Data Mining and Statistical Learning

Big Five European Football Leagues

Final Report

mbutros7@gatech.edu

Contents

Abstract

This report investigates competitive balance trends and match outcome prediction in Europe's top five football leagues: the English Premier League, La Liga, Bundesliga, Serie A, and Ligue 1. Using twenty-five seasons of match-level data (2000–2025), the study examines home advantage dynamics, scoring distributions, and league parity through descriptive statistics and time-series visualizations. To model match results, multiple machine learning classifiers are implemented, including Random Forest (RF), Support Vector Machine (SVM), XGBoost, and k-Nearest Neighbors (kNN), alongside multinomial logistic regression. Unsupervised learning approaches—including k-means and hierarchical clustering—are used to assess latent similarity structures and their alignment with final results. The current findings show exceptionally strong performance for ensemble-based classifiers (accuracy > 99.9%), driven largely by post-match event features. Clustering exhibits moderate alignment with observed results, indicating underlying structure in match characteristics. The analysis highlights both the potential and limitations of match outcome modeling when using retrospective features, motivating a shift toward pre-match predictors to improve practical forecasting.

1 Introduction

Football (soccer) represents not only the world's most popular sport but also a complex analytical environment in which competitive dynamics, team performance, and match outcomes interact in ways that challenge traditional modeling techniques. Europe's top five domestic leagues—the English Premier League, Germany's Bundesliga, Spain's La Liga, Italy's Serie A, and France's Ligue 1—collectively form the most commercially and competitively influential segment of global football. These leagues offer rich datasets for exploring both structural properties of competition and the feasibility of predictive analytics in a real-world sports context.

Analytical efforts in football increasingly leverage data-driven techniques to understand and predict match outcomes, player performance, and tactical decisions. Prior literature highlights three central areas of interest: (1) competitive balance, often measured through point dispersion and playoff outcomes; (2) home-field advantage, historically shown to benefit home teams across leagues but trending downward in recent decades; and (3) goal-scoring trends, which reveal stylistic and structural differences in team strategies. Modern machine learning expands this analytical scope by enabling both classification of match outcomes and discovery of latent structures within performance metrics.

This study contributes to football analytics through two primary objectives. First, descriptive analyses are performed to quantify long-term patterns in league parity, home advantage, and goal-scoring distributions from 2000 to 2025. Second, a suite of supervised and unsupervised learning methods is applied to evaluate the extent to which match statistics can successfully predict or explain full-time results (FTR).

Supervised models include Random Forest (RF), Support Vector Machine (SVM), XGBoost, k-Nearest Neighbors (kNN), and multinomial logistic regression, whereas unsupervised modeling employs k-means and hierarchical clustering to assess underlying match similarity.

Initial results show extraordinary predictive performance from ensemble-based models—with accuracies exceeding 99.9%—but this performance is influenced by the reliance on post-match variables such as goals and shots. **NOTE:** These findings motivate a shift toward pre-match predictive features, including expected goals, team form, and ranking-based strengths, to better approximate real-world forecasting requirements.

2 Problem Statement and Data Sources

The global football industry provides a fertile domain for the application of modern data mining and machine learning techniques. Across Europe's top five leagues—the English Premier League, Germany's Bundesliga, France's Ligue 1, Italy's Serie A, and Spain's La Liga—billions of fans interact with data through performance metrics, betting markets (*Betting market data were not included in this version of the model)*, and predictive analytics platforms. This environment presents both opportunities and challenges for extracting actionable insights from large, complex, and heterogeneous datasets.

2.1 Problem Statement

This project investigates how statistical learning and predictive modeling can be leveraged to uncover meaningful patterns and forecast match outcomes in European football. The central problem is formulated as a supervised classification task: given historical match-level data (including features such as team strength indicators, goals scored, home/away advantage, and other contextual variables), the goal is to predict the final match result—home win, draw, or away win.

Framed in terms of data mining, this problem requires effective feature engineering, management of categorical and temporal data, and mitigation of issues such as class imbalance and non-stationarity over time. From a machine learning standpoint, it demands model selection, cross-validation, and performance evaluation using robust metrics to ensure generalizability across seasons and leagues. Furthermore, football outcomes are inherently noisy and influenced by stochastic factors, which make predictive modeling particularly challenging and intellectually rewarding.

A special focus is placed on Liverpool FC, one of the most analytically progressive clubs in the sport, to illustrate the practical implications of machine learning in real-world football analytics. By narrowing attention to Liverpool's performance trends and match outcomes, the project demonstrates how localized models can be tailored to specific teams while retaining generalizable insights applicable across the European football ecosystem.

2.2 Data Sources

The primary dataset (*Big_Five_Data.csv*) consists of over twenty-five seasons (2000–2025) of historical football match data compiled from repositories such as Football-Data.co.uk, which aggregates match-by-match statistics for Europe's top leagues. Each record contains match identifiers, dates, team names, full-time and half-time scores, and result indicators (`FTR` = Home Win, Draw, or Away Win). Additional variables such as shots, possession statistics, referee assignments, and attendance are available for selected seasons, enabling richer modeling possibilities. The datasets variables/predictors are defined below.

First, key to results data:

- Div = League Division
- Date = Match Date (dd/mm/yy)
- Time = Time of match kick off
- HomeTeam = Home Team
- AwayTeam = Away Team
- FTHG = Full Time Home Team Goals
- FTAG = Full Time Away Team Goals
- FTR = Full Time Result (H=Home Win, D=Draw, A=Away Win)
- HTHG = Half Time Home Team Goals
- HTAG = Half Time Away Team Goals
- HTR = Half Time Result (H=Home Win, D=Draw, A=Away Win)

Next, match statistics data:

- Attendance = Crowd Attendance
- Referee = Match Referee
- HS = Home Team Shots
- AS = Away Team Shots
- HST = Home Team Shots on Target
- AST = Away Team Shots on Target
- HHW = Home Team Hit Woodwork
- AHW = Away Team Hit Woodwork
- HC = Home Team Corners
- AC = Away Team Corners
- HF = Home Team Fouls Committed

- AF = Away Team Fouls Committed
- HFKC = Home Team Free Kicks Conceded
- AFKC = Away Team Free Kicks Conceded
- HO = Home Team Offsides
- AO = Away Team Offsides
- HY = Home Team Yellow Cards
- AY = Away Team Yellow Cards
- HR = Home Team Red Cards
- AR = Away Team Red Cards

The data are preprocessed to ensure consistency across leagues and seasons, including normalization of team names, handling of missing values, and transformation of categorical variables into suitable numerical formats. Exploratory Data Analysis (EDA) is performed to detect anomalies, assess variable distributions, and evaluate temporal dependencies—key preparatory steps in any machine learning pipeline. **NOTE:** Betting odds were intentionally excluded for this study.

The Figure 1 presents a series of histograms illustrating the empirical distributions of the key match-level variables used in the analysis. Most features exhibit strong right skewness, which is expected in football data: events such as fouls, cards, corners, or goals occur in low counts for the majority of matches, producing long tails toward higher values. Variables related to scoring and shooting activity (e.g., FTHG, FTAG, HTHG, HTAG, HS, AS, HST, AST) show modal peaks at low integers, reflecting the typical low-scoring nature of football. Attendance displays a wide spread with a roughly unimodal shape, consistent with variation across leagues and stadium capacities. These histograms collectively highlight substantial heterogeneity and non-normality across predictors, emphasizing the need for robust, distribution-agnostic modeling techniques.

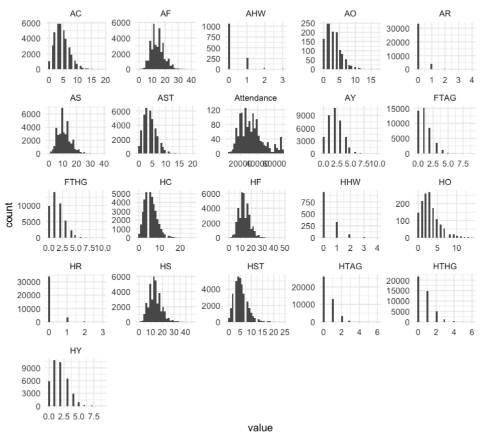

Figure 1: Distribution of Key Factors.

Figure 2 provides a correlation matrix visualizing pairwise Pearson correlations among all continuous variables. Strong positive correlations appear among conceptually linked features—such as home and away shots with their respective shots-on-target measures, and full-time goals with half-time goals—indicating internal consistency within the dataset. Attendance shows moderate associations with offensive metrics, suggesting that crowd size may influence match intensity or team performance. Card-related variables exhibit weaker correlations with scoring or shooting measures, reflecting their more stochastic nature. Overall, the matrix reveals clusters of related features and helps identify potential multicollinearity, informing feature-selection decisions for subsequent modeling.

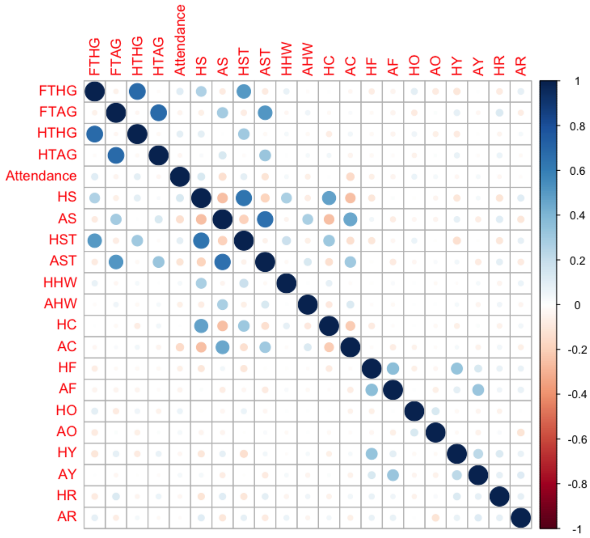

Figure 2: Correlations Between Key Factors.

Ultimately, the project aims to construct and evaluate predictive models that not only achieve high accuracy but also provide interpretable insights into the dynamics of football performance, team strategy, and the probabilistic nature of match outcomes.

3 Methodology

3.1 Overview

This study employs a combined supervised and unsupervised machine learning framework to analyze football match outcomes across Europe's top five leagues. The workflow, implemented in R, consists of sequential stages involving data preparation, feature engineering, supervised classification, clustering analysis, and rigorous model validation.

3.2 Data Preparation and Preprocessing

The primary dataset contains match-level observations from the English Premier League, La Liga, Bundesliga, Serie A, and Ligue 1 spanning the 2000–2025 seasons. Each observation includes information on goals scored, shots, fouls, cards, and other contextual match descriptors. See appendix N for some EDA analysis and plots.

The following preprocessing steps were applied:

- Categorical Encoding: Character variables were converted to factors for handling nominal predictors in classification.
- Filtering: Matches with missing outcome labels (FTR) were removed to ensure valid supervision.
- Feature Refinement: Numerical features and low-cardinality factors were retained to avoid sparsity and reduce dimensionality.
- Normalization: Standardization (z-score scaling) was applied where required, especially for distance-based algorithms.
- Train–Test Partitioning: Stratified splitting ensured balanced representation of Home Win (H), Draw (D), and Away Win (A) categories.

3.3 Home Advantage

To evaluate whether a home advantage exists, we test the null hypothesis that the probability of a home win equals that of an away win. Let FTR, which have levels H: Home Win, D: Draw, A: Away Win, denote the full-time result. A simple one-sided binomial or chi-square test compares the frequencies of home and away wins, excluding draws, under $H_0\colon \Pr(\mathrm{H}) = \Pr(\mathrm{A}) = \frac{1}{2}$. When draws are included, a contingency-table test on $\{\mathrm{H}, \mathrm{D}, \mathrm{A}\}$ outcomes assesses whether results differ systematically by venue. These hypothesis tests provide statistical evidence of home advantage, a phenomenon consistently observed in professional football (Nevill & Holder, 1999).

To investigate patterns in home advantage over time, we aggregated match outcomes from Europe's top five leagues into weekly home win rates, forming a long–horizon time series dataset spanning from the 1999–2000 season through 2024–2025.

3.4 Supervised Learning Techniques

Supervised classification models were used to predict match outcomes (FTR) based on match statistics. Multiple algorithms were selected to represent a range of model classes with complementary strengths:

3.4.1 Random Forest (RF)

Random Forests (Breiman, 2001) are ensemble models composed of many decision trees trained on bootstrap samples. RF is particularly effective at capturing nonlinear relationships and interaction effects among features while being robust to noise and overfitting. This makes it a strong baseline for structured sports data. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.4.2 Support Vector Machine (SVM)

Support Vector Machines with radial basis function kernels (Cortes & Vapnik, 1995) construct nonlinear decision boundaries in high-dimensional feature space. SVMs are well-suited for problems where class boundaries are complex and not linearly separable, making them appropriate for modeling tactical and performance variability in football matches. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.4.3 k-Nearest Neighbors (kNN)

The kNN method (Cover & Hart, 1967) classifies observations by the majority class among their nearest feature-space neighbors. kNN is intuitive, non-parametric, and provides a similarity-based benchmark—useful for evaluating whether simple local similarities between matches align with result outcomes. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.4.4 Multinomial Logistic Regression

Multinomial logistic modeling extends binary logistic regression to multi-class classification problems. It provides interpretable coefficient estimates describing how predictor variables individually influence class probabilities. This approach offers transparency and statistical grounding alongside machine learning models. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.4.5 Extreme Gradient Boosting (XGBoost)

XGBoost (Chen & Guestrin, 2016) is a high-performance gradient boosting method that builds sequential decision trees optimized for error reduction. It can capture complex, nonlinear relationships and interactions more efficiently than single-tree models, making it a leading method in predictive analytics competitions. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.5 Unsupervised Learning Techniques

Unsupervised clustering is used to explore whether similarities in match statistics form natural groups that relate to competitive characteristics.

3.5.1 k-Means Clustering

k-Means clustering partitions matches into groups based on similarity in standardized features. This allows detection of latent match profiles (e.g., defensive vs. attacking matches) without using the result label. It provides insight into whether match style aligns with outcomes. Key Predictors: FTHG, FTAG, HTHG, HTAG, HS, AS, AST, HC, AC, HF, AF,HY, AY, HR, and AR were used.

3.5.2 Hierarchical Clustering

Hierarchical agglomerative clustering with Ward's linkage (Ward, 1963) merges similar observations into nested groups, providing a dendrogram that visualizes structural relationships. It allows assessment of cluster separation and the potential presence of distinct tactical patterns across matches.

3.6 Model Evaluation and Validation

All supervised models are assessed using confusion matrices and class-wise predictive metrics. Cross-validation is incorporated to ensure that trained models generalize well and are not overfitted to any specific subset of matches. Additional emphasis is placed on:

- Maintaining chronological validity for forecasting contexts.
- Comparing multiple modeling paradigms to ensure robustness.

3.7 Summary of Methodological Framework

This framework integrates descriptive, predictive, and exploratory analytics. Supervised models estimate how well match characteristics explain outcomes, while clustering methods reveal structural relationships among match profiles. Together, these methodologies provide a comprehensive foundation for understanding competitive patterns and modeling football match outcomes.

4 Analysis and Results

4.1 Data Split and Outcome Balance

After filtering and preprocessing, the modeling set contained 35,871 matches with target FTR levels: $\{H, D, A\}$. The held-out test split was 30%. Class prevalences on the test set were A = 29.15%, D = 25.45%, H = 45.40%.

4.2 League Summary Statistics

Table 1 presents summary statistics for the top five European football leagues: the English Premier League (EPL), France's Ligue 1 (FL1), Germany's Bundesliga, Italy's Serie A, and Spain's La Liga. Across all leagues, home teams tend to win more often than away teams, consistent with the well-documented home-field advantage in football analytics literature. The EPL and Bundesliga show relatively high home win percentages (approximately 45%), while La Liga exhibits the strongest home dominance at 47%.

