UNIVERSITÀ DEGLI STUDI DI MILANO

FACOLTÀ DI SCIENZE E TECNOLOGIE

FACOLTÀ DI SCIENZE POLITICHE,
ECONOMICHE E SOCIALI

Master Degree in Data Science for Economics

Leveraging Match Statistics in Football Modeling: A Machine Learning Approach to Forecasting Shot Dominance

Advisor: prof. Silvia Salini

Co-advisor: prof. Nicolò Antonio Cesa-Bianchi

Thesis by
Antonino Pio Lupo
Student ID: 32896A

Academic Year 2024/2025

Contents

Chapter 1

Introduction

1.1 Background and Motivation

Football is widely acknowledged as the world's most popular sport, commanding unprecedented global attention and economic scale. FIFA estimates roughly five billion people are football fans worldwide (FIFA, 2020), and major events routinely draw massive audiences. The sport's economic footprint is similarly vast: top European leagues and clubs now generate record revenues, with Deloitte noting a €38billion market in 2023/24 (Deloitte, 2025).

In an age more and more dominated by the big data, data-driven analysis has revolutionized the manner in which football clubs strive to build competitive edge over rivals with strategy and scouting. Inspired by the "Moneyball" phenomenon in American baseball, clubs have been striving more and more to pursue gains through statistics in more recent years. Nowadays, companies such as Opta and Wyscout use cutting-edge tracking technologies to extract high-fidelity event data from each match and produce event logs and sell them to clubs, coaches, and analysts. Having the availability of fine-grained data (pass, shot, movement, etc.) brought with it a new "soccer analytics" element with an additional interest from academia and industry with intense enthusiasm towards empirical decision-making.

In the past two decades, extensive availability of dense football data has significantly helped to develop more advanced prediction models. Previous approaches were mostly grounded in statistical techniques, including the application of Poisson processes for modeling goal-scoring, but newer models have increasingly used more advanced methodologies, including regularization, rating techniques, and Bayesian models. Additionally, machine learning algorithms such as support vector machines, random forests, and neural networks have been used more and more in identifying patterns of game data and historical data to make predictions about game results and also to optimize betting strategies.

Parallel to these analytics advances, the global sports betting industry has expanded dramatically. Online betting platforms make it easy for fans around the world to place wagers on football. In addition to traditional bets on winners or total goals, a proliferation of "prop" markets now exists. Bettors can now wager on statistics such as total corners, yellow cards, or which team will take more shots. Recent research highlights that some of these niche markets are already quite popular: (Yip et al., 2023) observe that the "corner counts market is one of the most popular sidebet markets" offered by bookmakers, and they note that firms even build valuation frameworks for bets on total corners or goals. Because these newer markets have seen relatively little professional analysis and lower betting volume, they may be less efficient than the well-studied win/draw/lose market. Indeed, experts argue that ML models can exploit subtle inefficiencies in such betting markets (Galekwa et al., 2024). In short, the diversification of betting opportunities has created new avenues for data-driven prediction and value-seeking strategies.

1.2 Research Objectives

These converging trends—namely, the increasing availability and richness of football data, along with the expansion of betting markets—motivate the present thesis.

This study contends that prop bets represent an underexplored market segment with potential inefficiencies. In particular, shot-based markets are relatively new and have received less analytical attention compared to goal-based markets, suggesting they may offer value opportunities.

Accordingly, this thesis frames the task of predicting which team will take more total shots as a supervised classification problem. By leveraging historical match statistics and machine learning techniques, the aim is to address this gap and demonstrate how data-driven models can uncover value in this emerging area of sports betting.

1.3 Literature Review of Football Match Prediction

Football match forecasting has long been grounded in probabilistic models of scoring. Early work modeled goals using Poisson processes (Maher, 1982), which were refined by Dixon and Coles (Dixon & Coles, 1997) to include a bivariate adjustment capturing low-scoring outcomes. Modern studies confirm that such bivariate Poisson frameworks often yield strong predictive performance (Groll et al., 2015). These statistical approaches have traditionally targeted win/draw/loss probabilities and goal totals, laying the foundation for the sport's betting models. In recent years, a variety of machine learning techniques—from logistic regression and random forests to neural

networks—have been applied to similar tasks using team and match features. Notably, some works demonstrate modest betting profits; for example, Stübinger et al. (Stübinger et al., 2020) reported that an ensemble classifier achieved about a 1.58% return per wagered match. Collectively, these landmark studies have shaped today's predictive tools and betting strategies.

(Galekwa et al., 2024) give an extensive description of the use of machine learning in sports betting sport, elucidating existing practice in football prediction methods. They argue that contemporary systems use machine learning algorithms—e.g., support vector machines, random forests, and neural networks—on numerous sports, such as football, to handle past and up-to-date data for better predictive power. Under this category, machine learning facilitates enhanced forecasts and bookmaker odds calculation that can be modified, and data-driven bettors can capitalize on prevailing market flaws. The survey also identifies prevailing challenges, including data quality-related issues, the natural ambiguity of sporting events, and ethics around fairness. Of interest, (Galekwa et al., 2024) suggest future research into building adaptive, multimodal models that combine a variety of data sources with risk management strategies analogous to those in financial portfolio ones. Hence, the survey captures the most critical occurrences in football prediction of the betting type and captures the widespread requirement for using sound, evidence-based methods.

Alongside outcome prediction, recent work has begun to forecast specific match statistics. Wheatcroft (Wheatcroft, 2020a) showed that if one could know certain match metrics (like shots) in advance, outcome forecasts would be highly accurate. In practice, he introduced "Generalized Attacking Performance" (GAP) ratings to predict team shot totals before a match, demonstrating that these ratings add information beyond pre-game odds. Once again, Wheatcroft (Wheatcroft, 2020b) used GAP-based shot predictions to infer expected goals and match outcomes, yielding profitable over/under and win-probability forecasts. Empirical analyses also confirm the importance of shot-related metrics: for example, Castellano et al. (Castellano et al., 2012) found that winning teams average significantly more total shots (and shots on target) than losing teams.

In a related vein, Yip et al. (Yip et al., 2023) developed a Bayesian compound-Poisson regression explicitly to forecast the total number of corner kicks in a game. Despite these advances, dedicated ML models for predicting statistics like total shots, fouls, or cards remain relatively scarce.

Importantly, none of the above literature directly addresses our specific question: which team will take more total shots in a game.
Surveys and models to date have focused on goals, match outcomes, or aggregate stats such as corner counts, but not on comparative shot volumes. Indeed, shots forecast are only used to enrich the features space and never as a target variable. Moreover, this problem has always been treated as a regression problem, while in this work it

will be a classification task.

Therefore, predicting the team with higher shot count in a football match is not treated in past research. This gap highlights the novelty of our work.

Chapter 2

Data Description and Preprocessing

2.1 Data Sources

The first fundamental step in our research is the acquisition of relevant data. We draw upon two freely available public repositories—*Football-data.co.uk* and *Fbref.com*—which together provide both basic and advanced match statistics necessary for building a robust predictive model.

Football-data.co.uk offers historical datasets covering league competitions in 27 countries. Each match record includes essential event information such as:

- **Match details:** Home team, Away team, Date and kick-off time, Referee
- **Aggregate match statistics:** total goals, total shots, corners
- **Bookmakers' odds:** pre-match probabilities for 1X2 outcomes and Over/Under markets

While these basic statistics are valuable, the inherent complexity and randomness of football mean that goals, shots, and other summary counts alone are often insufficient for capturing nuanced performance patterns. To approach substantive predictive value—particularly for an offensive metric such as total shots—it is necessary to integrate more granular metrics that account for shot quality, passing dynamics, and chance creation.

To enrich the feature set, data has been collected from *Fbref.com*, a site powered by Opta that provides advanced per-match and per-player metrics. Unlike *Football-data.co.uk*, *Fbref.com* enables the collection of metrics that reveal subtleties hidden in ordinary statistics. For example, even if two teams may each register ten shots, their *expected goals* (xG) values might differ considerably—reflecting differences in shot location, angle, and context—enabling better predict future scoring performance.

The data drawn from *Fbref.com* fall into the following categories:

- **Event Information:**
  - *Matchweek, Date, Division:* Basic identifiers for contextualizing each fixture.
- **Standard Statistics:**
  - *HomeTeam, AwayTeam:* Team names.
  - *Goals:* Number of goals scored by each side.
  - *Shots:* Total attempts (on and off target).
  - *Shots on Target:* Attempts requiring a save or resulting in a goal.
  - *Free Kick Won:* Number of fouls earned (equivalent to free-kick won).
  - *Penalties Won:* Number of penalty kicks awarded.
- **Advanced Shooting Metrics:**
  - *Expected Goals (xG):* The probability that a given shot results in a goal, based on historical shot data including location, assist type, and build-up context.
  - *Average Shot Distance:* Mean distance from goal of all shots taken.
- **Advanced Passing Metrics:**
  - *Pass Completion :* Ratio of successful to attempted passes.
  - *Expected Assists (xA):* The probability that a given pass will become an assist, computed similarly to xG but for chances created.
  - *Key Passes:* Passes that directly lead to a shot.
  - *Passes into Final Third:* Entries into the attacking third.
  - *Penalty Area Passes (PPA):* Completed passes into the penalty area, excluding set pieces.
  - *Crosses into Penalty Area (CrsPA):* Completed crosses into the area, excluding set pieces.
  - *Progressive Passes:* Completed passes that advance the ball at least ten yards towards the opponent's goal or any completed pass into the penalty area, excluding passes from the defensive 40
- **Advanced Chance Creation Metrics:**

  - *Shot Creating Actions (SCA):* Two offensive actions directly leading to a shot (e.g. passes, dribbles, drawing fouls).
  - *Goal Creating Actions (GCA):* Two offensive actions directly leading to a goal.

- **Advanced Defensive Metrics:**
  - *Post-Shot Expected Goals (PSxG):* The xG value of a shot at the moment of contact, reflecting the quality of saved and missed opportunities.

We further integrate the following supplementary metrics from *Football-data.co.uk* to complement our *Fbref.com* data:

- *Half-Time Goals:* Goals scored by each team in the first half.
- *Corners:* Total corner kicks awarded.
- *Pre-Match Odds:* Bookmakers' implied probabilities for home win, draw, and away win.

The combined dataset covers the leading five European leagues (Premier League, Ligue 1, Bundesliga, La Liga, and Serie A) across the last eight seasons. This offers a sound foundation for building a forecasting model in the context of football analytics.

Notably, the use of certain performance metrics carries great informational content when predicting attacking actions, such as total shots. Additional metrics and insights may be accessed from the data already available. This analytical process will be revisited in greater detail in 2.3 *Additional Metric Generation.*

2.2 Data Integration and Cleaning

The final dataset was not the result of an automatic process, but rather the output of several intermediate steps, ranging from web scraping and data cleaning to merging. The R package `worldfootballR` has been used in order to extract the data from *FBref.* The package is extensively documented to scrape football data from various publicly accessible websites such as *FBref.com*, *Transfermarkt*, *Understat*, and *FotMob.*

The following steps are required to scrape, clean, and combine match-level statistics for multiple European leagues from *FBref* into a single, analysis-ready dataset. In more detail, the pipeline needs to:

- **Be exhaustive**: Never omit any match or team-level record when scraping or combining.

- **Be consistent**: Employ the same column names, data types, and team IDs between competitions and seasons.

- **Support integration**: Structure each interim table to facilitate joins on `Date`, `HomeTeam`, and `AwayTeam`.

The pipeline comprises three major phases: (1) scraping raw data; (2) standardized data transformations; and (3) dataset merging and export.

The pipeline is initiated with a clear and reproducible scraping methodology where a simple helper function builds the squad and match URLs for every league and season from a known list of team identifiers and season strings and subsequently invokes each such constructed URL using `worldfootballR::fb_team_match_log_stats()` across the major statistic domains (shooting, keeper, passing, and goal-creating actions), with occasional, programmatic pauses added in between to avoid overloading server limits; the systematic extraction is here programmed with the very specific aim of producing an entire set of raw tables while recording the provenance of each observation for completeness.

As soon as it has been pulled out, every raw table undergoes standard manipulations designed to prove consistent column types and identifiers: only relevant fields (e.g., `Team`, `ForAgainst`, `Date`, `Round` and the stat-specific metrics) are retained, and standard match view representation is enforced by splitting "For" vs. "Against" subsets, renaming metric columns through standard substitution to include `_home` or `_away`, and recombining later on a synthetic `MatchID`; in parallel all date strings are read in native `Date` objects and metric columns cast to numeric types so that subsequent statistical manipulations are safe and unambiguous.

Care over the quality of data is taken further by having an open, but conservative, policy about missing values and identifiers: any match record that has an incomplete set of metrics in all four stat types is discarded using `na.omit()` to leave the final analytic table completely filled out, whereas minor spelling differences in team names are normalized by conditional mapping (e.g. "Paris Saint Germain" vs. "Paris S-G") so team identifiers will be source-invariant.

Having stat tables standardized and cleaned, integration goes forward from a basis in fixture tables taken from *FBref* (retrieved through `worldfootballR::load_match_results()`) to which the keeper, passing, shooting, and GCA tables are added by series of left-joins on composite key (`Date`, `HomeTeam`, `AwayTeam`), preserving all matches from the *FBref* source, while appending available half-time statistics, corner counts, and odds from *Football-Data.co.uk*.

The join process ends with a final ranking of choosing a fixed column set (season IDs, fixture data, core statistics, and the `_home`/`_away` measures for each of the four stat categories) that forms the *FBref* backbone of the dataset. Simultaneously,

complementary match attribute values are ingested from *Football-Data.co.uk* by programmatically binning downloaded CSVs by league prefix, finding and reading them in-memory while enriching each row with an annotated `source_group`, parsing `Date` strings into native date objects, concatenating across files within each prefix and across prefixes to achieve a single metadata table, filtering and renaming only the fields of interest for analysis — division identifier, match date, teams, half-time goals (`HTHG`, `HTAG`), corner counts (`HC`, `AC`), and Bet365 odds (`B365H`, `B365D`, `B365A`) — to descriptive labels (e.g. `half_time_home_goals` and `odds_away_win`). A further harmonization step replaces minor team name variants (for example, "Paris SG" → "Paris S-G," "M'gladbach" → "M'Gladbach") based on a mapping dictionary which ensures that all team identifiers exactly match those produced by the *FBref* scraping pipeline.

The cleaned*Football-Data.co.uk* table is subsequently joined into the *FBref* base through a left-join on the combined key (`Date`, `HomeTeam`, `AwayTeam`). Any remaining observations with partially missing sets of these secondary variables are then dropped through `dropna()`, guaranteeing that the exported file contains only fully populated match observations.

The resulting dataset is now ready to be explored and manipulated in order to construct the features that will form the foundation of the predictive model, which represents the core objective of this research.

2.3 Additional Metric Generation

Feature engineering is essential for enhancing raw data, particularly when it comes to football match prediction modeling.

As a first step, a *match points* variable has been derived directly from the raw goal counts: if the home team's goals (`HomeGoals`) exceed those of the away team (`AwayGoals`), the home side is receives three points; on the other hand, if the away goals exceed home goals, the away side gets three points; a draw results in one point each.

Building on this, we compute two cumulative-season indicators: (i) total points accrued by each team up to a given matchday, and (ii) the corresponding league standing at that point. These season-to-date metrics provide a coarse proxy for global performance and consistency but are subject to biases which our simple ranking ignores.

