Total Goals Prediction Analysis of Premier League 23-24 Matches

Thant Thiha

2024-09-24

Introduction

This project analyzes the English Premier League (EPL) dataset for the 2023-24 season, focusing on predicting total goals scored in matches. The dataset includes various features such as Goals For (GF), Goals Against (GA), Expected Goals (xG), Shots (Sh), and other relevant statistics that can influence match outcomes.

The goal of this project is to develop predictive models that can estimate the total goals scored in matches based on these features. Key steps include - data cleaning, - exploration, - modeling, and - evaluation.

Methods/Analysis

Root Mean Square Error (RMSE) is a widely used metric for evaluating the performance of regression models. It measures the average magnitude of the errors between predicted values and actual observations. Specifically, RMSE is the square root of the average of the squares of the errors.

In this analysis, RMSE will be employed to assess the predictive capabilities of various models used to estimate total goals scored in English Premier League matches. By calculating RMSE for different models, we can identify the most effective approaches for accurate goal predictions.

Data Preparation

The dataset was imported and prepared to ensure all necessary variables were available for analysis. Key variables selected for analysis and modelling include:

- **GF**: Goal For
- **GA**: Goal Against
- **TG**: Total Goals (Goal For + Goal Against)
- **xG**: Expected Goals for the home team based on location and quality of shots
- **xGA**: Expected Goals for the away team based on location and quality of shots
- **Sh**: Shots taken
- **SoT**: Shots on Target
- **FK**: Freekicks awarded
- **PKatt**: Penalty Kick Attempts

The data was then split into a training set (90%) and a final holdout test set (10%) as below.