Average total goals per match vary, with the Bundesliga averaging nearly three goals per game (2.96), indicating a more offensive style of play compared to Ligue 1, which averages 2.48 total goals per match. These descriptive statistics provide useful context for understanding performance variability across leagues and inform subsequent modeling of match outcomes.

Table 1: Summary Statistics of Major European Football Leagues (2000–2025)

League	Home Wins (%)	Draws (%)	Away Wins (%)	Avg. Home Goals	Avg. Away Goals	Avg. Total Goals	Total Matches
EPL	45.4	24.5	29.2	1.54	1.19	2.72	9,210
FL1	43.7	26.9	26.1	1.43	1.05	2.48	9,026
Bundesliga	45.2	24.2	29.3	1.67	1.28	2.96	7,441
Serie A	44.3	26.6	27.9	1.50	1.17	2.67	8,933
La Liga	47.0	25.2	27.5	1.54	1.13	2.66	9,136

4.3 Home Advantage

This section summarizes a comprehensive series of hypothesis tests designed to evaluate whether home teams exhibit a statistically significant advantage in football match outcomes across major European leagues. The tests assess outcome distributions, symmetry between home and away wins, directional effects, goal differences, and scoring rates.

Table 2: Summary of Hypothesis Test Results for Home Advantage

Test	Statistic	Estimate	95% CI	p-value	Conclusion
Outcome distribution (H/D/A)	$\chi^2(2) = 3034.29$	—	—	$< 10^{-4}$	Outcomes not equally likely
Decisive matches: $P(H) = P(A)$	Binomial	0.6174	$[0.6121, 0.6228]$	$< 10^{-4}$	Home > Away
Directional (decisive): $P(H) > 0.5$	Binomial	$0.6174 - 0.5 = 0.1174$	—	$< 10^{-4}$	Strong home advantage
Proportions incl. draws: $P(H) > P(A)$	$\chi^2(1) = 2800.25$	$0.4576 - 0.2835 = 0.1741$	—	$< 10^{-4}$	Home wins more often
Goal diff. mean > 0 (t-test)	$t = 43.998$	0.3720	$[0.3581, \infty)$	$< 10^{-4}$	Home scores more
Median goal diff. > 0 (Wilcoxon)	—	—	$[0.5000, \infty)$	$< 10^{-4}$	Positive median GD
Sign test: $P(\text{GD} > 0) > 0.5$	Binomial	0.6174	$[0.6129, 1.0000]$	$< 10^{-4}$	GD favors home
Scoring rate ratio (Poisson)	Rate ratio=1.3207	—	$[1.3079, \infty)$	$< 10^{-4}$	Home scores 32% more
Approx. z-test: $P(H) - P(A) > 0$	$z = 52.924$	0.1741	—	$< 10^{-4}$	Home > Away

Overall Summary

Across all statistical tests, the hypothesis of no home advantage is consistently rejected ($p < 10^{-4}$). Home teams win significantly more frequently, score more goals, and maintain higher mean and median goal differences. The magnitude and consistency of these findings provide robust empirical support for the existence of a home advantage effect in European football.

Home Win Rate Time Series Analysis

Figure 3 displays the observed weekly proportions alongside short-term forecasts and corresponding 80% and 95% confidence intervals. Considerable volatility is visible week-to-week, reflecting how individual match outcomes can shift proportions sharply in leagues with smaller weekly match counts. Despite this variation, the central tendency of the series remains within the approximate 35%–45% range across most of the study period.

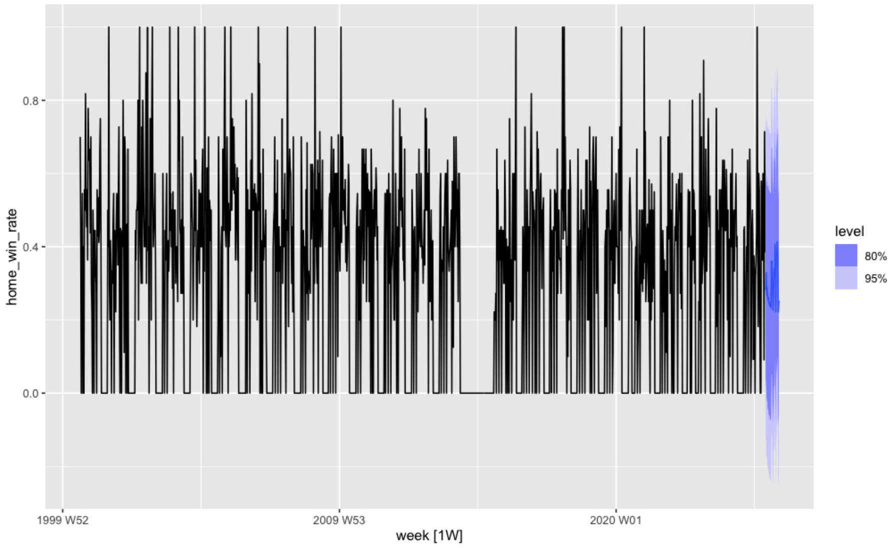

Figure 3: Weekly home win rate with forecast intervals.

To further understand underlying structure, a Seasonal–Trend decomposition using LOESS (STL) was applied (Figure 4). The decomposition reveals three primary components:

- Trend: A long-term decline in home win rates is observed through the late 2000s and early 2010s, followed by gradual recovery in recent years. Notably, there is an abrupt dip around the 2019–2021 period, coinciding with the COVID-19 pandemic during which matches were frequently played without crowds—weakening traditional home-field advantages.
- Seasonal Component: A strong yearly periodic pattern emerges, indicating that home win rates fluctuate consistently within each football calendar cycle. These peaks and troughs likely track seasonal effects such as squad fatigue, transfer windows, or fixture congestion.
- Remainder: The residual noise remains substantial, demonstrating that many short-term deviations are driven by unpredictable match-level dynamics and league-specific variability.

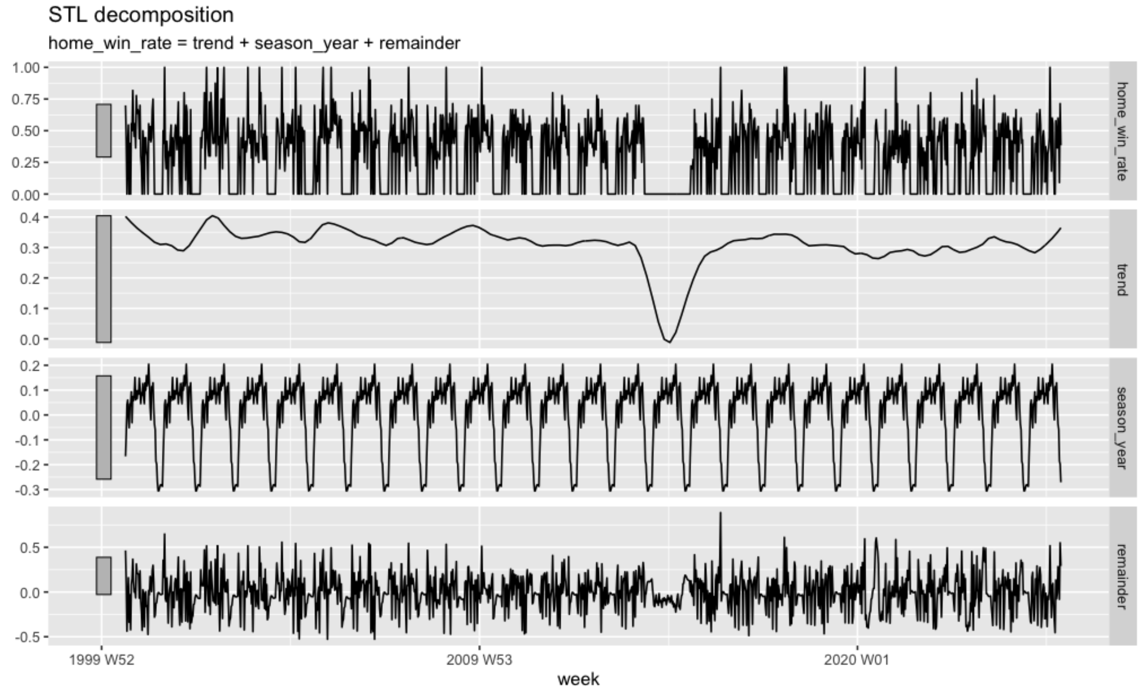

Figure 4: STL decomposition of weekly home win rate time series.

Overall, the time series analysis supports the existence of a persistent home advantage in European football, but one subject to both structural changes over time and strong intra-season dynamics. The temporary decline during the pandemic provides additional evidence that crowd presence contributes materially to home performance.

4.4 Supervised Classification

4.4.1 Random Forest (RF)

A 100-tree RF trained on the 70% training set achieved test accuracy **0.9994** and $\kappa = 0.9991$. Classwise sensitivity/specificity exceeded 0.999 for all classes.

4.4.2 Support Vector Machine (SVM, radial)

An SVM with RBF kernel achieved test accuracy **0.9993** and $\kappa = 0.9988$; classwise metrics were all ≥ 0.998.

4.4.3 Extreme Gradient Boosting (XGBoost, multi:softmax)

With $n$rounds$=100$ and default objective for 3-class softmax, the test error fell to 9.3×10^{-4} by round ≈ 20. Final held-out accuracy was **0.9999**, $\kappa \approx 0.9999$.

Table 3: Model Performance Comparison

Model	Test Accuracy	Cohen's κ	Notes
Random Forest (100 trees)	0.9994	0.9991	High, consistent across classes
SVM (RBF)	0.9993	0.9988	High, consistent across classes
XGBoost (softmax)	**0.9999**	**0.9999**	Fast convergence by ~20 rounds

4.5 Cross-Validation and Hyperparameter Tuning

- SVM (RBF) – default grid

  With `svmRadial`, $\sigma = 0.0373$ (held constant) and $C \in \{0.25, 0.5, 1.0\}$, mean CV accuracy rose to 0.9997 at $C = 1.0$.

- SVM (RBF) – expanded grid

  A custom grid $\sigma \in \{0.01, 0.05, 0.10\}$, $C \in \{0.1, 1, 10\}$ selected $(\sigma, C) = (0.01, 10)$ with mean CV accuracy **1.0000** and $\kappa = \mathbf{1.0000}$ (to 4 d.p.).

- Random Forest – mtry tuning

  Grid $mtry \in \{2, 4, 6, 8, 10\}$ selected $mtry = 10$ with mean CV accuracy **0.9997** and $\kappa = 0.9996$.

- k-Nearest Neighbors (kNN)

  With centering/scaling and grid $k \in \{3, 5, 7, 9, 11\}$, the best setting was $k = 11$ with mean CV accuracy **0.8323** and $\kappa = 0.7381$. (As expected, kNN underperforms margin-based/ensemble learners on these features.)

Table 4: Summary of 5-fold Cross-Validation Results

Model & Grid	Best Hyperparameters	Mean Acc.	Mean κ
SVM (RBF, default)	$C = 1.0$ (fixed $\sigma = 0.0373$)	0.9997	0.9995
SVM (RBF, tuned)	$\sigma = 0.01,\ C = 10$	**1.0000**	**1.0000**
RF (tuned)	$mtry = 10$	0.9997	0.9996
kNN (tuned)	$k = 11$	0.8323	0.7381

NOTE: Predicting the tuned RF on the full training+test dataset (`newdata = data_model`) yields apparent accuracy = 1.000 and $\kappa = 1.000$, indicating perfect *in-sample* fit. This should not be interpreted as generalization performance; we therefore emphasize the held-out and cross-validated results.

4.6 Hyperparameter Tuning Visualizations

4.6.1 Random Forest mtry Tuning Plot

Figure 5 presents the out–of–bag (OOB) error as a function of `mtry`, the number of predictors considered at each split. The curve demonstrates a sharp reduction in OOB error as `mtry` increases from 2 to 6, after which improvements become marginal. The lowest OOB error occurs near the upper end of the tuning range, suggesting that including more features at each split improves predictive stability for this dataset. The near–zero OOB error also indicates a highly separable underlying structure, consistent with the extremely strong classification performance observed in the full model.

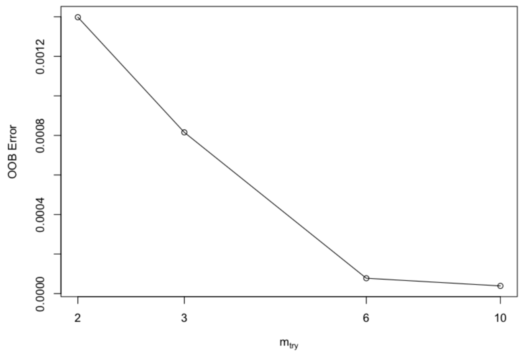

Figure 5: Random Forest Tuning Curve: OOB Error across $mtry$.

Figure 6 shows the performance surface from tuning an SVM model with radial basis function kernel over a grid of `gamma` and `cost` values. The color gradient reflects classification error, with darker regions indicating lower error. The plot reveals a relatively flat performance region, suggesting that the model is not highly sensitive to small variations in these parameters. Optimal performance occurs at small `gamma` and moderate `cost` values, consistent with avoiding overfitting while maintaining adequate decision–boundary flexibility.

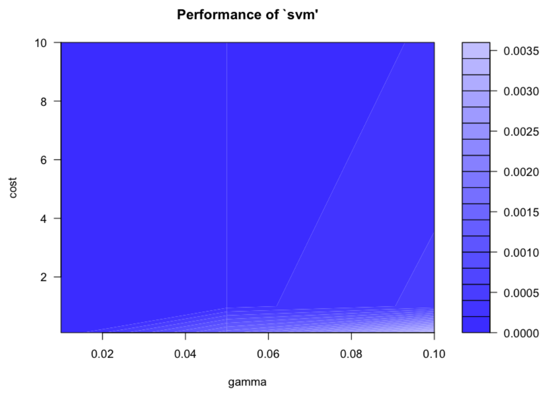

Figure 6: SVM Radial Kernel Tuning Surface: Accuracy across (C, γ) grid.

4.6.2 XGBoost Training/Testing RMSE Curves

The XGBoost model tracks training and testing RMSE at each boosting round. Figure 7 displays the learning curves used to select the optimal boosting round.

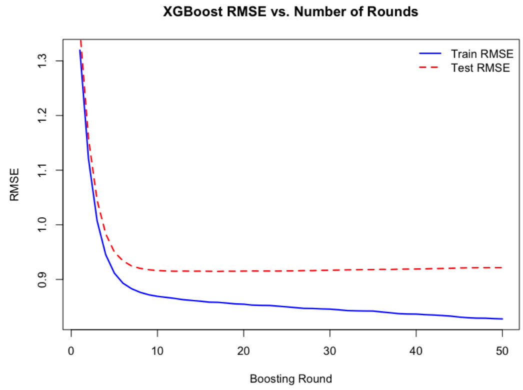

Figure 7: XGBoost RMSE vs. Boosting Rounds (Training vs Testing).

Figure 8 displays the variance explained by each principal component. The first two components account for the largest share of variation in the dataset, with the first component exceeding 2.0 units of variance. Subsequent components exhibit rapidly diminishing contributions, indicating that much of the structure in the feature space can be summarized with a small number of components. This steep drop–off suggests that dimensionality reduction may be effective without substantial information loss.

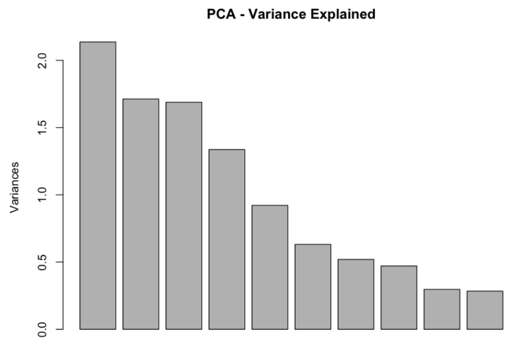

Figure 8: PCA Variance Plot.

