March Modeling

Quinn Fargen

Advised by:

Thomas Brandenburger, Ph.D.

Senior Capstone MATH 401

Department of Mathematics and Statistics

South Dakota State University

Fall 2017

1. Abstract

March Madness is the annual college basketball tournament which many fans participate each year in trying to fill out their prediction of the 64-team bracket. With many different regular season statistics available to predict bracket matchups, mathematical models do great at making decent bracket matchup predictions. With so many different statistical variables available, a random forest model is an appropriate initial method of choice.

The methods and processes of random forest models, along with the underlying decision trees, will be shown in detail. Possible error values will be found for the specific 2-class classification method that will be used with the random forest models. Multiple types of statistical models will be created to predict specific matchups in certain recent brackets. The results will be compared to three million bracket entries from ESPN to see how the random forest models stack up.

2. March Modeling

Every march the NCAA college men's basketball tournament takes place. This tournament consists of sixty-four teams in a single elimination bracket of sixty-three games to determine the champion. Every year fans fill out brackets trying to predict the outcomes of the games. Somehow nobody has yet to make a perfect bracket from the 9.2 quintillion possible combinations each year. Recently an article written by Matthew Menzel in Math Horizons tried to model the tournament. Menzel used historical probabilities from the records of the previous thirty-two years' of tournament matchups by similar seeds to build his model. The likelihood of past tournament results was found by multiplying the probability of specific matchups for the outcomes of all sixty-three games.

The amount of regular season statistics of each team online is staggering. There are many different models and methods for using this data to predict future tournaments. The difficult question is which statistics to use. A random forest is a viable option with its unique strength aligning perfectly for this question. Random forests have a way of determining which statistics are the most important through its method of creating decision trees. These methods and strengths will be better explained in the next section. Following that section will be an explanation of the data used and models that will be built to help fans fill out better brackets in years to come.

3. Decision Tree

A very summarized description of a random forest is using the average of a large collection of decision trees. Therefore, understanding a random forest will start from

understanding the makings of a decision tree. There are two main types of decision trees: regression and classification trees. The main difference is the response type being predicted since a regression tree is for continuous values while the classification tree is for categorical values.

Just like every real-life tree, there are branches and leaves on every decision tree. Every branch represents a partition of the data and leaves will be the final regions of the data grouped by specific branches. The variables can be denoted as X_1, X_2, …, X_P where P is the number of possible variables. Every new branch of the decision tree will be when a partition is created at value *t* within a variable X_j. This partition will create two new regions called R_j and R_{j+1}. All the observations in the data with values of X_j less than or equal to the value of the

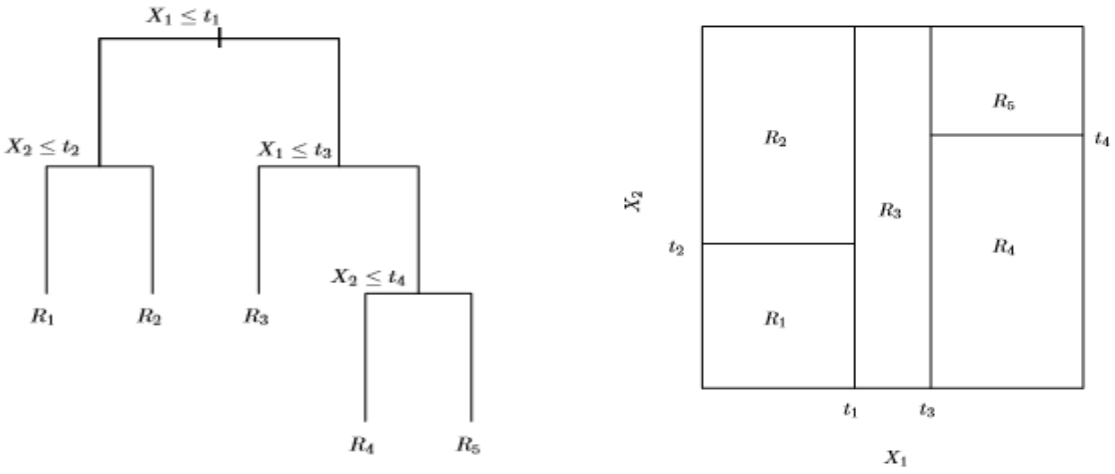


Figure 1: On the left is a decision tree. The picture to the right shows the corresponding regions [4].

partition value *t* will be in the first region and all the observations in the second region will be greater than the value *t*.

