thirty.Two	Round of Thirty Two

Some of the biggest errors occurred when the model predicted Florida to reach the Elite Eight, but they were eliminated in the Round of Thirty-Two, along with the model predicting Texas to lose in the Sweet Sixteen when they ended up making the Final Four.

Below are the model's predictions for the 2008 Season:

	PostW <ord>	predicted_PostW <ord>
North Carolina	Final.Four	Champion
Kansas	Champion	Runner Up
Memphis	Runner.Up	Final Four
UCLA	Final.Four	Final Four
Texas	Elite.Eight	Elite Eight
Duke	Round.of.Thirty.Two	Elite Eight
Siena	Round.of.Thirty.Two	Elite Eight
Drake	First.Round.Exit	Elite Eight
Davidson	Elite.Eight	Sweet Sixteen
Louisville	Elite.Eight	Sweet Sixteen
Tennessee	Sweet.Sixteen	Sweet Sixteen
WKU	Sweet.Sixteen	Sweet Sixteen
Wisconsin	Sweet.Sixteen	Sweet Sixteen
Georgetown	Round.of.Thirty.Two	Sweet Sixteen
Marquette	Round.of.Thirty.Two	Sweet Sixteen
Pittsburgh	Round.of.Thirty.Two	Sweet Sixteen
Xavier	Elite.Eight	Round of Thirty Two
Michigan St	Sweet.Sixteen	Round of Thirty Two
Stanford	Sweet.Sixteen	Round of Thirty Two
Villanova	Sweet.Sixteen	Round of Thirty Two
West Virginia	Sweet.Sixteen	Round of Thirty Two
Arkansas	Round.of.Thirty.Two	Round of Thirty Two
Butler	Round.of.Thirty.Two	Round of Thirty Two
Kansas St	Round.of.Thirty.Two	Round of Thirty Two
Miami FL	Round.of.Thirty.Two	Round of Thirty Two
Mississippi St	Round.of.Thirty.Two	Round of Thirty Two
Mt St Mary's	Round.of.Thirty.Two	Round of Thirty Two
Notre Dame	Round.of.Thirty.Two	Round of Thirty Two
Texas A&M	Round.of.Thirty.Two	Round of Thirty Two
BYU	First.Round.Exit	Round of Thirty Two
Oral Roberts	First.Round.Exit	Round of Thirty Two
UMBC	First.Round.Exit	Round of Thirty Two

Some of the biggest errors come from the model predicting Duke and Siena to make the Elite Eight when they both ended up losing in the Round of Thirty-Two. Another error occurred in the model predicting Xavier to lose in the Round of Thirty-Two, when they ended up making it to the Elite Eight.

Next, here are the model's predictions for the 2016 Season:

	PostW <ord>	predicted_PostW <ord>
Indiana	Sweet.Sixteen	Champion
Villanova	Champion	Runner Up
North Carolina	Runner.Up	Final Four
Kansas	Elite.Eight	Final Four
Oklahoma	Final.Four	Elite Eight
Notre Dame	Elite.Eight	Elite Eight
Oregon	Elite.Eight	Elite Eight
Duke	Sweet.Sixteen	Elite Eight
Virginia	Elite.Eight	Sweet Sixteen
Gonzaga	Sweet.Sixteen	Sweet Sixteen
Iowa St	Sweet.Sixteen	Sweet Sixteen
Maryland	Sweet.Sixteen	Sweet Sixteen
Kentucky	Round.of.Thirty.Two	Sweet Sixteen
Utah	Round.of.Thirty.Two	Sweet Sixteen
Xavier	Round.of.Thirty.Two	Sweet Sixteen
Chattanooga	First.Round.Exit	Sweet Sixteen
Texas A&M	Sweet.Sixteen	Round of Thirty Two
Wichita St	Sweet.Sixteen	Round of Thirty Two
Ark Little Rock	Round.of.Thirty.Two	Round of Thirty Two
Butler	Round.of.Thirty.Two	Round of Thirty Two
Connecticut	Round.of.Thirty.Two	Round of Thirty Two
Hawaii	Round.of.Thirty.Two	Round of Thirty Two
Iowa	Round.of.Thirty.Two	Round of Thirty Two
Michigan	Round.of.Thirty.Two	Round of Thirty Two
MTSU	Round.of.Thirty.Two	Round of Thirty Two
Providence	Round.of.Thirty.Two	Round of Thirty Two
SF Austin	Round.of.Thirty.Two	Round of Thirty Two
St Joseph's PA	Round.of.Thirty.Two	Round of Thirty Two
Yale	Round.of.Thirty.Two	Round of Thirty Two
Colorado	First.Round.Exit	Round of Thirty Two
CS Bakersfield	First.Round.Exit	Round of Thirty Two
Iona	First.Round.Exit	Round of Thirty Two
Syracuse	Final.Four	First Round Exit
Miami FL	Sweet.Sixteen	First Round Exit
VCU	Round.of.Thirty.Two	First Round Exit
Michigan St	First.Round.Exit	First Round Exit

Some of the biggest errors come from the model predicting Indiana to win the entire tournament when they ended up being eliminated in the Sweet Sixteen, along with predicting Syracuse to lose in the first round when in reality they made the Final Four. The largest error came in predicting Chattanooga to not only win in the first round (through the binary predictions), but to make the Sweet Sixteen, when in fact they were eliminated in the first round.