Grounded in the expected goals framework, an *Expected Points* (xPoints) feature is introduced to capture nuances beyond the classical goal-based and points-based metrics. The xPoints estimate the average points a team would win if a certain fixture were played multiple times under the same statistical circumstances (Varotto,

2023). Formally,

$$x\text{Points} \;=\; 3\,P(\text{win}) \;+\; 1\,P(\text{draw}) \;+\; 0\,P(\text{loss}),$$

where the probabilities $P(\cdot)$ are obtained via Monte Carlo simulation, which yield probabilistic estimates by performing many randomized trials that capture uncertainty.

The simulation process goes as follows:

1. Calculate the shot-conversion probability for each team $p = \dfrac{\text{sum total } xG}{\text{total shots}}$. This total sums over all shot attempts, so ignoring spatial and situational shot heterogeneity.

2. For each simulated match, sample goals for home and away teams from Binomial distributions Binomial(n_{shots}, p).

3. Do it all again for $N = 10{,}000$ iterations, sacrificing computational tractability for sampling error.

4. Obtain $P(\text{win})$, $P(\text{draw})$, and $P(\text{loss})$ as relative frequencies of results by simulations.

Although the above method leverages the interpretability of binomial models and the randomness of Monte Carlo, it makes three basic assumptions: shot independence, even probability of conversion per shot, and that a single p for a team is enough. In practice, xG already adjusts for spatial and contextual parameters at the shot level, so finer-grained sampling—say, sampling per-shot Bernoulli trials with specific levels of xG—should have the effect of producing more refined distributions at the cost of increased complexity. This degree of simplification is thought to be consonant with the aspirations of this research.

The same reasoning has been applied to *post-shot expected goals* (PSxG), hence calculating an equivalent $x\text{Points}_{\text{PS}}$ embodying goalkeeping quality. The two-level approach permits attacks and defences making up match expectancy to be distinguished between one another, although at the cost of longer simulation time and possible overfitting if added naively to forecasting models.

Finally, we formalise our target variable `Result` as a categorical label taking values 1 (home more shots), X (draw), or 2 (away more shots).

To summarize the new metrics added to enhance predictive modeling:

Metric	**Definition**
Points_Home	Match points awarded to the home team
Points_Away	Match points awarded to the away team
Standings_Points_Home	Cumulative season points of the home team up to the matchday
Standings_Points_Away	Cumulative season points of the away team up to the matchday
Standings_Rank_Home	League ranking of the home team up to the matchday
Standings_Rank_Away	League ranking of the away team up to the matchday
xPoints_Home	Expected match points for the home team based on xG Monte Carlo simulation
xPoints_Away	Expected match points for the away team based on xG Monte Carlo simulation
xPoints_postxg_Home	Expected match points for the home team based on post-shot xG (PSxG)
xPoints_postxg_Away	Expected match points for the away team based on post-shot xG (PSxG)
Result	Categorical match outcome: 1 (home more shots), X (draw), 2 (away more shots)

Table 1: Summary of newly engineered features

2.4 Exploratory Data Analysis

The dataset employed in this analysis contains 13,813 match observations and 63 variables capturing team identifiers, match outcomes, performance metrics, standings, and betting odds.

A concise overview of its composition is as follows:

- **Number of records:** 13,813 (all unique).
- **Number of variables:** 63.
- **Missing data:** None
- **Variable types:**
  - *Date:* 1 column (`Date`, already parsed as `datetime`).
  - *Categorical (object/string):* 4 columns (`HomeTeam`, `AwayTeam`, `Div`, `Result`).
  - *Integer (`int64`):* 5 columns (`Points_Home`, `Points_Away`, `match_id`, `Standings_Points_Home`, `Standings_Points_Away`).
  - *Continuous (`float64`):* 53 columns (including basic counts such as `HomeGoals`, `Sh_Standard_home`, advanced metrics like `xG_home`, performance indices, and betting odds).

A completely populated dataset with a balance of rich numerical features and categorical identifiers is shown by this preliminary investigation. The wide range of floating-point measures provides a rich feature space for further exploratory visualization and the creation of machine-learning models.

The goal variable—the total number of shots made throughout a game—is the main focus of this exploratory data analysis, and it is quantified differently for home and away teams. Significant variety and the existence of extreme observations are revealed by a thorough numerical description of both distributions. The average number of shots for home teams is 13.55, with a standard deviation of 5.26, and ranges from 1 to 47. The median is 13, and the interquartile range is 10 (the 25th percentile) to 17 (the 75th percentile). As demonstrated by the maximum significantly above the 75th percentile and a non-negligible percentage of outliers beyond the whiskers in conventional box-plot identification, this distribution has a right-skewed tail.
On the other hand, away teams average 11.28 shots (standard deviation 4.69), with interquartile values ranging from 8 to 14 and values ranging from 1 to 39. The shot count is typically lower than for home teams, as confirmed by the median of 11.

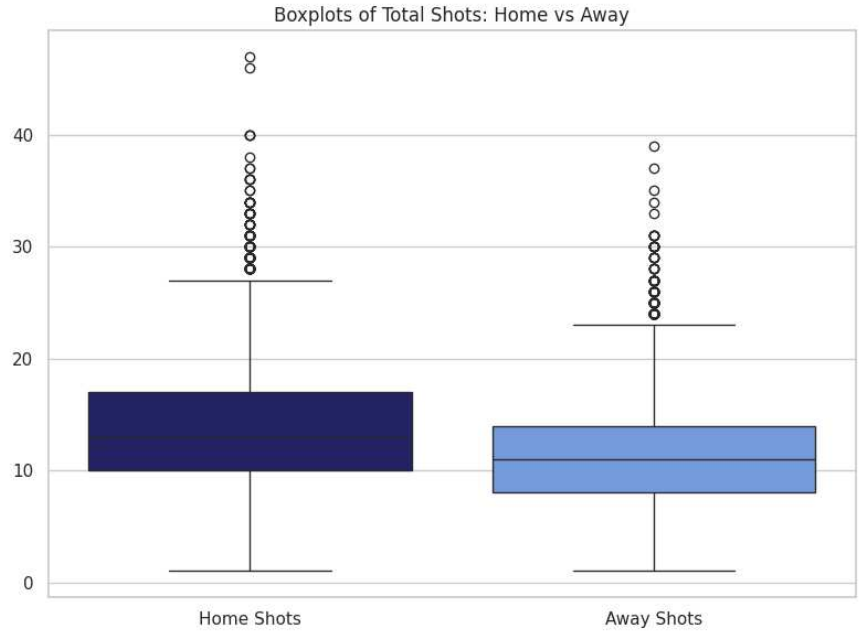

Figure 1: Total Shots Boxplot

While both distributions show wide ranges and the presence of outliers—suggesting that certain matches may involve exceptionally high offensive output due to factors like team mismatches or tactical strategies—these extreme values may not necessarily pose a significant issue in this context. Since the problem at hand is a classification task rather than a regression one, the presence of outliers in the target variable does not inherently undermine model performance.

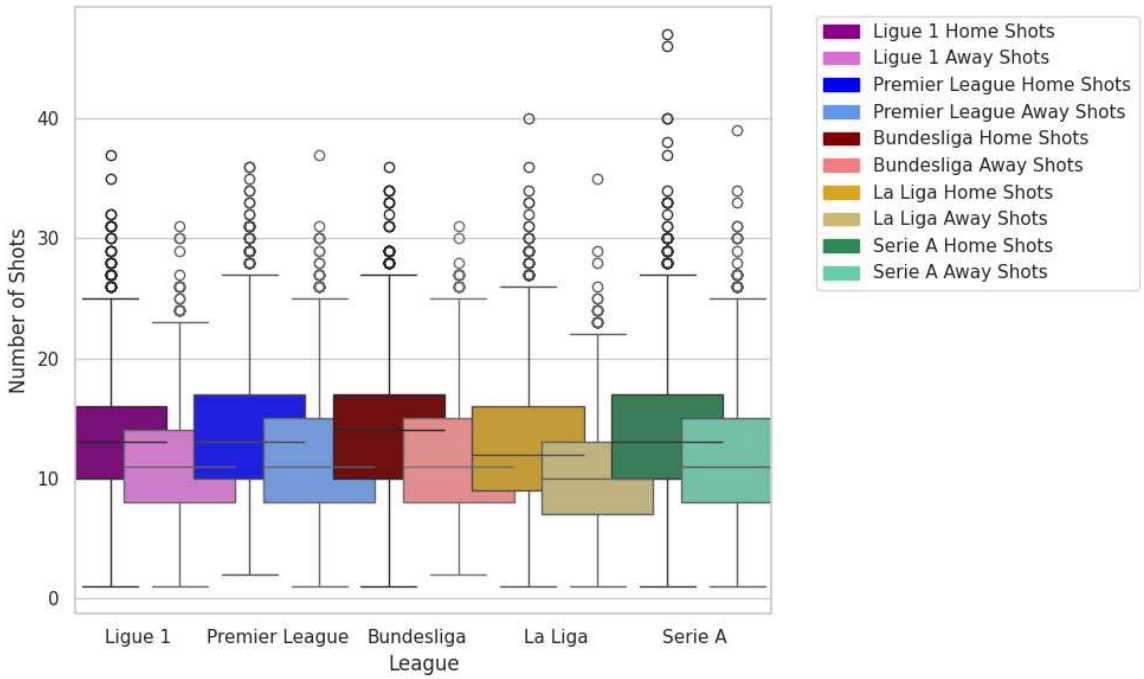

Figure 2: Total Shots Boxplots: Leagues Comparison

When disaggregated by league, box-plot comparisons reveal a consistent pattern: across all competitions examined, the central tendency and dispersion of shot counts are remarkably similar. This homogeneity suggests that, despite varying tactical cultures, pitch conditions, and competition levels, the fundamental shot-taking behavior remains stable at the aggregate level.

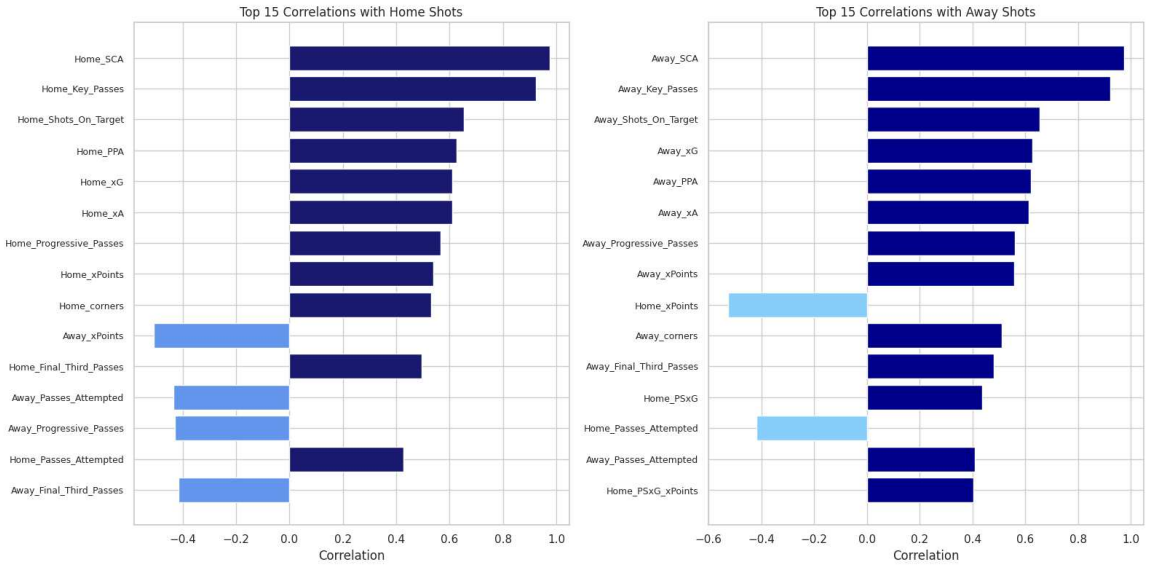

Figure 3: Top 15 Correlated Variables with Home and Away Shots

For what concerns correlated variables, the bivariate correlation analysis between shot counts and other match statistics offers insight into feature importance and selection prior to constructing the predictive model.
For home shots, the strongest positive correlates include Shot-Creating Actions (SCA; $r = 0.974$) and key passes ($r = 0.924$), followed by shots on target ($r = 0.654$), progressive passing metrics, and expected goals (xG; $r = 0.611$). Notably, away team expected points correlates negatively ($r = -0.509$) with home shots, hinting that dominant away teams tend to suppress home shooting volume.
Similarly, for away shots, SCA ($r = 0.973$) and key passes ($r = 0.922$) predominate, with expected away points ($r = 0.556$) reinforcing the link between team dominance and shot generation.

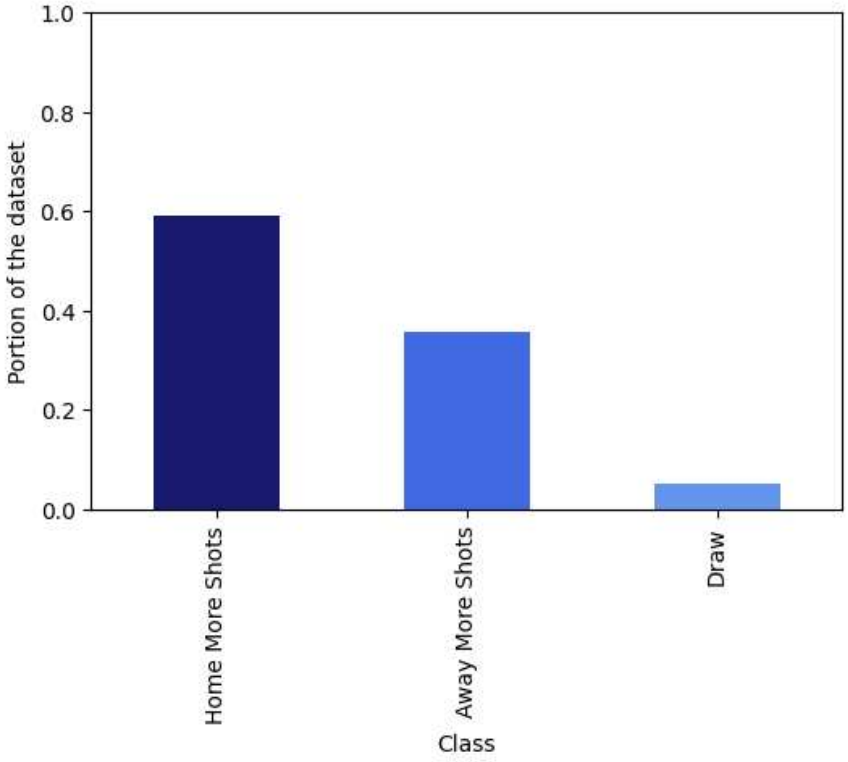

Figure 4: Class Distribution

Finally, the analysis of class imbalance in the categorical target—whether the home team, away team, or neither (equal shots) took more shots—reveals a pronounced skew. Home-dominant matches constitute approximately 60% of observations, away-dominant matches about 35%, and equal-shots matches only 5.25%. Such distributional imbalance poses significant challenges for multi-class classification algorithms, which may be biased toward majority classes and yield misleading overall accuracy. Moreover, the rarity of exact equality in shot counts, even lower than the typical draw frequency in 1X2 match-outcome studies, underscores the difficulty of confidently predicting this class.

The so-called home advantage is, hence, a salient and recurrent phenomenon: home teams consistently record a higher frequency of shots than their away counterparts. This effect—quantified here as a roughly 20% higher mean shot count for home teams—likely arises from multiple factors including crowd support, pitch familiarity, and favorable officiating. Its persistence across leagues and seasons underscores its importance as a structural bias in predictive modeling. If unaccounted for, home advantage may lead models to overpredict home team shot class.

