STAT 3341 Data Assignment #2

Cole Wagner (2153 words)

Table of contents

1 Abstract

This study seeks to create a win probability model for Major League Baseball (MLB) games. These models are crucial for sports gambling organizations, as knowing a team's probability of winning helps these organizations set point spreads and money lines. To accomplish this, I created basic logistic regression, elastic net logistic regression, and decision tree models. These models used various features to predict the probability of a team winning at any point in the game. I found that basic logistic regression was the best modeling framework in this case. I was able to make a useful model that predicted the correct outcome over 75% of the time. Unfortunately, the model did not teach me much about important factors in predicting the probability of a win, as it relied heavily on the post-pitch score of the game. Thus, it most often predicted the team that was currently winning to win in the end, which is intuitive but also not very interesting.

2 Introduction

The Major League Baseball (MLB) was founded in 1903 when the two largest professional baseball organizations at the time (the National League and the American League) merged. The best team from each league played in the World Series, which is still the name of the MLB's championship series. Baseball gained wide popularity in the U.S. and is still warmly referred to as "America's pastime". In 1947, the Brooklyn Dodgers hired Allan Roth, the first statistician to work for an MLB team. Since then, technology to collect and analyze data has exponentially improved, and MLB teams heavily rely on statisticians to make key in-game decisions like where a batter should be pitched to and where the fielders should play for each batter. This truly began taking form when Billy Beane, the former General Manager for the Oakland Athletics, embraced data analytics to find undervalued players and create a great team with a low salary cap. His controversial approach succeeded, and the 2002 Oakland Athletics outperformed their previous regular season record despite losing three of their best players to free agency. This anomaly was documented by author Michael Lewis, who later published a dramatized version of the season in his bestselling book *Moneyball.* The popularity of this book brought baseball analytics into the public eye, and every MLB team has embraced a scientific, analytical approach to baseball since then.

One key application of statistics in baseball is the creation of win probability models. These models produce the predicted probability of a team winning at any point in a game. These models are particularly interesting for baseball because of the sheer amount of data that is collected every single pitch. Additionally, because there are so many pitches per game and games per year, there is more data to analyze in baseball than any other sport. The result is more complex models that have higher predictive ability. In this study, I utilized logistic regression and tree-based modeling frameworks to create win probability models for MLB games from 07/23/2020 to 09/27/2020.

3 Methods

3.1 Data Overview

The dataset for this study was composed of statcast data scraped from the Baseball Savant website. As mentioned above, the dataset contained games from 07/23/2020 to 09/27/2020 and contained 66,434 rows and 91 columns with each row corresponding to an "event" in the game. Events were typically the conclusion of an at-bat, and the three most common events were a field out, a strikeout, and a single. Grouping these events into games, there were a total of 898 unique games to analyze in the data. To see the distribution of at-bats per game, I create the following histogram:

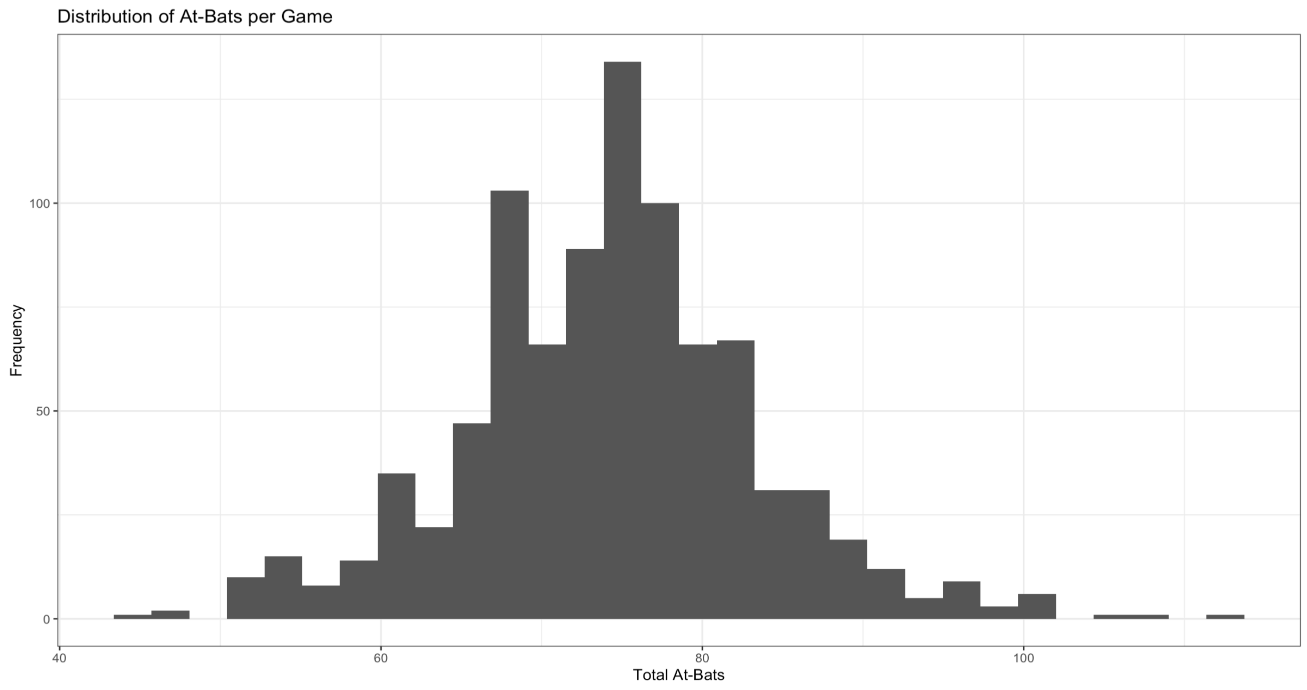


Figure 3.1: At-Bats Distribution Plot

From this plot, we can see there there were just under 80 at-bats per game on average, with very few games having over 100 at-bats or under 50 at-bats.

To begin my analysis, I selected all relevant variables and removed the rest. Thus, I was left with 30 columns. These contained the outcome (whether or not a team won the game) and 29 predictors describing various aspects of each pitch such as velocity, current balls/strikes for the at-bat, and the outcome of the pitch.

In order to organize games chronologically, I converted the game date variable from a character to a date variable. With this, I was able to create a train and set split in the data. The training data contained all games on or before 09/13/2020 and the testing data contained all games after this date.

For each of the modeling paradigms discussed below, the data was preprocessed by setting the game's date to an ID variable, dummy encoding all categorical predictors, normalizing all numeric predictors, and removing all predictors with a Pearson's correlation of 0.75 or higher. This was done via the `step_corr()` function, which takes the highly correlated variable pair, compares their average correlations to every other predictor in the dataset, and keeps only the predictor with the lower average correlation. The only variation in this preprocessing was that categorical variables were one-hot encoded for tree-based methods, but not for regression-based methods.

3.2 Statistical Models

In total, three different models were considered in this analysis: basic logistic regression, penalized logistic regression, and a decision tree. Both AUC and Brier Score were used to compare models.

3.2.1 Basic Logisitic Regression

Logistic regression is a classical method for predicting the probability of binary outcomes. It is similar to linear regression in that it is composed of a combination of coefficients multiplied to predictors. However, it differs from linear regression in its outcome. The outcome of a logistic regression model is the log of the odds of the outcome. The odds of an event are described as "the ratio of the probability of the event happening divided by the probability of the event not happening" (LaValley).

For this study, a logistic regression model was fit to the full training set, and model performance metrics were obtained from the testing set.

3.2.2 Elastic Net Logistic Regression

Elastic Net Logistic Regression is similar to basic logistic regression, but it adds a penalty that is a combination of the LASSO and ridge penalty (Algamal and Lee) with two tuning parameters to tune the weights of these two penalties. As a result, there is no closed-form solution to the model, and multiple tuning parameter combinations must be tested to find a model with sufficient predictive ability. In addition, elastic net can shrink coefficients to reduce their weight on the outcome and can even shrink coefficients to exactly zero, removing those predictors from the model. Like basic logistic regression, the outcome is the log of the odds of the event.

In this study, five-fold cross validation was used to tune the model, with five tuning parameter combinations selected via latin hypercube sampling. The tuning parameter pairs were judged by their resulting cross-validated AUCs, with the best model being the one with the highest AUC. This "best" model was then fit to the test set to obtain performance metrics.

3.2.3 Decision Tree

A decision tree is much different from logistic regression in that it splits up the predictor space into mutually exclusive regions by creating nodes or split points at certain values of different predictors. In the case of classification, the tree predicts the class that is most common in that region of the given predictor space (Song and Lu). Decision tree models have a variety of tuning parameters such as the maximum depth of the tree and the minimum number of observations per predictor space region.

For this study, I used five-fold cross validation to evaluate ten tuning parameter combinations that were selected via latin hypercube sampling. I chose to tun the maxiumum tree depth, the minimum number of observations per division of the predictor space, and the cost-complexity parameter. Models were evaluated based on mean cross-validated AUC, and the model with the highest AUC was trained on the full training set, and performance metrics were collected on the test set.

4 Results

4.1 Basic Logistic Regression

After creating the basic logistic regression model, I observed the following coefficient estimates:

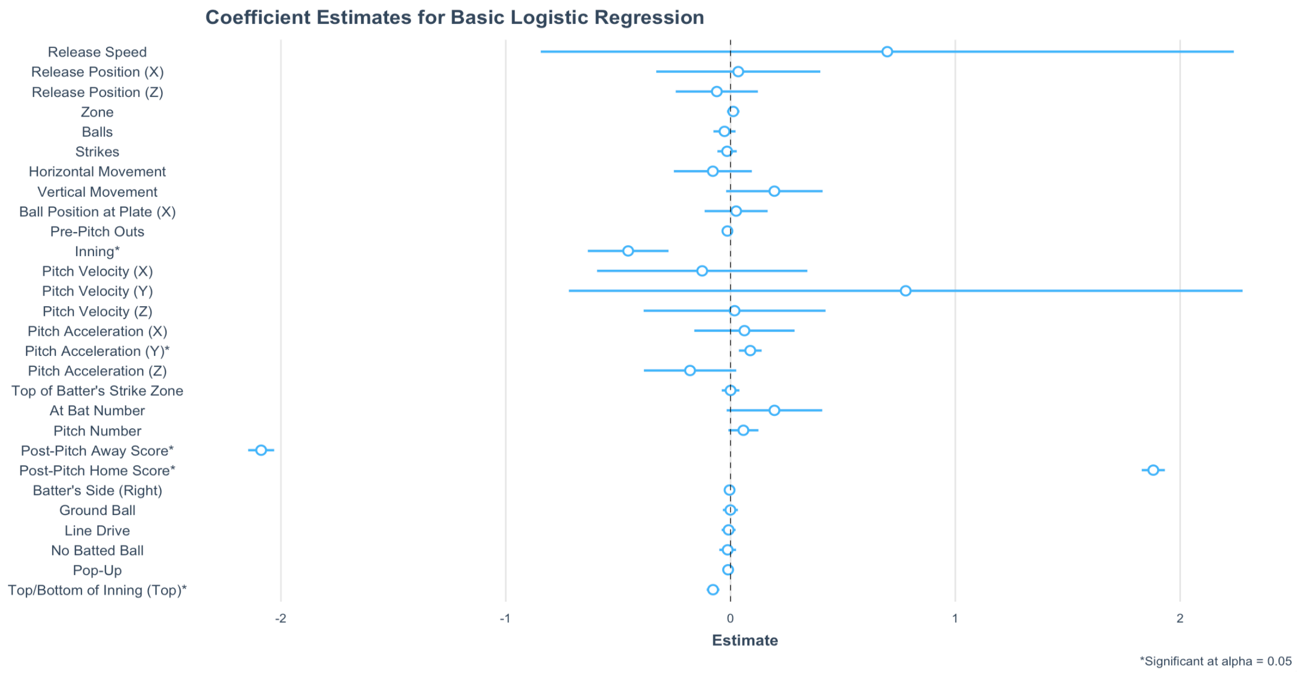


Figure 4.1: Basic Logistic Regression Coefficient Plot

It is important to note that only the coefficients marked with an asterisk above are statistically significant at the 0.05 level of significance. The statistically significant predictors with a positive effect on winning are pitch acceleration in the Y axis and the post-pitch home score. Inning, post-pitch away score, and top of the inning are all negatively associated with winning and are statistically significant.

To further investigate the model's performance, I created a graph of predicted probabilities by true outcome. For outcomes that are truly losses, I was hoping that the model would predict low probabilities of winning and vice versa.