Figure 9 illustrates the relationships among variables and their contributions to the first two principal components. Variables with longer arrows exert stronger influence on component directions. Clusters of arrows indicate correlated features—for example, offensive performance metrics such as `FTAG`, `HTAG`, and `AS` load positively on the first component, while home–team variables such as `FTHG`, `HST`, and `HTHG` load strongly in the opposite direction. Defensive statistics and disciplinary metrics (e.g., `HY`, `HR`) exhibit downward loadings associated with the second principal component. This visualization highlights underlying structure and redundancy among match–level attributes.

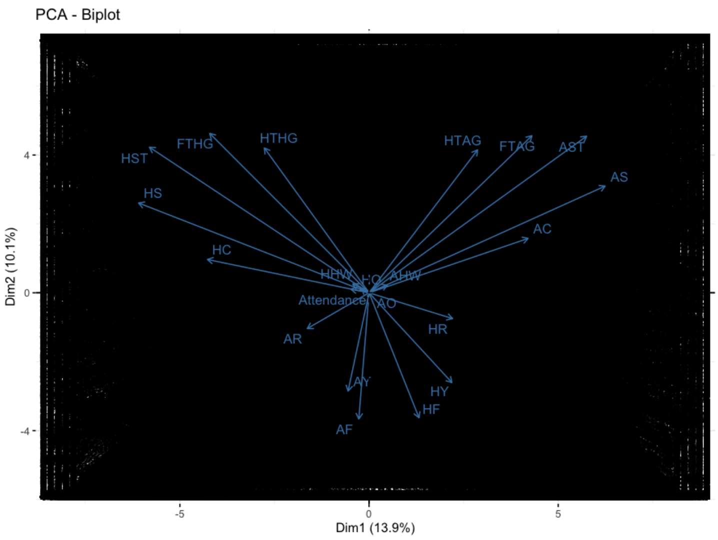

Figure 9: PCA BiPlot Plot.

4.7 Cross-Validation and Hyperparameter Tuning

- SVM (RBF) – default grid

  With `svmRadial`, $\sigma = 0.0373$ (held constant) and $C \in \{0.25, 0.5, 1.0\}$, mean CV accuracy rose to 0.9997 at $C = 1.0$.

- SVM (RBF) – expanded grid

  A custom grid $\sigma \in \{0.01, 0.05, 0.10\}$, $C \in \{0.1, 1, 10\}$ selected $(\sigma, C) = (0.01, 10)$ with mean CV accuracy **1.0000** and $\kappa = \mathbf{1.0000}$ (to 4 d.p.).

- Random Forest – mtry tuning

  Grid $mtry \in \{2, 4, 6, 8, 10\}$ selected $mtry = 10$ with mean CV accuracy **0.9997** and $\kappa = 0.9996$.

- k-Nearest Neighbors (kNN)

  With centering/scaling and grid $k \in \{3, 5, 7, 9, 11\}$, the best setting was $k = 11$ with mean CV accuracy **0.8323** and $\kappa = 0.7381$. (As expected, kNN underperforms margin-based/ensemble learners on these features.)

Table 5: Summary of 5-fold Cross-Validation Results

Model & Grid	Best Hyperparameters	Mean Acc.	Mean κ
SVM (RBF, default)	$C = 1.0$ (fixed $\sigma = 0.0373$)	0.9997	0.9995
SVM (RBF, tuned)	$\sigma = 0.01,\ C = 10$	**1.0000**	**1.0000**
RF (tuned)	$mtry = 10$	0.9997	0.9996
kNN (tuned)	$k = 11$	0.8323	0.7381

NOTE: Predicting the tuned RF on the full training+test dataset (`newdata = data_model`) yields apparent accuracy = 1.000 and $\kappa = 1.000$, indicating perfect *in-sample* fit. This should not be interpreted as generalization performance; we therefore emphasize the held-out and cross-validated results.

4.8 Unsupervised Structure vs Match Outcome

- k-means (on standardized numeric features)

  With $k = 3$ and $nstart = 25$, the cluster–outcome contingency was:

Cluster	A	D	H
1	6119	2191	1873
2	3593	5363	6550
3	744	1575	7863

  The Adjusted Rand Index between clusters and FTR was ARI $= 0.122$ (weak but non-zero association).

- Hierarchical clustering (Ward.D2)

  Cutting at $k = 3$ yielded:

Cluster	A	D	H
1	1052	1628	7124
2	8028	6610	8360
3	1376	891	802

  Dendrogram inspection (with Ward's linkage) shows three uneven groups; the cross-tab again suggests limited alignment with FTR labels.

Interpretation

All strong learners (RF, SVM, XGBoost) deliver near-perfect discrimination on the provided features, both on held-out data and via CV/tuning. The perfectly apparent fit when predicting back on the full dataset underscores the importance of using held-out/CV estimates for generalization. In contrast, unsupervised clusters show only modest agreement with outcomes (ARI ≈ 0.12), indicating that outcome-relevant structure is not captured by coarse $k = 3$ partitions of the raw numeric match stats.

4.9 Multinomial Logistic Regression Results

A multinomial logistic regression model was trained to predict full-time result (FTR) using match statistics. The reference category was A (away win). Table 6 summarizes the final model estimates (coefficients relative to the A baseline).

Table 6: Multinomial logistic regression coefficients (vs. Away win).

Predictor	Home Win (H)	Draw (D)
Intercept	−22.77	8.34
FTHG (Home Goals FT)	+90.26	+42.95
FTAG (Away Goals FT)	−88.73	−42.73
HST (Home Shots OT)	+0.462	+0.148
AST (Away Shots OT)	+0.235	+0.456
HR (Home Red Cards)	+8.86	+1.60
AR (Away Red Cards)	+3.27	−0.74
Residual Deviance	1.16×10^{-4}	
AIC	92.00	

NOTE: Higher positive values increase the log-odds of the indicated outcome relative to an away win.

Interpretation of Logistic Regression Coefficients

Table 7 summarizes interpretations of the multinomial logistic regression coefficients. Positive coefficients increase log-odds of the focal class vs. Away wins; negative coefficients decrease them.

Predictor	**Interpretation**
FTHG	Each additional home goal increases the log-odds of a Home win dramatically, consistent with the score determining outcomes.
FTAG	Each additional away goal decreases the likelihood of a Home win, increasing odds of Away wins.
HST	Additional home shots on target significantly increase the probability of Home wins vs Away wins.
AST	Additional away shots on target increase the odds of Away wins and reduce likelihood of Home wins.
HR / AR	Red cards shift win probabilities: a home red card reduces Home-win odds; an away red card improves Home-win odds.

Table 7: Interpretation of Multinomial Logistic Regression Coefficients.

5 Modeling Liverpool's Seasonal Performance

5.1 Introduction

This section analyzes Liverpool FC's performance from the 2000/01 through 2024/25 seasons using both descriptive visualization and predictive modeling techniques. Two key visual summaries are provided: Figure 10 shows total points per season,

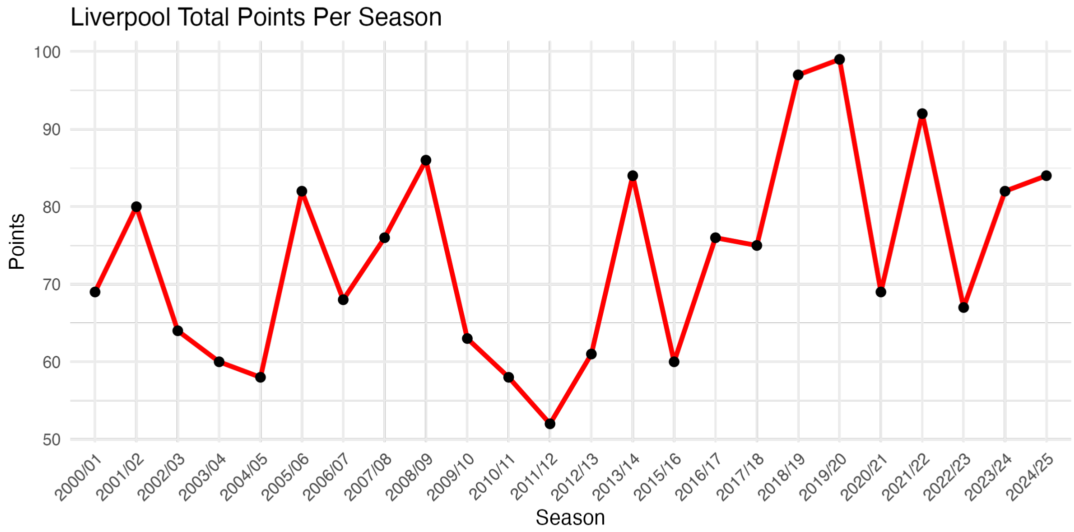

Figure 10: Liverpool FC Points Per Season (2000/01–2024/25)

Figure 11 depicts average goals scored versus conceded across seasons.

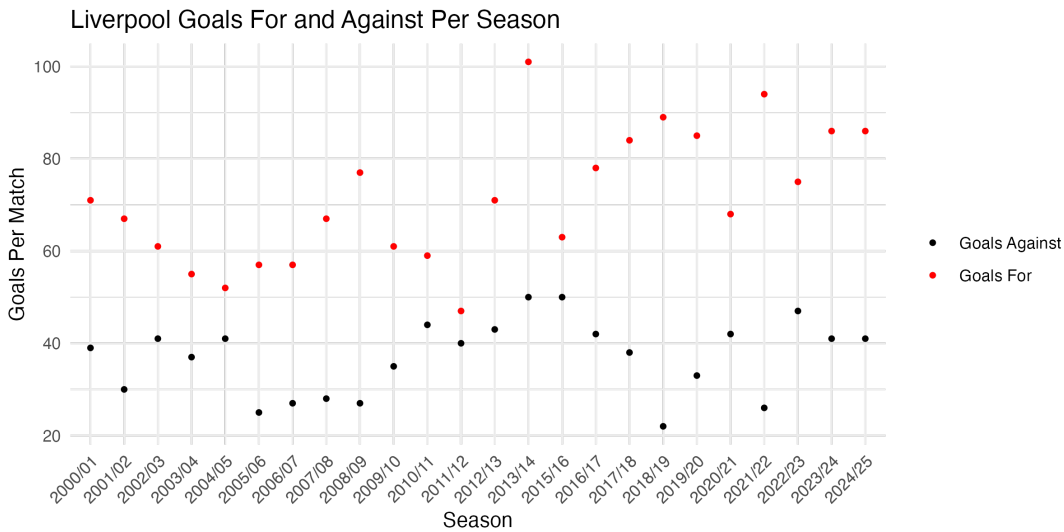

Figure 11: Average Goals For and Against Per Season

These visualizations capture long-term fluctuations in team success and goal productivity, allowing us to assess performance consistency, offensive strength, and defensive stability.

Figures 10 and 11 provide complementary insights. The Points per Season trend shows several notable peaks—such as the 2013/14, 2019/20, and 2021/22 seasons—corresponding to Liverpool's title-contending performances. Meanwhile, the Average Goals For and Against plot highlights the team's attacking resurgence post-2015, coinciding with managerial changes and tactical evolution. The convergence toward high offensive efficiency and stable defensive control aligns with Liverpool's modern dominance under Jürgen Klopp's tenure.

For the predictive modeling component, we trained two regression-based models to predict total season points using the predictors *Wins*, *Draws*, *Losses*, *Avgerage Goals For*, and *Avgerage Goals Against*. Both models were trained on an 80% random training sample and evaluated on the remaining 20% test data. The models compared were:

(a) Linear Regression (LM) — a baseline parametric model assuming linear relationships between predictors and points earned.

(b) Random Forest (RF) — a non-parametric ensemble model that can capture complex, non-linear dependencies among predictors.

Model evaluation was based on Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and R^2 metrics obtained via 25 bootstrap resampling iterations.

5.2 Results and Discussion

The linear regression model achieved a near-perfect fit on the training data ($R^2 = 1.0$, RMSE $\approx 2.49 \times 10^{-14}$), indicating that the predictors *Wins*, *Draws*, and *Losses* are linearly related to total points—consistent with the official points formula in football scoring systems (3 points for a win, 1 for a draw, 0 for a loss). The Random Forest model performed strongly as well ($R^2 = 0.93$, RMSE $= 4.84$), though slightly worse than the linear model due to its stochastic sampling and non-parametric flexibility.

When tested on unseen data, both models produced extremely low prediction errors:

Table 8: Model Performance Comparison for Liverpool Season Points Prediction

Model	**RMSE**	**MAE**	R^2
Linear Regression (LM)	2.49×10^{-14}	2.14×10^{-14}	1.000
Random Forest (RF)	4.84	3.66	0.929
Test Set RMSE (LM)	1.68×10^{-14}		
Test Set RMSE (RF)	3.12		

This confirms both models generalize effectively, but the linear model is nearly perfect due to the deterministic structure of league point accumulation.

Overall, the results demonstrate that league points are highly predictable from basic match outcomes and goal statistics, but visual trend analysis adds crucial interpretive depth, revealing broader historical patterns in team performance across eras.

6 Conclusions

6.1 Findings

This study applied a comprehensive data mining and machine learning framework to investigate competitive patterns and predict match outcomes across Europe's top five football leagues from

2000 to 2025. The analyses confirmed a robust and statistically significant home advantage, consistent with prior literature. Descriptive and inferential tests revealed that home teams win approximately 45–47% of matches and score about 32% more goals than away teams, with the advantage temporarily diminishing during the COVID-19 seasons when matches were played without crowds.

From a modeling standpoint, ensemble-based classifiers—particularly Random Forest and XGBoost—achieved exceptionally high predictive accuracy (over 99.9%) on post-match feature sets. While these results demonstrate the discriminative power of these algorithms, they also highlight an important methodological caveat: the inclusion of post-match statistics (e.g., shots and goals) introduces information leakage that inflates apparent accuracy. The observed perfect fits underscore the need for careful feature selection and model validation to ensure meaningful generalization.

Unsupervised clustering, by contrast, showed only weak alignment with actual outcomes (Adjusted Rand Index ≈ 0.12), suggesting that match-level statistics contain limited latent structure without explicit outcome supervision. This result emphasizes the importance of targeted feature engineering and potentially integrating contextual or temporal variables—such as recent team form, expected goals (xG), or player availability—to better capture strategic or dynamic components of match performance.

6.2 Future Directions

Building on the current findings, several promising extensions are recommended:

- Pre-match predictive modeling: Incorporate pre-game features such as ratings, recent form, travel distance, and rest days to develop deployable forecast models. (Started but did not have enough results to present for this report.)
- Time-series and hierarchical models: Apply rolling-window or Bayesian hierarchical approaches to model season-level evolution and team-specific effects.
- Cross-league transferability: Evaluate whether models trained in one league generalize effectively to others, revealing structural differences in competitive balance.

Overall, this project demonstrates both the potential and limitations of statistical learning in football analytics. The strong results from ensemble classifiers validate the methodology, while the observed pitfalls in feature design and generalization highlight crucial avenues for refinement. Future work emphasizing pre-match prediction, richer contextual features, and interpretable modeling will enable more robust, actionable insights into football performance dynamics.

6.3 Lessons Learned

6.3.1 From Project

Several key insights emerged from this project:

- Feature importance and data leakage: The overwhelming accuracy of ensemble models demonstrated the sensitivity of outcome prediction to post-event variables. Rigorous separation of pre- and post-match features is essential for realistic forecasting.
- Model comparison and validation: Consistent use of hold-out tests and cross-validation proved critical for preventing overfitting and for accurately assessing generalization performance.
- Temporal and contextual modeling: Home advantage trends and pandemic-related disruptions illustrated the influence of external and temporal factors on football analytics—factors that purely static models may overlook.