And finally, here are the predicted results if the 2020 March Madness tournament would have run uninterrupted:

	predicted_PostW <ord>
Gonzaga	Champion
Florida St	Runner Up
Syracuse	Final Four
UNC Greensboro	Final Four
Creighton	Elite Eight
Duke	Elite Eight
Furman	Elite Eight
Richmond	Elite Eight
Houston	Sweet Sixteen
Kansas	Sweet Sixteen
Maryland	Sweet Sixteen
Michigan St	Sweet Sixteen
Mississippi St	Sweet Sixteen
N Colorado	Sweet Sixteen
North Carolina	Sweet Sixteen
Wichita St	Sweet Sixteen
Arizona	Round of Thirty Two
Baylor	Round of Thirty Two
Bradley	Round of Thirty Two
Colgate	Round of Thirty Two
Connecticut	Round of Thirty Two
E Washington	Round of Thirty Two
Georgetown	Round of Thirty Two
Indiana	Round of Thirty Two
Iowa	Round of Thirty Two
NC State	Round of Thirty Two
Sacred Heart	Round of Thirty Two
San Diego St	Round of Thirty Two
St Louis	Round of Thirty Two
Villanova	Round of Thirty Two
West Virginia	Round of Thirty Two
Winthrop	Round of Thirty Two
Belmont	First Round Exit
Buffalo	First Round Exit
Dayton	First Round Exit
Duquesne	First Round Exit
ETSU	First Round Exit

My model predicts that Gonzaga would have beaten Florida St in the Championship game of the 2020 Tournament. The only results that are a little surprising are that UNC Greensboro is predicted to make the Final Four, along with Furman and Richmond being predicted to make the

Elite Eight. The culprit for these strange results along with the misclassifications in previous years may be due to the fact that the model does not consider strength of schedule, meaning that teams that put up similar numbers but one team faces tougher opponents than the other, they are treated as the same.

Conclusions and Future Analysis

Overall, even with our robust models, it is extremely difficult to predict March Madness games and outcomes, much less perfectly predict an entire 67 game bracket. However, with these models, the odds of performing well in your bracket pool can increase substantially. Going from a 50% chance to pick a bracket randomly to around 63% is a difference of about 9 games in a 67 game tournament, which is pretty significant.

While the methodologies of the models may have changed, the research questions remained mostly the same. The goal remained to create a model to best model and predict the NCAA Basketball Tournament. The only change from the start of the project to its conclusion is that there was less of a focus on the individual matchups of the tournament and more of a focus on the overall trends of the tournament, like what teams advanced into the further rounds, and who would end up winning.

In terms of the modeling techniques, all of them performed at almost the same success rate. The binary models all had an accuracy of around 0.65, and the final models all had areas under the curve around 0.62 – 0.63. This surprised me because I would have expected some of the more robust models to perform better, but it turns out that the simple stepwise selection performed just as well as the complex random forest model.

One area for further analysis lies in the class imbalance that still remains. Even after running the binary models, there still remains a lot more PostW values on the lower end (now Round of Thirty-Two instead of First Round Exit). While this isn't necessarily an issue, the problem arises because each PostW value is weighted equally. Since there are a lot of Round of Thirty-Two values, the models will predict a lot of these values and less of the higher values like Champion and Runner Up. While this is helpful in predicting games in that round, it's not as helpful in predicting the overall tournament. To combat this, there needs to be some sort of weighting system for PostW, where Champion is weighted more than Runner Up, which is weighted more than Final Four, and so on. In this same light, I would like to create a scoring function similar to that of ESPN to give a score to my predictions like it was a bracket submission. This will also help with weighting the PostW values differently, as correctly predicting the Champion adds more to the score than just predicting a First Round upset. As mentioned before, I would like to include strength of schedule information into the models, as it would most likely improve the models and reduce the number of outlier predictions. Finally, once the 2023-2024 Regular Season concludes, I want to evaluate my model by creating a bracket of my own for the 2024 Tournament and evaluate it in real time.

Appendix A – Data Sources

1. Kaggle: March Machine Learning Mania 2021 - NCAAM – Spread (https://www.kaggle.com/competitions/ncaam-march-mania-2021-spread/data)
2. Sports Reference (https://www.sports-reference.com/cbb/seasons/men/2003-school-stats.html)

Appendix B – Variable Summary

Basic Statistics:

- FGM – Field Goals made
- FGA – Field Goals attempted
- FGM3 – Three pointers made
- FGA3 – Three pointers attempted
- FTM – Free Throws made
- FTA – Free Throws attempted
- OR – Offensive Rebounds
- DR – Defensive Rebounds
- Ast – Assists
- TO – Turnovers Committed
- Stl – Steals
- Blk – Blocks
- PF – Personal Fouls

Advanced Statistics:

