COC257

F221887

INVESTIGATING THE USE OF NEURAL NETWORKS FOR FOOTBALL MATCH RESULT FORECASTING

by

Owen Waller

F221887

Supervisor: C Dawson

Department of Computer Science
Loughborough University

Submitted: 6th May 2025

Abstract

An investigation into the use of neural networks for predicting the outcomes of football matches for 8 European leagues from 2018-2024. In contrast to traditional models, this model will focus on using advanced stats, calculated from match data in a comprehensive data pre-processing stage. Multiple network architectures are evaluated for their suitability to the task in a comprehensive machine learning pipeline, with performance assessed using both standard predictive metrics and the model's potential to generate long-term returns when integrated into sports arbitrage strategies.

The final network developed outperformed all baseline models in accuracy and recall, and provided a long-term ROI of 12.29% when evaluated against market odds.

Acknowledgments

Thank you to Dr Christian Dawson for his expert advice and guidance throughout this project.

Contents

1. Introduction

1.1 Motivation

Football is undoubtedly the most popular sport globally, with billions of fans worldwide and a market estimated at around $60 billion, and a betting market which is worth over $4 billion alone (Statista). Consequently, there is a growing demand for analytical tools to predict match events and outcomes. Creating such tools can be extremely challenging, due to the sport's dynamic and unpredictable nature, with traditional models proving ineffective at accurately forecasting matches over a large sample size.

However, neural network approaches have been demonstrated to be more effective, with models achieving up to 60% accuracy. The motivation for this project lies in using the increasing availability of detailed and accurate football data, combined with the rise of more advanced statistics, in order to build an improved neural network for match forecasting which can be leveraged to improve understanding of dynamics and exploit undervalued outcomes within bookmaker odds.

Additionally, on a personal level, this project will improve machine learning, statistical analysis, and data pre-processing skills, and provide a great introduction into the processes and practises of the industry football match prediction industry.

1.2 Project Objectives

1. Conduct a literature review and research to understand previous forecasting methods – their successes, failures, and conclusions
2. Gather match data from credible sources to create a large dataset of fixtures from the relevant leagues and time period
3. Select relevant predictors for the model and process the match data in order to produce these predictors
4. Develop a regression neural network to predict the score difference between teams using the predictors as inputs, and evaluate performance
5. Develop a probabilistic neural network to give predicted odds for each match outcome
6. Develop the final network model using a combination of the previous networks and improvements to learning algorithms.
7. Evaluate network performance by using standard statistics as well as comparisons to other models and betting markets
8. Discuss limitations of both the model and its future scalability

2 Literature Review

The football betting industry is worth over 4 billion U.S. dollars as of 2024 (Statista), and as such it is probable that any highly successful predictive models may not have been published in literature but instead used or sold for profit.

2.1 Early Predictive Models

As Tuyls et al (2021) argue, Football is the most challenging sport to analyse of all major team sports. This is due to the nature of the game, with minimal scoring, few salient events, and a large number of players each with their own unique roles. Nevertheless, there have been many prior attempts to used computerised models to forecast the outcomes of matches.

Early football prediction models relied on simplified metrics to predict match outcomes, due to the lack of quality data available compared to currently. Joseph, Fenton, and Neil (2006) used Bayesian networks to predict the results of Tottenham Hotspur from 1995-1997, by categorize the quality of opposing teams on a three-point scale: high, medium, and low. This scale corresponded closely with final league positions, making it a reasonable proxy for team performance. They also used simple player data, by providing a true or false input based on whether key players were starting the match. This approach paved the way for football prediction by using quantitative team and player data, however was limited in its range and accuracy of predictors – it fails to consider multiple factors including, form, tactical matchups and more.

Another simplistic model used was that created by Hvattum and Arntzen (2010), which examined the value of assigning ratings based on their past performance in order to predict future results, similar to that of the chess ELO system: when a team beats another with a significantly higher ELO rating, they are rewarded more than they would have been if the oppositions ELO was lower than theirs. These elo ratings were then used to derive covariates which are then used within regression models. Whilst the final model was unable to meet its success criteria of outperforming the betting odds, it proved the value of using ELO ratings within sports prediction and as such they are a frequent predictor for more recent models.

2.2 Neural Networks

Neural network approaches have great potential in predicting sports results, due to their ability to capture many input predictors and work well with large data sets (Chen et al 2019). This is achieved by mimicking the human brain using layers of interconnected nodes, which are “trained” by adjusting their weighting using a backpropagation algorithm (J.E and J.M, 2001). Naturally, there have been many attempts to apply these neural networks in the context of football. Flitman(2006) developed a network-based model to predict the winner of matches in the Australian League, whilst Arabzad et al (2014) focused on the Iran Pro League. These models were both limited by their predictors, likely due to the lacking availability of data within

these leagues, forcing input features to the network to be broad and less accurate in predicting the winner.

Timmaraju et al (2013) and Baboota and Kaur (2019) both focused on predicting the results of the English Premier League, considered to be the league with the most advanced data readily available, and produced success rates of 50.1% and 56.5% respectively. Sujatha et al (2018) focused on the German Bundesliga, specifically the matches between rivals Bayern Munich and Borussia Dortmund. The success of these models was limited by their input predictors despite the large amounts of data, in part due to omissions such as possession or bookings, however more so because of the simplistic nature of the data used. This opens the scope for this project, building a match prediction neural network model which integrates more advanced statistics into its input parameters, as opposed to those which ignore context.

2.3 Advanced Statistics

The growing availability of detailed football data has fuelled the development of advanced statistical methods, which complement neural networks in the pursuit of more accurate match predictions. Modern football analytics use much more detailed datasets than previously available, which include player tracking data, expected goals (xG), passing networks, and pressing metrics.

The most widely used of these metrics in football prediction is expected goals (xG), which quantifies the likelihood of a shot resulting in a goal based on factors such as shot location, angle, and defensive pressure. Introduced by analysts like Caley (2015), xG models have revolutionized how football matches are analysed and predicted. By incorporating xG data into predictive models, researchers can better account for the quality of chances created by teams, rather than relying solely on raw goal counts – it aims to remove luck from the analysis of football matches (Tippett 2019). There have been attempts to build upon the logic used in expected goals to quantify other stats, such as expected assists and threat, although these have proven to be less accurate, likely due to the greater number of factors at play when analysing a longer period of play compared to taking a shot.

Player tracking data is another recent innovation in football analytics. Advances in wearable technology and computer vision have made it possible to capture detailed positional data for every player on the pitch. This data can be used to construct passing networks, which visualize how teams control and distribute the ball during matches. Studies like those by Bornn et al. (2018) have shown that passing networks can be highly predictive of match outcomes, as they reflect underlying team strategies and cohesion. However, this data is extremely expensive to access due to the nature of its collection and its potential ability to grant teams an advantage, and as such is not a suitable predictor for this neural network.

3. System Design

3.1 Selecting Predictors

3.1.1 Simplistic Predictors

All team stats given to the model as input are given in last 5 games average and last 50 game average form. This is to capture teams short term form whilst also balancing their quality over an extended period of time. The reason for the 50-game limit on long-term averages is to prevent teams with drastic change suffering from lag in their stats reflecting this change. Additionally, all averages were also applied in a head-to-head context, meaning that for each individual stat there will be 3 inputs to the network per team – average, last 5 games average and average vs opponent.

Additionally, due to the inclusion of both the top 2 English football leagues, clubs to appear in both divisions will be represented as two separate teams. This is because teams stats are likely to drastically differ between the two divisions, due to a large fluctuation in opponent difficulty.

Although a surface level stat, including "Goals For" (the number of goals a team scores) and "Goals Against" (the number of goals a team concedes) as predictors is crucial because they directly represent a team's offensive and defensive output. Goals scored reflect the general effectiveness of a team's attack, including the skill of their forwards and midfielders, as well as their team's overall system's ability to generate chances. Conversely, goals conceded reveal the strength (or weakness) of their defensive line and goalkeeping. Although they can be flawed due to match context – scoring 5 goals in a win against easy opposition followed by 0 in a loss in a more difficult match would suggest the team has capability to score 2.5 goals per game, whereas this might not be the case – the stat still provides a good overall indication of a team's quality.