Shown in Figure 1 is an example of a basic decision tree and the regions of the two values within it. This is a somewhat trivial decision tree since it only contains two variables, but

still gives a great representation of the basic structures. Starting at the top of the decision tree, "$X_t \leq t_1$" is the split for the first partition of the data. All values of X_1 are placed to the left of the tree if they are less than or equal to t_1. The rest of the values of X_1, which are greater than t_1 are placed to the right of the tree. This same partition of X_1 is visible in the region plot with t_1 shown on the bottom border of the plot. The two resulting regions on the left side of the tree are also shown on the left side of the region plot.

The partition of the data in R_j will then have the same process done to just one of the two created regions. This will create two new regions from a new partition within the previous partition. This process will continue to create regions within regions. A prediction for the response variable will be created and will be the same for every observation that falls within the same partitions as the model. The leaves of the tree will be where the predictions lie in the general picture of the model. The type of decision will depend on if the tree is a regression or classification model. A prediction from a regression tree most commonly will use the mean value of the test data in the region. A classification model will use typically the mode of the observations that fall into each leaf of the model.

The goal of every regression decision tree is to minimize the residual sum of squares (RSS) within every region of the tree. This is the equation for RSS,

$$RSS = \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \hat{y}_{R_j})^2 \quad (1)$$

where $\hat{y}_{R_j}$ is the designated prediction value for the entire region R_j and all the values of X_p that fall into that regions partition of the data [4]. It is easy to see that this equation is then just

the sum of each squared error of each value $\mathcal{Y}_i$ from the given value for the region $\hat{\mathcal{Y}}_{R_j}$. This equation is the basis of every partition value *t* in every new region.

The idea of recursively creating regions within regions is straightforward enough to follow. The next step the decision tree needs to decide is when to stop making partitions. If the model kept partitioning the data, every variable would have its own partition. This would excessively overfit the model to the training data set. On the other hand, stopping the partitions early might have the model miss a significant split further down a potential branch.

For regression decision trees an optimal tree size can be found using the cost-complexity pruning process. This starts off by creating the largest possible tree and only stopping the partitions when a selected minimum number of values are left in each node. Here is the equation for the cost-complexity value,

$$CP = \sum_{j=1}^{|\mathrm{T}|} \sum_{i:x_i \epsilon R_j} (\mathcal{Y}_i - \hat{\mathcal{Y}}_{R_j})^2 + \alpha|\mathrm{T}| \qquad (2)$$

where $|\mathrm{T}|$ is the number of nodes on the tree *T* which is a tree subset of the largest tree T_0 which was grown [4]. This formula is like the RSS equation (1) but has an added penalty variable $\alpha \geq 0$ multiplied to $|\mathrm{T}|$ added to the equation. This α variable allows the model to be tuned into the desired tradeoff between complexity of nodes and fit of the tree. The optimal tree size of T_0, is T_α which is equal to the cost-complexity value at α.

Classification decision trees are not able to use this same RSS or cost-complexity pruning method since they do not use continuous values for prediction. Since these predictions are instead based on the mode of a region, the error calculation is based off proportions of classes

in the given region. There are three commonly used calculations for classification trees that are used similarly to RSS. Here is how the proportion of a class *k* in a certain region *j* is calculated,

$$\hat{p}_{jk} = \frac{1}{N_j} \sum_{x_i \in R_j} I(\mathcal{Y}_i = k) \tag{3}$$

where N_j is the number of observations in the region and I is the indicator function giving a 1 for every $\mathcal{Y}_i = k$ [3].

