Final Project - Predicting March Madness

Anmol Sapru and Rohit Gunda

Introduction and Data

Motivation

Duke is synonymous with basketball. As Duke students who love Duke Basketball and March Madness, we are interested in performing a statistical analysis on the most thrilling tournament in sports. While watching the 2023 March Madness tournament and the many upsets that came with it, we were motivated to see if we could use statistical methods to predict March Madness winners. Upon scouring sites such as FiveThirtyEight and the KenPom rankings, we were inspired to create models of our own to predict tournament success.

Fundamental Research Question: What variables are important to March Madness success and which outliers over the past 15 years have existed that bring "Madness" to "March?

Packages

```
library(tidyverse)
library(tidymodels)
library(Stat2Data)
library(caret)
library(leaps)
library(MASS)
```

Data

Data was found from the Sports Reference college basketball team stats website and Bart Torvik analytics website, with these general and deeper stats being taken going back to 2008 (excluding the cancelled March Madness of 2020). Both of these sources allow for easy copy + pasting of CSV files with the annual stats for each team and their tournament performance. From here, the data was cleaned in Excel to separate the year and result of each team then joined for all of this data together. With the nature of March Madness being over time, the

data most definitely violates expectations of independence, with similar players, coaches, and more between years, but there would not be a reasonable way to complete such analysis without this independence.

Key Variables

- march_madness - Our key response variable that states the round each team was able to make it to in their tournament. Our overall goal is to predict this variable for teams in the 2023 March Madness Tournament
- ADJOE/ADJDE - Points scored per 100 possessions on offense/defense, adjusted for opponent strength and game location
- TOR/TORD - Turnovers committed/forced per game on offense/defense
- ADJT - Estimate possessions per game a team would have against the average tempo
- EFG% - Field goal percentage adjusted for value of baskets scored

Description of Data Cleaning

For further cleaning, the non-March Madness teams added in the join were removed and simple variable names and values were cleaned up to be easier to work with. First we joined our original `cbb` dataset with `background`, both of which we got from Bart Torvik's analytics website. The latter dataset contained details about each team's performance in the tournament for the relevant years. The observations of teams that did not make the Round of 64 for their tournament were then removed to focus on further prediction.

We also abbreviated each of the outcomes in the `march_madness` variable. Then, we joined the `cbb` and the `sportsreference` datasets by team and year to aggregate all of the statistics and data that were interested in analyzing. For the round each team made it to, the values were shortened to a shorter form (i.e. F4 instead of Final Four). Each round was split up to its own variable and dataset which showed the success of each team in each round (e.g. win or loss in the Round of 32). Finally, we created the `mm_WINS` variable which tracks how many wins a given team had in a tournament year. We deduced this from the outcome from the `march_madness` variable, and applied it to our new variable. This helped us with our EDA as we were able to see how many wins each conference/team had in any given tournament.

Exploratory Data Analysis

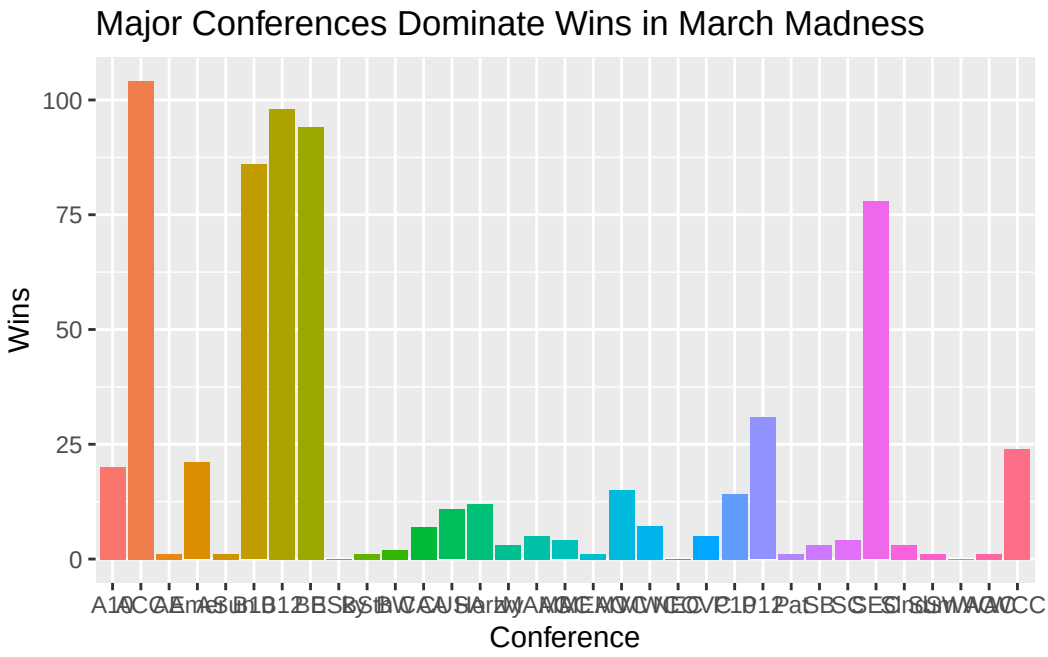
MAYBE ADD ONE MORE EDA

```
cbb |>
  mutate(mm_WINS = case_when(march_madness == "R64" ~ 0, march_madness =="R32" ~ 1,
```

