Using Machine Learning to Predict March Madness Tournament

Romtin Toranji (5235684) Hans Christensen (5150446) Kumman Liu (5699186)

March 11, 2020

Introduction

The goal of our project is to determine the winner of the NCAA March Madness tournament using machine learning. We can do this by first predicting the winner of each game individually. The importance behind this project is that we can make a lot of money from correctly predicting the tournament as ESPN offers 1 Million dollars to anyone who can correctly predict the entire tournament. Also, correctly predicting the tournament has a 1 in 9,223,372,036,854,775,808 or

$$(\frac{1}{2})^{63}$$

chance assuming you have a 50% chance of correctly guessing each game. We believe we can build a model to give us the best chance at correctly predicting the entire tournament. It is already known that each team does not have the same probability of winning the tournament as stronger top seeded teams play weaker bottom seeded teams to start the tournament. A lot of confounding variables such as player injuries and hot/cold streaks can change the tournament drastically as many teams rely on star players or perimeter shooting.

Our research question is how accurately can we predict the results of the NCAA March Madness Tournament.

Data

Our Dataset was retrieved from Kaggle's 2020 NCAA March Madness Machine Learning Competition. The dataset we used featured the every teams statistics for every game played during the regular season alongside the March Madness tournament along with the winning teams ID and losing teams ID where every team had a 4 digit ID corresponding to a school's name. There was no missing data in the dataset. Since the data was given in a CSV format in Excel, we used Pivot Tables to combine all of the statistics by each teams unique Team ID to create an average of all the statistics. So, for each team, we created an average of all of our quantitative variables for each teams wins and losses by using the winning teams ID and losing teams ID and retrieving their respective data from each game played. We then counted each team's wins and losses to assign each team an overall record for the regular season. However, a problem occurred when a certain team either went undefeated during the season or did not have any wins during the season. The more common case was a team going undefeated during the regular season which caused the Team ID to not have data for games lost, so we could not create losing statistics for the team during the regular season. To get around this, we created an extra row with all 0's for the undefeated team to have all of their losing statistics equal to 0. This allowed us to compute an undefeated regular season team's losing statistics (all 0's because they never lost) without affecting any true values.

Our data set is the regular season statistics for each team.

```
head(s2019)
```

```
##   Team.ID Average.of.LPF Average.of.LBlk Average.of.LStl Average.of.LTO
## 1    1101          20.83           2.500           8.000          15.33
## 2    1102          16.56           1.611           4.389          14.78
## 3    1103          17.56           2.000           4.688          11.25
## 4    1104          17.33           4.267           3.867          13.53
## 5    1105          18.37           1.556           7.333          15.26
## 6    1106          19.05           2.263           5.263          14.47
##   Average.of.LAst Average.of.LDR Average.of.LOR Average.of.LFTA Average.of.LFTM
## 1          12.000          19.33          9.000           17.00          12.167
## 2          11.278          23.06          6.833           14.22           9.556
## 3           9.625          25.12          9.312           13.88          10.875
## 4          11.733          24.13         10.400           18.47          11.867
## 5          11.630          20.81         10.259           13.37           8.333
## 6           9.895          20.32         11.842           20.05          12.737
##   Average.of.LFGA3 Average.of.LFGM3 Average.of.LFGA Average.of.LFGM
## 1            18.67            6.667           54.83           23.00
## 2            21.00            6.111           52.83           22.22
## 3            28.75            7.750           61.38           22.31
## 4            22.47            7.000           55.80           23.20
## 5            17.78            5.481           56.52           22.52
## 6            22.74            7.053           57.05           21.84
##   Average.of.NumOT Average.of.LScore Average.of.WScore Average.of.NumOT.1
## 1                0             64.83             73.52                  0
## 2                0             60.11             77.46                  0
## 3                0             63.25             73.80                  0
## 4                0             65.27             77.22                  0
## 5                0             58.85             69.80                  0
## 6                0             63.47             69.70                  0
##   Average.of.WFGM Average.of.WFGA Average.of.WFGM3 Average.of.WFGA3
## 1           25.96           55.35            7.391            18.96
## 2           28.23           58.77            8.846            24.38
## 3           25.67           56.20           10.067            27.20
## 4           26.61           57.72            7.111            19.50
## 5           25.60           56.20            6.800            19.20
## 6           23.60           55.10            7.100            20.50
##   Average.of.WFTM Average.of.WFTA Average.of.WOR Average.of.WDR Average.of.WAst
## 1           14.22           19.61          9.087          23.83           15.30
## 2           12.15           17.62          9.000          29.00           16.08
## 3           12.40           18.87          9.333          28.60           14.40
## 4           16.89           24.72         11.778          28.33           12.50
## 5           11.80           18.40          8.800          28.40           14.00
## 6           15.40           23.00         12.600          26.00            9.80
##   Average.of.WTO Average.of.WStl Average.of.WBlk Average.of.WPF Wins Losses
## 1          10.70           8.000           2.565          18.70   23      6
## 2          11.00           5.385           2.077          17.54   13     18
## 3          12.60           6.600           4.333          17.40   15     16
## 4          13.67           5.222           5.000          16.67   18     15
## 5          15.00           7.200           1.400          18.20    5     27
## 6          14.50           7.800           4.400          16.90   10     19
##   Average.of.PF Average.of.Blk Average.of.Stl Average.of.TO Average.of.Ast
## 1         19.14          2.552          8.000         11.66         14.621
## 2         16.97          1.806          4.806         13.19         13.290
## 3         17.48          3.129          5.613         11.90         11.935
## 4         16.97          4.667          4.606         13.61         12.152
```

```
## 5         18.34          1.531          7.312         15.22         12.000
## 6         18.31          3.000          6.138         14.48          9.862
##   Average.of.DR Average.of.OR Average.of.FTA Average.of.FTM Average.of.FGA3
## 1         22.90         9.069          19.07         13.793           18.90
## 2         25.55         7.742          15.65         10.645           22.42
## 3         26.81         9.323          16.29         11.613           28.00
## 4         26.42        11.152          21.88         14.606           20.85
## 5         22.00        10.031          14.16          8.875           18.00
## 6         22.28        12.103          21.07         13.655           21.97
##   Average.of.FGM3 Average.of.FGA Average.of.FGM Average.of.PPG  W.Per
## 1           7.241          55.24          25.34          71.72 0.7931
## 2           7.258          55.32          24.74          67.39 0.4194
## 3           8.871          58.87          23.94          68.35 0.4839
## 4           7.061          56.85          25.06          71.79 0.5455
## 5           5.687          56.47          23.00          60.56 0.1562
## 6           7.069          56.38          22.45          65.62 0.3448
```

Variables

Our dataset contains 14 variables and 5,463 observations. Our Qualitative variables included WLoc (The winning teams location categorized by Neutral, Home, or Away), WTeamID, LTeamID, and Season (given in the current year e.g. 2016). The quantitative statistics provided in the dataset were WFGM (Winning teams field goals made) , LFGM (losing teams field goals made) , WFGA , LFGA , along with other common basketball statistics such as Assists, Turnovers, 3 Pointers Made, 3 Pointers Attempted, Offensive and Defensive Rebounds, Steals, Blocks, and Personal Fouls. All of the quantitative statistics were counted for each game played during the Regular Season and were split by the winning and losing team of each regular season game. We created a season average by adding each teams winning statistics multiplied by their wins plus each teams losing statistics multiplied by their losses, and then divided by their total wins plus losses. For example, in calculating a teams average PPG (points per game) during the season, we took

$$\frac{(AveragePointsPerWin \times NumberofWins) + (AveragePointsPerLoss \times NumberofLosses)}{NumberofWins + NumberofLosses}$$

We used this method to compute the Per Game statistics for each numerical variable. After computing per game statistics for each team, we now have normalized variables that can be compared between each team to determine each teams success in the tournament.

Methods

Firstly, and most importantly, we must understand that player and coach turnover between NCAA basketball seasons is rather high due to the "One-and-Done" rule implemented by the NBA. This rule states that players must play at least one year of NCAA basketball before signing a rookie contract with the NBA. Since the best college players tend to move on to the NBA as soon as possible, a large percentage of the best NCAA players only play one year for their college teams. Therefore, teams change quite drastically every year and as a result, we can't use the same model from the previous years. For example, Loyola went to the final four in 2018 and did not even make the tournament in 2019. This was a result of the team losing a few of their key players because the athletes were graduating, heading to the National Basketball Association (NBA), injuries, etc.

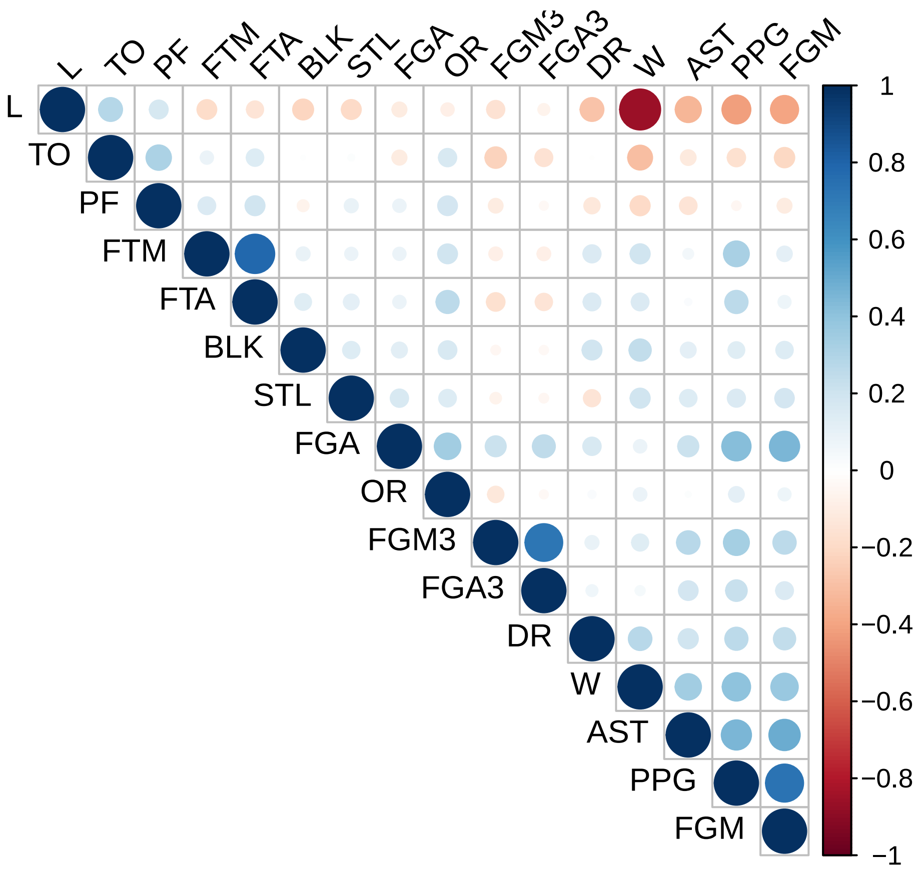
Training, Test, and Validation Sets

In our analysis, we chose not to use a validation set because our parameters are the in-season averages for each of the teams. Since we calculate a model for each individual season it would not be practical to split up our analysis of each season into a training and validation set. This would not work because we are using each NCAA regular season to predict the NCAA postseason tournament. So, we use the NCAA regular season as our training set and use our results to predict the March Madness tournament, our test set.

