MLB Data Final Project

Nicolas Thomas John Mayfield Konner Stine Preston McCaffrey Jacob Slack

***Abstract*—This is the project report for our Final Project in Statistical Learning, researching and implementing the modeling methods into a Major League Baseball Dataset. Throughout this document, the dataset is explored, and then delves into different modeling methods by select criterion. Then, multiple graphs and models are displayed, and paired with criteria from each model, we can conclude which model is the best for representing our Major League Baseball dataset. There is also an appendix with our code and modeling rough work, created in RStudio.**

Keywords—EDA, Logistic, Trees, Boosting

I. Introduction and Project Overview

For our project, we have selected the Raw MLB Player Dataset from Kaggle. This dataset contains offensive statistics on MLB players from 1947 until 2024. There are 43 variables (continuous and categorical) for each row where each observation is a different MLB player. There are 1286 observations in the dataset. With this dataset we aim to develop a model to predict whether a given player was put into the MLB Hall of Fame (HOF Status). We aimed to gain insights into the factors that contribute to a player being inducted into the MLB Hall of Fame and to develop a model that can accurately predict which players will be inducted in the future.

We used techniques learned in class involving exploratory data analysis, feature selection, data cleaning, model development, and model evaluation to complete our objective. The initial exploratory analysis included techniques such as summary statistics, pairwise data visualizations, and correlation calculations. For feature selection, in addition to exploratory data analysis we used methods learned in class such as statistical tests to determine individual impacts on the model. For the model development and evaluation, we used categorical predictive models such as logistic regression and different tree based methods, and the corresponding evaluation tools such as a confusion matrix.

II. Ease of Use

Taking a look at the raw data, we first checked to see if there are any missing values. Fortunately, there were no missing pieces within the data. The next step we took was to remove the Player variable, which contained the names of the players, as it would not be useful in training the model and would cause issues with feature factorization down the line.

III. Exploratory Data Analysis

Before you begin to format your paper, first write and save the content as a separate text file. Complete all content and organizational editing before formatting. Please note sections A-D below for more information on proofreading, spelling and grammar.

Keep your text and graphic files separate until after the text has been formatted and styled. Do not use hard tabs, and limit use of hard returns to only one return at the end of a paragraph. Do not add any kind of pagination anywhere in the paper. Do not number text heads-the template will do that for you.

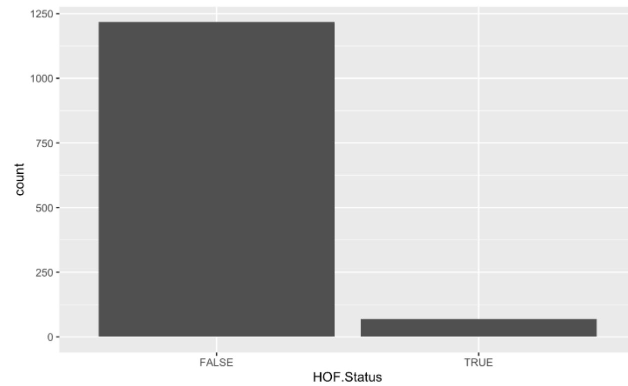


Fig. 1. HOF Status Distribution

Because this dataset contains a large number of predictors, our initial goal with exploratory data analysis is to explore each variable and ultimately reduce the number of variables that we will be using in our models if possible without losing too much predictive power. To do this, it is important to take a look at the features in relation to our target variable. To do this, we created pairwise scatter plots to see if there are any visual indications that they are important predictors for HOF.Status. Below are a few that visually appear to have different distributions depending on if HOF.Status is True or False.

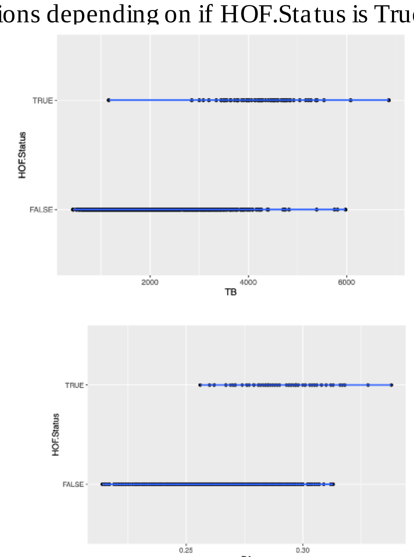


Fig. 2. TB and BA vs HOF Status

When looking at total bases (TB) it appears that when HOF.Status is True, total bases is higher than when HOF.Status is False on average. Similarly, it appears that the batting average (BA) is generally higher when HOF.Status is True than when it is False. These logically make sense as we would expect a higher performing player to be more likely to be put in the hall of fame.

In contrast to the two variables shown, we found several predictors that we could initially remove from our dataset due to no visual indication of importance. Based on the visual assessments and plentiful predictors we decided to remove Rfield, Rbaser, Rdp, and Rbaser…Rdp. Two of these predictors are shown below. Runs from fielding (Rfield) and Runs Base Running + Runs Double Play (Rbaser…Rdp) are two of the predictors that on average did not seem to have differences when grouping by Hall of Fame statuses.

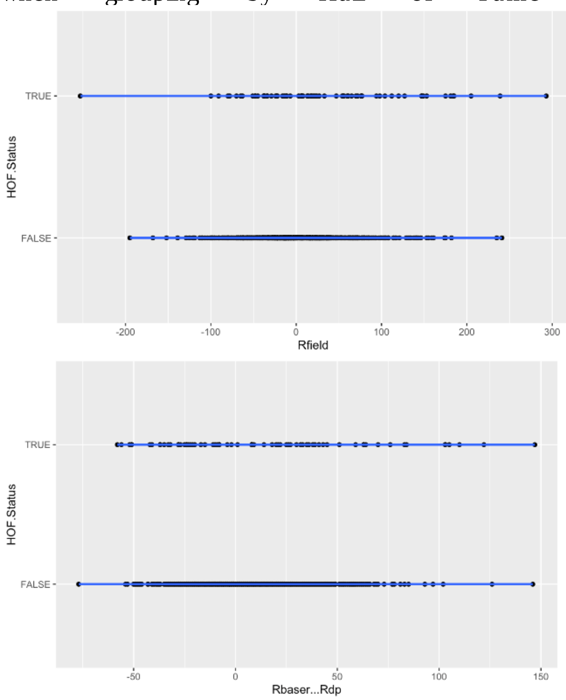


Fig. 3. Rbaser_Rdp and Rfield vs HOF Status

For the two categorical variables in the dataset, we found the mean and median of them when HOF.Status is True and when it is False.

Suspected.Steroids	mean	median	n
FALSE	0.0540752	0	1276
TRUE	0.0000000	0	10

Suspended	mean	median	n
FALSE	0.0541176	FALSE	1275
TRUE	0.0000000	FALSE	11

Fig. 4. Categorical Variable Distribution

It is evident from these findings that when a player has been inducted in the hall of fame, there has never been an instance where they were suspended or suspected of steroids. When HOF.Status is True no observations were suspended or suspected of steroids and in the hall of fame.

We created a correlation matrix to compare each variable in a pairwise fashion. There were several pairs of variables with high correlations to each other. In an attempt to reduce model complexity, we have removed one variable from each pair of highly correlated features. We removed the following variables since they had a correlation of 0.9 or greater with another variable:

"PA", "OPS", "X3B", "Final.Season.Age", "AB", "HR" , "IBB", "G", "Suspended",

"Debut.Age", "oWAR", "OBP", "RBI", "Rdp", "Suspected.Steroids"

The correlation matrix is as follows:

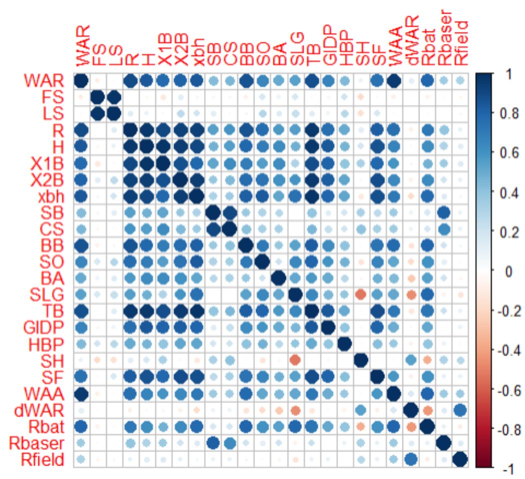


Fig. 5. Correlation Matrix

(FS = First Season, LS = Last Season)

We can also look at the anova table for our model to see which factors have the largest impact on if a player is in the Hall of Fame or not. The ANOVA table shows the reduction in deviance as each variable is added. This reduction in deviance is an indicator of predictive power and importance in the data, allowing us to gain insight regarding each variables importance.