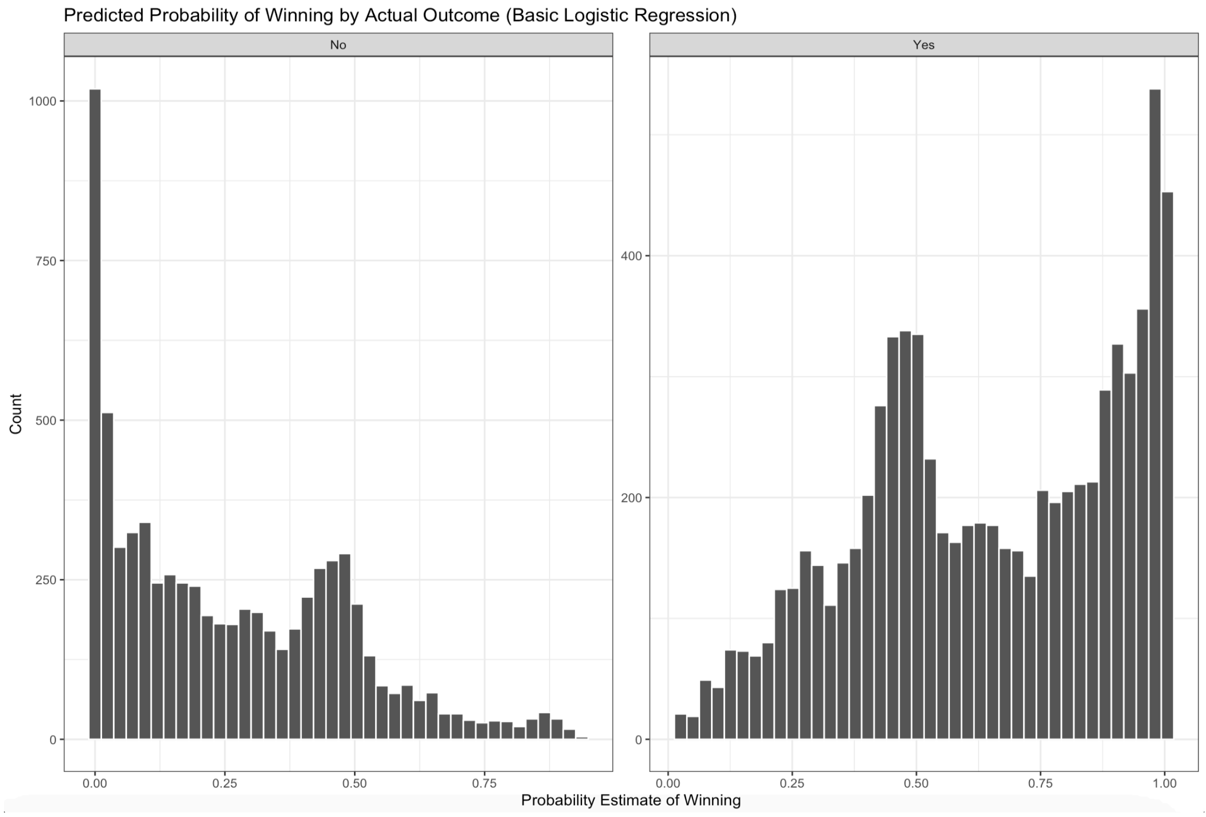


Figure 4.2: Basic Logistic Regression Probability Plot

Ideally, the graph on the left would be extremely right skewed, and the right graph would be extremely left skewed. The left graph is very right skewed, and shows that the model has strong performance in predicting losses. On the other hand, the right graph is only moderately left skewed, showing that the model is weaker in predicting wins. It is by no means a bad model, but it certainly has limitations when it comes to predicting high probabilities for winning teams.

4.2 Elastic Net Logistic Regression

After tuning the model with five-fold cross validation, I found that the optimal penalty was 0.00000000143 and the optimal mixture was 0.849. This shows that the model favors the LASSO penalty, taking a more aggressive route in shrinking and removing coefficients. In addition, the penalty term is very small, which means the model has a large amount of shrinkage.

From this tuning parameter combination, all predictors were retained except strikes, horizontal movement on pitch, vertical position of the ball as it crosses the plate, acceleration in the Z dimension, pitch being in the top of the strike zone, and plate appearance number. To begin exploring model performance, I created the same probability plot as was created for the basic logistic regression.

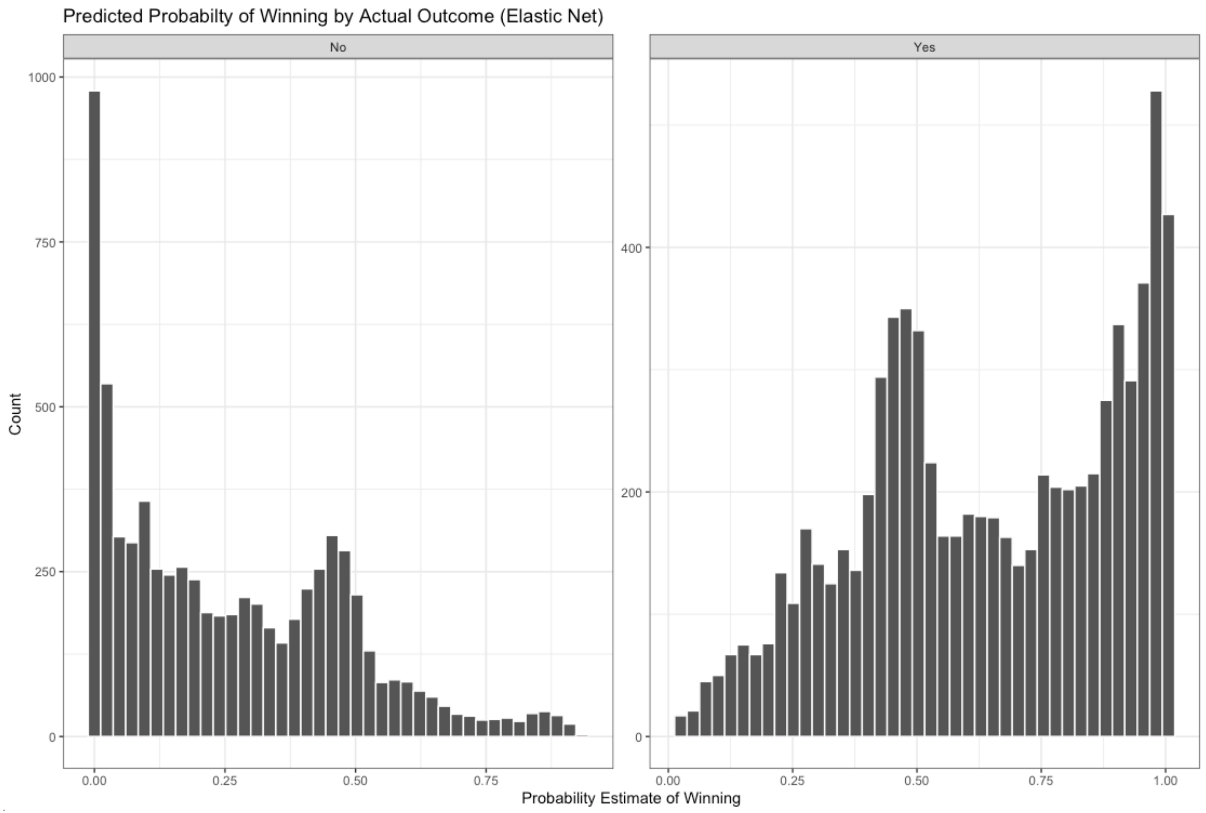


Figure 4.3: Elastic Net Logistic Regression Probability Plot

As we can see from the above figure, the elastic net model performs very similarly to the basic logistic regression model, and has the same strengths and weaknesses.