The first of the error calculations is the misclassification error. This is the most commonly used error term in classification trees. Here is the calculation for this error,

$$M = 1 - \hat{p}_{j\,mode(k)} = \frac{1}{N_j} \sum_{x_i \in R_j} I(\mathcal{Y}_i \neq max(\hat{p}_{jk})) \tag{4}$$

where *mode(k)* is the class *k* which is the mode of the region *j*. Also, shown here is $max(\hat{p}_{jk})$, meaning the maximum proportion of class *k* in region *j* [3]. The second common error calculation is the Gini index. Here is the calculation for this error [3].

$$G = \sum_{k=1}^{K} \hat{p}_{jk} (1 - \hat{p}_{jk}) \tag{5}$$

This calculation is straightforward as the sum of a proportion multiplied by one minus the proportion. The third calculation is a close alteration of the Gini index, and it is called Cross-Entropy [3].

$$C = -\sum_{k=1}^{K} \hat{p}_{jk} \log \hat{p}_{jk} \tag{6}$$

Since the $\hat{p}_{jk}$ will always be less than or equal to one, the $\log \hat{p}_{jk}$ will initially always be negative. This calculation will always be positive though like the other three since it includes a negative sign.

If a classification tree has only two classes, it is possible to show the values of the three error calculations that are possible. This is the exact scenario that will happen for the models in the next section. The two classes will be either win or loss since the models will be made to predict individual game matchups in the bracket. Since there are only two classes there will be only two proportions per region as p_{j0} and p_{j1}. The two proportions will be related as $p_{j1} = 1 - p_{j0}$ [3].

Misclassification error can be turned into a function of p_{j0} here,

$$M = 1 - \hat{p}_{j\,mode(1,2)}$$

$$M(p_{j0}) = 1 - max(p_{j0}\,,\ (1 - p_{j0})) \tag{7}$$

where *mode(1,2)* is the class which occurs most frequently in R_j. This function is not differentiable, but it is easy to think of the possible values. If p_{j0} is close to zero or one, the function will output an error close to zero. If p_{j0} is close to .5, then $(1 - p_{j0})$ will also be close to $\frac{1}{2}$ and this will be the where the function has the biggest error output of $\frac{1}{2}$. This makes sense since having a fifty-fifty proportion of either class does not help predict either way and should have the biggest error term.

Similarly, here is the Gini index turned into a function of p_{j0},

$$G = (p_{j0}(1 - p_{j0})) + (p_{j1}(1 - p_{j1}))$$

$$G(p_{j0}) = (p_{j0}(1 - p_{j0})) + (1 - p_{j0})(1 - (1 - p_{j0}))$$

$$= (p_{j0}(1 - p_{j0})) + (p_{j0}(1 - p_{j0}))$$

$$= 2(p_{j0}(1 - p_{j0})) \; = \; 2p_{j0} - 2{p_{j0}}^2 \tag{8}$$

which is differentiable to find the location of the maximum error. Setting the first derivative of the function equal to zero shows a critical point at $\frac{1}{2}$.

Finally, Cross-entropy can be turned into a function of p_{j0} here,

$$C = -(p_{j0}\log(p_{j0}) + p_{j1}\log(p_{j1}))$$

$$C(p_{j0}) = -\left(p_{j0}log(p_{j0})\right) - \left((1 - p_{j0})log(1 - p_{j0})\right) \tag{9}$$

which aslo can be differentiated to find the location of the maximum error. Here the first derivative is shown for the function of Cross-entropy,

$$C'(p_{j0}) = \; -\log(p_{j0}) + \log(1 - p_{j0})$$

Which setting this equal to zero shows another critical point at $\frac{1}{2}$. Each of the three error terms showed a maximum error at $\frac{1}{2}$ which will make sense when looking at the plot of the functions in figure 2 where the x-axis is values of p_{j0} and the y-axis is error values.