The anova table is as follows:

	Df	Deviance	Resid. Df	Resid. Dev
NULL			1285	65.298
mlbdata$WAR	1	24.5911	1284	40.707
mlbdata$First.Season	1	0.1584	1283	40.548
mlbdata$Last.Season	1	0.0472	1282	40.501
mlbdata$R	1	0.0537	1281	40.447
mlbdata$H	1	0.3206	1280	40.127
mlbdata$X1B	1	0.0260	1279	40.101
mlbdata$X2B	1	0.0744	1278	40.026
mlbdata$xbh	0	0.0000	1278	40.026
mlbdata$SB	1	0.0246	1277	40.002
mlbdata$CS	1	0.8251	1276	39.177
mlbdata$BB	1	0.1230	1275	39.054
mlbdata$SO	1	0.2311	1274	38.823
mlbdata$BA	1	0.0664	1273	38.756
mlbdata$SLG	1	0.2927	1272	38.464
mlbdata$TB	1	0.0024	1271	38.461
mlbdata$GIDP	1	0.0574	1270	38.404
mlbdata$HBP	1	0.0772	1269	38.326
mlbdata$SH	1	0.0419	1268	38.285
mlbdata$SF	1	0.4223	1267	37.862
mlbdata$WAA	1	0.8834	1266	36.979
mlbdata$dWAR	1	0.9478	1265	36.031
mlbdata$Rbat	1	0.0790	1264	35.952
mlbdata$Rbaser	1	0.0921	1263	35.860
mlbdata$Rfield	1	0.0977	1262	35.762

Fig. 6. Anova Table

Combining the insight gained from each of the phases of exploratory data analysis we were able to reduce the number of variables from 43 to 18. The following variables are the ones that remain:

WAR, dWAR, oWAR, R, H, X1B, X2B, xbh, SB, CS, BB, SO, OPS, TB, GIDP, HBP, SF, WAA, Rbat

IV. Further Feature Selection and Hyperparameter Tuning

After conducting exploratory data analysis, we removed many features based on initial inspections; however, since we started with 43 predictors and eliminated 25, we have 18 left. To further reduce the number of predictors in an attempt to help reduce model complexity, we implemented backwards elimination for each of our models. Backward elimination is a method that starts with all the predictors and removes the least important one in each step. Through this we can determine which features are most important for each model to effectively maintain model performance while reducing complexity.

Ultimately we are left with these selected variables:

"R" "X1B" "SB" "CS" "TB" "SF" "WAA"

With these features and initial versions of the models, we then implemented hyperparameter tuning using grid search cross validation. This was done in order to maximize model performance. The specific hyperparameters chosen can be seen in the appendix along with the backward elimination process.

V. Model Training and Evaluation

After finalizing feature selection and tuning the models, we then trained our models on the features we had selected. The models that we are using consist of logistic regression, simple decision tree, random forest, and a boosted decision tree. Logistic regression is easy to understand and useful for straightforward classification tasks. A simple decision tree is also intuitive and can handle non-linear patterns in the data. Random forests combine multiple decision trees to improve accuracy and reduce overfitting. Boosted decision trees build on each other to handle tough cases and usually offer high performance. Using these models gives us a mix of simplicity, interpretability, and powerful prediction capabilities.

It is important to note that the accuracy metric has the potential to be heavily skewed since we have significantly fewer observations with HOF.Status being True. Thus, looking at sensitivity (correctly predicted true rate) and specificity (correctly predicted false rate) are very integral to understanding the true performance. Below are the generated decision trees, Random Forest, and ROC Curve. For the decision trees they can be interpreted by starting from the top and following the logic with the given observation's features. This then leads to the ultimate categorization of the model will output. As well as the comparisons of the different models.

A tibble: 1 x 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
accuracy	binary	0.9767442

Fig. 7. Accuracy of model

term <chr>	estimate <dbl>	penalty <dbl>
(Intercept)	-7.973063e+00	0
R	8.261204e-04	0
X1B	1.538401e-03	0
SB	-5.356757e-05	0
CS	-4.029762e-03	0
TB	5.670251e-04	0
SF	4.427216e-03	0
WAA	5.361382e-02	0

Fig. 8. Model coefficients

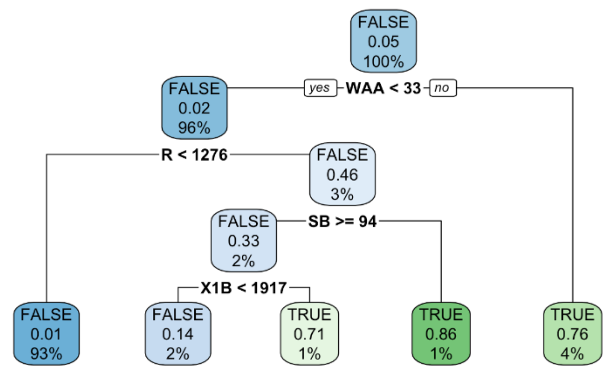


Fig. 9. Decision Tree with HOF Status as target

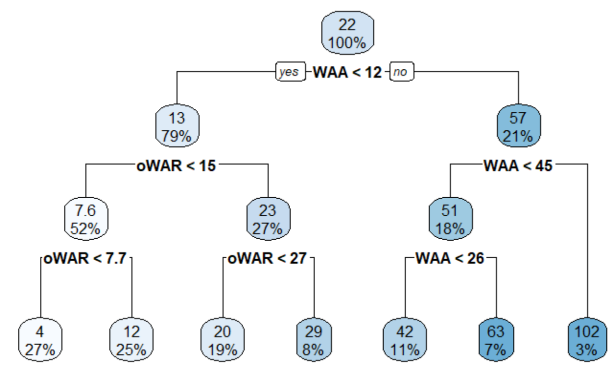


Fig. 10. Decision Tree with WAR Status as target

The first decision tree using HOF.status as a target, which is our intended prediction. The second decision tree uses WAR as a target. WAR (Wins above replacement) is an aggregated statistic to compare a player to the average player, which makes sense to be heavily predictive of Hall of Fame status. We fit a decision tree for WAR to gain further insight since it is so predictive of Hall of Fame status.

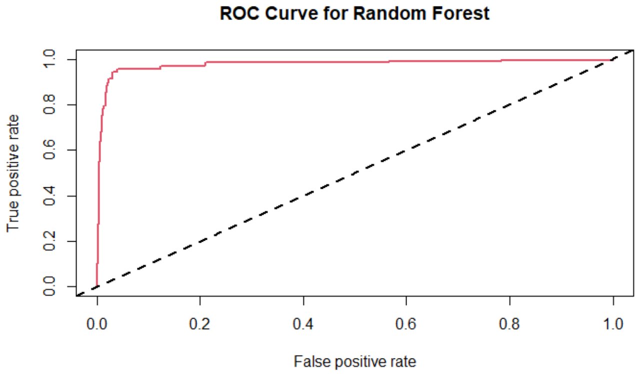


Fig. 11. ROC Curve for Random Forest

This ROC curve for the Random Forest model illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate at various threshold levels. The curve's closeness to the top-left corner indicates that the dataset is imbalanced, which we also discovered in the Exploratory Data Analysis.

Accuracy - (Logistic = Simple DT < Boosted DT)
Sensitivity - (Simple DT < Logistic < Boosted DT)
Specificity - (Logistic < Simple DT < Boosted DT)

From these results it is clear that the Boosted Decision Tree has the best overall performance metrics, which is not surprising due to its robust training process. These metrics, however, do not take into consideration complexity and interpretability of the models which are also very important concepts to consider when ultimately deciding on a model.

