Predicting NFL Game Winners

Code

Ahmad Rakha, Tyler Kurpanek, Advaith Ravishankar, Stephanie Ugochukwu, Akhil Subbarao

Introduction

The integration of data analytics into sports has revolutionized team operations in recent years and will continue to be a crucial determinant of success in the competitive landscape. From refining in-game strategies to enhancing talent recruitment, data-driven insights have become the backbone of modern sports analytics[1].

In our final project, we aim to answer the question: **"With what accuracy can we predict the winner of an NFL game using past game performance with the data from the 2019-2022 seasons?"** By attempting to predict the NFL game winner, we seek to uncover the distinguishing characteristics that set a championship contender team apart from others. Prior research utilizing Recursive Feature Elimination and logistic regression techniques has identified offensive touchdowns, rush attempts, and defensive sacks as critical predictors of success, which will be instrumental in our analysis[2]. Additionally, as the league trends towards higher-scoring games, offensive performance has become an increasingly significant predictor of Super Bowl winners[3].

Our objective is to investigate the key factors that influence a team's likelihood of winning by analyzing the *NFL Teams Stats and Outcomes Dataset* from Kaggle [4]. We will incorporate relative performance metrics, including ELO ratings [5], to enhance our predictive models. By doing so, we aim to identify the teams most likely to achieve success and provide actionable insights into the attributes that define a championship team, leveraging data to inform decisions and drive success at the highest level.

Question

With what accuracy can we predict the winner of an NFL game using using past game performance (Offensive Statistic History, Defensive Statistic History and ELO) from the 2019-2022 seasons? Does relative performance scores from the ELO rating system have an impact on predicting winners?

Load packages

Hide

```
library("skimr")
library(dplyr)
library(GGally)
library(tidyverse)
library(zoo)
```

The Data

The data comes from a dataset found on kaggle.com titled NFL Teams Stats and Outcomes [6]. The dataset focuses on the 2019-2022 NFL seasons with stats regarding rankings, offensive performance, defensive performance, and special teams stats. Home/away game results are also included.

There are four data sets separated based on the NFL season. Each data set has 22 columns and 2282 observations

1. Week: The week of the NFL season (numeric or categorical)
2. Day: The day of the week (categorical
3. Date: The date of the game (date)
4. Opp: The opponent (categorical)
5. Tm: The team (categorical)
6. Opp.1: The opponent's ranking (numeric)
7. 1stD: The team's first downs (numeric)
8. TotYd: The team's total yards (numeric)
9. PassY: The team's passing yards (numeric)
10. RushY: The team's rushing yards (numeric)
11. TO: The team's turnovers (numeric)
12. 1stD.1: The opponent's first downs (numeric)
13. TotYd.1: The opponent's total yards (numeric)
14. PassY.1: The opponent's passing yards (numeric)
15. RushY.1: The opponent's rushing yards (numeric)
16. TO.1: The opponent's turnovers (numeric
17. Offense: The team's offensive ranking (numeric)
18. Defense: The team's defensive ranking (numeric)
19. Sp. Tms: The team's special teams rank (numeric)
20. Unnamed..5: Recorded win or loss; name of column was not imported (categorical)
21. Home_team: The team who's stats are being displayed (categorical) 22: X: Row number (numeric)

Data Import

Hide

```
nfl_2019 = read.csv("Data/2019_NFL_COMBINE.csv")
nfl_2020 = read.csv("Data/2020_NFL_COMBINE.csv")
nfl_2021 = read.csv("Data/2021_NFL_COMBINE.csv")
nfl_2022 = read.csv("Data/2022_NFL_COMBINE.csv")
```

Data Wrangling

From kaggle.com, we decided to import the individual datasets for each season instead of their combined dataset, as the individual datasets contain additional columns regarding the day and week the games occurred. Because we are working with individual datasets, we have to combine them into one in order for easier data processing.

Hide

```
# combining datasets

nfl_2019 = nfl_2019 |> # convert week column into character values for combining data sets
  mutate(Week = as.character(Week))
nfl_2020 = nfl_2020 |>
  mutate(Week = as.character(Week))
nfl_2021 = nfl_2021 |>
  mutate(Week = as.character(Week))
nfl_2022 = nfl_2022 |>
  mutate(Week = as.character(Week))

nfl_combined = bind_rows( # combine datasets and add year column to signify season
  nfl_2019 |>
    mutate(Year = 2019),
  nfl_2020 |>
    mutate(Year = 2020),
  nfl_2021 |>
    mutate(Year = 2021),
  nfl_2022 |>
    mutate(Year = 2022)
)

colnames(nfl_combined)[colnames(nfl_combined) == "X"] = "ID" # name new column that numbers r
ows as "ID"
```

The following code changes the name of home_team to team because the column actually displays the name of the team who's stats are being displayed rather than which team is at home. We also changed the unnamed column to result because the column displays the results and changed from "W" and "L" to 1 and 0. In addition, we also removed all the NA values to complete the data cleaning.

Hide

```
names(nfl_combined)[names(nfl_combined) == "home_team"] <- "Team"
names(nfl_combined)[names(nfl_combined) == "Unnamed..5"] <- "result"

# Convert W/L to 1/0
nfl_combined$result <- ifelse(nfl_combined$result == "W", 1, 0)

nfl_combined_clean <- na.omit(nfl_combined)
```

As the data is cleaned, we added the elo rating. We did this by initializing all teams scores to be 800. If a team lost, their score decreases. If they won, their score increases. Depending on the difficulty of the opponent, the magnitude of increase is depicted by the following formula:

$$\text{Expected Score(A, B)} = \frac{1}{1 + 10^{\frac{\text{elo}_b - \text{elo}_a}{400}}}$$

We started at the earliest match and updated the Elo as each match pasess.