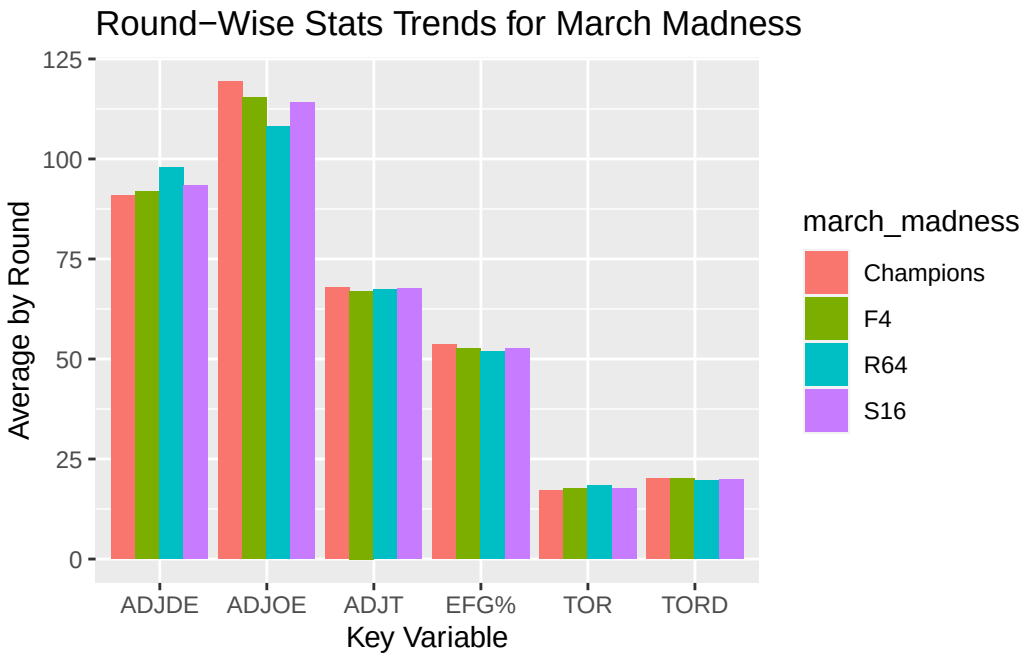
```
                              march_madness == "E16" ~ 2, march_madness == "E8" ~ 3,
                              march_madness == "F4" ~ 4, march_madness == "2ND" ~ 5,
                              march_madness == "Champions" ~ 6, TRUE ~ 0))
```

```
# A tibble: 896 x 38
      RK TEAM      G.x  WINS LOSSES ADJOE ADJDE `EFG%` `EFGD%`   TOR  TORD   ORB
   <dbl> <chr>   <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl> <dbl> <dbl>
 1   265 Pacific    28     9     19  97   106.    46.5    51.1  18.8  20.4  29.5
 2   291 North ~    29     9     20  93.3 104.    47.4    49.9  21.6  22.2  29.3
 3   131 Drexel     19    12      7 106.  104.    54.3    49.6  19.6  16.4  29.4
 4   142 Iona       17    12      5 101.   98.8   51.2    45.6  22.5  19.6  33.5
 5   194 Cal Po~    32    12     20 103.  105     46      49.3  14.7  18.1  30.9
 6   229 James ~    32    12     20 100.  106.    49.2    49.3  19.4  20.3  29.1
 7    56 George~    25    13     12 107.   93.7   49.6    48    21.7  16.2  32.4
 8    71 Villan~    32    13     19 109.   98.1   46.5    47.5  20.4  17.2  37.6
 9   156 Oral R~    23    13     10 108.  107.    53.6    50.4  15.7  18.2  23.2
10   235 Northw~    29    13     16  95.5 101.    47.2    47.7  20.5  22.7  31.9
# ... with 886 more rows, and 26 more variables: DRB <dbl>, FTR <dbl>,
#   FTRD <dbl>, `2P%` <dbl>, `2P%D` <dbl>, `3P%` <dbl>, `3P%D` <dbl>,
#   `3PR` <dbl>, `3PRD` <dbl>, `ADJ T` <dbl>, YEAR <dbl>, march_madness <chr>,
#   Conference <chr>, G.y <dbl>, `Overall W-L%` <dbl>, `Overall SRS` <dbl>,
#   `Overall SOS` <dbl>, `Conf. W-L%` <dbl>, `Home W-L%` <dbl>,
#   `Away W-L%` <dbl>, `AVG PPG` <dbl>, `AVG DPPG` <dbl>, `AVG PD` <dbl>,
#   `AST/TOV` <dbl>, `PF/G` <dbl>, mm_WINS <dbl>
```

```
ggplot(cbb, aes(x = Conference, y = mm_WINS, fill = Conference)) +
  geom_col() +
  theme(legend.position = "none") +
  labs(
    x = "Conference",
    y = "Wins",
    title = "Major Conferences Dominate Wins in March Madness"
  )
```




```
cbb |>
  subset(select = c(ADJOE, ADJDE, TOR, TORD, `ADJ T`, `EFG%`, march_madness)) |>
  group_by(march_madness) |>
  summarize(ADJOE = mean(ADJOE),
            ADJDE = mean(ADJDE),
            TOR = mean(TOR),
            TORD = mean(TORD),
            ADJT = mean(`ADJ T`),
            `EFG%` = mean(`EFG%`)) |>
  pivot_longer(cols = c(ADJOE, ADJDE, TOR, TORD, ADJT, `EFG%`)) |>
  subset(march_madness %in% c("R64", "S16", "F4", "Champions")) |>
  ggplot(aes(x = name, y = value, fill = march_madness)) +
  geom_col(position = "dodge") +
  labs(
    title = "Round-Wise Stats Trends for March Madness",
    x = "Key Variable",
    y = "Average by Round",
    legend = "March Madness Round"
  )
```



