Predicting The Outcome of The 2020 NCAA Men's Basketball Tournament Using Various Statistical Models

Nick Corona, Donald Turner, Lacee Vavra, Kallen Wagner

May 2020

1 Introduction

The purpose of this project is to understand how predicting the NCAA basketball tournament works. During the second or third week of March, the NCAA would have released a bracket with 64 men's basketball teams that have made it into the tournament. This is known as the Round of 64. Thirty-two basketball games will be played between the sixty-four teams. The winners of this first round of 64 are then advanced to the Round of 32. It will then be reduced to 16 teams then 8, 4, and ultimately the championship game with with the final 2 competing teams. Our project is aimed to predict the teams that are competing in each round of the NCAA tournament. Our first goal is to predict which basketball teams will make it into the tournament (Round of 64). Then, we'll attempt to predict the following teams that will remain in the Round of 32, Sweet Sixteen, and, in the end, the championship game. Unfortunately, the NCAA tournament was canceled this year due to the COVID-19 pandemic. In spite of this setback, we carried out our analysis, beginning instead with the 64 teams predicted by ESPN to make the playoffs.

2 Analysis

2.1 Data Description

The data selected for this project was pulled from a user-created dataset through Kaggle. This data contains team statistics of NCAA men's basketball teams from the 2014-2015 to the 2019-2020 season (6 total seasons). These team statistics include: Team, Conference, Games Played, Wins, Seed, Adjusted Offensive Efficiency, Adjusted Defensive Efficiency, BARTHAG, Effective Field Goal Percentage, Turnover Rate, Rebounding Rate, Free Throw Rate, 2-point Shooting Percentage, 3-point Shooting Percentage, Adjusted Tempo, Wins Above Bubble, Postseason finish, Seed, and Year. For the duration of this project, our response variable is the POSTSEASON variable; the team's performance in the NCAA tournament. This variable is a ranked variable meaning that the unique levels are in order from the weakest performance (lowest) to the best performance (highest). The observations of this variable include 0 (didn't make tournament), R64 (Round of 64), R32 (Round of 32), S16 (Sweet 16), E8 (Elite 8), F4 (Final Four), 2ND (runner up), and Champions.

2.2 Data Selection

The first step in our data selection was determining which variables in the original data were valuable in predicting postseason performance or produced bias. We went through the data and removed observations that directly interfered with the output, such as number of games played (G) because the more games a team played means they made it further in the tournament. Another example was the removal of the Wins (W) variable. We knew that it was biased in accordance to tournament performance. Intuitively, the more games a basketball team wins, the greater the team will perform in the NCAA tournament. We started out with 16 variables and, using prior knowledge, removed Wins (W), Games Played (G), Seed, and Wins Above the Bubble (WAB). A correlation plot (Fig. 1) was also created to see how each variable interacted with each other.