The significance of the home-advantage effect was tested statistically by determining whether the differences between shot numbers for home and away teams have a normal distribution. Shapiro–Wilk test of the paired differences resulted in a statistic $W = 0.9978$ with $p = 3.40 \times 10^{-13}$. The null hypothesis of normality was rejected conclusively ($p \ll 0.05$), and thus parametric procedures are unsuitable for these data.

The median difference in shots was then tested for significant deviation from zero using a Wilcoxon signed-rank test, producing a statistic of $W = 2.9056 \times 10^7$ with an associated p-value of 4.59×10^{-223}, demonstrating a statistically significant advantage for home teams in terms of shot volume.

Statistical Test	**Test Statistics**	**P-Value**
Shapiro Test	0.998	3.402e-13
Wilcoxon Test	29055557.5	4.590e-223

Table 2: Shapiro Test and Wilcoxon Test Results

To investigate whether this advantage extends to other performance dimensions, match-level variables were grouped into five categories aforementioned: Standard Statistics, Shooting Metrics, Passing Metrics, Chance Creation, and Defensive Metrics. The mean values of each category for home and away teams were calculated.

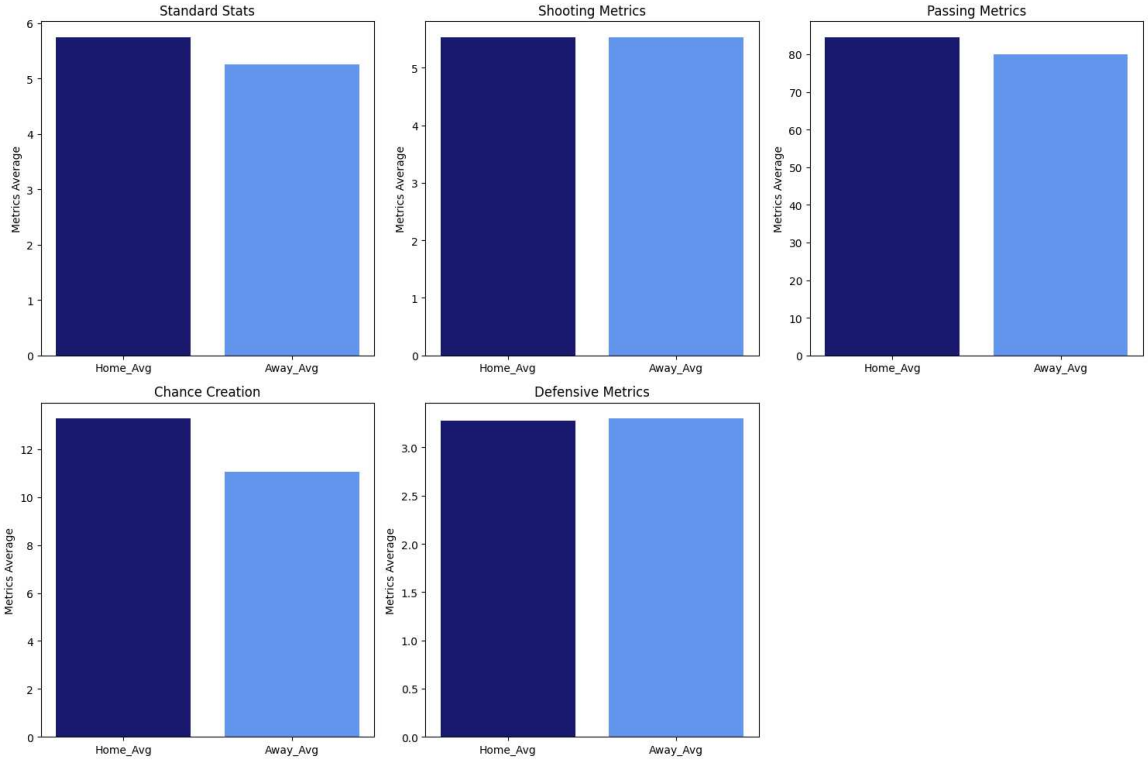

Figure 5: Home and Away Discrepancy in Stats Distribution

These results indicate that home teams exhibit modestly higher averages in Standard Statistics, Passing Metrics, and notably Chance Creation, whereas differences in Shooting Metrics and Defensive Metrics are negligible. The elevated Chance Creation values suggest that the home environment may foster more frequent goal-scoring opportunities beyond raw shot volume advantages.

To detect hidden patterns in match events and determine unique match identities based on shot dominance, the last phase of this exploratory study utilizes principal component analysis along with unsupervised clustering. In the initial phase of clustering, the dataset, standardized with 55 numerical features, was subjected to PCA. The data was then reduced to 24 principal components, preserving 90% of the total variance, in order to strike a balance between computational efficiency and informational integrity, Following this, the elbow method was employed to determine an ideal division, revealing four clusters that each depict a different type of match behavior.

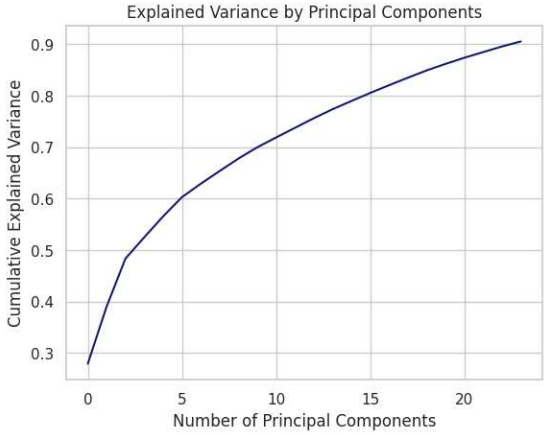

Figure 6: Explained variance of each principal component in PCA

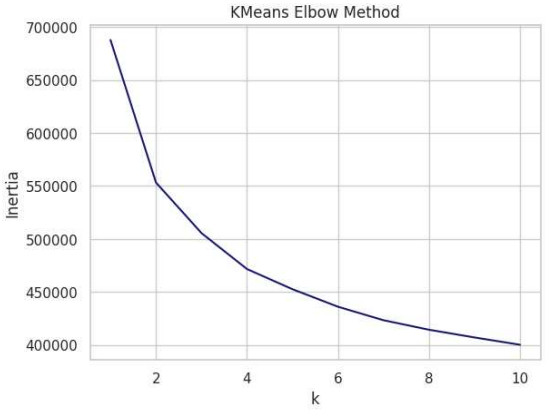

Figure 7: Elbow method to determine optimal number of clusters

A detailed examination of cluster-specific component scores shows the highest principal components in absolute mean magnitude.

Cluster 1 is defined by decent contribution from PC2, PC3, and PC4, indicating a relatively well-balanced profile: PC2 handles away-team goal building and defense play, PC3 mixes total shot pressure measurements, and PC4 handles passing accuracy and standings impact. The close-to-zero PC1 coefficients suggest that conventional measures of dominance are non-discriminative for this cluster. This outcome is consistent with the even partitioning of match shot results—i.e., superiority of shots by the home side (1), away superiority of shots (2), and draws (X).

Cluster 2 loads strongly negative on PC1 and has high positive scores on PC3 and PC4. PC1 describes largely superiority by the home team with high rates of shots, expected shots, and advanced possession values. Accordingly, the negative direction in this instance indicates an away-team shot generation trend supported by the high incidence of away shot dominant games within this cluster. The high PC3 component also confirms the away-team shot generation function, with signs for PC4's passing supporting away sides' tactical focus on possession of the ball.

Cluster 3 is the reverse of Cluster 2 for home-team advantage: high positive PC1 and PC3 indicate games in which the home team not only takes and has more shots but also puts consistent pressure on possession. The conjunction of high PC4 scores implies superior-shooting home teams add quality passing to sustain the possession control thesis.

Cluster 4 also has balanced results but is distinct from Cluster 1 in featuring a high positive PC2 and medium negative PC3. In this cluster, PC2 emphasizes away-team

cohesion and defensive solidarity, while the negative PC3 indicates fewer shot volume exchanges. Coupled with PC4's contrarian passing information, this cluster seems to recognize games where the home team accumulates more shooting totals without converting them into successful xG margins, leading to contentious outcomes.

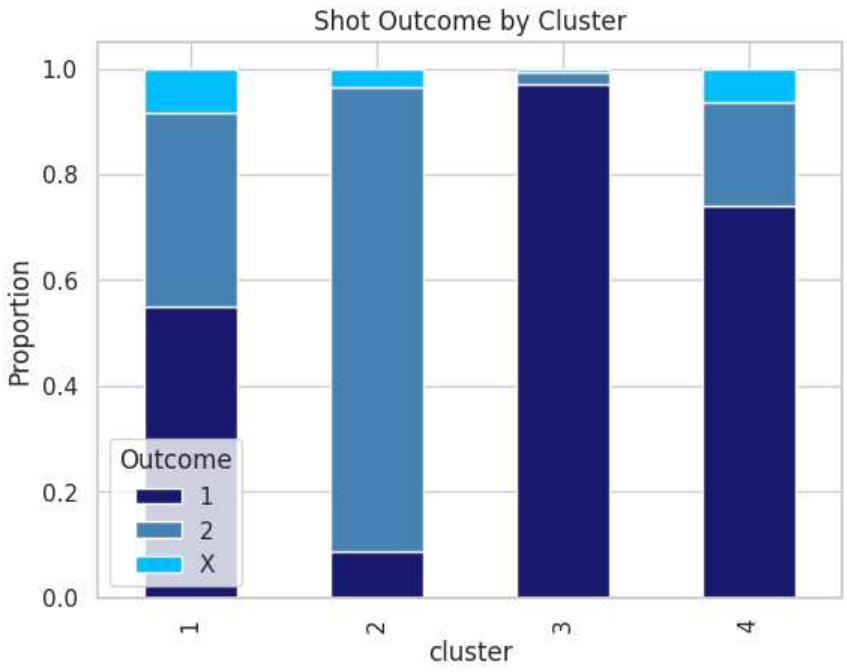

Figure 8: Shot Outcome by Cluster

Across all clusters, the distribution of principal components underscores two core dimensions: volume-based dominance (PC1 and PC3) and precision-oriented play (PC4). In conclusion, a taxonomy of four match identities has been identified by integrating PCA with k-means clustering, differentiating between balanced, home-dominant, and away-dominant matches. This framework may serve as an initial recognition for shot dominance patterns.

Chapter 3

Feature Engineering and Selection

3.1 Feature Construction

The efficacy of predictive models in soccer hinges critically on the design of features that capture team strengths and form prior to a fixture. Raw match statistics—such as goals scored, corners won, or shots on target—are inherently post hoc and thus unusable in a pre-match context. For instance, when forecasting the outcome of an upcoming match, data for goals and other in-game events are unavailable by definition. Consequently, a methodology for generating informative pre-match features that quantify both teams' strengths and current form, using only historical data preceding the target fixture, should be implemented.

In order to capture team form, football predictive modeling frequently uses historically aggregated match data. Traditional models like (Dixon & Coles, 1997) and (Angelini & De Angelis, 2017), for instance, fit Poisson regressions for goals using team attack and defense ratings derived from historical data. Rolling or moving-window features are conceptually comparable but adapt those ratings to recent form.

The primary difficulty in soccer outcome prediction, according to (Berrar et al., 2019), is producing useful pre-match features from temporally ordered match data (goals, teams, date, league, and season), which by their very nature lack clear predictive indications of future performance. Managing data sparsity at season beginnings (*start of season problem*), integrating different leagues/countries, accounting for performance variations, and contextual elements like opponent strength and venue effects are some of the main challenges. Importantly, features must offer pre-match insights based solely on historical data that is accessible prior to the target match, necessitating reliable techniques to convert unprocessed match results into predictive variables.

To this end, (Berrar et al., 2019) propose a domain-driven feature modeling framework centered on five dimensions derived from soccer knowledge: attacking performance (goal-scoring ability), defensive performance (goal prevention), recent performance (current form), strength of opposition (quality of past opponents), and home advantage. The framework's innovation lies in its two feature engineering methods. Recency feature extraction operationalizes these dimensions by aggregating statistics (e.g., goals scored/conceded, opponent goal difference, home/away flag) over a fixed window of recent matches. This explicitly tackles the pre-match requirement by using only data preceding the target fixture. Similarly, (Berrar et al., 2024) propose computing each feature from a team's n most recent games (using the mean or median of those matches).

Both (Berrar et al., 2019) and (Berrar et al., 2024) demonstrate that moving window-based aggregate metrics, although helpful for analyzing past performance, have a number of intrinsic limitations that may reduce the precision of predictive models. A significant issue is the *start of season problem*, where the lack of a sufficient number of games played prevents the calculation of reliable performance indicators at the beginning of a new season. This leads to a considerable reduction in usable training data and results in less robust features. Similarly, the *end of season problem* arises when the final matches of a league may not be fully competitive, as many outcomes (title, qualification for international competitions, or relegation) have already been decided. This puts into question the reliability of features derived from such matches.

This work adopts an approach analogous to the recency feature extraction method due to its relative simplicity and interpretability.
Specifically, for each numerical performance statistic and for each team, two rolling averages are computed over a window of the preceding t matches. For the home team, one rolling average captures the team's own performance (e.g., average expected goals, shots on target, or corners won), while the second measures the corresponding statistic conceded to visiting opponents in those fixtures. This dual averaging scheme yields a more holistic representation by concurrently encoding offensive and defensive tendencies at the specific venue. An identical procedure is applied symmetrically to the away team, enabling direct comparison of both teams' strengths under their respective contexts.

The *start of season problem* is addressed by computing the rolling average for early-season matches using historical data from the previous year's final matches. Although the features associated with this set of matches are less robust, the decision is justified by the following considerations: (i) the features become progressively more robust and accurate as the season advances; (ii) the lower quality of these features does not justify the significant data loss that would result from excluding all early-season matches within the selected time window.

With regard to the *end of season problem*, no variables are available to capture factors such as team motivation or mid-season changes. Nevertheless, it is considered that the features, computed over a window that is more indicative than that of the early season, may be regarded as reliable for the classification task and the dataset at hand.

An example of the feature-set construction is provided below. For the sake of simplicity and interpretability, only a single statistic—namely goals—will be considered.

Season	Matchweek	HomeTeam	AwayTeam	HomeGoals	AwayGoals	average_home_6_HomeGoals	average_home_6_AwayGoals
2025	21	Juventus	Milan	2	0	1.833	1.333
2025	23	Juventus	Empoli	4	1	2.000	1.333
2025	27	Juventus	Hellas Verona	2	0	2.333	1.167
2025	28	Juventus	Atalanta	0	4	2.333	1.167
2025	30	Juventus	Genoa	1	0	2.000	1.500
2025	32	Juventus	Lecce	2	1	1.833	1.167
2025	34	Juventus	Monza	2	0	1.833	1.000

Table 3: Rolling Averages for Juventus's Goals in the Last 6 Home Matches Prior to Matchweek 34 ($t = 6$)

In this instance, the match considered as an example (i.e., the match to be predicted) is Juventus vs. Monza, which took place during matchday 34 of the 2024/2025 Serie A season.

Four new features are generated in this context. As shown in the first table, *average_home_6_HomeGoals* represents the average number of goals scored by Juventus in their last six home matches (the choice of $t = 6$ is not essential to the purpose of the example, but will be discussed later in this section).
At the same time, *average_home_6_AwayGoals* denotes the average number of goals conceded by Juventus to their opponents over the same six preceding matches.