Methodology

Round-by-Round Logistic Regression

The following is an example of the regression that was ran on all of the Round of 64 teams to create a regression that predicts winners (round_64 = TRUE) against losers (round_64 = FALSE) for the round. We put every variable from our dataset into the logisitic regression, with the interaction terms based on variables that would be considered to be differential statistics (i.e. 3P% and 3P%D interact for 3-Pointer % Differential). The number of variables was then reduced to those considered appropriate using stepAIC works to attempt to limit the overfitting with the data. The stepAIC function was implemented starting at the upper model and going both directions. There is a potential problem in using glm fits with a variable scale, as in that case the deviance is not simply related to the maximized log-likelihood.

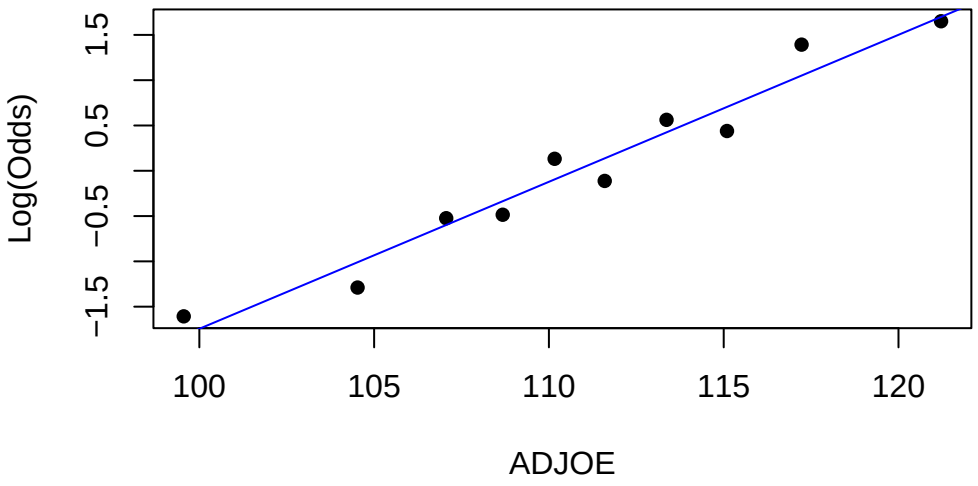
We chose to do a logistic model because our outcome variable is whether they had won in the respective round or not. For example round_64 measures whether the team won in the Round of 64. The outcome is either a win or loss, so a logistic regression model serves our analysis best. We did this for the first three rounds of the tournament – we avoided Elite Eight to the National Championship game as there simply was not enough data to create a viable predictive model (Northern Kentucky was predicted to win in the Final Four).

```r
round_64 <- na.omit(round_64)

round_64_max <- glm(round_64 ~ G.x + ADJOE + ADJDE + `EFG%` + `EFGD%` + TOR +
                      TORD + ORB + DRB + FTR + FTRD + `2P%` + `2P%D` + `3P%` +
                      `3P%D` + `3PR` + `3PRD` + `ADJ T` +`Conf. W-L%` + `Home W-L%` +
                      `Away W-L%` + `AVG PPG` + `AVG DPPG` + `AVG PD` +
                      `AST/TOV` + `PF/G` + ADJOE*ADJDE + `EFG%`*`EFGD%` + TOR*TORD +
                      ORB*DRB + FTR*FTRD + `2P%`*`2P%D` + `3P%`*`3P%D` + `3PR`*`3PRD` +
                      `2P%`*`3P%` + `2P%D`*`3P%D` + `AVG PPG`*`AVG DPPG`,
                      data = round_64,
                      family = "binomial")

round_64_min <- glm(round_64 ~ 1,
                    data = round_64,
                    family = "binomial")

round_64_model <- stepAIC(round_64_max,
        scope = list(lower = round_64_min, upper = round_64_max),
        data = round_64, direction = "both")
```