Data Analysis and Exploration

Initial Data Analysis: Correlation Matrix

```
corrplot(res, type = "upper", order = "hclust",
         tl.col = "black", tl.srt = 45)
```



From the correlation matrix, we see that Assists, Points Per Game, and Turnovers seem to have the highest correlation (absolute value) with wins.

Initial Data Analysis: Graphs

We made boxplots of the three aforementioned important variables vs number of wins for the 2019 regular season.

Assists

```
library(ggplot2)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

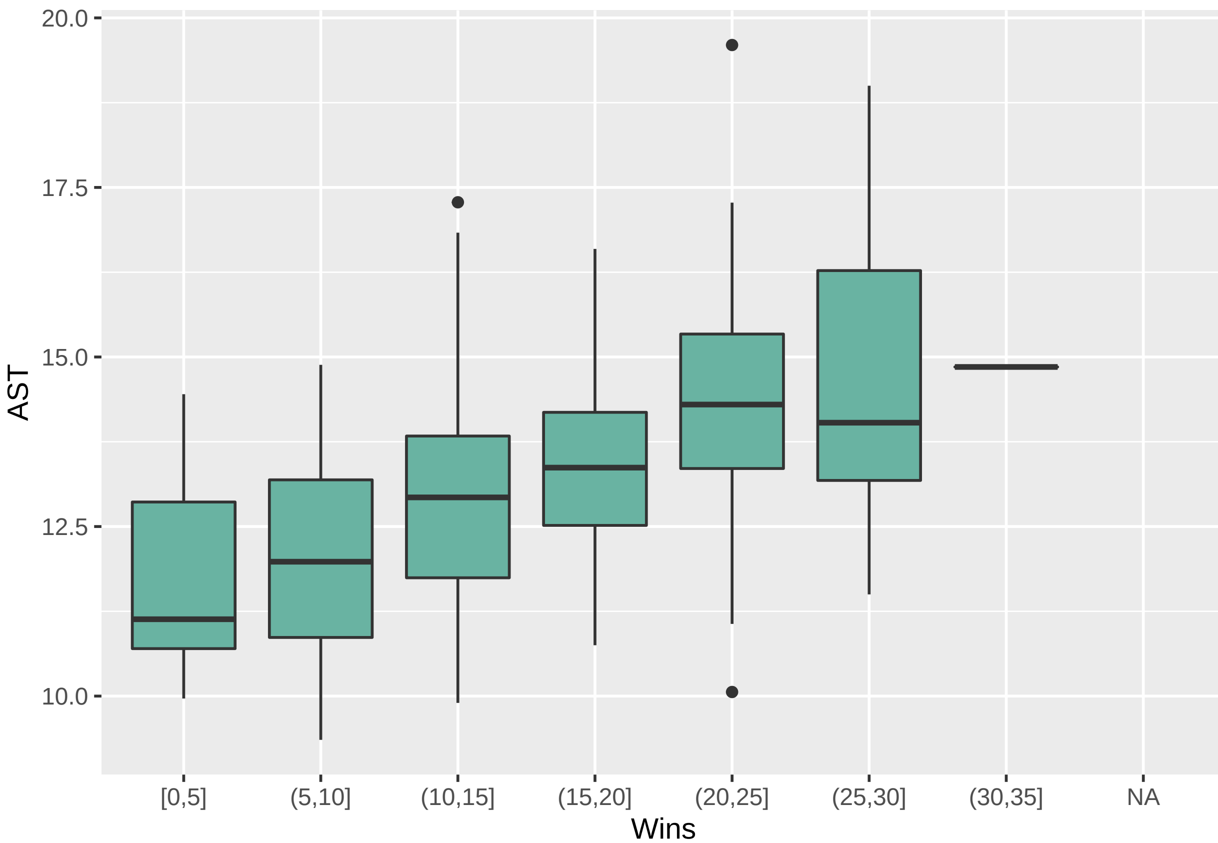
```
astGraph <- s2019_corr %>%

  # Add a new column called 'bin'
  mutate( bin=cut_width(W, width=5, boundary=0) ) %>%

  # plot
  ggplot( aes(x=bin, y=AST) ) +
    geom_boxplot(fill="#69b3a2") +
    xlab("Wins")

astGraph
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



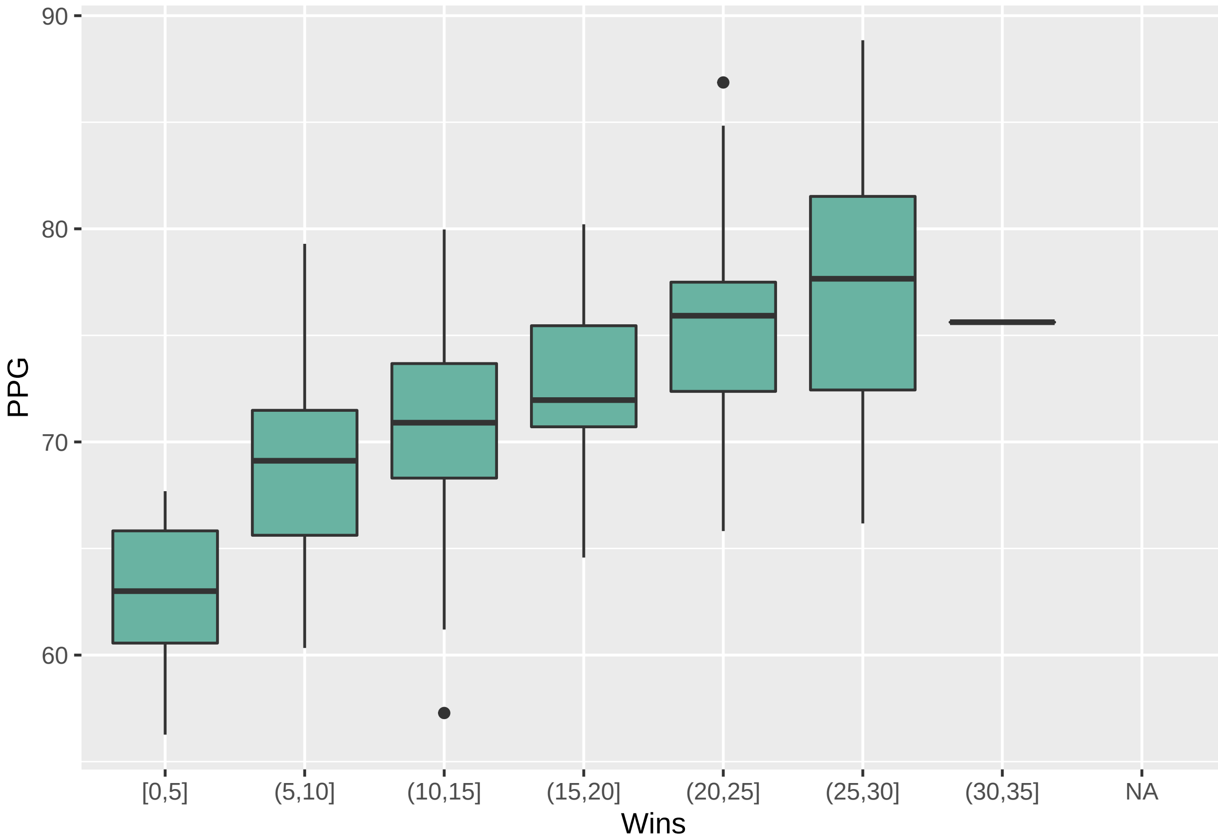
Points Per Game

```
ppgGraph <- s2019_corr %>%

  # Add a new column called 'bin'
  mutate( bin=cut_width(W, width=5, boundary=0) ) %>%
```

```
  # plot
  ggplot( aes(x=bin, y=PPG) ) +
    geom_boxplot(fill="#69b3a2") +
    xlab("Wins")
ppgGraph
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



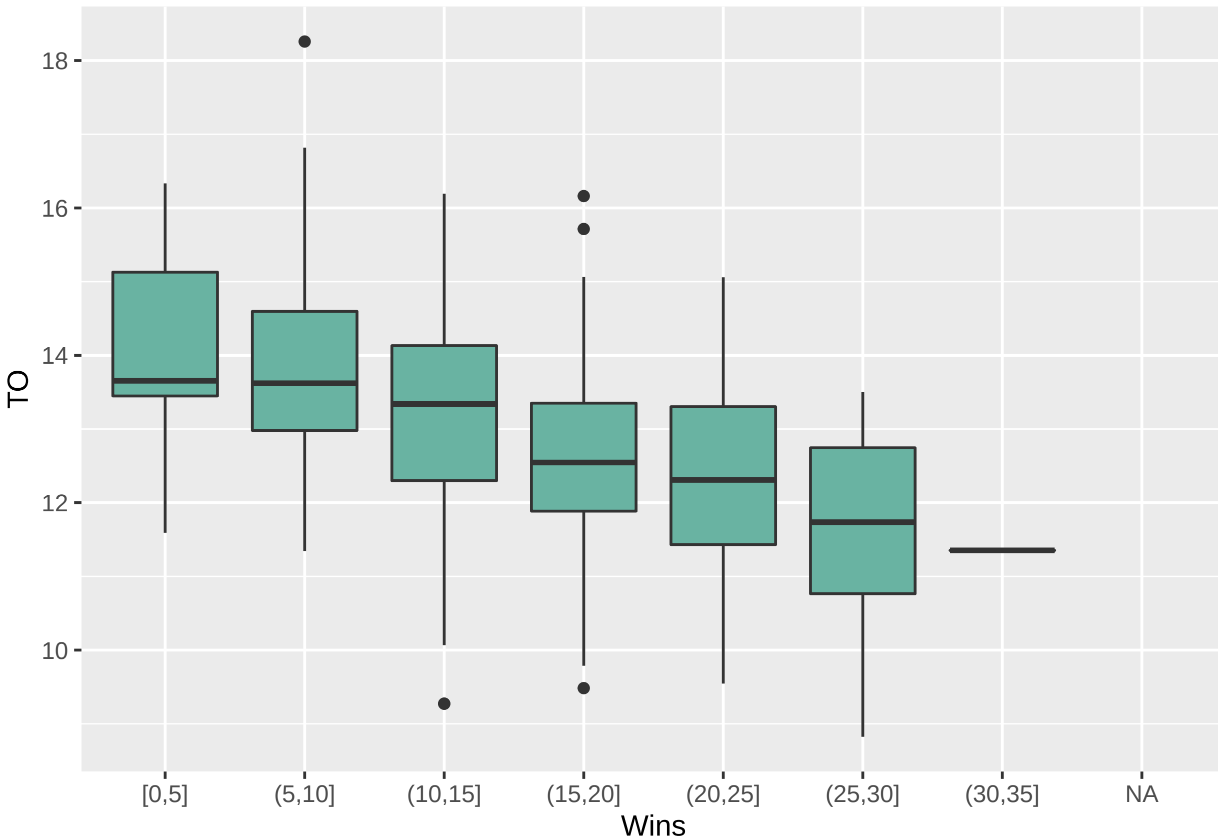
Turnovers

