Machine Learning Model Report

Group Name: `Group L`
Department of Computer Science
University College London
London, WC1E 6BT

December 17th, 2024

1 Introduction

1.1 Background

Our "Beat the Bookie" assignment is to create a football prediction model – a common objective with an abundance of prior research. The model will receive the names of two English Premier League football teams and any available team statistics (prior to the match). Using this extensive spread of information, the model will predict which team is more likely to win. The struggle with this goal comes from an uncountable number of minute factors that affect the result: the terrain, the weather, the current team spirit, etc. Bookmakers take all these various facets into consideration and efficiently provide betting odds for the game's winner, at an accuracy rate of 53%. Our goal is to extract the most important features and produce a model capable of overwhelming the average bookmaker's capacities, exceeding the previously mentioned accuracy rate.

In this paper, we will begin with a brief discussion of our strategies, followed by a description of our data and feature engineering phase. Next, we will thoroughly report our various models – feature selection versions and baseline versions. The paper will conclude with a thorough analysis of our experimentation and our final predictions on a selection of next year's EPL competitions.

1.2 Our Approach

Our approach revolves around the concept of **recency-based features** and is divided into two distinct parts: **Recency Feature Extraction** and **Rating Learning**. Inspired by the paper *"Incorporating Domain Knowledge in Machine Learning for Soccer Outcome Prediction"*, we sought to integrate the temporal dynamics of team performance into our models [3].

1.2.1 Recency Feature Extraction

For each feature group per team, we generate n features, where n represents the depth of the match time series from which the feature values are derived. For instance, considering the "Shots Taken" statistic, this approach results in features like *"average shots taken over the last 3 games"* through to *"average shots taken in the last game."* This methodology ensures that recent performance trends are systematically captured [3].

1.2.2 Rating Learning

The Rating Learning approach involves utilising established rating systems, including **Pi-Ratings**, **Generalised Attacking Performance (GAP) Ratings**, and **Elo Ratings**, to quantify team strength [1, 2, 4]. These ratings summarise historical performance and enable comparisons across teams, complementing the recency-based features by providing a more stable, long-term perspective on performance.

2 Data Transformation & Exploration

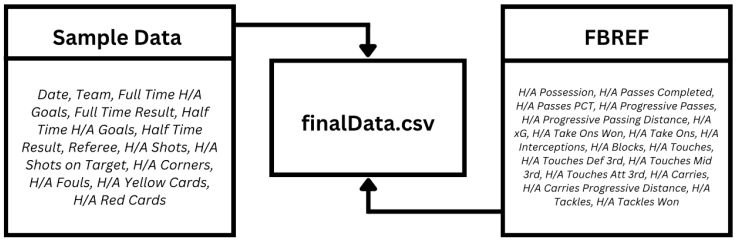
2.1 Data Import

To enrich our sammple dataset (below), we decided to scrape the web for more features. FBREF was ideal for this, as from 2017 onwards it has detailed match stats, and it also has index pages which contain links to each match in a given season. The scraper worked as follows. Each season index URL is built following this template:

\text{https://fbref.com/en/comps/9/{year}-{year+1}/schedule/{year}-{year+1}-Premier-League-Scores-and-Fixtures}

For each season index page, we extract all match urls, and then for each match, locate the stat table HTML element where we scrape data from, using html tags.

After getting the extra features, we needed to integrate them with the provided data. To do this, we first standardised names across the datasets- for example, we changed all instances of "Man United", to "Manchester United". After this, we then also standardised the date format to match the provided data. Finally we did an inner join on 'Date', 'Home Team', and 'Away Team', and dropped all columns before 2017, as the extra stats were not available before this time.



2.2 Feature Engineering

Our approach to feature engineering includes three main feature groups: Attacking Strength, Defensive Strength, and Midfield Strength. These feature groups were selected because they represent a team's ability to score goals, start and stop plays, and defend against the opposition's goals, respectively. We also opted to include performance metrics for each team as the Strength feature groups doesn't represent the outcome of the game. By including the following features: Goals Scored, Goals Conceded, and Match Outcome, we can consider the strength of the team corresponding to the outcome of the match.

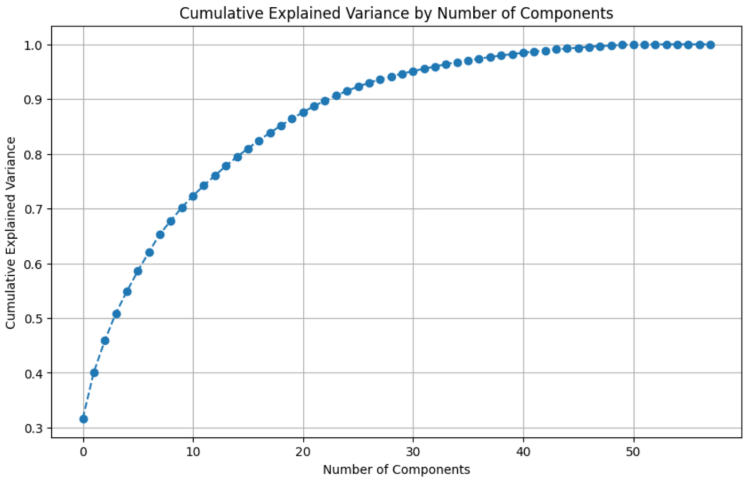
The features we have initially selected are all posterior features, **only observable after the match has concluded**. This limitation poses a challenge for our predictive task, as these outcomes cannot be known a priori. To address this problem, we leverage recency-based features, which serve as proxies for a team's performance in their current match. With the incorporation of average statistics from recent matches, we

approximate a prior distribution of team performance, enabling machine learning models to infer likely posterior outcomes.

To optimise the selection of features for predictive modelling, we employed two feature engineering techniques: Principal Component Analysis (PCA) and Chi-Squared Test. These techniques help identify the most significant features while reducing dimensionality and ensuring interpretability.

Principal Component Analysis (PCA): PCA transforms features into uncorrelated components to reduce the dataset's dimensions, which are ranked by the variance they capture [5]. Although this comes at the cost of accuracy, we are willing to trade some degree of accuracy for simplicity.

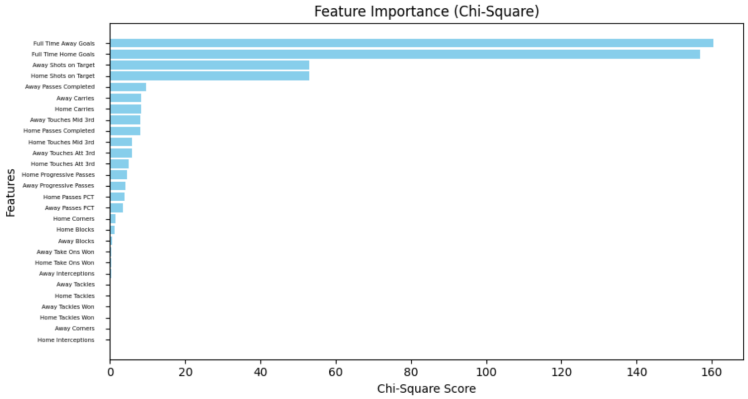
- The PCA transformation identified the number of components required to retain 95% of the dataset's variance to be 30.