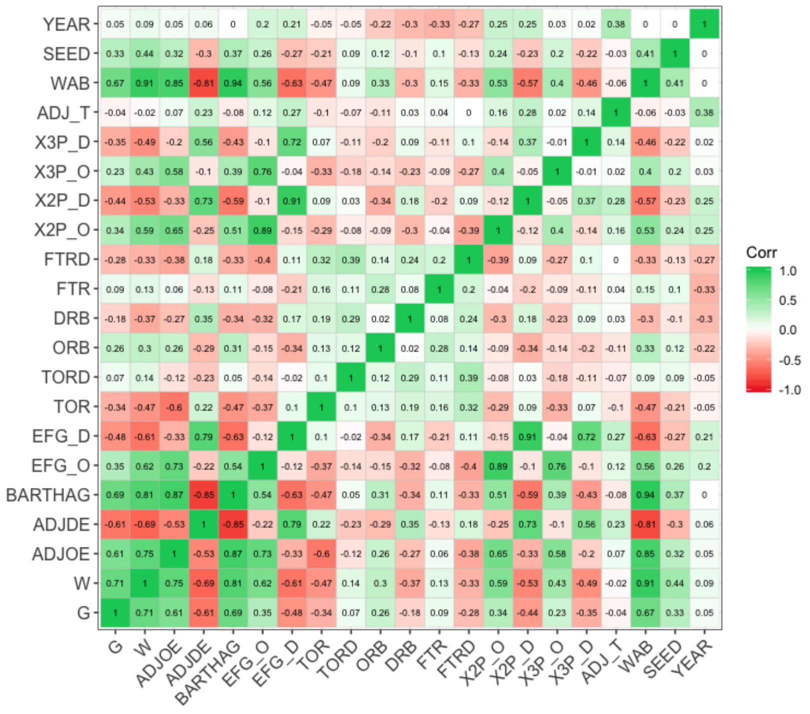


Figure 1: Correlation Plot

Most of the highly correlated variables were ones that we decided to remove based on knowledge of the data and college basketball. One relationship that was noticed in the correlation plot (Figure 1) was between the BARTHAG and ADJOE (Adjusted Offensive Efficiency) and ADJDE (Adjusted Defensive Efficiency) variables. Upon further investigation, BARTHAG is a function of ADJOE and ADJDE. Because of this, the BARTHAG variable was redundant for our model and ultimately removed from consideration.

3 Statistical Analysis

After the removal and selection of our data's explanatory variables, the new data was split into training and testing subsets in order to calculate and test different statistical models. Note that the observations used for these models excluded the teams that did not attend the NCAA Division I tournament (0).

3.1 Decision Tree

The data corresponding to years 2015-2018 were used for training, and the data from 2019 was used for testing. This data filtering was implemented in order to have more terminal nodes in the tree that contained higher ranked NCAA tournament rounds. Then, the training data was ran in R to output a decision tree. Most of the splits in the resulting tree were based on ADJDE (Adjusted Defensive Efficiency) and ADJOE (Adjusted Offensive Efficiency). The first decision tree created is displayed below (Fig. 2):

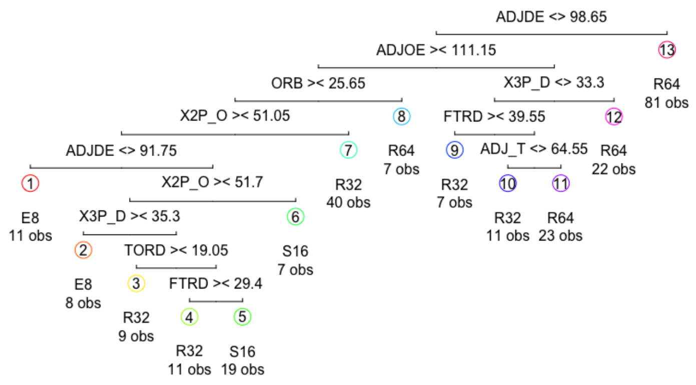


Figure 2: Decision Tree

A confusion matrix was then also performed to show the accuracy of our decision tree results (Fig. 3). The results are below:

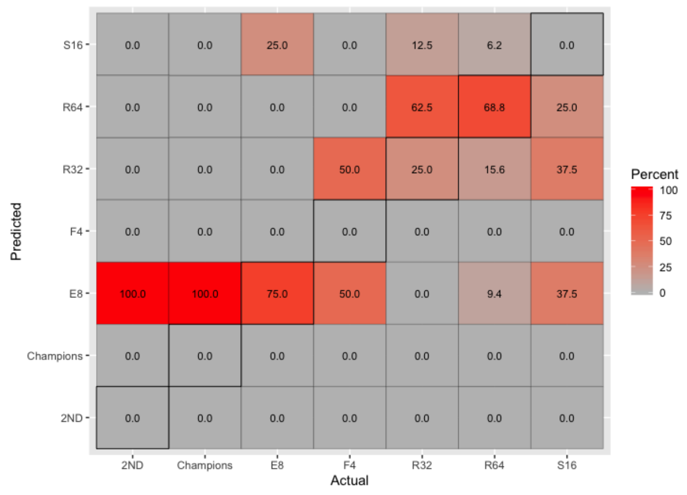
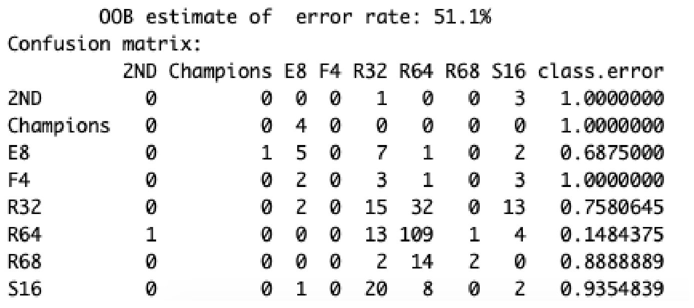


Figure 3: Confusion Matrix

The decision tree model that was created gave up the great accuracy when predicting the Rounds of 64, 32, Sweet 16, and the Elite 8. However the model failed to predict any team in the Final 4, Final 2, and the eventual championship.This can be due to numerous reasons such as failing to predict upset victories that could have eliminated the eventual champion earlier on.