```
turnoverGraph <- s2019_corr %>%

  # Add a new column called 'bin'
  mutate( bin=cut_width(W, width=5, boundary=0) ) %>%

  # plot
  ggplot( aes(x=bin, y=TO) ) +
    geom_boxplot(fill="#69b3a2") +
    xlab("Wins")
turnoverGraph
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



Initial Data Analysis: Trying Linear Regression

We first fit the data to a linear model fitting Wins against all of the variables in the dataset except for Wins, Losses, Win Percentage, Average Losing Score, Average Winning Score, number of overtimes played, and Team ID. We then used the summary function to get an idea of each variables effect on Wins. Predictors such as Points Per Game, Turnovers Per Game, and Turnovers Per Game were highly significant and we expected them to be important variables for our future methods.

```
lm_model_2019_w <-lm(s2019$Wins~.-Losses-W.Per -Average.of.LScore - Average.of.WScore - Average.of.NumO
summary(lm_model_2019_w)
```

```
##
## Call:
## lm(formula = s2019$Wins ~ . - Losses - W.Per - Average.of.LScore -
##     Average.of.WScore - Average.of.NumOT - Average.of.NumOT.1 -
##     Team.ID, data = s2019)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.384 -0.834  0.015  0.808  4.967
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)        21.298      2.107   10.11  < 2e-16 ***
## Average.of.LPF      0.579      0.149    3.89  0.00012 ***
## Average.of.LBlk    -0.322      0.231   -1.39  0.16498
## Average.of.LStl    -1.091      0.197   -5.53  6.9e-08 ***
## Average.of.LTO      1.020      0.158    6.47  3.8e-10 ***
```

```
## Average.of.LAst     0.205      0.151    1.36  0.17588
## Average.of.LDR     -0.940      0.123   -7.64  2.6e-13 ***
## Average.of.LOR     -0.592      0.187   -3.17  0.00167 **
## Average.of.LFTA     0.720      0.184    3.91  0.00011 ***
## Average.of.LFTM    -1.183      0.226   -5.23  3.1e-07 ***
## Average.of.LFGA3    0.288      0.163    1.77  0.07849 .
## Average.of.LFGM3   -0.744      0.282   -2.63  0.00884 **
## Average.of.LFGA     0.794      0.155    5.11  5.6e-07 ***
## Average.of.LFGM    -1.366      0.176   -7.78  1.1e-13 ***
## Average.of.WFGM    -1.236      0.183   -6.73  7.9e-11 ***
## Average.of.WFGA     0.716      0.146    4.91  1.5e-06 ***
## Average.of.WFGM3   -0.913      0.278   -3.29  0.00112 **
## Average.of.WFGA3    0.249      0.168    1.49  0.13841
## Average.of.WFTM    -0.941      0.220   -4.28  2.5e-05 ***
## Average.of.WFTA     0.632      0.192    3.30  0.00109 **
## Average.of.WOR     -0.495      0.188   -2.63  0.00888 **
## Average.of.WDR     -0.723      0.118   -6.13  2.7e-09 ***
## Average.of.WAst     0.346      0.146    2.38  0.01811 *
## Average.of.WTO      0.900      0.158    5.72  2.5e-08 ***
## Average.of.WStl    -1.039      0.196   -5.30  2.2e-07 ***
## Average.of.WBlk    -0.538      0.240   -2.24  0.02593 *
## Average.of.WPF      0.418      0.145    2.89  0.00410 **
## Average.of.PF      -0.969      0.278   -3.48  0.00056 ***
## Average.of.Blk      1.157      0.445    2.60  0.00977 **
## Average.of.Stl      2.585      0.364    7.11  8.1e-12 ***
## Average.of.TO      -2.447      0.286   -8.55  5.6e-16 ***
## Average.of.Ast     -0.525      0.270   -1.95  0.05247 .
## Average.of.DR       1.912      0.215    8.89  < 2e-16 ***
## Average.of.OR       1.568      0.348    4.51  9.1e-06 ***
## Average.of.FTA     -1.292      0.358   -3.61  0.00036 ***
## Average.of.FTM      2.124      0.423    5.02  8.6e-07 ***
## Average.of.FGA3    -0.467      0.304   -1.53  0.12594
## Average.of.FGM3     1.720      0.518    3.32  0.00101 **
## Average.of.FGA     -1.955      0.271   -7.22  3.9e-12 ***
## Average.of.FGM      2.864      0.322    8.88  < 2e-16 ***
## Average.of.PPG         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.37 on 313 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.957,  Adjusted R-squared:  0.951
## F-statistic:  177 on 39 and 313 DF,  p-value: <2e-16
```

The same was done for losses.

```
lm_model_2019_l <-lm(s2019$Losses~.-Wins-W.Per -Average.of.LScore - Average.of.WScore - Average.of.NumO
summary(lm_model_2019_l)
```

```
##
## Call:
## lm(formula = s2019$Losses ~ . - Wins - W.Per - Average.of.LScore -
##     Average.of.WScore - Average.of.NumOT - Average.of.NumOT.1 -
##     Team.ID, data = s2019)
##
```

```
## Residuals:
##    Min     1Q Median     3Q    Max
## -4.211 -0.695  0.091  0.730  4.725
##
## Coefficients: (1 not defined because of singularities)
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       18.6124     2.0018    9.30  < 2e-16 ***
## Average.of.LPF    -0.2592     0.1412   -1.84  0.06736 .
## Average.of.LBlk    0.1427     0.2196    0.65  0.51619
## Average.of.LStl    0.8947     0.1876    4.77  2.8e-06 ***
## Average.of.LTO    -1.0716     0.1498   -7.15  6.1e-12 ***
## Average.of.LAst   -0.0651     0.1434   -0.45  0.65035
## Average.of.LDR     1.0254     0.1168    8.78  < 2e-16 ***
## Average.of.LOR     0.5257     0.1773    2.97  0.00326 **
## Average.of.LFTA   -0.6360     0.1750   -3.63  0.00033 ***
## Average.of.LFTM    0.9915     0.2148    4.62  5.7e-06 ***
## Average.of.LFGA3  -0.2954     0.1549   -1.91  0.05747 .
## Average.of.LFGM3   0.7978     0.2682    2.97  0.00316 **
## Average.of.LFGA   -0.8123     0.1476   -5.50  7.8e-08 ***
## Average.of.LFGM    1.1124     0.1669    6.66  1.2e-10 ***
## Average.of.WFGM    0.9980     0.1743    5.73  2.4e-08 ***
## Average.of.WFGA   -0.5947     0.1385   -4.30  2.3e-05 ***
## Average.of.WFGM3   0.8927     0.2637    3.39  0.00080 ***
## Average.of.WFGA3  -0.3108     0.1595   -1.95  0.05216 .
## Average.of.WFTM    1.0196     0.2090    4.88  1.7e-06 ***
## Average.of.WFTA   -0.5462     0.1821   -3.00  0.00292 **
## Average.of.WOR     0.3755     0.1785    2.10  0.03619 *
## Average.of.WDR     0.8970     0.1122    8.00  2.5e-14 ***
## Average.of.WAst   -0.1242     0.1383   -0.90  0.36986
## Average.of.WTO    -0.8438     0.1496   -5.64  3.8e-08 ***
## Average.of.WStl    0.6572     0.1862    3.53  0.00048 ***
## Average.of.WBlk    0.4732     0.2283    2.07  0.03904 *
## Average.of.WPF    -0.3377     0.1373   -2.46  0.01444 *
## Average.of.PF      0.5559     0.2642    2.10  0.03614 *
## Average.of.Blk    -0.6853     0.4229   -1.62  0.10615
## Average.of.Stl    -1.7149     0.3455   -4.96  1.1e-06 ***
## Average.of.TO      1.9376     0.2719    7.13  7.1e-12 ***
## Average.of.Ast     0.2899     0.2561    1.13  0.25855
## Average.of.DR     -2.0258     0.2043   -9.92  < 2e-16 ***
## Average.of.OR     -1.0602     0.3302   -3.21  0.00146 **
## Average.of.FTA     1.2231     0.3402    3.60  0.00038 ***
## Average.of.FTM    -2.0735     0.4018   -5.16  4.4e-07 ***
## Average.of.FGA3    0.5377     0.2892    1.86  0.06389 .
## Average.of.FGM3   -1.7307     0.4921   -3.52  0.00050 ***
## Average.of.FGA     1.5852     0.2571    6.17  2.2e-09 ***
## Average.of.FGM    -2.3726     0.3063   -7.75  1.3e-13 ***
## Average.of.PPG         NA         NA      NA       NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.3 on 313 degrees of freedom
##   (1 observation deleted due to missingness)
## Multiple R-squared:  0.948,  Adjusted R-squared:  0.941
## F-statistic:  145 on 39 and 313 DF,  p-value: <2e-16
```