Season	Matchweek	HomeTeam	AwayTeam	HomeGoals	AwayGoals	average_away_6_HomeGoals	average_away_6_AwayGoals
2025	21	Bologna	Monza	3	1	1.333	1.167
2025	22	Genoa	Monza	2	0	1.833	0.833
2025	26	Roma	Monza	4	0	1.833	0.833
2025	28	Inter	Monza	3	2	2.333	0.667
2025	30	Cagliari	Monza	3	0	2.667	0.833
2025	32	Venezia	Monza	1	0	2.833	0.667
2025	34	Juventus	Monza	2	0	2.667	0.500

Table 4: Rolling Averages for Monza's Goals in the Last 6 Home Matches Prior to Matchweek 34 ($t = 6$)

A similar approach is applied to Juventus's opponent, Monza. The variable *average_away_6_AwayGoals* refers to the average number of goals scored by Monza in their last six away matches, while *average_away_6_HomeGoals* represents the average number of goals conceded by Monza in those same matches.

Thus, four new features are constructed that provide pre-match information regarding goal statistics. This same methodology is subsequently applied to the remaining match statistics.

Furthermore, to quantify contrast between the competitors, the model incorporates two differential features for each statistic. The offensive differential is defined as:

$$\text{Diff\_For} = \overline{S}^{(t)}_{\text{home}} - \overline{S}^{(t)}_{\text{away}}, \tag{1}$$

where $\overline{S}^{(t)}_{\text{team}}$ denotes the rolling average of statistic S over t prior matches for the specified team. Similarly, the defensive differential is given by:

$$\text{Diff\_Conceded} = \overline{C}^{(t)}_{\text{conceded\_home}} - \overline{C}^{(t)}_{\text{conceded\_away}}, \tag{2}$$

where $C^{(t)}_{\text{conceded\_team}}$ represents the average statistic conceded by the team over t matches. These differences capture the relative attacking and defensive capabilities of the two teams in a concise manner.

An example of the construction of these two new features is provided below, using data extracted from the previously discussed example.

Season	Matchweek	HomeTeam	AwayTeam	Diff_Goals	Diff_GoalsAgainst
2025	34	Juventus	Monza	1.333	-1.667

Table 5: Goals differential Features ($t = 6$) for Juventus vs. Monza, Matchweek 34 of the 2025 Season

(Berrar et al., 2019) examined competing approaches for figuring out the ideal number of previous matches (t) to inform predictions, each with unique implications. The choice of the window size t is a crucial question in rolling-average frameworks. They specifically specified t as a fixed window size for their recency feature extraction method.

They observed crucial trade-offs when they empirically assessed particular thresholds (t = 3, 6, 9, 12) on a subset of data: The *start of season problem* is made worse by smaller t (e.g., 3, 6) which increased feature relevance to current team form but decreased robustness due to limited data. A larger t (e.g., 12) increased robustness but ran the risk of incorporating out-of-date performances, which could have reduced predictive power. Additionally, computational cost increased dramatically with t. Their final choice, t=9, was justified as striking a balance between relevance and robustness, in line with "conventional soccer intuition" that 12 matches contained irrelevant data while 6 matches were insufficient.

Drawing on these insights, this thesis implements two window configurations. The first employs $t = 6$ to emphasize recent form and mitigate increased data loss. The

second adopts $t = 9$, paralleling (Berrar et al., 2019), to provide a broader performance perspective at the expense of potentially less representative form indicators and higher data loss. This dual-window strategy enables comparative analysis of model sensitivity to the form horizon and informs optimal window selection for predictive accuracy.

It is important to note that this rolling-average procedure applies exclusively to numerical match-derived features unavailable before the fixture. Exogenous variables, such as standing points, rank and bookmaker odds, remain unaffected by this methodology and are incorporated directly, given their inherent pre-match availability. The resulting feature set thus combines engineered rolling averages, differential statistics, and market-derived predictors, furnishing a robust and comprehensive basis for pre-match soccer outcome prediction.

This method generated a set of 152 features per match. The subsequent section will address feature selection techniques, which may prove essential for model simplification and the retention of only relevant features, thereby preventing potential *overfitting*. Regarding data loss, Table 6 estimates the volume of lost data corresponding to each window configuration employed.

Selected Window	Original Data Size	New Data Size	Data Loss
$t = 6$	13813	12598	1215
$t = 9$	13813	12005	**1808**

Table 6: Volume of data loss for $t = 6$ and $t = 9$

3.2 Feature Selection

Feature selection constitutes a critical procedure in which relevant variables are identified and retained for a given task, while superfluous ones are removed to improve classification accuracy, In studies involving large datasets, the dimensionality of the data must be reduced to allow the model to operate at its optimal level as well as perform effective computation. Given a dataset with 152 features, feature selection is critical in enhancing the quality of data and efficiency in classification techniques.

Decreasing the model's complexity, accelerating the learning process, and improving the efficiency of the classification are the key objectives of feature selection. By decreasing the variables, classification models can work more effectively and be more predictive in their predictions.

For the purposes of this work, the following feature selection techniques will be employed, as adopted in (Dip et al., 2024):

- *Analysis of Variance (ANOVA)*
- *Random Forest*

The study conducted by (Dip et al., 2024) is closely aligned with the objectives of the present thesis. Specifically, (Dip et al., 2024) aims to develop a predictive classifier using machine learning techniques to forecast match outcomes, leveraging a comprehensive set of features sourced from *Fbref.com*. Although the target variable differs, the correlation between the number of shots and the final result, along with the substantial overlap in the feature set, makes it possible to adopt a similar methodological framework. This approach, as demonstrated in (Dip et al., 2024), not only streamlined the models but also enhanced their predictive performance by reducing computational complexity and enabling more efficient processing.

Although the Pearson Correlation and Spearman's Rank methods were also used by (Dip et al., 2024), they were not included in the current analysis because they were incompatible with nominal variables. Because of the inherent comprehensiveness of the techniques already chosen and the desire to avoid adding more complexity to the training process, it was decided not to include any additional methods. An unmanageable number of model runs would be required due to this complexity, which could jeopardize the results' interpretability and robustness.

3.2.1 ANOVA

ANOVA (Analysis of Variance) is a parametric test that assesses whether the means of two or more groups are equal by comparing between-group to within-group variance. It computes an F-statistic by dividing the variance due to group differences by the variance due to within-group error:

$$F = \frac{MS_{\text{between}}}{MS_{\text{within}}} = \frac{SS_{\text{between}}/(k-1)}{SS_{\text{within}}/(N-k)},$$

where SS are sums of squares, k the number of groups, and N the total sample size. Under the null hypothesis of equal group means, F follows an F-distribution. ANOVA assumes independent observations, normally distributed residuals, and equal variances across groups (homoscedasticity) (Emerson, 2022). Violations (e.g., non-normality or unequal variance) can inflate Type I errors, so nonparametric alternatives (e.g., Kruskal–Wallis) are used when assumptions fail. ANOVA is known to be nearly optimal for balanced, normally distributed data (Hochkirchen, 2008).

Role in Feature Selection: In a classification context, ANOVA can serve as a univariate filter: each numeric feature is tested for mean differences across outcome classes. A large F (small p-value) indicates that feature values differ significantly between classes. Features are ranked by their F-statistics (or corresponding p-values)

and selected by a threshold or top-k scheme. Because ANOVA treats each feature independently, it does not inherently address multicollinearity; highly correlated features should be handled separately. Despite the presence of nonlinear relationships and connections within the dataset and the objectives of this study, using linear correlation methods are still widely applied for preliminary analyses due to their interpretability and computational efficiency.

3.2.2 Random Forest

Random forest is a nonparametric ensemble learning method that builds many decision trees and averages (or votes) their predictions. Each tree is trained on a bootstrap sample of the data and, at each split, a random subset of features is considered. For classification, each tree votes for a class and the forest predicts the majority. By aggregating over many trees, random forest reduces variance and overfitting compared to a single tree (Ho, 1995).

A common splitting criterion in each tree is the Gini impurity. For a node with class probabilities p_i, the Gini impurity is defined as

$$I_G = 1 - \sum_i p_i^2.$$

Splitting on a feature partitions the data and usually reduces impurity; the amount of impurity decrease indicates how well the feature separates the classes.

Role in Feature Selection: Random forests provide intrinsic feature importance scores. The standard measure is the mean decrease in impurity (MDI): each feature's importance is the total Gini-impurity reduction it produces, averaged over all trees (Breiman et al., 1984). Another measure is permutation importance: each feature's values are randomly permuted and the resulting increase in prediction error is recorded. Features causing larger impurity decreases or error increases are ranked as more important. Because random forests split data randomly, correlated features tend to share importance and may need post hoc redundancy filtering (Tolo̧si & Lengauer, 2011).

3.2.3 Feature Selection Strategy

Feature selection operations will be conducted as follows: All models used for training will initially be trained on the full set of features. Subsequently, the following filtering procedures will be applied:

ANOVA: only features with $p < 0.05$, and thus having a statistically significant impact on the target variable, will be selected. The training phase will then be

repeated while retaining *25%*, *50%*, and *75%* of the original features, respectively, prioritizing those with the highest *F-statistic* values.

Models based on decision trees are excluded from this training phase, as their procedure is described below.

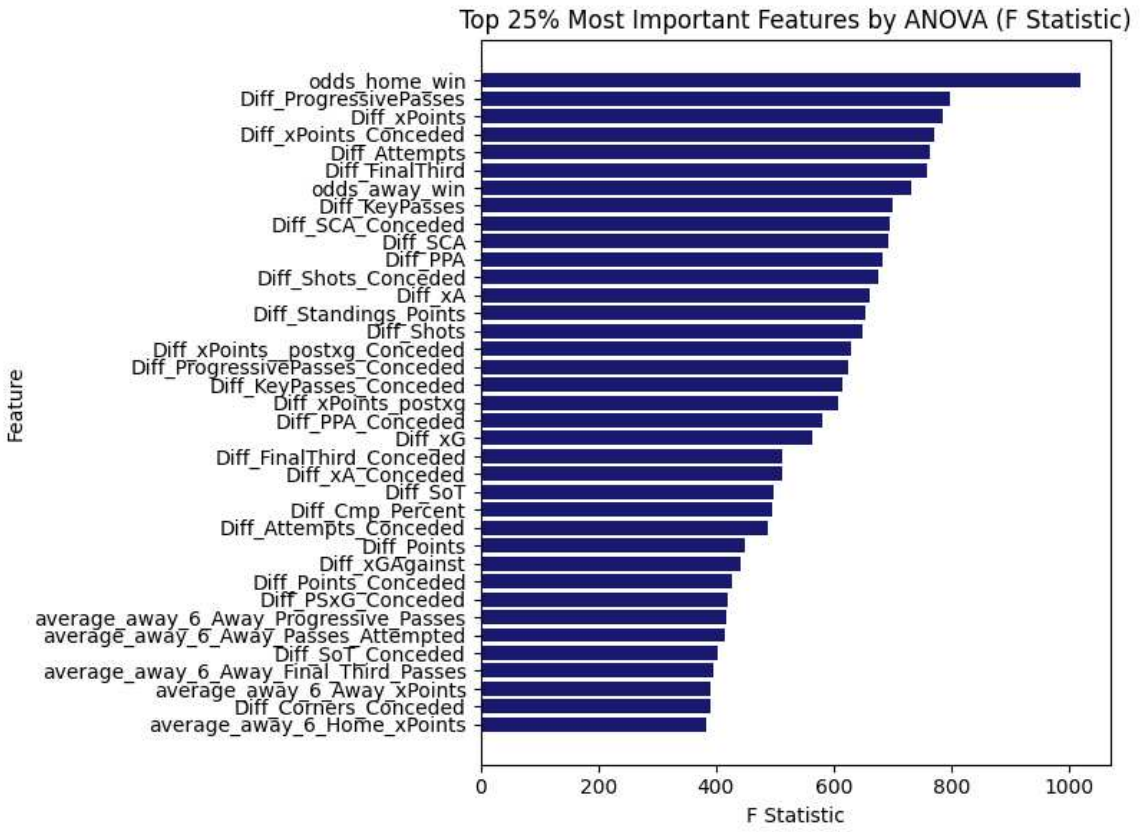

Figure 9: Top 25% features by ANOVA ($t = 6$)

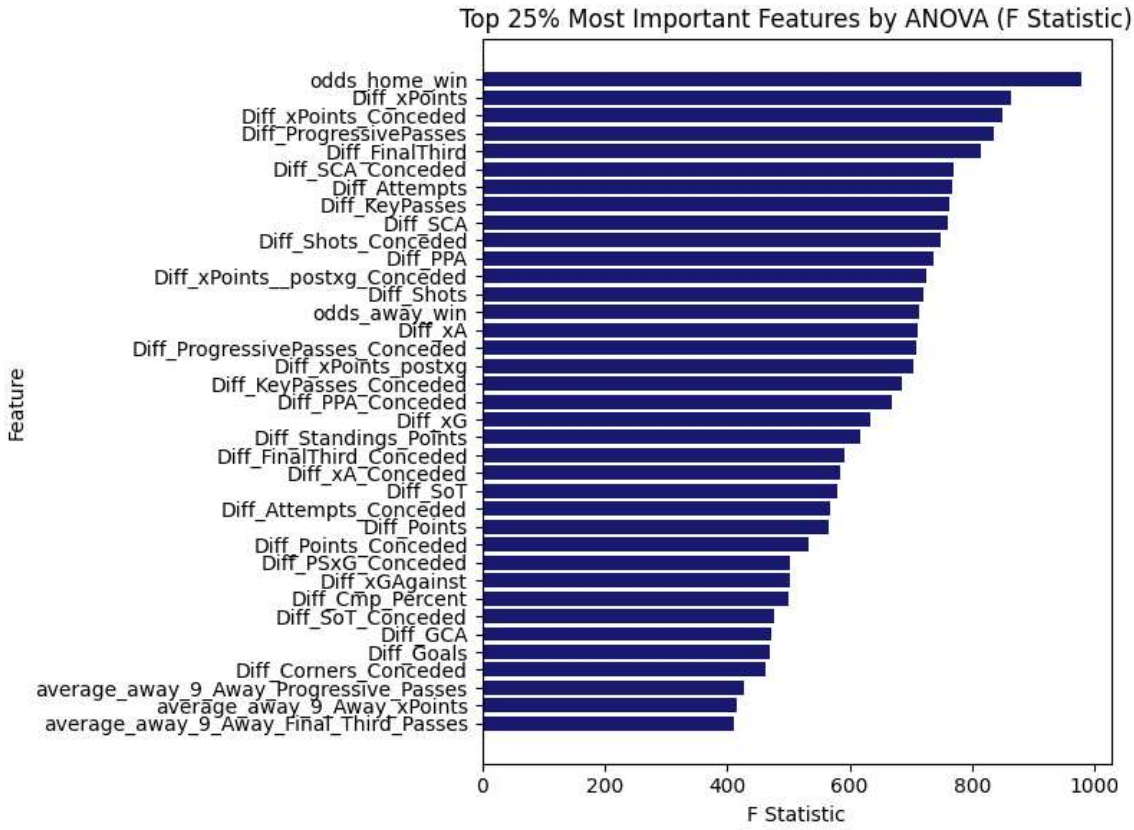

Figure 10: Top 25% features by ANOVA ($t = 9$)

Random Forest: the model will be trained using all available features. Afterwards, efforts will be made to reduce model complexity by eliminating all features exhibiting *near-zero importance*. Considering that the uniform importance of all features is equal to $\frac{1}{N}$, or approximately $\frac{1}{152} \approx 0.006$, a threshold of < 0.003—i.e., roughly half of the uniform value—will be applied. It should be noted that, should this cut-off fail to eliminate a sufficient number of features (i.e., at least 25%), a more stringent threshold would subsequently be applied.

