



Thesis Project in Data Science

Applied AI 180 credits

Predicting Premier League's match outcome via machine learning: New chess-inspired features can enhance predictions

Thesis Project in Data Science 15 credits

Halmstad 16 May, 2025

Christoffer Karlsson and Caio Araujo Rossmann Leite

Supervisor: Yuantao Fan

Predicting Premier League's match outcome via machine learning: New chess-inspired features can enhance predictions

16 May, 2025

Contents

Contents	i
1 Introduction	1
1 Background	1
2 Problem Statement	1
3 Purpose	2
4 Requirements	2
5 Related Work	2
2 Technical Background	3
1 Historical Approaches	3
2 Challenges	3
3 Supervised Learning for Football Match Prediction	4
4 Unsupervised Learning for Football Match Prediction	4
5 Feature Selection	4
5.1 Mutual Information (MI)	5
5.2 Lasso (L1 Regularization)	5
6 Incorporate ELO Rating as a feature	5
7 Gaussian Mixture Model (GMM)	6
8 Performance Metrics	6
8.1 Accuracy	6
8.2 Precision, Recall and F1-score	6
8.3 Confusion Matrix	7
8.4 Relevance to Our Models	7
9 Models Choices	7
9.1 Random Forest	7
9.2 XGBoost	8
9.3 Logistic Regression	8
10 Model Explainability and Feature Importance	8
3 Method	9
1 Data Collection	9
2 Data Preprocessing	10
3 Data Exploration	10
4 Baseline Model	11
5 Feature Selection	11
6 Feature Engineering	12

6.1	Play-Style Feature	12
6.2	Elo Rating with Goal Difference	12
6.3	Adjusted Form Features	13
7	Model Development	13
7.1	Data Preparation	13
7.2	Random Forest	14
7.3	XGBoost	15
7.4	Logistic Regression	15
4	Implementation	17
1	Elo Rating	17
2	Adjusted Form Features	18
3	Play-Style Feature	19
4	Features from other work	21
4.1	Form Differential	21
4.2	Streak Differential	22
4.3	Goal Difference	23
4.4	Past k goals Differential	24
5	Model Development	25
5.1	Feature Selection:	25
5.2	Random Forest	25
5.3	XGBoost	26
5.4	Logistic Regression (One-vs-Rest)	27
5	Results	29
1	Data Exploration	29
2	Baseline Model Results	30
2.1	Cross-Validation	31
2.2	Feature Importance with SHAP	32
3	Results of Our Models with Engineered Features	32
3.1	XGBoost	32
3.2	Multinomial Logistic Regression	33
3.3	Random Forest	34
4	Comparison to Baseline Model	37
6	Discussion	41
1	Discussion of Results	41
2	Societal Aspects	43
7	Conclusion	45
1	Achievements	45
2	Recommendations for Future Work	46
	References	47

Abstract

This thesis explores how machine learning models can be used to predict football match outcomes. Using data from 1,738 Premier League matches across four and a half seasons, we compare a baseline Random Forest classifier based on the work of Baboota & Kaur [1] with our own Random Forest model enhanced by dynamic, context-aware features. These include Elo-based strength ratings, form trends, team play-style, and adjusted performance metrics relative to the opponent. To address class imbalance, especially the underrepresentation of draws, we applied class weighting and a custom probability threshold. For evaluation, we used a time-aware fixed split: 60% of the data for training, 20% for validation, and the final 20% for testing. After tuning the model using the validation set, we retrained on 80% and evaluated on the final 20%. On this test set, our final model achieved a macro F1-score of 0.50, outperforming the baseline model. The most notable improvement was in predicting draws, where the baseline achieved an F1-score of 0.10 and our final model reached 0.29. To ensure robustness, we conducted 15-fold time series cross-validation. This revealed a more nuanced picture: the final model had a higher mean macro F1-score across folds (0.441 vs. 0.424), although the difference was not statistically significant. Feature engineering, especially with Elo-based and form-differentiated variables, improved class balance and interpretability. SHAP analysis further illustrated how context-rich features in the final model contributed more distinctly to predictions than the baseline's static attributes. This study demonstrates that thoughtfully engineered features can lead to more balanced football outcome predictions.

Sammanfattning

Denna rapport undersöker hur maskininlärningsmodeller kan användas för att förutsäga resultat i fotbollsmatcher. Med data från 1 738 Premier League-matcher över fyra och en halv säsong jämför vi en baslinjemodell baserad på en replikerad Random Forest-klassificerare från Baboota & Kaur [1] arbete med vår egen Random Forest-modell, som har förbättrats med dynamiska och kontextmedvetna egenskaper. Dessa inkluderar Elo-baserade styrkebetyg, formtrender, lagens spelstil och justerade prestationsmått i relation till motståndaren. För att hantera klassobalans, särskilt den låga förekomsten av oavgjorda matcher, använde vi klassviktning och ett anpassat sannolikhetsröskelvärde. För utvärdering använde vi en tidsbaserad uppdelning: 60% av datan för träning, 20% för validering och de sista 20% för testning. Efter att ha finjusterat modellen med hjälp av valideringsdatan tränade vi om modellen på 80% av datan och utvärderade den på de sista 20%. På testdatan uppnådde vår slutgiltiga modell ett makro-F1-värde på 0,50, vilket överträffade baslinjemodellen. Den mest anmärkningsvärda förbättringen sågs vid förutsägelser av oavgjorda matcher, där baslinjemodellen hade ett F1-värde på 0,10 och vår slutmodell nådde 0,29. För att säkerställa robusthet genomförde vi en 15-faldig tidsseriekorsvalidering. Detta visade en mer nyanserad bild: den slutgiltiga modellen hade ett högre genomsnittligt makro-F1-värde över foldsen (0,441 jämfört med 0,424), även om skillnaden inte var statistiskt signifikant. Funktionsdesignen, särskilt de som baserades på Elo och formdifferentiering, förbättrade klassbalansen och tolkbarheten. SHAP-analys visade dessutom hur de kontextuella egenskaperna i slutmodellen bidrog tydligare till förutsägelsena än basmodellens statistiska attribut. Studien visar att noggrant designade funktioner kan leda till mer balanserade förutsägelser av fotbollsresultat.

Chapter 1

Introduction

The English Premier League (EPL) is one of the most competitive and globally popular football leagues. Predicting match outcomes in the EPL is of significant interest to various stakeholders, including fans, betting companies, analysts and even football clubs themselves. This project aims to leverage Machine Learning to predict the outcomes of football matches.

Such predictions can have multiple applications. Fans can benefit even with the predicted outcome by gaining early insight into their team's expected performance. This can boost engagement, fuel discussions, and add excitement as they track predictions against real results and determine if betting companies are putting fair odds for those games. Betting companies can benefit by improving their predictive models to maintain competitiveness in such a dynamic industry. For football clubs, explainable models can provide insights into why specific outcomes are likely, helping them prepare for the upcoming season. Knowing their projected position in the league table can give clubs a head start in managing their finances, whether preparing to sell a key player or investing in new talent.

1 Background

The increasing availability of football-related data and advances in AI have enabled more sophisticated predictive models. While previous studies have primarily focused on historical results and team statistics, this project seeks to innovate by incorporating a form-based metric that considers recent team strength and a playstyle feature that will analyze the way teams plays.

2 Problem Statement

Predicting match outcomes is highly challenging due to football's dynamic and unpredictable nature. Numerous factors influence the result, including player injuries, home advantage, coaching decisions, red cards, and even weather conditions. Rather than relying solely on raw data, this research emphasizes the con-

struction of meaningful features derived from historical match statistics. Our goal is to develop a more accurate predictive model by capturing key aspects of a team's play style and recent form. The match forecasting task is framed as a multi-class classification problem. Let $\mathbf{x} \in \mathbb{R}^n$ represent a feature vector containing match-related attributes, and let $y \in \{0, 1, 2\}$ indicate the target class corresponding to away win (0), draw (1), and home win (2). The goal is to train a classifier function $f : \mathbf{x} \mapsto y$, where f is trained on a dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ consisting of historical match data. Once trained, the classifier estimates the most likely result by analyzing new feature inputs representing upcoming fixtures.

3 Purpose

The primary objective of this thesis is to develop an AI-based system capable of predicting the outcomes of EPL matches with high accuracy over all classes, specifically focusing on a high macro average F1-score. Unlike traditional models that rely mainly on team performance and historical data, our model will introduce playstyle components and form-based features to enhance predictions. This involves combining and transforming raw statistical data into meaningful features that better capture the tactical behavior and recent performance trends of each team.

4 Requirements

The system should be able to analyze statistical data from previous matches, including team performance and historical outcomes. It should include tactical features and use AI methods to achieve balanced predictions across all outcome classes. The system should also take into account form-based indicators, such as recent performances and differences in team playstyles. Since the task is a multi-class classification problem, the model's performance will be measured using Accuracy, Precision, Recall, and F1-Score, with a focus on achieving a high macro F1-Score as the main sign of a good model.

5 Related Work

Most previous studies predict football match outcomes using historical data, machine learning models (Random Forest, SVM, Gradient Boosting), betting odds, or player-based evaluations. Some focus on beating bookmakers, while others rank players based on their contribution to team success. However, they lack a tactical approach and form analysis. We want to go beyond just historical data by integrating playstyle features (Defensive, Neutral and Offensive) and form-based metrics (recent performances of the teams and ratings of teams). This makes our model more dynamic, context-aware and useful for football analysis rather than just betting or player ranking.

Chapter 2

Technical Background

1 Historical Approaches

Previous work shows that the application of machine learning has become a key tool in predicting football outcomes. Among some relevant studies in our field, Hampus Bergqvist explored whether machine learning could beat the bookies [3], while Alexander Jäderberg and Pontus Linde focused on optimizing match predictions using Support Vector Machines (SVM) and feature selection techniques[2]. Another study by Niek Tax and Yme Joustra examined the Dutch Eredivisie; their study incorporated various metrics like team performance, home advantage, streaks, and managerial changes. They applied various machine learning models such as Naïve Bayes, Random Forest and LogitBoost[8]. All these studies struggled particularly in predicting draws, largely because draws are underrepresented in the dataset. This class imbalance makes it difficult for models to learn the patterns associated with draws outcomes. They all highlighted the importance of feature selection, alternative data sources and model optimization. Achieving class balance in football outcome prediction is of high importance, especially because of the naturally imbalanced distribution of match results where draws are less frequent than wins or losses. In real-world usage, the ability to predict all three outcomes is critical, particularly in domains such as betting or tactical planning. Therefore, a model receiving a high accuracy can be misleading if it is only able to predict two of a total of three outcomes.