Points per game acts as another simple but effective predictor, as over a period of time it is likely to reflect a team's performance fairly. It is estimated that across Europe's top leagues, the team higher in the table (and therefore with higher points per game) wins or draws the match 60-70% of the time (TheStatsDontLie.com). This demonstrates the stats effectiveness at predicting the outcome of a match.

Possession represents the amount of time a team holds the ball within a match, expressed as a percentage. The ability to retain possession for extended periods of time has been suggested to be linked to team success (Penas, Dellal 2010), and as such it serves as a good predictor for capturing a team's quality and also playstyle. However, possession can be heavily influenced by game state – a team protecting a lead may not attempt to retain the ball and prefer to instead play a more risk-averse, low possession game. This means that the stat itself only has a weak correlation to match result, however combined with other statistics it helps to build a picture of a team's dynamics and quality.

Shots and shots on target, both for and against, illustrate whether a team is usually the more attacking side within their games. Teams that take more shots per game are usually of higher creative quality and also score more goals. Likewise, teams which concede more shots are often less defensively organised or have worse individual player quality within their backline. These metrics are limited by context, as a team could take many low-quality shots, but are good predictors to demonstrate whether a team is attacking or defensive minded, especially when coupled with possession.

3.1.2 Advanced Statistical Predictors

"Expected Goals For" (xG) and "Expected Goals Against" (xGA) are advanced metrics that provide deeper insight into a team's performance by quantifying the quality of scoring opportunities created and conceded. Unlike raw goals, which depend on outcomes, xG and xGA consider the likelihood of each chance being converted based on factors like shot location, angle, and type (Tippett 2019). This makes them powerful predictors because they account for underlying performance rather than surface results, which can be skewed by randomness or luck (e.g., a goalkeeper's extraordinary saves or a fluke goal). Including these metrics helps the model evaluate a team's ability to consistently create high-quality chances and their tendency to allow opponents similar opportunities. They also allow for better prediction of future performance, as they are less volatile than actual goals, making them essential for a reliable predictive model.

Expected goal difference (xG – xGA) is also used as a predictor. This offers separate value to the raw xG stats, as it shows the average winning or losing margin of the team, whereas the former shows whether games are high or low (expected) scoring typically.

An ELO system is also used as a feature. This is inspired by the chess ELO system and the work of Hvattum and Arntzen (2010). This assigns each team an ELO rating initially, and updates it following the result of a match based on the opponents ELO. The following formula is used:

1. Calculate the expected score:

$$E_A = \frac{1}{1 + 10^{\frac{(R_B - R_A)}{400}}}$$

Where RB and RA represent the respective elo ratings of each team.

2. Update the elo ratings based on the result of the match:

$$R'_A = R_A + K \cdot (S_A - E_A)$$

$$R'_B = R_B + K \cdot ((1 - S_A) - (1 - E_A))$$

Where R'A and R'B represent the new elo ratings, SA represents the result from the perspective of team A, and K represents the scaling factor. The scaling factor determines the sensitivity of the elo system to match results.

The ELO system serves as a good measure of club status and quality throughout the seasons. It can also capture intangibles which may be missed by traditional stats such as fear factor, and reacts quickly to teams drastic changes in form.

3.1.3 Other Predictors

Teams average number of yellow and red cards were also used as a feature. This represents the discipline of a team, and teams with higher numbers of cards can often lose games due to sending offs despite appearing the better team prior. Consequently, the network should apply an appropriate penalty to such teams.

Finally, implied betting odds from before kick-off are used as a feature in order to gain the consensus amongst fans and betting companies. These can often capture data statistics may have missed in the form of intangibles, such as player welfare or additional incentives for one side to win. Implied probabilities are calculated as follows:

$$P_{\text{Normalized, Outcome}} = \frac{\frac{1}{\text{Odds}_{\text{Outcome}}}}{\sum \left(\frac{1}{\text{Odds}_{\text{Win}}} + \frac{1}{\text{Odds}_{\text{Draw}}} + \frac{1}{\text{Odds}_{\text{Loss}}} \right)}$$

However, these probabilities are not always fully motivated by match predictions—the primary goal of betting firms is to generate profit rather than correctly predict a match's result, and as such, odds may be positioned in a way that isn't reflective of the actual favourites for a match.

3.2 Dataset

3.2.1 Dataset Coverage

The dataset used for this predictive mode is from "footystats.org" premium subscription. It is worth noting that with greater funding, it may be possible to obtain a higher quality dataset, however given the highly competitive and lucrative nature of the football industry, prices for this are extremely high due to targeting betting firms and clubs themselves as clients rather than individuals. The dataset for this project contains post-match data from the following leagues, from 2018/19 to December of 2024/25:

League	UEFA Coefficient	UEFA Ranking
English Premier League	110.267	1
Italian Serie A	95.543	2
Spanish La Liga	92.525	3
German Bundesliga	85.831	4
French Ligue 1	71.665	5
Portuguese Liga Nos	62.266	7
Belgian Pro League	56.850	8
English Championship	N/A	N/A

Table 1 - UEFA Coefficient and Ranking for Leagues Selected

UEFA coefficients are designed to measure the relative strength of a country's teams and therefore league by using their results in European competition (UEFA). As this is calculated per country and not per league, there is no available coefficient for the English Championship, given that it is the second division in England.

3.2.2 Dataset Format

The dataset contains post-match data for each match describing the events of the game and its metadata (full table can be found in Appendix A).

Column Name	Description
timestamp	Unix Timestamp of the Kick Off Time
date_GMT	Kick Off Time in GMT (e.g., Feb 18 2017 - 9:30am)
status	Match status: incomplete / complete / postponed / cancelled
attendance	Number of attendees at the stadium (if available)
team_a_name	Home Team Name
team_b_name	Away Team Name
refree	Full name of the Referee (if available)
home_ppg	Points Per Game for Home Team - Current
away_ppg	Points Per Game for Away Team - Current
home_team_goal_count	Number of goals scored by the home team
away_team_goal_count	Number of goals scored by the away team
total_goal_count	Total Number of Goals
total_goals_at_half_time	Total Number of Goals at Half-Time
home_team_goal_count_half_time	Goals by home team at half-time
away_team_goal_count_half_time	Goals by away team at half-time
home_team_goal_timings	Minutes at which the goals were scored - Home Team (if available)

away_team_goal_timings	Minutes at which the goals were scored - Away Team (if available)
home_team_corner_count	Corner count - Home Team
away_team_corner_count	Corner count - Away Team
home_team_possession	Possession % - Home Team
away_team_possession	Possession % - Away Team
team_a_xg	Expected Goals - Home Team
team_b_xg	Expected Goals - Away Team
btts_percentage_pre_match	Pre-match average % for Both Teams to Score
over_15_percentage_pre_match	Pre-match average % for Over 1.5 Goals
over_25_percentage_pre_match	Pre-match average % for Over 2.5 Goals
over_35_percentage_pre_match	Pre-match average % for Over 3.5 Goals
over_45_percentage_pre_match	Pre-match average % for Over 4.5 Goals
odds_ft_home_team_win	Full Time Win Odds - Home Team
odds_ft_draw	Full Time Draw Odds
odds_ft_away_team_win	Full Time Win Odds - Away Team

Table 2 – Format of Matches Dataset from FootyStats.Org – full table in Appendix A

3.3 Core Requirements

The core requirements of this system are stated below. It is essential that these are fulfilled in order for the project to be considered complete and functional.

1. **The system should support ingestion of historical football match data and check for nulls, outliers, and mismatches in database schema**
2. **The system should pre-process raw match data into features to be interpreted by the model**
3. **The system should ensure no data leakage takes place by supporting time-aware feature construction**
4. **The system should define and initialise both classification and regression neural network architectures**
5. **The system should support training the models using supervised learning**
6. **The system should improve the learning process of models using optimisation methods such as hyperparameter tuning**
7. **The system should evaluate the models using metrics such as accuracy and loss**
8. **The system should support comparisons between models and give reports of each models output on the validation set**

3.4 Success Criteria

The success criteria are a set of goals and standards the system should meet in order to be considered a success. They are intended to be revisited upon project completion. “The model” is assumed to be the pest performing neural network model developed by the system.

