# Thinking about Baseball

Chase Mathis

## Introduction

When looking at baseball and predicting the results of baseball games, many factors are at play:

- Who is pitching?
- Is the star player hurt?
- Is the baseball team on a hot streak or a cold run?

This is only a few out of many different factors. Sports betting and casino's now don't create lines based off factors, but are lazy and will depend on the wisdom of the masses. They engineer lines to get equal attraction on both sides of the spread and make money off of the margin.

## Importing Packages

```r
library(tidyverse) # data wrangling
library(tidymodels) # modeling
library(baseballr) # api for baseball data
library(knitr) # output
library(patchwork)
library(corrr) # correlation


ggplot2::theme_set(ggplot2::theme_minimal(base_size = 15))
```

## Data

I will retrieve the data from the baseballr package, where we can scrape data from baseball-reference.com. Instead of looking at one team and predicting off of their statistics (may be done later with a more machine learning approach), I will attempt to make a more general model of which factors make the most predictive model for any baseball team.

I randomly sampled six numbers that correspond to the final standings of last year's season. These will be the teams where I get the data.

```
standings <- read_csv("data/standings.csv") %>%
  select(Rk, Tm)
```

```
Rows: 31 Columns: 24
-- Column specification --------------------------------------------------------
Delimiter: ","
chr (14): Tm, pythWL, vEast, vCent, vWest, Inter, Home, Road, ExInn, 1Run, v...
dbl (10): Rk, W, L, W-L%, R, RA, Rdiff, SOS, SRS, Luck

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.
```

```
set.seed(123)

randnums <- sample(1:30, 6, replace=FALSE)

standings %>%
  filter(Rk %in% randnums) %>%
  select(Tm) %>%
  kable(col.names = "Teams Randomly Selected")
```

| Teams Randomly Selected |
|---|
| Tampa Bay Rays |
| St. Louis Cardinals |
| Cincinnati Reds |
| Philadelphia Phillies |
| Detroit Tigers |
| New York Mets |

```r
teams <- standings %>%
  filter(Rk %in% randnums) %>%
  select(Tm) %>%
  pull()

astros0 <- team_results_bref("HOU", 2021) #baseball reference
mets0 <- team_results_bref("NYM", 2021) #baseball reference
marlins0 <- team_results_bref("MIA", 2021) #baseball reference
nationals0 <- team_results_bref("WSN", 2021) #baseball reference
pirates0 <- team_results_bref("PIT", 2021) #baseball reference
diamond0 <- team_results_bref("ARI", 2021) #baseball reference


baseball<-bind_rows(astros0,mets0,marlins0,nationals0,pirates0,diamond0)
```

Our final data frame:

```r
baseball
```

```
-- MLB Team Results data from baseball-reference.com -------- baseballr 1.2.0 --

i Data updated: 2022-05-16 09:50:08 PDT

# A tibble: 972 x 22
      Gm Date      Tm    H_A   Opp   Result     R    RA Inn   Record  Rank GB
   <dbl> <chr>     <chr> <chr> <chr> <chr>  <dbl> <dbl> <chr> <chr>  <dbl> <chr>
 1     1 Thursday~ HOU   A     OAK   W          8     1 ""    1-0        1 Tied
 2     2 Friday, ~ HOU   A     OAK   W          9     5 ""    2-0        1 up 1~
 3     3 Saturday~ HOU   A     OAK   W          9     1 ""    3-0        1 up 1~
 4     4 Sunday, ~ HOU   A     OAK   W          9     2 ""    4-0        1 up 1~
 5     5 Monday, ~ HOU   A     LAA   L          6     7 ""    4-1        1 Tied
 6     6 Tuesday,~ HOU   A     LAA   W          4     2 ""    5-1        1 up 1~
 7     7 Thursday~ HOU   H     OAK   W          6     2 ""    6-1        1 up 1~
 8     8 Friday, ~ HOU   H     OAK   L          2     6 ""    6-2        1 Tied
 9     9 Saturday~ HOU   H     OAK   L          3     7 ""    6-3        1 Tied
10    10 Monday, ~ HOU   H     DET   L          2     6 ""    6-4        2 1.0
# ... with 962 more rows, and 10 more variables: Win <chr>, Loss <chr>,
#   Save <chr>, Time <chr>, `D/N` <chr>, Attendance <dbl>, cLI <dbl>,
#   Streak <dbl>, Orig_Scheduled <chr>, Year <dbl>
```

Given that we don't care how each team individually is predicted, I will remove the `Tm` variable. In addition, the `Orig_Scheduled` has no real values so we can remove that. The `Year` variable has no variance (all 2021), so we can remove that.