- SRS – Simple Rating System

    - Rating that considers average point differential and strength of schedule, where zero is average
- SOS – Strength of Schedule
    - Rating of strength of schedule, where zero is average
- Pace
    - Estimated number of possessions per 40 minute game
- Ortg – Offensive Rating
    - Estimated number of points scored per 100 possessions
- FTr – Free Throw Attempt Rate
    - Number of free throw attempts per field goal attempt
- 3PAr – 3 Point Attempt Rate
    - Percentage of field goal attempts from 3-point range
- TS% - True Shooting Percentage
    - Measure of shooting efficiency that takes into account 2 point field goals, 3 point field goals, and free throws
- TRB% - Total Rebound Percentage
    - Estimate of the percentage of available rebounds a player grabbed while they were on the floor
- AST% - Assist Percentage
    - Estimate of the percentage of teammate field goals a player assisted while on the floor
- STL% - Steal Percentage
    - Estimated percentage of opponent possessions that end with a steal by a player while on the floor
- BLK% - Block Percentage
    - Estimated percentage of opponent two point field goal attempts blocked by the player while on the floor
- eFG% - Effective Field Goal Percentage
    - Statistic that adjusts for the fact that a 3 point field goal is worth one more point than a 2 point field goal
- TOV% - Turnover Percentage
    - Estimated number of turnovers per 100 plays
- ORB% - Offensive Rebound Percentage
    - Estimated percentage of available offensive rebounds a player grabbed while on the floor

- FT/FGA – Free Throws per Field Goal Attempt

Appendix C – Variable Exploration

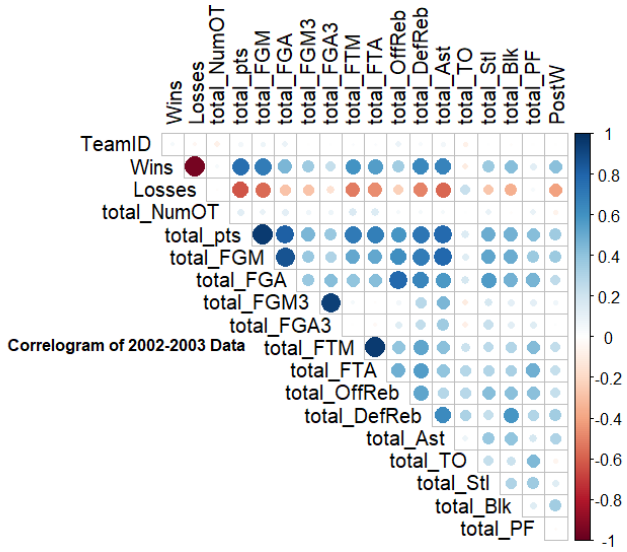

Figure 1: Correlogram of 2002-2003 Regular Season Data

	Abs_Correlation_with_PostW <dbl>
PostW	1.0000000000
Wins	0.4106481860
Losses	0.4002276450
total_FGM	0.3656145185
total_pts	0.3566948090
total_DefReb	0.3482725527
total_Blk	0.3441577293
total_Ast	0.3091191155
total_FGA	0.2520755029
total_FTM	0.2457530774

Figure 2: Variable Importance Table for 2002-2003 Regular Season Data

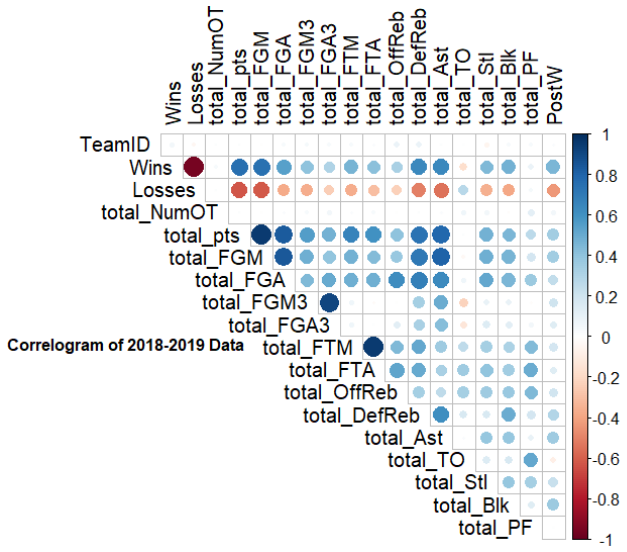

Figure 3: Correlogram of 2018-2019 Regular Season Data

	Abs_Correlation_with_PostW <dbl>
PostW	1.000000000
Wins	0.456677132
Losses	0.433628437
total_Blk	0.360496112
total_Ast	0.353301867
total_FGM	0.349301174
total_pts	0.345563879
total_DefReb	0.298290208
total_FGA	0.245659940
total_Stl	0.225706425

Figure 4: Variable Importance Table for 2018-2019 Regular Season Data

	Change_in_Correlation_with_PostW <dbl>
total_FGM3	0.132215170
total_FGA3	0.120927098
total_NumOT	0.113663996
total_Stl	0.092570658
Wins	0.046028946
total_Ast	0.044182752
total_PF	0.029154932
TeamID	0.022397163
total_Blk	0.016338382
PostW	0.000000000

Figure 5: Change in Variable Importance from 2002-2003 Regular Season to 2018-2019 Regular Season