2 Challenges

Given the dynamic nature of football, where tactical adjustments, player fatigue, and injuries frequently occur during matches, accurately predicting game outcomes based on end result statistics is highly challenging.

3 Supervised Learning for Football Match Prediction

Supervised Learning is the task of learning to predict a numerical or a categorical output for a given input sample. In such problems, you will either obtain or create a dataset with clearly marked outputs [5]. Supervised learning involves training a model on labeled data to make future predictions. In this project, we use classification models, where the goal is to assign one of three classes (Win, Draw, Loss) to a given match based on historical and form-based data.

- **Classification:** Classification is used when predicting categorical outcomes (e.g., Win, Draw, or Loss).
- **Regression:** Regression could be used to predict continuous values (e.g., exact scorelines), but this project focuses on classification due to its practical relevance in football analytics.

4 Unsupervised Learning for Football Match Prediction

Unsupervised learning is the task of discovering patterns, groupings, or structures in data without using labeled outputs. Instead of learning from predefined categories or values, the model analyzes the underlying structure of the data to find meaningful clusters or relationships [5]. In this project, we apply unsupervised learning to identify distinct team playstyles. By using historical match data and rolling averages of tactical features, the model groups teams based on how they typically play, without prior knowledge of predefined styles.

- **Clustering:** where the goal is to identify distinct groups within the data.
- **Dimension reduction:** Which aims to find a simpler, more compact representation of the data while preserving important information.

5 Feature Selection

Feature selection involves pruning away non-useful features to reduce the complexity of a model and improve computational efficiency without significantly impacting predictive accuracy [9]. In the context of football match prediction, selecting the right features can significantly improve model accuracy and interpretability. This project employs two feature selection methods to determine the most impactful variables for predicting match outcomes (Win, Draw, Loss).

We are using two different filter methods (MI and Lasso). Filtering techniques preprocess features to remove ones that are unlikely to be useful for the model. For example, one could compute the correlation or mutual information between each feature and the response variable, and filter out the features that fall below a threshold [9].

5.1 Mutual Information (MI)

Mutual Information measures the dependency between two random variables as a non-negative value [11]. In the context of feature selection, it quantifies how much information a feature provides about the target variable.

5.2 Lasso (L1 Regularization)

Lasso (Least Absolute Shrinkage and Selection Operator) is a linear model that uses an L1 penalty to shrink less important feature weights to zero. This makes it effective for feature selection in datasets with a large number of variables. It optimizes using coordinate descent and is sensitive to the alpha parameter, which controls the strength of regularization bigger values mean more feature weights get pushed to zero. This helps keep the model simpler and avoids overfitting by removing features that don't add much. [14]

6 Incorporate ELO Rating as a feature

For our prediction model, we propose using Elo Rating as a feature. The Elo rating system was originally developed by Arpad Elo for ranking chess players and has since been used in different sports to estimate the relative strength of competitors. In football, different types of modified Elo ratings has been used to track team performance over time and provide a numerical measure of form. Each team is assigned a rating, typically starting from a default value (e.g. 1500). After every match the ratings are updated based on the actual outcome compared to the expected outcome. The fundamental idea is that beating a stronger team results in a larger rating increase, while losing to a weaker team results in a larger decrease. The formula for Elo rating update can be seen in Equation 2.1.

$$R_{\text{new}} = R_{\text{old}} + K \cdot (S - E) \quad (2.1)$$

Where:

- R_{old} and R_{new} are the team's ratings before and after the match.
- K is a constant that controls the update speed (e.g. 20–40 in football settings).
- S is the match outcome (1 for a win, 0.5 for a draw, 0 for a loss).
- E is the expected score, computed as in Equation 2.2.

$$E = \frac{1}{1 + 10^{(R_{\text{opponent}} - R_{\text{team}})/400}} \quad (2.2)$$

Elo Rating reflect team strength, momentum, and performance trends better than static league position or win percentages. By incorporating Elo as a feature in a machine learning model, it becomes possible to give the model a meaningful indicator of how strong each team is relative to its opponent.

7 Gaussian Mixture Model (GMM)

Gaussian Mixture Model (GMM) is a soft clustering technique used in unsupervised learning that assigns probabilities to each data point for belonging to different clusters. This approach allows for overlapping clusters with more flexible, elliptical shapes, rather than forcing hard boundaries [15]. As a result, GMM can capture more nuanced groupings in complex datasets.

8 Performance Metrics

Evaluating the performance of machine learning models in football match prediction requires appropriate metrics that account for the multiclass nature of the problem (Win, Draw, Loss). Since predictions are not binary but instead belong to three possible categories, standard classification metrics must be adapted accordingly. The following key metrics are used to assess model performance:

8.1 Accuracy

measures the proportion of correctly predicted outcomes out of all predictions:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.3)$$

Equation: TP denote True Positives, TN denote True Negatives, FP denote False Positives, FN denote False Negatives. The formula calculates the ratio of correct predictions (true positives and true negatives) to the total number of predictions. While accuracy is useful, it can be misleading if one class (e.g. Wins) is significantly more frequent than others (e.g. Draws), leading to biased predictions.

8.2 Precision, Recall and F1-score

Given the imbalanced nature of football match results, Precision (2.4), Recall (2.5), and F1-score (2.6) offer more detailed insights than accuracy alone.

Precision

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2.4)$$

Equation: Precision measures how often a predicted class (Win, Draw, or Loss) is actually correct.

Recall

$$\text{Recall} = \frac{TP}{TP + FN} \quad (2.5)$$

Equation: Recall measures how many actual instances of a class were correctly predicted.

F1-score

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \quad (2.6)$$

Equation: The F1-score is the harmonic mean of Precision and Recall, and is particularly useful when dealing with imbalanced classes.

Macro F1-score

$$\text{Macro F1-score} = \frac{1}{N} \sum_{i=1}^N \text{F1}_i \quad (2.7)$$

Equation: Macro F1 averages the F1-scores across all classes, treating each class equally regardless of its frequency.

—
In this project, particular attention was given to the macro-averaged F1-score (2.7), which treats each class equally regardless of its frequency. This is important to ensure that the model performed consistently across all outcomes and did not neglect less frequent classes like draws.

8.3 Confusion Matrix

A confusion matrix provides a breakdown of how the model performs across each class (Win, Draw, Loss), showing the number of correct and incorrect predictions for each category. This helps in identifying where the model struggles the most (e.g. systematically misclassifying Draws).

8.4 Relevance to Our Models

Random Forest primarily outputs class labels, making evaluation metrics such as accuracy, F1-score, and the confusion matrix particularly relevant to our models. However, given the imbalance in football match outcomes where draws are less frequent than wins or losses, metrics like F1-score and class-specific recall offer more meaningful insights than accuracy alone.

9 Models Choices

9.1 Random Forest

Random Forest is an ensemble learning method that operates by constructing multiple decision trees and aggregating their predictions to improve accuracy and reduce overfitting[13]. It is a widely used algorithm for classification tasks due to its robustness and interpretability. It is also good for capturing non linear relationships within the data. Random Forest doesn't inherently track time-dependent

patterns like form or momentum. To compensate for its lack of sequential awareness, additional features such as rolling averages of recent performances and goal trends are incorporated to enhance its predictive power. The model will be re-trained as new data becomes available since it does not update dynamically.

9.2 XGBoost

XGBoost is an optimized version of gradient boosting and is a type of ensemble learning method. It uses decision trees as its base learners, combining them sequentially to improve the model's performance [13]. XGBoost is also known for its speed and accuracy, making it a strong choice for this project. We used it as a classification model to predict match outcomes (Win, Draw, Loss).

9.3 Logistic Regression

Logistic Regression is a statistical algorithm that analyzes the relationship between two data factors[13]. The goal is to predict the probability that an instance belongs to a given class or not. It is particularly effective for binary and multiclass classification. In this project, we use Logistic Regression to predict the outcomes of matches (Win, Draw, Loss).

10 Model Explainability and Feature Importance

Understanding which features contribute most to a machine learning model's predictions is especially important when applied settings like sports analytics and prediction. In this study we will use two approaches to evaluate feature importance and model interpretability.

First the Random Forest algorithm includes a built-in feature importance mechanism, which measures the average decrease in impurity brought by each feature across all decision trees in the ensemble.

We will also use SHAP (SHapley Additive exPlanations). SHAP values measure the contribution of each feature to the model's output. Unlike the feature importance method used in Random Forest, which only considers the average importance of features across the training set. SHAP interpret the prediction for a single sample in terms of how much each feature contributed to the output, providing a deep insight into specific decisions made by the model [13].

Chapter 3

Method

1 Data Collection

The data used in this project was collected from FBref.com, a well-established public source for detailed football statistics. This platform provides comprehensive historical data including match results, team performances, player statistics, and advanced metrics such as expected goals (xG).

To automate the data extraction process, we developed a custom web scraper using Python's `BeautifulSoup` library. This tool allowed us to systematically parse HTML content and extract relevant information such as match dates, teams, scores, xG, and various team-level performance metrics. The scraper was designed to navigate season-by-season pages and collect data across multiple English Premier League seasons.

Each match record in the dataset contains key attributes such as:

- Date of the match
- Home and away team
- Match outcome (Win, Draw, Loss)
- Goals scored by each team
- Passing types (short passes, long passes, crosses)
- Shot statistics (on target, off target, and distance)
- Possession (Defensive third, Middle third, Attacking third)
- Defensive statistics (e.g. tackles in different areas of the pitch)
- Team formations
- Goalkeeping statistics (saves, crosses faced, defensive actions outside the box)
- Different types of xG values

All extracted data was stored in structured CSV format for further preprocessing and feature engineering. By automating the collection pipeline, we ensured both reproducibility and scalability in gathering historical match data.

2 Data Preprocessing

Effective data preprocessing is crucial for building a robust predictive model. The following steps were undertaken to clean and prepare the data.

First, the dataset was sorted by the Date column, which was converted into DateTime format to facilitate temporal operations. Only Premier League games were retained for analysis, with all matches from other competitions removed to maintain dataset consistency.

To handle missing and duplicate data, we began by removing one match from the 2020 season that had incomplete values. Additionally, several columns were excluded because they either represented percentage values or other metrics that were simple mathematical combinations of existing features and therefore did not contribute additional information. Other columns were removed due to a high number of missing values, imputing them would risk introducing bias or inaccuracies, and they were not essential to our predictive goals. A detailed list of the removed features and the rationale behind their exclusion is provided in Table 3.1.