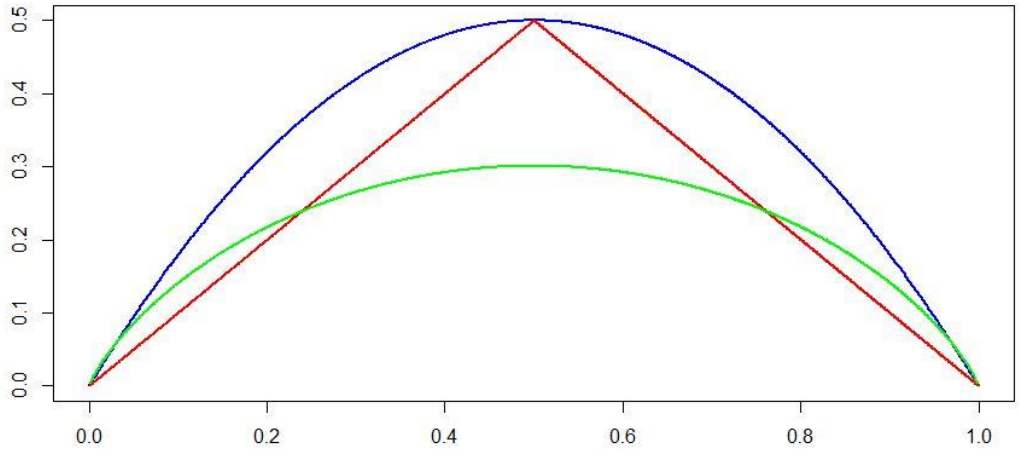


Figure 2: Error curves for Misclassification (Red), Gini index (Blue), & Cross-Entropy

4.

Random Forest

Making one decision tree will usually produce a reasonably accurate result. There are a few different types of models though that try to make use of many decision trees. These models have tweaked differences and use the average of the collection of trees to create a better overall model. The main types of these are called bagging, boosting, and random forests. Each type has a unique difference and goal to them. Bagging methods are trying to reduce the variance by using a bootstrap method of building many trees on the created training data, while in the boosting methods trees are built based off previous trees from modified versions of the original data with an emphasis on reducing bias [3].

Random forests are very similar to the bagging method with the same use of the bootstrap method of creating training data. From each bootstrapped dataset, a random forest with *B* number of trees are grown with the following rules. B is a selected value, but is usually chosen to be a large number. Every tree T_b will start by having *m* variables randomly selected from the *p* possible variables. For most models *m* is selected to be equal to $\sqrt{p}$, or a selected value. Next the best variable out of the *m* variables and the value *t* of the variable which results in the lowest RSS are both selected for regression trees [3].

If the forest is made up of classification trees, then the value *t* is chosen to reduce one of the three error terms explained. This will split the current tree into two branches based on the selected variable and value *t*. These two new branches will have the same process of random variables and *t* value selections occur within the branches. This will continue until a branch has the selected minimum number of observations which creating two new branches will not be attempted. This process will continue until *B* trees have been grown on each of the bootstrapped training datasets [4].

All the trees in the random forest can be shown as a collection as $\{\mathrm{T}_b\}_1^B$. Since in a regression tree the prediction value is the mean value, the prediction from a random forest for regression will look like this [3].

$$\hat{f}_{rf}^B(x) = \frac{1}{B}\sum_{b=1}^{B} \mathrm{T}_b(x) \tag{10}$$

The predicted value for x in the random forest is the mean of the predicted value from every tree in the forest. For a classification tree the prediction value is the mode of the region, so the prediction from a random forest for classification will look like this [3].

$$\hat{C}_{rf}^B(x) = mode\left\{\hat{C}_b(x)\right\}_1^B \tag{11}$$

5. Data Analysis

Most of the data to build the model will come from two sources. The first source is the website “Kaggle” that hosts a competition every year to see who can predict the tournament the most accurately each year [5]. They host datasets that have very detailed results of the regular seasons games as well as tournament game stats for the last fourteen seasons. These detailed datasets contain statistics on scores, field goals, free throws, rebounds, assists, turnovers, steals, blocks, fouls, and a few others for both the winning and losing teams. These include statistics for every game played since the beginning of the 2003 season. There are 76,636 regular season games as well as 914 tournament games included.