1. **The model executes within reasonable time on standard hardware**
2. **The model achieves superior accuracy to random guessing (> 33%)**
3. **The model has greater accuracy than pre-defined naïve models**
4. **The model clearly captures features of each outcome and its most common prediction for each result class matches said class**
5. **The hybrid model outperforms a simple classification in draw recall**
6. **The model yields a positive non-negligible ROI when integrated into a targeted betting strategy (> 1%)**

3.5 Network Architecture

The proposed network architecture combines a classification network and a regression network operating in parallel on standardized match data. The classifier predicts match outcomes directly, while the regressor estimates goal difference. A post-processing logic layer integrates both outputs to override low-confidence or contradictory predictions, improving overall robustness.

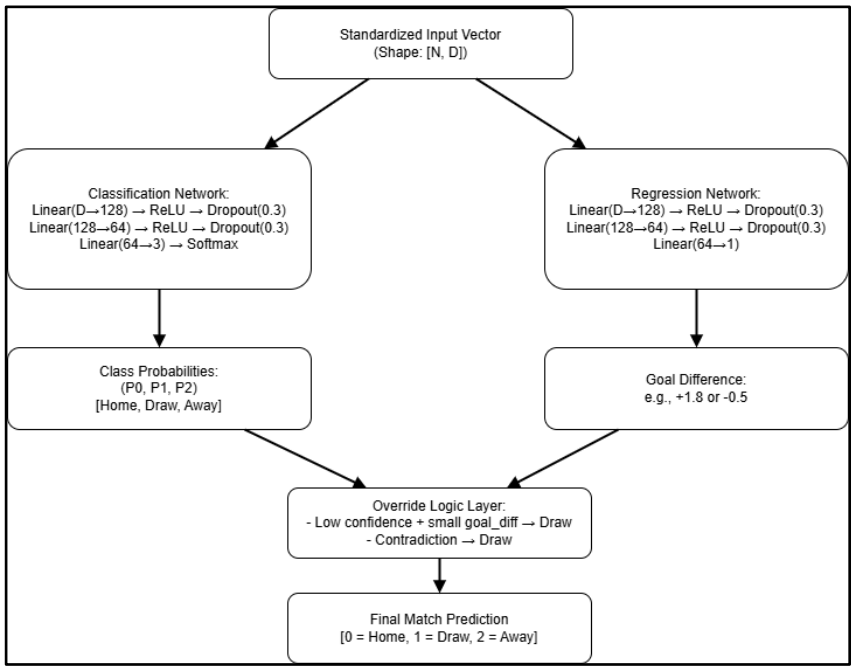

Figure 1 – High Level Abstraction of Proposed Network Architecture

4. System Implementation

4.1 Environment

For this system, Python was selected as the primary development language due to its simplicity, flexibility, and wide ecosystem of data analysis and machine learning libraries. The pandas library was used extensively for data manipulation and preprocessing tasks, while Pytorch served as the deep learning framework for building and training predictive models.

Python was installed from the official website, with the windows version selected as all development was completed on windows devices. The necessary libraries were then installed using pip, and a dedicated Python virtual environment was created to isolate the dependencies and ensure consistency across development sessions.

This environment was created and managed using anaconda, which provides a convenient interface for managing dependencies and keeping separate python projects on the device from interfering with one another.

To facilitate rapid prototyping and improve workflow efficiency, Jupyter Notebook was installed and used as the primary development interface. Its interactive environment allowed for immediate feedback during model experimentation and data exploration, making it an ideal tool for iterative development and model comparisons.

4.2 Software Engineering Methodology

The project will follow a “Water-Scrum-Fall” methodology, combining the structured planning and evaluation of Waterfall with the iterative flexibility of Agile Scrum. An initial Waterfall-style phase will be used to define objectives, gather requirements, and structure the system design, ensuring a solid foundation for development.

This will be followed by short, iterative Scrum cycles, during which core components will be developed and refined, such as feature generation or network training cycles. This hybrid approach will suit the investigative nature of the project by allowing flexibility during experimentation and performance evaluation, while still maintaining overall direction and stability. The final model will then be assessed against the success criteria to determine whether it has satisfied the predefined goals.

4.3 Data Pre-Processing

The first step of the project implementation is to derive the desired features from the dataset (described in 3.2), as for our predictive model the data is required in a pre-game format to prevent any potential data leakage.

4.3.1 Data Cleansing

Prior to data manipulation, the dataset was cleaned by removing any rows where key data was missing, defined by data from the following columns:

```
# Define all columns used in later stats and calculations
required_columns = [
    'home_team_name', 'away_team_name',
    'home_team_goal_count', 'away_team_goal_count',
    'team_a_xg', 'team_b_xg',
    'home_team_shots', 'away_team_shots',
    'home_team_shots_on_target', 'away_team_shots_on_target',
    'home_team_possession', 'away_team_possession',
    'home_team_yellow_cards', 'away_team_yellow_cards',
    'home_team_red_cards', 'away_team_red_cards',
    'date_GMT',
    'odds_ft_home_team_win', 'odds_ft_draw', 'odds_ft_away_team_win'
]
```

This results in a cleaned dataset consisting of just under 19000 rows, with approximately 2000 being dropped.

```
Cleaned data saved to ../processed_data/matches_cleansed.csv
Rows remaining: 18904
Rows dropped: 2127
```

4.3.2 Feature Generation

The feature generation system is built using a nested dictionary structure. Each team is assigned a dictionary that holds rolling lists of its historical match statistics as well as an Elo rating. Inside each team's dictionary, there is another dictionary (opponents) that tracks head-to-head results against specific opponents. This two-dimensional mapping allows quick access to both general team form and specific opponent-related stats.

This process is streamlined through the use of several helper functions, in order to keep the feature generation code clean and consistent, which is crucial to prevent data leakage or double updates of statistics:

- Get_match_result() for fetching the match result to be used in later calculations
- Get_K_factor() for dynamically adjusting the elo update intensity depending on result
- Update_elo() for calculating the new elo ratings for each side after the conclusion of the match
- Update_rolling() for updating the statistics within each team’s dictionary after a match

Key safeguards are used to ensure there is no data leakage throughout the process, with feature being built strictly from data available before the match. The dictionaries are only appended after the feature row for each match has been generated in order to prevent that games statistics leaking into rolling averages, and the same sequential logic is used for elo updates. Similarly, the betting odds used are exclusively those before the game has started.

Matches where no prior data is available for one team are marked with the "first_game" flag being set to true, and later removed from the dataset in order to prevent columns being filled with zeros or random noise.

The completed generation process results in the following input features (see **3.1 – Selecting Predictors** for justification and definitions):

- Features in Figure 4 are recorded for both home and away teams, for both the long-term rolling window and for the last 5 games

Feature	**Definition**
Average xG	Expected goals scored per game
Average xG Against	Expected goals conceded per game
PPG	Points per game
Average GS	Goals scored per game
Average GC	Goals conceded per game
Average shots for	Shots taken per game
Average shots against	Shots conceded per game
Average SOT	Shots on target taken per game
Average SOT against	Shots on target conceded per game
Average possession	Possession per game (% out of 100),
Average Yellow	Number of yellow cards per game
Average Red	Number of red cards per game

Table 3 - Team Based Features

- Table 4 shows the remaining features:

Feature	**Definition**
Home team elo	Elo rating calculated from prior form for the home team
Away team elo	Elo rating calculated from prior form for the away team
Home team ppg vs away team	Home team points per game vs opponent
Away team ppg vs home team	Away team points per game vs opponent
Home team goals vs away team	Home team goals per game vs opponent
Away team goals vs home team	Away team goals per game vs opponent
Prob Home Win	Implied probability of a home win from betting odds
Prob Draw	Implied probability of a draw from betting odds
Prob Away Win	Implied probability of an away win from betting odds

Table 4 - Remaining Features

Once features have been generated, any remaining unnecessary columns are dropped from the dataset, leaving only the target and input columns. This includes irrelevant columns such as referee and date (those very unlikely to reflect match outcomes), in addition to columns which give data about the match after it has happened, e.g. half-time goal count, goal timings.

4.4 Baseline and Naïve Models

Before evaluating the performance of more complex models, it is critical to establish baseline and naive models for comparison. These models provide a "sanity check" by showing how well extremely simple approaches perform without any sophisticated learning or feature engineering. If a complex model does not outperform basic strategies, then it is either overfitting or adding unnecessary complexity. These baseline models also provide a basis for our success criteria, as it is vital to improve upon them.