Table 3.1: Features that were removed due to missing values or percentage values that are already represented in other ways

Feature Name	Description	Reason for Removal
G/SoT	Goals per shot on target	Mathematical combination of existing features
Save%	Save percentage (shots saved / shots on target faced)	Percentage value
GK 40 Cmp%	Goalkeeper pass completion percentage within 40 yards	Percentage value
Gks Launch%	Percentage of goalkeeper passes that were launched (long)	Percentage value
Gks AvgLen	Average length of goalkeeper passes	Too many missing values
Stp%	Percentage of crosses stopped by the goalkeeper	Percentage value
AvgDist	Average distance of goalkeeper defensive actions	Too many missing values

3 Data Exploration

For data exploration, we looked into the distribution of match outcomes, We wanted to see how often home, away wins and draws occurred over the whole dataset and also per season. We also explored the distribution of goals, examining how many goals were typically scored in each match and how this varied over time. This analysis helped us identify general trends, assess data imbalance and

gain insight into potential biases.

4 Baseline Model

As a baseline we implemented a Random Forest model by replicating the methodology used by Baboota & Kaur [1] in their predictive analysis of English Premier League outcomes. We recreated their feature set using a newer dataset that spans more recent seasons than those used in their study. The features used for this model can be seen in the Table. [3.2]

Feature name	Feature abbreviation
Form differential	FormDifferential
Streak differential	StDifferential
Past k shots on target differential	STKPP
Past k goals differential	GKPP
Past k corners differential	CKPP
Attack rating differential	RelAttack
Midfield rating differential	RelMidField
Defence rating differential	RelDefense
Overall rating differential	RelOverall
Goal difference differential	GDDifferential
Weighted streak differential	StWeightedDifferential

Table 3.2: List of replicated features

One important notable difference from the original paper was in the treatment of team rating statistics obtained from FIFA. These ratings were no longer available for the most recent season in our dataset. To maintain consistency, we addressed this by assigning each team the same rating values they had at the end of the last season for which FIFA data was available. This allowed for continuity in the model’s input structure while also acknowledging the limitation in data availability.

The Random Forest model was trained on the engineered features using the same multi-class classification setup as in the original study, aiming to predict match outcomes as one of three possible classes: Outcomes = {home win, draw, away win}

5 Feature Selection

For feature selection, we explored two different approaches to ensure we captured both linear and non-linear relationships in the data. First, we used Lasso regression, which is a linear model that performs feature selection by shrinking less important feature coefficients to zero. We also applied Mutual Information, a non-linear method that measures the dependency between each feature and the

target variable. Combining these two methods gave us a more robust understanding of which features were more relevant for our project.

6 Feature Engineering

6.1 Play-Style Feature

To create our characteristic of the play style, we selected 20 features that reflect the playing style of a team, such as possession, crosses, and passing in the final third. We computed the rolling average over the last three matches to incorporate recent form and provide temporal context. Using these features, we applied an unsupervised learning model, specifically the Gaussian Mixture Model (GMM), to group teams based on their play style. Since GMM requires specifying the number of clusters (k). We determined the optimal value using two methods:

- Bayesian Information Criterion (BIC) [17]: measures model fit while penalizing complexity to prevent overfitting.
- Calinski-Harabasz score: The score is defined as the ratio of between-cluster dispersion to within-cluster dispersion [16].

6.2 Elo Rating with Goal Difference

To model evolving team strength over time, we incorporated a dynamic Elo rating system tailored for football. While Elo itself is a well-established algorithm used for team and player ranking, our method includes several key adaptations that are, to our knowledge, novel in the context of machine learning-based football outcome prediction.

First, we modified the Elo update by introducing a goal difference multiplier to better reflect the magnitude of a win. For non-draw results, the update was scaled using the log-based multiplier shown in Equation 3.1:

$$\text{Multiplier} = \log(|\text{Goal Difference}| + 1) \quad (3.1)$$

Second, to account for changes in team composition and performance between seasons, we applied a seasonal blending reset. At the start of each season, a team's rating was partially regressed toward the mean using the formula in Equation 3.2:

$$E_i^{\text{new}} = 0.8 \cdot E_i^{\text{last}} + 0.2 \cdot E_0 \quad (3.2)$$

where $E_0 = 1500$ is the base Elo rating.

Finally, instead of using the home and away Elo ratings as two separate features, we reduced dimensionality and emphasized relative strength by using their difference, as shown in Equation 4.8:

$$\Delta E = E_{\text{home}} - E_{\text{away}} \quad (3.3)$$

This difference (ΔE) was used as one of the core input features for our predictive models. Together, these adaptations allowed us to capture both absolute and comparative team strength dynamically, and to integrate Elo into a machine learning context in a way that reflects temporal performance and match outcome characteristics.

6.3 Adjusted Form Features

To capture short-term team performance trends, we engineered a set of form-based features using raw match statistics. However, simply averaging these statistics over time ignores the context of opponent strength. For example, accumulating 10 shots against a top-tier team is more impressive than doing the same against a bottom-ranked side.

As a novel contribution of this work, we adjusted each team’s raw match statistics based on the Elo rating of their opponent. To our knowledge, this adjustment method—scaling team performance features relative to opponent Elo—has not been applied in previous machine learning models for football outcome prediction.

The adjusted value was computed by scaling the raw statistic with the ratio between the opponent’s Elo and a baseline Elo of 1500, as shown in Equation 3.4:

$$\text{AdjustedStat} = \text{RawStat} \times \left(\frac{\text{OpponentElo}}{\text{BaseElo}} \right) \quad (3.4)$$

This formulation increases the impact of strong performances against high-quality teams and down-weights performances against weaker opposition. We used a baseline Elo of 1500 for consistency with our initial team Elo rating.

To capture recent form while avoiding data leakage, we computed rolling averages of these adjusted values using a 3-game history, excluding the current match. The rolling average was calculated as shown in Equation 3.5:

$$\text{RollingForm}_i = \frac{1}{N} \sum_{j=1}^N \text{AdjustedStat}_{i-j} \quad (3.5)$$

where $N = 3$, and $\text{AdjustedStat}_{i-j}$ is the adjusted statistic from the j -th previous match. This process created a set of rolling, opponent-aware features that reflect both recent performance and the difficulty of recent fixtures. These were used as inputs to the prediction model, offering a more nuanced view of team form than raw averages alone.

7 Model Development

7.1 Data Preparation

To predict the outcomes of English Premier League matches, this project treated the task as a multiclass classification problem with three possible classes: *Home*

Win, Draw, and Away Win. A `RandomForestClassifier` was selected for its ability to handle non-linear interactions, resistance to overfitting, and support for feature importance analysis. In addition to Random Forest, other models such as XGBoost and Logistic Regression (One-vs-Rest) were also implemented and evaluated.

The dataset was chronologically sorted and split into three parts:

- **Training set** (60%) — used for model fitting,
- **Validation set** (20%) — used for hyperparameter tuning and draw threshold optimization,
- **Test set** (20%) — used solely for final model evaluation.

Chronological splitting was used to preserve the temporal integrity of the data and avoid any forward-looking bias. This is particularly important in football modeling, where features such as form and Elo ratings are sequentially dependent on match history.

The models used a diverse feature set:

- **Rolling average features**, based on Elo-adjusted statistics,
- **Engineered differentials**, such as Elo Difference, Form Differential, and Goal Difference,
- **Opponent-adjusted statistics**, like `Adj_GF_Diff`, `Adj_Points_Diff`, etc.,
- **Categorical Play-Style indicators** as cluster-based features.

7.2 Random Forest

Hyperparameter tuning was performed using grid search over five key parameters:

- `n_estimators` – number of decision trees,
- `max_depth` – maximum tree depth,
- `min_samples_split` – minimum samples to split a node,
- `min_samples_leaf` – minimum samples at a leaf node,
- `max_features` – number of features considered at each split.

The search space was evaluated based on accuracy and macro-averaged F1-score using the validation set. After selecting the best model configuration, an additional step was introduced to enhance **draw prediction performance**, which is typically the most challenging class. A custom thresholding strategy was applied to the model's predicted probabilities: if the draw class probability exceeded a tuned threshold (e.g. 0.33), a draw prediction was forced.

This threshold was optimized on the validation set by maximizing macro F1-score. The final evaluation was conducted using this threshold strategy applied to the unseen test set.

7.3 XGBoost

Hyperparameter tuning was performed using grid search over five key parameters:

- `learning_rate` – controls step size reduction,
- `max_depth` – maximum tree depth,
- `n_estimators` – number of boosting rounds,
- `subsample` – fraction of samples used for tree building,
- `colsample_bytree` – fraction of features used per tree.

The search space was evaluated based on accuracy and macro-averaged F1-score using the validation set. After selecting the best model configuration, an additional step was introduced to enhance draw prediction performance, which is typically the most challenging class. A custom thresholding strategy was applied to the model's predicted probabilities: if the draw class probability exceeded a tuned threshold (e.g. 0.33), a draw prediction was forced. This threshold was optimized on the validation set by maximizing macro F1-score. The final evaluation was conducted using this threshold strategy applied to the unseen test set.

7.4 Logistic Regression

Hyperparameter tuning was performed using grid search over four key parameters:

- `C` – regularization strength,
- `penalty` – regularization type,
- `class_weight` – weight assignment strategy,
- `solver` – optimization algorithm,

The search space was evaluated based on macro-averaged F1-score using the validation set. After selecting the best model configuration, an additional step was introduced to enhance **draw prediction performance**, which is typically the most challenging class. A custom thresholding strategy was applied to the model's predicted probabilities: if the draw class probability exceeded a tuned threshold (e.g. 0.33), a draw prediction was forced. This threshold was optimized on the validation set by maximizing macro F1-score after that the model was retrained using both the train set and validation set. The final evaluation was conducted using this threshold strategy applied to the unseen test set.

Chapter 4

Implementation

1 Elo Rating

The Elo rating system was implemented to track team strength over time based on match results. Match data was sorted in chronological order to ensure accurate rating updates. Each team began with an initial Elo score of 1500.