With high-dimensional data, linear models overfit very easily. With PCA, we reduce the risk of overfitting while increasing computational efficiency. As the number of components to retain 95% of dataset's variance is identified to be 30, we will use 30 features to create the strength and performance feature groups.

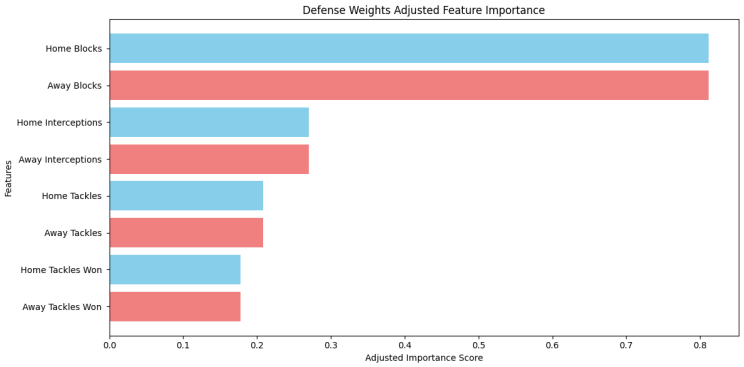
Chi-Squared Test: The Chi-Squared Test measures the independence between features and the label (Match Outcome) by comparing observed and expected frequencies in order to identify significant predictors [6].

- Input features were normalised to satisfy the non-negativity requirement of the Chi-Squared Test.



When constructing the Strength feature groups, we applied the Chi-Squared Test to determine the relative importance of features and used these weights to create a linear combination for each group. To ensure consistency, we assigned a baseline importance to each feature, calculated as one-third of the importance of the next higher-ranked feature:

$$\text{Baseline Importance} = \frac{\text{Importance of Next Feature}}{3}$$



By combining PCA for dimensionality reduction and the Chi-Squared Test for feature weighting, we balanced minimising dimensionality with maximising interpretability, resulting in features that more effectively represent the data for machine learning models.

2.3 Strength Features

The *Strength Features* are derived by combining various key metrics to quantify a team's performance across different aspects of the game:
Attacking Strength: A combination of *Shots on Target (Home/Away)*, *Progressive Passes (Home/Away)*, *Touches in the Attacking Third (Home/Away)*, *Take Ons Won (Home/Away)*, and *Corners (Home/Away)*.
Midfield Strength: A combination of *Touches in the Midfield Third (Home/Away)*, *Passes Completed (Home/Away)*, *Passes Percentage (Home/Away)*, and *Carries (Home/Away)*.
Defensive Strength: A combination of *Tackles and Tackles Won (Home/Away)*, *Blocks (Home/Away)*, and *Interceptions (Home/Away)*.

The aforementioned performance metrics **Goals Scored**, **Goals Conceded** already exist in the data and do not require further feature engineering. **Recent Performance** is calculated with the same rolling average strategy using the **Math Outcome** feature.

3 Methodology Overview

3.1 Recency Feature Extraction Model

The **Recency Feature Extraction Model** is a systematic approach designed to capture recent performance trends of teams through dynamic feature engineering. By focusing on time-sensitive rolling averages, the model ensures that temporal dynamics are included in the dataset, improving the representativeness of input features. Recency features reflect a rolling average of various performance metrics over the last x games for both home and away teams. This approach ensures that the model considers posterior statistics of previous data as average prior statistics for the current match. Below, we outline the key steps and components of the methodology used to implement this model.

Steps Taken:

- **Selection of Metrics:** We focused on key performance indicators engineered in feature engineering such as:

  - *Goals Scored:* Total goals scored by the team.
  - *Goals Conceded:* Total goals conceded by the team.
  - *Attack, Midfield, and Defense Strength:* Metrics assessing specific tactical areas.
  - *Recent Performance:* A numerical score based on match outcomes (win, draw, or loss).

- **Recency Parameters:** To incorporate a range of historical trends, we calculated rolling averages for the above metrics over varying lengths, $x = 1$ to $RECENCY\_NUM$. This provided features such as:
  - *Avg Goals Scored Home Last x Games*
  - *Avg Defense Strength Away Last x Games*

- **Rolling Statistics Calculation:**

$$\text{Rolling Average}_{M,x} = \frac{\sum_{i=n-x+1}^{n} M_i}{x}$$

  where **M** is the metric being calculated, x is the number of most recent matches considered, n is the index of the current match, and M_i is the value of metric **M** for match i. Note that the match data has to be in **chronological order** for the rolling average to consider the most recent games.
  - For each match, we aggregated historical data for the teams playing (both home and away).
  - A helper function calculated the rolling average of each metric over the last 1 to x games. The paper *"Incorporating Domain Knowledge in Machine Learning for Soccer Outcome Prediction"* which inspired this method found $n = 9$ to be the ideal depth [3]. However, we have found that $n = 5$ works better with our dataset via trial and error.
  - If a team had insufficient prior games to compute the average, the corresponding feature value was set to `NaN` and later removed during preprocessing.

- **Updating Team Stats:** After computing the rolling average statistics for the current match, the actual match statistics were added to the team's historical data. This ensured that subsequent matches included up-to-date information in their feature calculations.

- **Feature Space Expansion:** The iterative process generated 5 different features for each feature group from a *Rolling Average of Last 1 Game* to *Rolling Average of Last 5 Games* for each team dynamically. With this approach, a more complete understanding of recent performances of the team's were captured, allowing better predictive accuracy then taking a static window of 5 games.

- **Handling Missing Values:** Matches where insufficient historical data was available (first 5 games of each team in our dataset) were removed to maintain data quality. This meant that 3.62% of our dataset was lost due to recency calculations.

By incorporating these recency-based features, we effectively model the temporal dynamics of team performance, allowing the model to weigh the immediate past more heavily in its predictions. This methodology is particularly advantageous in dynamic environments like football, where recent trends often play a huge role in determining the outcome of a match.

3.2 Rating Learning

Now, we will discuss the three rating models we investigated.

3.2.1 ELO Rating Model

The ELO rating model dynamically updates team ratings based on goal difference (and consequentially, match outcomes), incorporating prior expectations with actual results. Each team's rating is initialised to 1500 (average club player rating goal determined by US Chess Federation for chess players) [7], and ratings are updated iteratively after every match [3].

Match Prediction For a given match, the probability of the home team (H) winning (γ_H) is calculated as:

$$\gamma_H = \frac{1}{1 + 10^{\frac{A_0 - H_0}{d}}}$$

where H_0 and A_0 are the pre-match ratings (prior probabilities) of the home and away teams and d the rating difference sensitivity. Here $d = 400$ is used because when the ELO system was designed by Arpad Elo, 400 was chosen as the rating difference corresponding to a 10x odds ratio in skill between two players [7]. The away team's probability of winning (γ_A) is complementary:

$$\gamma_A = 1 - \gamma_H$$

Result Encoding The outcome for the home team (α_H) is defined as:

$$\alpha_H = \begin{cases} 1 & \text{if home team wins,} \\ 0.5 & \text{if the match is a draw,} \\ 0 & \text{if the home team loses.} \end{cases}$$

The outcome for the away team (α_A) is complementary:

$$\alpha_A = 1 - \alpha_H$$

Rating Update A dynamic adjustment factor (k) is introduced, depending on the (|goal difference|):

$$k = 10 \cdot \sqrt{1 + |\text{goal difference}|}$$

Here k is calculated with a constant of 10, following the ELO_g rating system from [2]. The ratings for the home and away teams are updated as:

$$H_1 = H_0 + k \cdot (\alpha_H - \gamma_H)$$

$$A_1 = A_0 + k \cdot (\alpha_A - \gamma_A)$$

This update ensures that teams gain or lose points proportional to the discrepancy between expected and actual outcomes. Although [2] uses betting odds for the ELO calculation, we stick to using goal difference for team's ELO updates as per our coursework limitations.