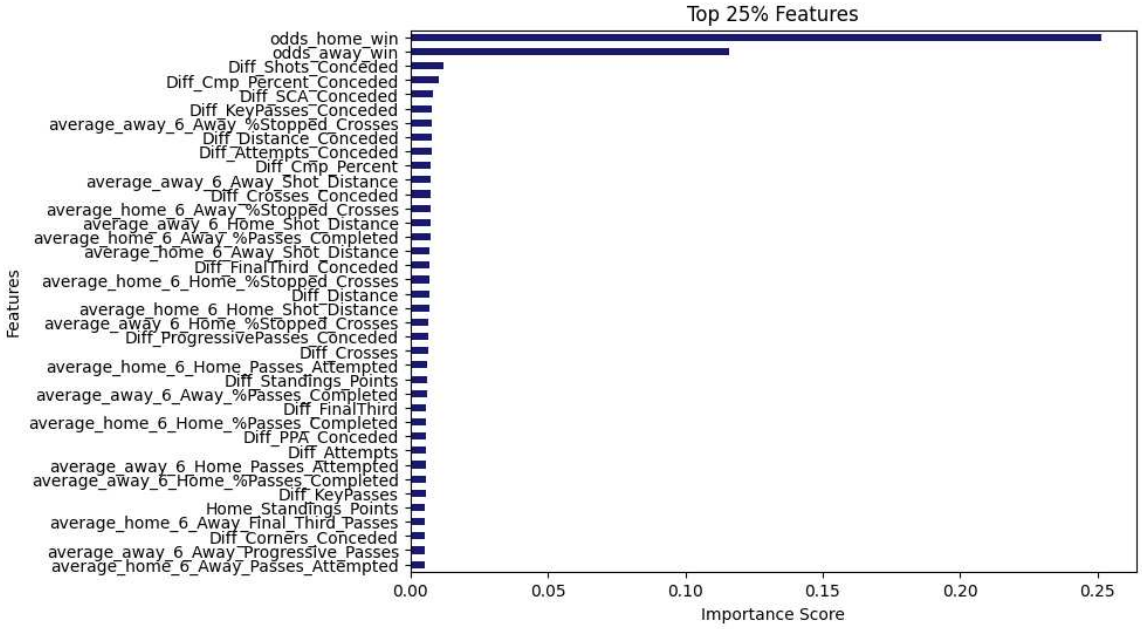

Figure 11: Top 25% Most Important Features from Random Forest ($t = 6$)

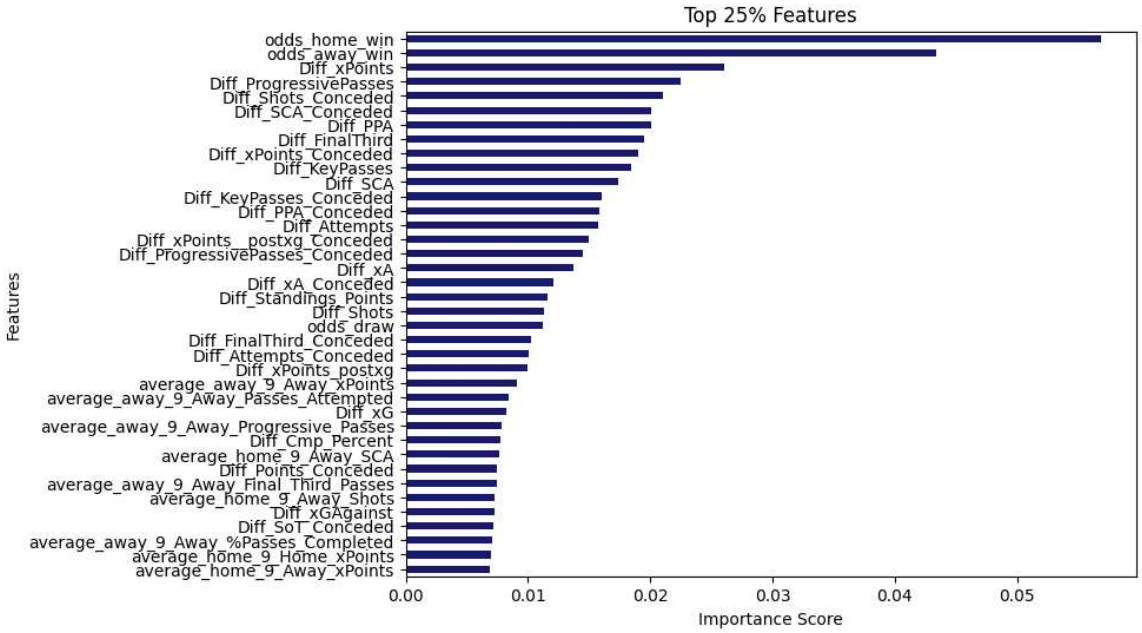

Figure 12: Top 25% Most Important Features from Random Forest ($t = 9$)

3.3 Data Augmentation

Data augmentation is a fundamental statistical technique in machine learning, designed to artificially expand the size and diversity of a training dataset. This is accomplished by either producing completely new synthetic data points or altered copies of preexisting data.
The main goals of this procedure are to lower the possibility of overfitting, improve the models' capacity for generalization, and raise overall prediction accuracy, especially when dealing with sparse or unbalanced datasets (Shorten & Khoshgoftaar, 2019).

There are numerous data augmentation methods, which can be broadly grouped according to how they alter or create data. *Transformations* entail changing preexisting data samples while maintaining their key attributes. Another popular method is *noise injection*, which adds random noise to the original data samples. By simulating real-world flaws and teaching models to ignore unimportant variations, this technique increases the models' resilience to noisy inputs. A more sophisticated category is represented by *Synthetic sampling* or *generation*. Methods such as the Synthetic Minority Over-sampling Technique (SMOTE), which targets minority classes in imbalanced datasets, interpolate between existing samples to produce new data points. More advanced techniques, like Generative Adversarial Networks (GANs), can effectively capture complex patterns and distributions within the data by producing completely new, realistic data points without depending on preexisting data.

In the section 2.4, the issue of class imbalance and the bias towards home shots was examined, thereby demonstrating the presence of the home advantage phenomenon. It was also discussed how this phenomenon could create difficulties in the training phase because it could result in machine learning models that are biased in favor of the majority class, which would then lead to poor predictive performance for the crucial minority classes.

Although feature engineering, choosing suitable machine learning models, and different performance metrics are all covered in great detail in the academic literature on football prediction, there are not many clear and in-depth discussions of data augmentation strategies designed to address class imbalance in the currently reviewed body of work.

To address extremely unbalanced output class labels, SMOTE has been implemented in football-related contexts. (Khamsan & Maskat, 2019) is a prime illustration. They successfully used a *multilayer SMOTE* approach in their study to balance a dataset in which the majority class was *Price Remain Unchanged*, while the significant minority classes were *Price Fall* and *Price Rise*. Even though it is not specifically for match results or shot predictions, this illustrates SMOTE's usefulness in a tabular dataset within a football-specific domain.

3.3.1 SMOTE

SMOTE is an oversampling algorithm specifically designed to mitigate the challenges posed by imbalanced datasets, where the minority class is significantly underrepresented (Chawla et al., 2002). It uses a methodical process that includes multiple crucial steps to create synthetic samples. The algorithm first determines which data points are members of the minority class. SMOTE then finds its k-nearest neighbors exclusively within the same minority class in the feature space for each minority instance that has been identified, represented by x_i. A typical value for k is 5 (Lauron & Pabico, 2016).

To create a synthetic sample, this algorithm randomly selects one of these k neighbors, denoted as x_R. A new synthetic instance (s) is then generated along the line segment connecting the original minority instance (x_i) and its chosen neighbor (x_R) (Lauron & Pabico, 2016). This interpolation is mathematically expressed by the formula (Blagus & Lusa, 2013):

$$s = x_i + \text{rand}(0, 1) \cdot (x_R - x_i),$$

where $\text{rand}(0, 1)$ represents a random number uniformly sampled between 0 and 1.

Continuous data is best suited for the SMOTE algorithm (Bystroński et al., 2025). This characteristic renders it directly applicable to the feature set, which consists predominantly of numerical variables.

The ability of SMOTE to produce synthetic samples while preserving fidelity to the underlying statistical relationships of the original data is one of its main strenghts. Unlike simple random oversampling, which merely duplicates existing samples, SMOTE generates synthetic samples by interpolating between existing instances, which ensures that the newly generated data points reside within the established feature space, thereby helping to preserve the intrinsic relationships and correlations between features (Bystroński et al., 2025).

For example, if a real shot from 10 yards has an xG value of 0.1 and another from 12 yards has an xG of 0.08, a synthetic shot interpolated between them will likely exhibit an xG value between 0.08 and 0.1, preserving the inverse relationship between distance and xG. This guarantees that the produced data points are plausible in addition to being numerically distinct.

3.3.2 SMOTE Limitations and Oversampling Strategy

While SMOTE effectively addresses the bias introduced by class imbalance, it inherently assumes a degree of linearity or smooth transition in the feature space for minority classes. For highly stochastic events in football, such as the specific outcome of *draw*, oversampling might generate synthetic samples that might not fully capture

the extreme rarity or the unique factors contributing to such events (Elreedy et al., 2023).
This implies a potential trade-off: SMOTE reduces bias and improves learning for general minority patterns, but extreme outliers or truly anomalous rare events might still present challenges. The synthetic *draw* generated might be "too regular" or "too similar" to each other, potentially leading to overfitting on these synthetic rare events rather than on the true, highly variable rare events.

Therefore, only the dominant represented class following *home more shots*, i.e., *away more shots*, will be influenced by the oversampling process, while the *draw* class remains unaltered.
This selective method is warranted as there is a fear of over-representing an inbuilt rare outcome that usually encompasses stochastic events. Overfitting on generated data, rather than offering improved generalization to actual, rare draws, can occur through the generation of too high a number of generated instances for this rare outcome, thereby introducing noise or spurious points. In accepting the unique and potentially more volatile character of *draw*, such a strategic option tries to improve the precision of the classification for the more common *home more shots* and *away more shots* outcomes.

Chapter 4

Modeling Methodology

A comprehensive examination of recent literature reveals distinct patterns in model selection, performance characteristics, and methodological approaches that inform the rationale for specific algorithmic choices in this domain.

Logistic Regression maintains significant relevance in football prediction literature primarily because of its interpretability advantages, despite potential limitations in capturing complex feature interactions.
In fact, in a high-dimensional space where nonlinear relations may have considerable influence on the eventual classification decision, its linearity assumption may turn out to be limiting. However, logistic regression may serve as a good performance baseline benchmark, an easy and understandable one against which more advanced models can be compared (Cranmer & Desmarais, 2016).

Random Forest has positioned itself as a strong benchmark for tabular sports data in its consistent appearance in football prediction studies. Tree-based methods have become standard with regards to sports prediction, (Bunker & Susnjak, 2022) reports, as Random Forest still beats all other approaches when paired with formalized match statistics.
The model is particularly well-adapted to football data with statistically high-dimensional numerics derived from rolling statistical averages due to its natural ability to work with mixed nature feature types and due to the fact that it is naturally overfitting-robust.

XGBoost has become a leading approach in modern football prediction research alongside Random Forest according to (Bunker & Susnjak, 2022).
The algorithm's computational efficiency together with its superior performance on tabular data makes it particularly suitable for shot prediction tasks. XGBoost handles the high-dimensional characteristics of the dataset through its integrated regularization features.

Neural Networks, represent the most complex end of the interpretability-accuracy

spectrum in football prediction research.
Artificial Neural Networks (ANNs) represent one of the most widely employed machine learning techniques in sports data analysis (Schumaker et al., 2010). A review of the literature reveals that numerous studies—particularly earlier ones—implemented ANNs as the sole predictive model, without conducting comparative evaluations against alternative algorithms. This tendency may have stemmed more from the availability of relevant software, tools, and algorithms at the time of research rather than from an inherent assumption of ANN superiority in predicting match outcomes in team sports (Bunker & Susnjak, 2022).

Indeed, real-world applications, and competitive ML environments (e.g., Kaggle competitions) do not support generalised claims of ANN dominance. The historical preference for ANNs in this domain is puzzling, particularly given the challenges associated with optimal parameter tuning and their propensity for overfitting—especially in contexts where datasets are relatively small, a common situation in sports analytics (Bunker & Susnjak, 2022).
Nonetheless, the models' capacity to capture intricate non-linear patterns in the feature set makes them theoretically well-suited for shot prediction tasks where subtle interactions between standard and advanced metrics may influence outcomes.

It should be noted that popular machine learning models, such as Support Vector Machines (SVM), were excluded from this process.
This decision is justified by the fact that the literature has proposed more performant alternatives, such as decision trees and boosting methods, and, above all, by their high computational resource requirements, which would render the training framework excessively slow. The following sections provide a theoretical background on the models selected for training.

4.1 Logistic Regression for Multiclass Classification

Multiclass logistic regression represents the direct extension of binary logistic regression for multiclass classification problems. Unlike binary classification, where observations are separated into two classes, multiclass logistic regression is used in scenarios where observations have to be classified into one of $K \geq 3$ mutually exclusive classes. The mathematical basis of multiclass logistic regression lies in the multinomial distribution and the softmax function, a higher-dimensional generalization of the logistic function where probabilistic interpretation is still ensured for classification problems. The method is among the core statistical methods of statistical learning and machine learning models where multistate categorical responses are to be predicted (Hastie et al., 2009). Even though more complex approaches will provide better predictive performance in specific instances, interpretability and theoretical validity of multiclass logistic regression render it especially useful for tasks where models are needed

to be understandable and baselines legitimate.

The theoretical framework of multiclass logistic regression builds upon the principle of maximum likelihood estimation and employs the softmax activation function to ensure that predicted class probabilities sum to unity. For a given input vector $\mathbf{x} \in \mathbb{R}^p$, the model computes the probability of belonging to class k among K possible classes through the softmax transformation:

$$P(Y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x} + b_k)}{\sum_{j=1}^{K} \exp(\mathbf{w}_j^T \mathbf{x} + b_j)}$$

where $\mathbf{w}_k \in \mathbb{R}^p$ represents the weight vector for class k, b_k denotes the bias term, and the denominator serves as a normalization factor ensuring $\sum_{k=1}^{K} P(Y = k|\mathbf{x}) = 1$. The softmax function, also known as the normalized exponential function, transforms the linear combinations $\mathbf{w}_k^T \mathbf{x} + b_k$ into valid probability distributions (Bishop, 2007).

To address the inherent identifiability issue in the multinomial model, where adding a constant to all linear predictors leaves the probabilities unchanged, one class is typically designated as the reference category. Setting the parameters for class K to zero ($\mathbf{w}_K = \mathbf{0}$ and $b_K = 0$), the probability expressions become:

$$P(Y = k|\mathbf{x}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x} + b_k)}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^T \mathbf{x} + b_j)}$$

for $k = 1, 2, \ldots, K-1$, and

$$P(Y = K|\mathbf{x}) = \frac{1}{1 + \sum_{j=1}^{K-1} \exp(\mathbf{w}_j^T \mathbf{x} + b_j)}$$

The model parameters are estimated by maximizing the log-likelihood function, which for n training examples takes the form:

$$\ell(\mathbf{W}, \mathbf{b}) = \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log P(Y = k|\mathbf{x}_i)$$

where y_{ik} is a binary indicator variable equal to 1 if observation i belongs to class k and 0 otherwise, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_{K-1}]$ represents the weight matrix, and $\mathbf{b} = [b_1, b_2, \ldots, b_{K-1}]^T$ contains the bias terms.

The optimization objective can equivalently be expressed as minimizing the cross-entropy loss function:

$$L(\mathbf{W}, \mathbf{b}) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log P(Y = k|\mathbf{x}_i)$$

This loss function penalizes predictions that deviate from the true class labels, with the logarithmic penalty becoming increasingly severe as the predicted probability for the correct class approaches zero.