Initialization

- Set initial Elo rating constant:

$$E_0 = 1500 \quad (4.1)$$

- For each team i , initialize:

$$E_i = E_0 \quad (4.2)$$

Expected Score Calculation The expected score for a team i against opponent j was calculated using the standard Elo formula:

$$\hat{S}_i = \frac{1}{1 + 10^{(E_j - E_i)/400}} \quad (4.3)$$

Rating Update After each match, ratings were updated using the formula in Equation 4.6 according to the rules in Equation 4.4 & 4.5:

- If the result is a win/loss:
 - Compute multiplier:

$$M = \log(\text{goaldiff} + 1) \quad (4.4)$$

- If the result is a draw:
 - Set multiplier:

$$M = 1 \quad (4.5)$$

- Update Elo rating:

$$E'_i = E_i + k \cdot M \cdot (S_i - \hat{S}_i) \quad (4.6)$$

where:

- E'_i : updated Elo rating for team i
- k : scaling factor (constant)
- S_i : actual result (1 = win, 0.5 = draw, 0 = loss)
- \hat{S}_i : expected result from Equation 4.3
- M : goal-difference multiplier (see Equations 4.4 and 4.5)

Season Reset Adjustment At the beginning of each new season, Elo scores were partially reset to account for team changes while preserving some historical performance. This was done with a weighted average:

$$E_i^{\text{new}} = 0.8 \cdot E_i^{\text{last}} + 0.2 \cdot E_0 \quad (4.7)$$

This retains 80% of the final Elo score from the previous season while blending in 20% of the default starting rating (1500, from Equation 4.1).

Feature Engineering from Elo For each match, Elo-based features were calculated for home and away teams as:

- Home team's adjusted Elo: E_{home}
- Away team's adjusted Elo: E_{away}

To emphasize relative team strength:

$$\Delta E = E_{\text{home}} - E_{\text{away}} \quad (4.8)$$

This feature (ΔE) represents the Elo rating difference and was used to capture the contextual strength of the home team versus the away team.

2 Adjusted Form Features

The process began by generating a match-level dataset using a custom function. Elo ratings were merged into this dataset for both home and away teams. The correct Elo rating was assigned to each row based on whether the team was playing at home or away. This value represented the opponent's Elo rating and was stored in a new column called `Opponent_Elo`.

Each match stat was adjusted using the following formula:

$$A = R \cdot \left(\frac{E_{\text{opp}}}{E_0} \right) \quad (4.9)$$

where:

- R = base rating adjustment constant
- E_{opp} = opponent's Elo rating
- E_0 = baseline Elo rating (e.g., 1500)

This formula increases or decreases the value of a raw statistic depending on the strength of the opponent. For example, a high opponent Elo boosts the impact of the stat, reflecting the difficulty of the match.

This adjustment was applied to the following features: GA, GF, xGA, GD, PassLive, Att_Pen, CPA, TB, PPA, and Points.

The constant E_0 was set to 1500.

Example: To compute the adjusted version of the PassLive statistic:

- Retrieve PassLive for each team.
- Multiply it by the ratio: E_{opp} / E_0 .
- Store the result as Adj_PassLive.

Rolling Averages To measure short-term form, a rolling average with a window size of 3 was applied to the adjusted statistics. A temporal shift was used to ensure only past games were included in the average, avoiding lookahead bias.

This was applied to all adjusted columns using a custom `rolling_adjusted()` function, grouped by Team and Season. The resulting features, such as `Rolling_Adj_PassLive`, `Rolling_Adj_PPA`, and `Rolling_Adj_Points`, were concatenated back into the main dataset. The final DataFrame contained both the adjusted match-level statistics and their rolling averages. For model input, we did not use separate `Rolling_Adj_*` features for home and away teams. Instead, we computed a single difference feature for each statistic to represent the relative form between the two teams:

$$\Delta R_s = R_s^{\text{home}} - R_s^{\text{away}} \quad (4.10)$$

Where:

- ΔR_s = Rolling feature difference for statistic s
- R_s^{home} = Rolling average of statistic s for the home team
- R_s^{away} = Rolling average of statistic s for the away team

This approach reduces dimensionality and directly captures the performance gap between the two teams. For each feature (e.g. `Rolling_Adj_PassLive`, `Rolling_Adj_PPA`, `Rolling_Adj_Points`), the difference between the home and away teams was calculated and used as a single input feature in the prediction model. This mirrors the strategy used earlier for Elo rating difference and keeps the feature set consistent and interpretable.

3 Play-Style Feature

The Playstyle feature was implemented by selecting 20 features [4.1]:

Table 4.1: Playstyle Features and Descriptions

Feature Name	Description
Poss	Possession percentage
PassLive	Completed live-ball passes
T0	Take-ons that lead to a shot
Def	Total defensive actions
Att.1, Att.2, Att.3	Attempted Short, Medium and Long Passes
Passing 1/3	Passes into the final third
PPA	Passes into the penalty area
CrsPA	Crosses into the penalty area
PrgP	Progressive passes
T_Def_Pen	Tackles in the defensive penalty area
T_Def_3rd	Tackles in the defensive third
T_Mid_3rd	Tackles in the midfield third
T_Att_3rd	Tackles in the attacking third
Att_Pen	Attacking touches in the penalty area
Tk1W	Tackles won
Tkl_Def_3rd	Tackles won in defensive third
Tkl_Mid_3rd	Tackles won in midfield third
Tkl_Att_3rd	Tackles won in attacking third

Rolling Averages for All Features To apply rolling averages across multiple adjusted features, the following steps were performed per team:

- Sort the match data chronologically by date.
- For each selected feature column:
 - Shift the values forward by one match to exclude the current game.
 - Apply a rolling mean with a window size of 3.
- Store the resulting values in new columns.
- Remove rows where any of the rolling values are missing due to insufficient history.

We tried two different methods to determine the number of clusters. BIC since it is well-suited for probabilistic models like Gaussian Mixture Models and helps balance model fit with complexity when capturing team playstyle patterns [4.1]. We also used the Calinski-Harabasz Index as an alternative approach. However, the results from the Calinski-Harabasz index did not align well with the structure we expected from the data and were not used in the final decision. We selected the number of clusters based on the result from BIC, which indicated that three clusters provided the best trade-off. These clusters represent different playstyles: offensive, balanced and defensive. We then incorporated them into two new features in the match dataset.

To use this tactical context into our model we made a new feature called *Clus-*

ter Matchup, which combines the home and away teams' play-style clusters into a single categorical variable (e.g, "0_1"). Importantly, we did **not** pass the individual team clusters to the model. Instead, we only included the *matchup* to let the model focus on how different combinations of play styles interact and influence the match outcomes. By one-hot encoding these matchups, the model can capture and learn from recurring patterns in play style pairings.

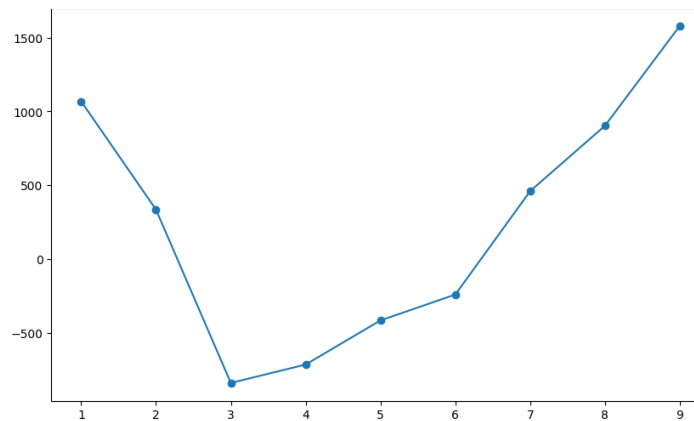


Figure 4.1: Result of BIC, BIC value on the y-axis and number of clusters evaluated on x-axis

4 Features from other work

To complement the features we engineered ourselves, we also adopted several features from an existing research paper by [1]. Their feature set includes high-level indicators of team performance and momentum that are not based on raw statistics alone, but on recursive and context-aware updates. Specifically, we integrated four of their features into our model:

- Form Differential
- Streak Differential
- Goal Difference
- Past k goals Differential (GKPP)

These features were implemented based on the formulas and methodology outlined in their work. Each is described in detail below, beginning with Form Differential.

4.1 Form Differential

The Form Differential feature is designed to represent a team's recent momentum by updating a running "form value" for each team based on match outcomes, while taking into account the strength of the opponent. This is done using a process

where teams "steal" form points from one another depending on the result of a match [1].

Each team's form is initialized to a value of 1.0 at the start of every season and is updated sequentially after each match. The update is based on match outcome and the opponent's form rating, using a fixed coefficient $\gamma = 0.33$ to control the amount of form "transferred" between teams. Wins transfer form from the losing team to the winning team, while draws equalize the form difference between both teams. These updates occur one match at a time in chronological order, ensuring temporal consistency. At the start of each new season, form values are reset to 1.0 for all teams. After each match, the form values are updated as follows:

Win Scenario (Team α beats Team β):

- Update team strengths after a win:

$$\xi_{\alpha}^{(j)} = \xi_{\alpha}^{(j-1)} + \gamma \cdot \xi_{\beta}^{(j-1)} \quad (4.11)$$

$$\xi_{\beta}^{(j)} = \xi_{\beta}^{(j-1)} - \gamma \cdot \xi_{\beta}^{(j-1)} \quad (4.12)$$

Draw Scenario:

- Update strengths in case of a draw:

$$\xi_{\alpha}^{(j)} = \xi_{\alpha}^{(j-1)} - \gamma \cdot (\xi_{\alpha}^{(j-1)} - \xi_{\beta}^{(j-1)}) \quad (4.13)$$

$$\xi_{\beta}^{(j)} = \xi_{\beta}^{(j-1)} - \gamma \cdot (\xi_{\beta}^{(j-1)} - \xi_{\alpha}^{(j-1)}) \quad (4.14)$$

Where:

- $\xi^{(j)}$ is the form rating of the team after match j
- γ is a constant called the stealing fraction, set to 0.33

This system allows wins against strong teams to increase a team's form more significantly, and losses against weaker teams to reduce it more harshly. For each match, we computed the Form Differential by subtracting the away team's form from the home team's. This resulting value was used as an input feature in the prediction model to capture a comparative measure of recent team performance and momentum.

4.2 Streak Differential

The Streak Differential feature captures recent trends in match outcomes for each team, quantifying whether a team has been consistently winning, drawing, or losing. It was designed to reflect the current winning tendency of a team over its most recent matches [1].