In addition, I will input a 9 for blank innings and make it a double; create a variable called `margin` that is the margin of victory or defeat; lastly I will remove some levels of the win or loss variable.

Baseball-reference denotes a walk off win differently than a regular win. To make things simpler, I will remove this distinction

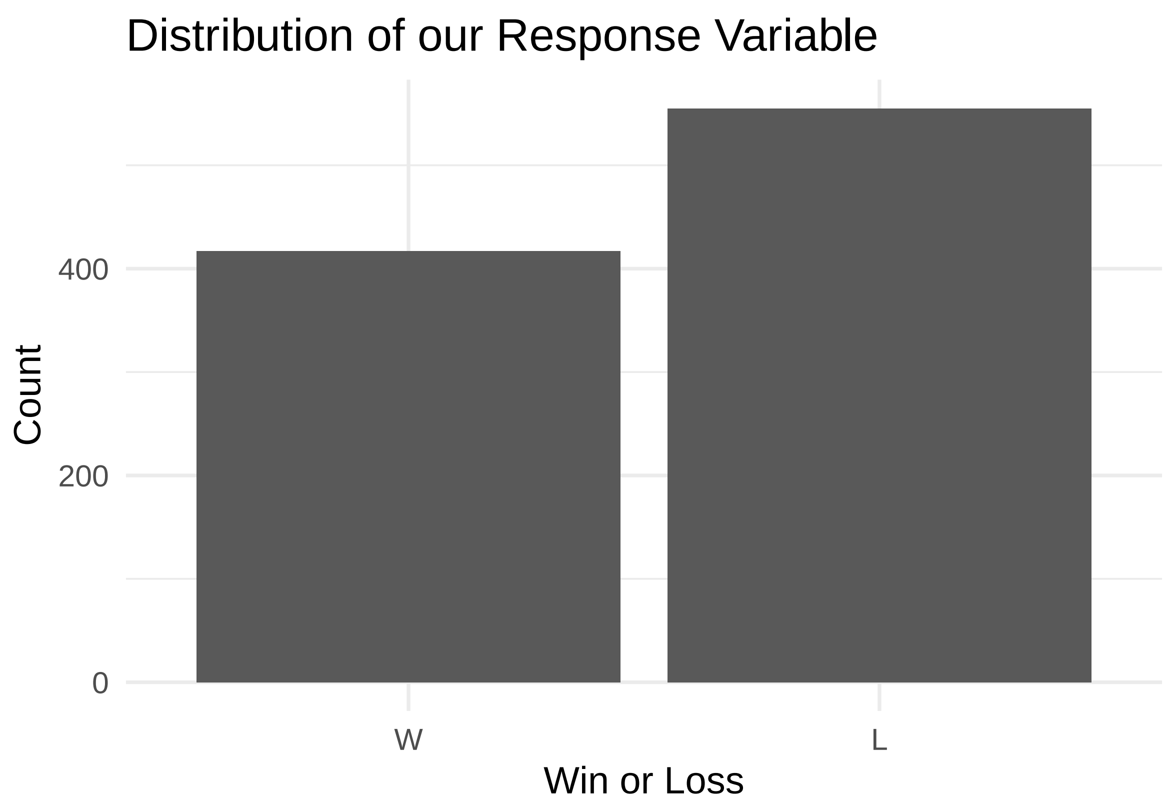
```r
baseball <- baseball %>%
  select(-c(Tm,Orig_Scheduled,Year)) %>%
  mutate(Inn = as.double(if_else(str_detect(Inn, ""), Inn, "9")),
         margin = R-RA,
         Result = if_else(str_detect(Result,"L"), "L", "W"),
         Won = if_else(Result == "W", 1, 0),
         Day = if_else(`D/N` == "D", 1,0),
         across(where(is.character),as_factor))
```

## Exploratory Data Analysis

### Distribution Analysis

```
baseball %>%
  ggplot(aes(x = Result)) +
  geom_histogram(stat = 'count') +
  labs(
    x = "Win or Loss",
    y = "Count",
    title = "Distribution of our Response Variable"
  )
```