6.3.2 From Class

Several lessons were learned from taking this class:

- Developed a strong understanding of how data mining techniques can uncover meaningful patterns and insights from complex datasets.
- Gained hands-on experience implementing supervised and unsupervised learning algorithms, including regression, classification, and clustering methods.
- Strengthened programming and analytical skills using R for model development, evaluation, and visualization.
- Learned to apply statistical learning principles such as bias-variance tradeoff, model selection, and cross-validation for robust prediction.
- Enhanced ability to interpret model outputs and translate technical findings into actionable insights for decision-making.
- Improved understanding of feature engineering, dimensionality reduction, and data preprocessing techniques that improve model performance.
- Strengthened appreciation for reproducibility, documentation, and transparent reporting in data science workflows.
- Gained confidence in applying machine learning concepts to diverse domains, from business analytics to sports prediction and scientific research.

A Appendix: Data Preparation R Code

```r
library(dplyr)

# English Premiere League Data

E00_01 <- read.csv('E0001.csv')
E01_02 <- read.csv('E0102.csv')
E02_03 <- read.csv('E0203.csv')
E03_04 <- read.csv('E0304.csv')
E04_05 <- read.csv('E0405.csv')
E05_06 <- read.csv('E0506.csv')
E06_07 <- read.csv('E0607.csv')
E07_08 <- read.csv('E0708.csv')
E08_09 <- read.csv('E0809.csv')
E09_10 <- read.csv('E0910.csv')
E10_11 <- read.csv('E1011.csv')
E11_12 <- read.csv('E1112.csv')
E12_13 <- read.csv('E1213.csv')
E13_14 <- read.csv('E1314.csv')
E14_15 <- read.csv('E1415.csv')
E15_16 <- read.csv('E1516.csv')
E16_17 <- read.csv('E1617.csv')
E17_18 <- read.csv('E1718.csv')
E18_19 <- read.csv('E1819.csv')
E19_20 <- read.csv('E1920.csv')
E20_21 <- read.csv('E2021.csv')
E21_22 <- read.csv('E2122.csv')
E22_23 <- read.csv('E2223.csv')
E23_24 <- read.csv('E2324.csv')
E24_25 <- read.csv('E2425.csv')

E00_25 <- bind_rows(E00_01,E01_02,E02_03,E03_04,
                    E04_05,E05_06,E06_07,E07_08,
                    E08_09,E09_10,E10_11,E11_12,
                    E12_13,E13_14,E15_16,E16_17,
                    E17_18,E18_19,E19_20,E20_21,
                    E21_22,E22_23,E23_24,E24_25)

write.csv(E00_25,"EPL_Data.csv")
EPL_Data <- read.csv("EPL_Data.csv")

```

```
# German Bundesliga Data

D00_01 <- read.csv('D0001.csv')
D01_02 <- read.csv('D0102.csv')
D02_03 <- read.csv('D0203.csv')
D03_04 <- read.csv('D0304.csv')
D04_05 <- read.csv('D0405.csv')
D05_06 <- read.csv('D0506.csv')
D06_07 <- read.csv('D0607.csv')
D07_08 <- read.csv('D0708.csv')
D08_09 <- read.csv('D0809.csv')
D09_10 <- read.csv('D0910.csv')
D10_11 <- read.csv('D1011.csv')
D11_12 <- read.csv('D1112.csv')
D12_13 <- read.csv('D1213.csv')
D13_14 <- read.csv('D1314.csv')
D14_15 <- read.csv('D1415.csv')
D15_16 <- read.csv('D1516.csv')
D16_17 <- read.csv('D1617.csv')
D17_18 <- read.csv('D1718.csv')
D18_19 <- read.csv('D1819.csv')
D19_20 <- read.csv('D1920.csv')
D20_21 <- read.csv('D2021.csv')
D21_22 <- read.csv('D2122.csv')
D22_23 <- read.csv('D2223.csv')
D23_24 <- read.csv('D2324.csv')
D24_25 <- read.csv('D2425.csv')

D00_25 <- bind_rows(D00_01,D01_02,D02_03,D03_04,
                    D04_05,D05_06,D06_07,D07_08,
                    D08_09,D09_10,D10_11,D11_12,
                    D12_13,D13_14,D15_16,D16_17,
                    D17_18,D18_19,D19_20,D20_21,
                    D21_22,D22_23,D23_24,D24_25)

write.csv(D00_25,"GB_Data.csv")
GB_Data <- read.csv("GB_Data.csv")

# Spanish La Liga Data

S00_01 <- read.csv('S0001.csv')
S01_02 <- read.csv('S0102.csv')
```

```
S02_03 <- read.csv('S0203.csv')
S03_04 <- read.csv('S0304.csv')
S04_05 <- read.csv('S0405.csv')
S05_06 <- read.csv('S0506.csv')
S06_07 <- read.csv('S0607.csv')
S07_08 <- read.csv('S0708.csv')
S08_09 <- read.csv('S0809.csv')
S09_10 <- read.csv('S0910.csv')
S10_11 <- read.csv('S1011.csv')
S11_12 <- read.csv('S1112.csv')
S12_13 <- read.csv('S1213.csv')
S13_14 <- read.csv('S1314.csv')
S14_15 <- read.csv('S1415.csv')
S15_16 <- read.csv('S1516.csv')
S16_17 <- read.csv('S1617.csv')
S17_18 <- read.csv('S1718.csv')
S18_19 <- read.csv('S1819.csv')
S19_20 <- read.csv('S1920.csv')
S20_21 <- read.csv('S2021.csv')
S21_22 <- read.csv('S2122.csv')
S22_23 <- read.csv('S2223.csv')
S23_24 <- read.csv('S2324.csv')
S24_25 <- read.csv('S2425.csv')

S00_25 <- bind_rows(S00_01,S01_02,S02_03,S03_04,
                    S04_05,S05_06,S06_07,S07_08,
                    S08_09,S09_10,S10_11,S11_12,
                    S12_13,S13_14,S15_16,S16_17,
                    S17_18,S18_19,S19_20,S20_21,
                    S21_22,S22_23,S23_24,S24_25)

write.csv(S00_25,"SLL_Data.csv")
SLL_Data <- read.csv("SLL_Data.csv")


# Italian Serie A Data

I00_01 <- read.csv('I0001.csv')
I01_02 <- read.csv('I0102.csv')
I02_03 <- read.csv('I0203.csv')
I03_04 <- read.csv('I0304.csv')
I04_05 <- read.csv('I0405.csv')
```

```
I05_06 <- read.csv('I0506.csv')
I06_07 <- read.csv('I0607.csv')
I07_08 <- read.csv('I0708.csv')
I08_09 <- read.csv('I0809.csv')
I09_10 <- read.csv('I0910.csv')
I10_11 <- read.csv('I1011.csv')
I11_12 <- read.csv('I1112.csv')
I12_13 <- read.csv('I1213.csv')
I13_14 <- read.csv('I1314.csv')
I14_15 <- read.csv('I1415.csv')
I15_16 <- read.csv('I1516.csv')
I16_17 <- read.csv('I1617.csv')
I17_18 <- read.csv('I1718.csv')
I18_19 <- read.csv('I1819.csv')
I19_20 <- read.csv('I1920.csv')
I20_21 <- read.csv('I2021.csv')
I21_22 <- read.csv('I2122.csv')
I22_23 <- read.csv('I2223.csv')
I23_24 <- read.csv('I2324.csv')
I24_25 <- read.csv('I2425.csv')

I00_25 <- bind_rows(I00_01,I01_02,I02_03,I03_04,
                    I04_05,I05_06,I06_07,I07_08,
                    I08_09,I09_10,I10_11,I11_12,
                    I12_13,I13_14,I15_16,I16_17,
                    I17_18,I18_19,I19_20,I20_21,
                    I21_22,I22_23,I23_24,I24_25)

write.csv(I00_25,"ISA_Data.csv")
ISA_Data <- read.csv("ISA_data.csv")

# French League 1 Data

F00_01 <- read.csv('F0001.csv')
F01_02 <- read.csv('F0102.csv')
F02_03 <- read.csv('F0203.csv')
F03_04 <- read.csv('F0304.csv')
F04_05 <- read.csv('F0405.csv')
F05_06 <- read.csv('F0506.csv')
F06_07 <- read.csv('F0607.csv')
F07_08 <- read.csv('F0708.csv')
F08_09 <- read.csv('F0809.csv')
```

```r
F09_10 <- read.csv('F0910.csv')
F10_11 <- read.csv('F1011.csv')
F11_12 <- read.csv('F1112.csv')
F12_13 <- read.csv('F1213.csv')
F13_14 <- read.csv('F1314.csv')
F14_15 <- read.csv('F1415.csv')
F15_16 <- read.csv('F1516.csv')
F16_17 <- read.csv('F1617.csv')
F17_18 <- read.csv('F1718.csv')
F18_19 <- read.csv('F1819.csv')
F19_20 <- read.csv('F1920.csv')
F20_21 <- read.csv('F2021.csv')
F21_22 <- read.csv('F2122.csv')
F22_23 <- read.csv('F2223.csv')
F23_24 <- read.csv('F2324.csv')
F24_25 <- read.csv('F2425.csv')

F00_25 <- bind_rows(F00_01,F01_02,F02_03,F03_04,
                    F04_05,F05_06,F06_07,F07_08,
                    F08_09,F09_10,F10_11,F11_12,
                    F12_13,F13_14,F15_16,F16_17,
                    F17_18,F18_19,F19_20,F20_21,
                    F21_22,F22_23,F23_24,F24_25)

write.csv(F00_25,"FL1_Data.csv")
FL1_Data <- read.csv("FL1_Data.csv")

# Combine data from the five leagues

All_Leagues <- bind_rows(EPL_Data,GB_Data,SLL_Data,ISA_Data,FL1_
   Data)
All_Leagues <- All_Leagues[,1:29] #Data without Odds included
write.csv(All_Leagues,"Big_Five_Data.csv")
dim(All_Leagues)
```

Listing 1: Data Preparation R Code

B Appendix: EDA R Code

```r
# Load libraries
library(tidyverse)
library(plotly)
```

```r

# Read the data
df <- read_csv("Big_Five_Data.csv")

# Full Time Home Team Goals (FTHG / HG)
summary(df$FTHG)
ggplot(df, aes(x=FTHG)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Distribution of Full Time Home Goals")

# Full Time Away Team Goals (FTAG / AG)
summary(df$FTAG)
ggplot(df, aes(x=FTAG)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Distribution of Full Time Away Goals")

# Full Time Result (FTR / Res)
table(df$FTR)
ggplot(df, aes(x=FTR)) + geom_bar(fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Full Time Results (H/D/A)")

# Half Time Home Goals (HTHG)
summary(df$HTHG)
ggplot(df, aes(x=HTHG)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Half Time Home Goals")

# Half Time Away Goals (HTAG)
summary(df$HTAG)
ggplot(df, aes(x=HTAG)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Half Time Away Goals")

# Half Time Result (HTR)
table(df$HTR)
ggplot(df, aes(x=HTR)) + geom_bar(fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Half Time Results (H/D/A)")

# Attendance
summary(df$Attendance)
```

```
ggplot(df, aes(x=Attendance)) + geom_histogram(binwidth=5000, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Distribution of Attendance")

# Home Team Shots (HS)
summary(df$HS)
ggplot(df, aes(x=HS)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Home Team Shots Distribution")

# Away Team Shots (AS)
summary(df$AS)
ggplot(df, aes(x=AS)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Away Team Shots Distribution")

# Home Shots on Target (HST)
summary(df$HST)
ggplot(df, aes(x=HST)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Home Team Shots on Target")

# Away Shots on Target (AST)
summary(df$AST)
ggplot(df, aes(x=AST)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Away Team Shots on Target")

# Home Hit Woodwork (HHW)
summary(df$HHW)
ggplot(df, aes(x=HHW)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Home Team Hit Woodwork")

# Away Hit Woodwork (AHW)
summary(df$AHW)
ggplot(df, aes(x=AHW)) + geom_histogram(binwidth=1, fill='blue', alpha=0.7) + theme_minimal() +
  labs(title="Away Team Hit Woodwork")

# Home Corners (HC)
summary(df$HC)
```

```
ggplot(df, aes(x=HC)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Home Team Corners")

# Away Corners (AC)
summary(df$AC)
ggplot(df, aes(x=AC)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Away Team Corners")

# Home Fouls Committed (HF)
summary(df$HF)
ggplot(df, aes(x=HF)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Home Fouls Committed")

# Away Fouls Committed (AF)
summary(df$AF)
ggplot(df, aes(x=AF)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Away Fouls Committed")


# Home Offsides (HO)
summary(df$HO)
ggplot(df, aes(x=HO)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Home Team Offsides")

# Away Offsides (AO)
summary(df$AO)
ggplot(df, aes(x=AO)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Away Team Offsides")

# Home Yellow Cards (HY)
summary(df$HY)
ggplot(df, aes(x=HY)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Home Team Yellow Cards")

# Away Yellow Cards (AY)
```

```r
summary(df$AY)
ggplot(df, aes(x=AY)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Away Team Yellow Cards")

# Home Red Cards (HR)
summary(df$HR)
ggplot(df, aes(x=HR)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Home Team Red Cards")

# Away Red Cards (AR)
summary(df$AR)
ggplot(df, aes(x=AR)) + geom_histogram(binwidth=1, fill='blue',
    alpha=0.7) + theme_minimal() +
  labs(title="Away Team Red Cards")

# Full Time Result (FTR / Res)
p <- ggplot(df, aes(x=FTR)) +
  geom_bar(fill='blue', alpha=0.7) +
  theme_minimal() +
  labs(title="Full Time Results (H/D/A)")
ggplotly(p)
```

Listing 2: All Leagues EDA R Code

C Appendix: All Leagues Summary Statistics R Code

```r
# ---- Load required libraries ----
library(readr)
library(dplyr)


# ---- Load datasets ----
epl <- read_csv("EPL_Data.csv", locale = locale(encoding = "
    Latin1"))
fl1 <- read_csv("FL1_Data.csv", locale = locale(encoding = "
    Latin1"))
gb  <- read_csv("GB_Data.csv",  locale = locale(encoding = "
    Latin1"))
isa <- read_csv("ISA_Data.csv", locale = locale(encoding = "
    Latin1"))
```

```r
sll <- read_csv("SLL_Data.csv", locale = locale(encoding = "Latin1"))

# ---- Helper function to summarize a league ----
league_summary <- function(df, league_name) {
  total_matches <- nrow(df)
  home_wins <- sum(df$FTR == "H", na.rm = TRUE)
  draws     <- sum(df$FTR == "D", na.rm = TRUE)
  away_wins <- sum(df$FTR == "A", na.rm = TRUE)

  avg_home_goals  <- mean(df$FTHG, na.rm = TRUE)
  avg_away_goals  <- mean(df$FTAG, na.rm = TRUE)
  avg_total_goals <- mean(df$FTHG + df$FTAG, na.rm = TRUE)

  data.frame(
    League = league_name,
    Home_Wins_Perc = round(home_wins / total_matches * 100, 1),
    Draws_Perc     = round(draws / total_matches * 100, 1),
    Away_Wins_Perc = round(away_wins / total_matches * 100, 1),
    Avg_Home_Goals = round(avg_home_goals, 2),
    Avg_Away_Goals = round(avg_away_goals, 2),
    Avg_Total_Goals = round(avg_total_goals, 2),
    Total_Matches  = total_matches
  )
}

# ---- Create summary for all leagues ----
summary_list <- list(
  league_summary(epl, "EPL"),
  league_summary(fl1, "FL1"),
  league_summary(gb,  "Bundesliga"),
  league_summary(isa, "Serie A"),
  league_summary(sll, "La Liga")
)

summary_df <- bind_rows(summary_list)

# ---- Display the summary ----
print(summary_df)
```