Feature Integration The updated ELO ratings (H_1 and A_1) are used alongside additional features defined in **2.2**, such as average attacking and defensive strengths, recent performances, and midfield strengths, to predict match outcomes.

3.2.2 PI Rating Model

The PI rating model dynamically adjusts team ratings based on goal discrepancies observed in matches. It balances long-term performance and short-term fluctuations with the following parameters [3]:

- **Learning Rate (λ)**: Determines how much recent match results influence ratings. A higher λ assigns a higher weight to more recent performances.

- **Diminishing Function ($\psi(e)$)**: Reduces the impact of large goal differences on ratings to prevent unrealistic adjustments:

$$\psi(e) = c \cdot \log_b(1 + |e|)$$

where $e = |go - gp|$, $b = 10$, and $c = 5$.

- **Home/Away Learning Rate (γ)**: Adjusts how home performances influence away ratings and vice versa, helping to account for home advantage.

- **Form Factor Parameters (ϕ, μ, δ)**: Capture short-term fluctuations that don't reflect a team's long-term ability:
  - ϕ: Threshold for triggering performance adjustments.
  - μ: Magnitude of provisional rating changes.
  - δ: Rate at which provisional ratings revert to background ratings.

The PI rating algorithm depends on the value of ϕ. If under/over-performance exceeds ϕ, provisional ratings (pr) differ from background ratings (br) until the team's performance stabilises [4]. Initially, ratings are assumed to be zero.

Background Rating Calculation The team's background rating is the average of its home (br_τ^H) and away (br_τ^A) ratings:

$$br_\tau = \frac{br_\tau^H + br_\tau^A}{2}$$

Home (x) and away (y) ratings are updated dynamically for each match [3]:

$$br_x^{H_t} = br_x^{H_{t-1}} + \psi_x(e) \cdot \lambda$$

$$br_x^{A_t} = br_x^{A_{t-1}} + \gamma \cdot (br_x^{H_t} - br_x^{H_{t-1}})$$

$$br_y^{A_t} = br_y^{A_{t-1}} + \psi_y(e) \cdot \lambda$$

$$br_y^{H_t} = br_y^{H_{t-1}} + \gamma \cdot (br_y^{A_t} - br_y^{A_{t-1}})$$

Goal discrepancies ($go = go_x - go_y$) are compared with expected discrepancies ($gp = gp_x - gp_y$), where gp_x and gp_y are calculated using logistic scaling, which is inspired from [3] for expected goals to be bounded and non-linear:

$$gp_x = \frac{3}{1 + \exp(-br_x^{H_{t-1}})}, \quad gp_y = \frac{3}{1 + \exp(-br_y^{A_{t-1}})}$$

The error is then passed into the diminishing function $\psi(e)$ to reduce the impact of large errors, where $e = |go - gp|$. Using the previous background rating and the error, the rating of the team is updated.

Provisional Rating Calculation Provisional ratings adjust dynamically when the form factor is triggered:

$$pr = \begin{cases} br + \mu \cdot \delta(\phi - \phi_{cx}) & \text{(overperformance)} \\ br - \mu \cdot \delta(\phi - \phi_{cx}) & \text{(underperformance)} \end{cases}$$

These adjustments continue until ratings stabilise and converge to the background ratings. The importance of the addition of provisional ratings is evident from the November and December 2024 performance of Manchester City, showing that even the best of teams have periods of underperformance that should be accounted for and vica versa.

3.2.3 GAP Rating Model

The GAP rating model described in Edward Wheatcroft's research paper [1] compares the offensive and defensive abilities of each team in the league. The potential of each team is divided into 4 categories: **home attacking rating**, **away attacking rating**, **home defensive rating**, **away defensive rating**. Here, an attacking rating corresponds to 'expected attacking rating against an average team' while defensive rating corresponds to 'expected attacking rating of an opposing average team'. For clarity, a skilled team would have a high attacking rating and low defensive rating. While very similar to the previous ratings, the difference in locations, the ability to build these features from any available data, and the reduced constraints on rating values, all contribute to a more powerful system than PI rating was capable of. Starting at 0, ratings for the *ith* home team against the *jth* away team are updated with the following equations [1]:

$$H_i^a = \max(H_i^a + \lambda\phi_1(S_h - \frac{H_i^a + A_j^d}{2}, 0)$$

$$A_i^a = \max(A_i^a + \lambda(1 - \phi_1)(S_h - \frac{H_i^a + A_j^d}{2}, 0)$$

$$H_i^d = \max(H_i^d + \lambda\phi_1(S_a - \frac{H_i^d + A_j^a}{2}, 0)$$

$$A_i^d = \max(A_i^d + \lambda(1 - \phi_1)(S_a - \frac{H_i^d + A_j^a}{2}, 0)$$

H^a, A^a, H^d, A^d are the 4 respective ratings from above. Similar equations are given to update the *jth* away team against the *ith* home team – where ϕ_1 is replaced by ϕ_2. λ is a parameter (greater than 0) that influences the impact of the most recent match on the ratings; ϕ_1 and ϕ_2 (ranging [0, 1]) influence the impact of a home match on away ratings and an away match on home ratings.

Finally, S_h and S_a are the match logistics used to update our scores. Wheatcroft hypothesised that ranking teams based on the number of goals scored (as PI rating does) would be inaccurate since goals are fairly luck-based. Wheatcroft then empirically proved that using **'Shots on Target'** and **'Corners'** led to more accurate results [1]. Following this research, we incorporated S_h and S_a as the sum of these two match statistics.

Since the paper focuses on betting odds, Wheatcroft built a Logistic Regression model with the sum of these 8 new rating features and an average of specific bookmaker odds [1]. Our task does not allow betting features so we chose to create a model using a combination of rating scores and recency attributes (last 5 games only).

Next, we had to find the optimal λ, ϕ_1, ϕ_2 parameters. Following the paper, we used an ignorance score to compare results. However, we found it hard to use the **Nelder-Mead** method to optimise the parameters under constraints. Instead, we utilised a **Grid Search** (discussed further in Section 3.3) over 1000 combinations of parameters. The resulting parameters were used to create our final GAP rating model.

3.3 Grid Search for Hyperparameter Optimization

Hyperparameter optimisation is a critical step in improving the performance of machine learning models. It involves finding the best combination of hyperparameters that maximise the model's accuracy or minimise its loss on a given dataset.

Grid search is a systematic approach to this problem. A predefined set of hyperparameter values is used to construct a Cartesian product of all possible combinations. Mathematically, if there are n hyperparameters, and each hyperparameter h_i has m_i possible values, then the total number of combinations to evaluate is [8]:

$$N = \prod_{i=1}^{n} m_i$$

where N is the total number of configurations. Each configuration is evaluated by training the model and measuring its performance on a validation set.