VI. CONCLUSION

With our initial goal of this project to develop an accurate model while gaining insights into the factors that contribute to a player being inducted into the MLB Hall of Fame, the model selected is the simple decision tree. This is because of its overall good performance and high interpretability in comparison to the other models. The simple decision tree had a prediction accuracy of 97.92%. It correctly predicted Hall of Fame players 80% of the time and correctly identified Non-Hall of Fame players 98.80% of the time.

This model performs well overall. For further future work, doing more research into the abundance of baseball statistics that exist would enable further insight into each of the predictors and potentially uncover why some are highly correlated or have such high predictive power.

REFERENCES

[1] Coxen, C. (2024, May 14). *Raw MLB Player Data*. Kaggle. https://www.kaggle.com/datasets/chriscoxen/raw-mlb-player-data/data

APPENDIX

```
```{r}
library(readr)
# install.packages("tidymodels")
library(tidymodels)

# bank_df <- read_csv2("bank-full.csv")

mlbdataframe <- glm(HOF.Status ~ R + X1B + SB + CS + TB + SF + WAA, data = mlbdata, family =
"binomial")

# Read the dataset and convert the target variable to a factor
# mlbdata <- mlbdata
# mlbdata$HOF.Status = as.factor(mlbdata$HOF.Status)

# Split data into train and test
set.seed(421)
split <- initial_split(mlbdata, prop = 0.8, strata = HOF.Status)
train <- split %>%
        training()
test <- split %>%
        testing()

# Train a logistic regression model
model <- logistic_reg(mixture = double(1), penalty = double(1)) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit( ~ ., data = mlbdata)

# Model summary
tidy(model)
```
```

```
library(conflicted)

data <- read.csv("MLB_Player_Data.csv")
data$Player <- NULL

summary(data)

##  HOF.Status      Suspended       Suspected.Steroids      WAR
##  Mode :logical   Mode :logical   Mode :logical      Min.   : -6.90
##  FALSE:1217      FALSE:1275      FALSE:1276         1st Qu.:  6.70
##  TRUE :69        TRUE :11        TRUE :10           Median : 14.70
##                                                     Mean   : 20.25
##                                                     3rd Qu.: 27.20
##                                                     Max.   :162.80
```

```
##   First.Season   Last.Season     Debut.Age     Final.Season.Age       G
##  Min.   :1947   Min.   :1955   Min.   :17.00   Min.   :28.00    Min.   : 750
##  1st Qu.:1966   1st Qu.:1980   1st Qu.:21.00   1st Qu.:33.00    1st Qu.: 976
##  Median :1981   Median :1993   Median :22.00   Median :35.00    Median :1292
##  Mean   :1980   Mean   :1992   Mean   :22.41   Mean   :34.94    Mean   :1396
##  3rd Qu.:1993   3rd Qu.:2006   3rd Qu.:24.00   3rd Qu.:37.00    3rd Qu.:1680
##  Max.   :2012   Max.   :2017   Max.   :30.00   Max.   :48.00    Max.   :3562
##        PA              AB               R                H
##  Min.   : 1512   Min.   : 1362   Min.   : 157.0   Min.   : 333.0
##  1st Qu.: 3341   1st Qu.: 2984   1st Qu.: 364.0   1st Qu.: 773.2
##  Median : 4652   Median : 4147   Median : 546.5   Median :1104.5
##  Mean   : 5193   Mean   : 4617   Mean   : 626.1   Mean   :1242.3
##  3rd Qu.: 6474   3rd Qu.: 5778   3rd Qu.: 780.0   3rd Qu.:1564.0
##  Max.   :15890   Max.   :14053   Max.   :2295.0   Max.   :4256.0
##       X1B              X2B              X3B            HR
##  Min.   : 227.0   Min.   : 38.0   Min.   :  1   Min.   :  1.00
##  1st Qu.: 529.2   1st Qu.:132.2   1st Qu.: 15   1st Qu.: 50.25
##  Median : 764.5   Median :193.0   Median : 25   Median :100.00
##  Mean   : 857.3   Mean   :221.6   Mean   : 31   Mean   :132.49
##  3rd Qu.:1087.0   3rd Qu.:282.0   3rd Qu.: 40   3rd Qu.:170.00
##  Max.   :3215.0   Max.   :746.0   Max.   :166   Max.   :762.00
##       xbh              RBI               SB                CS
##  Min.   :  53.0   Min.   : 112.0   Min.   :   0.00   Min.   :  0.00
##  1st Qu.: 222.0   1st Qu.: 342.0   1st Qu.:  19.25   1st Qu.: 18.00
##  Median : 324.5   Median : 486.5   Median :  47.00   Median : 31.00
##  Mean   : 385.1   Mean   : 594.1   Mean   :  89.10   Mean   : 42.36
##  3rd Qu.: 490.8   3rd Qu.: 744.0   3rd Qu.: 108.00   3rd Qu.: 57.00
##  Max.   :1477.0   Max.   :2297.0   Max.   :1406.00   Max.   :335.00
##        BB               SO               BA               OBP
##  Min.   :  76.0   Min.   : 108.0   Min.   :0.2140   Min.   :0.2520
##  1st Qu.: 257.2   1st Qu.: 444.0   1st Qu.:0.2520   1st Qu.:0.3150
##  Median : 383.0   Median : 635.0   Median :0.2640   Median :0.3310
##  Mean   : 467.4   Mean   : 723.8   Mean   :0.2647   Mean   :0.3326
##  3rd Qu.: 578.8   3rd Qu.: 926.0   3rd Qu.:0.2770   3rd Qu.:0.3490
##  Max.   :2558.0   Max.   :2597.0   Max.   :0.3380   Max.   :0.4440
##       SLG              OPS              OPS.             TB
##  Min.   :0.2590   Min.   :0.5290   Min.   : 46.0   Min.   : 433
##  1st Qu.:0.3660   1st Qu.:0.6880   1st Qu.: 87.0   1st Qu.:1154
##  Median :0.4060   Median :0.7380   Median :100.0   Median :1654
##  Mean   :0.4054   Mean   :0.7380   Mean   :100.1   Mean   :1923
##  3rd Qu.:0.4427   3rd Qu.:0.7857   3rd Qu.:112.0   3rd Qu.:2407
##  Max.   :0.6070   Max.   :1.0510   Max.   :182.0   Max.   :6856
##       GIDP           HBP               SH               SF
##  Min.   : 14.0   Min.   :  0.00   Min.   :  0.00   Min.   :  2.00
##  1st Qu.: 64.0   1st Qu.: 15.00   1st Qu.: 12.00   1st Qu.: 23.00
##  Median : 92.0   Median : 25.00   Median : 26.00   Median : 33.00
##  Mean   :105.3   Mean   : 34.62   Mean   : 33.66   Mean   : 39.38
##  3rd Qu.:133.8   3rd Qu.: 45.00   3rd Qu.: 47.00   3rd Qu.: 51.00
##  Max.   :350.0   Max.   :285.00   Max.   :256.00   Max.   :128.00
##       IBB              WAA               oWAR             dWAR
##  Min.   :  0.00   Min.   :-20.500   Min.   : -7.50   Min.   :-28.40000
```

```
##  1st Qu.: 18.00   1st Qu.: -5.375   1st Qu.:  7.10   1st Qu.: -5.40000
##  Median : 31.00   Median : -0.600   Median : 14.50   Median : -0.50000
##  Mean   : 44.76   Mean   :  3.052   Mean   : 20.04   Mean   :  0.07698
##  3rd Qu.: 55.00   3rd Qu.:  7.100   3rd Qu.: 26.65   3rd Qu.:  5.17500
##  Max.   :688.00   Max.   :123.800   Max.   :143.60   Max.   : 44.20000
##       Rbat               Rdp                 Rbaser          Rbaser...Rdp
##  Min.   :-305.00   Min.   :-42.0000   Min.   :-39.000   Min.   :-77.000
##  1st Qu.: -57.00   1st Qu.: -6.0000   1st Qu.: -8.000   1st Qu.:-12.000
##  Median :   3.00   Median : -1.0000   Median : -2.000   Median : -2.000
##  Mean   :  30.35   Mean   : -0.4565   Mean   :  1.738   Mean   :  1.303
##  3rd Qu.:  79.00   3rd Qu.:  5.0000   3rd Qu.:  7.000   3rd Qu.: 10.000
##  Max.   :1128.00   Max.   : 49.0000   Max.   :144.000   Max.   :147.000
##      Rfield
##  Min.   :-253.000
##  1st Qu.: -25.000
##  Median :   0.000
##  Mean   :   3.116
##  3rd Qu.:  27.000
##  Max.   : 293.000
```