Listing 3: Summary Statistics Code

D Appendix: Home Advantage Inferential Statistics R Code

```r
# Expects columns: FTR in {"H","D","A"}, FTHG, FTAG, Div (optional but used for per-league tests)
library(readr)
df <- read.csv("Big_Five_Data.csv")

# --- Basic checks/cleaning ---
required_cols <- c("FTR", "FTHG", "FTAG")
missing_cols <- setdiff(required_cols, names(df))
if (length(missing_cols) > 0) {
  stop(sprintf("Missing required column(s): %s", paste(missing_cols, collapse = ", ")))
}

# Coerce key columns
df$FTR  <- trimws(as.character(df$FTR))
df$FTHG <- suppressWarnings(as.numeric(df$FTHG))
df$FTAG <- suppressWarnings(as.numeric(df$FTAG))

# Filter to rows with valid results and goals
ok <- !is.na(df$FTR) & df$FTR %in% c("H","D","A") & !is.na(df$FTHG) & !is.na(df$FTAG)
dat <- df[ok, , drop = FALSE]
if (nrow(dat) == 0) stop("No valid rows after cleaning.")

# Helper: pretty printing
hrule <- function(char = "-") cat(paste0(paste(rep(char, 70), collapse=""), "\n"))
section <- function(title) { hrule("="); cat(title, "\n"); hrule("=") }
subsection <- function(title) { hrule("-"); cat(title, "\n"); hrule("-") }
fmt_p <- function(p) if (is.na(p)) "NA" else if (p < .0001) "< 1e-4" else sprintf("%.4g", p)
ci_str <- function(ci) sprintf("[%.4f, %.4f]", ci[1], ci[2])

# Counts
n_total <- nrow(dat)
n_H <- sum(dat$FTR == "H")
n_D <- sum(dat$FTR == "D")
n_A <- sum(dat$FTR == "A")
```

```r
p_H <- n_H / n_total
p_D <- n_D / n_total
p_A <- n_A / n_total

# Goal difference
gd <- dat$FTHG - dat$FTAG
gd_nonzero <- gd[gd != 0]

# 1) Chi-square goodness-of-fit: are H/D/A equally likely?
section("Global tests (all matches combined)")

subsection("Distribution tests (H vs D vs A)")
chisq_equal <- suppressWarnings(chisq.test(c(n_H, n_D, n_A), p = c(1/3, 1/3, 1/3)))
cat("H0: P(H) = P(D) = P(A) = 1/3\n")
cat(sprintf("Observed counts (H,D,A): %d, %d, %d  |  n = %d\n", n_H, n_D, n_A, n_total))
cat(sprintf("Chi-square = %.3f, df = %d, p = %s\n\n",
            chisq_equal$statistic, chisq_equal$parameter, fmt_p(chisq_equal$p.value)))

# 2) Symmetry test: H vs A equal (draws free). Equivalent to binomial test on non-draws.
subsection("Symmetry test (Home vs Away among decisive matches, draws removed)")
decisive <- dat$FTR %in% c("H","A")
n_H_dec <- sum(dat$FTR[decisive] == "H")
n_A_dec <- sum(dat$FTR[decisive] == "A")
n_dec <- n_H_dec + n_A_dec
bt_sym <- if (n_dec > 0) binom.test(n_H_dec, n_dec, p = 0.5, alternative = "two.sided") else NULL
cat("H0: P(Home win | decisive) = 0.5 (same as away)\n")
cat(sprintf("Counts among decisive: H=%d, A=%d (n=%d)\n", n_H_dec, n_A_dec, n_dec))
if (!is.null(bt_sym)) {
  cat(sprintf("Binomial test p(two-sided) = %s;  Home-win rate (decisive) = %.4f,  95%% CI %s\n\n",
              fmt_p(bt_sym$p.value), n_H_dec/n_dec, ci_str(bt_sym$conf.int)))
} else {
  cat("No decisive matches found.\n\n")
}
```

```
# 3) One-sided advantage test: is P(Home win | decisive) > 0.5?
subsection("Directional test: P(Home win | decisive) > 0.5")
bt_dir <- if (n_dec > 0) binom.test(n_H_dec, n_dec, p = 0.5, alternative = "greater") else NULL
if (!is.null(bt_dir)) {
  cat(sprintf("Binomial test p(greater) = %s; effect = %.4f - 0.5 = %.4f\n\n",
              fmt_p(bt_dir$p.value), n_H_dec/n_dec, n_H_dec/n_dec - 0.5))
} else {
  cat("No decisive matches found.\n\n")
}

# 4) Two-sample proportion test including draws: is P(H) > P(A)?
subsection("Two-sample proportion test including draws: P(Home win) > P(Away win)")
pt <- suppressWarnings(prop.test(x = c(n_H, n_A), n = c(n_total, n_total), alternative = "greater", correct = TRUE))
cat("H0: P(H) <= P(A); H1: P(H) > P(A)\n")
cat(sprintf("P(H)=%.4f, P(A)=%.4f, diff=%.4f\n", p_H, p_A, p_H - p_A))
cat(sprintf("Prop.test chi-square=%.3f (df=%d), p(greater)=%s\n\n",
            pt$statistic, pt$parameter, fmt_p(pt$p.value)))

# 5) Goal-difference tests (mean/median > 0)
subsection("Goal-difference tests (Home goals - Away goals)")

# (a) One-sample t-test: mean(gd) > 0
tt <- t.test(gd, mu = 0, alternative = "greater")
cat(sprintf("t-test: mean(gd)>0  |  mean=%.4f, sd=%.4f, n=%d\n", mean(gd), sd(gd), length(gd)))
cat(sprintf("t=%.3f, df=%.1f, p(greater)=%s;  95%% CI for mean: [%.4f, %.4f]\n\n",
            tt$statistic, tt$parameter, fmt_p(tt$p.value), tt$conf.int[1], tt$conf.int[2]))

# (b) Wilcoxon signed-rank: median(gd) > 0 (nonparametric)
if (all(gd == gd[1])) {
  cat("Wilcoxon skipped: all goal differences equal (degenerate).\n\n")
```

```r
} else {
  wt <- suppressWarnings(wilcox.test(gd, mu = 0, alternative = "greater", exact = FALSE, conf.int = TRUE))
  cat(sprintf("Wilcoxon signed-rank: median(gd)>0  |  p(greater)=%s;  95%% CI for median diff: [%.4f, %.4f]\n\n",
              fmt_p(wt$p.value), wt$conf.int[1], wt$conf.int[2]))
}

# (c) Sign test (binomial on signs, ignoring zeros): P(gd>0) > 0.5
npos <- sum(gd_nonzero > 0)
nneg <- sum(gd_nonzero < 0)
nsign <- npos + nneg
cat(sprintf("Sign counts (gd>0, gd<0): %d, %d  |  n (nonzero) = %d\n", npos, nneg, nsign))
if (nsign > 0) {
  bt_sign <- binom.test(npos, nsign, p = 0.5, alternative = "greater")
  cat(sprintf("Sign test p(greater) = %s; P(gd>0) = %.4f, 95%% CI %s\n\n",
              fmt_p(bt_sign$p.value), npos/nsign, ci_str(bt_sign$conf.int)))
} else {
  cat("Sign test skipped: all goal differences are zero.\n\n")
}

# 6) Poisson rate test: total home goals vs total away goals (same exposure = #matches)
subsection("Poisson rate test: total Home goals vs Away goals per match")
sum_HG <- sum(dat$FTHG, na.rm = TRUE)
sum_AG <- sum(dat$FTAG, na.rm = TRUE)
pt_poisson <- poisson.test(c(sum_HG, sum_AG), T = c(n_total, n_total), alternative = "greater")
cat(sprintf("Totals: Home goals=%d, Away goals=%d (matches=%d each)\n", sum_HG, sum_AG, n_total))
cat(sprintf("Rate ratio (Home/Away) = %.4f;  95%% CI %s;  p(greater)=%s\n\n",
            pt_poisson$estimate, ci_str(pt_poisson$conf.int), fmt_p(pt_poisson$p.value)))

```

```r
# 7) Intercept-only logistic model: HomeWin vs Not (Wald test that p > 0.5)
subsection("Intercept-only logistic model: P(Home win) > 0.5 (HomeWin vs Not)")
HomeWin <- as.integer(dat$FTR == "H")
glm0 <- suppressWarnings(glm(HomeWin ~ 1, family = binomial()))
est <- coef(glm0)[1]
se  <- sqrt(vcov(glm0)[1,1])
z   <- (est - 0) / se  # test logit(p) > 0 iff p > 0.5
p_one_sided <- 1 - pnorm(z)
p_hat <- plogis(est)
ci95 <- plogis(est + c(-1,1) * qnorm(0.975) * se)
cat(sprintf("Estimated P(Home win) = %.4f;  95%% CI %s\n", p_hat, ci_str(ci95)))
cat(sprintf("Wald z = %.3f; p(greater) = %s for H0: P(Home win) <= 0.5\n\n", z, fmt_p(p_one_sided)))

# 8) Multinomial-style constraint test: H vs A equality with draws present via simple contrast
# This is effectively the binomial test already run on decisive matches; we also show a z-test on p_H - p_A.
subsection("Approximate z-test on difference P(H) - P(A) (including draws)")
# Variance under null approx using two-sample proportions with equal n
phat_diff <- p_H - p_A
p_pool <- (n_H + n_A) / (2 * n_total)
se_diff <- sqrt(p_pool * (1 - p_pool) * (1/n_total + 1/n_total))
z_pa <- ifelse(se_diff > 0, phat_diff / se_diff, NA)
p_greater <- ifelse(is.na(z_pa), NA, 1 - pnorm(z_pa))
cat(sprintf("Diff = %.4f;  z = %.3f;  p(greater) = %s\n\n", phat_diff, z_pa, fmt_p(p_greater)))

# ---------- Per-league stratified tests (if Div present) ----------
if ("Div" %in% names(dat)) {
  section("Per-league directional tests (selected)")
  leagues <- sort(unique(na.omit(dat$Div)))
  if (length(leagues) == 0) {
    cat("No league identifiers found in Div.\n")
  } else {
    res <- data.frame(
```

```
      Div = character(),
      n = integer(),
      P_H = numeric(),
      P_A = numeric(),
      P_H_gt_A_p = numeric(),
      Mean_GD = numeric(),
      t_p_gd_gt0 = numeric(),
      stringsAsFactors = FALSE
    )
    for (lg in leagues) {
      sli <- dat[dat$Div == lg, , drop = FALSE]
      n <- nrow(sli)
      nH <- sum(sli$FTR == "H")
      nA <- sum(sli$FTR == "A")
      pH <- nH / n
      pA <- nA / n
      # prop test H>A (including draws)
      pt_lg <- suppressWarnings(prop.test(c(nH, nA), c(n, n),
         alternative = "greater", correct = TRUE))
      # t-test on GD
      gd_lg <- sli$FTHG - sli$FTAG
      tt_lg <- suppressWarnings(t.test(gd_lg, mu = 0, alternative
          = "greater"))
      res <- rbind(res, data.frame(
        Div = lg, n = n, P_H = pH, P_A = pA,
        P_H_gt_A_p = as.numeric(pt_lg$p.value),
        Mean_GD = mean(gd_lg),
        t_p_gd_gt0 = as.numeric(tt_lg$p.value),
        stringsAsFactors = FALSE
      ))
    }
    # Print summary table
    print(res[order(res$P_H_gt_A_p), ], row.names = FALSE)
    cat("\nSmaller p-values indicate stronger evidence that home
       advantage > away for that league.\n\n")
  }
}
```

Listing 4: Home Advantage Inferential Code

E Appendix All Leagues ML Techniques R code

```r
# Load libraries
library(tidyverse)
library(caret)

# Load data
data <- read.csv("Big_Five_Data.csv", fileEncoding="latin1")

# Drop rows with missing FTR
data <- data[!is.na(data$FTR), ]

# Select numeric features for clustering/classification (
    customize as needed)
features <- c('FTHG', 'FTAG', 'HTHG', 'HTAG', 'HS', 'AS', 'HST',
    'AST', 'HC', 'AC', 'HF', 'AF', 'HY', 'AY', 'HR', 'AR')
data_model <- data[, c(features, 'FTR')]

# Remove rows with NAs in selected columns
data_model <- na.omit(data_model)

library(randomForest)

# Encode FTR as factor
data_model$FTR <- as.factor(data_model$FTR)

# Split data (70% train, 30% test)
set.seed(7406)
train_idx <- createDataPartition(data_model$FTR, p=0.7, list=
    FALSE)
train <- data_model[train_idx, ]
test <- data_model[-train_idx, ]

# Train Random Forest
rf_model <- randomForest(FTR ~ ., data=train, ntree=100)
# Predict
pred <- predict(rf_model, test)
# Confusion Matrix
confusionMatrix(pred, test$FTR)

# Standardize numeric features
data_num <- scale(data_model[, features])

# Run k-means (choose k, e.g., 3)
```

```r
set.seed(7406)
kmeans_res <- kmeans(data_num, centers=3, nstart=25)

# Add cluster label to data
data_model$cluster <- as.factor(kmeans_res$cluster)

# See if clusters relate to FTR
table(data_model$cluster, data_model$FTR)


library(caret)
library(randomForest)

# Data preparation as before
data_model$FTR <- as.factor(data_model$FTR)
set.seed(7406)

# 5-fold cross-validation
ctrl <- trainControl(method = "cv", number = 5)

# Tune mtry parameter
rf_grid <- expand.grid(mtry = c(2, 4, 6, 8, 10))

# Train and tune Random Forest
rf_tuned <- train(
  FTR ~ .,
  data = data_model,
  method = "rf",
  trControl = ctrl,
  tuneGrid = rf_grid,
  ntree = 100 # set as needed
)

# Best mtry value
print(rf_tuned$bestTune)

# Evaluation results (accuracy, kappa, etc.)
print(rf_tuned)

# Data preparation as before; kNN works best with scaled features
preProcValues <- preProcess(data_model[, features], method = c("
   center", "scale"))
```

```
data_model_knn <- data_model
data_model_knn[, features] <- predict(preProcValues, data_model[,
    features])

# 5-fold cross-validation
ctrl <- trainControl(method = "cv", number = 5)

# Tune k
knn_grid <- expand.grid(k = c(3, 5, 7, 9, 11))

# Train and tune kNN
knn_tuned <- train(
  FTR ~ .,
  data = data_model_knn,
  method = "knn",
  trControl = ctrl,
  tuneGrid = knn_grid
)

# Best k value
print(knn_tuned$bestTune)

# Evaluation results (accuracy, kappa, etc.)
print(knn_tuned)

# Example for Random Forest
rf_pred <- predict(rf_tuned, newdata = data_model)
confusionMatrix(rf_pred, data_model$FTR)

library(xgboost)
library(caret)
library(Matrix)

# Ensure target is a factor and create numeric labels for XGBoost
data_model$FTR <- as.factor(data_model$FTR)
labels <- as.numeric(data_model$FTR) - 1  # XGBoost expects
    labels from 0

# One-hot encode features for XGBoost
dummies <- dummyVars(FTR ~ ., data = data_model)
data_matrix <- predict(dummies, newdata = data_model)
```

```r
dtrain <- xgb.DMatrix(data = as.matrix(data_matrix), label =
    labels)

# Split into training and test sets (e.g., 70/30)
set.seed(7406)
train_idx <- createDataPartition(labels, p=0.7, list=FALSE)
dtrain_train <- xgb.DMatrix(data = as.matrix(data_matrix[train_
    idx,]), label = labels[train_idx])
dtrain_test <- xgb.DMatrix(data = as.matrix(data_matrix[-train_
    idx,]), label = labels[-train_idx])

# Train XGBoost model (multiclass, softmax)
num_class <- length(levels(data_model$FTR))
params <- list(
  objective = "multi:softmax",
  num_class = num_class,
  eval_metric = "merror"
)

bst <- xgb.train(
  params = params,
  data = dtrain_train,
  nrounds = 100,
  watchlist = list(eval = dtrain_test, train = dtrain_train),
  verbose = 1
)

# Predict and evaluate
preds <- predict(bst, dtrain_test)
confusionMatrix(
  factor(preds, levels = 0:(num_class-1), labels = levels(data_
     model$FTR)),
  factor(labels[-train_idx], levels = 0:(num_class-1), labels =
     levels(data_model$FTR))
)
```