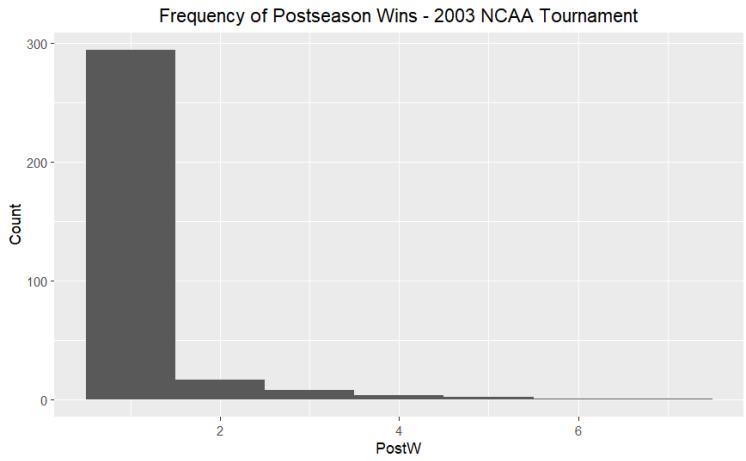

Figure 6: Histogram of Postseason Wins in 2003 NCAA Tournament

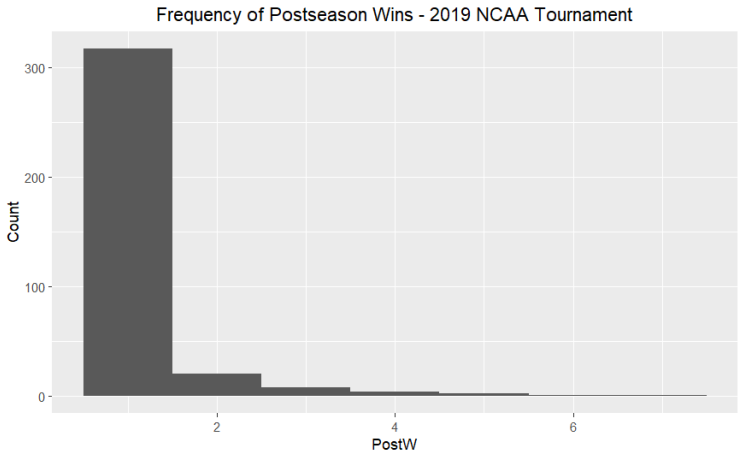

Figure 7: Histogram of Postseason Wins in 2019 NCAA Tournament

Appendix D – Initial Model Builds

Cross-Validated (20 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

	Reference						
Prediction	First.Round.Exit	Round.of.Thirty.Two	Sweet.Sixteen	Elite.Eight	Final.Four	Runner.Up	Champion
First.Round.Exit	59.3	8.2	3.8	1.6	0.5	0.0	0.0
Round.of.Thirty.Two	4.9	7.7	1.6	2.2	1.6	0.0	0.0
Sweet.Sixteen	1.1	1.1	1.6	1.1	0.5	0.0	0.5
Elite.Eight	0.0	0.5	0.0	0.5	0.0	0.0	0.5
Final.Four	0.0	0.0	0.5	0.0	0.0	0.0	0.0
Runner.Up	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Champion	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Accuracy (average) : 0.6923

Figure 1: Confusion Matrix for Jackknife and Leave-One-Out Cross Validation Method of 2003-2006 Data

	Overall <dbl>
total_FGM	100.000000
Wins	81.394585
total_TO	55.854974
total_Ast	47.837940
total_Blk	41.246167
total_NumOT	32.281814
total_Stl	13.669248
total_FGM3	12.120005
total_PF	8.898593
total_OffReb	7.227793

Figure 2: Variable Importance Table for Jackknife and Leave-One-Out Cross Validation Method of 2003-2006 Data