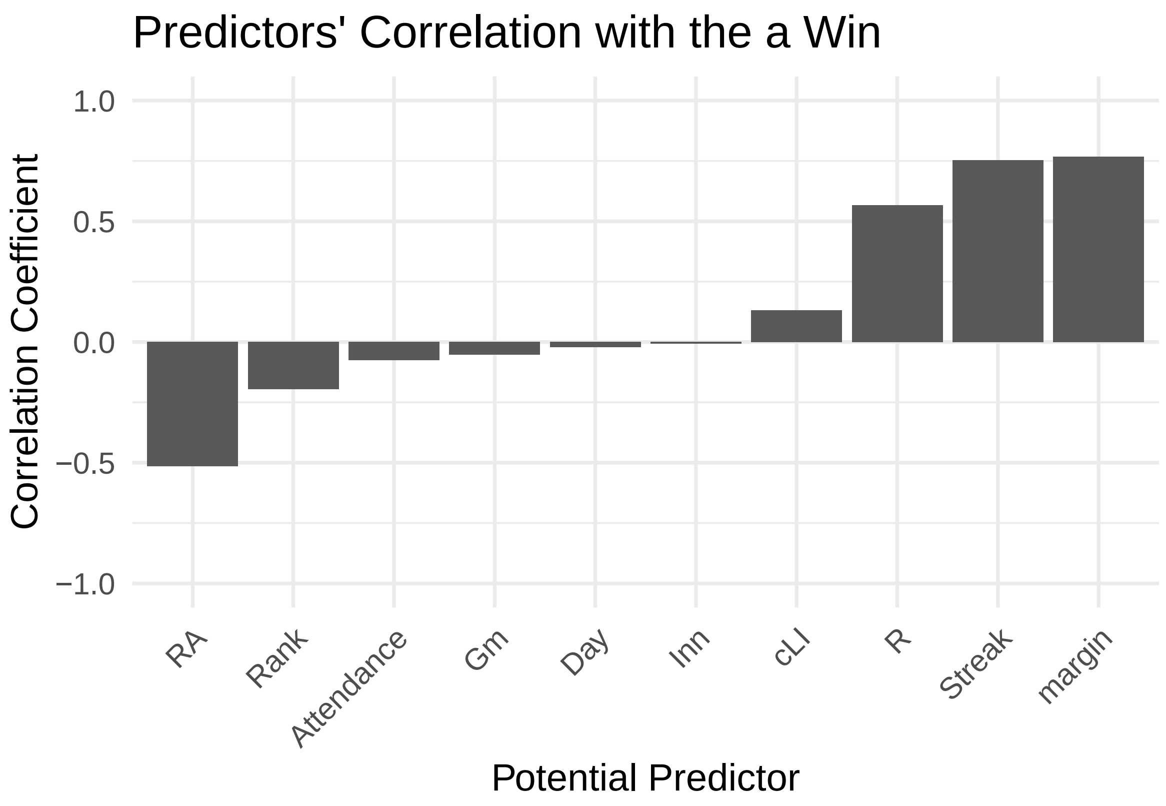
```
Warning: Ignoring unknown parameters: binwidth, bins, pad
```



We randomly selected six team that have lost more than won. This can be an issue with logistic regression, but there are ways to fix it as seen in the recipe step.

**Correlation Analysis**

```
correlation_table <- baseball %>%
  correlate() %>%
  select(term,Won) %>%
  drop_na()

correlation_table %>%
  ggplot(aes(x = fct_reorder(term, Won), y = Won)) +
  geom_col() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ylim(-1,1) +
  labs(
    x = "Potential Predictor",
    y = "Correlation Coefficient",
    title = "Predictors' Correlation with the a Win"
  )
```



```r
# change won to factor
baseball <- baseball %>%
  mutate(Won = as_factor(Won))
```

The correlation graph mostly makes sense. Runs are highly positively correlated with a win, while runs against are highly negatively correlated with the winning a baseball game.

