Introduction

Every March, millions of people submit their own bracket for the NCAA men's basketball tournament trying to predict the ultimate winner in order to win cash prizes or simply earn bragging rights among their coworkers and friends. This is usually not a very easy task. In fact, no one has ever correctly predicted a full bracket correctly before. More than 95% of all brackets have at least one incorrect prediction before the end of the first round of 32 games. The first round perhaps shows the most variation and wild cards of any other round of the tournament. Though givens like the fact that only one 16 seed have ever beaten their 1 seed counterpart in the history of the tournament, things can still get messy. Due to variation in strategy, bad games, injury and just wild luck, this can be frustrating to predict. This is where I would like to focus. Can any given game in the first round be predicted to a reasonable degree? Since the seed rankings of each teams do a pretty good job as a predictor already, with predicting 74% of the games correctly in the dataset used, the goal would be to build a model that can beat this prediction rate. Though, if the model was specifically good at predicting major upsets, a lower prediction rate could be tolerated.

I believe that such a goal can be accomplished by using certain statistics from the regular season to build a model that can detect the subtle things that makes an underdog able to beat a better team. These statistics come from Sports-Reference and are updated per season. Some of the things contained in the dataset are field goals, three-point percentage, number of turnovers and number of rebounds. These are just a small example of the 23 different single variables used in the model building which are detailed in the appendix. Since there are two teams in each game, there would be tons of factors just for one game. To reduce the number of dimensions, I took the high seed iterations of the variables and subtracted it by the low seed iterations of the variables. This has the same effect of using them individually while cutting the number of dimensions in half. There are also some variable interactions and variable transformation that could be useful as predictors. The source of these are outlined in the methods section, which brings the total amount of predictors to 63. This is all used to predict a binary response, the *winner* variable. 1 denotes that the high seed has won that match and 0 denotes that the low seed has won. So, the model will be using the difference in high and low seed statistics to predict the final, binary outcome. There are 621 observations in the data. 32 games per year from 1991 to 2018, besides the two observations removed due to missingness. 1991 was chosen for the start year since many variables that are measured now weren't back then. Even so, there were a lot more teams missing just one or two variables a season that would have to have been dealt with. Also, rules have changed since the beginning of the tournament began and players and teams have seemingly been getting better over the years. So, 1991 seemed like a good place to start the modern era of college basketball. In order to find the best model, multiple models using multiple methods should be used. For this project, logistic regression, K-nearest neighbors, and Support Vector Machine classifiers were used to make multiple models. The 2019 season was excluded from the training data in order to give the models a real application in which to predict the winners. This will be a fun way to end this project and to test the accuracy in a real tournament setting.

Methods

The final dataset was compiled using two smaller datasets. First, the regular season data for all the teams was scraped from Sports-Reference using a Python script. Since each year is in a separate dataset, each dataset from 1991 to 2018 was combined into one larger set with new variable *year* to distinguish team stats from year to year. This dataset will be called Complete Season Data. The other dataset that's combined into the final one is the NCAA Tournament Results by Michael Roy. This set was filtered to only contain *year* values from 1991 to 2018 and *round* values to only be round 1. Since the names of the school differ slightly between the two different datasets, they were adjusted to match each other which was a grueling process. The *team* and *team_2* variables in the Tournament dataset were renamed *high_seed* and *low_seed* since the first team listed was always the high seed and the second the low seed. Complete Season Data and NCAA Tournament Results were combined by *year* and *high_seed/low_seed* where the stats for high seed was given a prefix H and low seed stats were given the prefix L. To reduce the number of features, the difference of each variable, high minus low, was taken and stored in variables with the prefix D. The individual stats were then discarded. The response variable *winner* was added based on *DScore* and assigned 1 or 0 where 1 was the high seed winning the game and 0 was the low seed winning. To reduce complication, observations missing certain statistics were removed since there were only two for the entire dataset. Some interactions and variable manipulation variables were added based on correlation plots and partial residual plots. The final dataset used for the models is now called Complete Tournament Diff Data. For each of the models outlined in the methods section, a testing dataset and a training dataset was randomly assigned based on the seed (123) with 75% of the data going into the training set and the other 25% going into the testing set. For the final "real application" test, the 2019 tournament data was taken from the same sources but filtered to contain only round 1 data for the year 2019. The same variable interactions and modifications were added as needed in order to make the testing accurate.

First for the logistic regression, a full model was built using *glm* with all variables and observations in the training dataset. After the model was built, the summary was given, and a confusion matrix was built using the model on the testing data. Accuracy of the model was calculated by adding the true positive and false positive numbers divided by the total number of observations. A note was taken of the AIC and accuracy of this model. To refine the model, all non-significant predictors were taken out of the model where the significance was defined by the t-test section of the coefficients in the summary of the model. The same steps as above were taken until all factors were significant and taking any away would hurt the model's accuracy. To further refine the selected model, a variable amount of prediction thresholds used in classifying the confusion matrix which directly affects the accuracy were used (see appendix for example of this if confused). A loop was used to optimize this value by increasing the threshold by .001, hitting each value between 0 and 1. The highest accuracy out of these will be used in the final model. The variables in this model will be used as a basis for the other classifications.

The K-nearest neighbors, or K-nn, model was built using the *knn* function in the *class* library. Before any model building, the data had to be normalized. A base model was built using all variables contained in Complete Tournament Diff Data after it had been normalized. This was

optimized by various values of knots, k, which was implemented using a loop that returned the accuracy value after each value of k was run in the classifier. The values of k ranged from 1 to 100. A note of the k value was taken of the highest accuracy model. Then, another model was built using the same k optimization process, but this model only contained the variables that the optimized logistic model contained. Whichever of these models that had the highest accuracy will be used in the final application testing.