For a given team i , the **streak value** before match j is calculated using the following formula:

$$\delta_j^{(i)} = \frac{1}{3k} \sum_{p=j-k}^{j-1} \text{resp}_p^{(i)} \quad (4.15)$$

In this formula, k represents the rolling window size, which determines the number of past matches taken into account, typically set to 6 based on prior research. The term $\text{resp}_p^{(i)}$ denotes the result of match p for team i , encoded as a numerical value where 3 indicates a win, 1 indicates a draw, and 0 indicates a loss.

By summing the results from the last k matches and normalizing by $3k$, the streak value $\delta_j^{(i)}$ produces a score between 0 and 1 that reflects a team's recent form, closer to 1 for winning streaks and closer to 0 for losing streaks.

Once streak values are computed for both the home and away team for a given match, the **Streak Differential** is defined as:

$$\text{Streak Differential} = \delta_j^{\text{Home}} - \delta_j^{\text{Away}} \quad (4.16)$$

This differential value captures the relative momentum between the two teams. A positive value indicates the home team is on a stronger winning streak than the away team, while a negative value implies the opposite.

Practical Example:

- If a team has won 5 of its last 6 games:

$$\delta = \frac{3 + 3 + 3 + 3 + 3 + 0}{3 \cdot 6} = \frac{15}{18} \approx 0.83$$

- If the opponent has drawn 3 and lost 3:

$$\delta = \frac{1 + 1 + 1 + 0 + 0 + 0}{18} = \frac{3}{18} \approx 0.17$$

Then:

$$\text{Streak Differential} = 0.83 - 0.17 = 0.66$$

This would indicate a significant positive momentum for the home team.

4.3 Goal Difference

The Goal Difference is a metric that captures the balance between goals scored and goals conceded by a team across previous matches. It is a simple but effective feature for representing a team's strength in both offense and defense [1].

For a team entering its k -th match, the **Goal Difference (GD)** is defined as:

$$\text{GD}_k = \sum_{j=1}^{k-1} \text{GS}_j - \sum_{j=1}^{k-1} \text{GC}_j \quad (4.17)$$

Where:

- GS_j is the number of goals scored by the team in match j ,
- GC_j is the number of goals conceded by the team in match j ,
- The sum runs from match 1 to match $k-1$, so that the value does **not** include the current match (avoiding data leakage).

This feature is computed independently for each team from the beginning of each season. The authors explicitly note that no values are carried over between seasons, ensuring the metric is seasonal in scope.

To create a single comparative feature between the two teams in a match, the paper introduces the **Goal Difference Differential**:

$$\text{GDDifferential} = \text{Home GD}_k - \text{Away GD}_k \quad (4.18)$$

This differential expresses the net goal-scoring superiority of the home team versus the away team over the course of the season up to that point.

4.4 Past k goals Differential

To capture a team's recent offensive effectiveness, the paper introduces the **GKPP** feature, which stands for the **Past k Goals Differential**. This feature reflects how many goals a team has scored on average in its most recent k matches, compared to its opponent. It is designed to quantify recent goal-scoring momentum, without factoring in the goals conceded [1].

For a given team, the past- k goals metric μ_j^{Goals} before match j is computed as:

$$\mu_j^{\text{Goals}} = \frac{1}{k} \sum_{p=j-k}^{j-1} \text{Goals}_p \quad (4.19)$$

Where:

- Goals_p is the number of goals scored by the team in match p ,
- k is a fixed rolling window size (set to 6 in the paper),
- j denotes the index of the upcoming match.

This rolling average captures the recent attacking form of a team.

To convert this into a single differential feature for a match, the **GKPP** is calculated as:

$$\text{GKPP} = \mu_{\text{Home},j}^{\text{Goals}} - \mu_{\text{Away},j}^{\text{Goals}} \quad (4.20)$$

This differential expresses whether the home team has had a stronger or weaker recent goal-scoring run compared to the away team. It serves as an important input for machine learning models, reflecting short-term scoring trends.

5 Model Development

5.1 Feature Selection:

The model was trained using a combination of engineered features and matchup-specific indicators [4.2]. Three main groups of features were included:

- **Differential Features:** Metrics that capture the difference in performance between home and away teams (e.g., Elo, form, streak, goal difference).
- **Rolling Average Features:** Time-weighted averages of adjusted statistics.
- **Cluster Matchup Features:** Each team was assigned to a cluster (based on their specific play-style). Matchups between clusters were encoded using one-hot encoding (e.g., `Cluster_Matchup_1_0` means home team in cluster 1, away team in cluster 0).

Table 4.2: Selected Features and Descriptions

Feature Name	Description
FormDifferential	Difference in team form using form transfer mechanism
StDifferential	Streak difference based on recent match outcomes
GKPP	Past- k goals differential (recent scoring form)
GD_Differential	Net goal difference over the season (Home - Away)
Adj_GA_Diff	Adjusted goals against differential
Adj_xGA_Diff	Adjusted expected goals against differential
Elo_GD_Diff	Elo rating difference with goal-diff multiplier
Adj_PassLive_Diff	Adjusted passes into final third differential
Adj_Att_Pen_Diff	Adjusted attacking penalty box entries difference
Adj_CPA_Diff	Adjusted Crosses into Penalty Area difference
Adj_TB_Diff	Adjusted through ball differential
Adj_PPA_Diff	Adjusted penalty area passes differential
Adj_Points_Diff	Adjusted points earned difference
Adj_GF_Diff	Adjusted goals for differential
Cluster_Matchup_*	One-hot encoded cluster matchup combinations

5.2 Random Forest

Hyperparameter Tuning: A brute-force grid search was performed using the validation set. The parameter grid was defined as:

Table 4.3: Hyperparameter grid for model tuning

Hyperparameter	Values
n_estimators	{100, 200, 500}
max_depth	{10, 20, None}
min_samples_split	{2, 5, 10}
min_samples_leaf	{1, 2, 5}
max_features	{sqrt, log2}

Each configuration was evaluated using accuracy and macro-averaged F1-score. The final chosen hyperparameters for the RandomForestClassifier were:

- n_estimators = 100
- max_depth = 10
- min_samples_split = 10
- min_samples_leaf = 2
- max_features = 'sqrt'
- class_weight = 'balanced'

To address class imbalance in the match outcome labels, class_weight='balanced' was used to automatically adjust weights inversely proportional to class frequencies, ensuring fairer treatment of less frequent outcomes such as draws.

Custom Threshold Tuning for Draw Prediction: A custom prediction function was used to improve draw classification. Instead of choosing the class with the highest probability, the model checked if the draw class probability exceeded a tuned threshold. If so, the output was set to draw. This threshold was tuned using the validation set to maximize macro F1-score.

The final threshold selected was **0.33**.

5.3 XGBoost

A brute-force grid search was performed using the validation set. The parameter grid was defined as:

Table 4.4: XGBoost hyperparameter grid for tuning

Hyperparameter	Values
n_estimators	{100, 200, 500}
learning_rate	{0.01, 0.1}
colsample_bytree	{0.8, 1.0}
subsample	{0.8, 1.0}
max_depth	{3, 5, 7}

The selected probability threshold for draws was **0.30** and the final parameters for the model was the following:

- `n_estimators = 100`
- `learning_rate = 0.01`
- `colsample_bytree = 1.0`
- `subsample = 0.8`
- `max_depth = 3`

5.4 Logistic Regression (One-vs-Rest)

A brute-force grid search was performed using the validation set. The parameter grid was defined as:

Table 4.5: Logistic Regression hyperparameter grid for tuning

Hyperparameter	Values
<code>C</code>	{0.01, 0.1, 1, 10}
<code>penalty</code>	{l2, l1, None}
<code>solver</code>	{liblinear, saga, lbfgs}
<code>max_iter</code>	{1000}
<code>class_weight</code>	{None, balanced}

The selected probability threshold for draws was **0.39** and the final parameters for the model was the following:

- `C = 0.01`
- `penalty = l2`
- `solver = lbfgs`
- `max_iter = 1000`
- `class_weight = balanced`

Chapter 5

Results

1 Data Exploration

We began our analysis by examining the distribution of match outcomes across four and a half Premier League seasons, focusing on the proportion of home wins, away wins, and draws. One immediate anomaly emerged during the 2020/21 season, which coincided with the peak of the COVID-19 pandemic. That season, due to the absence of crowds and other disruptions, away teams performed unusually well, winning 153 matches compared to 144 for home teams[5.2], and the goal difference was nearly minimal (509 away goals to 513 home goals). This contrasts sharply with the three following seasons, where a clear home advantage re-emerged. From 2021/22 to 2023/24, home wins and goal totals steadily increased[5.3], suggesting a return to normal conditions and the return of home-field advantage. On average across the non-COVID seasons, home teams scored 1.57 goals per match, while away teams managed only 1.36. It is worth noting that while home advantage is apparent in match outcomes and goal metrics, possession and shots on target [5.1] and other statistics remained relatively stable across all 4 and a half seasons, indicating that home advantage may be more closely tied to efficiency rather than overall ball control or shots on target.

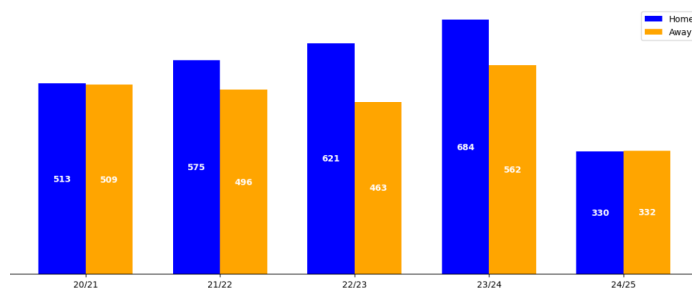


Figure 5.3: Goals Distribution Over Time.

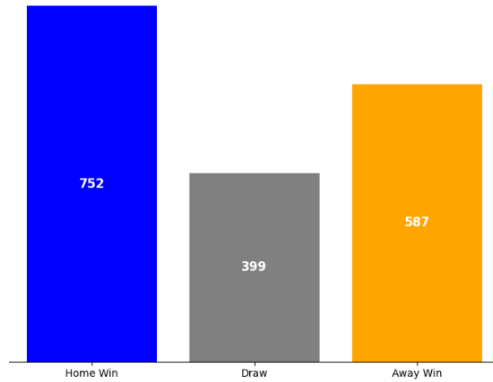


Figure 5.1: Game Distribution.

