march_madness_2021

Jared Klug

Introduction

In this project, I have gotten data from Kaggle, which contains extensive data of regular season games, and conferences for the NCAA Basketball seasons from 2003-2021. Using this data, I would like to predict the potential winners of the upcoming games in the 2021 March Madness Tournament.

I am exploring the following questions:

- Can I accurately predict the score differential of college basketball games?
- Using past team season data, can I accurately predict which team will win a specific game of basketball?
- Can I predict the winners of the NCAA March Madness Tournament games with more than 50% accuracy?

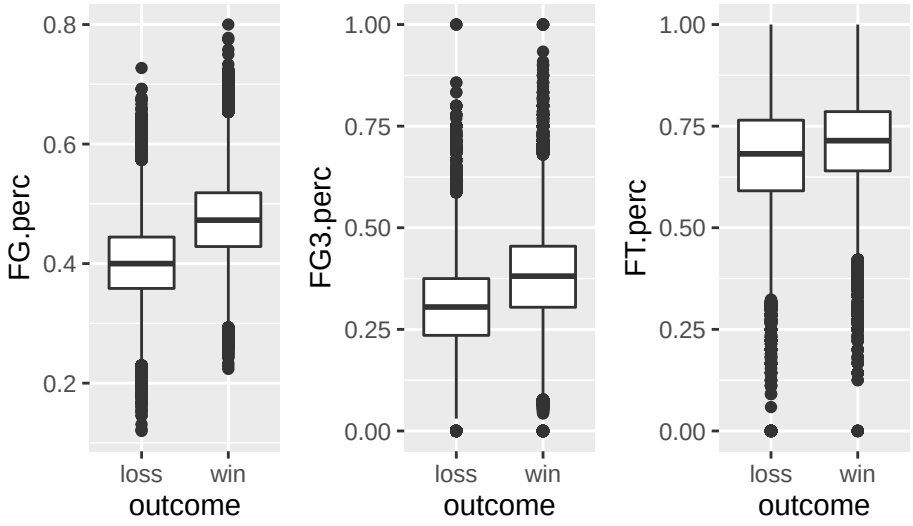
Using the "MRegularSeasonDetailedResults.csv" data from the extensive folder of information, the dataset from this file is 92832 rows with 34 columns. Each row corresponds to 1 basketball game from the 2003 season to 2020 season. Each row has data on which teams played, final scores of each team, and overall team stats from the game for each team such as number of shots attempted and made, penalties, rebounds, etc.

In order to build a model to output point differential of a game, I would first need to compute the point differential of each game, as well as change column names from "winning" and "losing" team stats so that it would not be confusing when inputting future games' team names.

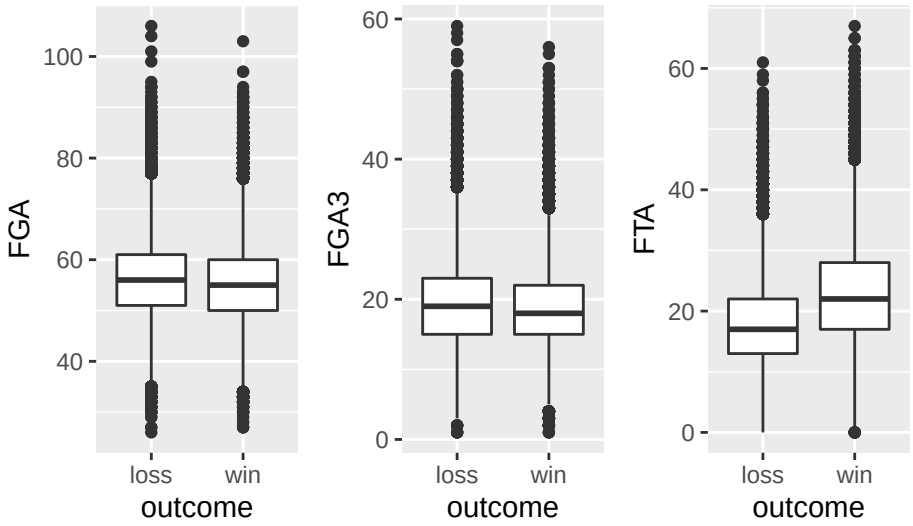
For exploratory data analysis, the data was manipulated so that instead of having 1 game with two team data, each row will correspond to 1 team and a game they played during a specified season and day. Using this data, it will be easier to explore factors that could lead to team wins or losses.