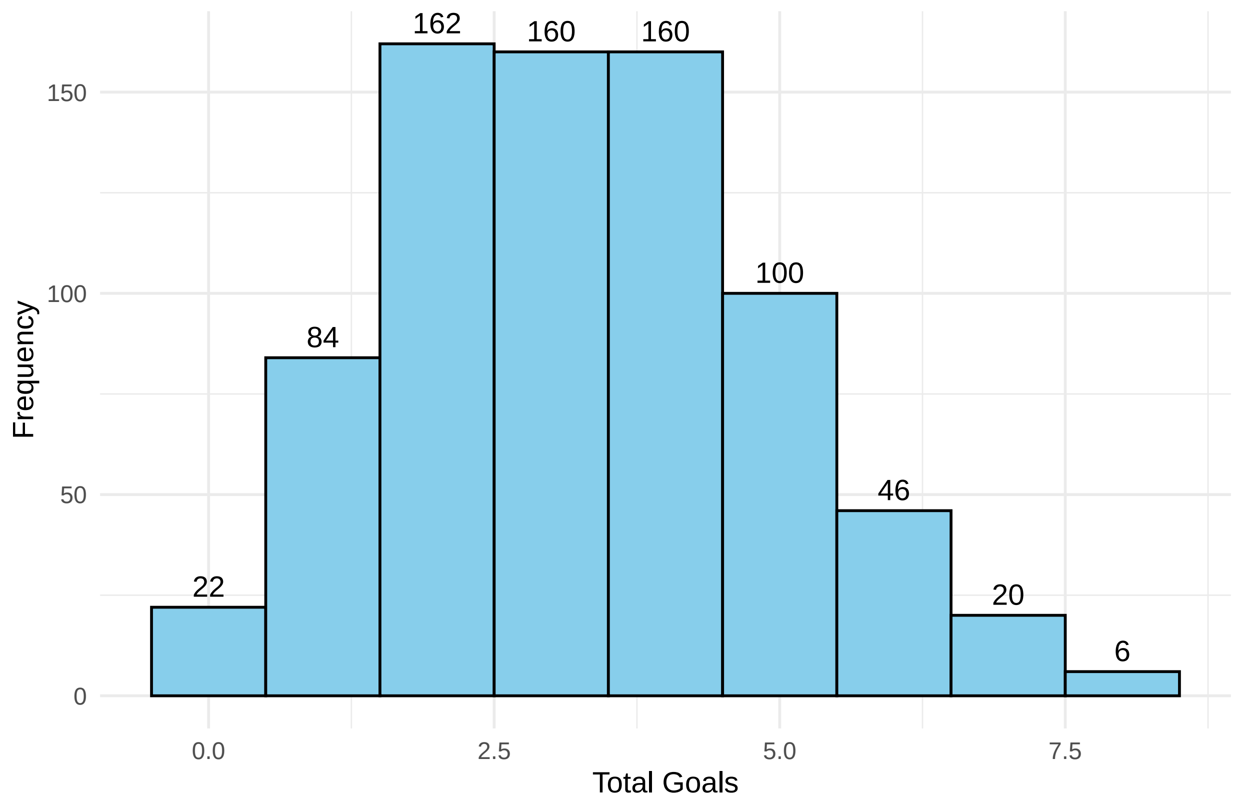
```r
# Create epl and final_holdout_test sets

# Note: this process could take a couple of minutes

if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(ggplot2)) install.packages("ggplot2", repos = "http://cran.us.r-project.org")

library(tidyverse)
library(caret)
library(ggplot2)

# English Premier League 2023-24 Matches dataset:
# https://drive.google.com/file/d/17fVQqXUfYjdLFTt6DfOXPS6gbRb5aPem/view?usp=sharing

options(timeout = 120)

file <- "premier-league-23-24-matches.csv"
url <- "https://drive.google.com/uc?export=download&id=17fVQqXUfYjdLFTt6DfOXPS6gbRb5aPem"

# Download the file if it doesn't exist locally
if(!file.exists(file))
  download.file(url, file, mode="wb")

# Read the CSV file into R
epl <- read.csv(file)

#Selecting only the necessary variables
epl <- epl %>%
  mutate(
  TG = GF + GA, # Total Goals
  TxG = xG + xGA #Total Expected Goals
  ) %>%
  select("Round", "Venue", "Result", "GF", "GA", "Opponent", "Team", "xG", "xGA", "Poss", "Sh", "SoT",


# Split the data into 90% training and 10% test sets
set.seed(1, sample.kind="Rounding")

test_index <- createDataPartition(epl$TG, times = 1, p = 0.1, list = FALSE)
epl_train <- epl[-test_index, ] # 90% Training Set
final_holdout_test <- epl[test_index, ] # 10% Test Set

knitr::opts_chunk$set(echo = TRUE, warning = FALSE, message = FALSE)
```