Exploratory Data Analysis

Target Variable

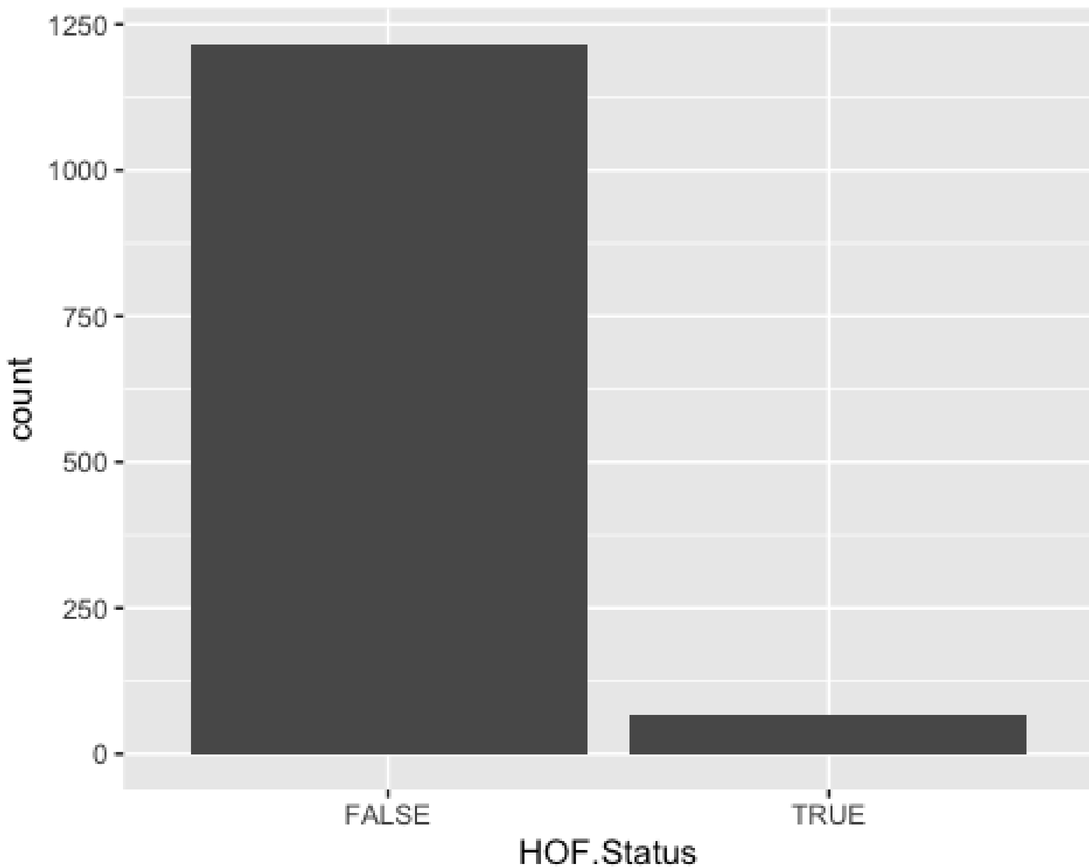
```r
summary(data$HOF.Status)
```

```
##    Mode   FALSE    TRUE
## logical    1217      69
```

```r
library(ggplot2)

ggplot(data , aes(x = HOF.Status)) +
  geom_histogram(stat="count")
```

```
## Warning in geom_histogram(stat = "count"): Ignoring unknown parameters:
## `binwidth`, `bins`, and `pad`
```




```
nrow(data [data$HOF.Status == 0, ])

## [1] 1217
```

Investigating Numeric Predictors – Bivariate Data Exploration

Report the correlation matrix of the numeric predictors.

```
# correlation matrix for numeric variables

numeric_data <- data[, sapply(data, is.numeric)]
cor.matrix <- cor(numeric_data[, !names(numeric_data) %in% "HOF.Status"])
#print("Correlation Matrix")
#round(cor.matrix, digits = 4)

# Find highly correlated variables (with a threshold of 0.9)
library(caret)

## Loading required package: lattice

highly_correlated <- findCorrelation(cor.matrix, cutoff = 0.9)

all_columns <- names(data)
columns_to_remove <- all_columns[!all_columns %in% "HOF.Status"][highly_correlated]
```

```
# Remove one variable from each pair of highly correlated variables, excluding
HOF.Status
reduced_data <- data[, !names(data) %in% columns_to_remove]

print("Removed Variables:")

## [1] "Removed Variables:"

print(columns_to_remove)

##  [1] "PA"                "OPS"               "X3B"
##  [4] "Final.Season.Age"  "AB"                "HR"
##  [7] "IBB"               "G"                 "Suspended"
## [10] "Debut.Age"         "oWAR"              "OBP"
## [13] "RBI"               "Rdp"               "Suspected.Steroids"

print("Remaining Variables:")

## [1] "Remaining Variables:"

print(names(reduced_data))

##  [1] "HOF.Status"   "WAR"          "First.Season" "Last.Season"  "R"
##  [6] "H"            "X1B"          "X2B"          "xbh"          "SB"
## [11] "CS"           "BB"           "SO"           "BA"           "SLG"
## [16] "OPS."         "TB"           "GIDP"         "HBP"          "SH"
## [21] "SF"           "WAA"          "dWAR"         "Rbat"         "Rbaser"
## [26] "Rbaser...Rdp" "Rfield"

library(dplyr)

# Calculate the mean of each variable for HOF.Status
mean_by_category <- data %>%
  group_by(HOF.Status) %>%
  summarise(across(everything(), ~ mean(.x, na.rm = TRUE)))

# Separate the means by HOF.Status categories
mean_false <- mean_by_category %>% dplyr::filter(HOF.Status == FALSE) %>% select(-
HOF.Status)
mean_true <- mean_by_category %>% dplyr::filter(HOF.Status == TRUE) %>% select(-
HOF.Status)

# Normalize
normalized_diff <- abs(mean_false - mean_true) / (mean_false + mean_true)

# threshold for considering means to be significantly different
threshold <- 0.1

# variables to keep (where normalized difference is greater than threshold)
variables_to_keep <- names(normalized_diff)[apply(normalized_diff, 1, function(x) x >
threshold)]
```

```r
print(variables_to_keep)
```

```
##  [1] "Suspended"          "Suspected.Steroids" "WAR"
##  [4] "G"                  "PA"                 "AB"
##  [7] "R"                  "H"                  "X1B"
## [10] "X2B"                "X3B"                "HR"
## [13] "xbh"                "RBI"                "SB"
## [16] "CS"                 "BB"                 "SO"
## [19] "OPS."               "TB"                 "GIDP"
## [22] "HBP"                "SF"                 "IBB"
## [25] "WAA"                "oWAR"               "dWAR"
## [28] "Rbat"               "Rbaser"             "Rbaser...Rdp"
## [31] "Rfield"
```

```r
variables_to_remove <- setdiff(names(mean_false), variables_to_keep)
print(variables_to_remove)
```

```
##  [1] "First.Season"     "Last.Season"      "Debut.Age"        "Final.Season.Age"
##  [5] "BA"               "OBP"              "SLG"              "OPS"
##  [9] "SH"               "Rdp"
```

```r
reduced_data <- reduced_data[, !names(reduced_data) %in% variables_to_remove]
```

Visualization

Create visual representations between the numeric variables and the target variable to identify variables most likely to predict the target.