3.2 Random Forest

The second statistical model that we attempted was a Random Forest model. The data used in this model was identical to previous models and only contained the basketball teams that did appear in the NCAA tournament. The highest reported accuracy using the Random Forest model was approximately 52.9 percent. A generated confusion matrix of this model is shown below (Fig. 4):



```
        OOB estimate of  error rate: 51.1%
Confusion matrix:
          2ND Champions E8 F4 R32 R64 R68 S16 class.error
2ND         0         0  0  0   1   0   0   3   1.0000000
Champions   0         0  4  0   0   0   0   0   1.0000000
E8          0         1  5  0   7   1   0   2   0.6875000
F4          0         0  2  0   3   1   0   3   1.0000000
R32         0         0  2  0  15  32   0  13   0.7580645
R64         1         0  0  0  13 109   1   4   0.1484375
R68         0         0  0  0   2  14   2   0   0.8888889
S16         0         0  1  0  20   8   0   2   0.9354839
```

Figure 4: Confusion Matrix for Random Forest Model

According to the confusion matrix, the computed Random Forest model had an extremely high classification error rate for the later rounds. The only acceptable classification error rate was for the first round (R64), however this is not significant considering these are the initial teams. In conclusion, our model had a very high out-of-the-bag error rate (50.7 percent) and overall failed in predicting teams in the higher rounds of the NCAA tournament. Next, we continued to explore different models in order to address this problem.

3.3 Ordinal Logistic Regression

Finally, we computed an ordinal logistic regression model in an attempt to address the ranking and order of our categorical response variable (POSTSEASON). Since the number of teams decrease as the tournament rounds further, our response variable is assumed to be ranked and to have an order. For instance, a team that made it to only the Round of 32 (second round) is ranked lower in their POSTSEASON variable value than a team that made it to the Elite Eight (fourth round). We used the polr function in the MASS package supplied by R to generate the logistic regression model. Variable coefficients, and their respective p-values were included in the model output. A confusion matrix was also generated to visualize the accuracy of the model (Fig. 5):

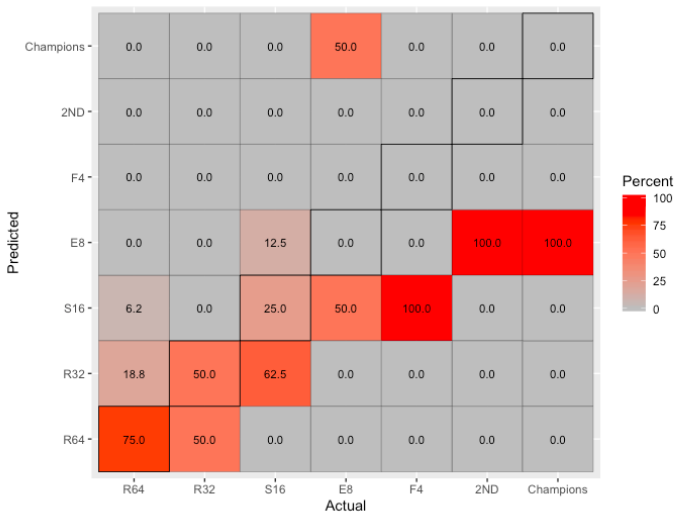


Figure 5: Ordinal Logistic Regression Confusion Matrix

While the accuracy of the Round of 64, 32, and Sweet 16 look promising, the accuracy in the model extremely lessens for the further rounds. Following the Sweet 16, the ordinal logistic model rarely identifies a team's final tournament round correctly. Overall, this model is an improvement compared to the previous Random Forest model. However, there are still flaws farther into the tournament.

3.4 Model Selection

After surveying the three models discussed previously, we collectively decided to use the decision tree model to apply to the 2020 NCAA basketball tournament. This model was chosen because it had the greatest success with the higher rounds of the tournament. Although, the accuracy was still low for the later rounds, it was significantly better than the Random Forest or ordinal logistic regression model.
Now that our model has been formally selected, our next step is to apply our decision tree model (Fig. 2) to the 2020 NCAA Division I basketball tournament and predict a winner.