Exploratory Data Analysis

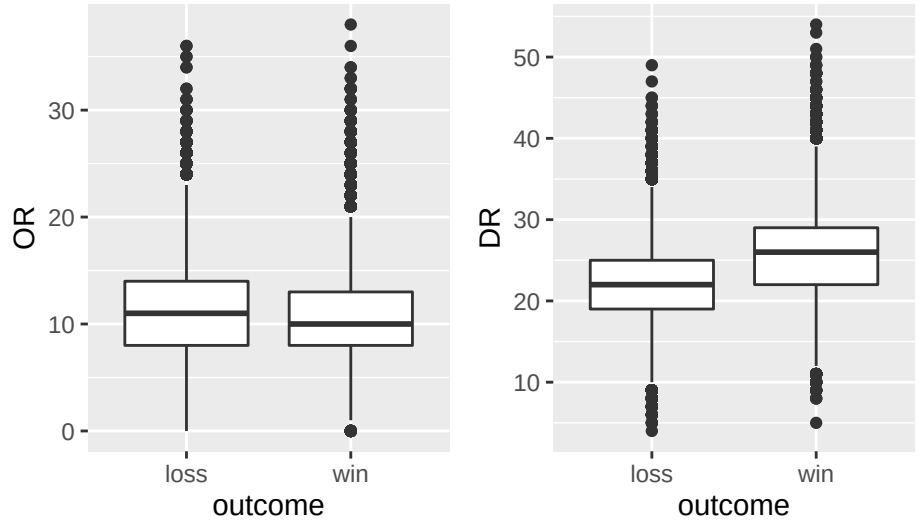
In a game of basketball obviously the team that scores the most points wins, so I did not think it would be necessary to compare the outcomes of games based on how many field goals, or 3-pointers, or free throws a team makes, because it is obviously the winning team will have higher stats. Instead, I will look at percentages made for each of those, as well as explore if the number of attempts of field goals, 3-pointers, or free throws has an obvious impact. I will also make a box plot for each of the other factors: offensive rebounds, defensive rebounds, time outs, steals, and blocks.



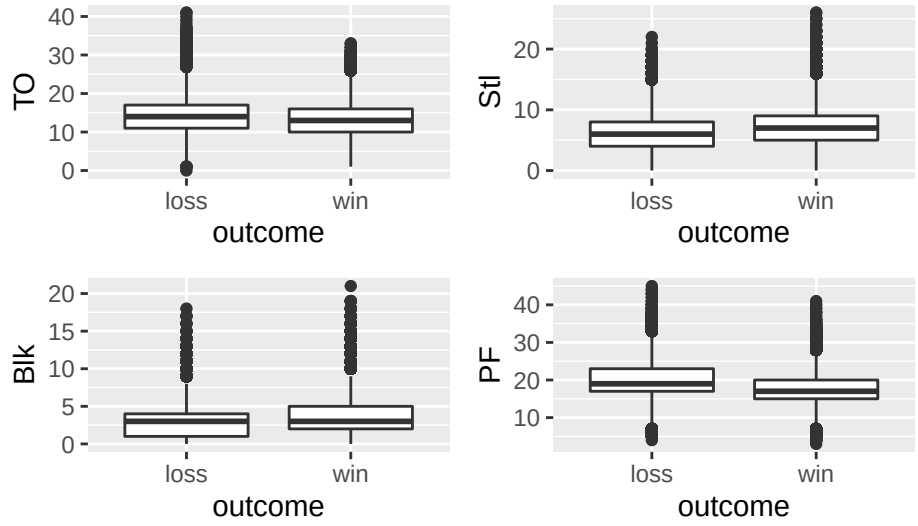
The results from these box plots is no surprise, the team that is consistently scoring more wins the games as indicated by the higher average across the 3 variables for the winning teams.



The results from these variables were very interesting. It shows that for 2 and 3 point shots, the losing team will, on average, have more shots attempted. However these averages are extremely close to each other, so it is hard to pull any conclusive evidence from these box plots. As for free throws, the winning team seems to have more free throws attempted, which makes sense as they're are likely being fouled more given more attempts at making extra points.



For offensive and defensive rebounds, we see opposite results. Interestingly the losing team will have more offensive rebounds. I can speculate that this is because they're missing shots and recovering the ball more. As for defensive rebounds, the winning team will typically have more defensive rebounds.



I found it very interesting that the losing team will have more time-out calls on average. It makes sense that the team that wins will typically have more steals, therefore gaining a chance to score while taking away a chance for the opponents to score. Blocks result is very interesting as the mean is almost right on top of each other for the winning and losing team, but the spread around the mean is typically greater for the winning team than the losing team. We can expect some small impact for blocking for the winning team. For personal fouls, it is intuitive that the team that is causing more fouls will likely lose as they give the opposing team more attempts to score.

Models: Ridge, Lasso, and Elastic Net

I will create 3 models that are well-known for the ability to select variables that are important. I am using these variable selection models because I believe there are variables that are provide unnecessary information

to the point differential outcome.

The tuning parameters for each model will be selected by repeated cross-validation, and the parameters with the smallest RMSE will be the chosen tune for the model.

The predictor variables for all models include the following stats from each game from the 2003 to 2020 season: field goals made and attempted, 3-pointers made and attempted, free throws made and attempted, offensive rebounds, defensive rebounds, assists, timeouts, steals, blocks, and personal fouls for each team. The output value is estimated point differential of the game (T1 score - T2 score). This output inherently carries the estimation of who will win the game as well.