Scatterplots between numeric variables and target variable is shown below.

```r
vars.numeric <- colnames(numeric_data)
for (i in vars.numeric) {
  plot <- ggplot(data, aes(x = numeric_data[, i], y = HOF.Status)) +
    geom_point() +
    geom_smooth(method = "lm", se = FALSE) +
    labs(x = i)
  # This is commented out to avoid overflowing document with 43 plots but in our
coding we    # gathered all 43 pairwise comparison plots
  #print(plot)
}

# Remove non predictive variables

reduced_data$Rfield <- NULL
reduced_data$Rbaser <- NULL
reduced_data$Rdp <- NULL
reduced_data$Rbaser...Rdp <- NULL
```

Factor Variables

```r
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✓ forcats   1.0.0     ✓ stringr   1.5.1
## ✓ lubridate 1.9.3     ✓ tibble    3.2.1
## ✓ purrr     1.0.2     ✓ tidyr     1.3.0
## ✓ readr     2.1.3
```

```
library(knitr)

vars.categorical <- c("Suspended", "Suspected.Steroids")

for (i in vars.categorical) {
  x <- data %>%
    group_by(!!sym(i)) %>%
    summarise(
      mean = mean(HOF.Status),
      median = median(HOF.Status),
      n = n()
    )

  print(kable(x, caption = paste("Summary for", i)))
}
```

```
##
##
## Table: Summary for Suspended
##
## |Suspended |      mean|median |    n|
## |:---------|---------:|:------|----:|
## |FALSE     | 0.0541176|FALSE  | 1275|
## |TRUE      | 0.0000000|FALSE  |   11|
##
##
## Table: Summary for Suspected.Steroids
##
## |Suspected.Steroids |      mean| median|    n|
## |:------------------|---------:|------:|----:|
## |FALSE              | 0.0540752|      0| 1276|
## |TRUE               | 0.0000000|      0|   10|
```

Exercise 5

Split the data into training and test sets

By using function, create the training and testing split.

```
library(caret)
set.seed(8964)
partition <- createDataPartition(reduced_data$HOF.Status, p = 0.75, list = FALSE)
modeldata.train <- reduced_data[partition, ]
modeldata.test <- reduced_data[-partition, ]
```

```r
print("modeldata.train")

## [1] "modeldata.train"

mean(modeldata.train$HOF.Status)

## [1] 0.05388601

print("modeldata.test")

## [1] "modeldata.test"

mean(modeldata.test$HOF.Status)

## [1] 0.0529595
```