## Modeling

### Training/testing split

```r
set.seed(123)
baseball_split <- initial_split(baseball)
baseball_training <- training(baseball_split)
baseball_testing <- testing(baseball_split)
```

### Specify a Model

```r
baseball_spec <- logistic_reg() %>%
  set_engine("glm")
```

### Recipe 1

```r
bball_rec_1 <- recipe(Won ~ Rank + Date, data = baseball_training) %>%
  update_role(Date, new_role = "id variable") %>%
  step_dummy(all_nominal_predictors()) %>% # dummy coding
  step_zv(all_predictors()) # remove zero variance variables

bball_rec_1
```

```
Recipe

Inputs:

        role #variables
 id variable          1
```

```
     outcome          1
   predictor          1

Operations:

Dummy variables from all_nominal_predictors()
Zero variance filter on all_predictors()
```

**Recipe 2**

```
bball_rec_2 <- recipe(Won ~ Date + Rank + cLI, data = baseball_training) %>%
  update_role(Date, new_role = "id variable") %>%
  step_dummy(all_nominal_predictors()) %>% # dummy coding
  step_zv(all_predictors()) # remove zero variance variables

bball_rec_2
```

```
Recipe

Inputs:

        role #variables
 id variable          1
     outcome          1
   predictor          2

Operations:

Dummy variables from all_nominal_predictors()
Zero variance filter on all_predictors()
```

**Recipe 3**

```
bball_rec_3 <- recipe(Won ~ Date + Rank + cLI + Day, data = baseball_training) %>%
  update_role(Date, new_role = "id variable") %>%
  step_dummy(all_nominal_predictors()) %>% # dummy coding
  step_zv(all_predictors()) # remove zero variance variables
```

### Create Workflows

```r
# workflow brings together recipe and model
bball_wflow1 <- workflow() %>%
  add_model(baseball_spec) %>%
  add_recipe(bball_rec_1)

# workflow brings together recipe and model
bball_wflow2 <- workflow() %>%
  add_model(baseball_spec) %>%
  add_recipe(bball_rec_2)

bball_wflow3 <- workflow() %>%
  add_model(baseball_spec) %>%
  add_recipe(bball_rec_3)
```

### Perform Cross Validation

```r
set.seed(345) # set seed

baseball_folds <- vfold_cv(baseball_training, v = 5) # split up data into 5 partitions

bball_fit_cv1 <- bball_wflow1 %>%
  fit_resamples(baseball_folds,
                control = control_resamples(save_pred = TRUE)) # fit all the folds using reci

bball_fit_cv2 <- bball_wflow2 %>%
  fit_resamples(baseball_folds,
                control = control_resamples(save_pred = TRUE)) # fit all the folds using reci

bball_fit_cv3 <- bball_wflow3 %>%
  fit_resamples(baseball_folds,
                control = control_resamples(save_pred = TRUE)) # fit all the folds using reci
```

```r
collect_metrics(bball_fit_cv1) %>%
  select(.metric,mean) %>%
  kable(digits = 3)
```

| .metric | mean |
|---|---|
| accuracy | 0.595 |
| roc_auc | 0.596 |

```r
collect_metrics(bball_fit_cv2) %>%
  select(.metric,mean) %>%
  kable(digits = 3)
```