4.4.1 Predicting Home Wins

Simple analysis of the dataset confirms the consensus that a home win is the most likely outcome in football, accounting for just over 43% of results:

```
Home win rate: 43.30%
Draw rate: 25.53%
Away win rate: 31.17%
```

This means we can form a baseline model with 43.3% accuracy which merely predicts the home team to win every game regardless.

4.4.2 Predicting from Goal Difference

Goal difference provides an overly simplistic measure of a team's quality; however it is still more accurate in predicting match outcomes then naïve guessing. By merely selecting the team with a greater pre-match average goal difference, we can achieve an accuracy of 49.84%.

```
Overall Net GD Accuracy: 49.84%
```

Applying this same formula to expected goal difference only yields a 0.28% increase in accuracy. This is due to expected goals for a team tending towards the true goas value over time.

```
Overall Net xG Accuracy: 50.12%
```

4.4.3 Predicting from Form

Predicting simply based on which team has won more points in recent games is unsurprisingly ineffective, as this fails to account for key context such as the strength of the opponents faced. Using this naïve model only gives an accuracy of 45.01% - a negligible increase from merely predicting a home win.

```
Naive Last-5 PPG Model Accuracy: 45.01%
```

4.5 Establishing a Network

4.5.1 Final Data Preparation

Before feeding input features into a neural network, it is essential to scale the data to ensure that all features contribute proportionally during training. Neural networks are sensitive to the scale of input values because the learning process relies on gradient-based optimization; unscaled features can lead to uneven gradient magnitudes, causing slower convergence or instability.

In this project, standardization was applied using StandardScaler, which transforms each feature to have a mean of zero and a standard deviation of one. This ensures that features with larger original ranges (e.g. shots or possession) do not dominate those with smaller ranges (xG, red cards), allowing the network to learn more efficiently.

Feature	Mean (Before Scaling)	Std Dev (Before Scaling)
home_team_avg_pos	49.9935	6.0822
away_team_avg_pos	50.0681	6.0756
home_team_avg_xg	1.4703	0.2311
away_team_avg_xg	1.4759	0.2319
home_team_avg_shots_for	12.5102	2.3831
away_team_avg_shots_for	12.5597	2.3927
prob_home_win	0.4334	0.1757
prob_draw	0.2539	0.0617
prob_away_win	0.3128	0.1599

Table 5 – Mean and Standard Deviation of Select Features

In order for the network to learn most efficiently and generalise well, it is important to remove any outliers within the input features. In most cases this involves removing values far from the mean based on using standard deviation or interquartile range, however given the nature of this problem extreme data can prove useful. Instead, we focus on removing any games where the match result does not reflect the events of the game; we aim to remove matches where luck had an extreme impact. This is implemented by removing any games from the dataset where one team had over 1xG greater than their opponent and still lost. This resulted in only 640 games being dropped, approximately 3% of the dataset, demonstrating the rarity of this type of match.

To evaluate the model's ability to generalise to unseen data, the dataset was split chronologically into training and testing subsets using an 80:20 ratio. This means the first 80% of the matches were used to train the neural networks, while the remaining 20% held out for final evaluation. A chronological split was chosen instead of random

shuffling to better simulate real-world forecasting conditions, where predictions are made on future games based on past data, and to prevent data leakage, as the model may pick up on changes in team's average stats and use this to draw conclusions.

To address class imbalance in the training data, an oversampling procedure was applied prior to model training. After splitting the dataset into training and testing subsets using an 80/20 chronological split, the training set was rebalanced to ensure that each match outcome class was equally represented. Specifically, the number of draw and away win samples was increased by random oversampling with replacement until they matched the number of home win samples. This oversampling was applied only to the training set, preserving the natural distribution in the test set for unbiased evaluation. Balancing the training data in this way is critical for mitigating bias toward the majority class during model learning, as without this adjustment the model will struggle to learn underrepresented outcomes leading to skewed predictions and degraded performance. The resampling ensures that the model has sufficient exposure to each class during training, enabling it to learn more discriminative features and improving its generalization across all outcome types.

4.5.2 Regression Network

The regression network in this project was implemented as a separate feedforward neural network designed to predict the expected goal difference of a match based on the input features. This model produces a single continuous output representing the predicted numerical difference between the home and away team goals. This implementation allows the model to identify the likely winner, as well as demonstrate when matches are likely to be closer or more one-sided.

Pytorch was the chosen library for the implementation due to a combination of its flexibility and my prior experience using it. The library allows for models to be defined or updated rapidly, and this allows for greater freedom during experimentation and debugging. The availability of great support was also a factor in its selection, with Pytorch' s widespread use within the industry leading to a large community and high-quality documentation.

The regression network is structured as a three-layer feedforward model. The input layer receives the full standardized feature vector and projects it into a higher-dimensional space of 128 neurons. This expanded representation is then processed by a hidden layer of 64 neurons, further condensing and refining the learned features. Dropout with a rate of 0.3 is applied after both the first and second layers to improve generalization by randomly deactivating neurons during training. The final output layer contains a single neuron with no activation function, producing a continuous-valued estimate of goal difference. This architecture strikes a balance between model complexity and efficiency, with sufficient depth to capture patterns in the data without becoming overly sensitive to noise or overfitting rapidly.

The ReLU (Rectified Linear Unit) activation function is used throughout the regression network to introduce non-linearity between layers. Defined as $f(x)=\max(0,x)$, ReLU allows the model to learn complex patterns by enabling certain neurons to activate

only when their input exceeds zero. This simplicity makes it computationally efficient and well-suited to deep learning tasks. ReLU is more appropriate than functions like tanh or sigmoid in this context because it avoids the issue of vanishing gradients, which can hinder learning in deeper networks. It is also computationally efficient due to using simpler operations like thresholding rather than exponentials, making it well suited to the large amount of numerical data being processed.

4.5.3 Classification Network

The classification network follows a similar structural philosophy to the regression model but is tailored to predict categorical match outcomes rather than goal difference. Implemented as a separate feedforward neural network in PyTorch, the classifier processes the same standardized input features and outputs a probability distribution over the three possible outcomes using a softmax activation in the final layer.

The network architecture consists of three linear layers. The first layer expands the input feature vector into 128 dimensions, followed by a hidden layer of 64 neurons, and finally an output layer with three neurons corresponding to the target classes. Once again, ReLU activation functions are applied after the first two layers to introduce non-linearity, and dropout layers with a rate of 0.3 are used in the same locations as in the regression network to mitigate overfitting. The final output layer uses a softmax function to convert raw logits into interpretable probabilities that sum to one, allowing the model to express varying degrees of confidence in each class.

This design enables the classification model to capture complex relationships between input features and match outcomes, while maintaining architectural consistency with the regression network. By aligning the structure of both models, it becomes easier to integrate their outputs later in the pipeline and during training. The classification network serves as the primary predictor of match outcome and forms the backbone of the final model.

4.5.4 Post-Prediction Logic

The post-prediction logic forms a crucial part of the hybrid architecture, acting as a decision layer that combines the strengths of both the classification and regression models. While the classification network provides the primary match outcome prediction (home win, draw, or away win), its output is not accepted blindly. Instead, it is passed through a logic gate that incorporates information from the regression network, which estimates the expected goal difference. This additional layer of reasoning allows the system to override the classifier's prediction in cases where its confidence is low, or its output contradicts the more precise numerical signal provided by the regression model. The main motivation behind the inclusion of this logic gate is to improve draw recall, as this is a metric which bookmaker’s odds are often severely lacking in and as such can be exploited.

This logic-based override mechanism is designed around a set of well-defined conditions. The first condition checks whether the classifier’s confidence in its

predicted class is sufficiently high. If the predicted class has a confidence (softmax probability) lower than a predetermined threshold, and if the regressed goal difference is close to zero, the final prediction is overridden to a draw. This rule helps the model hedge its bets in ambiguous cases, where the classifier leans slightly toward one team, but underlying number suggest a draw may be more likely. This is important to improve the models accuracy as draws are very difficult for the model to identify, due to their lack of distinct features and that many close games result in one side winning.

The second condition addresses contradiction between the classifier and regressor. if the classifier predicts a home win but the regressor estimates a negative goal difference below a threshold, the logic gate intervenes to override the prediction to a draw. This prevents clearly conflicting outputs from degrading the system's accuracy, especially in matches where raw classification confidence might be misleading.