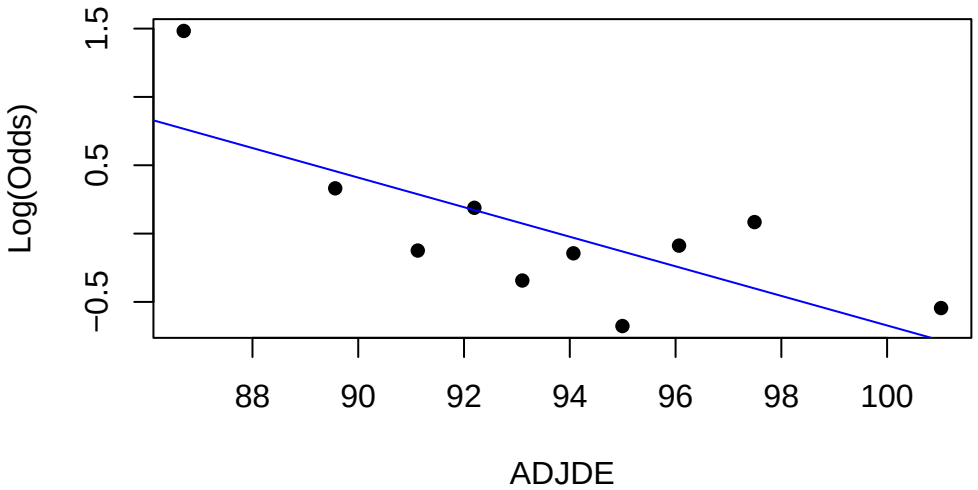
Assumptions

As discussed earlier, the independence assumption for the data cannot be well-assumed. There is also a linearity assumption that is based on there being a linear relationship between the log-odds of the response and the predictors.

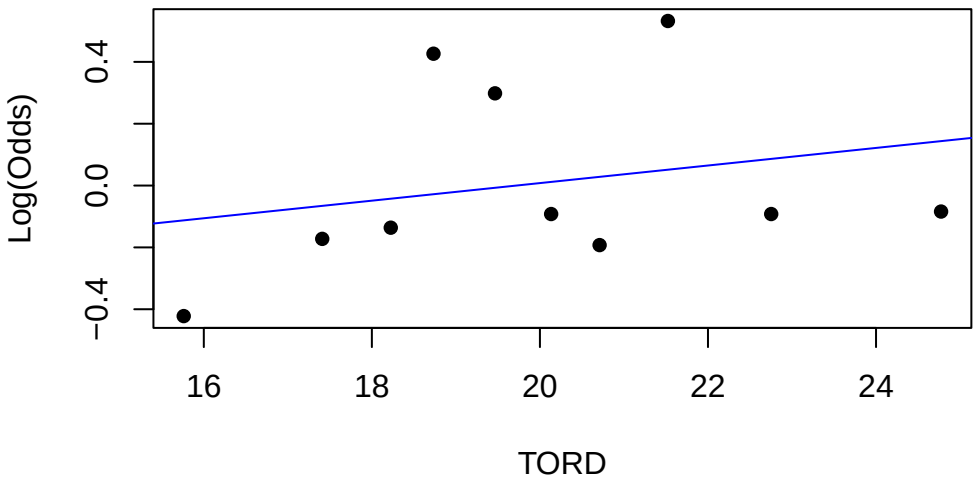
```r
emplogitplot1(round_64 ~ ADJOE, data = round_64, ngroups = 10)
```




```
emplogitplot1(round_32 ~ ADJDE, data = round_32, ngroups = 10)
```



```
emplogitplot1(sweet_sixteen ~ TORD, data = sweet_sixteen, ngroups = 10)
```



As can be seen above with just a few of the variables from each of the model, the linearity assumption does seem like it can be quite well assumed. We checked most of the variables and saw similar linearity assumptions to be well-assumed, but for the sake of space are just showing this small subset.