| .metric | mean |
|---|---|
| accuracy | 0.586 |
| roc_auc | 0.591 |

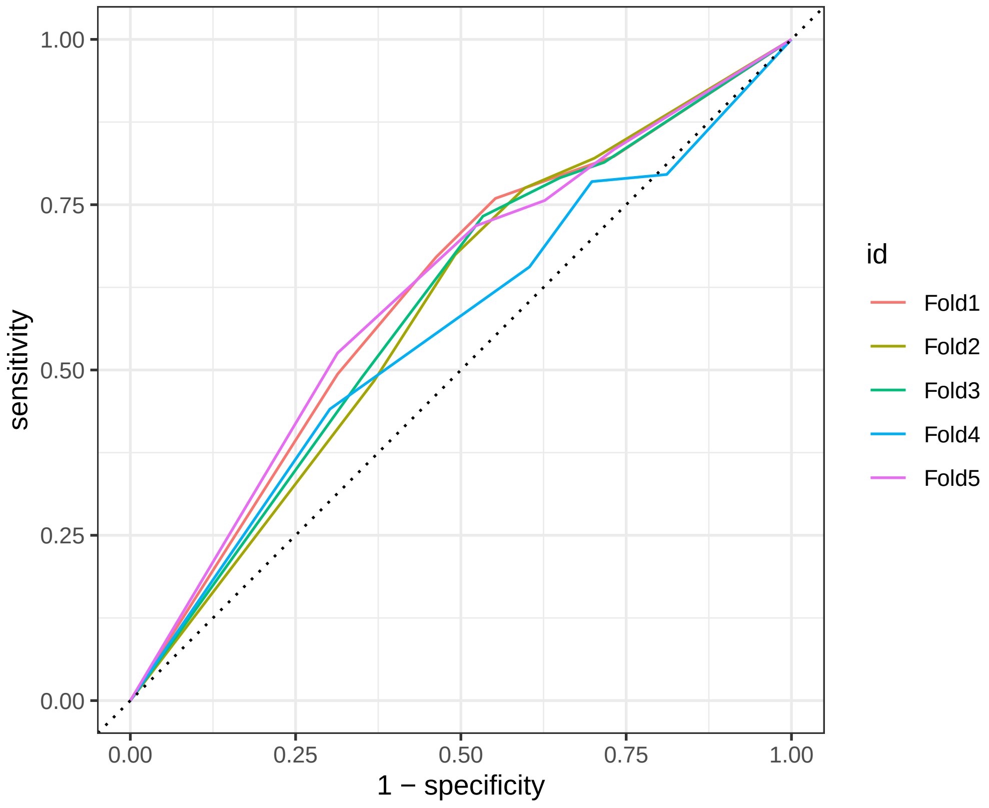
```r
collect_metrics(bball_fit_cv3) %>%
  select(.metric,mean) %>%
  kable(digits = 3)
```

| .metric | mean |
|---|---|
| accuracy | 0.583 |
| roc_auc | 0.589 |

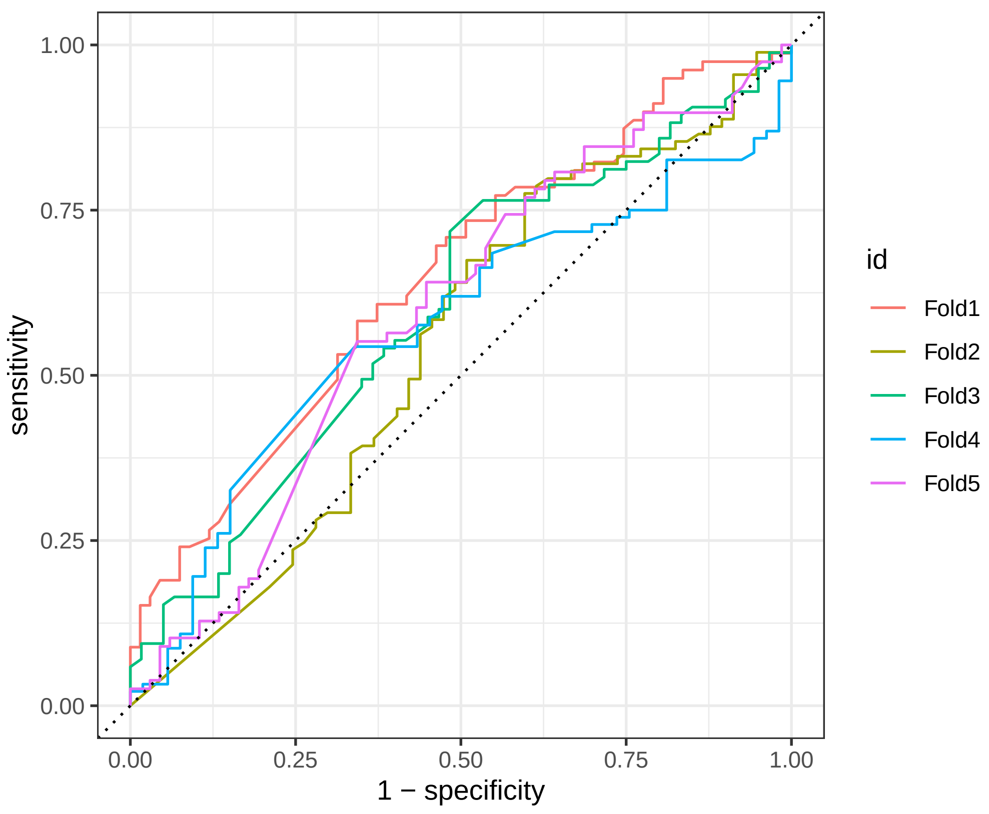
### Visualizing the ROC Curve

Recipe 1 through 3 shown below.

```r
p1 <- bball_fit_cv1 %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(
    truth = Won,
    .pred_0
  ) %>%
  autoplot()

p2 <- bball_fit_cv2 %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(
    truth = Won,
    .pred_0
  ) %>%
  autoplot()

p3 <- bball_fit_cv3 %>%
  collect_predictions() %>%
  group_by(id) %>%
  roc_curve(
    truth = Won,
    .pred_0
  ) %>%
  autoplot()

p1
```