Hide

```
# This code adds the elo score to the clean data (each team starts at 800)

nfl_combined_clean <- nfl_combined_clean |>
  mutate(Date = gsub("-", " ", Date))

# Combine Year and Date into a new column with DD-MON-YYYY format
nfl_combined_clean <- nfl_combined_clean |>
  mutate(
    Full_Date = as.Date(paste(Date, Year), format = "%d %b %Y")
  )

# Sort the dataset by the new Full_Date column
nfl_combined_clean <- nfl_combined_clean |>
  arrange(Full_Date)

# Initialize Elo ratings tracking dictionary
elo_dict <- setNames(rep(800, length(unique(c(nfl_combined_clean$Home_team, nfl_combined_clea
n$Opp)))),
                     unique(c(nfl_combined_clean$Home_team, nfl_combined_clean$Opp)))

# Function to calculate expected score
expected_score <- function(elo_a, elo_b) {
  return(1 / (1 + 10 ^ ((elo_b - elo_a) / 400)))
}

# Function to update Elo ratings
update_elo <- function(elo_a, elo_b, result, k = 20) {
  # Calculate expected scores
  exp_a <- expected_score(elo_a, elo_b)

  # Update ratings based on actual results
  new_elo_a <- elo_a + k * (result - exp_a)  # Increase for winner
  new_elo_b <- elo_b + k * ((1 - result) - (1 - exp_a))  # Decrease for loser

  return(c(new_elo_a, new_elo_b))
}

# Calculate Elo ratings for each game
for (i in 1:nrow(nfl_combined_clean)) {
  home_team <- nfl_combined_clean$Home_team[i]
  away_team <- nfl_combined_clean$Opp[i]

  # Get the current Elo ratings for home and away teams from the tracking dictionary
  home_elo_pre <- elo_dict[[home_team]]
  away_elo_pre <- elo_dict[[away_team]]

  # Assign pre-game Elo ratings to the dataset
  nfl_combined_clean$Elo_Home_Pre[i] <- home_elo_pre
  nfl_combined_clean$Elo_Away_Pre[i] <- away_elo_pre


  # Update Elo ratings based on the game result
  new_elos <- update_elo(home_elo_pre, away_elo_pre, nfl_combined_clean$result[i])

  # Assign post-game Elo ratings to the dataset
  nfl_combined_clean$Elo_Home_Post[i] <- new_elos[1]
  nfl_combined_clean$Elo_Away_Post[i] <- new_elos[2]
```

```
  # Update Elo ratings in the dictionary
  elo_dict[[home_team]] <- new_elos[1]  # Update home team Elo
  elo_dict[[away_team]] <- new_elos[2]  # Update away team Elo
}


# Print the first rows of the updated dataset
print(head(nfl_combined_clean))
```

```
##    ID      Week Day  Date result                 Opp Tm Opp.1 X1stD TotYd
## 1 124 Wild Card Sat 4 Jan      0       Houston Texans 19    22    24   425
## 2 390 Wild Card Sat 4 Jan      0     Tennessee Titans 13    20    18   307
## 3 518 Wild Card Sat 4 Jan      1        Buffalo Bills 22    19    19   360
## 4 538 Wild Card Sat 4 Jan      1 New England Patriots 20    13    19   272
## 5 462 Wild Card Sun 5 Jan      0    Minnesota Vikings 20    26    19   324
## 6 559 Wild Card Sun 5 Jan      1   New Orleans Saints 26    20    22   362
##   PassY RushY TO X1stD.1 TotYd.1 PassY.1 RushY.1 TO.1 Offense Defense Sp..Tms
## 1   253   172  1      19     360     219     141    1   -1.04   -2.61    1.38
## 2   209    98  2      19     272      71     201    1   -0.84   -2.83   -0.20
## 3   219   141  1      24     425     253     172    1    2.61    1.04   -1.38
## 4    71   201  1      18     307     209      98    2    2.83    0.84    0.20
## 5   227    97  2      22     362     226     136    1    1.72   -3.22   -3.43
## 6   226   136  1      19     324     227      97    2    3.22   -1.72    3.43
##              Home_team Year  Full_Date Elo_Home_Pre Elo_Away_Pre Elo_Home_Post
## 1        Buffalo Bills 2019 2019-01-04          800          800       790.000
## 2 New England Patriots 2019 2019-01-04          800          800       790.000
## 3       Houston Texans 2019 2019-01-04          810          790       819.425
## 4     Tennessee Titans 2019 2019-01-04          810          790       819.425
## 5   New Orleans Saints 2019 2019-01-05          800          800       790.000
## 6    Minnesota Vikings 2019 2019-01-05          810          790       819.425
##   Elo_Away_Post
## 1       810.000
## 2       810.000
## 3       780.575
## 4       780.575
## 5       810.000
## 6       780.575
```

EDA

Data Distributions

The first step of the EDA is to visualize our dataset's distributions. With the following code, we can get an understanding of the mean, standard deviation, missing vlaues and quartiles of each variable:

Hide

```
skimr::skim(nfl_combined_clean)
```

Data summary

Name	nfl_combined_clean
Number of rows	895
Number of columns	28

Column type frequency:	
character	5
Date	1
numeric	22
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Week	0	1	1	12	0	22	0
Day	0	1	3	3	0	6	0
Date	0	1	5	6	0	104	0
Opp	0	1	13	24	0	35	0
Home_team	0	1	13	20	0	31	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
Full_Date	0	1	2019-01-04	2022-10-16	2020-11-22	133