Fitting MLR

```r
library(caret)
library(MASS)
library(rpart)
library(xgboost)
library(dplyr)

set.seed(123)

data <- reduced_data
data$HOF.Status <- as.factor(data$HOF.Status)

# Split the data into training and testing sets
index <- createDataPartition(data$HOF.Status, p = 0.7, list = FALSE)
train_data <- data[index, ]
test_data <- data[-index, ]

# Perform backward selection for Logistic Regression
full_model <- glm(HOF.Status ~ ., data = train_data, family = binomial)
backward_model <- stepAIC(full_model, direction = "backward")

## Start:  AIC=167.41
## HOF.Status ~ WAR + R + H + X1B + X2B + xbh + SB + CS + BB + SO +
##     OPS. + TB + GIDP + HBP + SF + WAA + dWAR + Rbat
##
##
## Step:  AIC=167.41
## HOF.Status ~ WAR + R + H + X1B + X2B + SB + CS + BB + SO + OPS. +
##     TB + GIDP + HBP + SF + WAA + dWAR + Rbat
##
##        Df Deviance    AIC
## - H     1   131.41 165.41
## - X2B   1   131.42 165.42
## - WAR   1   131.42 165.42
## - X1B   1   131.43 165.43
```

```
## - dWAR  1   131.43 165.43
## - TB    1   131.45 165.45
## - GIDP  1   131.53 165.53
## - HBP   1   131.55 165.55
## - SO    1   131.73 165.73
## - WAA   1   131.83 165.83
## - BB    1   132.02 166.02
## - Rbat  1   132.19 166.19
## <none>      131.41 167.41
## - OPS.  1   133.55 167.55
## - SB    1   133.96 167.96
## - CS    1   134.12 168.12
## - R     1   136.57 170.57
## - SF    1   137.63 171.63
##
## Step:  AIC=165.41
## HOF.Status ~ WAR + R + X1B + X2B + SB + CS + BB + SO + OPS. +
##     TB + GIDP + HBP + SF + WAA + dWAR + Rbat
##
##        Df Deviance    AIC
## - WAR   1   131.42 163.42
## - dWAR  1   131.44 163.44
## - GIDP  1   131.53 163.53
## - HBP   1   131.56 163.56
## - X2B   1   131.68 163.68
## - SO    1   131.73 163.73
## - WAA   1   131.88 163.88
## - BB    1   132.03 164.03
## - Rbat  1   132.40 164.40
## <none>      131.41 165.41
## - OPS.  1   133.56 165.56
## - SB    1   134.07 166.07
## - CS    1   134.21 166.21
## - TB    1   134.50 166.50
## - R     1   136.57 168.57
## - SF    1   137.71 169.71
## - X1B   1   139.48 171.48
##
## Step:  AIC=163.42
## HOF.Status ~ R + X1B + X2B + SB + CS + BB + SO + OPS. + TB +
##     GIDP + HBP + SF + WAA + dWAR + Rbat
##
##        Df Deviance    AIC
## - dWAR  1   131.44 161.44
## - GIDP  1   131.54 161.54
## - HBP   1   131.56 161.56
## - X2B   1   131.72 161.72
## - SO    1   131.73 161.73
## - BB    1   132.28 162.28
## - Rbat  1   132.43 162.43
## - WAA   1   132.97 162.97
```

```
## <none>      131.42 163.42
## - OPS.  1   133.69 163.69
## - SB    1   134.12 164.12
## - CS    1   134.26 164.26
## - TB    1   136.93 166.93
## - R     1   137.24 167.24
## - SF    1   137.71 167.71
## - X1B   1   141.44 171.44
##
## Step:  AIC=161.44
## HOF.Status ~ R + X1B + X2B + SB + CS + BB + SO + OPS. + TB +
##     GIDP + HBP + SF + WAA + Rbat
##
##        Df Deviance    AIC
## - GIDP  1   131.54 159.54
## - HBP   1   131.57 159.57
## - X2B   1   131.72 159.72
## - SO    1   131.76 159.76
## - BB    1   132.29 160.29
## <none>      131.44 161.44
## - Rbat  1   133.61 161.61
## - OPS.  1   134.04 162.04
## - CS    1   134.60 162.60
## - SB    1   135.20 163.20
## - TB    1   136.94 164.94
## - R     1   137.24 165.24
## - SF    1   137.71 165.71
## - X1B   1   143.29 171.29
## - WAA   1   157.78 185.78
##
## Step:  AIC=159.54
## HOF.Status ~ R + X1B + X2B + SB + CS + BB + SO + OPS. + TB +
##     HBP + SF + WAA + Rbat
##
##        Df Deviance    AIC
## - HBP   1   131.70 157.70
## - X2B   1   131.84 157.84
## - SO    1   131.93 157.93
## - BB    1   132.56 158.56
## <none>      131.54 159.54
## - Rbat  1   133.61 159.61
## - OPS.  1   134.05 160.05
## - CS    1   135.15 161.15
## - SB    1   135.38 161.38
## - SF    1   137.85 163.85
## - TB    1   137.93 163.93
## - R     1   139.07 165.07
## - X1B   1   149.32 175.32
## - WAA   1   157.86 183.86
##
## Step:  AIC=157.7
```

```
## HOF.Status ~ R + X1B + X2B + SB + CS + BB + SO + OPS. + TB +
##     SF + WAA + Rbat
##
##        Df Deviance    AIC
## - X2B   1   132.04 156.04
## - SO    1   132.05 156.05
## - BB    1   132.58 156.58
## - Rbat  1   133.62 157.62
## <none>      131.70 157.70
## - OPS.  1   134.06 158.06
## - CS    1   135.18 159.18
## - SB    1   135.38 159.38
## - SF    1   137.96 161.96
## - TB    1   137.96 161.96
## - R     1   139.08 163.08
## - X1B   1   149.44 173.44
## - WAA   1   158.02 182.02
##
## Step:  AIC=156.04
## HOF.Status ~ R + X1B + SB + CS + BB + SO + OPS. + TB + SF + WAA +
##     Rbat
##
##        Df Deviance    AIC
## - SO    1   132.33 154.33
## - BB    1   132.72 154.72
## - Rbat  1   133.78 155.78
## <none>      132.04 156.04
## - OPS.  1   134.25 156.25
## - CS    1   135.54 157.54
## - SB    1   135.62 157.62
## - SF    1   138.16 160.16
## - TB    1   138.44 160.44
## - R     1   139.15 161.15
## - X1B   1   149.72 171.72
## - WAA   1   158.36 180.36
##
## Step:  AIC=154.33
## HOF.Status ~ R + X1B + SB + CS + BB + OPS. + TB + SF + WAA +
##     Rbat
##
##        Df Deviance    AIC
## - BB    1   133.38 153.38
## - Rbat  1   134.07 154.07
## <none>      132.33 154.33
## - OPS.  1   134.42 154.42
## - CS    1   135.54 155.54
## - SB    1   135.74 155.74
## - SF    1   139.03 159.03
## - R     1   139.36 159.36
## - TB    1   142.08 162.08
## - WAA   1   158.54 178.54
```

```
## - X1B   1   159.97 179.97
##
## Step:  AIC=153.38
## HOF.Status ~ R + X1B + SB + CS + OPS. + TB + SF + WAA + Rbat
##
##        Df Deviance    AIC
## - Rbat  1   134.47 152.47
## - OPS.  1   135.04 153.04
## <none>      133.38 153.38
## - SB    1   136.26 154.26
## - CS    1   136.70 154.70
## - SF    1   139.11 157.11
## - R     1   139.66 157.66
## - TB    1   142.34 160.34
## - WAA   1   159.00 177.00
## - X1B   1   160.01 178.01
##
## Step:  AIC=152.47
## HOF.Status ~ R + X1B + SB + CS + OPS. + TB + SF + WAA
##
##        Df Deviance    AIC
## - OPS.  1   135.24 151.24
## <none>      134.47 152.47
## - SB    1   137.54 153.54
## - CS    1   137.58 153.58
## - SF    1   140.41 156.41
## - TB    1   144.42 160.42
## - R     1   144.86 160.86
## - WAA   1   159.47 175.47
## - X1B   1   160.92 176.92
##
## Step:  AIC=151.24
## HOF.Status ~ R + X1B + SB + CS + TB + SF + WAA
##
##        Df Deviance    AIC
## <none>      135.24 151.24
## - SB    1   138.10 152.10
## - CS    1   138.26 152.26
## - SF    1   141.95 155.95
## - R     1   145.89 159.89
## - TB    1   149.89 163.89
## - X1B   1   164.29 178.29
## - WAA   1   169.57 183.57
```

```
remaining_predictors <- names(coef(backward_model))[-1]
print("Remaining predictors after backward selection:")
```

```
## [1] "Remaining predictors after backward selection:"
```

```
print(remaining_predictors)
```

```
## [1] "R"   "X1B" "SB"  "CS"  "TB"  "SF"  "WAA"
```

```r
# Prepare training control for hyperparameter tuning
train_control <- trainControl(method = "cv", number = 5)

# Train Logistic Regression Model with hyperparameter tuning
set.seed(123)
logistic_model <- train(as.formula(paste("HOF.Status ~", paste(remaining_predictors,
collapse = "+"))),
                        data = train_data, method = "glm",
                        family = binomial, trControl = train_control)

# Hyperparameter tuning for Decision Tree
set.seed(123)
tree_grid <- expand.grid(cp = seq(0.01, 0.1, by = 0.01))
tree_model <- train(as.formula(paste("HOF.Status ~", paste(remaining_predictors,
collapse = "+"))),
                    data = train_data, method = "rpart",
                    trControl = train_control, tuneGrid = tree_grid)



# Hyperparameter tuning for Boosted Decision Tree
set.seed(123)
boosted_grid <- expand.grid(nrounds = c(50, 100), max_depth = c(2, 4, 6),
                            eta = c(0.01, 0.1, 0.3), gamma = 0,
                            colsample_bytree = 1, min_child_weight = 1, subsample = 1)

boosted_model <- train(as.formula(paste("HOF.Status ~", paste(remaining_predictors,
collapse = "+"))),
                       data = train_data, method = "xgbTree",
                       trControl = train_control, tuneGrid = boosted_grid)


# Evaluate performance metrics
logistic_pred <- predict(logistic_model, newdata = test_data)
tree_pred <- predict(tree_model, newdata = test_data)
boosted_pred <- predict(boosted_model, newdata = test_data)

logistic_cm <- confusionMatrix(logistic_pred, test_data$HOF.Status)
tree_cm <- confusionMatrix(tree_pred, test_data$HOF.Status)
boosted_cm <- confusionMatrix(boosted_pred, test_data$HOF.Status)

print("Logistic Regression Performance:")

## [1] "Logistic Regression Performance:"

print(logistic_cm)

## Confusion Matrix and Statistics
##
##            Reference
## Prediction FALSE TRUE
##      FALSE   363    6
```

```
##      TRUE      2   14
##
##                Accuracy : 0.9792
##                  95% CI : (0.9595, 0.991)
##     No Information Rate : 0.9481
##     P-Value [Acc > NIR] : 0.001703
##
##                   Kappa : 0.767
##
##  Mcnemar's Test P-Value : 0.288844
##
##             Sensitivity : 0.9945
##             Specificity : 0.7000
##          Pos Pred Value : 0.9837
##          Neg Pred Value : 0.8750
##              Prevalence : 0.9481
##          Detection Rate : 0.9429
##    Detection Prevalence : 0.9584
##       Balanced Accuracy : 0.8473
##
##        'Positive' Class : FALSE
##
```

```r
print("Decision Tree Performance:")
```

```
## [1] "Decision Tree Performance:"
```

```r
print(tree_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   361    4
##      TRUE      4   16
##
##                Accuracy : 0.9792
##                  95% CI : (0.9595, 0.991)
##     No Information Rate : 0.9481
##     P-Value [Acc > NIR] : 0.001703
##
##                   Kappa : 0.789
##
##  Mcnemar's Test P-Value : 1.000000
##
##             Sensitivity : 0.9890
##             Specificity : 0.8000
##          Pos Pred Value : 0.9890
##          Neg Pred Value : 0.8000
##              Prevalence : 0.9481
##          Detection Rate : 0.9377
##    Detection Prevalence : 0.9481
```

```
##       Balanced Accuracy : 0.8945
##
##        'Positive' Class : FALSE
##
```

```r
print("Boosted Decision Tree Performance:")
```

```
## [1] "Boosted Decision Tree Performance:"
```

```r
print(boosted_cm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction FALSE TRUE
##      FALSE   362    2
##      TRUE      3   18
##
##                Accuracy : 0.987
##                  95% CI : (0.97, 0.9958)
##     No Information Rate : 0.9481
##     P-Value [Acc > NIR] : 5.276e-05
##
##                   Kappa : 0.8712
##
##  Mcnemar's Test P-Value : 1
##
##             Sensitivity : 0.9918
##             Specificity : 0.9000
##          Pos Pred Value : 0.9945
##          Neg Pred Value : 0.8571
##              Prevalence : 0.9481
##          Detection Rate : 0.9403
##    Detection Prevalence : 0.9455
##       Balanced Accuracy : 0.9459
##
##        'Positive' Class : FALSE
##
```

```r
# Print selected hyperparameters for Logistic Regression
print("Selected hyperparameters for Logistic Regression:")
```

```
## [1] "Selected hyperparameters for Logistic Regression:"
```

```r
print(logistic_model$bestTune)
```

```
##   parameter
## 1      none
```

```r
# Print selected hyperparameters for Decision Tree
print("Selected hyperparameters for Decision Tree:")
```

```
## [1] "Selected hyperparameters for Decision Tree:"
```

```r
print(tree_model$bestTune)

##     cp
## 3 0.03

# Print selected hyperparameters for Boosted Decision Tree
print("Selected hyperparameters for Boosted Decision Tree:")

## [1] "Selected hyperparameters for Boosted Decision Tree:"

print(boosted_model$bestTune)

##    nrounds max_depth eta gamma colsample_bytree min_child_weight subsample
## 18     100         6 0.3     0                1                1         1

# Load required libraries
library(caret)
library(rpart)
library(rpart.plot)

# Set seed for reproducibility
set.seed(123)

# Define the grid for parameter tuning
tree_grid <- expand.grid(cp = seq(0.01, 0.1, by = 0.01))

# Train the decision tree model
tree_model <- train(
  as.formula(paste("HOF.Status ~", paste(remaining_predictors, collapse = "+"))),
  data = train_data,
  method = "rpart",
  trControl = train_control,
  tuneGrid = tree_grid
)

# Extract the best model
best_tree <- tree_model$finalModel

# Plot the decision tree
#rpart.plot(best_tree)
```