4.3 Decision Tree

For my decision tree model, I found an optimal cost complexity of 0.000000254, tree depth of 11, and minimum observations per split of 34. It is difficult to put much weight in these parameters, as decision trees are very high variance and the parameter combination would have likely been very different given slightly different data. A variable importance plot given these tuning parameters is shown below:

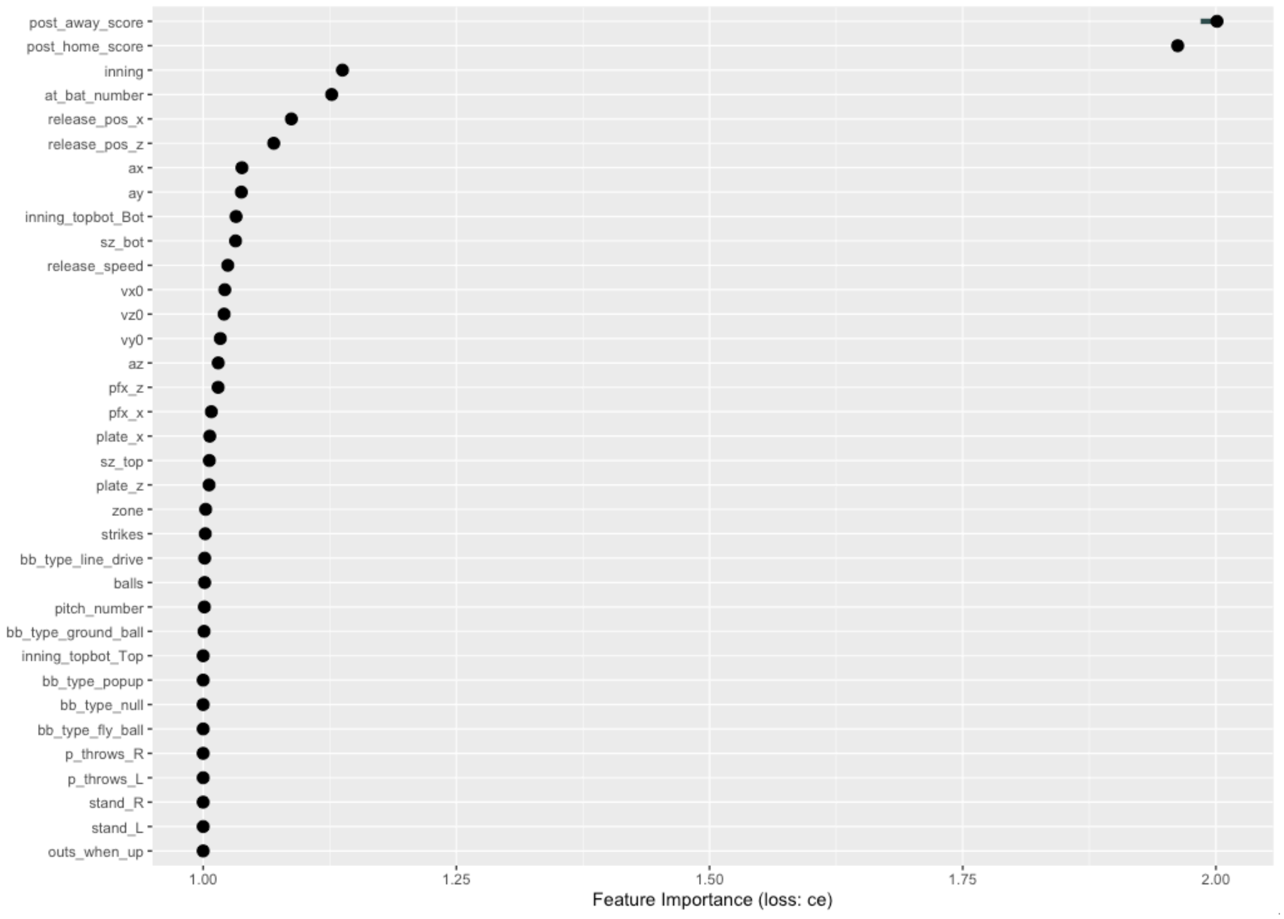
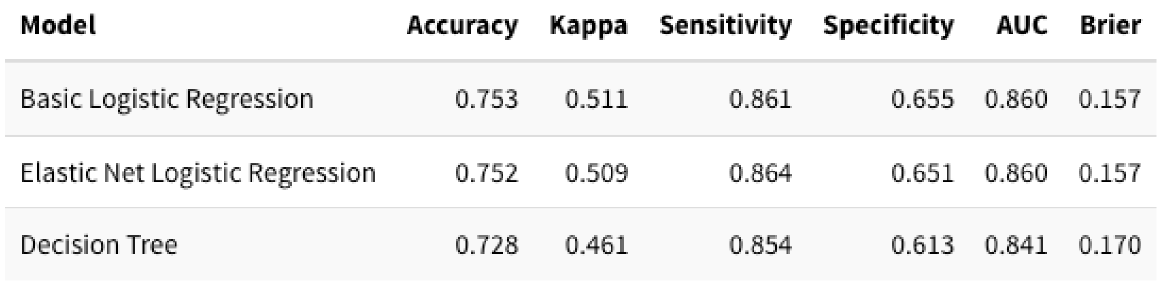