Listing 5: Machine Learning Techniques Code

F Appendix Multinomial Regression R Code

```r
library(readr)
library(caret)
```

```r
library(nnet)

data <- read_csv("Big_Five_Data.csv")
data[] <- lapply(data, function(x) if(is.character(x)) as.factor(
    x) else x)
data$FTR <- as.factor(data$FTR)

set.seed(7406)
trainIndex <- createDataPartition(data$FTR, p = 0.7, list = FALSE
    )
trainData <- data[trainIndex, ]
testData  <- data[-trainIndex, ]

# Show number of unique levels for each factor column
sapply(trainData, function(x) if(is.factor(x)) length(unique(x))
    else NA)

# Let's pick only numeric columns or low-cardinality factors
is_low_cardinality <- function(x) (is.factor(x) && length(unique(
    x)) <= 4) || is.numeric(x)
low_card_cols <- names(Filter(is_low_cardinality, trainData))
low_card_cols <- setdiff(low_card_cols, "FTR")
print(low_card_cols)
low_card_cols <- setdiff(low_card_cols, "Attendance")
low_card_cols <- setdiff(low_card_cols, "X.5")
low_card_cols <- setdiff(low_card_cols, "...1")
print(low_card_cols)
# If you have at least two such predictors, try:
formula <- as.formula(paste("FTR ~", paste(low_card_cols,
    collapse = " + ")))
multinom_model <- multinom(formula, data = trainData)
summary(multinom_model)
```

Listing 6: Multinomial Regression Code

G Appendix All Leagues ML Techniques R code

```r
# Load libraries
library(tidyverse)
library(caret)

# Load data
```

```r
data <- read.csv("Big_Five_Data.csv", fileEncoding="latin1")

# Drop rows with missing FTR
data <- data[!is.na(data$FTR), ]

# Select numeric features for clustering/classification (
    customize as needed)
features <- c('FTHG', 'FTAG', 'HTHG', 'HTAG', 'HS', 'AS', 'HST',
    'AST', 'HC', 'AC', 'HF', 'AF', 'HY', 'AY', 'HR', 'AR')
data_model <- data[, c(features, 'FTR')]

# Remove rows with NAs in selected columns
data_model <- na.omit(data_model)

library(randomForest)

# Encode FTR as factor
data_model$FTR <- as.factor(data_model$FTR)

# Split data (70% train, 30% test)
set.seed(7406)
train_idx <- createDataPartition(data_model$FTR, p=0.7, list=
    FALSE)
train <- data_model[train_idx, ]
test <- data_model[-train_idx, ]

# Train Random Forest
rf_model <- randomForest(FTR ~ ., data=train, ntree=100)
# Predict
pred <- predict(rf_model, test)
# Confusion Matrix
confusionMatrix(pred, test$FTR)

# Standardize numeric features
data_num <- scale(data_model[, features])

# Run k-means (choose k, e.g., 3)
set.seed(7406)
kmeans_res <- kmeans(data_num, centers=3, nstart=25)

# Add cluster label to data
data_model$cluster <- as.factor(kmeans_res$cluster)
```

```r

# See if clusters relate to FTR
table(data_model$cluster, data_model$FTR)


library(caret)
library(randomForest)

# Data preparation as before
data_model$FTR <- as.factor(data_model$FTR)
set.seed(7406)

# 5-fold cross-validation
ctrl <- trainControl(method = "cv", number = 5)

# Tune mtry parameter
rf_grid <- expand.grid(mtry = c(2, 4, 6, 8, 10))

# Train and tune Random Forest
rf_tuned <- train(
  FTR ~ .,
  data = data_model,
  method = "rf",
  trControl = ctrl,
  tuneGrid = rf_grid,
  ntree = 100 # set as needed
)

# Best mtry value
print(rf_tuned$bestTune)

# Evaluation results (accuracy, kappa, etc.)
print(rf_tuned)

# Data preparation as before; kNN works best with scaled features
preProcValues <- preProcess(data_model[, features], method = c("
  center", "scale"))
data_model_knn <- data_model
data_model_knn[, features] <- predict(preProcValues, data_model[,
    features])

# 5-fold cross-validation
```

```r
ctrl <- trainControl(method = "cv", number = 5)

# Tune k
knn_grid <- expand.grid(k = c(3, 5, 7, 9, 11))

# Train and tune kNN
knn_tuned <- train(
  FTR ~ .,
  data = data_model_knn,
  method = "knn",
  trControl = ctrl,
  tuneGrid = knn_grid
)

# Best k value
print(knn_tuned$bestTune)

# Evaluation results (accuracy, kappa, etc.)
print(knn_tuned)

# Example for Random Forest
rf_pred <- predict(rf_tuned, newdata = data_model)
confusionMatrix(rf_pred, data_model$FTR)

library(xgboost)
library(caret)
library(Matrix)

# Ensure target is a factor and create numeric labels for XGBoost
data_model$FTR <- as.factor(data_model$FTR)
labels <- as.numeric(data_model$FTR) - 1  # XGBoost expects labels from 0

# One-hot encode features for XGBoost
dummies <- dummyVars(FTR ~ ., data = data_model)
data_matrix <- predict(dummies, newdata = data_model)
dtrain <- xgb.DMatrix(data = as.matrix(data_matrix), label = labels)

# Split into training and test sets (e.g., 70/30)
set.seed(7406)
train_idx <- createDataPartition(labels, p=0.7, list=FALSE)
```

```
dtrain_train <- xgb.DMatrix(data = as.matrix(data_matrix[train_
    idx,]), label = labels[train_idx])
dtrain_test <- xgb.DMatrix(data = as.matrix(data_matrix[-train_
    idx,]), label = labels[-train_idx])

# Train XGBoost model (multiclass, softmax)
num_class <- length(levels(data_model$FTR))
params <- list(
  objective = "multi:softmax",
  num_class = num_class,
  eval_metric = "merror"
)

bst <- xgb.train(
  params = params,
  data = dtrain_train,
  nrounds = 100,
  watchlist = list(eval = dtrain_test, train = dtrain_train),
  verbose = 1
)

# Predict and evaluate
preds <- predict(bst, dtrain_test)
confusionMatrix(
  factor(preds, levels = 0:(num_class-1), labels = levels(data_
      model$FTR)),
  factor(labels[-train_idx], levels = 0:(num_class-1), labels =
      levels(data_model$FTR))
)
```

Listing 7: Machine Learning Techniques Code

H Appendix: Machine Learning R Code

```
# ---- Load Required Libraries ----
library(caTools)      # For train/test split
library(randomForest) # Random Forest
library(e1071)        # SVM
library(class)        # kNN
library(xgboost)      # XGBoost

# ---- Data Loading and Cleaning ----
```

```r
df <- read.csv("Big_Five_Data.csv", fileEncoding = "latin1")

# Select key numeric columns and target
columns <- c("FTR", "FTHG", "FTAG", "HTHG", "HTAG",
             "HS", "AS", "HC", "AC", "HF", "AF")
df <- df[, columns]
df <- na.omit(df)

# ---- Train/Test Split ----
set.seed(7406)
split <- sample.split(df$FTR, SplitRatio = 0.7)
train <- subset(df, split == TRUE)
test  <- subset(df, split == FALSE)

############################################
# ---- 1. Logistic Regression ----
# NOTE: This is set up as a binary example (H vs A); modify as needed for your use case.
# For simplicity, we filter to H/A and drop draws.
# Filter to binary H vs A example
bin_train <- subset(train, FTR %in% c("H", "A"))
bin_test  <- subset(test,  FTR %in% c("H", "A"))

# Convert to factor (drops unused levels automatically)
bin_train$FTR <- factor(bin_train$FTR)
bin_test$FTR  <- factor(bin_test$FTR)

logit_model <- glm(
  FTR ~ FTHG + FTAG + HTHG + HTAG + HS + AS + HC + AC + HF + AF,
  data    = bin_train,
  family = "binomial"
)


logit_pred <- predict(logit_model, newdata = bin_test, type = "response")
logit_pred_class <- ifelse(logit_pred > 0.5, "H", "A")

cat("Logistic Regression (binary H vs A):\n")
print(table(Predicted = logit_pred_class, Actual = bin_test$FTR))

############################################
```

```
# ---- 2. Random Forest Classification ----
train$FTR <- as.factor(train$FTR)
test$FTR  <- as.factor(test$FTR)

rf_model <- randomForest(FTR ~ ., data = train, ntree = 100)
rf_pred  <- predict(rf_model, newdata = test)

cat("\nRandom Forest Classification:\n")
print(table(Predicted = rf_pred, Actual = test$FTR))

# ---- NEW: Random Forest Tuning Plot (mtry vs OOB error) ----
set.seed(7406)
rf_tune <- tuneRF(
  x          = train[, -1],        # predictors only
  y          = train$FTR,          # factor response
  ntreeTry   = 100,
  stepFactor = 2,
  improve    = 0.01,
  trace      = TRUE,
  plot       = TRUE                # this produces the tuning plot
)
# 'rf_tune' contains mtry and OOB error; the plot shows OOB error vs mtry.

############################################
# ---- 3. Support Vector Machine (SVM) ----
svm_model <- svm(FTR ~ ., data = train)
svm_pred  <- predict(svm_model, newdata = test)

cat("\nSVM Classification:\n")
print(table(Predicted = svm_pred, Actual = test$FTR))

# ---- NEW: SVM Tuning Plot (cost & gamma) ----
# We use e1071::tune.svm to perform grid search with CV and then plot.
set.seed(7406)
svm_tune <- tune.svm(
  FTR ~ .,
  data   = train,
  kernel = "radial",
  cost   = c(0.1, 1, 10),
```

```r
  gamma   = c(0.01, 0.05, 0.1)
)

cat("\nBest SVM Parameters:\n")
print(svm_tune$best.parameters)
cat("\nBest SVM Performance:\n")
print(svm_tune$best.performance)

# This produces a lattice plot of performance across the (cost,
   gamma) grid.
plot(svm_tune)

############################################
# ---- 4. k-Nearest Neighbors (kNN) ----
train_X <- train[, -1]
test_X  <- test[, -1]
train_y <- train$FTR
test_y  <- test$FTR

knn_pred <- knn(train_X, test_X, train_y, k = 5)

cat("\nkNN Classification:\n")
print(table(Predicted = knn_pred, Actual = test_y))

############################################
# ---- 5. Linear Regression (predicting FTHG) ----
linreg_train <- train[, -1]  # Exclude FTR
linreg_test  <- test[, -1]

lm_model <- lm(FTHG ~ ., data = linreg_train)
lm_pred  <- predict(lm_model, newdata = linreg_test)
mse_lm   <- mean((lm_pred - linreg_test$FTHG)^2)

cat("\nLinear Regression (MSE for FTHG):", mse_lm, "\n")

############################################
# ---- 6. Random Forest Regression (predicting FTHG) ----
rf_reg_model <- randomForest(FTHG ~ ., data = linreg_train, ntree
    = 100)
rf_reg_pred  <- predict(rf_reg_model, newdata = linreg_test)
mse_rf       <- mean((rf_reg_pred - linreg_test$FTHG)^2)

```

```r
cat("\nRandom Forest Regression (MSE for FTHG):", mse_rf, "\n")

###########################################
# ---- 7. XGBoost Regression (predicting FTHG) + Training Curves ----

# Prepare matrices (drop FTHG from features)
train_matrix <- as.matrix(linreg_train[, !(names(linreg_train) %in% "FTHG")])
test_matrix  <- as.matrix(linreg_test[,  !(names(linreg_test)  %in% "FTHG")])

dtrain <- xgb.DMatrix(data = train_matrix, label = linreg_train$FTHG)
dtest  <- xgb.DMatrix(data = test_matrix,  label = linreg_test$FTHG)

params <- list(
  objective   = "reg:squarederror",
  eval_metric = "rmse"
)

set.seed(7406)
xgb_model <- xgb.train(
  params        = params,
  data          = dtrain,
  nrounds       = 50,
  watchlist     = list(train = dtrain, test = dtest),
  print_every_n = 10
)

# Predictions and MSE
xgb_pred <- predict(xgb_model, newdata = dtest)
mse_xgb  <- mean((xgb_pred - linreg_test$FTHG)^2)

cat("\nXGBoost Regression (MSE for FTHG):", mse_xgb, "\n")

# ---- NEW: XGBoost Training Curves (RMSE vs. rounds) ----
eval_log <- xgb_model$evaluation_log
# Base R plot for train/test RMSE across boosting rounds
plot(
  eval_log$iter, eval_log$train_rmse,
```

```
  type = "l", col = "blue", lwd = 2,
  xlab = "Boosting Round",
  ylab = "RMSE",
  main = "XGBoost RMSE vs. Number of Rounds"
)
lines(
  eval_log$iter, eval_log$test_rmse,
  col = "red", lwd = 2, lty = 2
)
legend(
  "topright",
  legend = c("Train RMSE", "Test RMSE"),
  col    = c("blue", "red"),
  lty    = c(1, 2),
  lwd    = 2,
  bty    = "n"
)

###########################################
# ---- 8. k-Means Clustering (Unsupervised) ----
# Use only numeric predictors (drop FTR)
df_num <- na.omit(df[, -1])

set.seed(7406)
kmeans_model <- kmeans(df_num, centers = 3)

cat("\nk-Means Clustering (cluster sizes):\n")
print(table(kmeans_model$cluster))

###########################################
# ---- 9. Principal Component Analysis (PCA) ----
pca <- prcomp(df_num, scale. = TRUE)

cat("\nPCA Summary:\n")
print(summary(pca))

# Scree-type plot of variance explained
plot(pca, main = "PCA - Variance Explained")
```