The distinction between binary and multiclass logistic regression extends beyond the mathematical formulation to encompass fundamental differences in model complexity, interpretation, and computational requirements. Binary logistic regression employs a single set of parameters to separate two classes, while multiclass logistic regression requires $K-1$ parameter vectors to distinguish among K classes. The decision boundary in binary classification is characterized by a single hyperplane, whereas multiclass problems involve multiple decision boundaries that partition the feature space into K regions.

Aspect	**Binary Logistic Regression**	**Multiclass Logistic Regression**
Number of Classes	2	$K \geq 3$
Activation Function	Sigmoid: $\sigma(z) = \frac{1}{1+e^{-z}}$	Softmax: $\frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}}$
Parameters	Single weight vector $\mathbf{w}$	$K-1$ weight vectors $\{\mathbf{w}_k\}$
Decision Boundaries	Single hyperplane	Multiple hyperplanes
Output Interpretation	Probability of positive class	Probability distribution over classes

4.2 Random Forest

Random Forests for classification are ensemble classifiers formed by aggregating a large collection of randomized decision trees. In Breiman's original formulation a random forest is "a classifier consisting of a collection of tree-structured classifiers $\{h(x, \tilde{\theta}_k), k = 1, \dots\}$ where the $\{\tilde{\theta}_k\}$ are independent identically distributed random vectors and each tree casts a unit vote for the most popular class at input x" (Breiman, 2001).

The ensemble decision is based on voting. For an ensemble of classifiers $h_1, \dots, h_K$ the margin function is defined as

$$\mathrm{mg}(X, Y) = \mathrm{av}_k \, \mathbb{I}\big(h_k(X) = Y\big) \; - \; \max_{j \neq Y} \mathrm{av}_k \, \mathbb{I}\big(h_k(X) = j\big),$$

where $\mathbb{I}(\cdot)$ is the indicator and av_k denotes the average over trees. The generalization error of the ensemble is

$$\mathrm{PE}^* = P_{X,Y}\big(\mathrm{mg}(X, Y) < 0\big),$$

i.e., the probability that the average vote for the true class is not larger than the best alternative (Breiman, 2001).

Breiman gives a probabilistic limit result showing that, as the number of trees grows, the ensemble error converges (almost surely) to the probability obtained when each tree is drawn according to the tree-generating random vector distribution:

$$\mathrm{PE}^* \xrightarrow[K\to\infty]{} P_{X,Y}\Big(P_{\tilde{\theta}}\big(h(X,\tilde{\theta}) = Y\big) - \max_{j\neq Y} P_{\tilde{\theta}}\big(h(X,\tilde{\theta}) = j\big) < 0 \Big).$$

This theorem (via the Strong Law of Large Numbers) explains why adding trees does not lead to overfitting in random forests (Breiman, 2001).

To characterize accuracy Breiman introduces the forest margin $mr(X,Y)$ and the ensemble strength

$$mr(X,Y) = P_{\tilde{\theta}}\big(h(X,\tilde{\theta}) = Y\big) - \max_{j\neq Y} P_{\tilde{\theta}}\big(h(X,\tilde{\theta}) = j\big), \qquad s = E_{X,Y}\big[mr(X,Y)\big],$$

and shows that the generalization error can be bounded in terms of the strength s and an average raw-margin correlation $\bar{\rho}$. Using Chebyshev and algebraic identities he derives the upper bound

$$\mathrm{PE}^* \leq \bar{\rho}\,\frac{1-s^2}{s^2},$$

so that lower correlation between tree raw margins and higher individual tree strength lead to a better (smaller) bound on generalization error (Breiman, 2001).

Algorithmically Breiman's random forest combines two key sources of randomness. First, each tree is grown on a bootstrap sample (bagging): a training set of size N is sampled with replacement to produce each tree's construction set. Second, at each internal node a random subset of the input variables (features) is selected and the best split is chosen from that subset; denote the number of randomly chosen candidate features by F (often noted in implementations as `mtry`). Trees are grown to maximum size (unpruned) and the ensemble decision for an input x is the plurality (majority) vote over trees, equivalently the class maximizing $P_{\tilde{\theta}}(h(x,\tilde{\theta}) = j)$ (Breiman, 2001).

4.3 XGBoost

XGBoost (eXtreme Gradient Boosting) is a boosting system for trees that is optimized with the purpose of efficiency and performance in machine learning problems, notably classification. It integrates gradient boosting with sophisticated regularization to attain state-of-the-art performance. The flexibility of the system regarding tuning and handling large data ensure that it represents one the most popular model for classification tasks (Chen & Guestrin, 2016).

XGBoost is an ensemble learning method that builds a predictive model by combining multiple weak learners (decision trees) in an additive manner. For a given dataset with n examples and m features $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$, where $\mathbf{x}_i \in \mathbb{R}^m$ and $y_i \in \mathbb{R}$, the model predicts the output using K additive functions:

$$\hat{y}_i = \phi(\mathbf{x}_i) = \sum_{k=1}^{K} f_k(\mathbf{x}_i), \quad f_k \in \mathcal{F}, \tag{3}$$

where $\mathcal{F}$ is the space of regression trees. Each tree f_k maps an example to a leaf index $q(\mathbf{x})$ and assigns a weight $w_{q(\mathbf{x})}$ to that leaf. The model's objective function includes a loss term and a regularization term to prevent overfitting:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \tag{4}$$

where $\Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$. Here, l is a differentiable convex loss function (e.g., logistic loss for classification), γ controls the complexity penalty for the number of leaves T, and λ regularizes the leaf weights w (Chen & Guestrin, 2016).

The training process involves greedily adding trees that minimize the objective. At each iteration t, the optimal weight w_j^* for leaf j is computed as:

$$w_j^* = -\frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}, \tag{5}$$

where g_i and h_i are the first and second-order gradients of the loss function. The quality of a tree structure q is evaluated using the scoring function:

$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T. \tag{6}$$

4.4 Neural Networks

Neural networks are a broad and complex class of predictive models applicable to tasks such as categorization, regression, and sequence prediction, where they are constructed through the composition of simple predictors of the form $g(x) = \sigma(w^\top x)$, with $\sigma : \mathbb{R} \to \mathbb{R}$ denoting a nonlinear activation function.
In the context of classification, the network learns a mapping from an input space $\mathbb{R}^d$ to an output space $\mathbb{R}^n$, where the output is subsequently interpreted as a class label or a probability distribution over classes.

This work focuses on feedforward neural networks, in which the computation follows a directed acyclic graph $G = (V, E)$. The node set V is partitioned into

the input layer V_{in}, hidden layers V_{hid}, and output layer V_{out}, with edges $(i,j) \in E$ parameterized by weights $w_{i,j} \in \mathbb{R}$. Given a weight matrix W and an activation function σ, the network implements a function $f_{G,W,\sigma} : \mathbb{R}^d \to \mathbb{R}^n$.

For an input $x \in \mathbb{R}^d$, the computation proceeds as follows: each input node $i \in V_{\text{in}}$ takes the value $v_i = x_i$; each non-input node $j \in V \setminus V_{\text{in}}$ computes

$$v_j = \sigma\big(w(j)^\top v(j)\big),$$

where $w(j)$ and $v(j)$ are the vectors of incoming weights and corresponding source node values, respectively. The network output is then defined by $f_k(x) = v_k$ for every output node k. It is the mechanism behind classification since it allows the network to map raw inputs into representable forms which can be linearly separable in the output space.

The activation function σ plays a central role in classification. Two examples are the sigmoid $\sigma(z) = \frac{1}{1+e^{-z}}$ and the leaky ReLU $\sigma(z) = (\alpha \mathbf{I}\{z < 0\} + \mathbf{I}\{z \geq 0\})z$, with α typically of order 10^{-2}.

From a representational standpoint, feedforward networks with a single hidden layer and an appropriate activation can compute any binary classifier $f : \{-1,1\}^d \to \{-1,1\}$, given sufficiently many hidden units. For real-valued inputs in $[-1,1]^d$, such architectures can approximate any Lipschitz function to arbitrary precision, albeit sometimes requiring an exponentially large hidden layer in d. These universality results, while theoretically powerful, highlight that the network size is a crucial parameter in practice.

The training of neural networks for classification is typically cast as an empirical risk minimization (ERM) problem over a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$ with a chosen loss function $\ell(y, \hat{y})$. In classification, the loss often measures the discrepancy between the true label y and the predicted output $\hat{y} = f_{G,W,\sigma}(x)$.

The optimization is performed using stochastic gradient descent (SGD):

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{\partial \ell_{Z_t}(W)}{\partial w_{i,j}}, \quad (i,j) \in E,$$

where η_t is the learning rate and Z_t indexes a random training example. In the minibatch variant,

$$w_{i,j} \leftarrow w_{i,j} - \eta_t \frac{1}{|S_t|} \sum_{s \in S_t} \frac{\partial \ell_s(W)}{\partial w_{i,j}},$$

where S_t is a random subset of training samples. Because the training error is a nonconvex function of W, these methods generally find local minima without guarantees of global optimality.

The computation of gradients in multilayer networks is efficiently implemented via

the backpropagation algorithm, which applies the chain rule for composite functions:

$$\frac{df(g(x))}{dx} = \frac{df(g)}{dg} \cdot \frac{dg(x)}{dx}.$$

For a single-output network with the sigmoidal activation σ and square loss, let node 0 denote the output. Writing $s_0 = w^{(0)\top} v^{(0)}$ and $v_0 = \sigma(s_0)$, the derivative for a weight $(i, 0) \in E$ is

$$\frac{\partial \ell_t(W)}{\partial w_{i,0}} = \ell'(v_0)\, \sigma'(s_0)\, v_i.$$

For a weight (i, j) connecting to the second-to-last layer,

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \ell'(v_0)\, \sigma'(s_0)\, w_{j,0}\, \sigma'(s_j)\, v_i.$$

In deeper layers, the recursive dependency of errors is handled by defining

$$\delta_j = \begin{cases} \ell'(v_0)\, \sigma'(s_0), & j = 0, \\ \sigma'(s_j) \sum_{k:(j,k)\in E} \delta_k\, w_{j,k}, & \text{otherwise}, \end{cases}$$

so that for any $(i, j) \in E$,

$$\frac{\partial \ell_t(W)}{\partial w_{i,j}} = \delta_j\, v_i.$$

This formulation makes explicit the contribution of each node's activation and incoming value to the gradient update.

4.5 Model Training Protocol

4.5.1 Methodology

Given the preprocessed input data described in Chapter 2, two distinct feature sets are generated as outlined in the section 3.1, employing rolling windows of $t = 6$ and $t = 9$. The resulting feature matrices are subsequently divided into training, validation, and test subsets to preserve an untouched holdout set for final assessment. The validation split is allocated to hyperparameter tuning and early stopping procedures, while the test portion is strictly reserved for definitive evaluation.

Baseline experiments maintain the native class distribution to establish reference performance. Thereafter, the SMOTE technique is applied to the training data in order to mitigate class imbalance and evaluate potential improvements.

Feature selection is conducted on the training subset to create multiple scoped

feature configurations. ANOVA rankings are employed to derive nested subsets preserving 100%, 75%, 50%, and 25% of the original variable count. In parallel, a model-based filtering procedure relies on feature importances extracted from tree ensemble methods to construct an alternative reduced set. This step permits a systematic investigation of the effect of dimensionality reduction on model performance.

Four model families are trained across all combinations of temporal resolution, augmentation regime, and feature subset. Logistic regression is adopted as a linear baseline, whereas feedforward neural networks provide parametric non-linear modelling capabilities, incorporating configurable hidden layers alongside validation-based early stopping. Ensemble tree-based learners are represented by Random Forest and XGBoost.

For tree-based models that naturally estimate the feature importance, model-based filtering strategy is the only one employed in a bid to diminish the dimension. It helps in streamlining the training instances and optimizes the utilization of computational resources.

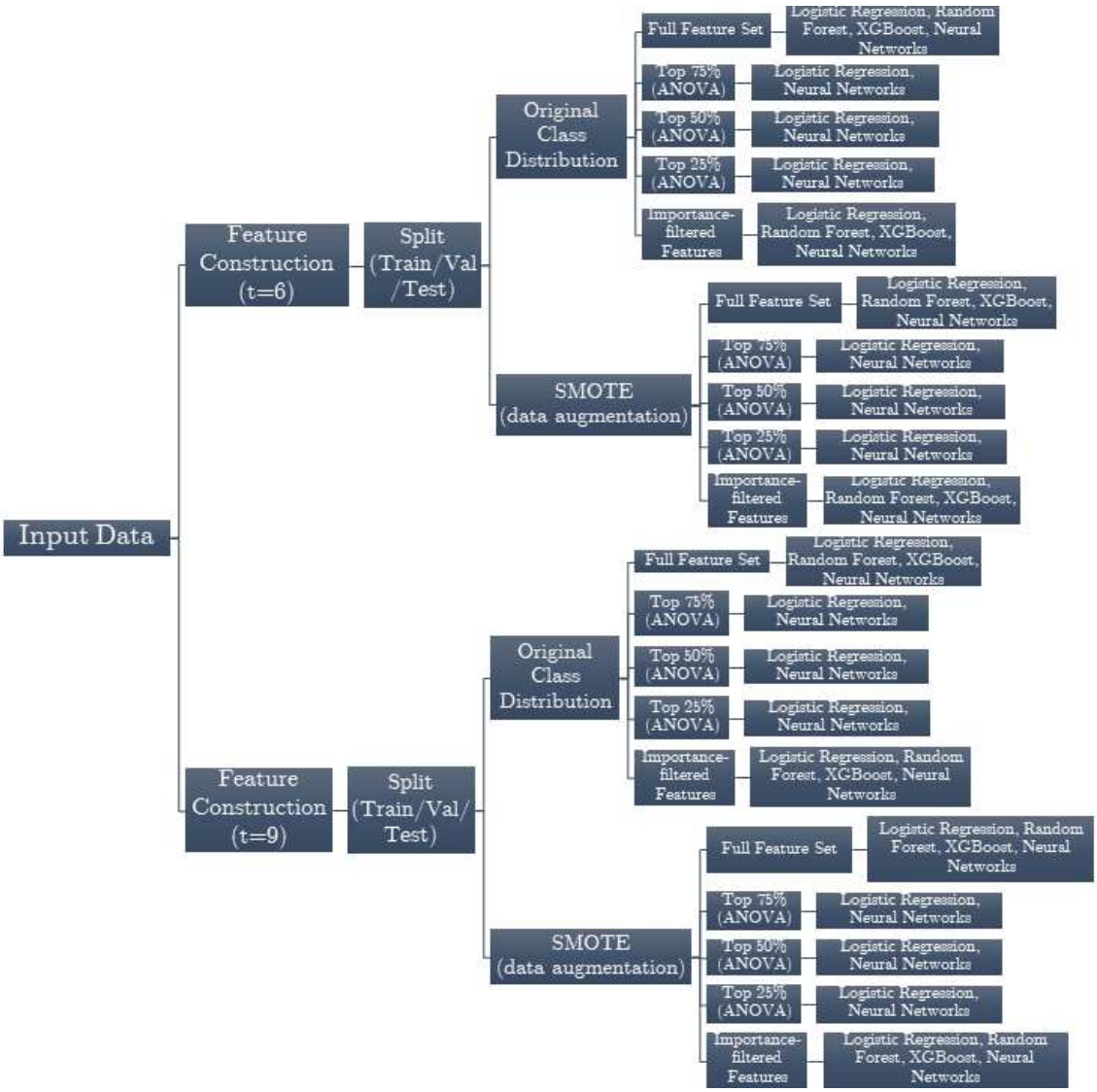

Figure 13: Block Diagram describing the model training protocol

4.5.2 Hyperparameter Tuning

A *hyperparameter* is a configuration variable whose value is set prior to the commencement of the training process, as opposed to *model parameters*, which are learned directly from data during optimization.