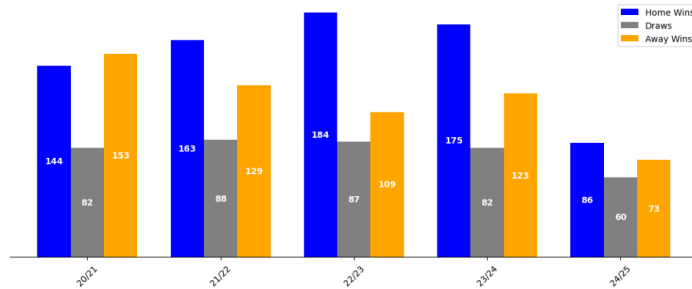


Figure 5.2: Game Distribution Over Time.

Table 5.1: Shots on Target and Possession Trends by Season

Season	Shots on Target		Possession	
	Home Avg	Away Avg	Home %	Away %
20/21	4.3	3.9	51.5%	48.5%
21/22	4.4	3.9	50.6%	49.4%
22/23	4.6	3.7	50.5%	49.5%
23/24	5.1	4.2	51.6%	48.4%
24/25	4.8	4.2	50.5%	49.5%

2 Baseline Model Results

The replicated Random Forest baseline model achieved an overall accuracy of 0.48 on the test set. Class-wise, the model performed best in predicting class 2 (*home win*), with a precision of 0.53, recall of 0.66, and F1-score of 0.59 as seen in table [5.2]. Performance was weaker for class 1 (*draw*), where the model showed the lowest recall (0.07) and F1-score (0.10). The macro-averaged F1-score was 0.40, and the weighted average F1-score was 0.43 which reflects a moderate imbalance in class-wise prediction quality.

Table 5.2: Classification Report on Test Data for Baseline Random Forest Model

Class	Precision	Recall	F1-Score	Support
Away Win	0.46	0.57	0.51	110
Draw	0.23	0.07	0.10	91
Home Win	0.53	0.66	0.59	147
Accuracy			0.48	348
Macro Avg	0.40	0.43	0.40	348
Weighted Avg	0.43	0.48	0.43	348

2.1 Cross-Validation

To evaluate the *baseline model's robustness* over time, we applied time series cross-validation using 15 consecutive folds. The average macro F1-score across folds was 0.4242 (± 0.0359). *Class-wise* performance was highest for the *home win* class with an average F1-score of 0.6331, followed by *away win* (0.5390). The model *consistently struggled* to predict the *draw* class, yielding an average F1-score of just 0.1178. Figure 5.4 visualizes the macro F1-score distribution with error bars representing one standard deviation across the 15 folds. Performance remained relatively stable, especially for the dominant classes.

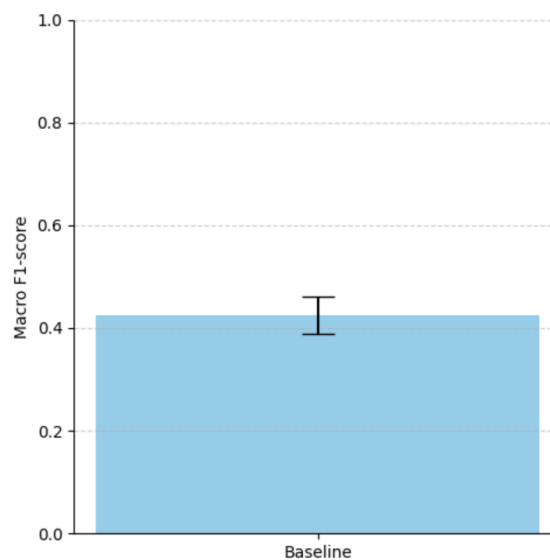


Figure 5.4: Macro F1 performance of the Baseline Random Forest model across 15 *TimeSeriesSplit* cross-validation folds. The bar height indicates the mean F1-score, and the error bar represents one standard deviation. The model achieved a mean macro F1-score of 0.4242 with a standard deviation of 0.0359.

2.2 Feature Importance with SHAP

To better understand which features most influenced the Random Forest baseline model's predictions, SHAP (Shapley Additive Explanations) values were computed [5.5]. The analysis revealed that `Att_Dif` (attack rating differential), `CKPP` (corner differential), and `Ovr_Dif` (overall rating differential) had the highest average impact on model output across all classes. These were followed by defensive-related features such as `Def_Dif` and `GD_Differential`. While many features contributed to predictions across all three classes, `Att_Dif` in particular showed strong influence across *home win*, *away win*, and *draw* predictions, underlining its overall importance in the model.

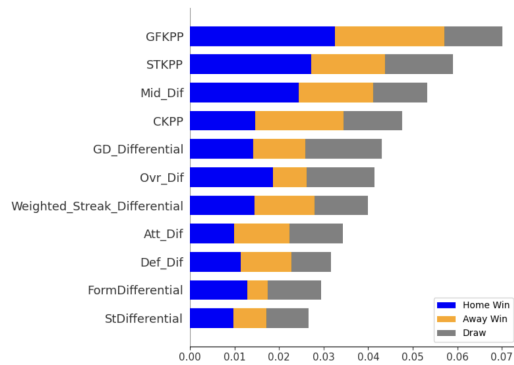


Figure 5.5: Mean SHAP value, average impact on model output margin

3 Results of Our Models with Engineered Features

3.1 XGBoost

The XGBoost model achieved an overall test set accuracy of 0.53, outperforming the Random Forest baseline [5.3]. The model performed best on the *home win* class, reaching a precision of 0.61, recall of 0.71, and F1-score of 0.65. The *away win* class also showed solid performance with an F1-score of 0.56. However, similar to the baseline, the model struggled to correctly predict the *draw* class, yielding a low recall of 0.09 and an F1-score of just 0.13. The macro-averaged F1-score was 0.45, and the weighted average F1-score was 0.49.

Table 5.3: Classification report for XGBoost model

Class	Precision	Recall	F1-Score	Support
Away Win	0.49	0.65	0.56	110
Draw	0.24	0.09	0.13	91
Home Win	0.61	0.71	0.65	147
Accuracy			0.53	348
Macro Avg	0.45	0.48	0.45	348
Weighted Avg	0.48	0.53	0.49	348

Cross-Validation

The model achieved a macro-averaged F1-score of 0.4333 (± 0.0389) across the 15 time series folds. *Class-wise F1-score* trends remained relatively stable, with the *home win* class consistently achieving the highest F1 values (peaking at 0.72), followed by the *away win* class. The *draw* class again remained the most difficult to predict, with F1-scores generally ranging between 0.00 and 0.34 across folds. These results reflect a similar pattern to the test set evaluation, highlighting the model’s strength in predicting wins while continuing to struggle with draws.

3.2 Multinomial Logistic Regression

The Logistic Regression model achieved an overall test accuracy of 0.49. The macro and weighted average F1-scores were both 0.44 & 0.48 respectively [5.4]. The model performed best on the *home win* class, with a precision of 0.61 and an F1-score of 0.62. Performance on the *away win* class was moderate (F1-score of 0.52). Even here the *draw* class was by far the hardest class to predict with an F1 Score of 0.18.

Table 5.4: Classification report for Logistic Regression model

Class	Precision	Recall	F1-Score	Support
Away Win	0.48	0.58	0.52	110
Draw	0.23	0.15	0.18	91
Home Win	0.61	0.63	0.62	147
Accuracy			0.49	348
Macro Avg	0.44	0.46	0.44	348
Weighted Avg	0.47	0.49	0.48	348

Cross-Validation

The model achieved a macro-averaged F1-score of 0.4321 (± 0.0541) across the 15 time series folds. *Class-wise*, the model performed best on the *home win* class, which achieved an average F1-score of 0.6211. The *away win* class followed

with a mean F1-score of 0.5071, while the *draw* class again proved the most difficult, averaging an F1-score of just 0.1701. Performance trends across folds remained relatively consistent, showing that the logistic regression model offers a stable but limited predictive capability, with ongoing challenges in predicting draws.

3.3 Random Forest

The final Random Forest model achieved the strongest performance among all tested models (See table 5.5). On the test set we got an overall accuracy of 54%, a macro-average F1-score of 0.50, and a weighted average F1-score of 0.53. The *home win* class remained the most accurately predicted, achieving an F1-score of 0.65, while the *away win* class followed closely with 0.56. Notably, the model also showed improved capability in identifying the *draw* class, which had an F1-score of 0.29, which is higher than in any previous model. These results suggest that this tuned Random Forest configuration provides the most balanced and effective performance across all outcome categories.

Table 5.5: Classification report for Random Forest Model

Class	Precision	Recall	F1-Score	Support
Away Win	0.55	0.58	0.56	110
Draw	0.35	0.25	0.29	91
Home Win	0.61	0.69	0.65	147
Accuracy			0.54	348
Macro Avg	0.50	0.51	0.50	348
Weighted Avg	0.52	0.54	0.53	348

The confusion matrix for the test set (see figure 5.6) supports these findings. The majority of *home win* instances (101 out of 147) were correctly classified, and the model also identified 64 out of 110 *away win* outcomes correctly. *Draws* were more frequently misclassified, with only 23 out of 91 correctly predicted. Among the errors, a slightly larger number were mistaken for a *home win* (40 instances) compared to an *away win* (28 instances). This reinforces the trend that *home wins* are the easiest to predict, whereas *draws* remain the most difficult class to distinguish for the model.

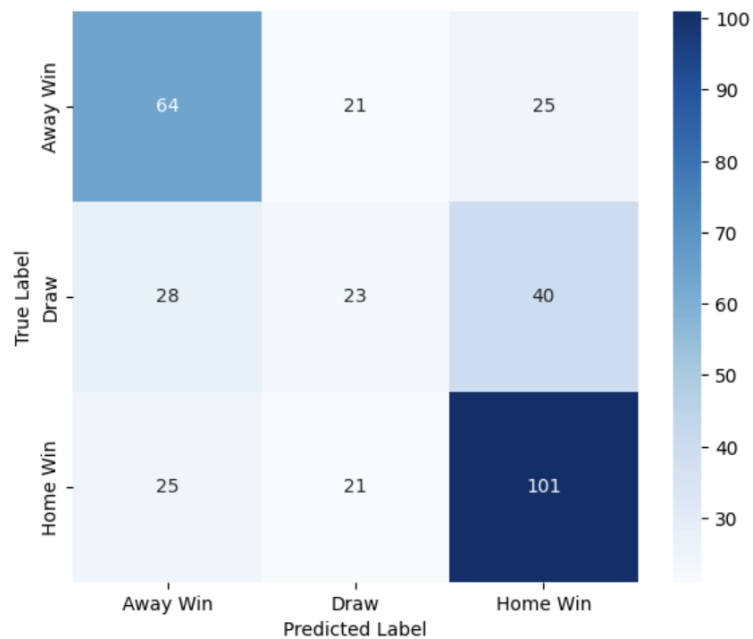


Figure 5.6: Confusion Matrix showing the predicted label vs the true labels on test set

Cross-Validation

The model achieved an average macro F1-score of 0.4411 (standard deviation = 0.0453) across 15 *TimeSeriesSplit* folds. Class-wise, the *home win* outcome consistently yielded the highest F1-scores, ranging from 0.5116 to 0.7091. The *away win* class followed, with scores between 0.3514 and 0.6111. As expected, the *draw* class remained the most difficult to classify, with F1-scores ranging from 0.1026 to 0.3830.