Logistic Regression

Next, we fit our data to a Logistic Regression model fitting Wins against all of our other variables. We use the glm function and set the parameter family="Binomial" to compute a Logistic Regression model. This is our most basic model with our highest expected error since we are simply trying to predict win or loss using all of our available variables. We obtained an error rate of 31.34%. This is a fairly high error rate, but is to be expected since college basketball games have a lot of variance and are difficult to predict.

```
# Logistic regression model
glmmodel <- glm(W ~ Average.of.PF + Average.of.Blk + Average.of.Stl + Average.of.TO + Average.of.Ast + A

pred <- predict(glmmodel, train, type = "response")

# Turn predictions from percentage to actual outcome, with 0.5 cutoff
for(i in 1:length(pred)){
    if(pred[i]>0.5){
        pred[i] <- 1
    } else {
        pred[i] <- 0
    }
}
trainError <- table(pred, train$W)

# Train error
1 - sum(diag(trainError))/sum(trainError)
```

```
## [1] 0.2859
```

```
testpred <- predict(glmmodel, tourney_2019, type = "response")
# Change predictions to 1 for > 50% win, 0 for < 50% win
for(i in 1:length(testpred)){
    if(testpred[i]>0.5){
        testpred[i] <- 1
    } else {
        testpred[i] <- 0
    }
}

error <- table(testpred, tourney_2019$W)
# Test error on NCAA Tournament
(error[1,]/(error[1,] + error[2,]))
```

```
## [1] 0.3134
```

Decision Tree

We also decided to fit a Decision Tree model because splitting our response variable based on key variables might be able to get us better results. After pruning the tree using Cross-Validation, and we get the lowest standard deviation by reducing our tree to size four. The model splits on the difference of Average Points Per Game, and predicts a team to win if they average more than 1.8 PPG than their opponent, and to lose if they average less than 1.8 PPG than their opponent. Using this Decision Tree model, we obtained a test error rate of 53.73%, significantly less accurate than our Logistic Regression model and below the random guess threshold, so we decided that Decisions Trees would not be a good fit for predicting our response variable.

```r
train$W <- as.factor(train$W)
tree.s2019 = tree(W ~ Average.of.PF + Average.of.Blk + Average.of.Stl + Average.of.TO + Average.of.Ast +

tree.cv = cv.tree(tree.s2019, FUN=prune.misclass, K=10)
tree.cv
```

```
## $size
## [1] 4 2 1
##
## $dev
## [1] 2034 2050 2802
##
## $k
## [1] -Inf    0  742
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

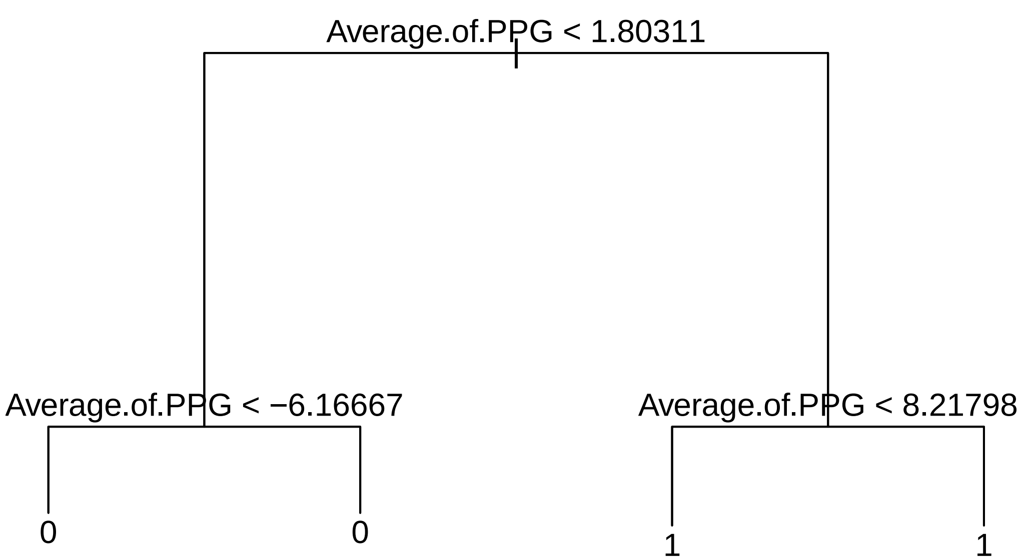
Since the lowest standard deviation is at size 4, we prune for size 4.

```r
tree.prune = prune.misclass(tree.s2019, best=4)

tree.err = table(treePred=predict(tree.prune, newdata=tourney_2019, type="class"),truth=tourney_2019$W)
# Tree test error
(tree.err[1,]/(tree.err[1,] + tree.err[2,]))
```

```
## [1] 0.5373
```

```r
# Plot of decision tree
plot(tree.prune)
text(tree.prune)
```



Random Forest

We decided to fit a Random Forest model next because we had a lot of variables in our dataset and want to use Cross-Validation to reduce our number of variables. We decided to use 500 trees and set our mtry parameter to be 4 because for classification we use $\sqrt{p}$ and $\sqrt{14} \approx 4$. We obtained an error rate of 35.82%

```
# Random Forest, m = 4 since m is sqrt(p) for random forest and sqrt(14) ~ 4
rf.s2019 = randomForest(W ~ Average.of.PF + Average.of.Blk + Average.of.Stl + Average.of.TO + Average.o
rf.s2019
```

```
##
## Call:
##  randomForest(formula = W ~ Average.of.PF + Average.of.Blk + Average.of.Stl +      Average.of.TO + A
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##         OOB estimate of  error rate: 33.26%
## Confusion matrix:
##      0    1 class.error
## 0 1884  848      0.3104
## 1  969 1762      0.3548
```

```
yhat.rf = predict (rf.s2019, newdata = tourney_2019)
rf.err = table(pred = yhat.rf, truth = tourney_2019$W)
# Random Forest Test Error
(rf.err[1,]/(rf.err[1,] + rf.err[2,]))
```

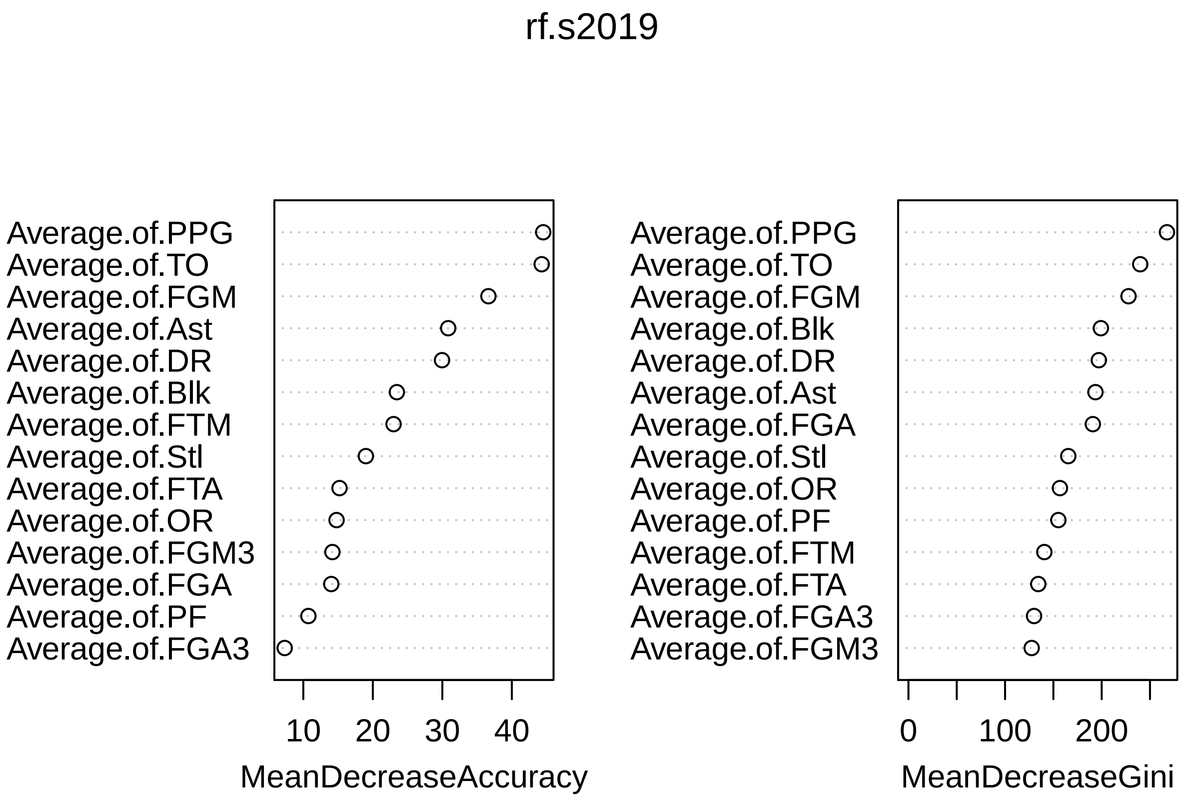
```
## [1] 0.3582
```

Variable Importance

```
importance(rf.s2019)
```

```
##                         0      1 MeanDecreaseAccuracy MeanDecreaseGini
## Average.of.PF    8.690  5.259               10.732            155.2
## Average.of.Blk  19.446 12.567               23.458            199.2
## Average.of.Stl  12.684 12.783               19.001            165.4
## Average.of.TO   35.188 23.774               44.292            239.9
## Average.of.Ast  10.707 27.629               30.844            193.6
## Average.of.DR   22.412 16.763               29.957            197.1
## Average.of.OR    7.265 12.394               14.787            156.8
## Average.of.FTA   9.276  8.486               15.213            134.4
## Average.of.FTM  10.709 16.706               22.985            140.6
## Average.of.FGA3  2.505  6.038                7.326            130.0
## Average.of.FGM3  8.065  7.478               14.188            127.5
## Average.of.FGA   6.272 10.795               14.029            190.9
## Average.of.FGM  13.558 27.375               36.629            227.8
## Average.of.PPG  18.640 29.545               44.510            267.6
```