Overall Ordinal Regression

We chose to complete an ordinal regression model as each of the progressing rounds of the tournament are in an ordinal progession. The variables used in the model were those that consistently showed up in the Round of 64, Round of 32, and Sweet Sixteen models after running the StepAIC function, meaning they can confidently be thought as meaningful predictors of success in the rounds of March Madness.

```
cbb$march_madness <- fct_relevel(cbb$march_madness,
                                 c("R64", "R32", "S16", "E8",
                                   "F4", "2ND", "Champions"))

mmb_ord_model <- polr(march_madness ~ ADJOE + ADJDE + TORD +
```

```
                          FTRD + `2P%D` + `3P%D` + `AVG PPG` +
                          `AVG DPPG` + `PF/G`, data = cbb)

summary(mmb_ord_model)
```

```
Re-fitting to get Hessian

Call:
polr(formula = march_madness ~ ADJOE + ADJDE + TORD + FTRD +
    `2P%D` + `3P%D` + `AVG PPG` + `AVG DPPG` + `PF/G`, data = cbb)

Coefficients:
               Value Std. Error t value
ADJOE       0.165964    0.01579 10.5136
ADJDE      -0.228849    0.02414 -9.4790
TORD        0.035456    0.03449  1.0279
FTRD        0.067047    0.02809  2.3865
`2P%D`      0.035198    0.03971  0.8864
`3P%D`      0.023942    0.04088  0.5857
`AVG PPG`   0.052121    0.02645  1.9706
`AVG DPPG` -0.008595    0.03136 -0.2741
`PF/G`     -0.311339    0.09341 -3.3329

Intercepts:
              Value     Std. Error t value
R64|R32         -0.2630    0.0010  -258.8674
R32|S16          1.3164    0.0963    13.6647
S16|E8           2.4566    0.1334    18.4125
E8|F4            3.3948    0.1726    19.6637
F4|2ND           4.2343    0.2247    18.8467
2ND|Champions    5.0224    0.2976    16.8770

Residual Deviance: 2011.75
AIC: 2041.75
```

The assumption required for an ordinal regression is the proportional odds assumption, which applied to this situation is the idea that the same conditional relationship with odds of making it from one round to the next no matter which round that is. This is reasonable due to the fact the proportion of teams that makes it on from one round to the next stays at a constant 50 percent, no matter which round they are participating in.