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
ID	0	1	283.42	166.82	0.00	137.00	282.00	430.50	595.00	▇▇▇▇▇
result	0	1	0.50	0.50	0.00	0.00	1.00	1.00	1.00	▇▁▁▁▇
Tm	0	1	23.23	9.86	0.00	17.00	23.00	30.00	56.00	▂▇▇▃▁
Opp.1	0	1	23.03	9.89	0.00	17.00	23.00	30.00	56.00	▂▇▇▃▁
X1stD	0	1	20.50	4.99	6.00	17.00	21.00	24.00	36.00	▁▃▇▃▁
TotYd	0	1	348.05	85.87	105.00	285.50	354.00	405.50	592.00	▁▃▇▇▂
PassY	0	1	231.83	79.25	-6.00	175.00	229.00	282.50	490.00	▁▃▇▃▁
RushY	0	1	116.22	51.80	3.00	78.00	109.00	145.00	404.00	▃▅▂▁▁
TO	0	1	1.77	1.00	1.00	1.00	1.00	2.00	7.00	▇▁▁▁▁
X1stD.1	0	1	20.46	5.04	6.00	17.00	21.00	24.00	36.00	▁▃▇▃▁
TotYd.1	0	1	346.21	86.11	105.00	283.00	352.00	403.50	592.00	▁▃▇▇▂
PassY.1	0	1	230.55	79.54	-6.00	174.00	228.00	280.50	490.00	▁▃▇▃▁
RushY.1	0	1	115.66	51.48	3.00	78.00	108.00	144.00	404.00	▃▅▂▁▁
TO.1	0	1	1.78	1.01	1.00	1.00	1.00	2.00	7.00	▇▁▁▁▁

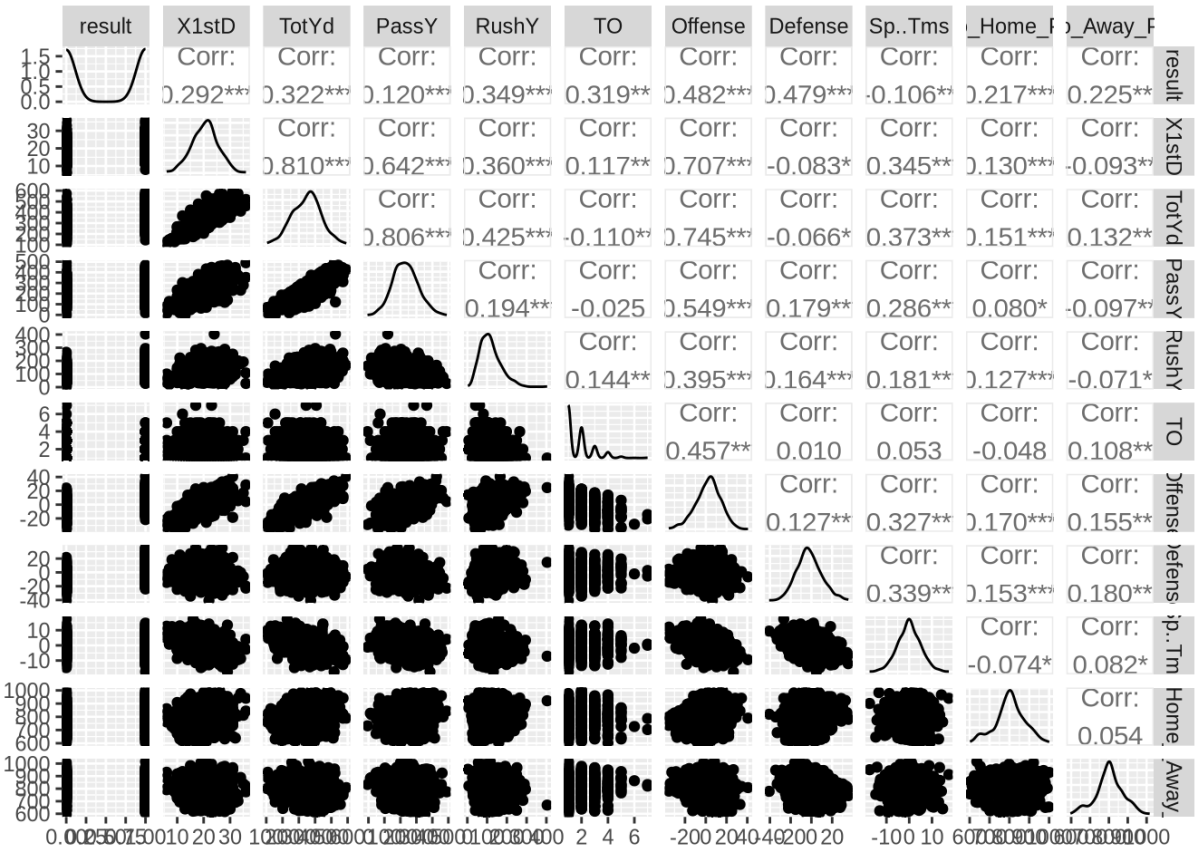
skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Offense	0	1	2.35	11.94	-35.65	-4.80	3.21	9.93	40.69	
Defense	0	1	-2.11	12.01	-40.69	-9.88	-2.83	5.31	35.65	
Sp..Tms	0	1	-0.03	5.01	-17.39	-3.41	-0.06	3.36	17.39	
Year	0	1	2020.17	0.98	2019.00	2019.00	2020.00	2021.00	2022.00	
Elo_Home_Pre	0	1	804.07	73.00	601.82	765.46	804.36	849.21	1000.73	
Elo_Away_Pre	0	1	803.32	71.57	596.03	764.43	802.17	847.54	1016.24	
Elo_Home_Post	0	1	804.13	73.94	596.03	762.69	804.51	850.52	1008.70	
Elo_Away_Post	0	1	803.26	72.64	601.82	764.08	804.36	848.11	1016.24	

Performance Stats Correlations

When trying to build a model to predict the future winners, it's important to first determine which performance stats directly correlate to each other. One aspect of gameplay alone will not dictate the outcome of the season or game, however different facets of the game may work together to either increase or decrease a team's chance of winning. We used two plots, a pair plot and correlation matrix. Both do show correlation matrices but a pair plot has the added benefit of relative distribution and correlation matrix has colors making it easier to identify hot spots.