For example, if tuning involves two hyperparameters, such as the learning rate (λ) and the number of layers (l), with respective values $\eta \in \{0.01, 0.1, 1\}$ and $l \in \{2, 3, 4\}$, the grid search evaluates all $3 \times 3 = 9$ combinations. Grid search is effective for small grids, however gets more computationally expensive as the number of hyperparameters and the range of values they can take increase.

4 Model Training & Validation

4.1 Training

4.1.1 Overview of Machine Learning Models

This section provides a brief explanation of the models used in the experiments.

Logistic Regression (LR):

Logistic Regression is a linear model commenly used for classification tasks. In our case, we are using Multinomical Logistic Regression as we have 3 types of dependent variable (Win, Draw, Loss). Logistic Regression predicts the probability of an outcome applying the logistic (sigmoid) function to a linear combination of input features [9]:

$$P(y = 1 \mid X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p)}}$$

Here, β_0 represents the intercept, and β_i are the coefficients for the input features X_i. Logistic Regression is going to be particularly effective in our case because the logistic function maps the weighted combination of our recency-based features to probabilities, ensuring predictions remain interpretable and grounded in recent team performance. This makes Logistic Regression a suitable choice for leveraging short-term trends in predicting football match outcomes.

Support Vector Machine (SVM):

Support Vector Machines (SVMs) are supervised learning models used to classify data by finding the hyperplane that best separates different classes. The goal is to maximize the margin, which is the distance between the hyperplane and the closest data points from each class, called support vectors [10].

When the data isn't linearly separable, SVMs use kernel functions to transform the data into a higher-dimensional space where a linear hyperplane can separate the classes. Common kernel functions include the linear kernel, polynomial kernel, and radial basis function (RBF) kernel. The RBF kernel is especially useful for capturing non-linear patterns by measuring the similarity between the points based on their distance in this transformed space [10].

SVMs are a good choice for football prediction using our recency features because they can handle complex relationships between features like recent attacking and defensive strengths or team performance. By using a kernel function, SVMs can capture non-linear patterns in how these recent trends influence match outcomes, potentially providing a better prediction compared to linear models.

k-Nearest Neighbors (KNN):

KNN is a non-parametric, instance-based learning algorithm. Instead of building a model during training, it classifies a data point based on the majority class among its k-nearest neighbors in the feature space [11]. The *closeness* between points is usually measured using a distance metric such as Euclidean distance, which calculates how far two points are from each other in the feature space.

While KNN works well for smaller datasets because of its simplicity, it can become slow and less practical as the dataset size grows, since it needs to calculate distances for every point in the dataset during prediction. However, as our dataset is quite small, and KNN having the ability to adapt to decision boundaries without assuming a specific model [11], it is a great tool to test for our purposes.

Long Short-Term Memory (LSTM):

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network (RNN) specifically designed to work with sequential data by learning patterns and relationships over time. Unlike traditional RNNs, LSTMs use special mechanisms called [12]: *input gate*, *forget gate*, and *output gate* to handle the flow of information. These gates allow the network to retain important information over long sequences and discard irrelevant details, which helps overcome issues like *vanishing or exploding gradients* which occur often in regular RNNs which lack the gate mechanism [12].

LSTMs are particularly useful for tasks involving sequences, such as natural language processing or time-series data, where understanding the order and timing of events is critical. For our football prediction task, LSTMs are well-suited because they can analyze recency features—like recent attacking and defensive strengths or team performance over the last few games—as sequential data. This helps the model capture trends and momentum, such as a team's improving form or declining performance, to make more accurate predictions about match outcomes.

Random Forest (RF):

Random Forest is an ensemble learning method that builds multiple decision trees and combines their predictions to improve the accuracy and stability of the overall model. Each tree is trained on a random subset of the data using bootstrapping, and at each individual decision tree, only a random subset of features is considered. This randomness helps make the individual trees diverse and reduces the risk of overfitting. For classification tasks, the final prediction is made by majority voting across the trees.

For our football prediction task, Random Forest is a strong choice because it can effectively analyse the various recency-based features, such as average defensive strengths or team performance over recent matches. By using multiple decision trees, it can capture non-linear relationships and interactions between these features, such as how an increase in

defensive strength might impact performance when paired with a drop in attacking strength, making random forest a valuable tool to explore.

4.2 Feature Selection

Feature engineering produced 60 distinct recency features. However, the PCA function previously calculated an optimal number of 30 components. Training a model on an excessive number of features can lead to overfitting: where the learning algorithm begins to perfectly match the training dataset and loses the ability to correctly predict nuanced test cases.

Our solution was to explore diverse feature selection methods [13]. For each of our 5 chosen machine learning models – Logistic Regression, KNN, SVM, LSTM, RF – we attempted model-specific supervised feature selection methods. More specifically, we chose wrapper methods (training several models on subsets of the features and comparing them), filter methods (analysing each feature's statistical relation with the model's target), and embedded methods (a combination of wrapper and filter methods that writes directly into the learning algorithm).

4.2.1 Logistic Regression

For the Logistic Regression models, we implemented one wrapper method (generally computationally inefficient as they compare several models) and the three most common embedded methods. We chose **Recursive Feature Elimination** [13] as it considers a variety of characteristics like feature relevance, redundancy, interactions, and more. In this RFE procedure, we begin with a full logistic regression model and remove a single feature at a time, using the aforementioned attributes. Then, we train a new logistic regression model, once again isolate the least important feature, and repeat the method until satisfied. Figure 1 below reveals the process in which RFE isolated the optimal number of components, features ranked 1 were subsequently used to train the **LR with RFE** model.

Moving on to the embedded feature selection methods, we chose to experiment with LASSO regression (also known as l1 regularisation), ridge regression (l2 regularisation), and Elastic Net regression (l1 & l2 regularisation). The optimisation problem for **LASSO regression** is given below:

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)})^2 \text{ subject to: } ||\mathbf{w}||_1 \leq t$$

The equation describes a simple Ordinary Least Squares optimisation problem with a constraint on the l1 norm of $\mathbf{w}$ weights, for some threshold t. The constraint itself is the regularisation technique. LASSO enforces a penalty such that the weights of certain features (loosely associated with the target data) are driven down to 0, essentially removing the redundant features from the **l1 regularisation** model and thereby performing feature selection. The new feature set can be found on the Jupyter Notebook.

Next is the **l2 regularisation** model (Ridge Regression), formula given below:

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)})^2 \text{ subject to: } ||\mathbf{w}||_2^2 \leq t$$

Within this particular equation, a constraint is applied on the l2 norm of $\mathbf{w}$ weights, for some threshold t. By changing this constraint, the model reduces complexity (and therefore overfitting tendencies), encourages more evenly distributed weights, and forces weights of redundant features down to near-zero values.

The optimisation problem for Elastic Net regression is given below:

$$\underset{w}{\operatorname{argmin}} \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \mathbf{w}\cdot\mathbf{x}^{(i)})^2 + \frac{\lambda}{2}(\alpha||\mathbf{w}||_2^2 + (1-\alpha)||\mathbf{w}||_1)$$

This equation is a mixture of the l1 and l2 regularisation techniques above. Two additional variables are introduced: λ defines the severity of penalties on the weights while α parameterises the level of blend between regularisation techniques. By experimenting with the differing variables, we reached an optimal mixture of embedded techniques (displayed within the Jupyter Notebook) and created the **l1 & l2 regularisation** model.