Results

```
`2023sr` <- read_csv("data/2023sportsreference.csv")
```

```
Rows: 363 Columns: 13
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (1): School
dbl (12): G, Overall W-L%, Overall SRS, Overall SOS, Conf. W-L%, Home W-L%, ...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
`2023analytics` <- read_csv("data/2023torvik.csv")
```

```
Rows: 363 Columns: 22
-- Column specification --------------------------------------------------------
Delimiter: ","
chr  (1): TEAM
dbl (21): RK, G, WINS, LOSSES, ADJOE, ADJDE, EFG%, EFGD%, TOR, TORD, ORB, DR...

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
`2023stats` <- left_join(`2023analytics`, `2023sr`, by = c("TEAM" = "School"))

`2023teams` <- read_csv("data/2023teams.csv")
```

```
Rows: 64 Columns: 1
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (1): TEAM

i Use `spec()` to retrieve the full column specification for this data.
i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
`2023stats` <- `2023stats` |>
  filter(`2023stats`$TEAM %in% `2023teams`$TEAM)

tibble(predict(round_64_model, `2023stats`)) |>
  mutate(rank = seq(1:64)) |>
  left_join(mutate(`2023stats`, rank = seq(1:64))) |>
  arrange(desc(predict(round_64_model, `2023stats`)))
```

```
Joining, by = "rank"
```

```
# A tibble: 64 x 36
   predi~1  rank    RK TEAM    G.x  WINS LOSSES ADJOE ADJDE `EFG%` `EFGD%`   TOR
     <dbl> <int> <dbl> <chr> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl>
 1    3.59     1     1 Hous~    34    31      3  117.  88     52.7    42.5  15.3
 2    3.41     3     3 UCLA     34    29      5  113.  87.4   50.9    46.8  15.3
 3    2.50    12    12 San ~    32    26      6  111.  90.1   50.1    47.5  17.6
 4    2.48     2     2 Alab~    34    29      5  115.  88.3   52.7    41.5  19
 5    2.24     6     6 Purd~    34    29      5  118.  92.6   52.2    47.2  17
 6    2.23    11    11 Marq~    34    28      6  119.  96.1   56      51.1  15.2
 7    2.08    13    13 Kans~    34    27      7  113.  91.5   52.4    47.1  17.5
 8    1.96     9     9 Texas    34    26      8  115.  91.6   52.7    47.8  16.5
 9    1.89     4     4 Tenn~    33    23     10  111.  86.2   50.3    42.4  18.1
10    1.74     5     5 Conn~    33    25      8  119.  92.5   53.5    45.5  18.9
# ... with 54 more rows, 24 more variables: TORD <dbl>, ORB <dbl>, DRB <dbl>,
#   FTR <dbl>, FTRD <dbl>, `2P%` <dbl>, `2P%D` <dbl>, `3P%` <dbl>,
#   `3P%D` <dbl>, `3PR` <dbl>, `3PRD` <dbl>, `ADJ T` <dbl>, G.y <dbl>,
#   `Overall W-L%` <dbl>, `Overall SRS` <dbl>, `Overall SOS` <dbl>,
#   `Conf. W-L%` <dbl>, `Home W-L%` <dbl>, `Away W-L%` <dbl>, `AVG PPG` <dbl>,
#   `AVG DPPG` <dbl>, `AVG PD` <dbl>, `AST/TOV` <dbl>, `PF/G` <dbl>, and
#   abbreviated variable name 1: `predict(round_64_model, \`2023stats\`)`
```

```
tibble(predict(round_32_model, `2023stats`)) |>
  mutate(rank = seq(1:64)) |>
  left_join(mutate(`2023stats`, rank = seq(1:64))) |>
  arrange(desc(predict(round_32_model, `2023stats`)))
```

```
Joining, by = "rank"
```

```
# A tibble: 64 x 36
```

```
   predi~1  rank    RK TEAM    G.x  WINS LOSSES ADJOE ADJDE `EFG%` `EFGD%`   TOR
     <dbl> <int> <dbl> <chr> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl>
 1   2.18      1     1 Hous~    34    31      3  117.  88     52.7    42.5  15.3
 2   1.50      3     3 UCLA     34    29      5  113.  87.4   50.9    46.8  15.3
 3   1.27      2     2 Alab~    34    29      5  115.  88.3   52.7    41.5  19
 4   1.03     10    10 Gonz~    32    27      5  123.  98.6   58.5    51.7  14.6
 5   1.00     12    12 San ~    32    26      6  111.  90.1   50.1    47.5  17.6
 6   0.910     6     6 Purd~    34    29      5  118.  92.6   52.2    47.2  17
 7   0.880    38    42 Penn~    35    22     13  115.  99.9   55.5    49.2  13.7
 8   0.853    15    15 Bayl~    32    22     10  121.  99.5   53.1    51.4  18.1
 9   0.848     4     4 Tenn~    33    23     10  111.  86.2   50.3    42.4  18.1
10   0.779    14    14 Crei~    33    21     12  114.  92.8   54.3    47.3  16.6
# ... with 54 more rows, 24 more variables: TORD <dbl>, ORB <dbl>, DRB <dbl>,
#   FTR <dbl>, FTRD <dbl>, `2P%` <dbl>, `2P%D` <dbl>, `3P%` <dbl>,
#   `3P%D` <dbl>, `3PR` <dbl>, `3PRD` <dbl>, `ADJ T` <dbl>, G.y <dbl>,
#   `Overall W-L%` <dbl>, `Overall SRS` <dbl>, `Overall SOS` <dbl>,
#   `Conf. W-L%` <dbl>, `Home W-L%` <dbl>, `Away W-L%` <dbl>, `AVG PPG` <dbl>,
#   `AVG DPPG` <dbl>, `AVG PD` <dbl>, `AST/TOV` <dbl>, `PF/G` <dbl>, and
#   abbreviated variable name 1: `predict(round_32_model, \`2023stats\`)`
```

```
tibble(predict(sweet_sixteen_model, `2023stats`)) |>
  mutate(rank = seq(1:64)) |>
  left_join(mutate(`2023stats`, rank = seq(1:64))) |>
  arrange(desc(predict(sweet_sixteen_model, `2023stats`)))
```

```
Joining, by = "rank"
```

```
# A tibble: 64 x 36
   predi~1  rank    RK TEAM    G.x  WINS LOSSES ADJOE ADJDE `EFG%` `EFGD%`   TOR
     <dbl> <int> <dbl> <chr> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl>
 1   2.03      1     1 Hous~    34    31      3  117.  88     52.7    42.5  15.3
 2   2.01      6     6 Purd~    34    29      5  118.  92.6   52.2    47.2  17
 3   1.09     22    22 Texa~    34    25      9  113.  94.8   49      47.9  18.3
 4   1.03      5     5 Conn~    33    25      8  119.  92.5   53.5    45.5  18.9
 5   0.961    38    42 Penn~    35    22     13  115.  99.9   55.5    49.2  13.7
 6   0.674    23    23 Aubu~    32    20     12  111.  93.2   49.6    45.6  18.1
 7   0.624    13    13 Kans~    34    27      7  113.  91.5   52.4    47.1  17.5
 8   0.532     2     2 Alab~    34    29      5  115.  88.3   52.7    41.5  19
 9   0.467    26    26 TCU      33    21     12  110.  93.4   50.2    47.8  17.1
10   0.449    12    12 San ~    32    26      6  111.  90.1   50.1    47.5  17.6
# ... with 54 more rows, 24 more variables: TORD <dbl>, ORB <dbl>, DRB <dbl>,
```

```
#   FTR <dbl>, FTRD <dbl>, `2P%` <dbl>, `2P%D` <dbl>, `3P%` <dbl>,
#   `3P%D` <dbl>, `3PR` <dbl>, `3PRD` <dbl>, `ADJ T` <dbl>, G.y <dbl>,
#   `Overall W-L%` <dbl>, `Overall SRS` <dbl>, `Overall SOS` <dbl>,
#   `Conf. W-L%` <dbl>, `Home W-L%` <dbl>, `Away W-L%` <dbl>, `AVG PPG` <dbl>,
#   `AVG DPPG` <dbl>, `AVG PD` <dbl>, `AST/TOV` <dbl>, `PF/G` <dbl>, and
#   abbreviated variable name ...
```

```
tibble(predict(mmb_ord_model, `2023stats`)) |>
  mutate(rank = seq(1:64)) |>
  left_join(mutate(`2023stats`, rank = seq(1:64))) |>
  arrange(predict(mmb_ord_model, `2023stats`, type = "probs"))
```

```
Joining, by = "rank"
```