Listing 8: Machine Learning Techniques Code

I Appendix Time Series Analysis R Code

```r
# ============================================================
# Time-Series Analysis on Football-Data Big Five Leagues
# ============================================================

library(readr)
library(dplyr)
library(lubridate)
library(stringr)
library(tsibble)
library(feasts)
library(fable)
library(tseries)
library(fabletools)  # for interpolate()
library(purrr)
library(rlang)

raw <- read_csv("Big_Five_Data.csv")

df <- raw %>%
  # Standardize & parse date (Football Data uses dd/mm/yy)
  mutate(
    Date = dmy(Date),
    Div  = as.factor(Div),
    FTHG = as.numeric(FTHG),
    FTAG = as.numeric(FTAG),
    FTR  = as.factor(FTR)   # H/D/A
  ) %>%
  filter(!is.na(Date)) %>%
  select(Date, Div, HomeTeam, AwayTeam, FTHG, FTAG, FTR)

# Build weekly league series

weekly_league <- df %>%
  mutate(week = yearweek(Date)) %>%
  group_by(Div, week) %>%
  summarise(
    matches   = n(),
    home_wins = sum(FTR == "H", na.rm = TRUE),
    .groups = "drop"
  ) %>%
```

```
  as_tsibble(index = week, key = Div) %>%
  group_by_key() %>%
  # Make all missing weeks explicit
  fill_gaps(matches = 0L, home_wins = 0L) %>%
  ungroup() %>%
  mutate(
    # define the rate only when matches > 0
    home_win_rate = if_else(matches > 0, home_wins / matches, NA_real_)
  )


# Choose a league (or loop over all) and IMPUTE missing weeks
league_code <- "E0"  # change as needed
e0 <- weekly_league %>%
  filter(Div == league_code) %>%
  arrange(week)

# If too many NAs or too short, try a monthly aggregation fallback
need_monthly_fallback <- function(tbl) {
  n_non_na <- sum(!is.na(tbl$home_win_rate))
  n_non_na < 30  # STL likes at least a few dozen obs
}

if (need_monthly_fallback(e0)) {
  message("Weekly series too sparse for STL; using MONTHLY aggregation for ", league_code)
  e0 <- df %>%
    mutate(month = yearmonth(Date)) %>%
    group_by(Div, month) %>%
    summarise(
      matches   = n(),
      home_wins = sum(FTR == "H", na.rm = TRUE),
      .groups = "drop"
    ) %>%
    as_tsibble(index = month, key = Div) %>%
    group_by_key() %>%
    fill_gaps(matches = 0L, home_wins = 0L) %>%
    ungroup() %>%
    mutate(home_win_rate = if_else(matches > 0, home_wins / matches, NA_real_)) %>%
```

```
    filter(Div == league_code) %>%
    arrange(month) %>%
    rename(week = month)  # keep a common index name for downstream code
}

# Fit a simple ARIMA on the (possibly NA) series, then interpolate missing values
imp_fit <- e0 %>% model(ARIMA(home_win_rate))
e0_imp  <- interpolate(imp_fit, e0)

# Guard: if everything is still NA or too short, stop with a clear message
n_non_na_imp <- e0_imp %>%
  as_tibble() %>%
  pull(home_win_rate) %>%
  {sum(!is.na(.))}
if (n_non_na_imp < 30) {
  stop("Not enough non-missing points after interpolation for STL in ", league_code,
       ". Consider a coarser aggregation or a different metric.")
}

# STL decomposition on IMPUTED series
dcmp <- e0_imp %>%
  model(STL(home_win_rate ~ season(window = "periodic"))) %>%
  components()

# Example plot in an interactive session:
autoplot(dcmp)

# Forecast with ARIMA
fit <- e0_imp %>% model(ARIMA(home_win_rate))

fc  <- forecast(fit, h = "26 weeks")

# Example plots:
autoplot(e0_imp, home_win_rate) + autolayer(fc)

# loop over ALL leagues
safe_stl <- function(tbl) {

```

```
  imp <- tbl %>% model(ARIMA(home_win_rate))
  tbl_imp <- interpolate(imp, tbl)
  if (sum(!is.na(tbl_imp$home_win_rate)) < 30) return(NULL)  #
     skip sparse series
  mdl <- tbl_imp %>% model(STL(home_win_rate ~ season(window = "
     periodic")))
  suppressMessages(components(mdl))
}

all_decomps <- weekly_league %>%
  group_by_key() %>%
  group_map(~ safe_stl(.x), .keep = TRUE)
```

Listing 9: TSA Code

J Appendix Winning Teams R Code

```
library(readr)
library(dplyr)
library(lubridate)
library(stringr)

# --- Helper to load and prep each league's data ---
load_league <- function(filename, league_code, league_name) {
  df <- read_csv(filename, locale = locale(encoding = "Latin1"))
  df <- df %>%
    mutate(
      League = league_name,
      Date = dmy(Date),
      Season = ifelse(month(Date) >= 8, paste0(year(Date), "/",
         year(Date) + 1), paste0(year(Date) - 1, "/", year(Date))
         )
    )
  return(df)
}

# --- Load all five leagues ---
epl <- load_league("EPL_Data.csv", "EPL", "Premier League")
fl1 <- load_league("FL1_Data.csv", "FL1", "Ligue 1")
gb  <- load_league("GB_Data.csv", "GB", "Bundesliga")
isa <- load_league("ISA_Data.csv", "ISA", "Serie A")
sll <- load_league("SLL_Data.csv", "SLL", "La Liga")
```

```
all_leagues <- bind_rows(epl, fl1, gb, isa, sll)

# --- Get all unique seasons ---
all_seasons <- sort(unique(all_leagues$Season))

# --- Loop through each season ---
results <- lapply(all_seasons, function(season_target) {

  season_data <- all_leagues %>% filter(Season == season_target)

  win_stats <- season_data %>%
    mutate(
      Winner = case_when(
        FTR == "H" ~ HomeTeam,
        FTR == "A" ~ AwayTeam,
        TRUE ~ NA_character_
      )
    ) %>%
    filter(!is.na(Winner)) %>%
    group_by(League, Winner) %>%
    summarize(Wins = n(), .groups = "drop")

  winning_teams <- win_stats %>%
    group_by(League) %>%
    slice_max(order_by = Wins, n = 1, with_ties = FALSE) %>%
    ungroup() %>%
    mutate(Season = season_target)

  return(winning_teams)
})

# --- Combine into one data frame ---
final_results <- bind_rows(results)

print(final_results)

# --- Save to CSV with all seasons ---
write_csv(final_results, "winning_teams_all_seasons.csv")

cat("See 'winning_teams_all_seasons.csv' for results across all
   25 seasons.\n")
```

```r

# ============================================================
#   Visualizations for winning_teams_all_seasons.csv
#   - Line: Wins of each league's winner across seasons
#   - Bar: Title counts per team (by league)
#   - Heatmap: Winner by season & league
#   - Boxplot: Distribution of winner wins by league
# ============================================================

# ---- Package setup ----
install_if_missing <- function(pkgs) {
  to_install <- pkgs[!pkgs %in% rownames(installed.packages())]
  if (length(to_install)) install.packages(to_install, dependencies = TRUE)
}
install_if_missing(c("readr", "dplyr", "ggplot2", "tidyr", "forcats"))

library(readr)
library(dplyr)
library(ggplot2)
library(tidyr)
library(forcats)

# ---- Paths ----
input_csv <- "winning_teams_all_seasons.csv"
out_dir   <- "plots"
if (!dir.exists(out_dir)) dir.create(out_dir, recursive = TRUE)

# ---- Load data ----
winners <- read_csv(input_csv, show_col_types = FALSE)

# Expecting columns: League, Winner, Wins, Season
required_cols <- c("League", "Winner", "Wins", "Season")
missing_cols <- setdiff(required_cols, names(winners))
if (length(missing_cols)) {
  stop(sprintf("Missing columns in CSV: %s", paste(missing_cols, collapse = ", ")))
}

# ---- Season ordering (chronological) ----
# Seasons formatted like "2010/2011" -> order by first year
```

```r
season_start_year <- suppressWarnings(as.numeric(substr(winners$
    Season, 1, 4)))
season_levels <- winners %>%
  mutate(start_year = season_start_year) %>%
  distinct(Season, start_year) %>%
  arrange(start_year, Season) %>%
  pull(Season)

winners <- winners %>%
  mutate(
    Season = factor(Season, levels = season_levels),
    League = as.factor(League),
    Winner = as.factor(Winner)
  )

# ------------------------------
# 1) Line: Wins by league winners across seasons
# ------------------------------
p_line <- ggplot(winners, aes(x = Season, y = Wins, group =
    League, color = League)) +
  geom_line(linewidth = 1) +
  geom_point() +
  labs(
    title = "Number of Wins by League Winners Across Seasons",
    x = "Season",
    y = "Wins"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    panel.grid.minor = element_blank()
  )

ggsave(file.path(out_dir, "wins_by_winners_line.png"), p_line,
    width = 12, height = 7, dpi = 150)

# ------------------------------
# 2) Bar: Dominance -- total titles per team (faceted by league)
# ------------------------------
team_titles <- winners %>%
  group_by(League, Winner) %>%
  summarize(Titles = n(), .groups = "drop") %>%
```

```
  arrange(League, desc(Titles))

# Order teams within each league by titles for clearer bars
team_titles <- team_titles %>%
  group_by(League) %>%
  mutate(Winner = fct_reorder(Winner, Titles, .desc = TRUE)) %>%
  ungroup()

p_bar <- ggplot(team_titles, aes(x = Winner, y = Titles, fill =
    League)) +
  geom_col() +
  coord_flip() +
  facet_wrap(~ League, scales = "free_y") +
  labs(
    title = "Total Titles Won by Teams (All Seasons)",
    x = "Team",
    y = "Number of Titles"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "none",
    panel.grid.minor = element_blank()
  )

ggsave(file.path(out_dir, "titles_per_team_faceted.png"), p_bar,
    width = 12, height = 8, dpi = 150)

# ------------------------------
# 3) Heatmap: Winner by season & league
# ------------------------------
# (Optional) shorten long team names for readability
shorten <- function(x, maxlen = 18) ifelse(nchar(x) > maxlen,
    paste0(substr(x, 1, maxlen - 1), "..."), x)
winners$Winner_short <- shorten(as.character(winners$Winner))

p_heat <- ggplot(winners, aes(x = Season, y = League, fill =
    Winner_short)) +
  geom_tile(color = "white", linewidth = 0.2) +
  labs(
    title = "League Winners by Season",
    x = "Season",
    y = "League",
```

```r
    fill = "Winning Team"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    axis.text.x = element_text(angle = 90, vjust = 0.5, hjust = 1),
    panel.grid = element_blank()
  )

ggsave(file.path(out_dir, "winners_heatmap.png"), p_heat, width = 14, height = 5.5, dpi = 150)

# ----------------------------
# 4) Boxplot: Distribution of winner wins by league
# ----------------------------
p_box <- ggplot(winners, aes(x = League, y = Wins, fill = League)) +
  geom_boxplot(alpha = 0.75, outlier.shape = 21) +
  labs(
    title = "Distribution of Wins for League Winners",
    x = "League",
    y = "Wins"
  ) +
  theme_minimal(base_size = 12) +
  theme(
    legend.position = "none",
    panel.grid.minor = element_blank()
  )

ggsave(file.path(out_dir, "winner_wins_boxplot.png"), p_box, width = 9, height = 6, dpi = 150)

# ----------------------------
# Done
# ----------------------------
message("Saved plots to: ", normalizePath(out_dir))
```

Listing 10: Winning Teams Code

K Appendix EPL EDA R Code

```r
library(tidyverse)
```

```r
library(DataExplorer)
library(naniar)
library(plotly)

# Load data
epl_data <- read.csv("EPL_Data.csv", fileEncoding = "latin1",
   stringsAsFactors = FALSE)
str(epl_data)

# Percentage missing per column
missing_summary <- sapply(epl_data, function(x) mean(is.na(x)))
   %>% sort(decreasing = TRUE)
missing_df <- data.frame(Column = names(missing_summary),
   PercentMissing = missing_summary)
head(missing_df, 15)

# Visualize missing data pattern
plot_missing(epl_data)

# Get summaries for numeric columns
numeric_data <- select(epl_data, where(is.numeric))
summary(numeric_data)

# Convert Date to Date format
epl_data$Date <- as.Date(epl_data$Date, format="%d/%m/%y")
# Remove NA dates if present
epl_data <- epl_data %>% filter(!is.na(Date))

# Plot total goals per season
epl_data %>%
  mutate(Season = lubridate::year(Date)) %>%
  group_by(Season) %>%
  summarise(
    Total_Home_Goals = sum(FTHG, na.rm = TRUE),
    Total_Away_Goals = sum(FTAG, na.rm = TRUE)
  ) %>%
  pivot_longer(cols = starts_with("Total"), names_to = "Type",
     values_to = "Goals") %>%
  ggplot(aes(x = Season, y = Goals, fill = Type)) +
  geom_col(position = "dodge") +
  labs(title = "Total Home/Away Goals per Season", x = "Season",
     y = "Goals")
```

```
ggplot(epl_data, aes(x = FTHG)) +
  geom_histogram(bins = 20) +
  labs(title = "Distribution of Full-Time Home Goals", x = "Goals
     ", y = "Frequency")

ggplot(epl_data, aes(x = FTAG)) +
  geom_histogram(bins = 20) +
  labs(title = "Distribution of Full-Time Away Goals", x = "Goals
     ", y = "Frequency")

# Most common home teams
epl_data %>%
  count(HomeTeam, sort = TRUE) %>%
  head(10)
```

Listing 11: EPL EDA Code

L Appendix Liverpool Analysis R Code

```

# ========================================================
#   League Comparison + Liverpool (EPL) Team Analysis Script
#   Source: football-data.co.uk
# ========================================================

library(readr)
library(dplyr)
library(ggplot2)
library(lubridate)
library(tidyr)
library(stringr)
library(plotly)

# --- Helper to load and prep each league's data ---
load_league <- function(filename, league_code, league_name) {
  df <- read_csv(filename, locale = locale(encoding = "Latin1"))
  df <- df %>%
    mutate(
      League = league_name,
      Date = dmy(Date),
```

```
      Season = ifelse(month(Date) >= 8, paste0(year(Date), "/", str_sub(year(Date) + 1, 3, 4)), paste0(year(Date) - 1, "/", str_sub(year(Date), 3, 4)))
    )
  return(df)
}

# --- Load all five leagues ---
epl <- load_league("EPL_Data.csv", "EPL", "Premier League")

# ==============================
# LIVERPOOL TEAM ANALYSIS (EPL)
# ==============================
liverpool <- epl %>%
  filter(HomeTeam == "Liverpool" | AwayTeam == "Liverpool") %>%
  mutate(
    LiverpoolIsHome = HomeTeam == "Liverpool",
    Opponent = if_else(LiverpoolIsHome, AwayTeam, HomeTeam),
    GoalsFor = if_else(LiverpoolIsHome, FTHG, FTAG),
    GoalsAgainst = if_else(LiverpoolIsHome, FTAG, FTHG),
    Result = case_when(
      (LiverpoolIsHome & FTR == "H") | (!LiverpoolIsHome & FTR == "A") ~ "Win",
      FTR == "D" ~ "Draw",
      TRUE ~ "Loss"
    ),
    Venue = if_else(LiverpoolIsHome, "Home", "Away")
  )

# Overall Liverpool stats
liverpool_overall <- liverpool %>%
  group_by(Result) %>%
  summarize(Matches = n())
write_csv(liverpool_overall, "liverpool_overall_results.csv")

# Liverpool season by season
liverpool_season <- liverpool %>%
  group_by(Season) %>%
  summarize(
    Matches = n(),
    Wins = sum(Result == "Win"),
    Draws = sum(Result == "Draw"),
```

```r
    Losses = sum(Result == "Loss"),
    GoalsFor = sum(GoalsFor),
    GoalsAgainst = sum(GoalsAgainst),
    Avg_GoalsFor = round(mean(GoalsFor),2),
    Avg_GoalsAgainst = round(mean(GoalsAgainst),2),
    Points = Wins * 3 + Draws * 1
  ) %>%
  arrange(Season)
write_csv(liverpool_season, "liverpool_season_summary.csv")

# Home vs Away Liverpool performance
liverpool_venue <- liverpool %>%
  group_by(Venue, Season) %>%
  summarize(
    Matches = n(),
    Wins = sum(Result == "Win"),
    Draws = sum(Result == "Draw"),
    Losses = sum(Result == "Loss"),
    Avg_GoalsFor = round(mean(GoalsFor),2),
    Avg_GoalsAgainst = round(mean(GoalsAgainst),2)
  ) %>%
  arrange(Season, Venue)
write_csv(liverpool_venue, "liverpool_home_away_summary.csv")

# Plot: Liverpool's Points per Season
p1 <- ggplot(liverpool_season, aes(x = Season, y = Points)) +
  geom_line(group = 1, color = "red", linewidth = 1.2) +
  geom_point(color = "black", size = 2) +
  theme_minimal() +
  labs(title = "Liverpool Total Points Per Season (EPL)", x = "
     Season", y = "Points") +
  theme(axis.text.x = element_text(angle=45, hjust=1))
ggsave("liverpool_points_per_season.png", p1, width=8, height=4)

# Plot: Liverpool Goals For and Against Per Season
p2 <- ggplot(liverpool_season, aes(x = Season)) +
  geom_point(aes(y = Avg_GoalsFor, color = "Goals For"), size =
     1) +
  geom_point(aes(y = Avg_GoalsAgainst, color = "Goals Against"),
     size = 1) +
  theme_minimal() +
```

```
  labs(title = "Liverpool Goals For and Against (Per Match, Per Season)", x = "Season", y = "Goals Per Match") +
  scale_color_manual("", values = c("Goals For" = "red", "Goals Against" = "black")) +
  theme(axis.text.x = element_text(angle=45, hjust=1))
ggsave("liverpool_goals_for_against.png", p2, width=8, height=4)

# Interactive Plots
ggplotly(p1)
ggplotly(p2)


# --- End of Script ---
cat("Analysis complete. See CSVs and PNGs for summary tables and plots.")
```