```
varImpPlot(rf.s2019)
```

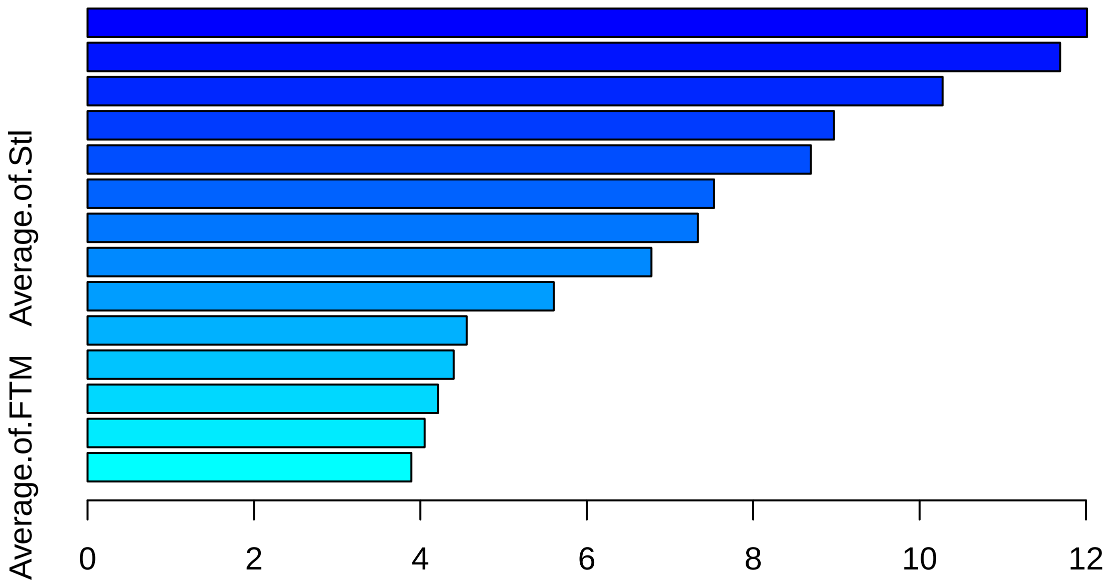


We see that Point Per Game and Turnovers are the most important variables, just like we saw in our initial data exploration.

Boosting

We also fit a gbm (Boosting) model to compare to our Random Forest model. We decided to use 500 trees and our interaction depth to be 4. This provided us with an error rate of 35.82%, which is the same as the error rate from our Random Forest model.

```
boost.s2019 = gbm((unclass(W)-1) ~ Average.of.PF + Average.of.Blk + Average.of.Stl + Average.of.TO + Ave
summary(boost.s2019)
```




```
##                                   var rel.inf
## Average.of.TO      Average.of.TO  12.013
## Average.of.PPG    Average.of.PPG  11.690
## Average.of.FGA    Average.of.FGA  10.277
## Average.of.DR      Average.of.DR   8.971
## Average.of.OR      Average.of.OR   8.694
## Average.of.Blk    Average.of.Blk   7.530
## Average.of.Stl    Average.of.Stl   7.335
## Average.of.FGM    Average.of.FGM   6.777
## Average.of.Ast    Average.of.Ast   5.603
## Average.of.PF      Average.of.PF   4.556
## Average.of.FGM3 Average.of.FGM3   4.400
## Average.of.FGA3 Average.of.FGA3   4.212
## Average.of.FTA    Average.of.FTA   4.051
## Average.of.FTM    Average.of.FTM   3.892
```

```
yhat.boost = predict(boost.s2019, newdata = tourney_2019, n.trees=500)
boost.err = table(pred = yhat.rf, truth = tourney_2019$W)
(boost.err[1,]/(boost.err[1,] + boost.err[2,]))
```

```
## [1] 0.3582
```

K Nearest Neighbors

We decided to try a K Nearest Neighbor model because KNN models tend to work well with classification prediction. We used Leave One Out Cross Validation to select the optimal value of K. We obtained an error rate of 32.83%. Our KNN model performed fairly well compared to our previous models but still slightly more inaccurate compared to our Logistic Regression model.

```
library(class)
YTrain <- train$W
XTrain <- train %>% select(Average.of.PF, Average.of.Blk , Average.of.Stl , Average.of.TO , Average.of.
```

```r
XTest <- tourney_2019 %>% select(Average.of.PF, Average.of.Blk , Average.of.Stl , Average.of.TO , Averag

trainpred = knn(train=XTrain, test=XTrain, cl=YTrain, k=2)
# Training Error
trainError = table(predicted = trainpred, observed=YTrain)
1 - sum(diag(trainError)/sum(trainError))
```

```
## [1] 0.1962
```

```r
testpred <- knn(train = XTrain, test = XTest, cl = YTrain, k = 2)
testError <- table(predicted = testpred, observed = tourney_2019$W)
# Test Error
(testError[1,]/(testError[1,] + testError[2,]))
```

```
## [1] 0.3731
```

LOOCV for K Nearest Neighbors

```r
# Code from Lab on K Nearest Neighbors
allK = 1:50
validation.error = NULL
for (i in allK){
  pred.Yval = knn.cv(train=XTrain, cl=YTrain, k=i) # Predict on the left-out validation set
  validation.error = c(validation.error, mean(pred.Yval!=YTrain)) # Combine all validation errors
}
numneighbor = max(allK[validation.error == min(validation.error)])
numneighbor
```

```
## [1] 37
```

```r
# Get test error rate for CV Selected NN Classifier
pred.YTest = knn(train=XTrain, test=XTest, cl=YTrain, k=numneighbor)
testError = table(predicted=pred.YTest, true= tourney_2019$W)
(testError[1,]/(testError[1,] + testError[2,]))
```

```
## [1] 0.3284
```

Principal Component Analysis

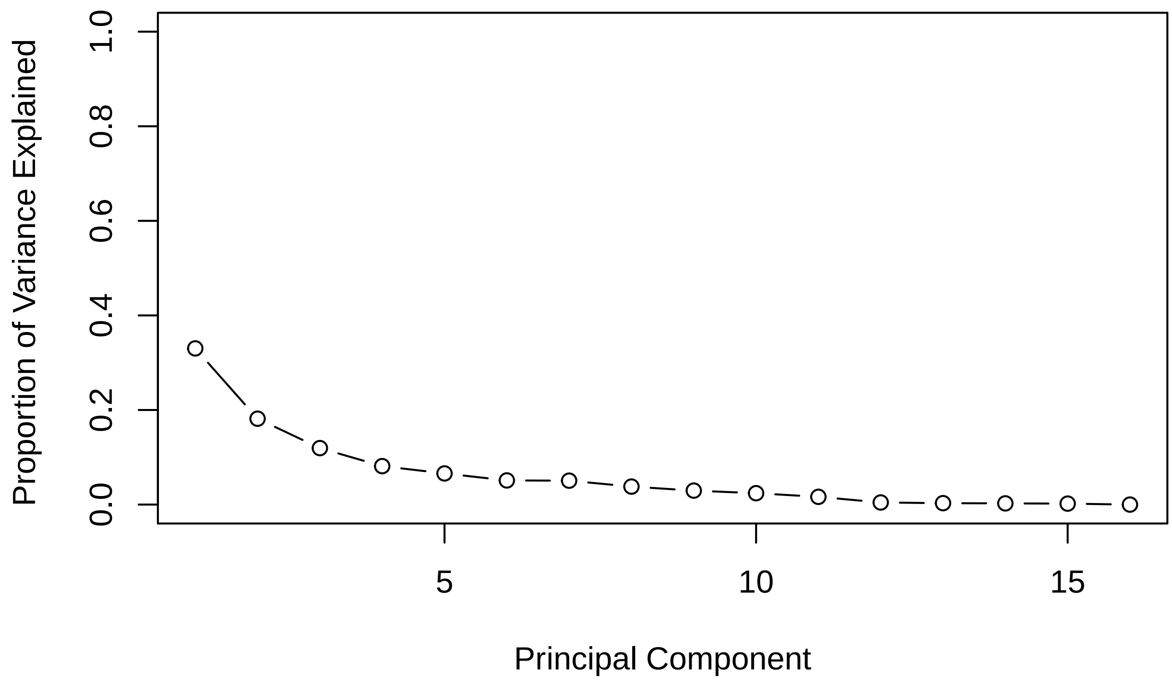
After trying Principal Component Analysis on our dataset, it became clear that the method was not appropriate. The principal components explained very little proportion of variance. The main principal component explained less than 40% while the others explain less than 20%. We feel that reducing noise was not necessary, especially with such low explanations in the proportion of variance.

```r
compdata <- s2019_corr
compdata <- compdata[-354,]
comp <- prcomp(compdata, scale = TRUE)
comp
```

```
## Standard deviations (1, .., p=16):
##  [1] 2.299e+00 1.705e+00 1.382e+00 1.141e+00 1.026e+00 9.040e-01 8.992e-01
##  [8] 7.799e-01 6.872e-01 6.204e-01 5.118e-01 2.663e-01 2.192e-01 2.012e-01
## [15] 1.797e-01 6.649e-10
##
## Rotation (n x k) = (16 x 16):
##             PC1        PC2        PC3       PC4       PC5       PC6       PC7        PC8
```