Hide

```
nfl_combined_clean |>
  select(result, X1stD, TotYd, PassY, RushY, TO, Offense, Defense, Sp..Tms, Elo_Home_Pre, Elo
_Away_Pre) |>
  GGally::ggpairs()
```



Hide

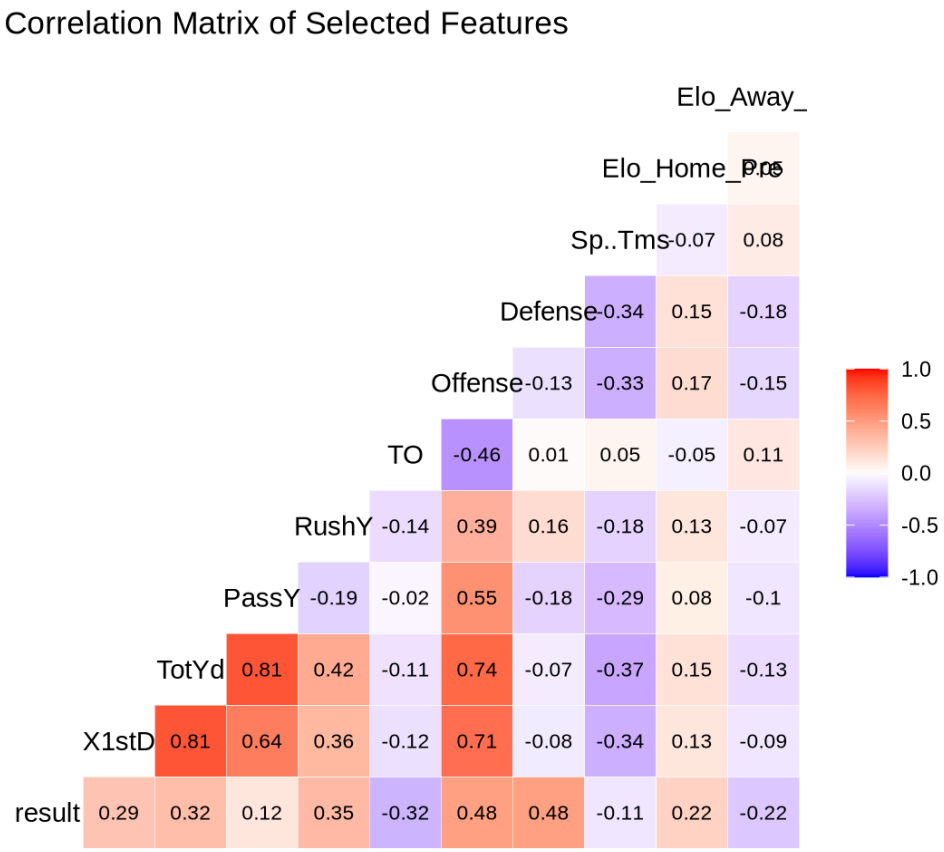
```
  ggplot2::labs(title = "Pair Plot of Performance Stats") +
  theme_minimal()
```

```
## NULL
```

Hide

```
# Select the relevant columns for the correlation matrix
correlation_data <- nfl_combined_clean |>
  select(result, X1stD, TotYd, PassY, RushY, TO, Offense, Defense, Sp..Tms, Elo_Home_Pre, Elo
_Away_Pre)

# Plot the correlation matrix
ggcorr(correlation_data,
       method = c("everything", "pearson"),
       label = TRUE,
       label_round = 2,
       label_size = 3,
       low = "blue",
       mid = "white",
       high = "red") +
  ggtitle("Correlation Matrix of Selected Features") +
  theme_minimal()
```



Using correlation plots, we wanted to explore the relationship of data with each other and our outcome result (1 is win for home team, 0 is a loss).

For a relative comparison, offensive statistics such as Number of First Downs (X1stD) has a strong positive correlation with Total Yards (0.81), Passing Yards (0.64), Rushing Yards (0.36) and Offensive Rating (0.71). By looking deeper, Offensive rating is strongly correlated with all offensive stats Total Yards (0.74), Pass Yards (0.55), Rush Yards (0.39). As Offensive rating is a derived metric from these stats, it is reasonable that they are strongly correlated.

For correlation with wins, ELO home team having a score of 0.22 and ELO away team -0.22 shows that if the home team is strong, they are likely to win and if the away team is strong, the home team is likely to lose. X1stD (0.29), TotYd (0.32), RushY (0.35), Offense (0.48) and Defense (0.48) are all positively correlated to winning showcasing that a strong offense and defense in our data set leads to winning.

These insights motivate us to build our model winning based on offensive statistics, defense rating and ELO (relative performance).