Nic’s code:

```{r}
summary(mlbdata)

mlbdata$HOF.Status <- factor(mlbdata$HOF.Status)
mylogit <- glm(HOF.Status ~ R + X1B + SB + CS + TB + SF + WAA, data = mlbdata, family =
"binomial")
summary(mylogit)
```

````
confint(mylogit)
confint.default(mylogit)

# install.packages("aod")
library(aod)
wald.test(b = coef(mylogit), Sigma = vcov(mylogit), Terms = 4:6)

exp(coef(mylogit))
exp(cbind(OR = coef(mylogit), confint(mylogit)))
```

```{r}
library(readr)
# install.packages("tidymodels")
library(tidymodels)


# bank_df <- read_csv2("bank-full.csv")

mlbdataframe <- glm(HOF.Status ~ R + X1B + SB + CS + TB + SF + WAA, data = mlbdata, family =
"binomial")

mlbdata2 <- mlbdata %>%
  select(HOF.Status, R, X1B, SB, CS, TB, SF, WAA)

mlbdata2$HOF.Status <- factor(mlbdata2$HOF.Status)

# Read the dataset and convert the target variable to a factor
# mlbdata <- mlbdata
# mlbdata$HOF.Status = as.factor(mlbdata$HOF.Status)

# Split data into train and test
set.seed(421)
split <- initial_split(mlbdata, prop = 0.8, strata = HOF.Status)
train <- split %>%
	training()
test <- split %>%
	testing()

library(parsnip)
# install.packages("glmnet")
library(glmnet)

# Train a logistic regression model
model <- logistic_reg(mixture = double(1), penalty = double(1)) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(HOF.Status ~ ., data = mlbdata)
````

```
# Model summary
tidy(model)

# Class Predictions
pred_class <- predict(model,
                      new_data = test,
                      type = "class")

# Class Probabilities
pred_proba <- predict(model,
                      new_data = test,
                      type = "prob")
results <- test %>%
            select(HOF.Status) %>%
            bind_cols(pred_class, pred_proba)

accuracy(results, truth = HOF.Status, estimate = .pred_class)


# Train a logistic regression model
model <- logistic_reg(mixture = double(1), penalty = double(1)) %>%
  set_engine("glmnet") %>%
  set_mode("classification") %>%
  fit(HOF.Status ~ ., data = mlbdata2)

# Model summary
tidy(model)

# Class Predictions
pred_class <- predict(model,
                      new_data = test,
                      type = "class")

# Class Probabilities
pred_proba <- predict(model,
                      new_data = test,
                      type = "prob")
results <- test %>%
            select(HOF.Status) %>%
            bind_cols(pred_class, pred_proba)

accuracy(results, truth = HOF.Status, estimate = .pred_class)


# Define the logistic regression model with penalty and mixture hyperparameters
log_reg <- logistic_reg(mixture = tune(), penalty = tune(), engine = "glmnet")
```

```
# Define the grid search for the hyperparameters
grid <- grid_regular(mixture(),  penalty(), levels = c(mixture = 4, penalty = 3))

# Define the workflow for the model
log_reg_wf <- workflow() %>%
  add_model(log_reg) %>%
  add_formula(HOF.Status ~ .)

# Define the resampling method for the grid search
folds <- vfold_cv(train, v = 5)

# Tune the hyperparameters using the grid search
log_reg_tuned <- tune_grid(
  log_reg_wf,
  resamples = folds,
  grid = grid,
  control = control_grid(save_pred = TRUE)
)

select_best(log_reg_tuned, metric = "roc_auc")
```

A tibble: 1 x 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
accuracy	binary	0.9651163

1 row

A tibble: 1 x 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
accuracy	binary	0.9767442

term <chr>	estimate <dbl>	penalty <dbl>
(Intercept)	-7.973063e+00	0
R	8.261204e-04	0
X1B	1.538401e-03	0
SB	-5.356757e-05	0
CS	-4.029762e-03	0
TB	5.670251e-04	0
SF	4.427216e-03	0
WAA	5.361382e-02	0

```{r}
# Fit the model using the optimal hyperparameters
log_reg_final <- logistic_reg(penalty = 0.0000000001, mixture = 0) %>%
                        set_engine("glmnet")  %>%
                        set_mode("classification")  %>%
                        fit(HOF.Status~., data = mlbdata2)
```

```
# Evaluate the model performance on the testing set
pred_class <- predict(log_reg_final,
                      new_data = test,
                      type = "class")
results <- test %>%
  select(HOF.Status) %>%
  bind_cols(pred_class, pred_proba)

# Create confusion matrix
conf_mat(results, truth = HOF.Status,
         estimate = .pred_class)
library(yardstick)
precision(results,  truth = HOF.Status,
          estimate = .pred_class)

recall(results,  truth = HOF.Status,
       estimate = .pred_class)

# coeff <- tidy(log_reg_final) %>%
#   arrange(desc(abs(estimate)))  %>%
#   filter(abs(estimate) > 0.5)
#
# ggplot(coeff, aes(x = term, y = estimate,  fill = term)) + geom_col()  + coord_flip()
```

```
                 Truth
Prediction FALSE TRUE
     FALSE   247    4
     TRUE      2    5
```

A tibble: 1 x 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
precision	binary	0.9840637

A tibble: 1 x 3

.metric <chr>	.estimator <chr>	.estimate <dbl>
recall	binary	0.9919679

```{r}

mlbdata2
```

```
ggplot(mlbdata2, aes(R, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(X1B, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(SB, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(CS, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(TB, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(SF, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

ggplot(mlbdata2, aes(WAA, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

# Load necessary packages
library(ggplot2)
# install.packages("gridExtra")
library(gridExtra)

# Create individual plots
plot1 <- ggplot(mlbdata2, aes(R, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

plot2 <- ggplot(mlbdata2, aes(X1B, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

plot3 <- ggplot(mlbdata2, aes(SB, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

plot4 <- ggplot(mlbdata2, aes(CS, fill = HOF.Status)) +
        geom_bar() +
```