```
## L     0.34936  0.011197  0.288545  0.06566 -0.10985  0.23417 -0.08324  0.395732
## W    -0.34612  0.001736 -0.329234 -0.07485  0.10101 -0.23857  0.10751 -0.331035
## PPG  -0.40240  0.034249  0.119810  0.11878  0.11654  0.17699 -0.10952  0.110329
## FGM  -0.38020 -0.046078  0.061728 -0.07522 -0.04472  0.37813 -0.18568  0.003928
## FGA  -0.26394  0.032735  0.417824 -0.18357 -0.20154  0.34380  0.26089  0.116748
## FGM3 -0.20792 -0.359449  0.333209  0.15409  0.14922 -0.34151 -0.01308  0.045938
## FGA3 -0.16089 -0.311265  0.438222  0.09328  0.06335 -0.42040  0.10656  0.118277
## FTM  -0.17909  0.416976 -0.026542  0.35479  0.29302 -0.04333  0.08177  0.250860
## FTA  -0.14686  0.478048 -0.011733  0.27477  0.21376 -0.05027  0.06339  0.231795
## OR   -0.11266  0.332513  0.232307 -0.24055 -0.28591 -0.01459  0.47846 -0.294281
## DR   -0.24712  0.011442 -0.094757  0.43725 -0.48108  0.01020 -0.04375 -0.128849
## AST  -0.31540 -0.102576 -0.001276 -0.02385 -0.05897  0.16245 -0.49305 -0.121093
## TO    0.16768  0.299724  0.241991  0.09583 -0.29144 -0.20484 -0.52873 -0.175236
## STL  -0.14452  0.183155  0.036904 -0.61691  0.30177 -0.09821 -0.26505  0.159004
## BLK  -0.17185  0.129720 -0.199130 -0.25072 -0.50387 -0.43541 -0.04984  0.497269
## PF    0.08648  0.324947  0.386231 -0.02362  0.13095 -0.17340 -0.13607 -0.392557
##            PC9     PC10     PC11     PC12     PC13     PC14      PC15
## L    -0.063152  0.04678 -0.14758 -0.37940  0.01591 -0.22538 -0.580401
## W     0.007062 -0.01792  0.03361  0.04799  0.07752 -0.06762 -0.750703
## PPG   0.074220  0.10976  0.33057 -0.12665 -0.06186 -0.15144  0.028351
## FGM   0.163907  0.11105  0.40784 -0.16895  0.29725 -0.16558  0.053068
## FGA   0.167698 -0.12598 -0.13555  0.50378 -0.12709  0.32555 -0.205390
## FGM3 -0.056269  0.04742  0.14811 -0.35845 -0.39702  0.44531 -0.059042
## FGA3 -0.114740 -0.13636 -0.11742  0.22027  0.44206 -0.41926  0.071610
## FTM  -0.086084  0.04079 -0.02295  0.23314 -0.47523 -0.37045  0.010471
## FTA  -0.135667 -0.04609 -0.11567 -0.14242  0.53033  0.48252 -0.004990
## OR   -0.453839  0.06137  0.05792 -0.33681 -0.07004 -0.14648  0.120562
## DR    0.283366 -0.47938 -0.29884 -0.26377 -0.07518 -0.08819  0.078543
## AST  -0.454854  0.31914 -0.53512  0.05840 -0.04136  0.02252  0.032920
## TO   -0.232296 -0.23223  0.41968  0.26118 -0.02089  0.07859 -0.135256
## STL   0.064116 -0.52487 -0.15799 -0.20532 -0.10751 -0.07048  0.064306
## BLK   0.170393  0.35624  0.01793  0.01398 -0.02090  0.01625  0.027191
## PF    0.559512  0.37532 -0.23035 -0.08209  0.02389 -0.05485  0.008338
##            PC16
## L     1.655e-10
## W     1.127e-10
## PPG  -7.543e-01
## FGM   5.583e-01
## FGA  -1.934e-12
## FGM3  1.907e-01
## FGA3  3.752e-11
## FTM   2.881e-01
## FTA  -1.575e-10
## OR    2.218e-11
## DR   -5.364e-11
## AST  -9.595e-11
## TO    8.882e-11
## STL  -2.936e-11
## BLK   1.164e-11
## PF   -3.610e-11
```

```r
comp.var <- comp$sdev ^ 2
pve <- comp.var/sum(comp.var)
plot(pve, xlab="Principal Component", ylab="Proportion of Variance Explained ", ylim=c(0,1),type='b')
```



Model Selection

```
Method <- c('LR' , 'DT' , 'KNN' , 'RF' , 'BOOST')
Test_Error <- c('31.34%','53.73%','32.83%','35.82%','35.82%')
test.errors <- data.frame(Method,Test_Error)
test.errors
```

```
##   Method Test_Error
## 1     LR     31.34%
## 2     DT     53.73%
## 3    KNN     32.83%
## 4     RF     35.82%
## 5  BOOST     35.82%
```

We arrived at our final model by choosing the classification method that most accurately predicted the NCAA March Madness Tournament results from training on the NCAA Regular Season. We tried a variety of different modeling methods that we believed would most accurately predict our binary response variable. After trying advanced modeling methods such as Random Forest and Boosting, but the Logistic Regression model still had the lowest test error rate of 31.34%.

Extending Logistic Regression to 2018 Data Set

We tested using logistic regression on the 2018 regular season data set to see if we would get a similar test error rate (on the NCAA Tournament) compared to the 2019 data set.

```
glmmodel <- glm(W ~ Average.of.PF + Average.of.Blk + Average.of.Stl + Average.of.TO + Average.of.Ast + A

pred <- predict(glmmodel, train, type = "response")
```

```r
for(i in 1:length(pred)){
    if(pred[i]>0.5){
        pred[i] <- 1
    } else {
        pred[i] <- 0
    }
}

trainError <- table(pred, train$W)

# Train error
1 - sum(diag(trainError))/sum(trainError)
```

```
## [1] 0.2818
```

```r
testpred <- predict(glmmodel, tourney_2018, type = "response")

# Change predictions to 1 for > 50% win, 0 for < 50% win
for(i in 1:length(testpred)){
    if(testpred[i]>0.5){
        testpred[i] <- 1
    } else {
        testpred[i] <- 0
    }
}

error <- table(testpred, tourney_2018$W)

# Train error on NCAA Tournament
(error[1,]/(error[1,] + error[2,]))
```

```
## [1] 0.3433
```

The test error is 34.33%, slightly higher but still similar to our test error for the 2019 data set.

Conclusion

After testing several candidate models, in the end we chose the Logistic Regression model because it gave us the lowest test error and was easy to interpret. This model produced a test error of 31.34%, which was more accurate in predicting tournament games than decision trees and KNN.

In the future, we would like to predict this year's March Madness tournament. Unfortunately, the tournament seeds and teams are set on March 15 at 4 p.m. PST. Therefore, we will not be able to predict the results in time for the project due date. If you would like to know our 2020 predictions, feel free to contact us.

In addition, we filled out the 2018 and 2019 predictions from our backets. The brackets are below. The red boxes are incorrect predictions, while the rest are correctly predicted.

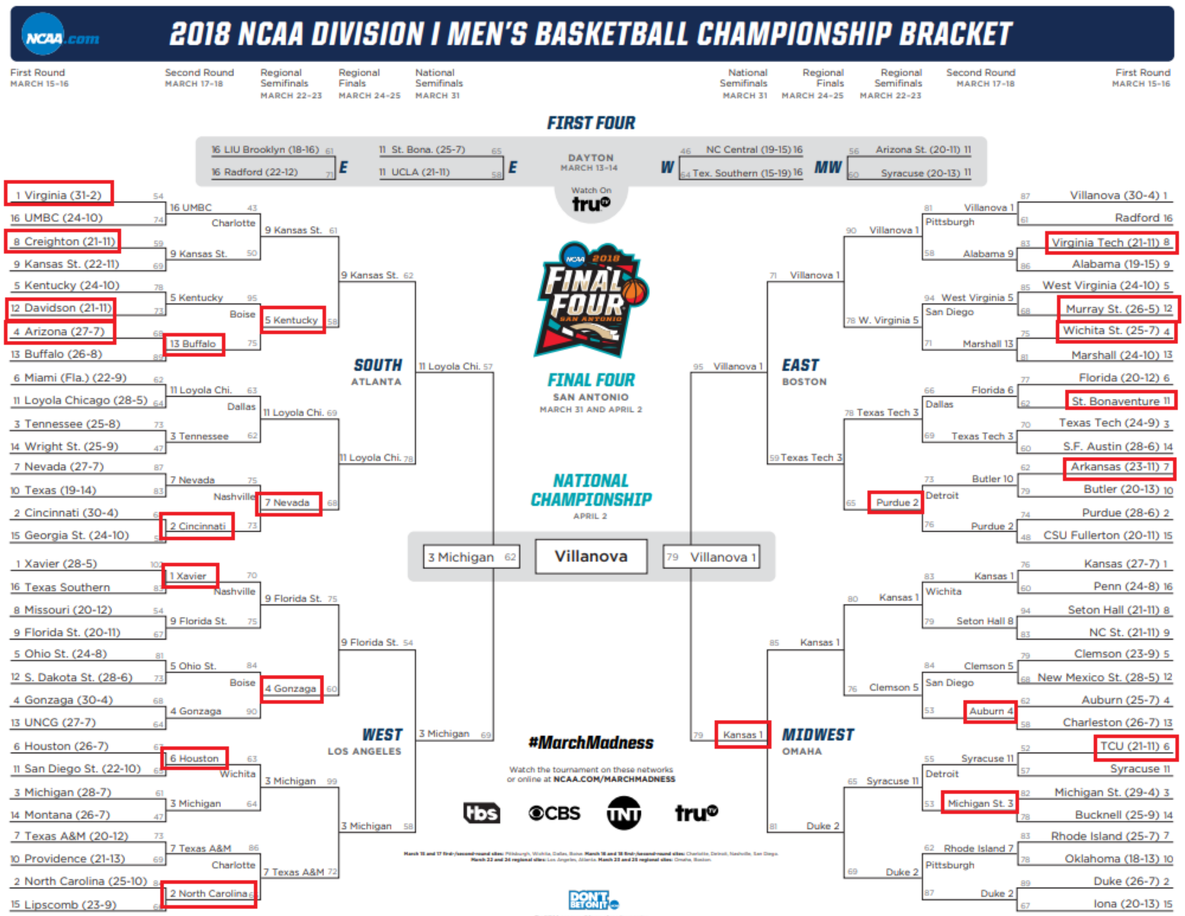


Figure 1: 2018 NCAA Tournament Predictions

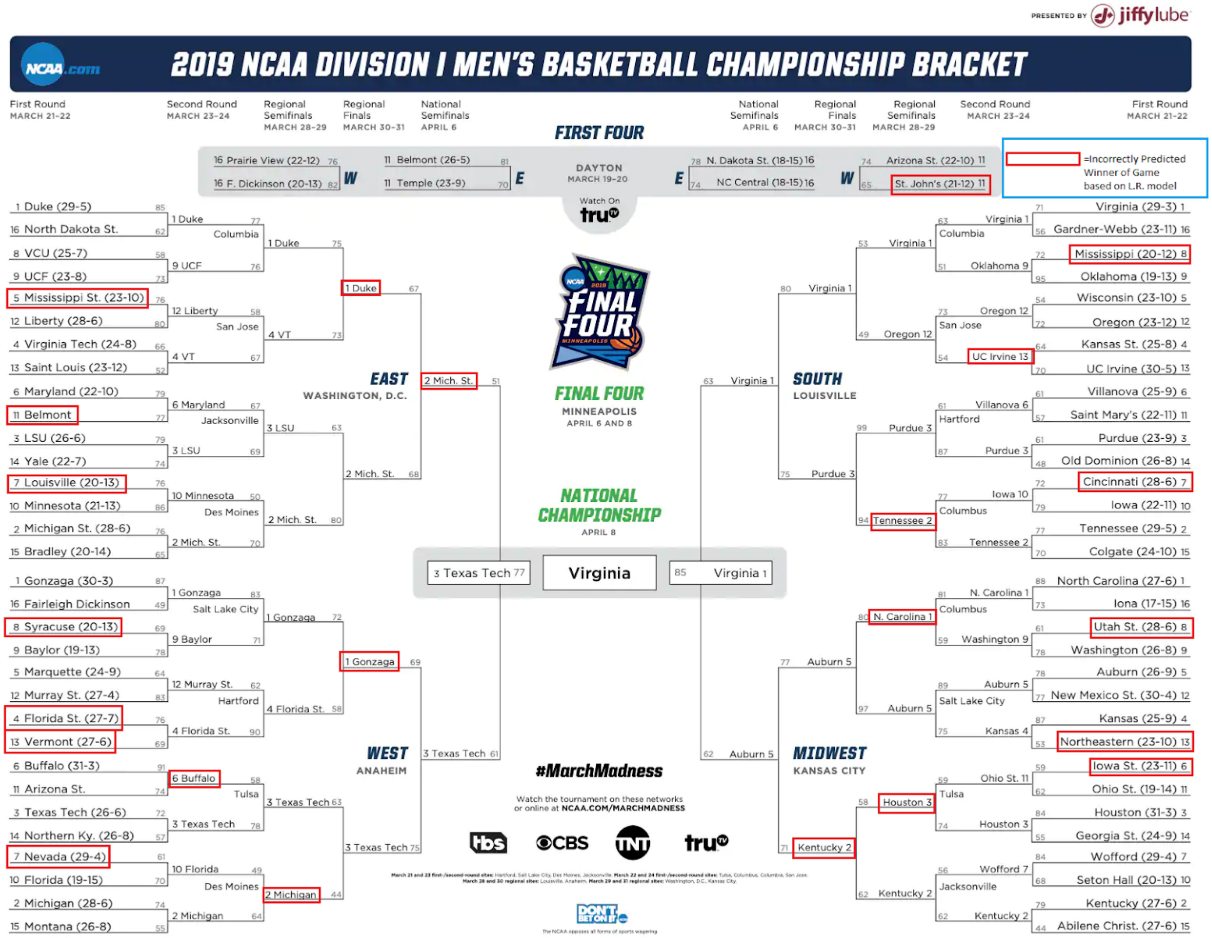


Figure 2: 2019 NCAA Tournament Predictions

Appendix

Initial data manipulation for both 2019 and 2018