Realtive Comparison: Winning Teams and Losing Teams

Variable Deltas for Winning vs Losing

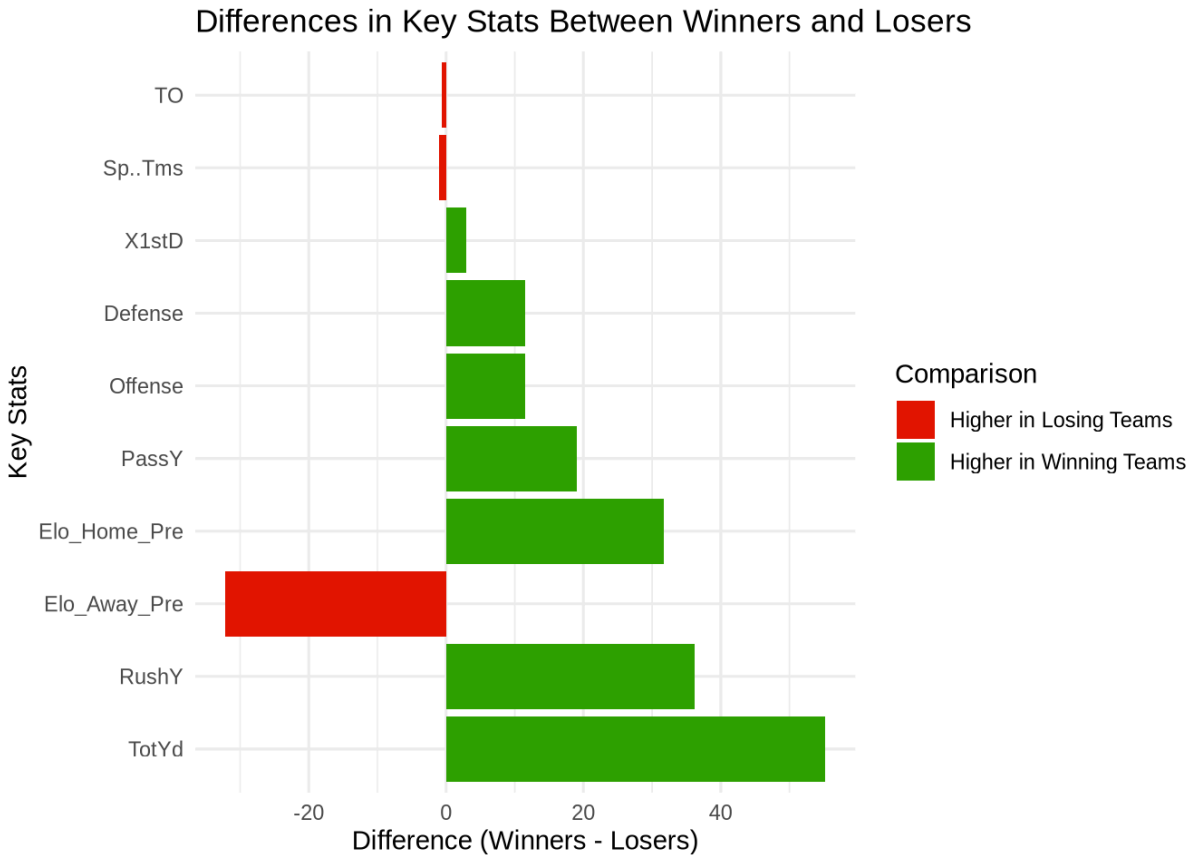
Now that we have a better understanding of which stats are directly influencing each other, we want to gain a better understanding of how these stats present themselves in the records of the winning teams for each season. Doing so will provide further context on which relationships will be more valuable for predicting winners.

Hide

```
key_stats <- nfl_combined_clean |>
  group_by(result) |>
  summarize(across(c(X1stD, TotYd, PassY, RushY, TO, Offense, Defense, Sp..Tms, Elo_Home_Pre,
Elo_Away_Pre), mean, na.rm = TRUE))

# Pivot into a comparison format
key_stats_comparison <- key_stats |>
  pivot_longer(-result, names_to = "Stat", values_to = "Average") |>
  pivot_wider(names_from = result, values_from = Average, names_prefix = "Avg_") |>
  mutate(Difference = Avg_1 - Avg_0) |>
  arrange(desc(abs(Difference)))

# Divergent bar chart
ggplot(key_stats_comparison, aes(x = reorder(Stat, -abs(Difference)), y = Difference, fill =
Difference > 0)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("TRUE" = "#31A100", "FALSE" = "#E31600"),
                    labels = c("Higher in Losing Teams", "Higher in Winning Teams")) +
  labs(title = "Differences in Key Stats Between Winners and Losers",
       x = "Key Stats",
       y = "Difference (Winners - Losers)",
       fill = "Comparison") +
  theme_minimal() +
  coord_flip()
```



This graph highlights that there are significant differences between the winning and losing teams most notably in the total yards, rushing yards, and passing yards metrics. This emphasizes the importance of offensive efficiency in the NFL as a winning factor. We also see that losing teams have more turnovers in games which is expected and reinforces the importance of keeping possession in a football game. The graph also shows that losing teams have higher ranked special teams which may seem counter intuitive at first, but this can be due to losing teams being forced into situations where their special teams have to be more involved due to failed offensive drives.

When looking at relative ELOs, ELOs tend to be higher in winning teams and ELOs tend to be lower in losing teams. This means that when facing weaker opponents, teams teams tend to win, showcasing that our metric captures the important quality of relative performance.

Variable Deltas For Superbowl Winner vs League average

Superbowl winning teams are usually the strongest teams in the league. We can analyze their stats with the league average.