Ridge

```
##    alpha    lambda
## 34     0 0.3832056
```

Lasso

```
##    alpha     lambda
## 19     1 0.06105877
```

Elastic Net

```
##         alpha     lambda
## 419 0.5714286 0.06105877
```

```
##
## Call:
## summary.resamples(object = resamp)
##
## Models: enet, lasso, ridge
## Number of resamples: 50
##
## MAE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  0.2029393 0.2053545 0.2062744 0.2064518 0.2077258 0.2097972    0
## lasso 0.2101013 0.2122817 0.2129844 0.2133457 0.2143474 0.2174918    0
## ridge 1.0392369 1.0514555 1.0564033 1.0577201 1.0647153 1.0775693    0
##
## RMSE
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  0.2628587 0.2669591 0.2687630 0.2686562 0.2705211 0.2751236    0
## lasso 0.2717422 0.2770691 0.2790036 0.2791613 0.2808093 0.2856864    0
## ridge 1.3262047 1.3374382 1.3452174 1.3472911 1.3586527 1.3663550    0
##
## Rsquared
##            Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
## enet  0.9997252 0.9997351 0.9997407 0.9997409 0.9997461 0.9997666    0
## lasso 0.9998070 0.9998200 0.9998267 0.9998260 0.9998297 0.9998437    0
## ridge 0.9830379 0.9835610 0.9838637 0.9838727 0.9841870 0.9848835    0
```

Table 1: RMSE Based on the 20% Partitioned Training Data

model	rmse
enet	0.0737674
lasso	0.0795314
ridge	1.8398809

Based off the resamples, the lasso and enet performances were very similar, while ridge regression had a very high RMSE relative to the other models. Enet still had the best performance as far as predictions, however lasso fit the data very marginally better according to the R-squared means.

When comparing the model's RMSE from the 20% of data partitioned to be the testing data, we see very similar results as what we saw from the resamples. Because the elastic net model had the best prediction performance, it was the model I have chosen to try and make the predictions of the NCAA March Madness Tournament.

In order to make these predictions, I first have to generate the input data, which is every single possible match-up of the tournament, as well as impute team stats for the model to use. Because the training data was able to use stats from those games that were recorded a posteriori. I chose to use average team stats from the 2020 regular season as a comparison. My concept is to expect that every game these teams will play in the tournament, they will have an average performance relative to their regular season.

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  0.183944725
## T1_FGM       1.907061095
## T1_FGA       .
## T1_FGM3      0.960808440
## T1_FGA3      .
## T1_FTM       0.933594642
## T1_FTA       0.007083194
## T1_OR        .
## T1_DR        0.040921069
## T1_Ast       0.017857613
## T1_TO       -0.032728209
## T1_Stl       0.003275752
## T1_Blk       .
## T1_PF        .
## T2_FGM      -1.908316253
## T2_FGA       .
## T2_FGM3     -0.963043880
## T2_FGA3      .
## T2_FTM      -0.940372364
## T2_FTA      -0.005710494
## T2_OR        .
## T2_DR       -0.038536269
## T2_Ast      -0.009110885
## T2_TO        0.037403469
## T2_Stl       .
## T2_Blk       .
## T2_PF        .
```

```
##   T1_name  T2_name MM21_pred
```

```
## 1 Gonzaga   Baylor 16.098436
## 2 Gonzaga Illinois 16.422389
## 3 Gonzaga Michigan 12.185137
## 4 Gonzaga  Alabama  5.688844
## 5 Gonzaga  Ohio St 15.148605
## 6 Gonzaga     Iowa  9.713755
```

Results of Tournament

As of now, there are still 12 games left in the tournament, when the tournament concludes I will report the final performance of the model. As of now, including the first 4 games of the 8 teams which had to play to make it to the official tournament, the model had predicted the correct winning team 37 times out of 54 games which is a 68.5% correct results. Unfortunately, the point differential output has been all over the place, and I will likely calculate an overall RMSE once the tournament concludes for point differential.

Because of the nature of the input data, I'm not shocked that the model would not have an accurate point differential output to what has been happening in the tournament. However, I'm extremely satisfied with how the model performed as far as estimating the winning team. The biggest issue with the input data is that if a team in the tournament has only played lower tier teams during the regular season, they're likely to have inflated game statistics in their favor, making the model more heavily favor the team.

Conclusion

Out of this project, I just wanted to make a model that chose the winning team more than 50% of the time (which would be equivalent to randomly guessing). I am more than shocked with the results of the model and am extremely happy with the results so far. In the future I would like to figure out a method to scale the data to account for opposing team performance. I.e. if the opposing team did not play well, the winning team's stats would be scaled down somewhat.

I was overall surprised by the variables the model chose and how they impacted the estimations. The most influencing variables were the amount of baskets scored, and the amount of attempts were driven down to zero. I was surprised to see that offensive rebound were also driven down to zero and most shocked by how the model placed a negative coefficient for time outs – meaning that a team that is calling a time out will negatively impact their score.