4.2.2 k-Nearest Neighbours

KNN models do not inherently support RFE or embedded selection methods, as non-parametric, instance-based models do not calculate feature importance or contain weights to penalise. There are ways to overcome these limitations but we chose to simply implement two filter methods: Variance Thresholding, followed by Mutual Information. **Variance Thresholding** [13] identifies features with low variance – redundant features that remain constant over a large dataset. As KNN predictions are based on the distance between data points, the model is highly sensitive to these redundant features, found in figure 2 below. Next, **Mutual Information** [13] compares the entropies of the data and the target, locating irrelevant features. Figure 3 below shows the final subset of features used to train the feature selection version of KNN.

4.2.3 Support Vector Machine

For the SVM model, we chose to follow an approach similar to the LR model. First, we applied the **RFE** technique with an SVM model, results found in figure 4 below. For the next models, we also explored embedded algorithms by slightly modifying the objective function to incorporate penalties: **l1 regularisation**, to encourage sparsity in the weights and **l2 regularisation** to discourage large coefficients without sparsity. We briefly attempted Elastic Net regression, a combination of the l1 and l2 norm, but saw no differences between the models and removed it early on.

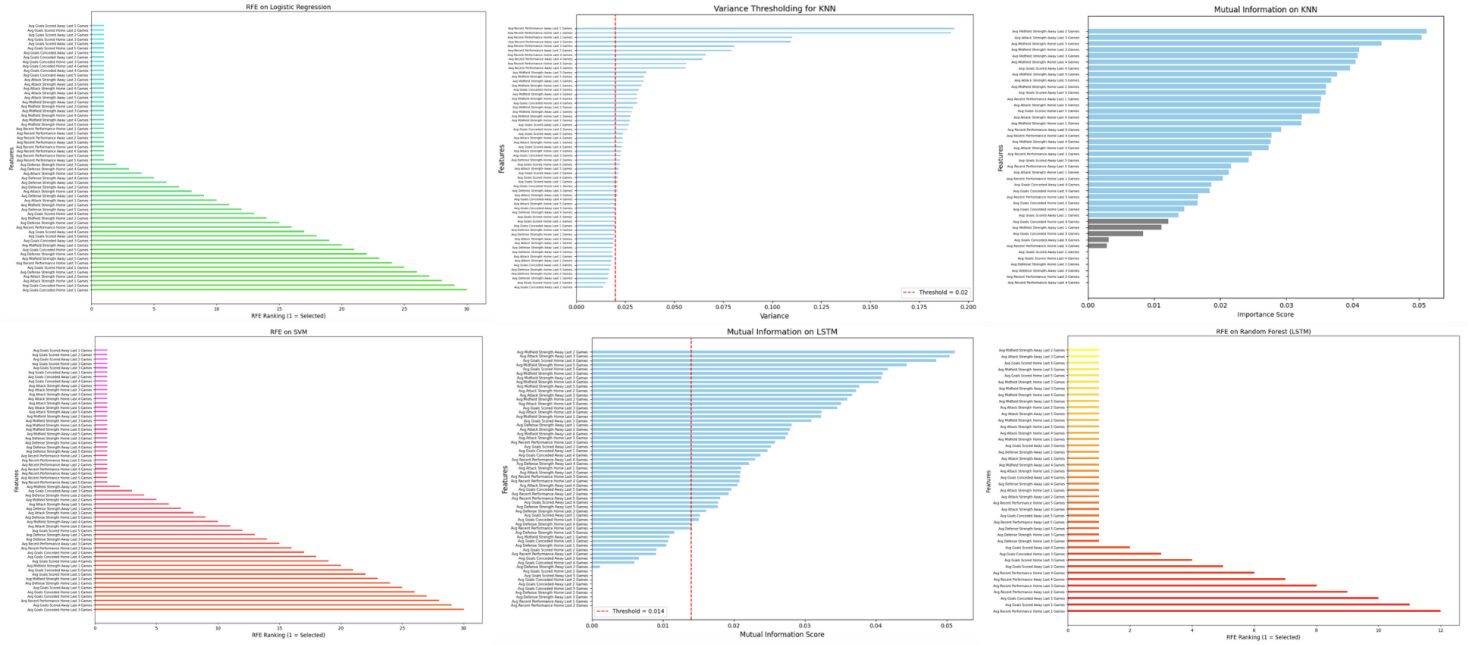
4.2.4 Long Short Term Memory

As LSTMs are sensitive to noise, we first removed irrelevant features (see figure 5 below) using **Mutual Information**. Next, we performed **RFE** with a tree-based model to rank the remaining features. Tree-based models excel at identifying patterns, making them perfect for temporal LSTMs. The final subset of features can be found in figure 6 below.

Finally, embedded feature selection is also compatible with LSTM models. On top of analysing **l1**, **l2**, and **l1 & l2 regularised** models (with similar constraints on weights as in logistic regression), we also decided to test a new embedded technique: **dropout regularisation** [14]. The equation is described below:

$$h_i' = r_i \cdot h_i$$

Only applicable to neural networks, this equation describes a new layer of neurons placed after each original layer in the model. Each new neuron h_i' is calculated by multiplying r_i by the previous layer's neuron h_i. r_i is a binary value calculated using a Bernoulli random variable. By choosing the optimal hyper-parameter for the random variable, we randomly 'drop



Feature Selection Figure. This figure contains all the graphs for feature selection, arranged in order from left to right and then top to bottom: (1) RFE on Logistic Regression, (2) Variance Thresholding for KNN, (3) Mutual Information for KNN, (4) RFE on SVM, (5) Mutual Information on LSTM, (6) RFE for LSTM.

out' sets of neurons. This effectively prevents overfitting by forcing the model to learn more features and avoid dependency on specific neurons (features).

4.2.5 Random Forest

For Random Forest feature selection, we implemented three complimentary filtering methods: ANOVA F-value, Variance Threshold, and Mutual information.

Same as with the previous models, **Variance Threshold** (threshold=0.01) removes features which have near-zero variance – as RF performs poorly when given constant features that give little discriminative power to split trees.

Anova F-value measures the linear relationships between features and the target by comparing the variance between various target classes to the variance within classes. Greater F-values indicate features that create clear separations between target classes, making them suitable for tree splits. **Mutual Information**, once again, encapsulates linear and non-linear relationships, which compliments F-test by finding complex patterns that RF can use in its decision trees. We normalize F-scores and Mutual Information scores into a 0-1 range, and combine them as follows to get a single metric:

$$\text{Combined Score} = \frac{\text{Normalized F-Score} + \text{Normalized MI-Score}}{2}$$

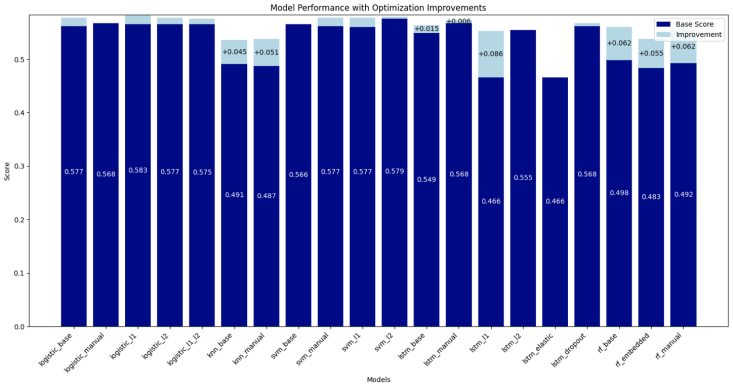
The selected features consist of features that pass the variance threshold, and have the highest 'combined score' values.