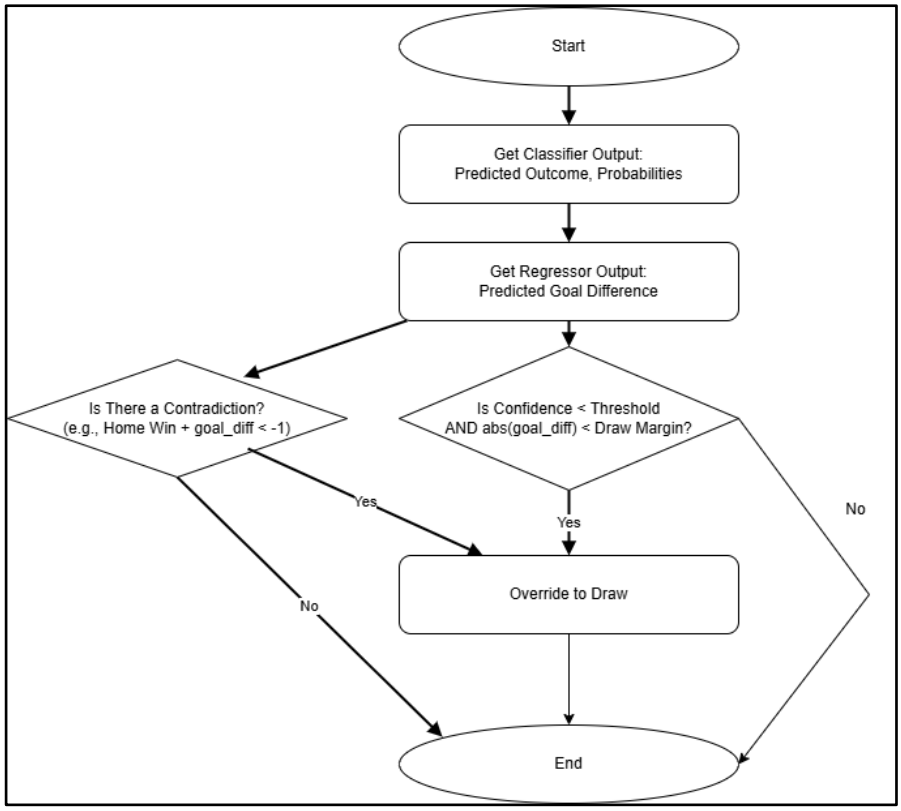

Figure 2 - Flowchart Representation of Override Logic

The override conditions are based upon thresholds which have been optimised through experimentation, defining "low confidence" and "strong negative goal difference" (see **4.8.2 – Override Threshold Optimisation**) These thresholds are conservative by design, as they exist to support the model rather than frequently "re-predict" games.

4.6 Network Training

Batch training was used throughout the training process, with a batch size of 32. This means that during each iteration of training, the model updated its parameters based on the average loss calculated from 32 examples rather than processing the entire dataset at once or a single example at a time (stochastic training). Batch training provides a practical balance between computational efficiency and learning stability. It enables faster training compared to full-batch approaches while also smoothing out the noise that can arise from single-sample updates. A batch size of 32 is a commonly used default, as it fits well within memory constraints while still providing representative gradients.

Two separate loss functions were used to train the classification and regression models in parallel, reflecting the different nature of their outputs. The classification network was trained using the cross-entropy loss function, which is standard for multi-class classification problems. It measures the divergence between the predicted class probabilities and the actual class labels, penalizing incorrect predictions more heavily when the model is confident but wrong. For the regression model, mean squared error (MSE) loss was employed. This loss function calculates the average of the squared differences between the predicted and actual goal differences, emphasizing larger errors more significantly. Using distinct loss functions ensures that each model is optimized appropriately for its task—categorical accuracy for classification and numerical precision for regression.

Both networks were trained using the Adam optimizer, a widely used stochastic gradient descent algorithm with adaptive learning rates and momentum. Adam was chosen due to its proven effectiveness in handling sparse gradients and noisy data, which is common in real-world football match datasets. The optimizer combines the benefits of two other extensions of stochastic gradient descent in AdaGrad and RMSProp by maintaining per-parameter learning rates that are adapted based on the first and second moments of the gradients.

The classification and regression models were trained in parallel using the same batches of input data. During each epoch, both models received the same feature vectors, but each produced its respective output and computed its own loss. These losses were backpropagated independently through each network, and their gradients were used to update the parameters separately via the Adam optimizer.

Although the networks operate independently, training them together in the same loop allows for consistent exposure to the same data distributions. This setup ensures alignment between the models, enabling coherent integration of their outputs later in the prediction pipeline. A total of 35 epochs were used for training, as this proved most successful in allowing both networks to converge without overfitting.

4.7 Network Output

The network output is in the form of a csv table, which includes the predictions of each model, the combined prediction and also details about the models confidence and whether the override functionality was triggered. This provides a comprehensive report on the models predictions on the testing set, and allow for greater analysis and visualisation opportunities.

Column Name	Description
match_index	Index of the match within the full dataset (post-train/test split).
actual_result	The true outcome of the match (0 = Home Win, 1 = Draw, 2 = Away Win).
predicted_result	Final predicted result after applying override logic.
original_classifier_output	The predicted class from the classifier before any override was applied.
override	Boolean flag indicating whether the prediction was overridden (True/False).
goal_diff	Actual goal difference (home goals – away goals) from the match.
goal_diff_pred	Predicted goal difference from the regression model.
model_prob_home	Classifier's predicted probability for a home win.
model_prob_draw	Classifier's predicted probability for a draw.
model_prob_away	Classifier's predicted probability for an away win.
implied_prob_home	Implied probability of a home win from betting odds.
implied_prob_draw	Implied probability of a draw from betting odds.
implied_prob_away	Implied probability of an away win from betting odds.

Table 6 - Network Output Table Format

4.8 Improvements to Learning

4.8.1 Hyperparameter Optimisation

The system allows for hyperparameter optimisation based on the overall accuracy of the predictions, evaluating each model on the testing set. This was done using a grid search method, testing combinations of different values for learning rate, dropout rate and number of epochs. This approach ensures that the impact of each

hyperparameter is thoroughly explored across a controlled range, reducing the risk of relying on suboptimal default values , and offers a more exhaustive method compared to manual tuning. Grid search can be computationally expensive, however only needed to be executed once for this model and is well suited due to the fixed network architecture.

The results of hyperparameter tuning are shown in table X:

Learning rate	dropout	epochs	accuracy	precision
0.0001	0.3	35	0.52881	0.48588
0.0001	0.2	50	0.52521	0.4839
0.0001	0.3	50	0.52493	0.48443
0.0001	0.4	25	0.52327	0.48322
0.0001	0.2	35	0.52133	0.47492

Table 7 - Hyperparameter Tuning Results

As illustrated by table 7, a combination of learning rate 0.0001and dropout layer strength 0.3 trained over 35 epochs resulted in the best performing model.

4.8.2 Override Threshold Optimisation

The output format of the network allows for detailed evaluation of the override logic, calculating its accuracy compared to both the original prediction and actual result. This analysis can be used to fine-tune the thresholds within the function to either increase or decrease intervention. A grid search approach is also well suited here to systematically refine the override logic, focusing on keeping the highest accuracy whilst improving draw recall.

The results of threshold tuning are shown in table 8:

Threshold / Metric	Value
draw_margin_home	0.25
draw_margin_away	0.2
contradiction_margin	1
min_class_confidence	0.46
min_away_confidence	0.39
top2_accuracy	81.11%
gain_in_draw_recall	10.1%

Table 8 – Results of Threshold Tuning for Override Logic

5. Network Evaluation and Performance

5.1 Performance Metrics

Table 9 contains a breakdown of key performance metrics:

Metric	Value	Comments
Overall Accuracy	50.39%	To be compared to other models
Regression RMSE	0.602	An acceptable RMSE score, averaging 0.599 goals away from the true score.
Sign Accuracy	70.91%	The model predicts the team which will "play better" 71.25% of the time – this is a fairly impressive rate, suggesting the model is successfully understanding patterns between team matchups.
Top-2 Accuracy	80.69%	Model is consistent at identifying the worse team – it very rarely predicts the wrong team to win in non-draw situations
Weighted F1 score	50.88%	This is a fairly disappointing score in general machine learning terms, however is understandable given the 3 class and unpredictable nature of the problem.