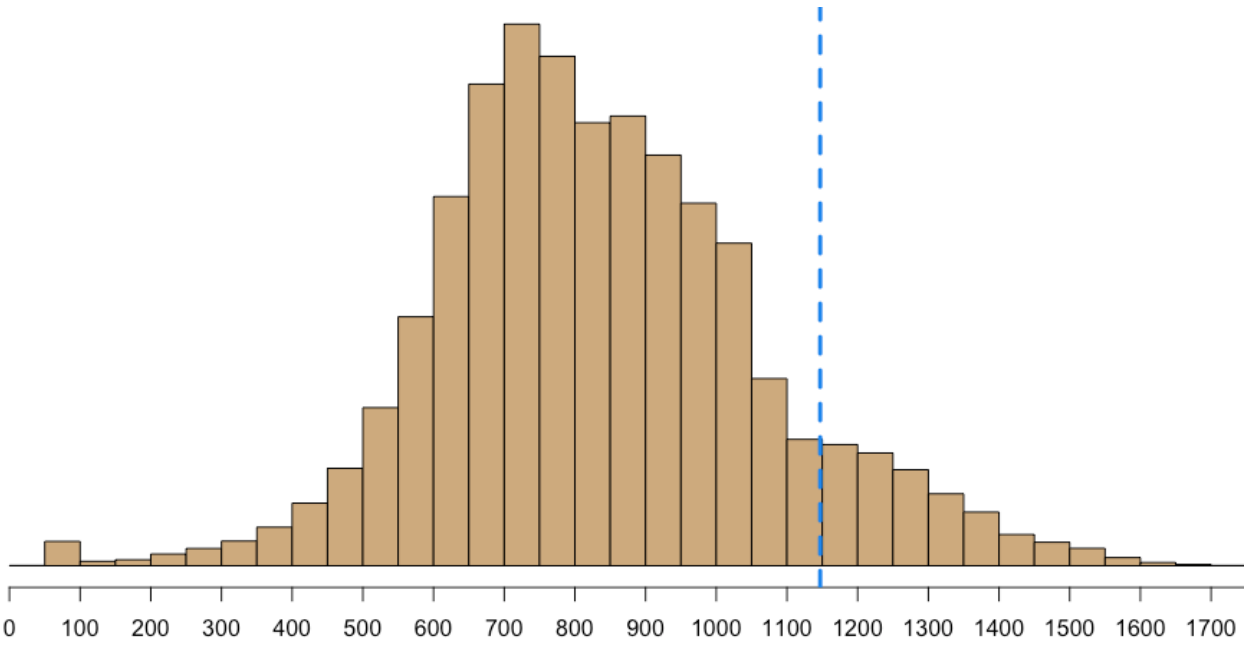
The second source of data will be the ranking of the teams from Adit Deshpande, who scraped some important data from the website “Sports-Reference” [7]. This data had a dataset

for each of the fourteen seasons with three important variables: strength of schedule(SOS), offensive simple rating system(OSRS), and defensive simple rating system(DSRS). The strength of schedule statistic will help show the difference between two teams with 25 wins a piece, by showing which of the 25 wins were tougher to obtain. The offensive simple rating system is a combination of many offensive stats and likewise the defensive rating. These will help determine which teams are much better or weak at certain aspects of the game. It will be interesting to see a defensive minded team matchup with an offensively challenged team.