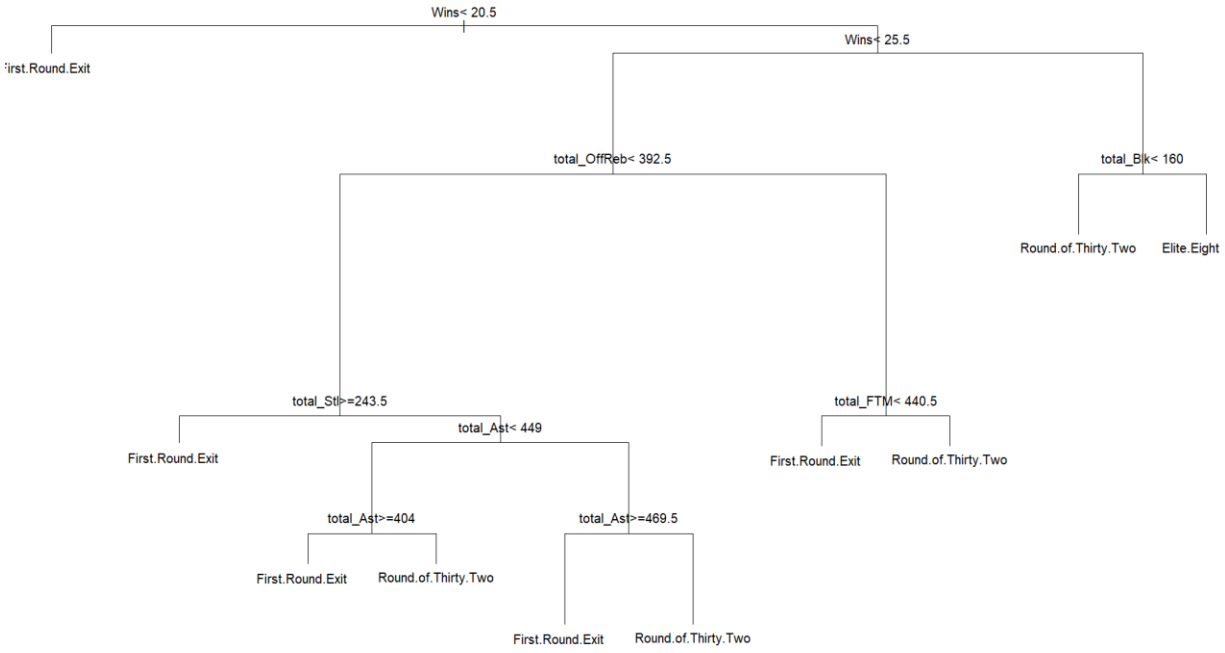

Figure 3: Visual Decision Tree for Initial Model of 2003-2006 Data

```
Confusion Matrix and Statistics

                    Reference
Prediction           First.Round.Exit Round.of.Thirty.Two Sweet.Sixteen Elite.Eight Final.Four Runner.Up Champion
  First.Round.Exit                31                   6             3           1          1         0        0
  Round.of.Thirty.Two              2                   1             1           1          0         0        0
  Sweet.Sixteen                    0                   1             0           0          0         0        0
  Elite.Eight                      1                   0             0           0          0         0        0
  Final.Four                       0                   0             0           0          0         0        0
  Runner.Up                        0                   0             0           0          0         0        0
  Champion                         0                   0             0           0          0         0        0

Overall Statistics

               Accuracy : 0.6531
                 95% CI : (0.5036, 0.7833)
    No Information Rate : 0.6939
    P-Value [Acc > NIR] : 0.7831

                  Kappa : 0.1014

 Mcnemar's Test P-Value : NA
```

Figure 4: Confusion Matrix for Decision Tree Method of 2003-2006 Data

```
reg.step2003_06_pred  First.Round.Exit Round.of.Thirty.Two Sweet.Sixteen Elite.Eight Final.Four Runner.Up Champion
  First.Round.Exit                32                   6             3           1          1         0        0
  Round.of.Thirty.Two              1                   2             1           1          0         0        0
  Sweet.Sixteen                    0                   0             0           0          0         0        0
  Elite.Eight                      1                   0             0           0          0         0        0
  Final.Four                       0                   0             0           0          0         0        0
  Runner.Up                        0                   0             0           0          0         0        0
  Champion                         0                   0             0           0          0         0        0
Accuracy: 0.6938776
```

Figure 5: Confusion Matrix for Stepwise Selection Method of 2003-2006 Data

```
Call:
 randomForest(formula = PostW ~ ., data = initial_model_03_06_forest[split.down_03_06,      ], ntree = 1000, importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 3

        OOB estimate of  error rate: 54.17%
Confusion matrix:
                  First.Round.Exit Round.of.Thirty.Two Sweet.Sixteen Elite.Eight Final.Four Runner.Up Champion class.error
First.Round.Exit                15                   2             1           0          0         0        2   0.2500000
Round.of.Thirty.Two              2                   5             1           1          0         0        0   0.4444444
Sweet.Sixteen                    4                   1             2           0          1         0        0   0.7500000
Elite.Eight                      2                   0             0           0          0         0        1   1.0000000
Final.Four                       1                   1             1           0          0         0        0   1.0000000
Runner.Up                        1                   1             0           0          0         0        0   1.0000000
Champion                         1                   0             1           1          0         0        0   1.0000000
```