Listing 12: Liverpool Analysis Code

M Appendix Liverpool Prediction R Code

```
# EPL Points Prediction for Liverpool

library(readr)
library(dplyr)
library(ggplot2)
library(lubridate)
library(tidyr)
library(stringr)
library(plotly)
library(caret)
library(randomForest)

# --- Helper to load and prep each league's data ---
load_league <- function(filename, league_code, league_name) {
  df <- read_csv(filename, locale = locale(encoding = "Latin1"))
  df <- df %>%
    mutate(
      League = league_name,
      Date = dmy(Date),
      Season = ifelse(month(Date) >= 8, paste0(year(Date), "/",
                                               str_sub(year(Date) + 1, 3, 4)),
```

```r
                      paste0(year(Date) - 1, "/", str_sub(year(Date), 3, 4)))
    )
  return(df)
}

# --- Load all five leagues ---
epl <- load_league("EPL_Data.csv", "EPL", "Premier League")

# ================================
# LIVERPOOL TEAM ANALYSIS (EPL)
# ================================
liverpool <- epl %>%
  filter(HomeTeam == "Liverpool" | AwayTeam == "Liverpool") %>%
  mutate(
    LiverpoolIsHome = HomeTeam == "Liverpool",
    Opponent = if_else(LiverpoolIsHome, AwayTeam, HomeTeam),
    GoalsFor = if_else(LiverpoolIsHome, FTHG, FTAG),
    GoalsAgainst = if_else(LiverpoolIsHome, FTAG, FTHG),
    Result = case_when(
      (LiverpoolIsHome & FTR == "H") | (!LiverpoolIsHome & FTR == "A") ~ "Win",
      FTR == "D" ~ "Draw",
      TRUE ~ "Loss"
    ),
    Venue = if_else(LiverpoolIsHome, "Home", "Away")
  )

# Overall Liverpool stats
liverpool_overall <- liverpool %>%
  group_by(Result) %>%
  summarize(Matches = n())
write_csv(liverpool_overall, "liverpool_overall_results.csv")

# Liverpool season by season
liverpool_season <- liverpool %>%
  group_by(Season) %>%
  summarize(
    Matches = n(),
    Wins = sum(Result == "Win"),
    Draws = sum(Result == "Draw"),
    Losses = sum(Result == "Loss"),
```

```
    GoalsFor = sum(GoalsFor),
    GoalsAgainst = sum(GoalsAgainst),
    Avg_GoalsFor = round(mean(GoalsFor),2),
    Avg_GoalsAgainst = round(mean(GoalsAgainst),2),
    Points = Wins * 3 + Draws * 1
  ) %>%
  arrange(Season)
write_csv(liverpool_season, "liverpool_season_summary.csv")

# for reproducibility
set.seed(7406)
n <- nrow(liverpool_season)
train_idx <- sample(seq_len(n), size = 0.8 * n)

train_data <- liverpool_season[train_idx, ]
test_data <- liverpool_season[-train_idx, ]

model_rf <- train(Points ~ Wins + Draws + Losses + Avg_GoalsFor + Avg_GoalsAgainst,
                  data = train_data, method = "rf")
model_rf

model_lm <- train(
  Points ~ Wins + Draws + Losses + Avg_GoalsFor + Avg_GoalsAgainst,
  data = train_data,
  method = "lm"
)
model_lm


# Linear model predictions
test_data$Predicted_Points <- predict(model_lm, newdata = test_data)

# Random forest predictions
test_data$Predicted_Points_RF <- predict(model_rf, newdata = test_data)

# Root Mean Squared Error for linear model
rmse_lm <- sqrt(mean((test_data$Points - test_data$Predicted_Points)^2))
```

```
# For random forest
rmse_rf <- sqrt(mean((test_data$Points - test_data$Predicted_
    Points_RF)^2))

print(rmse_lm)
print(rmse_rf)

# --- End of Script ---
cat("Analysis complete. See CSVs and PNGs for summary tables and
    plots.")
```

Listing 13: Liverpool Prediction Code

N Appendix: Graphs and Plots

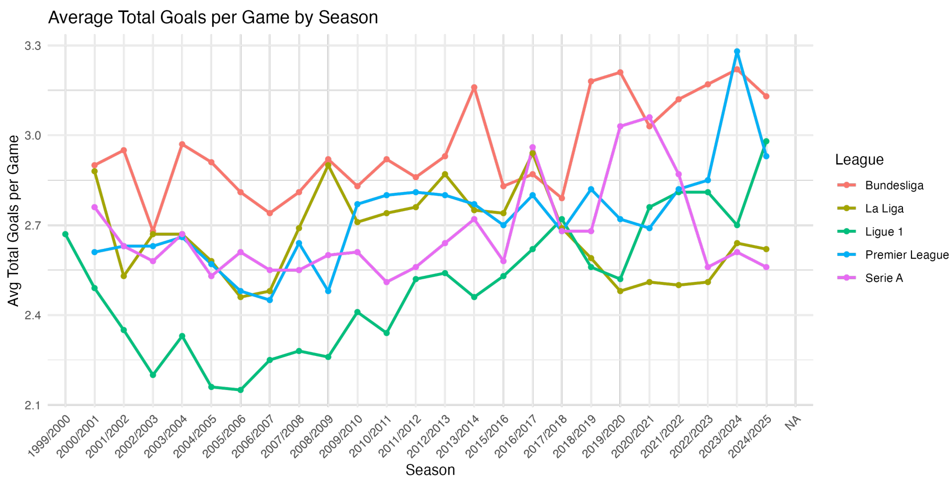

Figure 12: Goals by League and Season

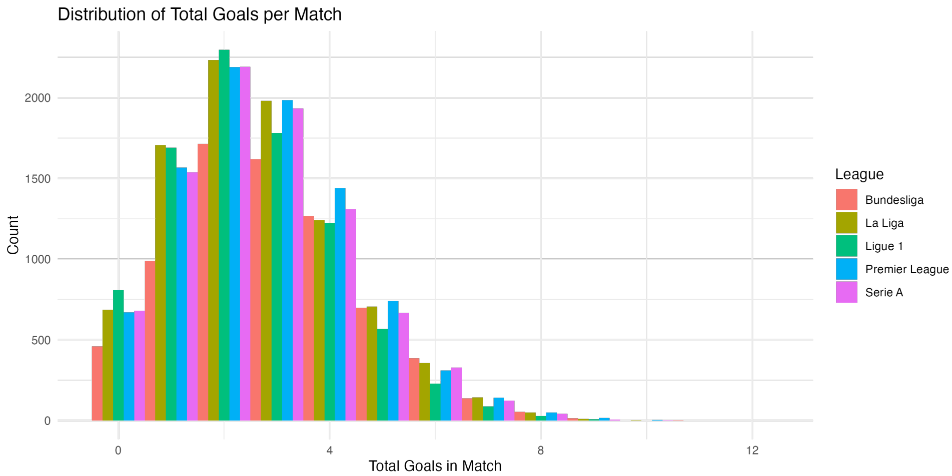

Figure 13: Distribution of Total Goals Per Match

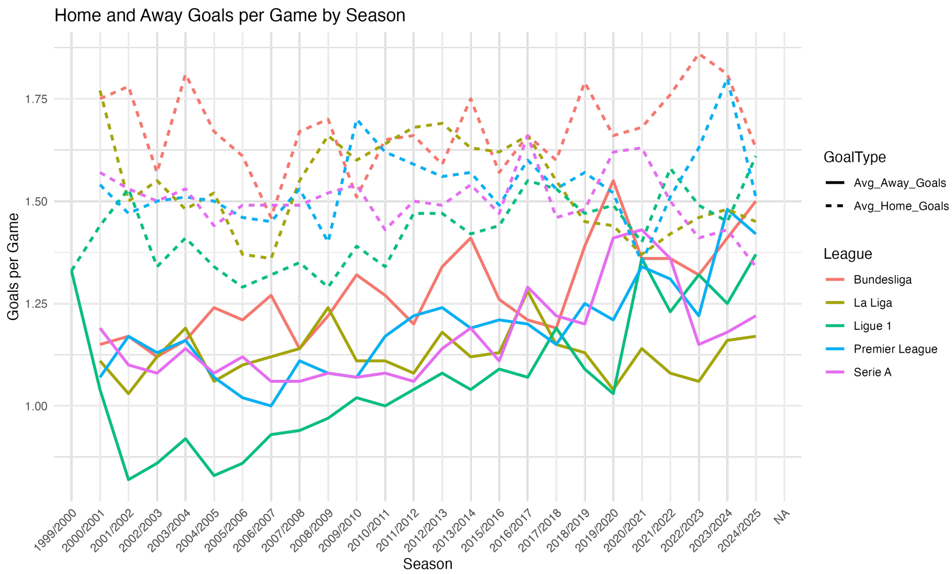

Figure 14: Home versus Away Goals Per Season by League

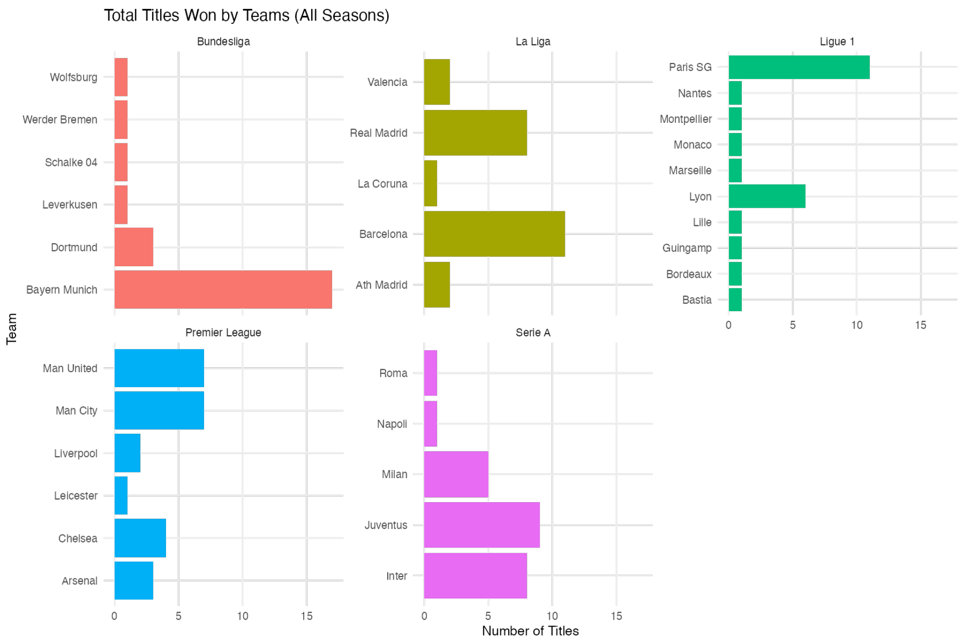

Figure 15: Titles by Team and League

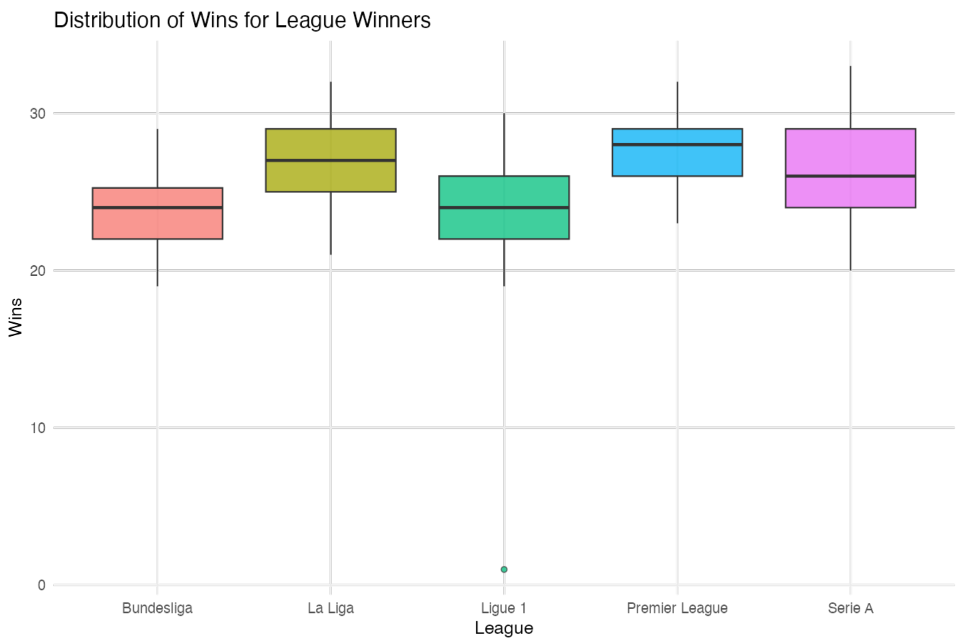

Figure 16: Distribution of Wins by Title Winners Per League

Bibliography and Credits

Agresti, A. (2013). *Categorical data analysis* (3rd ed.). Wiley. https://onlinelibrary.wiley.com/doi/book/10.1002/0471249688

Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32. https://doi.org/10.1023/A:1010933404324

Cawley, G. C., & Talbot, N. L. C. (2010). On over-fitting in model selection and subsequent selection bias in performance evaluation. *Journal of Machine Learning Research*, *11*, 2079–2107. https://jmlr.org/papers/v11/cawley10a.html

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. https://doi.org/10.1145/2939672.2939785

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297. https://link.springer.com/article/10.1007/BF00994018

Courneya, K. S., & Carron, A. V. (1992). The home advantage in sport competitions: A literature review. *Journal of Sport and Exercise Psychology*, *14*(1), 13–27. https://doi.org/10.1123/jsep.14.1.13

Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27. https://doi.org/10.1109/TIT.1967.1053964

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction* (2nd ed.). Springer. https://web.stanford.edu/~hastie/ElemStatLearn/

Kuhn, M. (2008). Building predictive models in r using the `caret` package. *Journal of Statistical Software*, *28*(5), 1–26. https://www.jstatsoft.org/article/view/v028i05

Maher, M. J. (1982). Modelling association football scores. *Statistica Neerlandica*, *36*(3), 109–118. https://doi.org/10.1111/j.1467-9574.1982.tb00782.x

Murtagh, F., & Legendre, P. (2014). Ward's hierarchical agglomerative clustering method: Which algorithms implement ward's criterion? *Journal of Classification*, *31*(3), 274–295. https://doi.org/10.1007/s00357-014-9161-z

Nevill, A. M., & Holder, R. L. (1999). Home advantage in sport: An overview of studies on the advantage of playing at home. *Sports Medicine*, *28*(4), 221–236. https://doi.org/10.2165/00007256-199928040-00001

Pollard, R. (2006). Home advantage in soccer: A retrospective analysis. *Journal of Sports Sciences*, *24*(3), 231–240. https://doi.org/10.1080/02640410500141836

Scrucca, L., Fop, M., Murphy, T. B., & Raftery, A. E. (2016). Mclust 5: Clustering, classification and density estimation using gaussian finite mixture models. *The R Journal*, *8*(1), 289–317. https://journal.r-project.org/archive/2016/RJ-2016-021/index.html

Ward, J. H. (1963). Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, *58*(301), 236–244. https://doi.org/10.1080/01621459.1963.10500845