Data Exploration

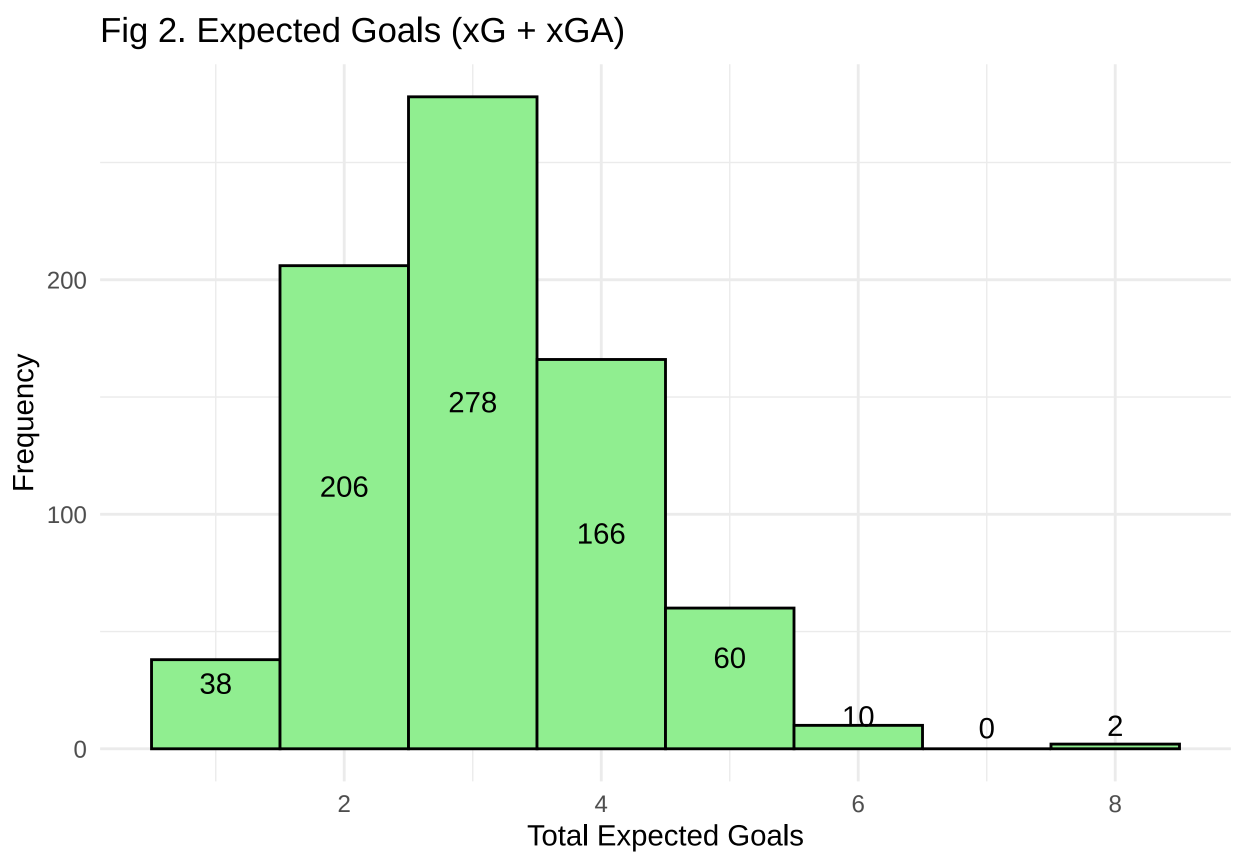
Exploratory data analysis (EDA) was conducted to understand the relationships between variables. Visualizations highlighted patterns in total goals scored and the influence of shots and expected goals.

```r
# 1. Total number of goals scored (TG)
ggplot(epl, aes(TG)) +
  geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
  geom_text(stat='count', aes(label=..count..), vjust=-0.5) +
  labs(title = "Fig 1. Total Number of Goals Scored (TG)",
            x = "Total Goals",
          y = "Frequency") +
     theme_minimal()
```

Fig 1. Total Number of Goals Scored (TG)



```r
# 2. Expected Goals (xG + xGA)
ggplot(epl, aes(x = TxG)) +
     geom_histogram(binwidth = 1, fill = "lightgreen", color = "black") +
     stat_bin(aes(label=..count..), geom="text",
              position=position_stack(vjust=0.5),
              binwidth = 1,
              vjust=-0.5) +
     labs(title = "Fig 2. Expected Goals (xG + xGA)",
          x = "Total Expected Goals",
          y = "Frequency") +
     theme_minimal()
```



```r
# 3. Number of shots (Sh) and shots on target (SoT) comparison
 epl_long <- epl %>%
     select(Sh, SoT) %>%
     gather(key = "Metric", value = "Value")

ggplot(epl_long, aes(x = Metric, y = Value)) +
     geom_boxplot(aes(fill = Metric), alpha = 0.7) +
     labs(title = "Fig.3 Comparison of Shots (Sh) and Shots on Target (SoT)",
          x = "Metric",
          y = "Value") +
     theme_minimal()
```