For embedded feature selection in RF, we used SelectFromModel. This utilises the model's built in feature importance scores to identify the best features, while considering the linear and non-linear relationships.

4.3 Validation

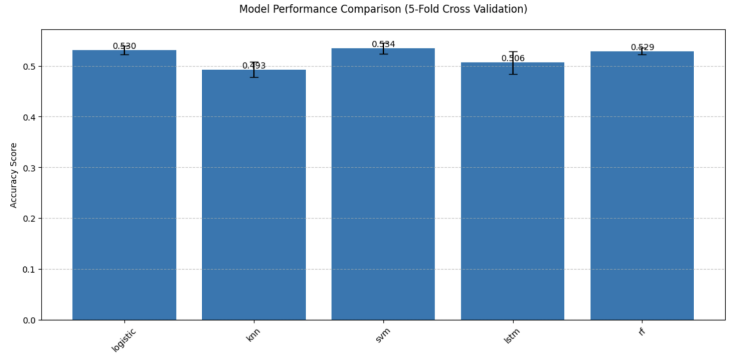
4.3.1 Recency Validation

We validated our recency models in two stages. First, we decided to select the best model of each family before model evaluation. In this process, we used 5-fold cross-validation in grid-search to find the best parameters for each model variant.



The strongest models for each family were as follows: Logistic Regression - Manual Feature Selection, KNN - Manual Feature Selection, SVM - L2 regression, LSTM - Manual Feature Selection, Random Forest - Base Model.

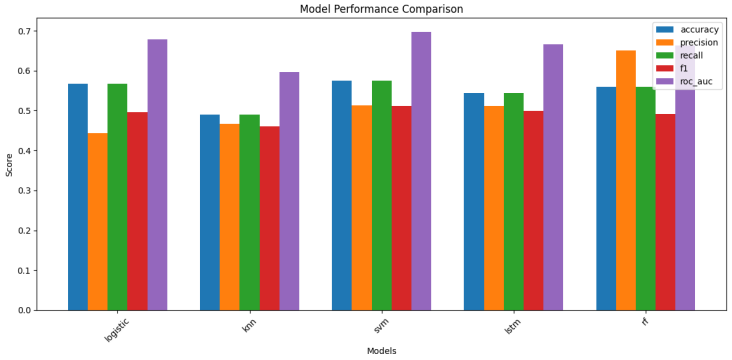
Secondly, the best model from each family underwent 5-fold cross-validaiton once more to guarantee robust performance assessment.



The results are shown above. The highest accuracy variance was 0.023, and the lowest was 0.007, giving stable results across folds for each family

5 Results

5.1 Recency Model Results

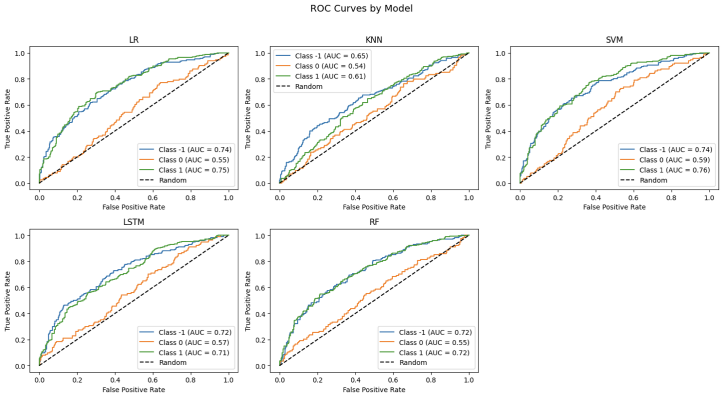


	Model	Accuracy	F1 Score	ROC AUC	Precision	Recall
0	logistic	0.568	0.495	0.678	0.444	0.568
1	knn	0.491	0.460	0.597	0.467	0.491
2	svm	0.575	0.511	0.697	0.513	0.575
3	lstm	0.543	0.499	0.665	0.511	0.543
4	rf	0.560	0.492	0.663	0.651	0.560

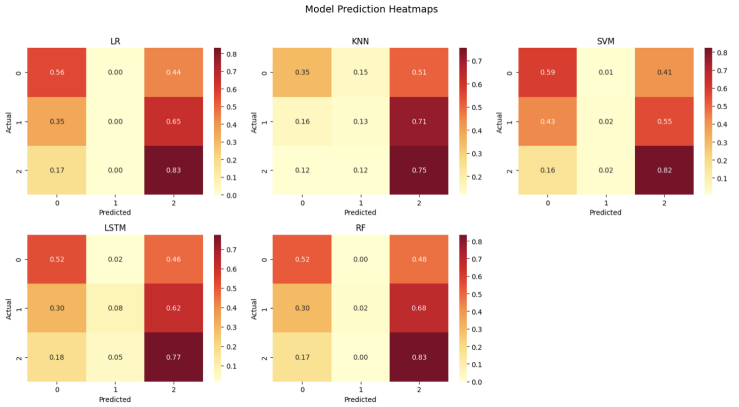
The comparative analysis of our 5 different recency models showed diversified levels of performance across different metrics. SVM proved to be the strongest overall performer, achieving the highest accuracy and ROC AUC score, demonstrating the best discriminative ability across classes. Moreover, SVM also showed the highest precision, indicating strong reliability in win predictions. Our LR model had both the second highest accuracy and ROC AUC, showing relatively strong prediction capabilities. LSTM performed moderately across all metrics. The KNN model had the highest F1 score, indicating a good balance between precision and recall, but performed weaker relative to other models in other metrics. Lastly, the Random Forest model performed moderately across all metrics, with its best metric being in ROC AUC.

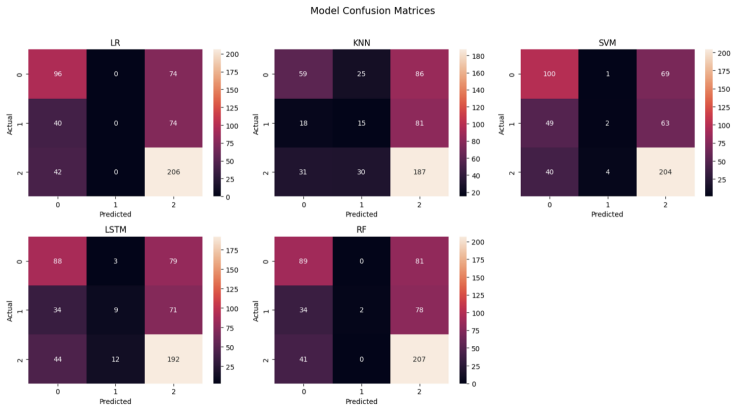
Overall, the relatively small variation in metrics across models, for example accuracy ranging from 0.491 to 0.575 prompted us to explore different areas of model analysis in order to find our best Recency Model.

5.2 ROC, Heatmap, Confusion Matrix



To better understand how each model classifies specific match outcomes — win, loss, and draw — we analysed their prediction patterns with further visualisations. The prediction heatmap shows overall classification tendencies, while our confusion matrices give more detailed insights into class-specific performance.