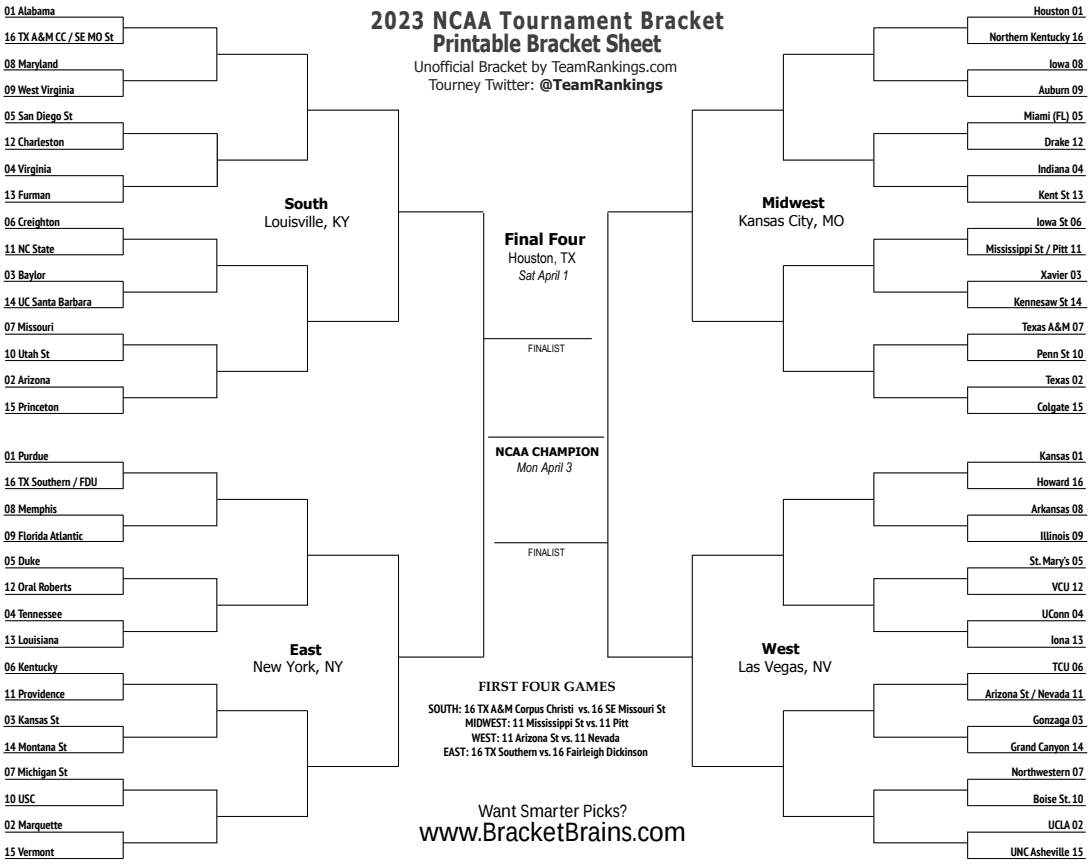
```
# A tibble: 64 x 36
   predi~1  rank    RK TEAM    G.x  WINS LOSSES ADJOE ADJDE `EFG%` `EFGD%`   TOR
   <fct>   <int> <dbl> <chr> <dbl> <dbl>  <dbl> <dbl> <dbl>  <dbl>   <dbl> <dbl>
 1 E8          1     1 Hous~    34    31      3  117.  88     52.7    42.5  15.3
 2 S16         3     3 UCLA     34    29      5  113.  87.4   50.9    46.8  15.3
 3 S16         5     5 Conn~    33    25      8  119.  92.5   53.5    45.5  18.9
 4 S16         4     4 Tenn~    33    23     10  111.  86.2   50.3    42.4  18.1
 5 R32         6     6 Purd~    34    29      5  118.  92.6   52.2    47.2  17
 6 R32         2     2 Alab~    34    29      5  115.  88.3   52.7    41.5  19
 7 R32         9     9 Texas    34    26      8  115.  91.6   52.7    47.8  16.5
 8 R32        11    11 Marq~    34    28      6  119.  96.1   56      51.1  15.2
 9 R32        10    10 Gonz~    32    27      5  123.  98.6   58.5    51.7  14.6
10 R32         8     8 Sain~    32    25      7  112.  89.1   52.5    46.7  16.4
# ... with 54 more rows, 24 more variables: TORD <dbl>, ORB <dbl>, DRB <dbl>,
#   FTR <dbl>, FTRD <dbl>, `2P%` <dbl>, `2P%D` <dbl>, `3P%` <dbl>,
#   `3P%D` <dbl>, `3PR` <dbl>, `3PRD` <dbl>, `ADJ T` <dbl>, G.y <dbl>,
#   `Overall W-L%` <dbl>, `Overall SRS` <dbl>, `Overall SOS` <dbl>,
#   `Conf. W-L%` <dbl>, `Home W-L%` <dbl>, `Away W-L%` <dbl>, `AVG PPG` <dbl>,
#   `AVG DPPG` <dbl>, `AVG PD` <dbl>, `AST/TOV` <dbl>, `PF/G` <dbl>, and
#   abbreviated variable name 1: `predict(mmb_ord_model, \`2023stats\`)`
```