```
# Create an average of each statistic by combining win statistic and loss statistics
s2019$Average.of.PF  <- (((s2019$Losses * s2019$Average.of.LPF) + (s2019$Wins * s2019$Average.of.WPF))/
s2019$Average.of.Blk <- (((s2019$Losses * s2019$Average.of.LBlk) + (s2019$Wins * s2019$Average.of.WBlk)
s2019$Average.of.Stl <- (((s2019$Losses * s2019$Average.of.LStl) + (s2019$Wins * s2019$Average.of.WStl)
s2019$Average.of.TO <- (((s2019$Losses * s2019$Average.of.LTO) + (s2019$Wins * s2019$Average.of.WTO))/(
s2019$Average.of.Ast <- (((s2019$Losses * s2019$Average.of.LAst) + (s2019$Wins * s2019$Average.of.WAst)
s2019$Average.of.DR <- (((s2019$Losses * s2019$Average.of.LDR) + (s2019$Wins * s2019$Average.of.WDR))/(
s2019$Average.of.OR <- (((s2019$Losses * s2019$Average.of.LOR) + (s2019$Wins * s2019$Average.of.WOR))/(
s2019$Average.of.FTA <- (((s2019$Losses * s2019$Average.of.LFTA) + (s2019$Wins * s2019$Average.of.WFTA)
s2019$Average.of.FTM <- (((s2019$Losses * s2019$Average.of.LFTM) + (s2019$Wins * s2019$Average.of.WFTM)
s2019$Average.of.FGA3 <- (((s2019$Losses * s2019$Average.of.LFGA3) + (s2019$Wins * s2019$Average.of.WFG
s2019$Average.of.FGM3 <- (((s2019$Losses * s2019$Average.of.LFGM3) + (s2019$Wins * s2019$Average.of.WFG
s2019$Average.of.FGA <- (((s2019$Losses * s2019$Average.of.LFGA) + (s2019$Wins * s2019$Average.of.WFGA)
s2019$Average.of.FGM <- (((s2019$Losses * s2019$Average.of.LFGM) + (s2019$Wins * s2019$Average.of.WFGM)
s2019$Average.of.PPG  <- (((s2019$Losses * s2019$Average.of.LScore) + (s2019$Wins * s2019$Average.of.WS
s2019$W.Per  <- (s2019$Wins)/(s2019$Wins + s2019$Losses)
```

```
s2018$Average.of.PF  <- (((s2018$Losses * s2018$Average.of.LPF) + (s2018$Wins * s2018$Average.of.WPF))/
s2018$Average.of.Blk <- (((s2018$Losses * s2018$Average.of.LBlk) + (s2018$Wins * s2018$Average.of.WBlk)
s2018$Average.of.Stl <- (((s2018$Losses * s2018$Average.of.LStl) + (s2018$Wins * s2018$Average.of.WStl)
s2018$Average.of.TO <- (((s2018$Losses * s2018$Average.of.LTO) + (s2018$Wins * s2018$Average.of.WTO))/(
s2018$Average.of.Ast <- (((s2018$Losses * s2018$Average.of.LAst) + (s2018$Wins * s2018$Average.of.WAst)
s2018$Average.of.DR <- (((s2018$Losses * s2018$Average.of.LDR) + (s2018$Wins * s2018$Average.of.WDR))/(
s2018$Average.of.OR <- (((s2018$Losses * s2018$Average.of.LOR) + (s2018$Wins * s2018$Average.of.WOR))/(
s2018$Average.of.FTA <- (((s2018$Losses * s2018$Average.of.LFTA) + (s2018$Wins * s2018$Average.of.WFTA)
s2018$Average.of.FTM <- (((s2018$Losses * s2018$Average.of.LFTM) + (s2018$Wins * s2018$Average.of.WFTM)
s2018$Average.of.FGA3 <- (((s2018$Losses * s2018$Average.of.LFGA3) + (s2018$Wins * s2018$Average.of.WFG
s2018$Average.of.FGM3 <- (((s2018$Losses * s2018$Average.of.LFGM3) + (s2018$Wins * s2018$Average.of.WFG
s2018$Average.of.FGA <- (((s2018$Losses * s2018$Average.of.LFGA) + (s2018$Wins * s2018$Average.of.WFGA)
s2018$Average.of.FGM <- (((s2018$Losses * s2018$Average.of.LFGM) + (s2018$Wins * s2018$Average.of.WFGM)
s2018$Average.of.PPG  <- (((s2018$Losses * s2018$Average.of.LScore) + (s2018$Wins * s2018$Average.of.WS
s2018$W.Per  <- (s2018$Wins)/(s2018$Wins + s2018$Losses)
```

Getting difference of statistics between teams for regular season matches

```
# Set response variable to 1, since in data set all are wins.
reg_season_2019_matchups$W <- 1

# Get difference in statistics between the two teams for each match in the regular season
for(i in 1:nrow(reg_season_2019_matchups)){
    reg_season_2019_matchups$Average.of.LPF[i] <- (((s2019$Average.of.LPF[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.LBlk[i] <- (((s2019$Average.of.LBlk[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LStl[i] <- (((s2019$Average.of.LStl[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LTO[i] <- (((s2019$Average.of.LTO[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.LAst[i] <- (((s2019$Average.of.LAst[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LDR[i] <- (((s2019$Average.of.LDR[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.LOR[i] <- (((s2019$Average.of.LOR[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.LFTA[i] <- (((s2019$Average.of.LFTA[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LFTM[i] <- (((s2019$Average.of.LFTM[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LFGA3[i] <- (((s2019$Average.of.LFGA3[s2019$Team.ID ==reg_seaso
    reg_season_2019_matchups$Average.of.LFGM3[i] <- (((s2019$Average.of.LFGM3[s2019$Team.ID ==reg_seaso
```

```
    reg_season_2019_matchups$Average.of.LFGA[i] <- (((s2019$Average.of.LFGA[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.LFGM[i] <- (((s2019$Average.of.LFGM[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.NumOT[i] <- (((s2019$Average.of.NumOT[s2019$Team.ID ==reg_seasor
    reg_season_2019_matchups$Average.of.LScore[i] <- (((s2019$Average.of.LScore[s2019$Team.ID ==reg_seas
    reg_season_2019_matchups$Average.of.WScore[i] <- (((s2019$Average.of.WScore[s2019$Team.ID ==reg_seas
    reg_season_2019_matchups$Average.of.NumOT.1[i] <- (((s2019$Average.of.NumOT.1[s2019$Team.ID ==reg_se
    reg_season_2019_matchups$Average.of.WFGM[i] <- (((s2019$Average.of.WFGM[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WFGA[i] <- (((s2019$Average.of.WFGA[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WFGM3[i] <- (((s2019$Average.of.WFGM3[s2019$Team.ID ==reg_seasor
    reg_season_2019_matchups$Average.of.WFGA3[i] <- (((s2019$Average.of.WFGA3[s2019$Team.ID ==reg_seasor
    reg_season_2019_matchups$Average.of.WFTM[i] <- (((s2019$Average.of.WFTM[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WFTA[i] <- (((s2019$Average.of.WFTA[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WOR[i] <- (((s2019$Average.of.WOR[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.WDR[i] <- (((s2019$Average.of.WDR[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.WAst[i] <- (((s2019$Average.of.WAst[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WTO[i] <- (((s2019$Average.of.WTO[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.WStl[i] <- (((s2019$Average.of.WStl[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WBlk[i] <- (((s2019$Average.of.WBlk[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.WPF[i] <- (((s2019$Average.of.WPF[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.PF[i] <- (((s2019$Average.of.PF[s2019$Team.ID ==reg_season_2019
    reg_season_2019_matchups$Average.of.Blk[i] <- (((s2019$Average.of.Blk[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.Stl[i] <- (((s2019$Average.of.Stl[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.TO[i] <- (((s2019$Average.of.TO[s2019$Team.ID ==reg_season_2019
    reg_season_2019_matchups$Average.of.Ast[i] <- (((s2019$Average.of.Ast[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.DR[i] <- (((s2019$Average.of.DR[s2019$Team.ID ==reg_season_2019
    reg_season_2019_matchups$Average.of.OR[i] <- (((s2019$Average.of.OR[s2019$Team.ID ==reg_season_2019
    reg_season_2019_matchups$Average.of.FTA[i] <- (((s2019$Average.of.FTA[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.FTM[i] <- (((s2019$Average.of.FTM[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.FGA3[i] <- (((s2019$Average.of.FGA3[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.FGM3[i] <- (((s2019$Average.of.FGM3[s2019$Team.ID ==reg_season_
    reg_season_2019_matchups$Average.of.FGA[i] <- (((s2019$Average.of.FGA[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.FGM[i] <- (((s2019$Average.of.FGM[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$Average.of.PPG[i] <- (((s2019$Average.of.PPG[s2019$Team.ID ==reg_season_20
    reg_season_2019_matchups$W.Per[i] <- (((s2019$W.Per[s2019$Team.ID ==reg_season_2019_matchups$WTeamI
}
```

Building train set for all methods