4 Application to 2020 NCAA Season

4.1 Disclaimer

Due to the COVID-19 virus, the actual 2020 NCAA Basketball Tournament was cancelled. Because of this, we were not able to obtain the data needed for the 2020 Tournament. During application, we will use the estimates provided by ESPN Bracketology. We will use their estimate of the teams in the first round (Round of 64), then use our determined model to predict further rounds.

4.2 Round of 64

In order to have a round of 64 to base our predictions from we used the projected 64 teams to make the tournament from ESPN Bracketology. ESPN predicted these teams by using a combination of conference winners and ESPN BPI, or Basketball Power Index. There are 32 conferences in NCAA Men's Division 1 basketball and all 32 conference winners automatically get a spot in the tournament. The remaining 32 teams were predicted by ESPN using their BPI which ranks all teams in men's division 1 basketball. The top 32 teams who were not automatically selected were placed into the projected Round of 64.

round 1 clean.png round 1 clean.png

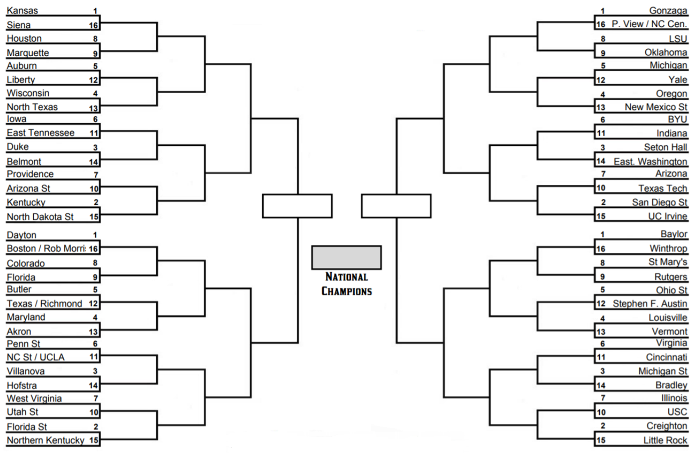


Figure 6: Projected bracket round 1 by ESPN Bracketology

4.3 Round of 32

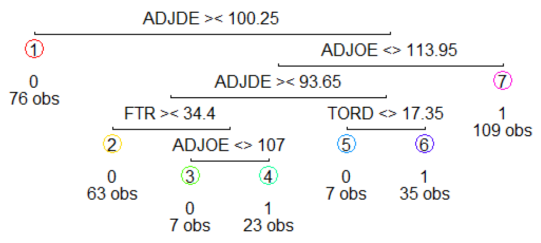


Figure 7: Decision tree used to predict the round of 32.

Using this decision tree we will predict a bracket of the round of 32.

clean.png clean.png

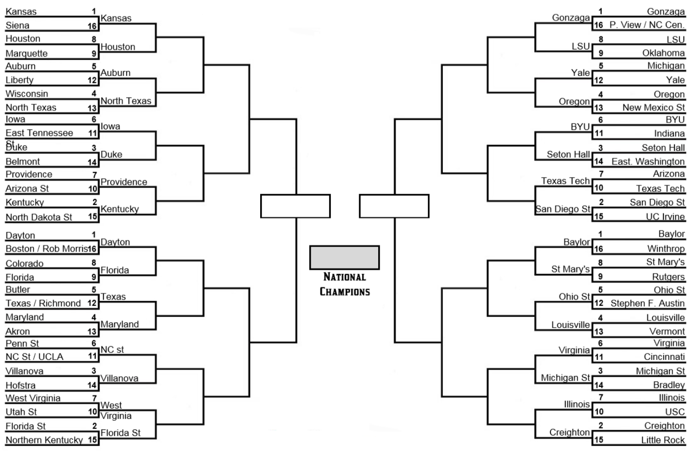


Figure 8: Bracket with predicted round of 32.