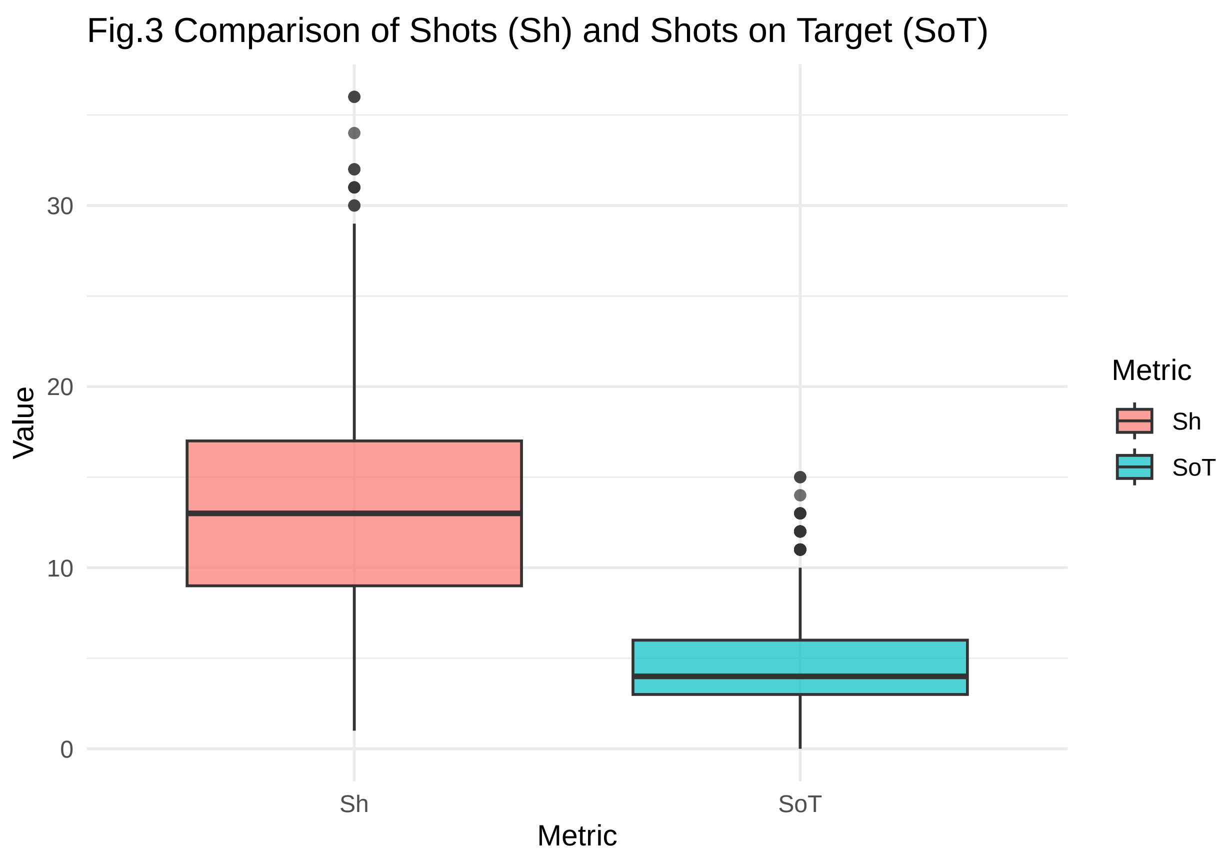


Fig 1. 2-4 goals, with a combined 482 occurrences, indicating that this is the most frequent total number of goals scored in matches. 1 goal and 5 goals appeared frequently as well with extremes like 0 goal and more than 7 are less common.

Fig 2. The 2-4 total expected goals range is the most common, with 278 occurrences. This suggests that in most matches, the combined expected goals for both teams fall within this range. The extremes (0-1 goals and more than 6 goals) are less frequent, suggesting that very low or very high expected goal totals are uncommon.

Fig 3. Shots (Sh): The median value of 13 indicates that, on average, teams take about 13 shots per game. The presence of outliers suggests that some games have significantly higher shot counts, possibly due to more aggressive offensive strategies or weaker defenses. Less than half of the shots are on target.