The Support Vector Machine, or SVM, model was built with the *svm* function in the *e1071* library. Just like K-nn, the data must be modified before building a model. The scale function was used to scale the data in the Complete Tournament Diff Data dataset. Two models were built, one with all variables in the dataset and one with just the variables used in the logistic model from earlier. Optimization in the SVM includes changing the kernel, which in R has four different options, and messing around with the different settings specific to each kernel. It was difficult to find a way to automate this optimization, so I just played around with the different options for a while until something was found that was up to par with the previous accuracies. The best of the two trained models will also be used in the final accuracy test.

Results

Logistic Model

For the logistic regression, three main models were made until the last one was determined the best. The first model was the full model using all variables in the dataset. The significant features were then included on the next model then the significant ones from the next model were included in the last one. The other two models are defined as:

MODEL 1: $$winner = \beta_0 + DGames + DWins + SeedDiffSq + WinsxSOS + GamesSq + STLxSD + ThreePxTRB + FGxAST + FGxThreeP + DTOV + DSOS + DSRS$$

$MODEL\ 2$: $$winner = \beta_0 + DGames + DWins + WinsxSOS + GamesSq + FGxAST + DTOV + DSRS$$

Where β_0 is the intercept and the variable names are the variable times their respective $\beta_i's$. During the last model, taking away features reduced the accuracy and/or increased the AIC, both of which is how I determined how "good" a model was. The following table outlines the models, accuracy, AIC and the probability threshold used when predicting the testing probabilities.

Model	Accuracy	AIC	Prob Threshold
Full Model	.87019	396.58	.499
Model 1	.87019	391.84	.549
Model 2	.87500	370.06	.558

Model 2 has a slightly better accuracy than the others but a significantly better AIC. The amount of correctly predicted upset games (the true positive in the confusion matrix) on the test data only

differed by plus one on Model 2 than the others. Given these results, Model 2 is clearly better at predicting the testing data and will be used in the final tournament challenge.

K-nn Model

There were two K-nn models made, one based on the full set of training data and the other based on the variables from the best logistic model. This was a pretty simple classifier with optimization for the k value which was automized. The following table outlines the two models with their k value, accuracy and the correct number of upsets rate (true negative rate)

Model	K	Accuracy	Correct Upset Rate
Full Model	11	. 85577	.53448
Reduced Model	15	. 85577	.62068

The accuracy for both of these models are the same but the difference in correct upsets is significant enough to take note of. Those rates are not something magical but its more than half the time which is better than guessing blind.

SVM Model

Again, two models were made with the SVM method if classification. Optimizing these models was a bit more complex and I was not able to automize it except for the probability threshold. After tinkering a while with different kernels and options to go with them, I found that just the linear kernel with no other modifications worked surprisingly well. The following table shows the accuracy, probability threshold and correct upset rate for each SVM model.

Model	Accuracy	Prob Threshold	Correct Upset Rate
Full Model	.88461	.66	.85417
Reduced Model	.87019	-.109	.84444

Surprisingly, the full model did better than the reduced model here. The accuracy is very close to 90% which is more than acceptable for this project. The correct upset rate is high as well which makes this the best model we have seen.

Final Testing

For the practical testing, model 2 from the logistic models, the reduced K-nn model and the full SVM model was tested against the real tournament matchups from the 2019 tournament. The “high seed selection” method has also been added for comparison. In the table below, the accuracy ratios and percentages are shown for each model.

Model	Correct Predictions	Incorrect Predictions	Accuracy	Correct Upset Rate
High Seed	20	12	.625	0
Logistic Model 2	23	9	.71875	.8
K-nn Reduced	22	10	.6875	.41667
SVM Full	22	10	.6875	.6

Each model performed better than just picking the high seed for each game. The logistic model performed the best in not only accuracy, but in the correctly predicted upset rate. This is surprising because the SVM model performed so much better than any of them but now it is on par with the worst model.

Discussion

With the training data, the SVM proved to be a very accurate predictor of round 1 winners. With an accuracy rating of 88% and an upset accuracy rating of 85%, this was the clearly the best model. However, when it came to an actual tournament that the models have never seen, the SVM model did not come out on top. While none of the models proved outstanding in the final test, they all did better than just picking the high seed, slightly. The logistic regression model managed to come out on top on this test and did outstanding in predicting upsets with an 80% prediction rate there. The K-nn model kind of fell by the wayside even though the accuracy was still great with the training data. Overall, I am surprised but pleased that the models did this well. They all did better than the high seed predictor which I was skeptical out. I had figured that since the high seed seemed to work a lot and was a variable in the dataset, it would be the main feature and have a similar accuracy to just picking the high seed. However, it wasn't even used in the logistic model nor the K-nn model and the logistic model did well in the final test. This project was a success since I managed to find a fairly successful model and did beat out the high seed predictor.

Now to talk about why the models didn't perform so well in the final test and what could be done to fix it. Since the models were tested with a larger dataset, there is more wiggle room when it comes to errors in predictions. The test datasets had 156 observations while the real tournament data had 32 observations. Another theory is that since K-nn and SVM are a little more complex in fitting training data, they may have overfit some of it. Since logistic regression is a little more straightforward, it may have left a little more room to allow for variation between variables. One of the biggest issues was my lack of knowledge with variable selection. Since I have a little bit of knowledge about how that goes on with linear regression, it translates almost the same to logistic, I was better able to get the right variables for the logistic model. I used the same on for the K-nn and SVM reduced models and they may have not been a great fit. The best SVM model, I believe, chose its own variables from the entire dataset. I'm not entirely sure how that works just yet in R, but the SVM reduced model was different so choosing specific features to include had to change it. That is something I need to work on in the future.