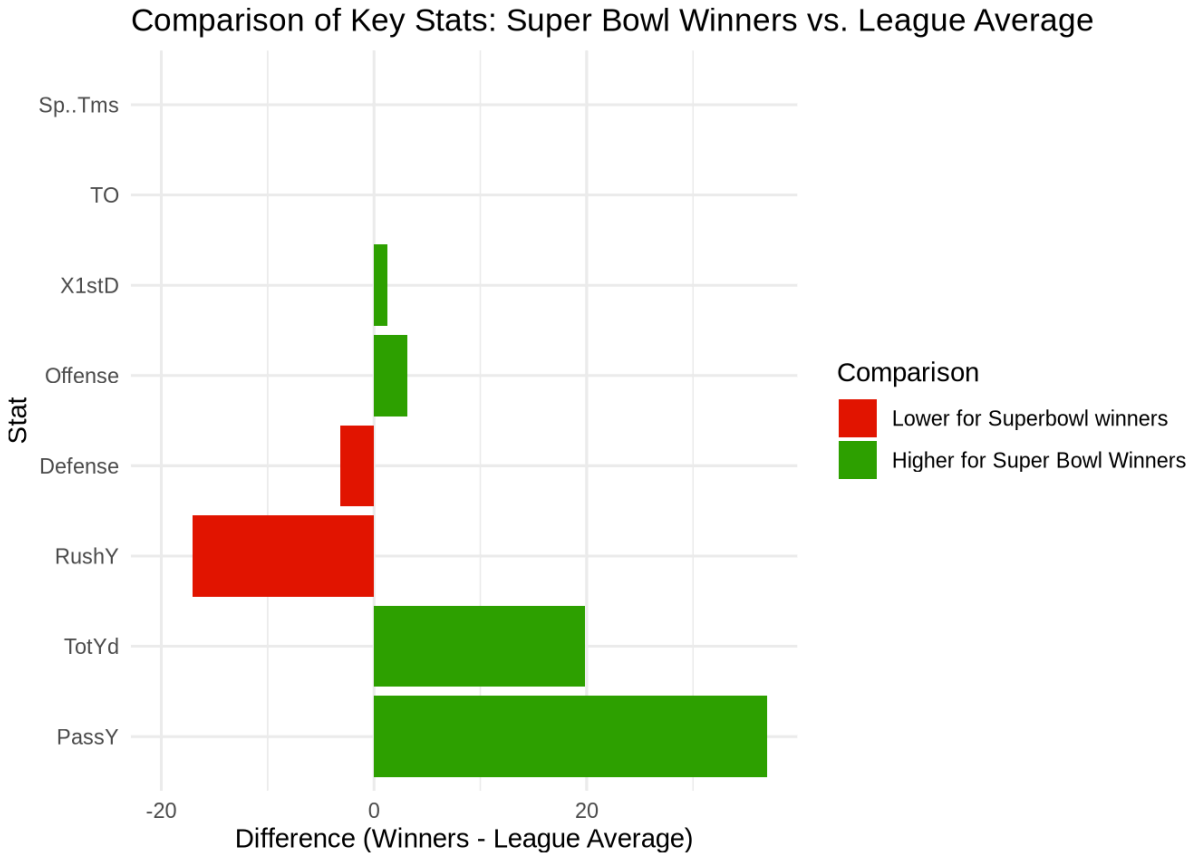
Hide

```
# Calculate summary stats for Super Bowl winners
sb_winners_stats <- nfl_combined_clean |>
  filter(
    ((Home_team == "Kansas City Chiefs" | Opp == "Kansas City Chiefs") & Year == 2019) |
    ((Home_team == "Tampa Bay Buccaneers" | Opp == "Tampa Bay Buccaneers") & Year == 2020) |
    ((Home_team == "Los Angeles Rams" | Opp == "Los Angeles Rams") & Year == 2021)
  ) |>
  summarize(across(c(X1stD, TotYd, PassY, RushY, TO, Offense, Defense, Sp..Tms),
                   mean, na.rm = TRUE))

# Calculate overall league averages
league_averages <- nfl_combined_clean |>
  summarize(across(c(X1stD, TotYd, PassY, RushY, TO, Offense, Defense, Sp..Tms),
                   mean, na.rm = TRUE))

comparison_stats <- bind_rows(
  "Super Bowl Winners" = sb_winners_stats,
  "League Average" = league_averages,
  .id = "Group"
) |>
  pivot_longer(
    cols = -Group,
    names_to = "Stat",
    values_to = "Value"
  ) |>
  pivot_wider(
    names_from = Group,
    values_from = Value
  ) |>
  mutate(Difference = `Super Bowl Winners` - `League Average`)

# bar chart to show differences
ggplot(comparison_stats, aes(x = reorder(Stat, -abs(Difference)), y = Difference, fill = Diff
erence > 0)) +
  geom_bar(stat = "identity") +
  scale_fill_manual(values = c("TRUE" = "#31A100", "FALSE" = "#E31600"),
                    labels = c("Lower for Superbowl winners", "Higher for Super Bowl Winner
s")) +
  labs(title = "Comparison of Key Stats: Super Bowl Winners vs. League Average",
       x = "Stat",
       y = "Difference (Winners - League Average)",
       fill = "Comparison") +
  theme_minimal() +
  expand_limits(y = -20) +
  coord_flip()
```



This graph compares the average stats of the Super Bowl-winning teams from 2019, 2020, and 2021 with the league average. It suggests that Super Bowl-winning teams relied much more on gaining yards through passing than rushing. Interestingly, while winning teams generally tend to have higher defensive ratings than losing teams, Super Bowl champions had defensive ratings lower than the league average.

Analysis & Results

Data Setup

As the offensive stats, defensive rating and ELO strongly signify winning, we now construct the dataset by looking at the the team's average statistics over the past 3 games and their pre-game ELO rating for only the game in question. We will train how to predict who will win and then use that to predict game winners.

Hide

```
# Sort data by Full Date
nfl_combined_clean <- nfl_combined_clean |>
  arrange(`Full_Date`)

# Define the columns to average
stats_to_average <- c("X1stD", "TotYd", "PassY", "RushY", "TO",
                      "X1stD.1", "TotYd.1", "PassY.1", "RushY.1", "TO.1",
                      "Offense", "Defense", "Sp..Tms")

# Create a new dataset with averages of past 3 home games excluding current game
model_data <- nfl_combined_clean |>
  group_by(Home_team) |>
  arrange(`Full_Date`) |>
  mutate(across(all_of(stats_to_average),
                ~ zoo::rollapplyr(lag(.x, 1), width = 3, FUN = mean, fill = NA, partial = FAL
SE))) |>
  ungroup()

# Drop rows with NA values
model_data <- model_data |>
  drop_na()


# Check the result
head(model_data)
```

```
## # A tibble: 6 × 28
##      ID Week  Day   Date  result Opp      Tm Opp.1 X1stD TotYd PassY RushY    TO
##   <int> <chr> <chr> <chr>  <dbl> <chr> <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1     1 2     Sun   15 S…      1 Cinc…    41    17  19.7  305   163. 142.   2
## 2     2 3     Sun   22 S…      1 Pitt…    24    20  21.7  393.  227. 166    1.67
## 3   446 3     Sun   22 S…      1 Seat…    33    27  19.7  359.  259. 101.   1.33
## 4   502 3     Sun   22 S…      1 Los …    27    20  21.3  405.  267  138.   1
## 5   109 4     Sun   29 S…      0 New …    10    16  24    404.  245. 158.   2.33
## 6   162 4     Sun   29 S…      1 Detr…    34    30  26.3  433.  340   92.7  1.33
## # ℹ 15 more variables: X1stD.1 <dbl>, TotYd.1 <dbl>, PassY.1 <dbl>,
## #   RushY.1 <dbl>, TO.1 <dbl>, Offense <dbl>, Defense <dbl>, Sp..Tms <dbl>,
## #   Home_team <chr>, Year <dbl>, Full_Date <date>, Elo_Home_Pre <dbl>,
## #   Elo_Away_Pre <dbl>, Elo_Home_Post <dbl>, Elo_Away_Post <dbl>
```