```
# Randomly change half of outcomes to losses and invert statistic differences so model works appropriat
trainrows <- sample(nrow(reg_season_2019_matchups), nrow(reg_season_2019_matchups) * 0.5)
winset <- reg_season_2019_matchups[trainrows,]
loseset <- reg_season_2019_matchups[-trainrows,]
# Invert differences in statistics

loseset <- loseset %>%
    mutate_if(is.numeric, funs(. * -1))
# Change actual outcome to loss
loseset$W <- 0
train <- rbind(winset, loseset)
```

2019 Tourney Data Manipulation

```
tourney <- read.csv("MNCAATourneyCompactResults.csv")
tourney_2019 <- tourney[tourney$Season == 2019,]
```

```r
# Data manipulation to add statistic differences to the March Madness Tournament
for(i in 1:nrow(tourney_2019)){
    tourney_2019$Average.of.LPF[i] <- (((s2019$Average.of.LPF[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.LBlk[i] <- (((s2019$Average.of.LBlk[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LStl[i] <- (((s2019$Average.of.LStl[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LTO[i] <- (((s2019$Average.of.LTO[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.LAst[i] <- (((s2019$Average.of.LAst[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LDR[i] <- (((s2019$Average.of.LDR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.LOR[i] <- (((s2019$Average.of.LOR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.LFTA[i] <- (((s2019$Average.of.LFTA[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LFTM[i] <- (((s2019$Average.of.LFTM[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LFGA3[i] <- (((s2019$Average.of.LFGA3[s2019$Team.ID ==tourney_2019$WTeamID[
    tourney_2019$Average.of.LFGM3[i] <- (((s2019$Average.of.LFGM3[s2019$Team.ID ==tourney_2019$WTeamID[
    tourney_2019$Average.of.LFGA[i] <- (((s2019$Average.of.LFGA[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.LFGM[i] <- (((s2019$Average.of.LFGM[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.NumOT[i] <- (((s2019$Average.of.NumOT[s2019$Team.ID ==tourney_2019$WTeamID[
    tourney_2019$Average.of.LScore[i] <- (((s2019$Average.of.LScore[s2019$Team.ID ==tourney_2019$WTeamI
    tourney_2019$Average.of.WScore[i] <- (((s2019$Average.of.WScore[s2019$Team.ID ==tourney_2019$WTeamI
    tourney_2019$Average.of.NumOT.1[i] <- (((s2019$Average.of.NumOT.1[s2019$Team.ID ==tourney_2019$WTea
    tourney_2019$Average.of.WFGM[i] <- (((s2019$Average.of.WFGM[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WFGA[i] <- (((s2019$Average.of.WFGA[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WFGM3[i] <- (((s2019$Average.of.WFGM3[s2019$Team.ID ==tourney_2019$WTeamID[
    tourney_2019$Average.of.WFGA3[i] <- (((s2019$Average.of.WFGA3[s2019$Team.ID ==tourney_2019$WTeamID[
    tourney_2019$Average.of.WFTM[i] <- (((s2019$Average.of.WFTM[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WFTA[i] <- (((s2019$Average.of.WFTA[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WOR[i] <- (((s2019$Average.of.WOR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.WDR[i] <- (((s2019$Average.of.WDR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.WAst[i] <- (((s2019$Average.of.WAst[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WTO[i] <- (((s2019$Average.of.WTO[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.WStl[i] <- (((s2019$Average.of.WStl[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WBlk[i] <- (((s2019$Average.of.WBlk[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.WPF[i] <- (((s2019$Average.of.WPF[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.PF[i] <- (((s2019$Average.of.PF[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.Blk[i] <- (((s2019$Average.of.Blk[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.Stl[i] <- (((s2019$Average.of.Stl[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.TO[i] <- (((s2019$Average.of.TO[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.Ast[i] <- (((s2019$Average.of.Ast[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.DR[i] <- (((s2019$Average.of.DR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.OR[i] <- (((s2019$Average.of.OR[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.FTA[i] <- (((s2019$Average.of.FTA[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.FTM[i] <- (((s2019$Average.of.FTM[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.FGA3[i] <- (((s2019$Average.of.FGA3[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.FGM3[i] <- (((s2019$Average.of.FGM3[s2019$Team.ID ==tourney_2019$WTeamID[i]
    tourney_2019$Average.of.FGA[i] <- (((s2019$Average.of.FGA[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.FGM[i] <- (((s2019$Average.of.FGM[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$Average.of.PPG[i] <- (((s2019$Average.of.PPG[s2019$Team.ID ==tourney_2019$WTeamID[i]])-
    tourney_2019$W.Per[i] <- (((s2019$W.Per[s2019$Team.ID ==tourney_2019$WTeamID[i]])- (s2019$W.Per[s20
}

tourney_2019$W <- 1
```

2018 Tourney Data Manipulation

```r
tourney <- read.csv("MNCAATourneyCompactResults.csv")
tourney_2018 <- tourney[tourney$Season == 2018,]
```

```r
for(i in 1:nrow(tourney_2018)){

    tourney_2018$Average.of.LPF[i] <- (((s2018$Average.of.LPF[s2018$Team.ID ==tourney_2018$WTeamID[i]])

    tourney_2018$Average.of.LBlk[i] <- (((s2018$Average.of.LBlk[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LStl[i] <- (((s2018$Average.of.LStl[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LTO[i] <- (((s2018$Average.of.LTO[s2018$Team.ID ==tourney_2018$WTeamID[i]])

    tourney_2018$Average.of.LAst[i] <- (((s2018$Average.of.LAst[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LDR[i] <- (((s2018$Average.of.LDR[s2018$Team.ID ==tourney_2018$WTeamID[i]])

    tourney_2018$Average.of.LOR[i] <- (((s2018$Average.of.LOR[s2018$Team.ID ==tourney_2018$WTeamID[i]])

    tourney_2018$Average.of.LFTA[i] <- (((s2018$Average.of.LFTA[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LFTM[i] <- (((s2018$Average.of.LFTM[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LFGA3[i] <- (((s2018$Average.of.LFGA3[s2018$Team.ID ==tourney_2018$WTeamID[

    tourney_2018$Average.of.LFGM3[i] <- (((s2018$Average.of.LFGM3[s2018$Team.ID ==tourney_2018$WTeamID[

    tourney_2018$Average.of.LFGA[i] <- (((s2018$Average.of.LFGA[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.LFGM[i] <- (((s2018$Average.of.LFGM[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.NumOT[i] <- (((s2018$Average.of.NumOT[s2018$Team.ID ==tourney_2018$WTeamID[

    tourney_2018$Average.of.LScore[i] <- (((s2018$Average.of.LScore[s2018$Team.ID ==tourney_2018$WTeamI

    tourney_2018$Average.of.WScore[i] <- (((s2018$Average.of.WScore[s2018$Team.ID ==tourney_2018$WTeamI

    tourney_2018$Average.of.NumOT.1[i] <- (((s2018$Average.of.NumOT.1[s2018$Team.ID ==tourney_2018$WTea

    tourney_2018$Average.of.WFGM[i] <- (((s2018$Average.of.WFGM[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.WFGA[i] <- (((s2018$Average.of.WFGA[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.WFGM3[i] <- (((s2018$Average.of.WFGM3[s2018$Team.ID ==tourney_2018$WTeamID[

    tourney_2018$Average.of.WFGA3[i] <- (((s2018$Average.of.WFGA3[s2018$Team.ID ==tourney_2018$WTeamID[

    tourney_2018$Average.of.WFTM[i] <- (((s2018$Average.of.WFTM[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.WFTA[i] <- (((s2018$Average.of.WFTA[s2018$Team.ID ==tourney_2018$WTeamID[i]

    tourney_2018$Average.of.WOR[i] <- (((s2018$Average.of.WOR[s2018$Team.ID ==tourney_2018$WTeamID[i]])

    tourney_2018$Average.of.WDR[i] <- (((s2018$Average.of.WDR[s2018$Team.ID ==tourney_2018$WTeamID[i]])
```

```
    tourney_2018$Average.of.WAst[i] <- (((s2018$Average.of.WAst[s2018$Team.ID ==tourney_2018$WTeamID[i]]

    tourney_2018$Average.of.WTO[i] <- (((s2018$Average.of.WTO[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.WStl[i] <- (((s2018$Average.of.WStl[s2018$Team.ID ==tourney_2018$WTeamID[i]]

    tourney_2018$Average.of.WBlk[i] <- (((s2018$Average.of.WBlk[s2018$Team.ID ==tourney_2018$WTeamID[i]]

    tourney_2018$Average.of.WPF[i] <- (((s2018$Average.of.WPF[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.PF[i] <- (((s2018$Average.of.PF[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.Blk[i] <- (((s2018$Average.of.Blk[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.Stl[i] <- (((s2018$Average.of.Stl[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.TO[i] <- (((s2018$Average.of.TO[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.Ast[i] <- (((s2018$Average.of.Ast[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.DR[i] <- (((s2018$Average.of.DR[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.OR[i] <- (((s2018$Average.of.OR[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.FTA[i] <- (((s2018$Average.of.FTA[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.FTM[i] <- (((s2018$Average.of.FTM[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.FGA3[i] <- (((s2018$Average.of.FGA3[s2018$Team.ID ==tourney_2018$WTeamID[i]]

    tourney_2018$Average.of.FGM3[i] <- (((s2018$Average.of.FGM3[s2018$Team.ID ==tourney_2018$WTeamID[i]]

    tourney_2018$Average.of.FGA[i] <- (((s2018$Average.of.FGA[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.FGM[i] <- (((s2018$Average.of.FGM[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$Average.of.PPG[i] <- (((s2018$Average.of.PPG[s2018$Team.ID ==tourney_2018$WTeamID[i]])-

    tourney_2018$W.Per[i] <- (((s2018$W.Per[s2018$Team.ID ==tourney_2018$WTeamID[i]])- (s2018$W.Per[s20

}
tourney_2018$W <- 1
```