4.4 Sweet 16

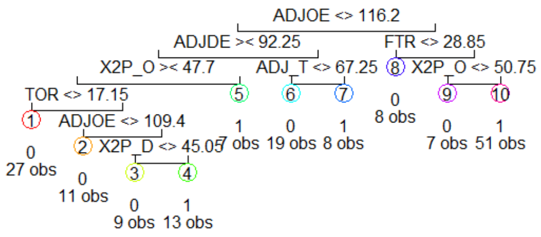


Figure 9: Decision tree used to predict the sweet 16.

clean.png clean.png

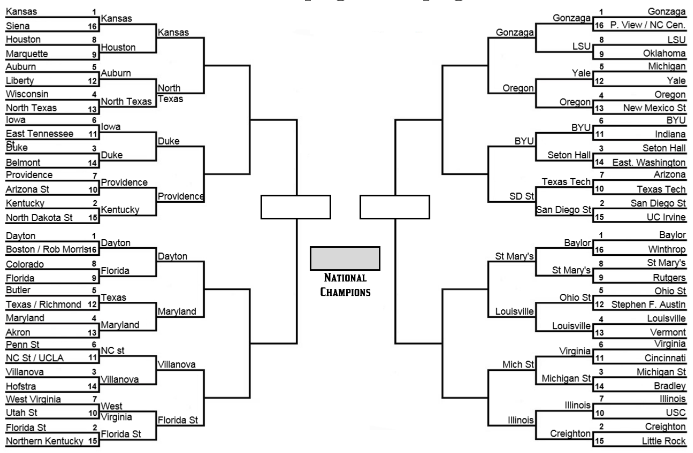


Figure 10: Bracket with predicted Sweet 16.

4.5 Elite 8

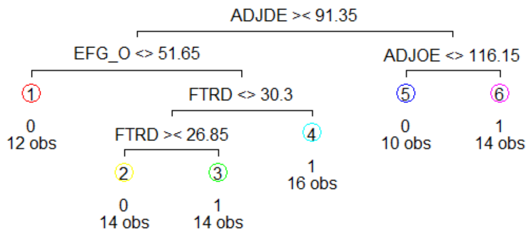


Figure 11: Decision tree used to predict the Elite 8.

clean.png clean.png

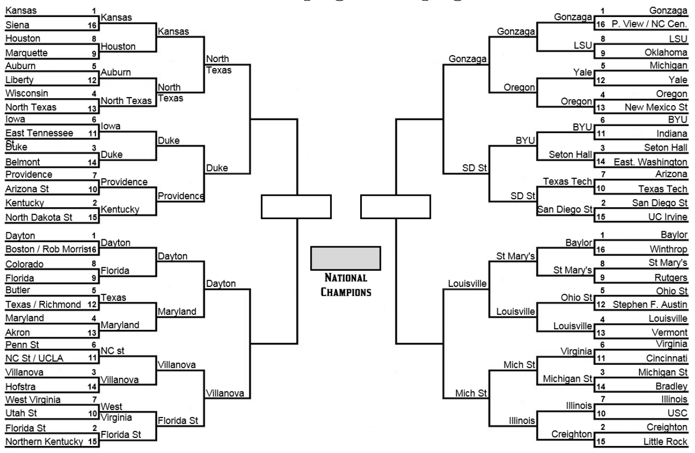


Figure 12: Bracket with predicted Elite 8.

4.6 Final Four

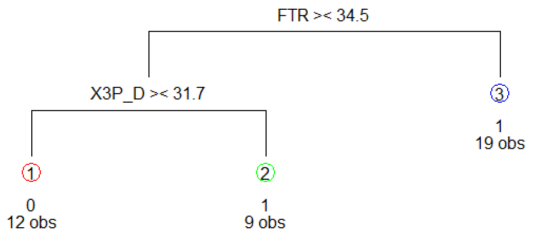


Figure 13: Decision tree used to predict the Final Four.

clean.png clean.png

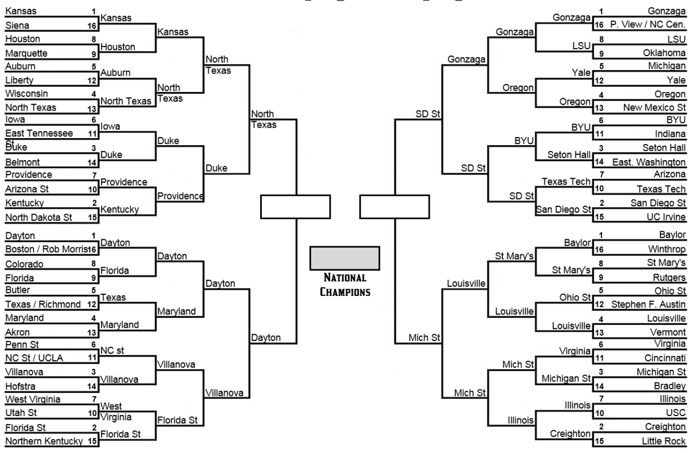


Figure 14: Bracket with predicted Final Four.