Modeling Approach

Several models were developed:

1. **Global Average Model**: Baseline model predicting total goals based on historical averages of matches.
2. **Shots (Sh) and Shots on Target (SoT) Model**: Predicts total goals using only shots data.
3. **Freekicks and Penalty Kick Attempt Model**: Evaluates the impact of set pieces on total goals.
4. **Expected Goals (xG and xGA) Model**: Analyzes the relationship between expected goals and total goals.
5. **Expected Goals + Shots + Shots on Target Model**: Combines expected goals and shots for prediction.

6. **Regularized Expected Goals + Shots + Shots on Target Model**: Implements regularization to prevent overfitting.

```r
# 1. Global Average Model
global_avg <- mean(epl_train$TG)

# Predict using global average for the training set
global_avg_preds <- rep(global_avg, nrow(epl_train))

# Calculate RMSE for the global average model (optional)
rmse_global_avg <- RMSE(global_avg_preds, epl_train$TG)

cat("Global Average Model RMSE (Training Set):", rmse_global_avg, "\n")
```

```
## Global Average Model RMSE (Training Set): 1.659679
```

```r
# 2. Shots (Sh) and Shots on Target (SoT) Effect Model
model_sh_sot <- train(TG ~ Sh + SoT, data = epl_train, method = "lm")

# Predicting on the training set
sh_sot_preds <- predict(model_sh_sot, epl_train)

# Calculate RMSE for the Shots and Shots on Target model (optional)
rmse_sh_sot <- RMSE(epl_train$TG, sh_sot_preds)

cat("Shots (Sh) and Shots on Target (SoT) Model RMSE (Training Set):", rmse_sh_sot, "\n")
```

```
## Shots (Sh) and Shots on Target (SoT) Model RMSE (Training Set): 1.556533
```

```r
# 3. Freekicks (FK) and Penalty Kick Attempt (PKatt) Model
model_fk_pk <- train(TG ~ FK + PKatt, data = epl_train, method = "lm")

# Predicting on the training set
fk_pk_preds <- predict(model_fk_pk, epl_train)

# Calculate RMSE for the Freekicks and Penalty Kick Attempt model (optional)
rmse_fk_pk <- RMSE(epl_train$TG, fk_pk_preds)

cat("Freekicks (FK) and Penalty Kick Attempt (PKatt) Model RMSE (Training Set):", rmse_fk_pk, "\n")
```

```
## Freekicks (FK) and Penalty Kick Attempt (PKatt) Model RMSE (Training Set): 1.635054
```

```r
# 4. Expected Goals (xG and xGA) Model
model_xg_xga <- train(TG ~ xG + xGA, data = epl_train, method = "lm")

# Predicting on the training set
xg_xga_preds <- predict(model_xg_xga, epl_train)

# Calculate RMSE for the Expected Goals model (optional)
rmse_xg_xga <- RMSE(epl_train$TG, xg_xga_preds)

cat("Expected Goals (xG and xGA) Model RMSE (Training Set):", rmse_xg_xga, "\n")
```

```
## Expected Goals (xG and xGA) Model RMSE (Training Set): 1.435788
```

```r
# 5. Expected Goals + Shots (Sh) and Shots on Target (SoT) Effect Model
model_xg_xga_sh_sot <- train(TG ~ xG + xGA + Sh + SoT, data = epl_train, method = "lm")
```

```
# Predicting on the training set
xg_xga_sh_sot_preds <- predict(model_xg_xga_sh_sot, epl_train)

# Calculate RMSE for the Expected Goals + Shots model (optional)
rmse_xg_xga_sh_sot <- RMSE(epl_train$TG, xg_xga_sh_sot_preds)

cat("Expected Goals + Shots and Shots on Target Model RMSE (Training Set):", rmse_xg_xga_sh_sot, "\n")
```

```
## Expected Goals + Shots and Shots on Target Model RMSE (Training Set): 1.372331
```