The slope coefficient for average points per game for the Sweet Sixteen logistic model is 1.036. Holding all other variables constant, for every one point increase in average points per game, the odds of winning in the Sweet Sixteen is associated with a e^1.036 = 2.818 multiplicative factor on the odds of winning in this round.

The slope coefficient for average points per game for the ordinal regression model is 0.052121. For every one point increase in average points per game, the odds of moving onto the next

round (the odds of winning in a given round) is predicted to be multiplied by e^0.052121 = 1.054 - holding all other predictors in the model constant.

Predictive Power via 2023 March Madness Predictions



The ordinal model was able to predict 23/32 and 12/16 of the teams to make it on to the Round of 32 and Sweet Sixteen, respectively. For comparison, this bracket predicted around the average score of my brackets, which were likely better than average since I grew up a UConn fan and often had them going far. The successes in the earlier rounds fell off onto later rounds, of course due to chance, but possibly also due to quite low levels of data for prediction of success.

Key Results

An unsurprising key result found was the randomness of the tournament, with a simple wrong decision in the earlier rounds multiplying into the lack of success for point totals in the later

rounds. Based on the variables that ended up being chosen following the StepAIC function, summarizing variables that consider games played and strength of opponents over the season are stronger predictors than some of the overall percentage stats about schedule success (e.g. Home/Away Win %). From the variables that consistently showed up in the logisitic models and were thus used in the ordinal model, there was a higher emphasis on defensive stats compared to those of the offense.

Discussion

Learned about research question

Analysis limitations and improvements

Reliability and validity

appropriateness of the statistical analysis

Outliers and Randomness of March Madness

HYPOTHESIS TESTS

COOK'S DISTANCE

Ideas for Future Work