Figure 4.4: Decision Tree Variable Importance Plot

From the figure above, we can see that post-pitch home and away scores are the most important variables by a wide margin. This is intuitive, as the score after any given pitch would be a very strong indicator of which team will win. The next two most important variables are inning and at-bat number. I would expect these variables to work well with the post-pitch scores, as these scores would be more indicative of the final outcome the closer the game is to ending.

4.4 Model Comparison

To compare the performance of all three models, I created the following table using a suite of metrics designed to empirically measure the efficacy of classification models.



Model	Accuracy	Kappa	Sensitivity	Specificity	AUC	Brier
Basic Logistic Regression	0.753	0.511	0.861	0.655	0.860	0.157
Elastic Net Logistic Regression	0.752	0.509	0.864	0.651	0.860	0.157
Decision Tree	0.728	0.461	0.854	0.613	0.841	0.170

From the above table, we can see that the basic and elastic net logistic regression models are nearly identical in performance, with the decision tree lagging far behind. While the top two models are difficult to distinguish performance-wise, I would choose the basic logistic regression as the best model in this case. The factors behind this decision are discussed below.

5 Discussion

Overall, the best model in this study was the basic logistic regression because, although it had nearly identical metrics to the elastic net model, it is the simpler modeling framework of the two and is thus more desirable. Base logistic regression has more interpretability because you can calculate standard errors and p-values for model coefficients. This model is certainly useful, as it has over 75% accuracy and a kappa of 0.511, which is above the accepted threshold for a useful model of 0.3. In addition, the model has a strong AUC of 0.860, and good sensitivity. The model's biggest weakness is its mediocre specificity. This is unsurprising considering the probability plot from section 4.1. The model has some difficulty confidently predicting that a team will win, and often gives winning teams close to a 0.5 probability of actually winning. This does not disqualify the model from being useful, though. The model seems to highly rely on post-pitch scores when making predictions on the final outcome. While this makes sense, it seems that the model can't account for other factors that may lead to comebacks by the losing team. To improve the model, we could add more predictors to allow the model to find more nuanced trends in the data. There are more than enough observations in this dataset to support a much larger predictor space.

6 References

Algamal, Z. Y., & Lee, M. H. (2015). Applying Penalized Binary Logistic Regression with Correlation Based Elastic Net for Variables Selection. *Journal of Modern Applied Statistical Methods*, 14(1), 168–179.
https://doi.org/10.22237/jmasm/1430453640

LaValley, M. P. (2008). Logistic regression. *Circulation*, 117(18), 2395–2399.
https://doi.org/10.1161/circulationaha.106.682658

Lealos, S. S., & Heyssel, P. (2023, November 1). *Moneyball True Story: How Accurate the Baseball Movie Is.*
ScreenRant. https://screenrant.com/moneyball-true-story-baseball-movie-how-accurate/

Messier, A. (2023, September 14). *History of the MLB: From Early Baseball Beginnings to Monumental Moments.* Fox News.
https://www.foxnews.com/sports/mlb-baseball-history

Sabermetrics: Baseball Analytics and the Science of Winning [Infographic]. Syracuse University. (n.d.). https://onlinegrad.syracuse.edu/blog/sabermetrics-baseball-analytics-the-science-of-winning/

SONG, Y., & LU, Y. (2015). Decision tree methods: applications for classification and prediction.
Shanghai Arch Psychiatry, 27(2), 130–135. https://doi.org/10.11919/j.issn.1002-0829.215044

7 Appendix

```r
library(tidyverse)
library(tidymodels)
library(doParallel)
library(iml)
library(rpart)
library(rpart.plot)
library(jtools)
library(kableExtra)

statcast <- read.csv("/Users/colewagner632/Library/CloudStorage/OneDrive-SouthernMethodist
statcast.names <- colnames(statcast)[c(2:5, 15, 18:19, 24:26, 28:31, 35:37,
                                       45:52, 77:78, 84:85, 91)]
statcast2 <- statcast %>%
  select(all_of(statcast.names))
```