Figure 6: Confusion Matrix for Initial Random Forest Model of 2003-2006 Data

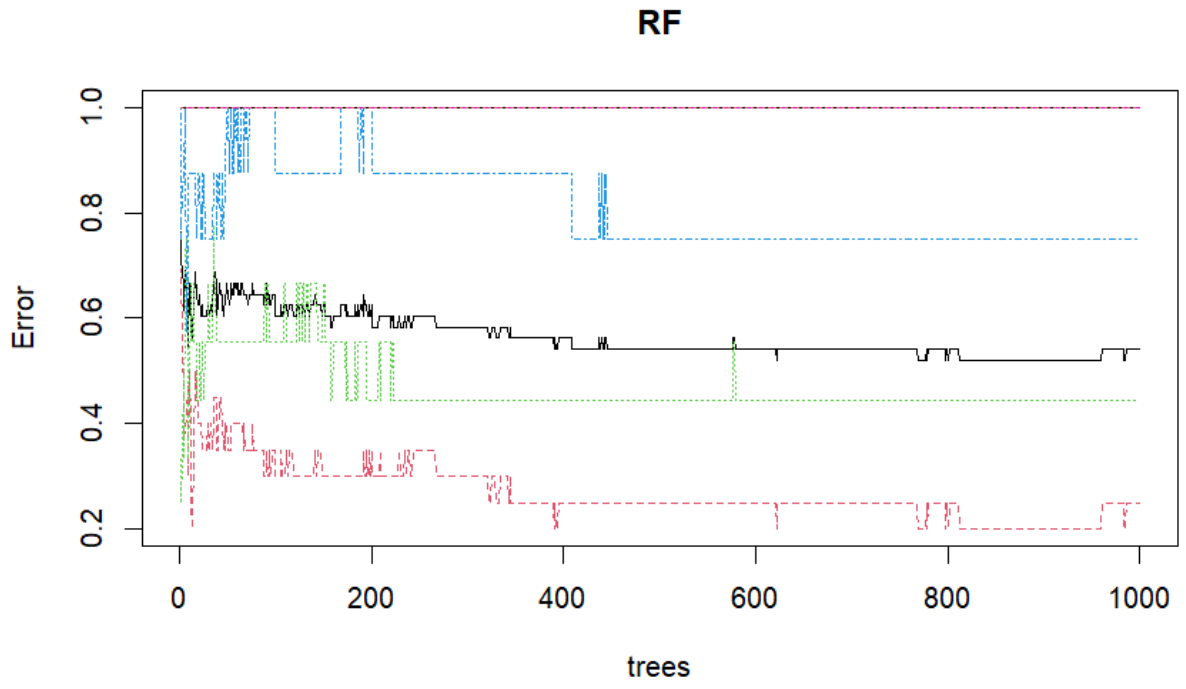

Figure 7: Graph of Misclassification Rates vs Number of Trees for Random Forest Model of 2003-2006 Data

```
Confusion Matrix and Statistics

                  Reference
Prediction         First.Round.Exit Round.of.Thirty.Two Sweet.Sixteen Elite.Eight Final.Four Runner.Up Champion
  First.Round.Exit               144                 24            9           8          4         0        0
  Round.of.Thirty.Two              4                  9            3           3          0         0        0
  Sweet.Sixteen                    2                  2            0           0          0         0        0
  Elite.Eight                      0                  0            0           0          0         0        0
  Final.Four                       0                  0            0           0          0         0        0
  Runner.Up                        0                  0            0           0          0         0        0
  Champion                         0                  0            0           0          0         0        0

Overall Statistics

               Accuracy : 0.7217
                 95% CI : (0.6562, 0.7809)
    No Information Rate : 0.7075
    P-Value [Acc > NIR] : 0.3563

                  Kappa : 0.2124

 Mcnemar's Test P-Value : NA
```

Figure 8: Confusion Matrix for Predictions of Random Forest Model of 2003-2006 Data

Appendix E – Secondary Model Builds

```
Cross-Validated (20 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction level_0 level_1
   level_0    30.7    16.7
   level_1    18.0    34.5

 Accuracy (average) : 0.6528
```

Figure 1: Binary Confusion Matrix for Jackknife and LOOCV Method of All Data

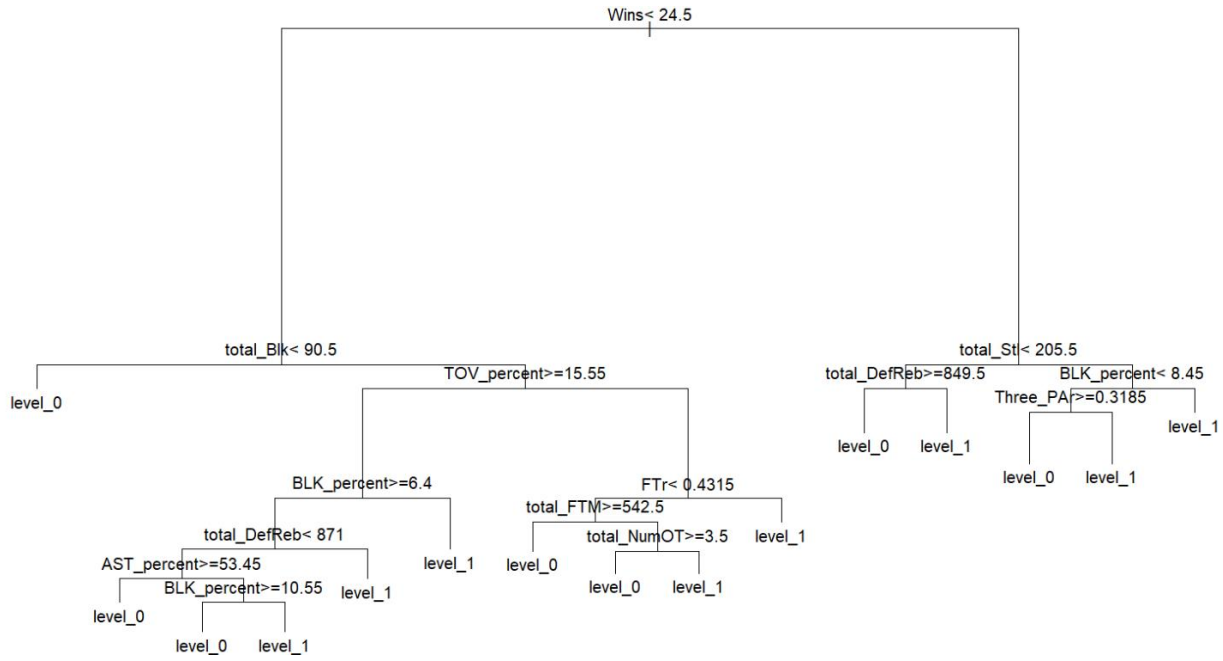

Figure 2: Visual Decision Tree for Binary Model of All Data

```
Confusion Matrix and Statistics

          Reference
Prediction level_0 level_1
   level_0     231     156
   level_1      45     134

               Accuracy : 0.6449
                 95% CI : (0.6039, 0.6843)
    No Information Rate : 0.5124
    P-Value [Acc > NIR] : 1.352e-10

                  Kappa : 0.2962

 Mcnemar's Test P-Value : 8.574e-15
```