Despite fluctuations in early folds, the model stabilized in later iterations, suggesting improved predictive consistency as more training data became available. Figure 5.7 shows macro F1 performance with error bars indicating the standard deviation. These results indicate that the final Random Forest model performs reliably across temporally split data, especially in predicting wins, while draws continue to pose a significant classification challenge.

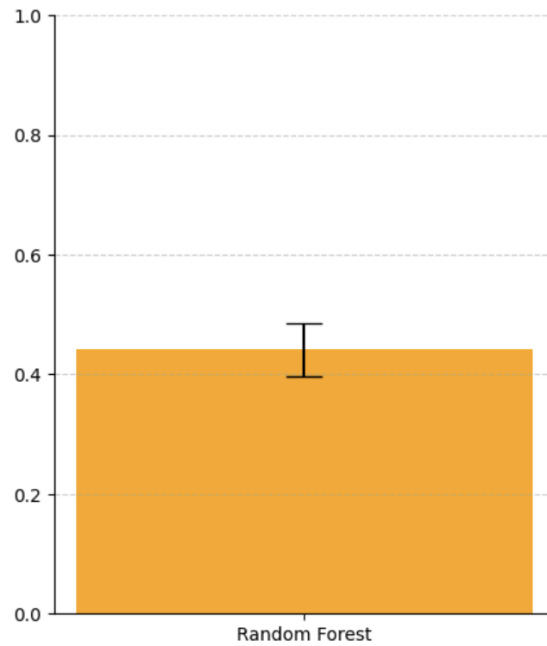


Figure 5.7: Macro F1 performance of the Random Forest model across 15 *Time-SeriesSplit* cross-validation folds. The bar height represents the mean F1-score, while the error bar indicates one standard deviation. The model achieved a mean macro F1-score of 0.4411 with a standard deviation of 0.0453.

Feature Importance

To interpret the final Random Forest model, both SHAP (Shapley Additive Explanations) values and built-in feature importance scores were analyzed. The SHAP summary plot reveals that `Elo_GD_Diff` (Elo-based goal difference differential) had the largest impact on the model’s predictions across all classes, especially for *home win* and *away win* outcomes. This feature was followed closely by `Adj_CPA_Diff` (adjusted crosses into penalty area), and `Adj_Att_Pen_Diff` (adjusted attacking penalty area actions), which also contributed meaningfully across all outcome types. Other influential features included such as `Adj_xGA_Diff`, `Adj_PPA_Diff`, and `FormDifferential` (Figure [5.8]).

The Random Forest’s internal feature importance rankings (Figure [5.9]) largely support the SHAP findings, with `Elo_GD_Diff`, `Adj_CPA_Diff`, and `FormDifferential` ranking as the top three most important predictors. Notably, engineered features like `GD_Differential`, `Adj_PassLive_Diff`, and `Adj_Points_Diff` also ranked highly, reinforcing their predictive relevance. In contrast, matchup cluster features contributed minimally to the model’s decisions, as indicated by both SHAP and the Random Forest importance scores.

Overall, the results indicate that adjusted performance metrics and relative team strength indicators, particularly those involving goal difference and recent

form, are the most critical for predicting match outcomes in this setting.

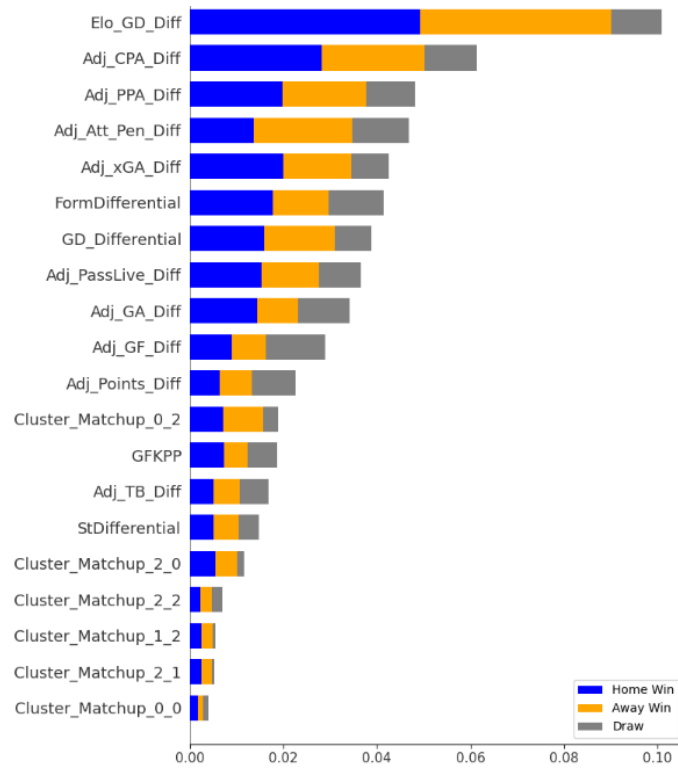


Figure 5.8: Mean SHAP value, average impact on model output magnit

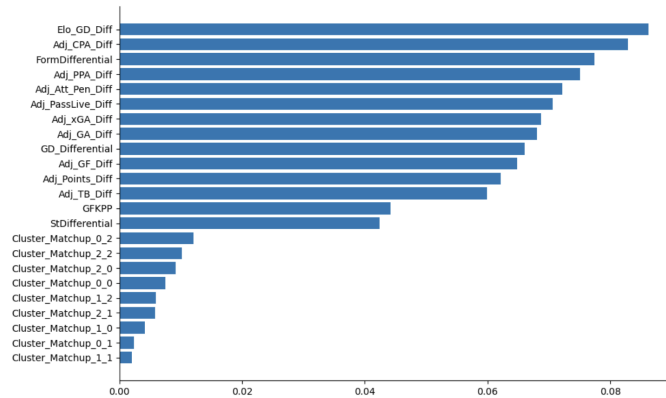


Figure 5.9: Random Forest built in Feature Importance Result

4 Comparison to Baseline Model

To evaluate the performance improvement of the final Random Forest model over the baseline Macro F1 was used as evaluation metric and compared across cross-

validation folds, accompanied by visual and statistical analysis.

Figure [5.10] shows the distribution of Macro F1-scores for both the baseline and final model. The Random Forest model achieved a higher mean F1-score (0.441) than the baseline (0.424). While the difference is modest, it reflects an improved ability to balance predictions across classes. The final model also showed a slightly higher standard deviation (0.045 vs. 0.036), indicating increased variability. A paired t-test comparing the F1-scores yielded a p-value of 0.19, suggesting that the observed improvement is not statistically significant.

To capture both the overall performance and its variability across models, Figure 5.11 displays a bar plot of the mean Macro F1-scores and their 85% confidence intervals. The plot illustrates a visible gain in the final model, indicating improved fairness and balance across outcomes.

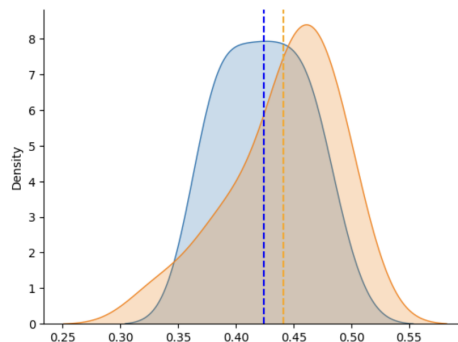


Figure 5.10: Distribution of Macro F1-score across cross-validation folds for Baseline (blue) and Final Model (orange). The Final Model shows a higher mean F1-score (0.441) compared to the Baseline (0.424), with slightly more variance.

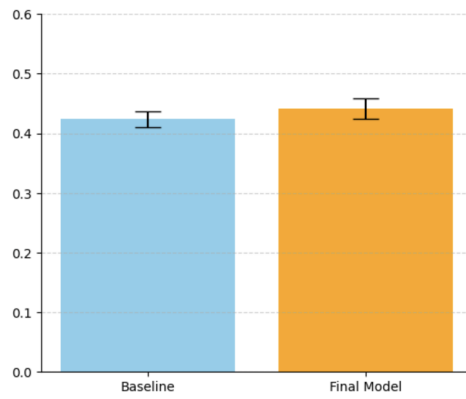


Figure 5.11: Mean Macro F1-score with 85% confidence intervals. Shows that the Final Model achieves a slightly higher mean Macro F1-score compared to the Baseline. While the confidence intervals are relatively tight and the visual trend favors the Final Model, the overlap between intervals and statistical testing suggest the difference is not significant. Nonetheless, the result may indicate a modest improvement in balanced performance across classes.

Chapter 6

Discussion

1 Discussion of Results

Our results show a shift in performance characteristics compared to both our replicated baseline and prior studies, particularly in addressing the known difficulty of predicting draw as an outcome. As shown in Figure 5.5, our final Random Forest model achieved a macro F1-score of 0.50, with individual class F1-scores of 0.65 (home win), 0.56 (away win), and 0.29 (draw). These metrics outperform the earlier replicated version of Baboota & Kaur’s [1] model, which on our data achieved a macro F1-score of 0.40, particularly underperforming on draws (F1-score of 0.10). Their original study, conducted on older data, achieved better balance with F1-scores of 0.68 (home win), 0.55 (away win), and 0.28 (draw), but their model lacked solutions to handle class imbalance.

Our solution for handling the class imbalance consisted of two targeted improvements: using `class_weight = 'balanced'` and applying a custom probability threshold to promote draw classification when the predicted draw probability exceeded 0.33. These changes improved the model’s ability to correctly identify drawn outcomes, historically the weakest-performing class in the literature. This result is consistent with the direction highlighted in prior studies such as Tax [8] and Jäderberg & Linde [2], which emphasized that draws were especially challenging and needed either innovative features or calibration techniques to be modeled effectively.