Regularization of Expected Goals + Shots + Shots on Target Model

To apply regularization and avoid overfitting in the Expected Goals + Shots (Sh) and Shots on Target (SoT) model, We are using Ridge regression (alpha = 0) for regularization, which adds a penalty on large coefficients, helping reduce overfitting and `glmnet` method with cross validation to find the optimal `lambda` value, which controls the regularization strength.

```
# 6. Regularized Expected Goals + Shots + Shots on Target Model

# Load necessary libraries
if(!require(glmnet)) install.packages("glmnet", repos = "http://cran.us.r-project.org")
library(glmnet)

# Set seed for reproducibility
set.seed(1)

# Prepare the input matrix (as required by glmnet)
x <- model.matrix(TG ~ xG + xGA + Sh + SoT, data = epl_train)[, -1]  # Remove intercept
y <- epl_train$TG  # Target variable

# Perform 10-fold cross-validation to select the best lambda
cv_model <- cv.glmnet(x, y, alpha = 0)  # alpha = 0 for Ridge regularization

# Get the optimal lambda that minimizes the cross-validation error
best_lambda <- cv_model$lambda.min
cat("Optimal lambda value:", best_lambda, "\n")
```

```
## Optimal lambda value: 0.049906
```

```
# Train the final model using the best lambda
final_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)

# Predicting on the training set
xg_xga_sh_sot_reg_preds <- predict(final_model, newx = x)

# Calculate RMSE for the regularized model on the training set
rmse_xg_xga_sh_sot_reg <- RMSE(y, xg_xga_sh_sot_reg_preds)

cat("Regularized Expected Goals + Shots Model RMSE (Training Set):", rmse_xg_xga_sh_sot_reg, "\n")
```

```
## Regularized Expected Goals + Shots Model RMSE (Training Set): 1.373197
```

Final Model Evaluation aginst holdout test set

```r
# Prepare the input matrix for the final holdout test set
x_test <- model.matrix(TG ~ xG + xGA + Sh + SoT, data = final_holdout_test)[, -1]  # Remove intercept

# Predicting on the final holdout test set using the regularized model
reg_xg_xga_sh_sot_preds <- predict(final_model, newx = x_test)

# Calculate RMSE for the regularized model on the final holdout test set
rmse_reg_xg_xga_sh_sot <- RMSE(final_holdout_test$TG, reg_xg_xga_sh_sot_preds)

cat("Regularized Expected Goals + Shots Model RMSE (Final Holdout Test):", rmse_reg_xg_xga_sh_sot, "\n"
```

```
## Regularized Expected Goals + Shots Model RMSE (Final Holdout Test): 1.38956
```

Summary of Results

The performance of each model was evaluated using Root Mean Square Error (RMSE):

- **Global Average Model RMSE**: 1.659679
- **Shots and Shots on Target Model RMSE**: 1.556533
- **Freekicks and Penalty Kick Attempt Model RMSE**: 1.635054
- **Expected Goals Model RMSE**: 1.435788
- **Expected Goals + Shots Model RMSE**: 1.372331
- **Regularized Expected Goals + Shots Model RMSE (Training Set)**: 1.373197
- **Regularized Expected Goals + Shots Model RMSE (Final Holdout Test Set)**: 1.38956

The results indicate that the regularized Expected Goals + Shots model performed best, both on the training and holdout test sets, suggesting it captures the complexities of goal-scoring effectively.

Conclusion

This analysis demonstrates the potential for predictive modeling in football match outcomes. While the regularized Expected Goals + Shots model shows promise, further improvements could be made by incorporating additional features or exploring more complex algorithms.

Limitations: The models rely heavily on the selected features which cannot be known before a match; missing or noisy data could impact predictions. Additionally, this analysis focuses solely on the EPL, and results may not generalize to other leagues.

Future Work: Future analyses could incorporate more detailed player statistics, betting odds, team ranks, match conditions, and historical performance data to enhance model accuracy.

References

- Kaggle Dataset: English Premier League Football Season 2023-24
- HarvardX Data Science Professional Certificate course: Data Analysis and Prediction Algorithms with R by Rafael A. Irizarry