Figure 3: Confusion Matrix for Binary Decision Tree Method of All Data

```
reg.step_binary_class level_0 level_1
                    0     180      98
                    1      96     192
Accuracy: 0.6572438
```

Figure 4: Confusion Matrix for Binary Stepwise Selection of All Data

```
        OOB estimate of  error rate: 36.75%
Confusion matrix:
        level_0 level_1 class.error
level_0     175     101   0.3659420
level_1     107     183   0.3689655
```

Figure 5: Confusion Matrix for Binary Random Forest Model of All Data

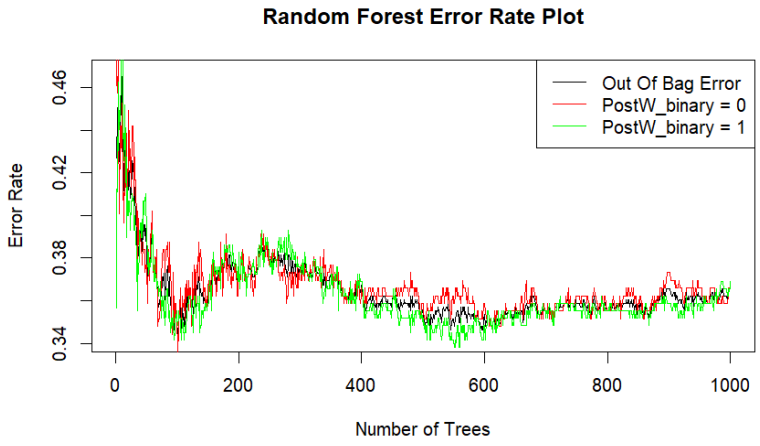

Figure 6: Graph of Misclassification Rates vs Number of Trees for Binary Random Forest Model

```
Confusion Matrix and Statistics

          Reference
Prediction level_0 level_1
   level_0     190     107
   level_1      86     183

               Accuracy : 0.659
                 95% CI : (0.6183, 0.698)
    No Information Rate : 0.5124
    P-Value [Acc > NIR] : 1.239e-12
```

Figure 7: Confusion Matrix for Predictions of Binary Random Forest Model of All Data

Appendix F – Final Model Builds

```
Call:
multiclass.roc.default(response = test$PostW, predictor = pred_matrix_jk)

Data: multivariate predictor pred_matrix_jk with 7 levels of test$PostW: First.Round.Exit, Ro
Multi-class area under the curve: 0.6318
```