7.1 Data Preprocessing

```r
# number of unique games
n_distinct(statcast$game_pk)

# numeric summary of at bats per game
# filter down to the total number of at bats for each game
ab_per_game <- statcast %>%
  group_by(game_pk) %>%
  summarise(
    total_ab = max(at_bat_number)
  ) %>%
  ungroup()

# frequency table for events
table(statcast$events)
```

```r
# set game_date as a date variable
# make outcome a factor
statcast2 <- statcast2 %>%
  mutate(game_date = as.Date(game_date, format = "%m/%d/%y"),
         win = factor(ifelse(home.win == 1, "Yes", "No"),
                      levels = c("No", "Yes"))) %>%
  select(-home.win)

# train/test split
tidy_split <- make_splits(statcast2 %>% filter(game_date <= '2020-09-13'),
                          statcast2 %>% filter(game_date > '2020-09-13'))

statcast.tidy.train <- training(tidy_split)
statcast.tidy.test <- testing(tidy_split)
```

7.2 Basic Logistic Regression

```r
base_rec <- recipe(win ~., data = statcast.tidy.train) %>%
  update_role(game_date, new_role = "id") %>%
  step_dummy(all_nominal_predictors()) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_corr()

base_model <- logistic_reg() %>%
                 set_mode("classification") %>%
                 set_engine("glm")

base_wf <- workflow() %>%
  add_recipe(base_rec) %>%
  add_model(base_model)

base_coefs <- base_wf %>%
  fit(statcast.tidy.train) %>%
  extract_fit_parsnip() %>%
  tidy()

# fit model to test data
base_fit <- last_fit(base_wf, tidy_split,
                     metrics = metric_set(accuracy, roc_auc, kap, sens, spec, brier_class)
```

```r
base_metrics <- base_fit %>% collect_metrics()

# get predictions
base_pred <- collect_predictions(base_fit)

# fit ROC curve
base_roc <- roc_curve(base_pred, truth = win, .pred_Yes, event_level = "second") %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_bw()

# probabilities faceted by true outcome
base_probs_facet <- base_pred %>%
  ggplot(aes(.pred_Yes)) +
  geom_histogram(col = "white", bins = 40) +
  facet_wrap(~ win, ncol = 2, scales = "free") +
  theme_bw() +
  labs(x = "Probability Estimate of Winning", y = "Count", title = "Predicted Probability

base_native_fit <- extract_fit_engine(base_fit)

base_coef_viz <- plot_summs(base_native_fit, coefs = c("Release Speed" = "release_speed",
                                    "Release Position (Z)" = "release_pos_z", "Zone" = "z
                                    "Strikes" = "strikes", "Horizontal Movement" = "pfx_x
                                    "Ball Position at Plate (X)" = "plate_x", "Ball Posit
                                    "Pre-Pitch Outs" = "outs_when_up", "Inning*" = "innin
                                    "Pitch Velocity (Y)" = "vy0", "Pitch Velocity (Z)" =
                                    "Pitch Acceleration (X)" = "ax", "Pitch Acceleration
                                    "Pitch Acceleration (Z)" = "az", "Top of Batter's Str
                                    "Bottom of Batter's Strike Zone*" = "sz_bottom", "At
                                    "Pitch Number" = "pitch_number", "Post-Pitch Away Sco
                                    "Post-Pitch Home Score*" = "post_home_score", "Batter
                                    "Pitcher's Handedness (Right)" = "p_throws_R*", "Grou
                                    "Line Drive" = "bb_type_line_drive", "No Batted Ball"
                                    "Pop-Up" = "bb_type_popup", "Top/Bottom of Inning (To
    labs(title = "Coefficient Estimates for Basic Logistic Regression", caption = "*Signif


# see which variables are significant at alpha = 0.05
```

```r
base_coefs$term[which(base_coefs$p.value <= 0.05)]
```

7.3 Elastic Net Logistic Regression

```r
# create 5-fold cv object
set.seed(3341)
pen_kfold = vfold_cv(statcast.tidy.train, v = 5, strata = win)

en_model <- logistic_reg(penalty = tune(), mixture = tune()) %>%
  set_mode("classification") %>%
  set_engine("glmnet")

en_wf <- workflow() %>%
  add_recipe(base_rec) %>%
  add_model(en_model)

#Tune the model
set.seed(3341)
en_tune <- tune_grid(
  en_wf,
  resamples = pen_kfold,
  grid = 5,
  metrics = metric_set(accuracy, roc_auc))
```

```r
#Extract the best tuning parameter combination
en_param <- en_tune %>% select_best(metric = "roc_auc")

#Finalize our models
en_model2 <- finalize_model(en_model, en_param)

#Note that we need to update our workflow
en_wf2 <- en_wf %>%
  update_model(en_model2)

# get model coefficents
en_coefs <- en_wf2 %>%
  fit(statcast.tidy.train) %>%
  extract_fit_parsnip() %>%
  tidy()
```

```r
# fit model to test data
en_fit <- last_fit(en_wf2, tidy_split,
                   metrics = metric_set(accuracy, roc_auc, kap, sens, spec, brier_class))

en_metrics <- en_fit %>% collect_metrics()

# get predictions
en_pred <- collect_predictions(en_fit)

# fit ROC curve
en_roc <- roc_curve(en_pred, truth = win, .pred_Yes, event_level = "second") %>%
  ggplot(aes(x = 1 - specificity, y = sensitivity)) +
  geom_path() +
  geom_abline(lty = 3) +
  coord_equal() +
  theme_bw()

# probabilities faceted by true outcome
en_probs_facet <- en_pred %>%
  ggplot(aes(.pred_Yes)) +
  geom_histogram(col = "white", bins = 40) +
  facet_wrap(~ win, ncol = 2, scales = "free") +
  theme_bw() +
  labs(x = "Probability Estimate of Winning", y = "Count", title = "Predicted Probabilty o
```