Hyperparameter tuning consists in the systematic process of finding the best set of hyperparameter values to minimize model performance on validation sets. Tuning is required since incorrect hyperparameters can result in slow convergence, underfitting, or overfitting. By tuning, one attempts to find hyperparameter settings that are able to control bias and variance and hence improve the generalization ability of the model.

Traditional methods are *grid search* and *random search* (Bergstra & Bengio, 2012), which search exhaustively or at random in pre-specified hyperparameter spaces. More advanced methods, like *Bayesian optimization* (Snoek et al., 2012), approximate the mapping from performance to hyperparameters in order to better direct the search. Even with these advances, real-world limitations remain: exhaustive search is intractable in terms of numbers of hyperparameters and dataset size and each evaluation involves full or partial model training, with high computational cost.

This work employs a Bayesian optimization approach to efficiently explore the hyperparameter space and identify the configuration that maximizes predictive performance. Bayesian optimization is selected for its ability to model the performance landscape probabilistically, guiding the search toward promising regions and reducing computational cost while maintaining robust and reproducible results.

Hyperparameter Tuning in Logistic Regression

In the context of Logistic Regression, hyperparameter tuning seeks to balance the bias–variance trade-off via regularization strength, penalty type, convergence behavior, and class imbalance handling, thereby enhancing predictive accuracy and robustness.

Inverse Regularization Strength (C). The parameter C controls the inverse of the regularization strength in the loss function. Smaller values of C correspond to stronger regularization, which reduces variance at the potential cost of increased bias. Conversely, larger C values permit the model coefficients to take larger magnitudes to reduce bias but risk overfitting (Hastie et al., 2009).

Penalty. The penalty hyperparameter specifies the norm used for regularization. Common choices include:

- **L2 regularization**, which penalizes the square of coefficient magnitudes to encourage small but distributed weights.
- **L1 regularization**, which penalizes the absolute values of coefficients to promote sparsity (i.e., feature selection).

(Hastie et al., 2009).

Solver. The `solver` selects the optimization algorithm to apply in order to minimize the logistic loss. Options like `lbfgs` use quasi-Newton optimizer appropriate for L2 and no penalty, while `saga` uses stochastic variance-reduced gradient descent appropriate for L1, L2, and Elastic Net penalties Tibshirani et al., 2010. Solver selection has an effect on convergence rate and numerical stability.

Maximum Iterations (`max_iter`). The parameter `max_iter` represents the number of iterations to converge for the optimization process. Too few iteration numbers can cause premature convergence and biased estimates of the coefficients, and values that are far too large incur additional computational cost with no improvement in performance.

Class Weight (`class_weight`). The `class_weight` parameter addresses imbalanced datasets by assigning higher penalty to misclassification of minority classes. Common settings include:

- `None` (uniform weights),
- `balanced`, which adjusts weights inversely proportional to class frequencies.

This reduces bias toward majority classes.

Elastic Net Mixing Parameter ($l1\_ratio$). When using the Elastic Net penalty (a convex combination of L1 and L2), the parameter $l1\_ratio$ determines the relative contribution of each norm. A value of 0 yields pure L2 regularization, 1 yields pure L1, and intermediate values blend both, integrating feature selection with coefficient shrinkage (Tibshirani et al., 2010).

Table 7: Search space and rationale (Multinomial Logistic Regression)

Hyperparameter	**Candidate values**	**Rationale**
`solver`	`{lbfgs, sag}`	`lbfgs` and `sag` are efficient solvers suitable for large datasets, with `sag` optimized for convergence in high-dimensional settings.
`C`	`[1e-3, 1e3] (log-uniform)`	Wide logarithmic range to capture the optimal bias–variance trade-off between strong and weak regularization.
`max_iter`	`{500, 1000, 2000}`	Ensure convergence across solvers with progressively higher iteration limits.
`penalty`	`{l2, none, elasticnet}`	Compare no regularization, pure L2, and Elastic Net strategies.
`tol`	`[1e-5, 1e-2] (log-uniform)`	Adjust tolerance for stopping criterion to balance precision and computational efficiency.
`class_weight`	`{None, balanced}`	Assess impact of imbalance adjustment on classification performance.

Hyperparameter Tuning in Random Forest

In the context of Random Forests, hyperparameter tuning seeks to balance the bias–variance trade-off by controlling tree complexity, ensemble diversity, sampling behavior, and class-imbalance handling, thereby improving predictive accuracy and robustness.

Number of Estimators (`n_estimators`). The parameter `n_estimators` sets the total number of decision trees in the forest. Increasing `n_estimators` generally reduces model variance through averaging of more learners but increases computational cost and yields diminishing returns on performance as the ensemble grows. Empirically, larger forests stabilize predictions but require longer training and prediction times. (Breiman, 2001).

Maximum Depth (`max_depth`). The `max_depth` parameter constrains the maximum height of each tree in the forest. Deeper trees can model more complex interactions (lower bias) but are more likely to overfit (higher variance); shallower trees encourage generalization at the cost of potentially underfitting the data. Choosing an appropriate depth is a direct way to control individual-tree complexity and the aggregate bias–variance behavior. (Hastie et al., 2009).

Minimum Samples for Node Splitting (`min_samples_split`). `min_samples_split` specifies the smallest number of samples required to split an internal node. Larger values produce simpler trees (increasing bias, reducing variance) by preventing very small, data-specific splits; smaller values allow more granular splits (reducing bias, increasing variance), which can capture fine structure but risk overfitting. (Louppe, 2015).

Minimum Samples per Leaf (`min_samples_leaf`). The `min_samples_leaf` hyperparameter enforces a lower bound on the number of observations in each leaf node. Raising this bound prunes overly specific branches and smooths predictions (bias↑, variance↓), while very small leaf sizes permit highly specific rules that may increase variance on unseen data. (Louppe, 2015).

Maximum Features (`max_features`). `max_features` limits the number of candidate predictors considered at each split, which promotes diversity and decorrelation among trees in the ensemble. Typical options include:

- `sqrt` (square root of the number of features),
- `log2`,
- `None` (consider all features).

Smaller values increase randomness and ensemble diversity (often lowering variance), while larger values let individual trees use more of the feature space (potentially reducing bias but increasing correlation between trees). (Breiman, 2001).

Bootstrap Sampling (`bootstrap`). The `bootstrap` flag determines whether each tree is trained on a sampled subset of the data with replacement. Enabling bootstrapping (`True`) injects randomness, reduces variance through aggregation, and enables out-of-bag error estimation; disabling it (`False`) trains trees on the full dataset, which can reduce bias but may increase correlation among estimators and thereby limit variance reduction benefits from averaging. (Breiman, 2001).

Class Weight (`class_weight`). The `class_weight` parameter addresses imbalanced classification by changing the penalty for misclassifying different classes. Common settings include:

- `None` (uniform weights),
- `balanced`, which scales weights inversely to class frequencies.

Applying class weighting often reduces bias toward majority classes, improving minority-class recall at the possible cost of increased variance in decision boundaries for under-represented classes. (Breiman, 2001).

Table 8: Search space and rationale (Random Forest)

Hyperparameter	Candidate values	Rationale
n_estimators	[100, 2000]	Covers small ensembles for efficiency through very large forests that minimize variance; higher values improve stability at the cost of compute and potential diminishing returns.
max_depth	[3, 50]	Allows shallow trees (depth 3) for strong regularization and very deep trees (up to 50) to capture complex patterns, testing the balance between underfitting and overfitting.
min_samples_split	[2, 20]	Wider range captures highly flexible splitting (2 samples) through stricter requirements (20 samples), affecting variance reduction versus generalization stability.
min_samples_leaf	[1, 20]	Includes single-instance leaves for fine-grained splits up to larger leaf sizes that enforce smoothing and reduce variance, directly controlling model regularization.
max_features	['sqrt', 'log2', None]	Explores subspace sampling strategies: square root and log2 promote tree diversity, while None tests whether full feature availability yields stronger splits.
bootstrap	[True, False]	Maintains standard bagging-based variance reduction (True) versus using the entire dataset per tree (False), testing impact on bias and correlation among trees.
class_weight	[None, 'balanced']	Evaluates the model's robustness to class imbalance, crucial for minority-class performance.
criterion	['gini', 'entropy', 'log_loss']	Tests different split quality measures: Gini for efficiency, entropy for information gain, and log loss for probabilistic consistency.
max_leaf_nodes	[10, 1000]	Constrains tree growth by limiting leaves; fewer nodes promote simpler, interpretable trees, while more nodes enable modeling finer-grained patterns.
min_impurity_decrease	[0.0, 0.5]	Requires a minimum impurity drop for splits; low thresholds allow many splits, while higher ones enforce conservative tree growth and stronger regularization.
ccp_alpha	[1e-6, 1e-1]	Controls post-pruning with minimal cost–complexity; very small values allow full trees, while larger values prune aggressively to prevent overfitting.

Hyperparameter Tuning in XGBoost

In the context of XGBoost, hyperparameter tuning aims to optimize the model's predictive performance by balancing learning speed, model complexity, regularization strength, and sampling behavior, thereby controlling overfitting, accelerating convergence, and improving generalization.

Number of Boosting Rounds (`n_estimators`). The hyperparameter `n_estimators` specifies the maximum number of sequential trees (boosting rounds) to be added to the ensemble. A larger number of rounds allows the model to fit more residuals and reduce training error, at the cost of increased computation and potential overfitting if rounds exceed the point of diminishing returns. (Chen & Guestrin, 2016; Friedman, 2000).

Learning Rate (`learning_rate`). Also known as eta, the learning rate scales the contribution of each new tree. Lower values yield slower, more conservative updates—mitigating overfitting by requiring more trees—whereas higher values speed convergence but risk fitting noise. (Chen & Guestrin, 2016).

Maximum Tree Depth (`max_depth`). The `max_depth` parameter constrains the depth of each regression tree. Shallower trees enforce simplicity and improve generalization by reducing variance, while deeper trees capture complex feature interactions at the risk of overfitting. (Friedman, 2000).

Subsample Ratio (`subsample`). `subsample` controls the fraction of training instances randomly sampled for each boosting round. Values below 1.0 inject stochasticity, reducing variance and overfitting by preventing each tree from seeing the full dataset. (Chen & Guestrin, 2016).

Column Sampling by Tree (`colsample_bytree`). Analogous to Random Forest feature bagging, `colsample_bytree` sets the fraction of features considered when constructing each tree. Lower feature fractions enhance decorrelation among trees—reducing variance—whereas full feature usage may decrease bias but increase inter-tree correlation. (Chen & Guestrin, 2016).

L1 Regularization (`reg_alpha`). `reg_alpha` applies an L1 penalty to leaf weights, encouraging sparsity by shrinking less important features' contributions to zero. A zero value disables L1 regularization, while a positive value enforces shrinkage, promoting simpler models and mitigating overfitting. (Chen & Guestrin, 2016).

L2 Regularization (`reg_lambda`). `reg_lambda` imposes an L2 penalty on leaf weights, smoothing weight estimates and reducing model complexity. A higher L2 penalty controls large weight magnitudes, improving stability at the cost of potentially underfitting; setting it to 0 disables L2 regularization entirely. (Chen & Guestrin, 2016).

Early Stopping (`early_stopping_rounds`). Early stopping monitors validation performance and halts training when improvement stalls for a specified number of rounds. The procedure terminates boosting if no validation loss reduction occurs

within a set number of consecutive trees, preventing unnecessary rounds and overfitting. (Prechelt, 2000).

Table 9: Search space and rationale (XGBoost)

Hyperparameter	**Candidate values**	**Rationale**
`n_estimators`	`[100, 2000]`	Ranges from 100 to 2000 to allow both compact ensembles for fast training and very large ensembles when paired with small learning rates; the upper bound enables fine-grained residual fitting at the cost of compute and higher overfitting risk.
`learning_rate`	`[0.001, 0.5]`	Covers very conservative rates (0.001) that require many trees for stable, fine-grained updates, up to aggressive rates (0.5) for fast convergence with fewer estimators — enabling trade-offs between stability, bias, and training time.
`max_depth`	`[2, 15]`	Expanded to integers 2–15 to capture models from very shallow (high-bias, fast) trees up to deep trees that model complex interactions; depths toward 15 can improve fit on complex data but increase overfitting risk and should be balanced with regularization.
`subsample`	`[0.3, 1.0]`	Wider range (0.3–1.0) to test aggressive row sampling for strong stochastic regularization (reducing variance) up to full-data boosting; lower bounds reduce correlation between trees but may require more estimators.
`colsample_bytree`	`[0.3, 1.0]`	Varying feature-subsampling from 0.3 to 1.0 lets the search trade off aggressive feature regularization (improved decorrelation and speed) against using the full feature set for maximal predictive power.
`reg_alpha`	`[0, 0.5]`	Tests the effect of L1 regularization for encouraging sparsity and simpler models.
`reg_lambda`	`[0, 0.5]`	Examines the role of L2 regularization in smoothing weight estimates and controlling complexity.
`early_stopping_rounds`	`[15]`	Prevents unnecessary rounds and overfitting by halting training when validation performance plateaus.
`gamma`	`[0.0, 10.0]`	Searches minimum loss reduction thresholds from 0 (allow any split) up to 10 (strong split-conservatism); larger values enforce pruning and can substantially reduce overfitting for deep trees.

Hyperparameter Tuning in Neural Networks

In the context of fully-connected feedforward Neural Networks, hyperparameter tuning aims to optimize the model's predictive performance by balancing model capacity, optimization dynamics, regularization strength, and computational trade-offs—thereby controlling overfitting, accelerating stable convergence, and improving generalization.

Hidden-layer configuration (`hidden_layers`). Specifies the number and size of dense layers (for example, `(64,64)`, `(128,64,32)`, `(128,128,64,32)`). Wider and deeper architectures increase representational capacity and permit fitting more complex functions, but they also raise the risk of memorizing noise and overfitting if not paired with adequate regularization. (Goodfellow et al., 2016).

Activation function (`activation`). Introduces element-wise nonlinearity. Rectified Linear Units (ReLU) preserve positive activations and mitigate vanishing gradients in deep networks, while hyperbolic tangent (tanh) yields zero-centered outputs that can accelerate optimization in some shallow or moderately deep architectures. Choice of activation affects training dynamics and final performance. (LeCun et al., 2012; Nair & Hinton, 2010).

Optimizer (`optimizer`). Determines the weight-update rule and adaptation behavior. Stochastic Gradient Descent (SGD) with momentum provides controlled, stable descent but can be slower; Adam adapts per-parameter learning rates using first- and second-moment estimates and often converges faster in practice. Optimizer interacts with learning rate and batch size to shape convergence and generalization. (Kingma & Ba, 2017; Sutskever et al., 2013).

Learning rate (`learning_rate`). Scales gradient steps. Larger values (e.g., 1e−2) accelerate progress but may cause divergence; smaller values (e.g., 1e−4) produce stable, fine-grained updates at the cost of slower training. A logarithmic grid spanning $10^{-2}, 10^{-3}, 10^{-4}$ captures coarse-to-fine regimes for optimization. (Goodfellow et al., 2016).

Dropout rate (`dropout_rate`). Probabilistically deactivates units during training to reduce co-adaptation and overfitting. Moderate (0.3) and aggressive (0.5) dropout settings explore the trade-off between preserving capacity and enforcing robustness. (Srivastava et al., 2014).