There are 364 teams included to some degree throughout the regular season stats. Every tournament only includes 64 team though, since play-in games will be ignored because they are not required to be picked in ESPN bracket pools. Most of the data cleaning will involve preparing the data to have a summary of regular season stats for every team entering the given postseason tournament. Of the 64 teams in each tournament, there will be 63 matchups for which the models will give a probability of having Team 1 or Team 2 win. The model will be given some regular season summary stats in the form of the difference between each team's stats at each of the nineteen variables. Through this comparative method of taking the difference, the vector of stats will be able to show the separation of qualities and showcase the excellence of certain teams over the opponent.

Once the models predicted brackets are scored, they will be compared to the distribution of brackets from the 2015 entries on ESPN. Kanat Bekt posted three million bracket entries he scraped from ESPN's 2015 NCAA Tournament Challenge [2]. This will give a good

understanding of how well the models would have done in an actual bracket pool. Shown below in figure 3 is the distribution of the three million bracket scores in 2015.



6. Model Results

Four different models were used to predict three recent years' bracket scores. Random Forest, Decision Tree, Bagging, and Boosting were the four types of models. For each matchup in the tournament, a team was randomly assigned to be either Team 1 or Team 2. The models were created to predict the probability that Team 1 won the matchup. For each team the following statistics were used as variables shown in figure 4. Each team in every matchup had

					Rank	SOS	Seed	OSRS	DSRS				
Wins	Losses	Score	Sfgpct	Sfg3pct	Sftpct	Sor	Sdr	Sast	Sto	Sstl	Sblk	Spf	Round

Figure 4: These are the 19 total variables. The purple first top five are the most important of the variables.

these variables for the specific year. The difference of Team 1's minus Team 2's variables is what the model was given for each matchup.

The first five variables in purple are summary statistics that represent the overall team's abilities. Rank is a statistic from the "sports-reference" website that placed all the D1 schools in order of best to worst each season. SOS is a statistic for strength of schedule which shows how tough of games a team played, also from "sports-reference". Seed is the placement of the team by the NCAA in the region of the bracket from 1 to 16. OSRS and DSRS are the offensive and defensive simple rating system scores that represent the quality of a teams play on either side of the court, which is also from "sports-reference".

The second group of 14 variables in green are mostly regular season average statistics. The first two variables of Wins and losses are the number of regular season games won or loss. The next eleven are averages from regular season games. In order here are the full names of the eleven abbreviations shown above as score, field goal percentage, three-point field goal percentage, free throw percentage, offensive rebounds, defensive rebounds, assists, steals, blocks, and player fouls.

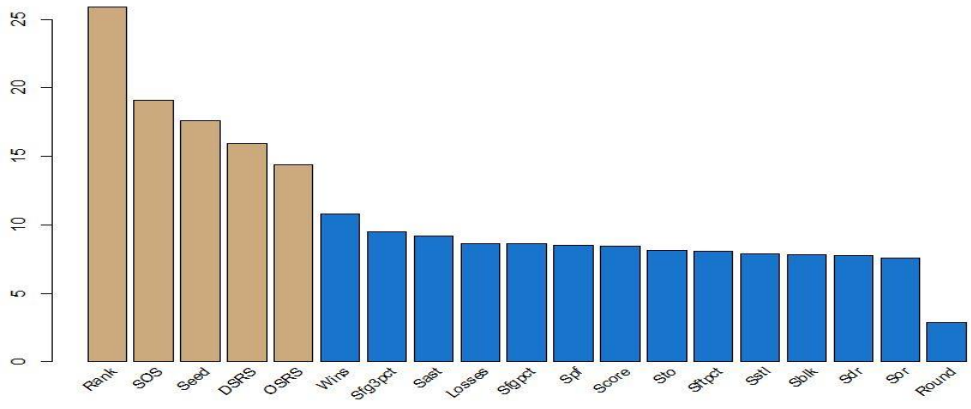


Figure 5: Plot of the importance of variables from a specific random forest model.