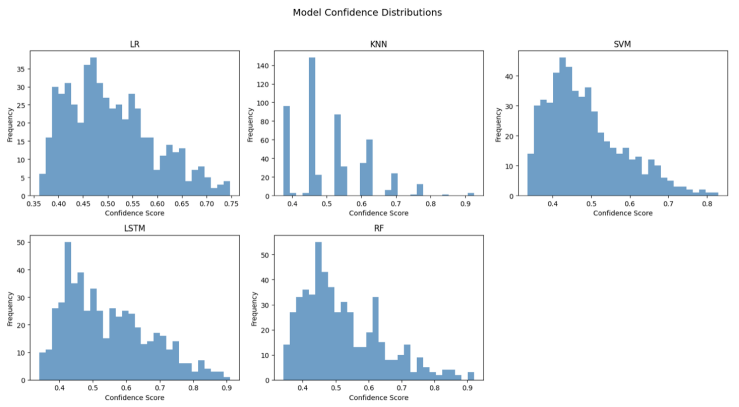


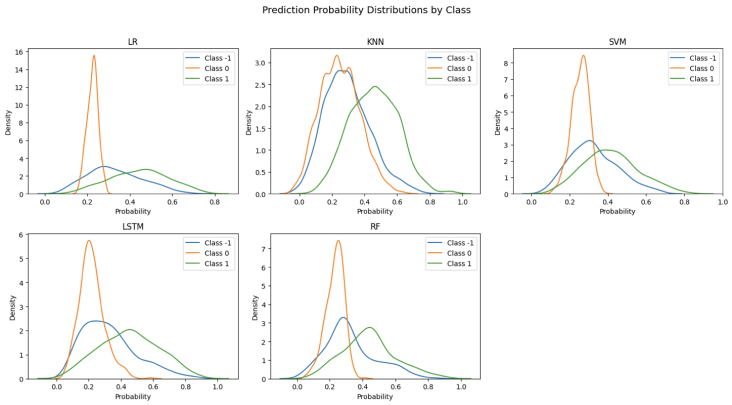


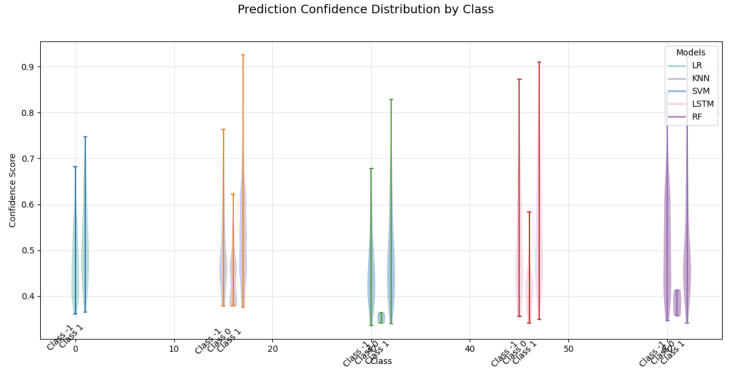
ROC curves demonstrate each model's ability to discriminate between classes across different classification thresholds.

Observing the confusion matrices and prediction heat maps we can easily see a consistent pattern across the models- there is a systemic difficulty in predicting draws. In Contrast to this, KNN made 70 draw predictions (with 21 correct), and LSTM made 24, while all other models showed extreme resistance to predict draws, often making only 1-2 draw predictions across all cases. The ROC curves demonstrate that the models particularly struggle with classifying draws, with the class 0 curve being slightly above random for all models. Interestingly, the unoptimised Random Forest model didn't exhibit this behaviour before being optimised, and correctly predicted 69 draws. However after optimisation, accuracy improved by 6.2%, and the model began exhibiting the same resistance towards draw predictions as the rest of the models. This may be because draws are the least frequent outcome in our dataset, creating a class imbalance problem. Moreover, draws may often result from games where team performances are well balanced, which is difficult to capture with a limited dataset. Our models improving accuracy when essentially avoiding draw predictions may suggest that statistical patterns leading to draws are more complex, subtle, and generally more difficult to identify.

5.3 Prediction Confidence Analysis





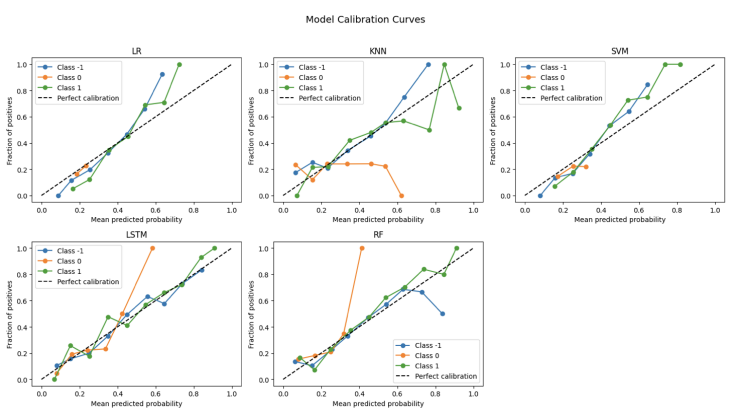


The pattern across all of the models' histograms is quite consistent. They are relatively right skewed, and concentrated at values of 0.4 - 0.6 confidence score. This suggests the models' often find it difficult to decide on the outcome of a match with high confidence, and hedge their bets. Since wins and losses are more common in the dataset, this may result in the models predicting wins/losses more than draws.

The density plot further reinforces our models reluctance to predict draws, where all models show sharp and narrow peaks at Class 0, at low probabilities. Contrasting to this, Classes -1 and 1 display broader distributions into higher probabilities, again emphasising the models inclination to make directional predictions. Some models have reasonably clear separation in directional classes, whilst SVM has particularly flat distributions for directional predictions.

This pattern is further validated by the Prediction Confidence by Class violin plot, where we can see consistently lower confidence scores for Class 0 predictions across all of our models– that is, when models do predict draws, they do so with significantly less confidence than directional predictions.

5.4 Calibration Curves



The calibration plots, in conjunction with our previous analysis allow us to make some insights to each model.

For LR, calibration plots show small deviations from the perfect line for win/loss predictions. It displays a sharp spike for draw predictions at approximately 0.4 probability, but given the model only made 24 draw predictions in total, this spike is actually misleading and draws attention to the model's fundamental inability to properly calibrate draw probabilities, despite reasonable calibration for win/loss predictions.

KNN's plot displays the most erratic patterns of all models, with significant zig-zagging around the perfect calibration line; however, it's the only model making a reasonable amount of draw predictions which can be seen in the more complete line for class 0 in both plots.