Logistic Regrssion Model Training

As the task is a binary prediction task, a logistic model was selected as it works directly in the probability space. the model predicts whether a team will win (1) or lose (0) based on various game statistics.

The model uses these game statistics as predictors:

1. Team Stats: First Downs (X1stD), Total Yards (TotYd), Passing Yards (PassY), Rushing Yards (RushY), Turnovers (TO)
2. Opponent Stats: First Downs (X1stD.1), Total Yards (TotYd.1), Passing Yards (PassY.1), Rushing Yards (RushY.1), Turnovers (TO.1)
3. Performance Metrics: Offense, Defense, Special Teams (Sp..Tms)
4. Relative Ranking: Elo_Home_Pre, Elo_Away_Pre

The data is split into training (70%) and testing (30%) sets. If the predicted probability is > 0.5, it predicts a win (1); if ≤ 0.5, it predicts a loss (0) model output. Accuracy (the percentage of correct predictions) is reported on the test set.

Hide

```
# Split the cleaned data
set.seed(123)
train_index <- sample(1:nrow(model_data), 0.7 * nrow(model_data))
train_data <- model_data[train_index, ]
test_data <- model_data[-train_index, ]

# Create the model
model <- glm(result ~ X1stD + TotYd + PassY + RushY + TO +
             X1stD.1 + TotYd.1 + PassY.1 + RushY.1 + TO.1 +
             Offense + Defense + Sp..Tms + Elo_Home_Pre + Elo_Away_Pre,
             data = train_data,
             family = "binomial")

# Make predictions
predictions <- predict(model, test_data, type = "response")
predicted_classes <- ifelse(predictions > 0.5, 1, 0)

# Calculate accuracy
accuracy <- mean(predicted_classes == test_data$result)
print(paste("Accuracy:", round(accuracy, 3)))
```

```
## [1] "Accuracy: 0.68"
```

Confusion Matrix

As this is a categorical task, we use a confusion matrix to see which classes are confuseable.

Hide

```
# Create the confusion matrix
confusion <- table(Predicted = predicted_classes, Actual = test_data$result)

print(confusion)
```

```
##          Actual
## Predicted  0  1
##         0 81 34
##         1 43 83
```

From this we have 83 True Positives, 81 True Negatives, 43 False Positives, and 34 False Negatives. This shows that our model does confuse classes often by only correctly prediction the winner out come 68% of the time.

Significant Variables

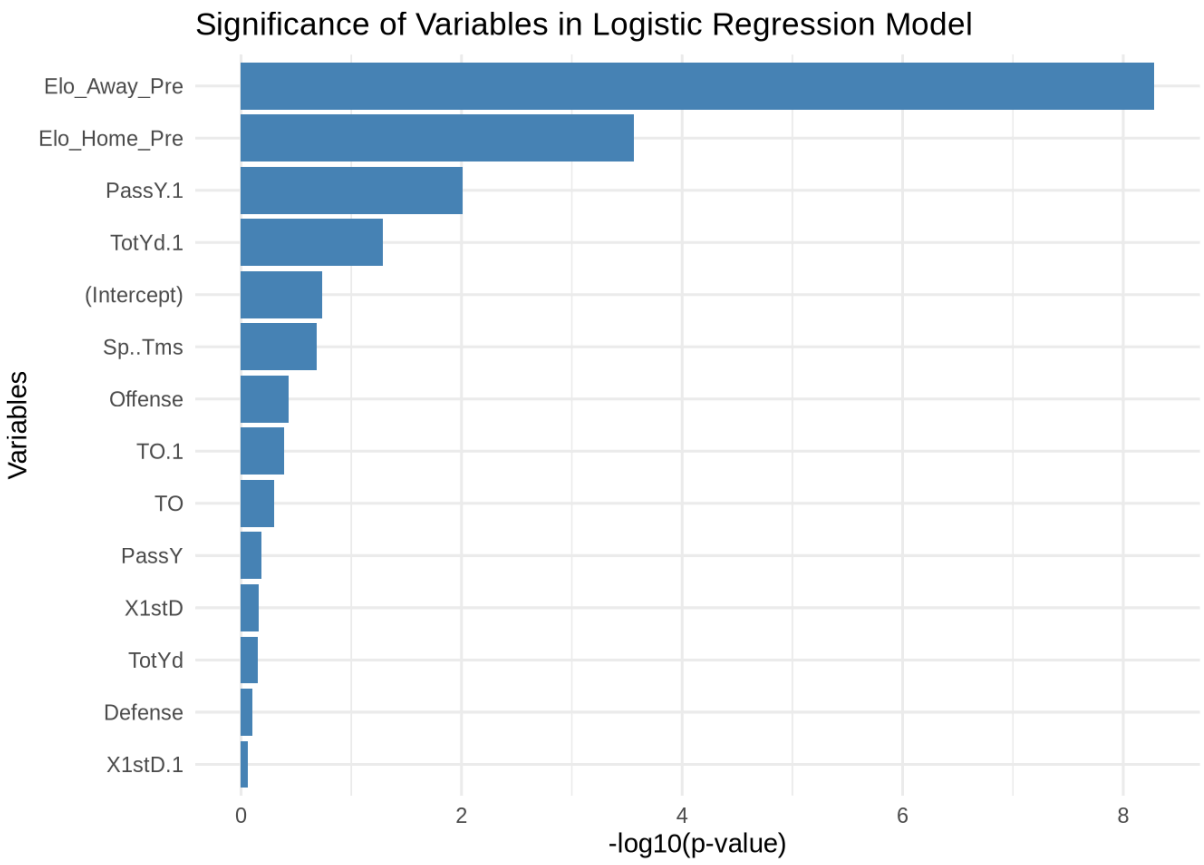
Hide

```
# Extract the summary of the model
model_summary <- summary(model)

# Create a data frame with coefficients, standard errors, and p-values
coefficients_df <- as.data.frame(model_summary$coefficients)
coefficients_df$Variable <- rownames(coefficients_df)
rownames(coefficients_df) <- NULL
colnames(coefficients_df) <- c("Estimate", "StdError", "zValue", "pValue", "Variable")

# Sort by significance (p-value)
coefficients_df <- coefficients_df[order(coefficients_df$pValue), ]

# Plot the p-values of the coefficients
library(ggplot2)
ggplot(coefficients_df, aes(x = reorder(Variable, -pValue), y = -log10(pValue))) +
  geom_bar(stat = "identity", fill = "steelblue") +
  coord_flip() +
  labs(
    title = "Significance of Variables in Logistic Regression Model",
    x = "Variables",
    y = "-log10(p-value)"
  ) +
  theme_minimal()
```