These are the common standard of statistics that are kept for every regular season game. The last green variable is Round, which is the round in the bracket for which each game takes place from 1 to 6. This variable was used to see if for example, playing in a championship game had any importance over just a first-round game. Shown in figure 5 is the plot of importance of variables from one of the random forest models. From this plot it is clear to see that the first five variables have a much higher statistical importance in this model than the other 14. Seeing this in the results of the model of using all 19 variables caused me to test to see what results only the important five variables would show. Therefore, I used two different sets of variables on all four different model types.

The results of these eight models are shown in the two tables on the next page. These tables show the scores from the models and for the three recent years that were predicted. For the three models other than the decision tree, the models were trained on 60% of the tournament games from 2003 to 2013. Then each model was run using 100 different random seeds from which each model was used to predict each matchup from 2014 – 2016. These are the three most recent years of data I had available. For the decision tree, since the same tree will be created if run twice on the same dataset (even with random seeds), I randomly sampled another 60% from the same subset 60% used for the other models, 100 times. This created 100 decision tree models and subsequently produced 100 predicted and scored brackets for the same three years as was done with the other models. The value that is shown for each model and year above in the table is the average of the 100 predicted and scored brackets for the given model, year, and group of variables (19 or 5).

19 Variables	2014	2015	[%]	2016	Averages (SD)
Random Forest (.265)	811	1147	[.90]	1296	1085 (212)
Decision Tree (.335)	906	1026	[.79]	1093	1008 (211)
Bagging (.269)	807	915	[.63]	1311	1011 (221)
Boosting (.262)	792	984	[.73]	1315	1030 (217)
Averages (M error)	829 (126)	1018 (142)	[.78]	1254 (123)	1040 (215)

Table 1: 19 variable set of models.

5 Variables	2014	2015	[%]	2016	Averages (SD)
Random Forest (.304)	769	1219	[.95]	1313	1100 (240)
Decision Tree (.315)	935	1061	[.83]	1164	1054 (210)
Bagging (.314)	736	1222	[.95]	1294	1084 (250)
Boosting (.260)	900	916	[.63]	1263	1026 (169)
Averages (M error)	835 (122)	1104 (169)	[.87]	1259 (106)	1058 (221)

Table 2: 5 variable set of models.

M error: Misclassification Error next to model names, [%]: Percentile of 2015 ESPN scores, (SD): Standard Deviation for either entire year, specific model, or entire variable set.

Looking at the model averages in the right column of each table shows that random forest was the best in each of the variable sets. Interestingly the standard deviation was similar among years and between models. It seems fairly clear that the 5-variable set did better almost across the board in every average except only one in the 19-variable table. Looking at the results for the year 2015 is the best explanation of how these models will do on average compared to the real world results of the ESPN entries. This bracket data was only available for 2015, but it would have been possible to create a similar distribution for every year since around 2010.

These scores above are comparable from year to year and between models, but having the context of what the distribution from three million ESPN entries helps a lot. In figure 3, the blue dotted line is the average score for the random forest model in year 2015 of the 19-variable group table. This average score would place it in the 90th percentile of scores for 2015, but looking at the 5-variable random forest average score on that distribution puts it in 95th percentile. These scores are not perfect brackets by any means. If you were in a bracket pool with friends and the objective was to get the highest score, there is a very good chance you would win all your friends' money for that year.

One other thing that is noticeable right away is the clear difference in averages among years of almost 200-point staggers between the three years. This is actually explained fairly well by understanding the outcome of Menzel's article. The main results of the article were placing the brackets from 1985 to the current year in order of likelihood. It's called March Madness for the sole reason of upsets which go right along with making the brackets difficult to predict. The table to the right are the results from Menzel's work for the years in the 2000's [6]. It is clear to see that 2014 was a very difficult year to predict since it was the 4th most unlikely outcome of a bracket in the 2000's, which explains the dip in scores for that year. This chart having 2016 more unlikely than 2015 does not correspond the same as the scores from the models.