4.7 Second Place

The higher rounds in the tournament have significantly less data than the lower rounds. Because of this, our decision trees and predictions in the higher rounds are much less accurate as you can see by the complexity of the decision trees the higher the rounds go. For this reason, the Final Four and Championship games are highly variable and unpredictable. Nonetheless, we have included a bracket for these higher rounds however they are much less significant than the lower rounds.

clean.png clean.png

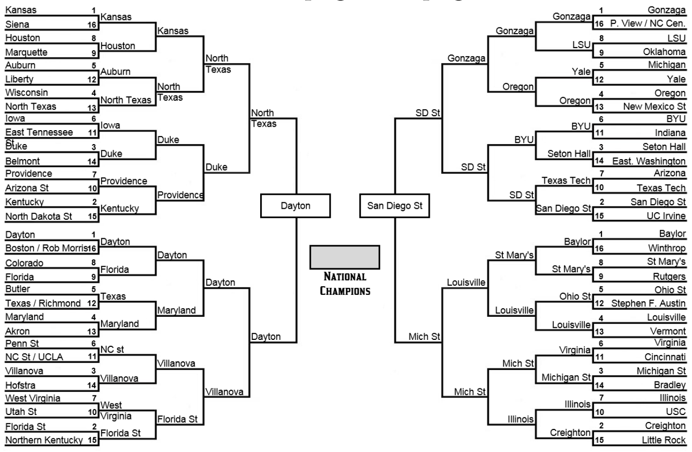


Figure 15: Bracket with predicted Championship game participants.

4.8 Champions

clean.png clean.png

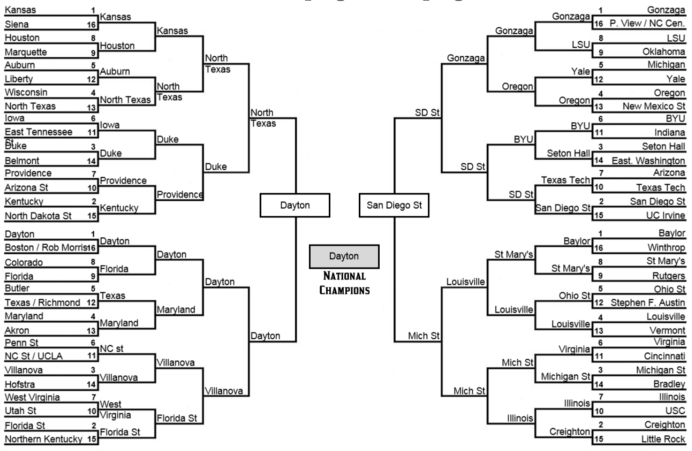


Figure 16: Completed bracket with our predicted champions, the Dayton Flyers.

5 Conclusion

Based on the models that we used; the decision tree produced the best model for predicting the outcome of the NCAA tournament. The decision tree produced a 75 percent accuracy when predicting the Elite 8. What is also notable is that when predicting the Elite 8, the champion and the second place team made it into the round 100 percent of the time. However, past the Elite 8, the model could not predict any of the correct teams. This could be explained by numerous reasons, such as: models unable to predict upsets. The other models produced very low accuracies, so they were unable to assist in predicting the bracket. As stated earlier, due to COVID-19, the data was incomplete due to the season being cancelled early. With the information predicted, we used a starting bracket from ESPN and used what data was available to predict the winner. Our model predicted that the NCAA champion would have been Dayton.

The next steps of the project would be to research models that allow for ranked choices and correction based on the output. Unfortunately, due to March Madness being known for upsets, it will be hard to predict something based on that. For that more information needs to be pulled from games that were considered upsets to determine if it can be incorporated into the model. Another possibility would be to assign a likelihood rating to each game predicting a possible upset.