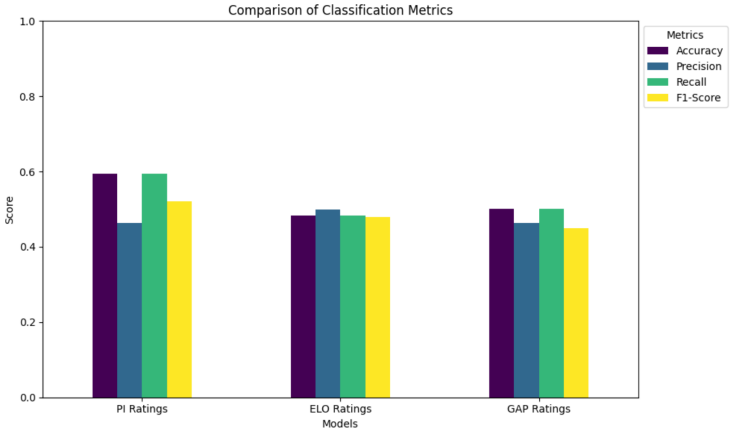
SVM has the most consistent calibration across probability ranges, as we can observe smooth lines that follow the diagonal reference lines– however, the short lines in the plot display its reluctance to make high-confidence classifications, and this is particularly the case for draws.

LSTM and RF show similar patterns of miscalibration, LSTM has more consistent curves across all classes but shows high levels of deviation at higher probabilities. In contrast, RF presents a much more volatile calibration curve. For both models, we can see the Class 0 lines steeply increase at approximately 0.4 predicted probability, suggesting challenges in accurately classifying draws at this probability level.
These calibration analyses show important details in the behaviour of our model's classification decisions that weren't apparent from accuracy metrics alone. LSTM achieved the best prediction accuracy, but its calibration curves show it becomes less reliable when making high confidence predictions. SVM demonstrates the most consistent and well-calibrated probability estimates, which is very valuable in applications where confidence is important but predicts draws much less than LSTM. KNN's ability to predict draws much better than the other models comes with a tradeoff of erratic calibration, while LR and RF show reasonable calibration, but struggle with draw predictions.

5.4.1 Best Recency Model

Our analyses of the models' performances have allowed us to reason about which model is our best Recency Model. Although models like KNN are considerably better at successfully predicting draws than other models, this comes at the cost of erratic calibration. SVM had the highest accuracy scores, and demonstrates the most consistent and well-calibrated probability estimates- which is very valuable in applications where confidence is important. As such it would be the best model to use out of the Recency Models.

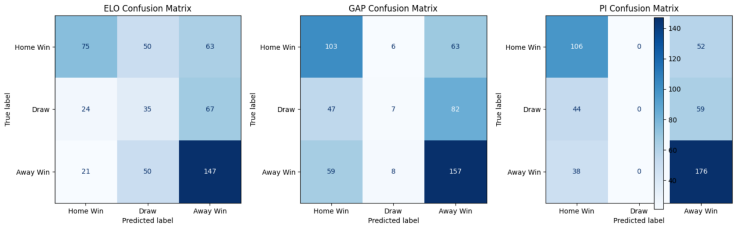
5.5 Rating Model Results



All rating model's utilised the recency features engineered on top of the team ratings calculated for each match. To prevent the *rating feature* from being overshadowed by other recency features, only the *Average Last 5 Games* features were used. This meant that average of last 4, ..., 1 games were discarded for ratings. All rating models leveraged regression as suggested by their respective papers [1,2,4].

ELO ratings used two Linear Ridge Regression models for predicting expected goals for both home and away team. A threshold of 0.15 between expected goals were chosen to determine the match outcome as a "Draw".

PI and GAP ratings both used Logistic Regression. They also leveraged Grid Search for hyperparameter optimization. As seen on the bar chart above, PI ratings is the clear winner with an accuracy of 59.3%. An important thing to note here for both the recency and the rating models is that **they all use last 5 games average statistic**, therefore, they perform best with up-to-date data, and the lack thereof can significantly impact performance.



To better understand how these rating models classified the matches, we can further analyse their prediction patterns. As seen above, ELO ratings model is the one that predicts the most draws, often inaccurately. GAP rating model predicts very few draws and PI rating model does not predict any draws at all, turning this task into binary classification.

5.6 Overall Best Model

If pure accuracy is considered, the best model is the PI rating model. However, the model fails to predict a single match as "Draw", basically turning the task from multi-class classification to binary classification. Due to this reason, we have opted to use the model with the second-best predictive accuracy, which is the SVM - L2 regression model. This SVM model is able to predict all 3 labels and has a higher precision, and therefore is going to be our model of choice for our final predictions.

6 Final Predictions on Test Set

Here are the results using the SVM - L2 regression model:

predictions_submission.csv

```
HomeTeam,AwayTeam,FTR
AFC Bournemouth,Liverpool,A
Arsenal,Man City,H
Brentford,Spurs,A
Chelsea,West Ham,H
Everton,Leicester City,H
Man Utd,Crystal Palace,H
Newcastle,Fulham,A
Nottingham Forest,Brighton,A
Wolves,Aston Villa,A
```

7 Conclusion

Our detailed analysis and feature engineering has revealed the challenges inherent in football match prediction. Through implementing recency-based metrics and rating systems, we developed novel models which approach, and many of which exceed traditional bookmaking accuracy of 53%. The consistent difficulty of predicting draws we faced across all models, despite varying approaches to feature selection and model architecture has shown that certain aspects of football prediction remain challenging. Although our PI ratings model achieved encouraging results with 59.3% accuracy on our test set, this came at the expense of precision, and the model ended up guessing no draws. We have decided that our SVM - L2 regression model is our best one yet, with 57.5% prediction accuracy on our test set, better precision and predictions for all labels. We realise that our accuracy compared to the bookkeepers is quite high, and the simple explanation is, our models had a total of 2761 data points to begin, and guessed almost exclusively for one of the two teams to win. From the shortcomings of our project, we encourage future work on football prediction to explore other approaches to the class imbalance problem, specifically for predicting draws and exploring other features that may encapsulate football matches more effectively.

References

[1] Wheatcroft, E. (2020). A profitable model for predicting the over/under market in football. International Journal of Forecasting, 36(3), 916–932.

[2] Hvattum, L. M., and Arntzen, H. (2010). Using ELO ratings for match result prediction in association football. International Journal of Forecasting, 26(3), 460–470.

[3] Hubáček, O., Sourek, G., and Železný, F. (2019). Incorporating domain knowledge in machine learning for soccer outcome prediction. Machine Learning, 108, 97–123.

[4] Constantinou, A. C. (2019). Dolores: a model that predicts football match outcomes from all over the world. Machine Learning, 108, 49–75.

[5] Agrawal, R. (2024). Principal Component Analysis (PCA) Explained: Step-by-Step. Built In.

[6] Chen, A. (2024). Chi-Square Statistic: Definition, Formula, and Uses. Investopedia.

[7] Wikipedia contributors. (2024). Elo rating system. In Wikipedia, The Free Encyclopedia.

[8] W3Schools. (n.d.). Machine Learning - Grid Search

[9] GeeksforGeeks. (2024). Understanding Logistic Regression. *GeeksforGeeks*.

[10] GeeksforGeeks. (2024). Support Vector Machine (SVM) Algorithm. *GeeksforGeeks*.

[11] GeeksforGeeks. (2024). K-Nearest Neighbor (KNN) Algorithm. *GeeksforGeeks*.

[12] Onur Akkose. Deep Learning Türkiye. (2024). Uzun-Kısa Vadeli Bellek (LSTM). *Medium*.

[13] Oleszak, M. (2022). Feature Selection Methods and How to Choose Them. *Neptune.ai*.

[14] GeeksforGeeks. (2024). Dropout Regularization in Deep Learning. *GeeksforGeeks*.