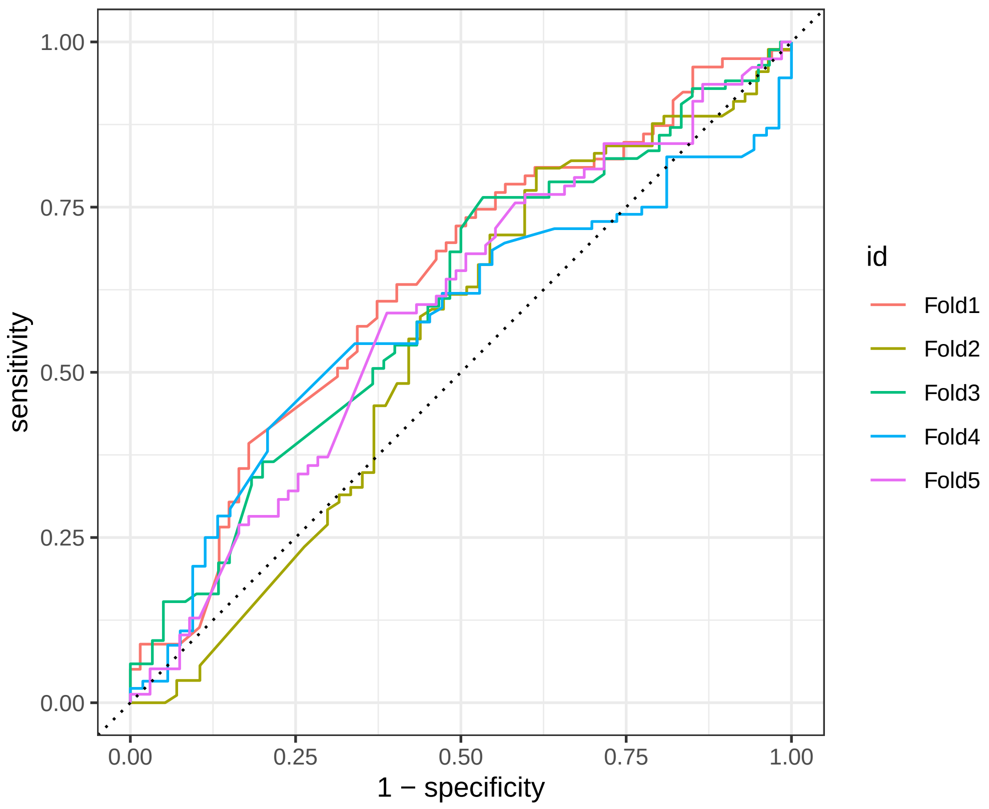


```
p2
```



1.00
0.75
0.50
0.25
0.00
sensitivity
0.00
0.25
0.50
0.75
1.00
1 − specificity
id
Fold1
Fold2
Fold3
Fold4
Fold5

```
p3
```



### Model Selection

Clearly Model 1 is the best model and also the most parsimonious. It has the most consistent ROC Curve, and the highest metrics for accuracy.

### Apply Model to testing Data

```r
final_fit <- last_fit(
  bball_wflow1,
  split = baseball_split
  )

collect_predictions(final_fit) %>%
  conf_mat(Won, .pred_class)
```

```
          Truth
```

```
Prediction   0   1
         0 116  79
         1  14  34
```

```
collect_metrics(final_fit) %>%
  kable(digits = 3)
```

| .metric | .estimator | .estimate | .config |
|---|---|---|---|
| accuracy | binary | 0.617 | Preprocessor1_Model1 |
| roc_auc | binary | 0.647 | Preprocessor1_Model1 |

By simply looking at the baseball's team rank in their respective division, we can achieve a 61.7% accuracy in predicting whether the baseball team will win or lose.