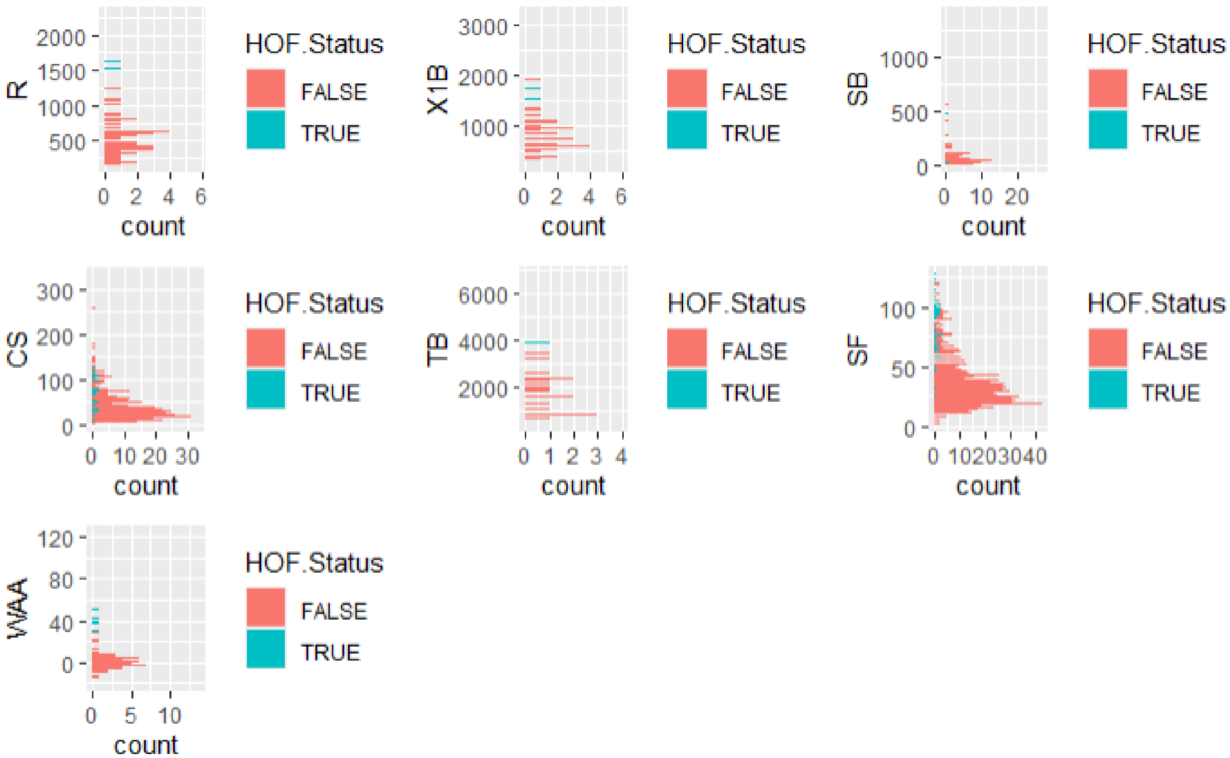
```
        coord_flip()

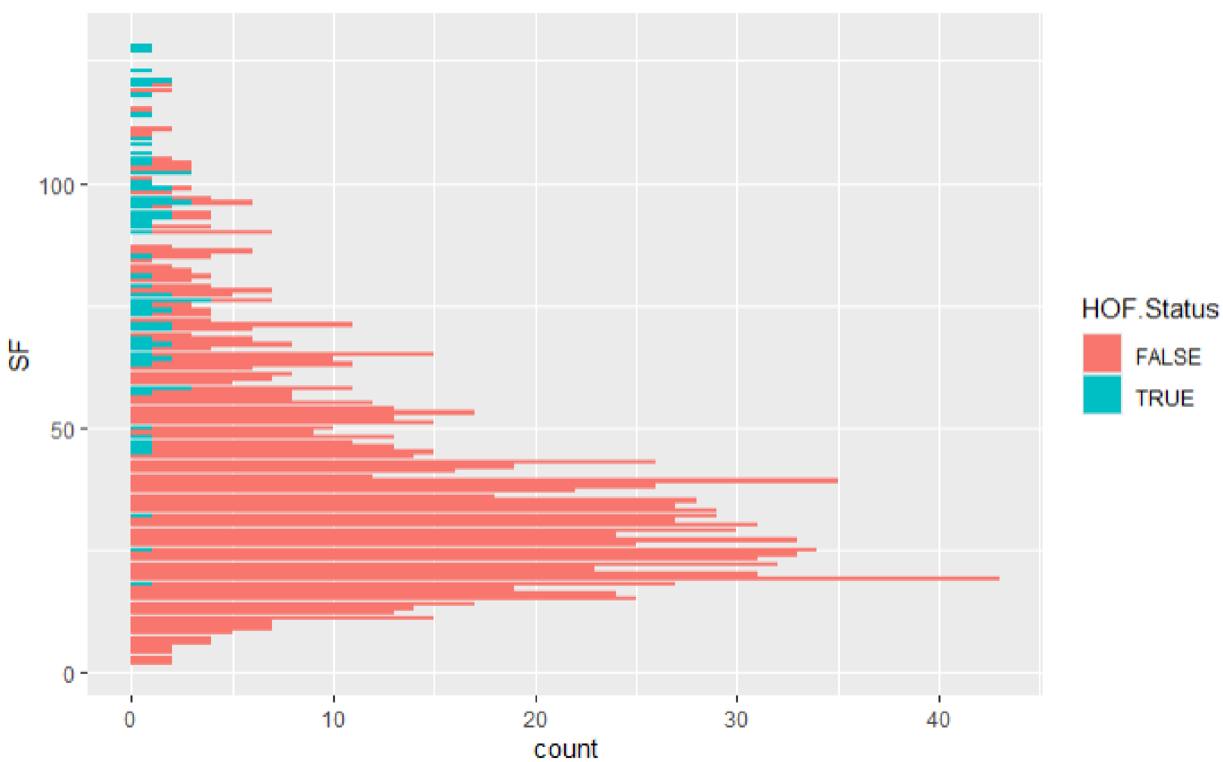
plot5 <- ggplot(mlbdata2, aes(TB, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

plot6 <- ggplot(mlbdata2, aes(SF, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

plot7 <- ggplot(mlbdata2, aes(WAA, fill = HOF.Status)) +
        geom_bar() +
        coord_flip()

# Arrange plots in a grid layout
grid.arrange(plot1, plot2, plot3, plot4, plot5, plot6, plot7, ncol = 3)
```





---
title: "Hall of Fame Status Prediction"
author: "John Mayfield"

````
date: "`r Sys.Date()`"
output: html_document
---

```{r setup, include=FALSE}
#knitr::opts_chunk$set(echo = TRUE)
```



#Load the data
```{r data}
library(ggplot2)
library(caret)
library(randomForest)
library(ROCR)
library(rpart)
library(rpart.plot)

MLB_data <- read.csv("MLB_Player_Data.csv")
MLB_data <- na.omit(MLB_data)

#Only include relevant columns
#MLB_data <- MLB_data[, names(MLB_data) %in% c("R", "X1B", "SB", "CS", "TB", "SF", "WAA")]

#set up dataframe
MLB_data <- as.data.frame(MLB_data)
dim(MLB_data)
#str(MLB_data)

#Make HOF status dependant variable
MLB_data$HOF.Status <- as.factor(MLB_data$HOF.Status)


```

##Simple Decision Tree

```{r Simple Decision Tree}
#First the data must be shuffled or it will not perform properly
head(MLB_data)
tail(MLB_data)

shuffle_index <- sample(1:nrow(MLB_data))

MLB_data2 <- read.csv("MLB_Player_Data.csv")
MLB_data2 <- MLB_data2[shuffle_index, ]
head(MLB_data2)
````
````

```
#Only keep important variables
MLB_data2 <- MLB_data2[, names(MLB_data2) %in% c("HOF.status","R","X1B", "CS", "TB", "WAR",
"WAA", "oWAR", "H", "PA", "AB", "Rbat", "G", "IBB", "xbh", "RBI", "Last.Season")]
head(MLB_data2)

# Split the data for training and testing
create_train_test <- function(MLB_data2, size = 0.7, train = TRUE) {
  n_row <- nrow(MLB_data2)
  total_row <- floor(size * n_row)
  train_sample <- 1:total_row

  if (train) {
    return(MLB_data2[train_sample, ])
  } else {
    return(MLB_data2[-train_sample, ])
  }
}

#Check to ensure successful split
data_train <- create_train_test(MLB_data2, 0.75, train = TRUE)
data_test <- create_train_test(MLB_data2, 0.75, train = FALSE)
head(data_train)
head(data_test)
dim(data_train)
dim(data_test)

#Check randomization
prop.table(table(data_train$WAR))
prop.table(table(data_test$WAR))

#HOF.status is returning'NULL'. So I instead ran this tree based on WAR, since it is the best predictor of
HOF.status = TRUE
#fit <- rpart(data_train$WAR~., data = data_train, method = "anova")
fit <- rpart(data_train$WAR~., data = data_train, method = "anova", control = rpart.control(cp = 0.01,
maxcompete = 0, maxsurrogate = 0))
rpart.plot(fit)

fit2 <- rpart(data_test$WAR~., data = data_test, method = "anova", control = rpart.control(cp = 0.01,
maxcompete = 0, maxsurrogate = 0))
rpart.plot(fit2)Paper Title* (use style: paper title)
```