

# Final Project - Predicting March Madness

Anmol Sapru and Rohit Gunda

## Introduction and Data

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 8 years have existed that bring the “Madness” to the tournament?

The following packages were used:

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

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). 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.

Key Variables

- ADJOE: How much a team scores per 100 possessions on offense, adjusted for opponent strength and game location (home or away)
- ADJDE: How many points a team allows per 100 possessions on defense, adjusted for opponent strength and game location (home or away)

- TOR: Turnovers committed per game on offense
- TORD: Turnovers forced per game on defense
- ADJT:
- EFG%: Field goal percentage

## EXPLORATORY DATA ANALYSIS

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.

## ADD CLEARER DESCRIPTION OF DATA CLEANING

## Methodology

We determined it would be best to try different regressions, with logistic regressions by round and an ordinal regression—each with their different assumptions to look into. The issue with the first of these is the low levels of data that limit the creation of a model beyond around the Elite Eight. In preparing for the round-by-round logistic regressions we split up the data from the original data sets into multiple data sets for each round, with a TRUE or FALSE value of whether they won their game in that round.

## WHY CHOSE EACH VARIABLE

### 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. Using stepAIC works to attempt to limit the overfitting with the data. This was also done for the Round of 32 and Sweet Sixteen.

### WHY LOGISTIC REG?

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

round_64_max <- glm(round_64 ~ G.x + WINS + LOSSES + 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` + WINS*G.x + LOSSES*G.x +
  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")

```

The following are the regression models that were found from the three rounds of data and regression models run through stepAIC:

## Overall Ordinal Regression

```

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)

```

## ADD ASSESSMENT OF CONDITIONS + DIAGNOSTICS

## 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	3.67	1	1	Hous~	34	31	3	117.	88	52.7	42.5	15.3
2	3.37	3	3	UCLA	34	29	5	113.	87.4	50.9	46.8	15.3
3	2.54	6	6	Purd~	34	29	5	118.	92.6	52.2	47.2	17
4	2.48	12	12	San ~	32	26	6	111.	90.1	50.1	47.5	17.6
5	2.47	2	2	Alab~	34	29	5	115.	88.3	52.7	41.5	19
6	2.10	9	9	Texas	34	26	8	115.	91.6	52.7	47.8	16.5
7	2.07	11	11	Marq~	34	28	6	119.	96.1	56	51.1	15.2
8	2.05	4	4	Tenn~	33	23	10	111.	86.2	50.3	42.4	18.1
9	1.98	13	13	Kans~	34	27	7	113.	91.5	52.4	47.1	17.5
10	1.85	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`)))
```

Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
prediction from a rank-deficient fit may be misleading

Joining, by = "rank"

Warning in predict.lm(object, newdata, se.fit, scale = 1, type = if (type == :  
prediction from a rank-deficient fit may be misleading

# 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	2.04	1	1	Hous~	34	31	3	117.	88	52.7	42.5	15.3
2	1.78	3	3	UCLA	34	29	5	113.	87.4	50.9	46.8	15.3
3	1.67	38	42	Penn~	35	22	13	115.	99.9	55.5	49.2	13.7
4	1.49	2	2	Alab~	34	29	5	115.	88.3	52.7	41.5	19
5	1.03	12	12	San ~	32	26	6	111.	90.1	50.1	47.5	17.6

```

6  0.996    6    6 Purd~    34    29    5  118.  92.6  52.2  47.2  17
7  0.750   16   16 Memp~   34    26    8  114.  94.2  53.2  46.8  18.4
8  0.635   14   14 Crei~   33    21   12  114.  92.8  54.3  47.3  16.6
9  0.603   10   10 Gonz~   32    27    5  123.  98.6  58.5  51.7  14.6
10 0.574   50   78 Oral~   30    26    4  112. 101.  56.1  48.5  13.2
# ... 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  2.47     6    6 Purd~   34    29    5  118.  92.6  52.2  47.2  17
2  1.84     1    1 Hous~   34    31    3  117.  88    52.7  42.5  15.3
3  1.30     5    5 Conn~   33    25    8  119.  92.5  53.5  45.5  18.9
4  1.22    13   13 Kans~   34    27    7  113.  91.5  52.4  47.1  17.5
5  0.880    22   22 Texa~   34    25    9  113.  94.8  49    47.9  18.3
6  0.826    24   24 Kans~   32    23    9  110.  93.1  51.5  47.5  20
7  0.752    21   21 Duke   34    26    8  112.  93.1  51.1  46.3  18.2
8  0.714    12   12 San ~   32    26    6  111.  90.1  50.1  47.5  17.6
9  0.657    38   42 Penn~   35    22   13  115.  99.9  55.5  49.2  13.7
10 0.606    26   26 TCU    33    21   12  110.  93.4  50.2  47.8  17.1
# ... 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(desc(predict(mmb_ord_model, `2023stats`)))
```

Joining, by = "rank"

# A tibble: 64 x 36

	predi~1 <fct>	rank <int>	RK <dbl>	TEAM <chr>	G.x <dbl>	WINS <dbl>	LOSSES <dbl>	ADJOE <dbl>	ADJDE <dbl>	`EFG%` <dbl>	`EFGD%` <dbl>	TOR <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	4	4	Tenn~	33	23	10	111.	86.2	50.3	42.4	18.1
4	S16	5	5	Conn~	33	25	8	119.	92.5	53.5	45.5	18.9
5	R32	2	2	Alab~	34	29	5	115.	88.3	52.7	41.5	19
6	R32	6	6	Purd~	34	29	5	118.	92.6	52.2	47.2	17
7	R32	7	7	Ariz~	34	28	6	120.	95.4	56.8	47.2	18.4
8	R32	8	8	Sain~	32	25	7	112.	89.1	52.5	46.7	16.4
9	R32	9	9	Texas	34	26	8	115.	91.6	52.7	47.8	16.5
10	R32	10	10	Gonz~	32	27	5	123.	98.6	58.5	51.7	14.6

# ... 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(mmb\_ord\_model, `2023stats`))`

INTERPRET SOMETHING - MOST IMPORTANT VARIABLES

ADD ASSUMPTIONS

MODEL'S PREDICTIVE POWER

Outliers and Randomness of March Madness

## Discussion

MISSING ASSUMPTIONS

HYPOTHESIS TESTS

**COOK'S DISTANCE**

**IDEA'S FOR FUTURE WORK**