Table 9 - Key Metrics for Model Performance

Figure 5 shows the confusion matrix for the model:

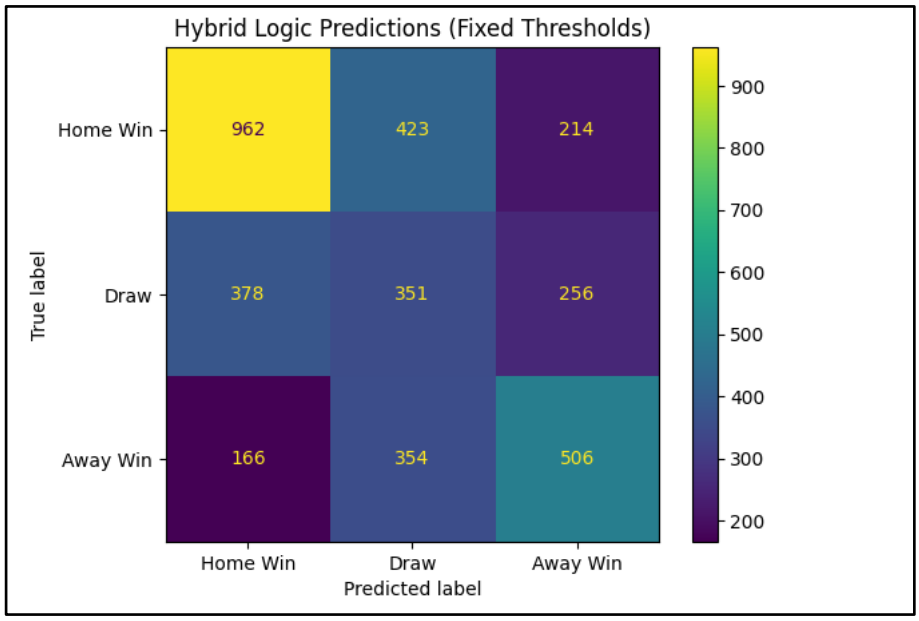

Figure 3 – Confusion Matrix for Model Performance

As shown, the model struggles to identify draws, however generally has good accuracy and precision on wins for either team.

5.2 Comparisons to Baseline Models

Figure 4 shows the breakdown of accuracy for each outcome, for the baseline models and the fine-tuned network. Table 10 shows the overall accuracy (see **4.4 – Naïve and Baseline Models** for full explanations).

Model	Overall Accuracy
Home Wins Only	43%
Net GD	49.84
Net XG	50.12
Naïve PPG	45.01
Fine-Tuned Model	50.39

Table 10 - Comparison of Overall Accuracy between Models

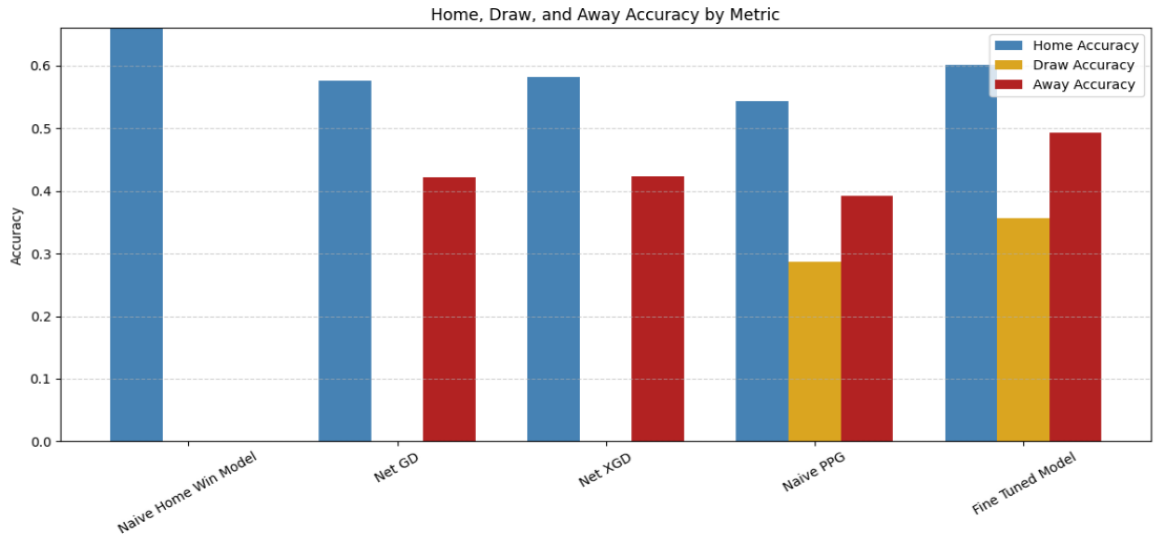

Figure 4 – Recall per Class for each Model

Due to the nature of the naïve models, overall accuracy can mask the fact they are incapable of recalling draws. The fine-tuned model clearly boasts an advantage in this area as well as when predicting away wins. Accuracy on the home win class is fairly consistent (with the obvious exception of the home win model getting 100% due to always predicting home wins), as this is the most common class and the one with the greatest number of defining features – a home win is less likely to be a close game than an away win due to the inherent home advantage.

The model clearly shows improvement over baseline models, especially identifying draw outcomes, which will prove important in generating a good ROI when applied to betting strategies (see next section **5.3 – Application of Model vs Market Odds**)

5.3 Application of Model vs Market Odds

Bookmakers' odds are typically well-calibrated due to the strong financial incentive to accurately reflect match probabilities. Their primary objective, however, is not to predict outcomes per se, but to maximize profit and minimize risk exposure. This distinction often leads to systematic distortions in the odds they offer—particularly around less popular betting outcomes. For example, draws are frequently underrepresented in bookmaker odds because the general betting public tends to prefer backing a winner. Since bookmakers must balance their books and ensure profitability regardless of the match result, they may offer less competitive odds-on outcomes that attract fewer bets, such as draws, to encourage more balanced wagering. This can be seen clearly in figure 5, which suggests a draw is almost never the implied outcome from bookmaker odds.

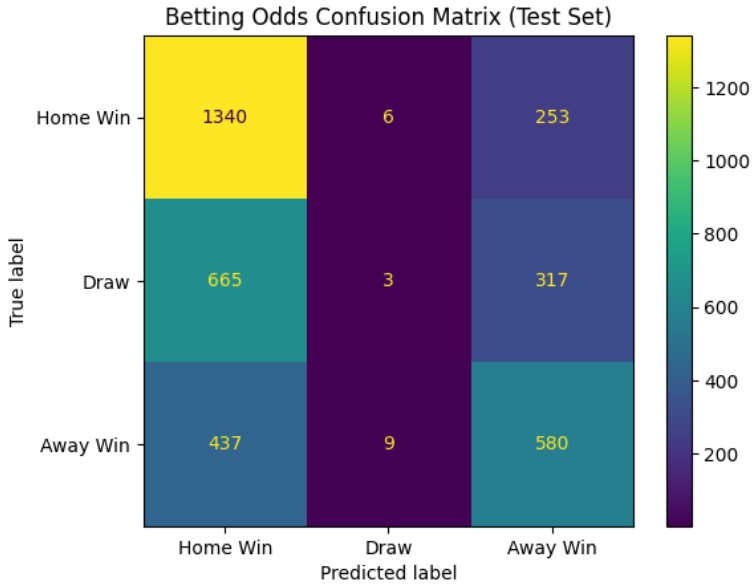

Figure 5 - Confusion Matrix for Bookmaker Odds

Whilst the overall accuracy of odds-based models may seem higher due to the increased frequency of home wins compared to other outcomes, the extremely low draw recall presents an opportunity which can be exploited by predictive models such as the hybrid system developed.

5.3.1 Long-term ROI

To evaluate the real-world utility of the model, a simulated betting strategy was applied using model predictions and bookmaker odds. For each match, the model's predicted outcome was paired with the corresponding implied bookmaker odds, and an expected value (EV) was calculated. Bets were placed only when the EV was positive, indicating that the model's estimated probability for an outcome exceeded the probability implied by the bookmaker's odds. A fixed stake of £1 was used for each qualifying bet. This simulation tracks the number of bets placed, the proportion of correct predictions, total money staked and returned, and overall return on investment (ROI). In addition, outcome-specific statistics (home win, draw, away win) are recorded to assess which types of predictions contributed most to profitability. This

method tests whether the model's probabilistic predictions can take advantage of the exploitable inefficiencies in betting markets.