6 Appendix

```
#Random Forest
library(ISLR)
library(caret)
#library(lubridate)
library(tidyr)

setwd("/Users/laceevavra/Documents/482 Work")

#this data contains teams that did NOT make playoffs too.
teams = read.csv("teams_ROUNDS.csv")
attach(teams)
#replace NA's in playoff rounds with 0's. Matches other columns
    (PLAYOFFS)
teams[is.na(teams)] = 0
#take out teams that did NOT attend tournament (0).
playoffs = teams[!(teams$POSTSEASON == 0), ]

#remove biased variables
teams_sub = subset(playoffs, select = -c(G,W,CONF,BARTHAG, TEAM, SEED,
    WAB, S16, E8, F4,
                       x2ND, Champions, R64, R32))

#training
four_years = teams_sub[which(teams_sub$YEAR != 2019), ]
four_years1 = subset(four_years, select = -c(YEAR))
four_years1A = na.omit(four_years1)
```

```
#test
test_year = teams_sub[which(YEAR == 2019), ]
test_year = subset(test_year, select = -c(YEAR))
test_year = na.omit(test_year)

four_years1A$POSTSEASON = factor(four_years1A$POSTSEASON)
#Training/Testing Split: 75% training, 25% Test
set.seed(430)
attach(four_years)
#model training
sim_rf_mod = train(
  POSTSEASON ~ ., data = four_years1A, method = "rf",
  #method = cv for Cross Validation
  trControl = trainControl(method = "cv",number=5),
  tuneLength = 5)
```

```
#Ordinal Logistic Regression
library(ISLR)
library(tree)
library(MASS)
setwd("/Users/laceevavra/Documents/482 Work")

teams = read.csv("teams_ROUNDS.csv")
attach(teams)
#replace NA's in playoff rounds with 0's. Matches other columns
    (PLAYOFFS)
teams[is.na(teams)] = 0

#Look at POSTSEASON variable. Different levels for each round.
#Includes teams that did not appear.
teams64_subset = subset(teams, select = -c(G,W,CONF, BARTHAG, TEAM,
    YEAR, SEED, WAB, S16, E8, F4,
                                    x2ND, Champions, R64, R32))

fit = polr(POSTSEASON ~ ., data = teams64_subset, Hess = TRUE)
summary(fit)

#calculating the p-value for each variable coefficient
ctable = coef(summary(fit))
p = pnorm(abs(ctable[ , "t value"]), lower.tail = FALSE) * 2
ctable = cbind(ctable, "p value" = p)
ctable

#odds ratio
exp(coef(fit))
```

```r
# Packages
library(rpart)
library(rpart.plot)
library(maptree)
library(ggplot2)
library(ggcorrplot)

# Reading in the data
dta=read.csv("cbb_CLEAN_TeamsGroup.csv")

# Restricting data to only teams which made playoffs
dta64 = dta[which(POSTSEASON != '0' & POSTSEASON != 'R68'),]

# Correlation plot for variable selection
CORR <- cor(dta64[sapply(dta, is.numeric)])
ggcorrplot(CORR,
          type = 'full',
          lab = TRUE,
          lab_size = 2.25,
          method = 'square',
          ggtheme= theme_bw ,
          colors = c('firebrick2', 'white', 'springgreen3'))

# Splitting data into training and testing sets
dta64train = dta64[which(dta64$YEAR!=2019),]
dta6419 = dta64[which(dta64$YEAR==2019),]

# Creating decision tree model with selected variables
tree = rpart(POSTSEASON ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
               ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O + X3P_D +
               ADJ_T, data=dta64train, method='class')
draw.tree(tree64)

# Making predictions and drawing confusion matrix
# (confusion matrix code taken from
    https://ragrawal.wordpress.com/tag/heatmap/)
pred = predict(tree, newdata=dta6419, type='class')

actual = as.data.frame(table(dta6419$POSTSEASON))
names(actual) = c("Actual","ActualFreq")
confusion = as.data.frame(table(dta6419$POSTSEASON, rpred))
names(confusion) = c("Actual","Predicted","Freq")
confusion = merge(confusion, actual, by=c('Actual'))
confusion$Percent = confusion$Freq/confusion$ActualFreq*100
tile <- ggplot() +
  geom_tile(aes(x=Actual, y=Predicted,fill=Percent),data=confusion,
      color="black",size=0.1) +
  labs(x="Actual",y="Predicted")
```

```
tile = tile +
  geom_text(aes(x=Actual,y=Predicted, label=sprintf("%.1f",
       Percent)),data=confusion, size=3, colour="black") +
  scale_fill_gradient(low="grey",high="red")
tile = tile +
  geom_tile(aes(x=Actual,y=Predicted),
            data=subset(confusion,
                 as.character(Actual)==as.character(Predicted)),
            color="black",size=0.3, fill="black", alpha=0)
tile
```