Compared to many historical approaches that rely solely on raw historical outcomes or betting odds, our model integrates a rich, dynamic set of features. These include form-based statistics from the work of Baboota & Kaur [1] such as `FormDifferential` and `StDifferential`, as well as our context-aware and tactically meaningful metrics such as `Elo_GD_Diff`, `Adj_CPA_Diff`, and `Adj_PPA_Diff`. This combination enables our model to move beyond traditional statistical features, aligning more closely with tactical match dynamics, a limitation frequently highlighted in previous work and introduces more context-aware features through targeted feature engineering.

Jäderberg & Linde [2] used FIFA ratings such as attack, midfield, and defense

as key features in their model. These ratings are static and based on subjective assessments from a video game. In contrast, our Elo-based rating is dynamic and updates after each match, taking into account both the result and the relative strength of the opponent. This makes our approach more grounded in real performance and better suited to capturing team form and context.

Our improvements aimed to solve concrete gaps identified in related work: namely, handling class imbalance, increasing draw prediction capability, and integrating richer, game-aware features. These refinements contribute to a more interpretable and practical prediction model for football outcomes.

The result of our 15-fold `TimeSeriesSplit` cross-validation showed an average macro F1-score of 0.441 [5.10] for the final model, compared to 0.424 for the baseline. The standard deviations were 0.0453 in the final model, indicating modest variability across folds. This suggests that while the model performs fairly consistently, its F1-score improvement over the baseline was not statistically significant ($p = 0.19$).

However, the limited size of our dataset in the earlier cross-validation folds clearly affected performance. In particular, folds with smaller training sets underperformed. A reason for this can be the fact that some of our most important features, such as the Elo rating, require sufficient match history to adapt and provide meaningful information. As more data became available in later folds, the model's performance improved.

One important factor in this pattern is the behavior of dynamic features like Elo ratings. Since Elo scores are initially set to a default value (1500) and are updated only after each match result, they require enough games to meaningfully differentiate between teams. In early folds with limited match history, Elo values remain too close to the baseline to offer strong predictive power. This could mean the model lacks access to well-adapted team strength signals early on, which weakens its ability to separate match outcomes, especially in tightly matched or less predictable scenarios like draws. Over time, as Elo values adapt to match performance, they become more informative, contributing to the improved performance observed in the later folds.

This trend highlights a common challenge in time series-based sports prediction: early-season data is sparse and lacks sufficient signal for reliable classification, particularly for rare outcomes. It also highlights the importance of historical depth when using adaptive features and suggests that future work could benefit from pre-training Elo or similar metrics using data from prior seasons or leagues.

To better understand the decision-making process of our models, we analyzed feature importance scores from both the baseline [5.5] and [5.9] this was done using both SHAP and the built in Random Forest feature importance. This comparison highlights clear differences in the influence of various features, providing insights into how model structure and feature engineering affected predictive performance.

In the baseline model, the most influential features were primarily static attributes such as `Att_Dif`, `Def_Dif`, `Mid_Dif`, and `Ovr_Dif`, all derived from FIFA

ratings. These features had a relatively balanced contribution across all classes but lacked deeper tactical or contextual grounding. While these features contributed to a decent overall accuracy, the baseline struggled significantly in identifying draws, as reflected in both cross-validation performance and the relatively shallow draw contribution in the SHAP distributions.

In contrast, the SHAP analysis of our final model shows that features like `Elo_GD_Diff`, `Adj_CPA_Diff`, and `Adj_Att_Pen_Diff` dominate the importance ranking. These features are not only more dynamic and context-aware but also demonstrate clearer class-specific contributions, especially in helping separate home wins from away wins and improving draw sensitivity. For instance, Elo-based metrics, which adapt after every match, allow the model to respond to recent performance trends and matchup-specific dynamics capabilities that the static FIFA-based features in the baseline simply cannot match.

The differences between the two SHAP plots underscore the value of our feature engineering when combined with subjective and static team descriptors. In conclusion, the SHAP comparison illustrates that predictive quality is driven by the quality and nature of features.

2 Societal Aspects

Today, many football fans may use AI tools to guess the results of matches before they happen. This can change the way fans feel about the game. For example, if a model thinks their team will probably lose, some fans might feel less excited or more nervous before the match even starts. Instead of just relying on gut feeling or team spirit, more fans now use stats in their discussions. This can be more fun instead of just relying upon a trained AI model to predict outcomes, which takes away the fun in football. In some cases, this can even affect betting and fantasy football decisions.

Chapter 7

Conclusion

1 Achievements

For this study, the main question was how accurately machine learning models can predict football outcomes with feature engineering. To test this, we used data from 1,738 Premier League matches over four and a half seasons. The results we obtained showed improved performance in key areas, achieving a macro F1-score of 0.50 and a weighted F1-score of 0.53, representing a meaningful improvement in class balance compared to the baseline, which reached a macro F1 of 0.40 and a weighted F1-score of 0.43. Our final model demonstrated improvement over the baseline in class balance, where the highest gain was in draw prediction, with an F1-score of 0.29 compared to just 0.10. This shows that our feature engineering strategies contributed to a more balanced performance across all outcomes, particularly improving draw prediction.

A major contributor to this improvement was the construction of our dynamic and context-aware features. For example, our Elo ratings, unlike traditional static team ratings, were adapted after each match and further enhanced by integrating goal differential. We extended this logic by adjusting many raw statistical features relative to the opponent's Elo rating. This allowed our model to better interpret a team's performance in its true context rather than in isolation.

Our use of unsupervised clustering to categorize teams by their play style also introduced an innovative tactical layer to our analysis. By incorporating this as cluster matchup variables, our model could capture how different play styles perform against each other, making the model more tactically aware.

Finally, one of the most impactful improvements came from our targeted probability threshold tuning for draw predictions. Combined with the use of adjusting the class weights, this directly addressed the severe class imbalance problem. By shifting the model's sensitivity to recognize a draw when its probability exceeded a certain threshold, we were able to improve recall and overall F1-score for this class without harming the model's macro F1-score. This trade-off is often justified in real-world applications: improving draw prediction can be highly valuable depending on the context, especially in betting, simulation, or tactical planning.

2 Recommendations for Future Work

For future work, we recommend using more data, not necessarily from older seasons, but from different leagues within the same time period. This could improve the accuracy of the Elo rating by providing a wider range of match outcomes and team performances. However, adding data from other leagues may require some adjustments for differences in play style and competitiveness between leagues.

Additionally, if it is possible to collect heatmaps or more detailed tactical data, that could help improve the playstyle feature. Instead of only classifying teams as offensive, neutral, or defensive, we could identify more specific strategies, such as counter-attacking, possession-based, or “parking the bus.” This would give a clearer picture of how each team actually plays and make the model more tactically aware. Such improvements could enhance not only predictive performance but also the interpretability of the model by explaining how and why specific outcomes are likely. However, accessing high-quality tactical data may be a challenge, and future work may need to explore partnerships or alternative data sources.

Another area for development lies in further improving the Elo-based feature, `Elo_GD_Diff`. Currently, this feature incorporates goal differential into the Elo update but still relies on a single update factor (K-value) and a uniform initial score of 1500 for all teams. Future work could explore tuning K-values based on team performance volatility or competition type, or even making K dynamic across the season. Also, consider pre-training Elo ratings using historical data or external rankings, which could offer more realistic starting values. Incorporating additional performance factors such as expected goals (xG) into the Elo update mechanism could also result in an even more effective rating system.

Another promising direction for future improvement lies in improving the model’s ability to predict draws. While our approach used a combination of adjusting the class weights to account for class imbalance by assigning higher importance to less frequent outcomes, such as draws and a custom probability threshold to encourage draw predictions, there are several alternative techniques that could potentially yield better results.

References

- [1] R. Baboota and H. Kaur, "Predictive analysis and modelling football results using machine learning approach for english premier league," *International Journal of Forecasting*, 2018. DOI: 10.1016/j.ijforecast.2018.01.003.
- [2] A. Jäderberg and P. Linde, *Enhancing football match outcome prediction: A comparative study of feature selection techniques and support vector machines*, KTH Royal Institute of Technology, Bachelor's thesis, 2024.
- [3] H. Bergqvist, *Is it possible to beat the bookie - machine learning and football*, Linnéuniversitetet, Bachelor's thesis, 2020.
- [4] G. Grbec, N. Bašić, and M. Tkalčić, "Ranking footballers with multilevel modeling," in *CEUR Workshop Proceedings*, 2024.
- [5] A. Pajankar and A. Joshi, *Hands-on Machine Learning with Python: Implement Neural Network Solutions with Scikit-learn and PyTorch*. Springer, 2022.
- [6] J. VanderPlas, *Python Data Science Handbook: Essential Tools for Working with Data*. O'Reilly Media, Inc., 2016.
- [7] A. C. Müller and S. Guido, *Introduction to Machine Learning with Python: A Guide for Data Scientists*. O'Reilly Media, Inc., 2016.
- [8] N. Tax and Y. Joustra, "Predicting the dutch football competition using public data: A machine learning approach," *Transactions on Knowledge and Data Engineering*, 2015. DOI: 10.13140/RG.2.1.1383.4729.
- [9] G. Dong and H. Liu, *Feature engineering for machine learning and data analytics*. CRC press, 2018.
- [10] FBref, *FBref - Football Statistics and History*, Accessed: 2025-03-04, 2025. [Online]. Available: <https://fbref.com/en/>.
- [11] Scikit Learn, *Scikit Learn Mutual Information*, Accessed: 2025-03-04, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.mutual_info_classif.html.
- [12] M. Kuhn and K. Johnson, *Applied Predictive Modeling*. Springer, 2013.
- [13] S. Klosterman, *Data Science Projects with Python*. Packt Publishing, 2021.

- [14] Scikit Learn, *Scikit Learn Lasso*, Accessed: 2025-03-04, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html.
- [15] Scikit Learn, *Scikit Learn GMM*, Accessed: 2025-03-04, 2025. [Online]. Available: <https://scikit-learn.org/stable/modules/mixture.html>.
- [16] Scikit Learn, *Scikit Learn Calinski-Harabasz score*, Accessed: 2025-03-04, 2025. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.calinski_harabasz_score.html.
- [17] Scikit Learn, *Scikit Learn Gaussian Mixture Model Selection*, Accessed: 2025-03-04, 2025. [Online]. Available: https://scikit-learn.org/stable/auto_examples/mixture/plot_gmm_selection.html.