7.4 Decision Tree

```r
# same at previous recipe, but one hot encode now
tree_rec <- recipe(win ~., data = statcast.tidy.train) %>%
  update_role(game_date, new_role = "id") %>%
  step_dummy(all_nominal_predictors(), one_hot = T) %>%
  step_normalize(all_numeric_predictors()) %>%
  step_corr()

tree_juice <- juice(prep(tree_rec))

# create decision tree model
tree_model <- decision_tree(tree_depth = tune(),
                            min_n = tune(),
                                        cost_complexity = tune()) %>%
                    set_mode("classification") %>%
                    set_engine("rpart")

# create workflow
tree_wf <- workflow() %>%
  add_recipe(tree_rec) %>%
  add_model(tree_model)

# get tuning parameter grid
tree_param <- extract_parameter_set_dials(tree_model)

glh_tree <- grid_latin_hypercube(tree_param, size=10)

# tune model
# ncores <- 7
# cl <- makeCluster(ncores)
# registerDoParallel(cl)
#
# tree_tune <- tune_grid(
#   tree_wf,
#   resamples = pen_kfold,
#   grid = glh_tree)
# stopCluster(cl)
#
# saveRDS(tree_tune, '/Users/colewagner632/Library/CloudStorage/OneDrive-SouthernMethodist
tree_tune <- readRDS('/Users/colewagner632/Library/CloudStorage/OneDrive-SouthernMethodist
```

```r
# update workflow
tree_best_params <- tree_tune %>% select_best(metric = "roc_auc")

tree_model2 <- finalize_model(tree_model, tree_best_params)

#Note that we need to update our workflow
tree_wf2 <- tree_wf %>%
  update_model(tree_model2)

tree_fit <- last_fit(tree_wf2, tidy_split,
                          metrics = metric_set(accuracy, roc_auc, kap, sens, spec, brier_c

# create a variable importance plot via the iml package
```

```
tree_native_fit <- extract_fit_engine(tree_fit)

tree_predictors <- tree_juice %>%
                dplyr::select(-game_date, -win)

predictor.tree <- Predictor$new(model = tree_native_fit,
                                  data = tree_predictors,
                                  y = tree_juice$win)

##Feature Importance
# imp_tree <- FeatureImp$new(predictor.tree, loss = "ce")
# saveRDS(imp_tree, '/Users/colewagner632/Library/CloudStorage/OneDrive-SouthernMethodistU
imp_tree <- readRDS('/Users/colewagner632/Library/CloudStorage/OneDrive-SouthernMethodistU

plot(imp_tree)

# see model performance metrics
tree_metrics <- tree_fit %>% collect_metrics()
```

7.5 At-Bats Histogram

```
# at-bats histogram
ab_hist <- ab_per_game %>%
  ggplot(aes(x = total_ab)) +
  geom_histogram() +
  theme_bw() +
  labs(x = "Total At-Bats", y = "Frequency", title = "Distribution of At-Bats per Game")
```

7.6 Model Comparison Table

```
base_metrics_form <- base_metrics %>%
  select(.metric, .estimate) %>%
  pivot_longer(-1) %>%
  pivot_wider(names_from = 1, values_from = value) %>%
  mutate(Model = "Basic Logistic Regression", .before = name) %>%
  select(-name)
```

```
en_metrics_form <- en_metrics %>%
  select(.metric, .estimate) %>%
  pivot_longer(-1) %>%
  pivot_wider(names_from = 1, values_from = value) %>%
  mutate(Model = "Elastic Net Logistic Regression", .before = name) %>%
  select(-name)

tree_metrics_form <- tree_metrics %>%
  select(.metric, .estimate) %>%
  pivot_longer(-1) %>%
  pivot_wider(names_from = 1, values_from = value) %>%
  mutate(Model = "Decision Tree", .before = name) %>%
  select(-name)


comb_metrics <- rbind(base_metrics_form, en_metrics_form, tree_metrics_form) %>%
  rename(Accuracy = accuracy, Brier = brier_class, Kappa = kap, AUC = roc_auc, Sensitivity

metrics_table <- kable(comb_metrics, digits = 3) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
metrics_table
```