Code used to create bracket prediction

```
cbb_clean <- read.csv("cbb_clean.csv")
bracket_teams <- read.csv("cbb2020_bracket_teams.csv")
library(rpart)
tree64f = rpart(POSTSEASON ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                 ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O + X3P_D +
                 ADJ_T, data=cbb_clean, method='class')
data64train = cbb_clean[which(cbb_clean$YEAR!=2019),]
data64 = cbb_clean[which(cbb_clean$POSTSEASON != '0' &
     cbb_clean$POSTSEASON != 'R68'),]
library(maptree)
draw.tree(tree64f)
predict(tree64f, bracket_teams, type = "class")
##########################################################################################
#Model to predict who makes round 32, given round of 64
round64 <- read.csv("round64.csv")
round64$R32[is.na(round64$R32)] <- 0
predictR32 <- rpart(R32 ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                    ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                        X3P_D +
                    ADJ_T, data=round64, method='class')
R32_model <- predict(predictR32, round64, type = 'class')
table(round64$R32, R32_model)
##########################################################################################
#Model to predict who makes sweet 16, given round 32
round32 <- read.csv("round32.csv")
round32$S16[is.na(round32$S16)] <- 0
predictS16 <- rpart(S16 ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                    ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                        X3P_D +
                    ADJ_T, data=round32, method='class')
S16_model <- predict(predictS16, round32, type = 'class')
table(round32$S16, S16_model)
##########################################################################################
#Model to predict who makes elite 8, given sweet 16
sweet16 <- read.csv("sweet16.csv")
sweet16$E8[is.na(sweet16$E8)] <- 0
```

```
predictE8 <- rpart(E8 ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                    ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                        X3P_D +
                    ADJ_T, data=sweet16, method='class')
E8_model <- predict(predictE8, sweet16, type = 'class')
table(sweet16$E8, E8_model)
#########################################################################################
#Model to predict who makes final 4, given elite 8
elite8 <- read.csv("elite8.csv")
elite8$F4[is.na(elite8$F4)] <- 0
predictF4 <- rpart(F4 ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                   ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                       X3P_D +
                   ADJ_T, data=elite8, method='class')
F4_model <- predict(predictF4, elite8, type = 'class')
table(elite8$F4, F4_model)
########################################################################################
#Model to predict who makes championship, given final 4
final4 <- read.csv("final4.csv")
final4$x2ND[is.na(final4$x2ND)] <- 0
predictx2ND <- rpart(x2ND ~ ADJOE + ADJDE + EFG_O + EFG_D + TOR + TORD +
                   ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                       X3P_D +
                   ADJ_T, data=final4, method='class')
x2ND_model <- predict(predictx2ND, final4, type = 'class')
table(final4$x2ND, x2ND_model)
########################################################################################
#Model to predict champions, given appears in championship
x2ND <- read.csv("x2ND.csv")
x2ND$Champions[is.na(x2ND$Champions)] <- 0
predictChampions <- rpart(Champions ~ ADJOE + ADJDE + EFG_O + EFG_D +
    TOR + TORD +
                     ORB + DRB + FTR + FTRD + X2P_O + X2P_D + X3P_O +
                         X3P_D +
                     ADJ_T, data=x2ND, method='class')
Champions_model <- predict(predictChampions, x2ND, type = 'class')
table(x2ND$Champions, Champions_model)
########################################################################################
#Now using 2020 data to predict a bracket
#Predicting who makes round of 32, given the round of 64
predict(predictR32, bracket_teams, type = 'class')
draw.tree(predictR32)
########################################################################################
#Predicting sweet 16 based off round 32
x2020round32 <- read.csv("2020round32.csv")
predict(predictS16, x2020round32, type = 'class')
draw.tree(predictS16)
########################################################################################
#Predicting elite 8 based off sweet 16
x2020sweet16 <- read.csv("2020sweet16.csv")
```

```r
predict(predictE8, x2020sweet16, type = 'class')
testmod <- predict(predictE8, x2020sweet16, type = 'class')
library(maptree)
draw.tree(predictE8)
#######################################################################################
#Predicting final 4 based off elite 8
x2020elite8 <- read.csv("2020elite8.csv")
predict(predictF4, x2020elite8, type = 'class')
draw.tree(predictF4)
#######################################################################################
#Predicting championship game based off final 4
x2020final4 <- read.csv("2020final4.csv")
predict(predictx2ND, x2020final4, type = 'class')
draw.tree(predictx2ND)
#########################################################################################
#Predicting champions based off championship game
x2020x2ND <- read.csv("2020x2ND.csv")
predict(predictChampions, x2020x2ND, type = 'class')
draw.tree(predictChampions)
```