1st	2013
2nd	2010
3rd	2012
4th	2014
5th	2011
6th	2006
7th	2000
8th	2001
9th	2016
10th	2002
11th	2005
12th	2015
13th	2004
14th	2003
15th	2008
16th	2009
17th	2007

Menzel's method was to multiply the probability of each matchup outcome for the entire bracket based on historical outcomes of specific seeding matchups. This does not though give more importance to later games in the bracket like the scoring method does on ESPN. Looking specifically at 2015, seven seeded Michigan St. made it to the final four round. This is a

very unlikely occurrence and most of the models did not correctly predict this. Many more upsets occurred in 2016 than 2015 which lead to a higher unlikelihood rank from Menzel's findings, but this does not directly mean 2016 will be scored lower because of the weighting of scoring. Knowing and understanding that the seasons vary so much leads to the assumption that the model results for 2014 would also be below average for scoring that year. I am confident that the scores for 2014 would be in similar percentiles as the 2015 results if the data was available.

Overall I am pleased with the results from the models. Being in the 90th and 95th percentile of any year will win just about every pool you choose to enter. I find it interesting that random forest was clearly better than the other two ensemble models. Going into this project I thought random forest would do the best of sorting through all the variables for which it did with the 19-variable set, but finding it performed the best also with just the 5-variable set was surprising. The model will be improved and tweaked further before using it in this coming March. The code used for data cleaning and models will be uploaded to RPubs shortly for your own use.

Bibliography

1. Kanat Bekt, “Espn-brackets.” Github, US: 2015. https://github.com/Bekt/espn-brackets (accessed April 11, 2017).

    Kanat Bekt scraped the data for figure 3 for the three million bracket entries on ESPN. His data was scored in the same method that ESPN uses and then plotted in the distribution for the figure.

2. Adit Deshpande, “March-Madness-2017.” Github, US: 2017. https://github.com/adeshpande3/March-Madness-2017 (accessed August 29, 2017).

    Adit Deshpande scraped the ranked data from “Sports Reference”[7] that was used in the model. His data was cleaned into the same format as the data from the “Kaggle”[

3. T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning*. New York: Springer Media, LLC, 2009.

    This book was used mostly for learning the different error terms for classification trees and the details of random forests. Two of these editors also helped with [4]. This book is a more detailed version of [4].

4. G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning.* New York: Springer Media, LLC, 2013.

    This book was used mostly to learn the methods of decision trees. This book is the introductory version of [3] and has more of an emphasis on coding the examples in R.

5. Kenneth Massey, “March Machine Learning Mania 2017.” Kaggle, US: 2017. https://www.kaggle.com/c/march-machine-learning-mania-2017/data (accessed April 11, 2017).

This website helped supply all the regular season data that was used in the models. The website gave notice that Kenneth Massey was the supplier of the data for the website.

6. Matthew Menzel, “Modeling March Madness.” *Math Horizons* 24 (2017), 5-7.

This was the article that got me interesting in pursuing this topic. Menzel’s results of the ranking of unlikeliness of the recent tournaments was used in the last section of the paper.

7. Sports Reference LLC, “School Ratings.” College Basketball at Sports-Reference.com (2017). http://www.sports-reference.com/cbb (accessed April 11, 2017).

This website was the source of the data used for most of the variables in the 5-variable models. Would have been an excellent source to find regular season data as well. The source of Massey’s data in [5] could not be confirmed, but this very well could have been the source of his data aslo.

Quinn Fargen is from Elkton, South Dakota and is currently an undergraduate student at South Dakota State University. He wrote this paper as part of his senior capstone class. In the Fall of 2017 he will complete his B.S. in Mathematics with minors in Accounting and Statistics. He will continue his education at SDSU in the Spring of 2018 in graduate school to pursue a M.S. in Data Science.

Quinn hopes to attain a data scientist position somewhere in the Midwest. When not working or doing school work, he loves to watch sports and go running. He is a Notre Dame enthusiast and perpetually guarantees they will win March Madness every year.