Figure 1: Multiclass AUC for Final Jackknife and LOOCV Model

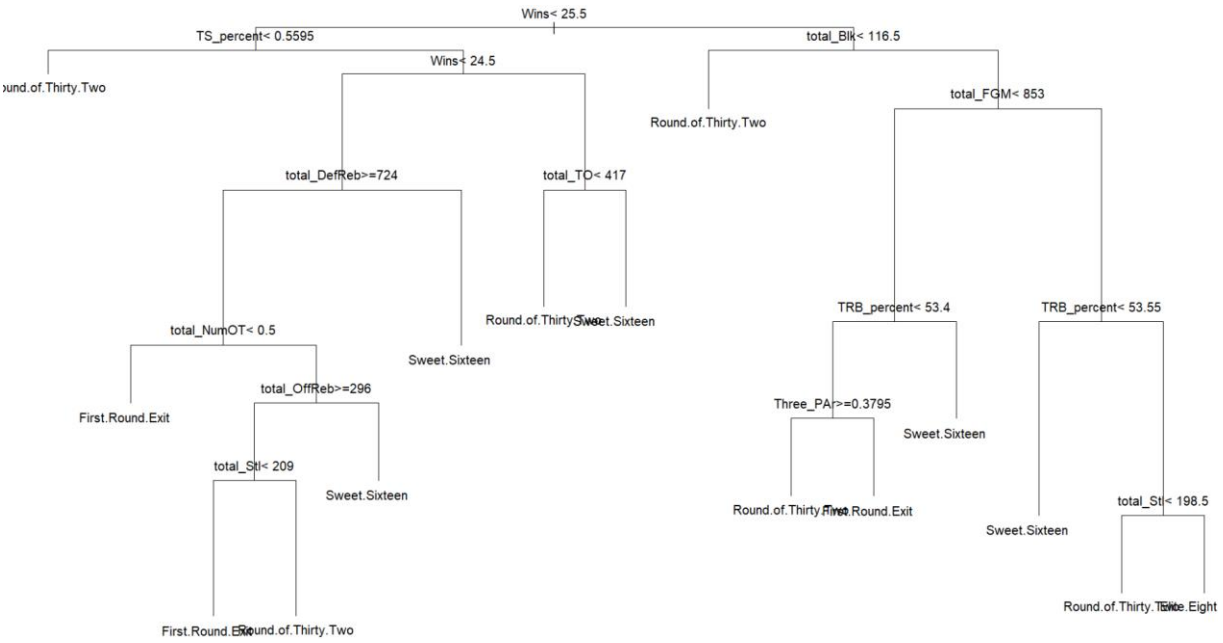

Figure 2: Visual Decision Tree for Final Model

```
Call:
multiclass.roc.default(response = test$PostW, predictor = pred_prob_tree)

Data: pred_prob_tree with 7 levels of test$PostW: First.Round.Exit, Round.of.Th
Multi-class area under the curve: 0.625
```

Figure 3: Multiclass AUC for Final Decision Tree Model

```
Call:
multiclass.roc.default(response = test$PostW, predictor = rf_pred)

Data: rf_pred with 7 levels of test$PostW: 0, 1, 2, 3, 4, 5, 6.
Multi-class area under the curve: 0.6265
```

Figure 4: Multiclass AUC for Final Stepwise Selection Model

```
Call:
 randomForest(formula = PostW ~ ., data = train, ntree = 1000,      importance = TRUE)
               Type of random forest: classification
                     Number of trees: 1000
No. of variables tried at each split: 4

        OOB estimate of  error rate: 62.5%
Confusion matrix:
  0  1  2 3 4 5 6 class.error
0 4 34  5 0 0 0 0   0.9069767
1 5 89 18 2 0 0 0   0.2192982
2 7 37 10 3 0 0 0   0.8245614
3 3 18 11 2 0 0 1   0.9428571
4 0 11  2 2 0 0 0   1.0000000
5 1  3  1 2 0 0 0   1.0000000
6 1  0  7 1 0 0 0   1.0000000
```

Figure 5: Confusion Matrix for Final Random Forest Model of All Data

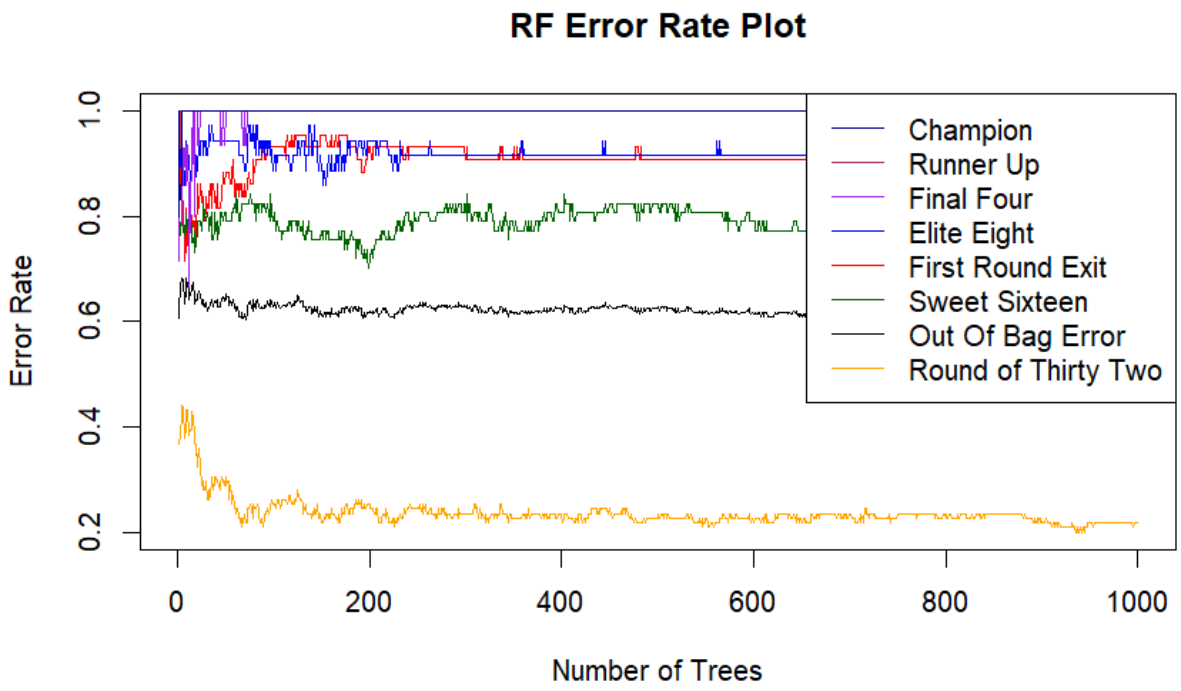

Figure 6: Graph of Misclassification Rates vs Number of Trees for Final Random Forest Model

```
Call:
multiclass.roc.default(response = test$PostW, predictor = rf_pred)

Data: rf_pred with 7 levels of test$PostW: 0, 1, 2, 3, 4, 5, 6.
Multi-class area under the curve: 0.6265
```

Figure 7: Multiclass AUC for Final Random Forest Model