Results of the simulation are shown in table 11.

Outcome	Bets Placed	Total Staked (£)	Total Returned (£)	ROI
Home Win	911	911.0	967.93	6.25%
Draw	1098	1098.0	1218.83	11.00%
Away Win	802	802.0	815.80	1.72%
Total	2811	2811.0	3002.56	6.81%

Table 11 - Results of Betting Simulation prior to Optimisation

As demonstrated, the model clearly identifies opportunities for profit, particularly within the draw outcome, making the most bets within this class.

5.3.2 Strategies to Maximise ROI

The arbitrage strategies used can be optimised to minimise risk and increase ROI by employing a similar grid search strategy to explore all EV and confidence thresholds, in order to find that which grants the best returns. 729 strategies were evaluated, with the optimal results of the search shown in table 12.

Metric	Value
draw_ev_thresh	0.02
draw_prob_thresh	0.35
home_ev_thresh	0.05
home_prob_thresh	0.6
away_ev_thresh	0.1
away_prob_thresh	0.6
roi	0.1229
net_profit	177.01
total_staked	1440
total_returned	1617.01

Table 12 - Result of Optimisation of Betting Strategy

The betting simulation was run again, with these values implemented, and the results are shown in table 13.

Outcome	Bets Placed	Total Staked (£)	Total Returned (£)	ROI
Home Win	388	388	421.54	8.64%
Draw	842	842	978.33	16.19%
Away Win	210	210	217.15	3.4%
Total	1440	1440	1617.01	12.29%

Table 13 - Results of Betting Simulation after Optimisation

As illustrated, this strategy places much fewer bets with a greater emphasis on draws, and yields an improve ROI of 12.29%. This is a substantial return and demonstrates the models effectiveness when partnered with optimal exploitative strategies.

5.4 Fulfilment of Success Criteria

Criterion	Achieved	Comments
The model executes within reasonable time on standard hardware	Yes	Model runtime is negligible, training run time
The model achieves superior accuracy to random guessing (> 33%)	Yes	Achieved overall accuracy of
The model has greater accuracy than pre-defined naïve models	Yes	See **5.2 – Comparison to Baseline Models**
The model clearly captures features of each outcome and its most common prediction for each result class matches said class	No	This is the case for Home and Away wins, however the former is overpredicted for the draw outcome.
The hybrid model outperforms a simple classification in draw recall	Yes	Gain of ~ 10% in draw recall
The model yields a positive non-negligible ROI when integrated into a targeted betting strategy (> 1%)	Yes	See **5.3.2 – Strategies to maximise ROI**

Table 14 - Success Criteria Fulfilment Table

5.5 Model Limitations and Future Scalability

5.5.1 Data Quality and the Nature of The Problem

The current hybrid neural network model demonstrates effective performance within the structural and data limitations inherent to football match prediction, however the model's performance is bounded by the quality and detail of available data. It relies primarily on publicly accessible datasets that include aggregated match-level features such as recent form, expected goals and implied win probabilities. While these inputs provide useful signals, they do not encompass the full complexity of the sport. Football matches are often influenced by player-specific factors, tactical systems, real-time events, and match-day conditions—variables not captured within the current feature space.

A key constraint in this regard is the high cost associated with more detailed datasets. Comprehensive football data that includes player-level metrics, positional tracking, physical performance indicators, and tactical dynamics is typically available only through commercial providers and at substantial cost. As a result, models developed in academic or resource-constrained environments such as this are limited to less granular data sources, which restricts the depth of representable information.

In addition to data constraints, the model must contend with the inherently unpredictable nature of football itself. Outcomes are often determined by rare or chaotic events, such as refereeing decisions, deflected goals or intangible factors affecting player performance that are difficult or impossible to model reliably. These low-frequency but high-impact incidents introduce noise into the outcome distribution, placing a natural ceiling on the predictive accuracy achievable, regardless of model complexity or data quality.

The current implementation also simplifies each match into a static vector of aggregated team-level inputs, abstracting away from important contextual factors. Elements such as player availability, specific tactical matchups and venue-specific effects are not explicitly modelled, and this is once again due to the limited resources available for data collection in this setting.

5.5.2 Class Separability

We can use PCA to demonstrate the class separability and gain a greater understanding of the challenges of this problem. PCA (Principal Component Analysis) is a dimensionality reduction technique that projects high-dimensional data onto a lower-dimensional space while preserving as much variance as possible. It helps reveal underlying structure in the data and is often used to visualize complex datasets like this one in two dimensions.

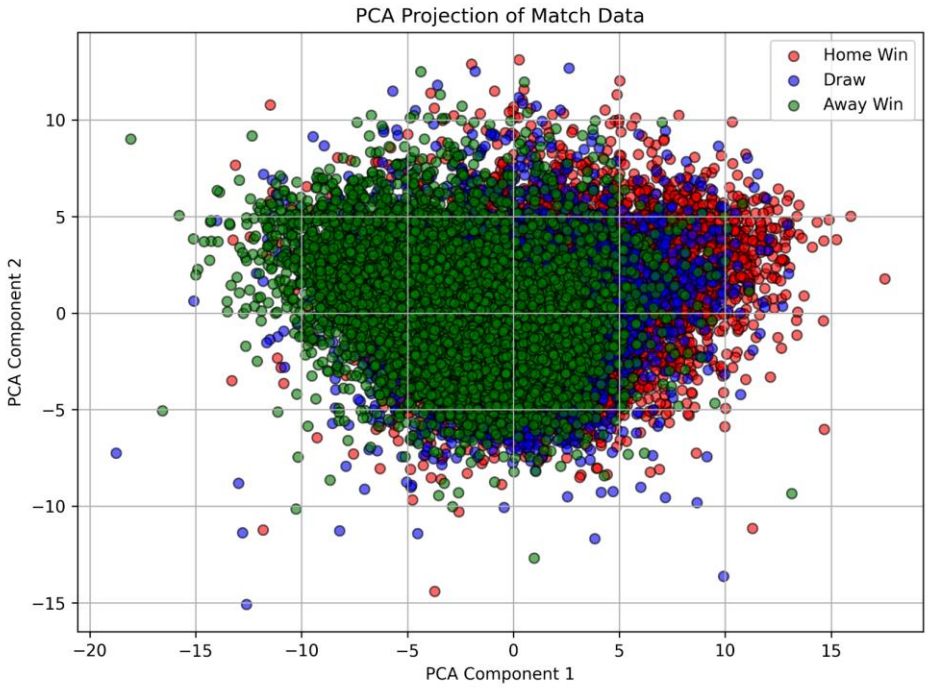

Figure 6 - PCA Projection of Data

The PCA plot in Figure 6 shows the class separability for the possible outcomes. There's significant overlap between the three result classes which suggests the model is working with feature distributions that often look very similar across outcomes. Draws in particular are tightly packed near the centre, which makes sense: they

usually occur when teams are evenly matched on metrics like xG or recent form. None of the classes occupy clearly distinct regions, resulting in no obvious boundary to separate them. This highlights that even in a simplified, low-dimensional projection, the data doesn't line up in a way that makes clean classification easy. It reinforces the need for nonlinear models and post-prediction logic—like the override mechanism used here—to deal with edge cases where the classifier alone might struggle.

5.5.3 Potential Scalability of the Approach

In a scenario where resource constraints are removed, the system could be scaled and enhanced to achieve greater predictive precision and adaptability. The most impactful improvement would involve integrating high-resolution, player-level data into the model architecture. Access to datasets capturing player positioning, sprint patterns, stamina, pressing intensity, and injury status would allow for dynamic modelling of match conditions. Evidence of such models already exists, as demonstrated by the data-driven successes of the likes of Brentford and Brighton.