From the graph, the most important features in this model are ELO_Away_Pre which tells us facing a strong opponent directly influences winning. Having a strong team given by ELO_Home_Pre is the next strongest predictor telling us that your history of relative performance is a strong indicator of winning.

The next most impactful variables are Passing Yards and Total Yards. This paired with our correlation matrix values of 0.32 and 0.12 with our result variable, tells us that better offensive metrics lead to more winning which is in line with results from our literature research.

Discussion

Strengths

From our logistical regression model, we achieved a win prediction of 68% which is better than a random guess. This model is based on performance of the teams past 3 games' offensive and defesnive metrics telling us that we can infact predict who will win. From the model, we found that ELO_Home_Pre and ELO_Away_Pre ratings from before the game are the strongest indicators for who will win an NFL game, showing that relative performance does have a significant impact on predictions. We also found that offensive metrics like passing yards and total yards are also strong influences which match our findings from out literature search.

However, our model does get confused with 83 True Postives, 81 True Negatives, 43 False Positives, and 34 False Negatives. This is because of the size of our dataset is only 802 data points after cleaning and taking performance history for 3 games. This is a small dataset which after taking a 70% training set leads to not enough data to learn a distribution.

Limitations

Due to our dataset only consiting of 802 datapoints, it was difficult to model the distribution of win rate. Therefore by adding more data by taking the last 10-20 years of game data, we can better model win rate.

We only investigated logistical regression. One possible improvement is to have an ablation study with SVMs, KNNs, Random Forest, and XGBoost which will help us converge onto a better solution.

Conclusion

From our analysis, we achieved a win prediction accuracy of 68% using a logistic regression model, which is significantly better than a random guess (50%). This suggests that past performance metrics, when aggregated over the last 3 home games, provide meaningful insights into predicting game outcomes. Furthermore, the ELO ratings for both the home and away teams prior to the game emerged as the strongest predictors of game outcomes, reinforcing the idea that relative performance—captured by ELO—has a significant impact on determining winners in the NFL.

In addition to ELO, offensive metrics such as passing yards and total yards also played a key role, aligning with existing literature that highlights the importance of offensive performance in football. However, despite these strengths, our model faced challenges with classification, as evidenced by the confusion matrix: 83 True Positives, 81 True Negatives, 43 False Positives, and 34 False Negatives. These errors can be attributed to the small dataset size (802 datapoints after cleaning and processing), which limited the model's ability to fully capture the distribution of game outcomes.

---

1. https://towardsdatascience.com/super-bowl-prediction-model-99048f366fed (https://towardsdatascience.com/super-bowl-prediction-model-99048f366fed)↩
2. https://jaydpauley.medium.com/analysis-of-the-offensive-and-defensive-ranks-for-super-bowl-champions-2dbb354ebb60 (https://jaydpauley.medium.com/analysis-of-the-offensive-and-defensive-ranks-for-super-bowl-champions-2dbb354ebb60)↩
3. https://archive.nytimes.com/fifthdown.blogs.nytimes.com/2011/02/03/keeping-score-which-stats-can-predict-a-super-bowl-winner/ (https://archive.nytimes.com/fifthdown.blogs.nytimes.com/2011/02/03/keeping-score-which-stats-can-predict-a-super-bowl-winner/)↩
4. https://www.kaggle.com/datasets/thedevastator/nfl-team-stats-and-outcomes (https://www.kaggle.com/datasets/thedevastator/nfl-team-stats-and-outcomes)↩
5. https://stanislav-stankovic.medium.com/elo-rating-system-6196cc59941e# (https://stanislav-stankovic.medium.com/elo-rating-system-6196cc59941e#):~:text=After%20each%20match%2C%20the%20Elo,the%20match%20and%20SA%20is↩

6. https://www.kaggle.com/datasets/thedevastator/nfl-team-stats-and-outcomes (https://www.kaggle.com/datasets/thedevastator/nfl-team-stats-and-outcomes)↩