6. Project Conclusion and Reflection

6.1 Project Evaluation

6.1.1 Objectives

Objective	Completed	Comments / Applicable Sections
Conduct a literature review and research to understand previous forecasting methods – their successes, failures and conclusions	Yes	Provided a great opportunity to further my knowledge on the specific subject and identify a purpose for this project. **2 – Literature Review**
Gather match data from credible sources to create a large dataset of fixtures from the relevant leagues and time period	Yes	Gathered using CSVs as the cost of these was significantly cheaper than industry standard APIs – see **5.5.1 - Data Quality and the Nature of The Problem**
Select relevant predictors for the model and process the match data in order to produce these predictors	Yes	Predictors were assessed and selected based on knowledge gained from the literature review and statistical analysis
Develop a regression neural network to predict the score difference between teams using the predictors as inputs, and evaluate performance	Yes	**4.5.2 - Regression Network**
Develop a probabilistic neural network to give	Yes	**4.5.3 – Classification Network**

predicted odds for each match outcome		
Develop the final network model using a combination of the previous networks and improvements to learning algorithms	Yes	**4.5.4 – Post-Prediction Logic**
Evaluate network performance by using standard statistics as well as comparisons to other models and betting markets	Yes	See **5 – Network Evaluation and Performance**
Discuss limitations of both the model and its future scalability	Yes	See **5.5 – Model Limitations and Future Scalability**

Table 15 - Project Objectives Reflection Table

6.1.2 Workflow

The water-scrum-fall style development allowed for a consistent but adaptive workflow, with sprints often varying in length but the implementation still being completed in the designated time. I believe the disciplined and consistent workflow was crucial to the completion of this project, as it allowed for experimentation without the risk of failing to achieve goals before their respective deadlines.

6.2 Personal Evaluation

The completion of this project represented a significant milestone in my personal and academic growth. Over the course of the project I developed and enhanced my skills in python, data pre-processing and neural network development, whilst deepening my understanding of machine learning and what is required for a successful predictive model. I also feel my ability to deal with unexpected setbacks was improved, having to adapt the model to a more ROI focused approach when it became apparent that extremely high accuracy was not a feasible goal with the resources available.

In addition to technical and analytical development, the project fostered the improvement of professional skills. The writing of a comprehensive investigative report was a new challenge to me, and one which I believe has improved my ability to evaluate evidence and structure arguments logically and concisely.

6.3 Project Conclusion

The motivation for this project stemmed from the widespread interest in football prediction and the limitations of traditional statistical models in capturing the game's chaotic and low-scoring nature. With billions wagered annually and a growing market for analytics-driven betting and performance tools, the project aimed to explore whether neural networks could provide a measurable edge in match result forecasting.

The system developed combines a classification and regression network, trained in parallel on time-aware, curated data. Post-prediction logic was introduced to override low-confidence outputs, particularly to improve recall on draws, historically the weakest prediction class for both bookmakers and models. This approach led to moderate predictive performance (50.39% accuracy), and more importantly, a demonstrable betting edge, with a simulated ROI of 12.29% under optimised conditions.

The project demonstrates that, even with constraints on data granularity and resources, it is possible to build models that outperform naive benchmarks and expose inefficiencies in market odds. The work also highlights that predictive success in football does not come from maximising accuracy alone, but from understanding where models can consistently find underpriced outcomes.

Although further progress will require more granular, player-level data and dynamic temporal modelling, this project successfully met its objectives and provides a solid foundation for future development in sports forecasting and analytics-driven arbitrage.

7. Bibliography

Baboota, R., & Kaur, H. (2019). Predictive analysis and modelling football results using machine learning approach for English Premier League. International Journal of Forecasting, 35(2), 741–755.

Chen, Y., Wang, Z., & Liu, H. (2019). Neural network of NBA MVP. Journal of Artificial Intelligence, 7(1), 1–10.

Flitman, A. (2006). Networks that predicted outcome: Australia. Journal of Applied Research on Industrial Engineering, 1(3), 162.

Hubáček, O., Šourek, G., & Železný, F. (2019). Learning to predict soccer results from relational data with gradient boosted trees. Machine Learning, 108(1), 29–47.

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

Joseph, A., Fenton, N. E., & Neil, M. (2006). Predicting football results using Bayesian nets and other machine learning techniques. Knowledge-Based Systems, 19(7), 544–553.

Lago-Peñas, C., & Dellal, A. (2010). Ball possession strategies in elite soccer according to the evolution of the match-score: The influence of situational variables. Journal of Human Kinetics, 25, 93–100.

Sujatha, K., Chandrasekaran, R. M., & Chandrasekar, C. (2018). Predicting football match results using transfer spending: Market inefficiencies in modelling. International Journal of Mathematical and Computational Methods.

Timmaraju, A. S., Palnitkar, A., & Khanna, V. (2013). Game on! Predicting English Premier League match outcomes. In Proceedings of the 2013 International Conference on Machine Learning Applications.

Tippett, M. (2019). xG Philosophy: Expected Goals in Football. Retrieved from https://xgphilosophy.com

Tuyls, K., Devlin, S., Bransen, L., & Gudmundsson, J. (2021). Game Plan: What can AI do for football and what football can do for AI. Journal of Artificial Intelligence Research, 71, 41–100.

Arabzad, S. M., Tayebi Araghi, M. E., Sadi-Nezhad, S., & Ghofrani, N. (2014). Football match results prediction using artificial neural networks: The case of Iran Pro League. Journal of Applied Research on Industrial Engineering, 1(3), 159–179.

FootyStats. (n.d.). Football Statistics & Analytics. Retrieved 20/12/24 from https://footystats.org

8. Appendix

All code used for this project has been submitted alongside the report, and will be available with the report when posted online.

Appendix A – Full dataset format for raw match data

Column Name	Description
timestamp	Unix Timestamp of the Kick Off Time
date_GMT	Kick Off Time in GMT (e.g., Feb 18 2017 - 9:30am)
status	Match status: incomplete / complete / postponed / cancelled
attendance	Number of attendees at the stadium (if available)
team_a_name	Home Team Name
team_b_name	Away Team Name
refree	Full name of the Referee (if available)
Pre-Match PPG (Home)	Points Per Game for Home Team - Pre Match
Pre-Match PPG (Away)	Points Per Game for Away Team - Pre Match
home_ppg	Points Per Game for Home Team - Current
away_ppg	Points Per Game for Away Team - Current
home_team_goal_count	Number of goals scored by the home team
away_team_goal_count	Number of goals scored by the away team
total_goal_count	Total Number of Goals
total_goals_at_half_time	Total Number of Goals at Half-Time

home_team_goal_count_half_time	Goals by home team at half-time
away_team_goal_count_half_time	Goals by away team at half-time
home_team_goal_timings	Minutes at which the goals were scored - Home Team (if available)
away_team_goal_timings	Minutes at which the goals were scored - Away Team (if available)
home_team_corner_count	Corner count - Home Team
away_team_corner_count	Corner count - Away Team
home_team_yellow_cards	Yellow cards - Home Team
home_team_red_cards	Red cards - Home Team
away_team_yellow_cards	Yellow cards - Away Team
away_team_red_cards	Red cards - Away Team
home_team_first_half_cards	First Half Cards - Home Team
away_team_first_half_cards	First Half Cards - Away Team
home_team_second_half_cards	Second Half Cards - Home Team
away_team_second_half_cards	Second Half Cards - Away Team
home_team_shots	Shots by Home Team
away_team_shots	Shots by Away Team
home_team_shots_on_target	Shots on target - Home Team
away_team_shots_on_target	Shots on target - Away Team
home_team_shots_off_target	Shots off target - Home Team
away_team_shots_off_target	Shots off target - Away Team
home_team_fouls	Fouls committed - Home Team
away_team_fouls	Fouls committed - Away Team
home_team_possession	Possession % - Home Team
away_team_possession	Possession % - Away Team
team_a_xg	Expected Goals - Home Team
team_b_xg	Expected Goals - Away Team
btts_percentage_pre_match	Pre-match average % for Both Teams to Score
over_15_percentage_pre_match	Pre-match average % for Over 1.5 Goals
over_25_percentage_pre_match	Pre-match average % for Over 2.5 Goals
over_35_percentage_pre_match	Pre-match average % for Over 3.5 Goals
over_45_percentage_pre_match	Pre-match average % for Over 4.5 Goals
odds_ft_home_team_win	Full Time Win Odds - Home Team
odds_ft_draw	Full Time Draw Odds
odds_ft_away_team_win	Full Time Win Odds - Away Team
odds_ft_over15	Odds - Over 1.5 Goals
odds_ft_over25	Odds - Over 2.5 Goals
odds_ft_over35	Odds - Over 3.5 Goals
odds_ft_over45	Odds - Over 4.5 Goals
odds_btts_yes	Odds – Both teams to score Yes
odds_btts_no	Odds – Both teams to score No
stadium_name	Name of the Stadium