L2 regularization strength (`l2_penalty`). Adds an L2 penalty $\lambda\|w\|_2^2$ to the loss to discourage large weights and promote smoother functions. Smaller values lightly constrain weights while larger values (within the tested grid) enforce stronger shrinkage, helping generalization when overfitting is present. (Ng, 2004).

Batch size (`batch_size`). Controls the number of samples per gradient update. Smaller batches (e.g., 32) introduce higher gradient noise that may help escape sharp minima; larger batches (e.g., 64) produce more stable updates and can be more computationally efficient on modern hardware. (Keskar et al., 2017).

Number of epochs (`epochs`). Specifies the maximum number of full passes over the dataset. A fixed budget (300 epochs) provides sufficient opportunity for convergence across configurations while bounding compute; in practice this is frequently paired with validation-based stopping. (Goodfellow et al., 2016).

Search method and validation (`search_method`, `validation_strategy`). An exhaustive grid search was selected to evaluate all specified configurations for interpretability and completeness, and each combination is assessed via stratified 5-fold cross-validation to ensure robustness to class imbalance and fold variability. The optimal configuration is chosen by maximizing the mean cross-validated performance metric (e.g., validation accuracy) while taking fold-to-fold variance into account. (Bergstra & Bengio, 2012; Kohavi, 2001).

Table 10: Search space and rationale (Neural Networks)

Hyperparameter	Candidate values	Rationale
`hidden_layers`	`[(128, 64, 32), (128, 128, 64, 32), (256, 128, 64), (128, 64)]`	Expands the search to include both deeper/wider and shallower architectures, allowing the model to adjust representational capacity to task complexity while maintaining a manageable search space.
`activation`	`['relu', 'tanh']`	Includes both ReLU for efficient training and sparse activations, and tanh for bounded, smooth activations, enabling exploration of non-linearities affecting convergence and generalization.
`optimizer`	`['adam', 'sgd']`	Compares adaptive (Adam) and classic gradient-based (SGD) optimizers to balance convergence speed with generalization control.
`learning_rate`	`[1e-4, 1e-1]`	Covers a wide logarithmic range from very small (1e-4) to moderate (1e-1) step sizes, allowing exploration of conservative updates versus faster convergence regimes.
`dropout_rate`	`[0.1, 0.6]`	Tests light to strong stochastic regularization, enabling the search to balance capacity retention with overfitting prevention.
`l2_penalty`	`[1e-5, 1e-2]`	Explores minimal to stronger weight decay for smoother weights and overfitting control across different model scales.
`batch_size`	`[16, 128]`	Evaluates small batches for noisy, potentially better generalizing updates versus larger batches for stable, efficient training.
`epochs`	`[50, 200]`	Allows flexible training budgets to accommodate different architectures and learning rates, while preventing under- or over-training by setting reasonable lower and upper bounds.

Chapter 5

Training Results

5.1 Evaluation Metrics and Models' Performance

Evaluating machine learning classifiers is crucial for developing predictive models, as it offers a structured way to assess how effectively a model meets its goals and to contrast different methodological choices. Typically, evaluation focuses on metrics like classification accuracy, precision, recall, and similar measures that evaluate the correctness of categorical predictions. However, in this specific context, the aim extends beyond merely determining the most probable categorical outcome, but it involves measuring uncertainty through probability estimates. The emphasis is on creating classifiers that generate accurately calibrated probabilistic forecasts rather than achieving perfect classification of match outcomes. Indeed, since football is subject to a considerable degree of aleatoric uncertainty, which arises from the random, unpredictable nature of the game itself, numerous micro-events exert stochastic influences that cannot be eliminated, regardless of the sophistication of the predictive model.

The importance of probability calibration becomes evident when connecting model outputs to the applied context of betting markets. Bookmaker odds are a direct expression of implied probabilities, calculated simply as the reciprocal of the decimal odds, i.e., $p = \frac{1}{\text{odds}}$. For instance, if a bookmaker offers decimal odds of 2.50 for *Home More Shots*, the implied probability is $\frac{1}{2.50} = 0.40$, or 40%. A rational betting strategy based on value arises when the bettor's model assigns a higher probability than the bookmaker's implied probability. In the example above, if the model predicts that the probability of *Home More Shots* is 0.47 (47%), then the bettor has identified a value bet, since the model suggests the bookmaker has underestimated the likelihood of this outcome. However, the effectiveness of such a strategy depends entirely on the accuracy of the predicted probabilities. A model that outputs probabilities that are systematically over- or under-confident will generate misleading signals, hindering the

possibility of creating a robust betting strategy and resulting into poor performance.

From a methodological perspective, there are several metrics that can be used to evaluate classifiers, each providing various perspectives through which one can assess performance. While traditional metrics like accuracy, precision, recall, and F1-score are effective for evaluating categorical accuracy, they are inadequate for directly assessing how well probabilities are calibrated. Conversely, probabilistic measures such as logarithmic loss offer a robust approach to gauge how accurately predicted probability distributions reflect actual outcomes. For this reason, while this thesis reports multiple evaluation measures to provide a comprehensive performance overview, log loss is treated as the most critical metric, since it quantifies probabilistic reliability.

Accuracy: Accuracy is defined as the proportion of correctly classified instances among all evaluated instances, formally expressed as

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}.$$

In the multiclass setting considered here, accuracy generalizes to the proportion of cases where the predicted class with the highest assigned probability matches the observed outcome.

While accuracy provides an intuitive measure and is frequently reported in classification studies, it is not ideal for this particular application. This is due to the fact that there is a slight imbalance toward the *Home More Shots* class, meaning that a naïve classifier that consistently predicts this outcome would achieve a modest accuracy.
More importantly, accuracy ignores the quality of probability estimates: a model that guesses 0.51 compared to 0.99 probability for the correct result is rewarded equally, even though the latter would likely be ill-calibrated if such confidence is unwarranted. Consequently, while accuracy offers useful background information, it is not prioritized in this thesis.

Precision: Precision is defined as the proportion of correctly predicted positive instances out of all instances classified as positive, i.e.,

$$\text{Precision} = \frac{TP}{TP + FP}.$$

In this thesis, precision extends to the multiclass case by computing it for each class separately and averaging. Accuracy indicates the degree to which to trust positive predictions to occur, and how often they match with the correct outcome.

While useful in capturing model trends, precision again fails to capture the fidelity of probability distributions.

Recall: Recall, also known as sensitivity, is defined as

$$\text{Recall} = \frac{TP}{TP + FN}.$$

Recall measures the proportion of actual instances of a given class that the model successfully identifies. High recall guarantees that the model does not systematically overlook particular outcomes.

Nevertheless, recall does not penalize overconfident or underconfident probability estimates, which makes it inadequate for evaluating probability calibration.

F1-Score: The F1-score is the harmonic mean of precision and recall, defined as

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

It provides a balanced metric that accounts for both false positives and false negatives. The F1-score is particularly informative when classes are imbalanced, as in this work's case, since it avoids accuracy's bias toward the majority class. Nevertheless, like accuracy, precision, and recall, the F1-score evaluates categorical predictions rather than the quality of probability estimates.

Log Loss: Logarithmic Loss (Log Loss) is defined for multiclass classification as

$$\text{Log Loss} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{j=1}^{M} y_{ij} \log(\hat{p}_{ij}),$$

where N is the number of samples, M the number of classes, y_{ij} a binary indicator of whether observation i belongs to class j, and $\hat{p}_{ij}$ the predicted probability assigned to class j for observation i.

Log loss penalizes the divergence between predicted probabilities and actual outcomes, assigning a large penalty to confident but incorrect predictions. This makes it the most critical metric for the present study, as it directly evaluates the alignment between model-predicted probabilities and real-world frequencies. A low log loss indicates that the model produces probability distributions that closely match reality. Thus, among all the evaluation metrics discussed, log loss is prioritized as the principal criterion for model assessment in this thesis, as well as for hyperparameter optimization.

5.1.1 Logistic Regression

$t = \mathbf{6}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.68	0.64	0.68	0.65	0.74849
75% Anova Filter	0.68	0.64	0.68	0.65	0.74786
50% Anova Filter	0.68	0.63	0.68	0.64	0.74783
25% Anova Filter	0.68	0.64	0.68	0.65	0.74789
Feature Importance Filter	0.68	0.64	0.68	0.64	0.74772

Table 11: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.65	0.63	0.65	0.64	0.77319
75% Anova Filter	0.66	0.63	0.66	0.64	0.77273
50% Anova Filter	0.65	0.63	0.65	0.64	0.77066
25% Anova Filter	0.65	0.63	0.65	0.64	0.77171
Feature Importance Filter	0.65	0.63	0.65	0.64	0.77173

Table 12: With SMOTE application

$t = \mathbf{9}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.63	0.75042
75% Anova Filter	0.66	0.62	0.66	0.63	0.75028
50% Anova Filter	0.67	0.62	0.67	0.63	0.74932
25% Anova Filter	0.67	0.63	0.67	0.63	0.74891
Feature Importance Filter	0.66	0.62	0.66	0.63	0.75189

Table 13: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.64	0.61	0.64	0.63	0.77182
75% Anova Filter	0.64	0.61	0.64	0.62	0.76954
50% Anova Filter	0.64	0.62	0.64	0.63	0.76967
25% Anova Filter	0.64	0.61	0.64	0.62	0.77003
Feature Importance Filter	0.64	0.62	0.64	0.63	0.77236

Table 14: With SMOTE application

The best-performing experimental setting is $t = 6$ without SMOTE using the Feature Importance filter, which achieves the lowest Log Loss of 0.74772. Across the tables, models trained without SMOTE consistently show lower logistic loss than their SMOTE counterparts, indicating that applying SMOTE systematically increased uncertainty for this logistic model by roughly 0.02–0.03 in log loss. Comparing window sizes, the $t = 6$ experiments slightly outperform $t = 9$: the best $t = 6$ log loss is 0.74772 versus the best $t = 9$ log loss of 0.74891 (25% ANOVA filter), with a small gain of ≈ 0.0012. Feature selection yields modest improvements: for $t = 6$ the Feature Importance filter produced the single best result, while for $t = 9$ ANOVA-based pruning gave the lowest values. The other metrics mirror this pattern—higher for non-SMOTE runs (accuracies $\sim$0.66–0.68) and slightly reduced when SMOTE is used—supporting the conclusion that no-SMOTE, $t = 6$ with Feature Importance is preferable here.

5.1.2 Random Forest

$t = \mathbf{6}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.67	0.62	0.67	0.64	0.74752
Feature Importance Filter	0.67	0.62	0.67	0.64	0.7498

Table 15: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.64	0.75731
Feature Importance Filter	0.67	0.63	0.67	0.65	0.75787

Table 16: With SMOTE application

$t = \mathbf{9}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.63	0.74853
Feature Importance Filter	0.66	0.62	0.66	0.63	0.74993

Table 17: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.65	0.61	0.65	0.63	0.75781
Feature Importance Filter	0.65	0.61	0.65	0.63	0.75909

Table 18: With SMOTE application

The best-performing configuration is the full feature set with $t = 6$ and no SMOTE, which attains the lowest log loss of 0.74752 (accuracy 0.67, weighted F1 0.64). Closely following is the $t = 9$ full model without SMOTE (log loss 0.74853), indicating a little negative effect when extending the averaging window from 6 to 9 matches. Applying SMOTE consistently increases log loss across both t values (for example, the full models' log losses rise to 0.75731 at $t = 6$ and 0.75781 at $t = 9$), suggesting that oversampling reduces the model's probabilistic calibration despite similar accuracy and F1 figures. The feature-importance filter versions show marginally higher log loss than their respective full-set counterparts in all conditions, implying the filter either removed weak-but-helpful signals or slightly harmed probability estimation. Overall, differences in accuracy, precision and F1 are minimal (all metrics remain near 0.65–0.67), so log loss is the decisive discriminator: the $t = 6$, no-SMOTE, full-feature model is preferable for producing the most confident and better-calibrated probability estimates.

5.1.3 XGBoost

$t = \mathbf{6}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.67	0.63	0.67	0.64	0.7461
Feature Importance Filter	0.66	0.62	0.66	0.63	0.74437

Table 19: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.64	0.75731
Feature Importance Filter	0.67	0.63	0.67	0.65	0.75787

Table 20: With SMOTE application

$t = \mathbf{9}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.63	0.74898
Feature Importance Filter	0.66	0.62	0.66	0.64	0.74833

Table 21: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.65	0.61	0.65	0.63	0.75781
Feature Importance Filter	0.65	0.61	0.65	0.63	0.75909

Table 22: With SMOTE application

Across all tested configurations, the lowest log loss value is observed for $t = 6$ with feature importance filtering and without SMOTE, achieving 0.74437. Without applying feature filtering, the same model yields a marginally increased log loss of 0.7461, implying that feature selection plays a modest role in enhancing performance. When SMOTE is implemented at $t = 6$, log loss values increase to approximately 0.757, indicating that oversampling causes some decline in model calibration even as the weighted F1-Score shows negligible improvements. At $t = 9$, the findings are more stable: absent SMOTE, the log loss values are 0.74898 and 0.74833, quite similar to those at $t = 6$, yet slightly inferior to the optimal scenario. Nevertheless, when applying SMOTE at $t = 9$, the performance once again deteriorates, with log loss climbing to around 0.758–0.759. In general, shorter averaging periods coupled with omitting SMOTE seem to improve probability calibration, with the best scenario occurring at $t = 6$ with feature filtering and no SMOTE usage.

5.1.4 Neural Networks

$t = \mathbf{6}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.67	0.62	0.67	0.64	0.74854
75% Anova Filter	0.68	0.64	0.68	0.65	0.74837
50% Anova Filter	0.67	0.63	0.67	0.65	0.74387
25% Anova Filter	0.68	0.64	0.68	0.65	0.74228
Feature Importance Filter	0.67	0.63	0.67	0.65	0.74571

Table 23: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.66	0.62	0.66	0.64	0.76924
75% Anova Filter	0.65	0.63	0.65	0.64	0.77186
50% Anova Filter	0.62	0.57	0.62	0.57	0.80545
25% Anova Filter	0.67	0.63	0.67	0.65	0.76207
Feature Importance Filter	0.63	0.60	0.63	0.62	0.80532

Table 24: With SMOTE application

$t = \mathbf{9}$

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.65	0.61	0.65	0.62	0.75346
75% Anova Filter	0.66	0.61	0.66	0.62	0.75018
50% Anova Filter	0.66	0.62	0.66	0.64	0.75045
25% Anova Filter	0.66	0.62	0.66	0.63	0.75621
Feature Importance Filter	0.66	0.63	0.66	0.63	0.75487

Table 25: Without SMOTE application

	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Full	0.65	0.61	0.65	0.63	0.78117
75% Anova Filter	0.64	0.62	0.64	0.63	0.83300
50% Anova Filter	0.65	0.61	0.65	0.62	0.78134
25% Anova Filter	0.64	0.61	0.64	0.62	0.79293
Feature Importance Filter	0.66	0.61	0.66	0.63	0.77481

Table 26: With SMOTE application

Among all settings, the lowest log loss value of 0.74228 is achieved using $t = 6$ without SMOTE and applying the 25% Anova filter. In contrast, the introduction of SMOTE consistently increases log loss values across both $t = 6$ and $t = 9$. Oversampling yields less calibrated probability estimates and decreased accuracy levels, with lower performance when applying 50% Anova filter and tree-based selection method. When comparing the two averaging windows, the $t = 6$ condition generally outperforms $t = 9$, as the latter rarely drops below 0.750 in log loss without SMOTE and performs even worse with SMOTE, exceeding 0.78 in most cases. Overall, shorter-term averaging without SMOTE and with more selective feature filtering appears to yield superior predictive reliability.

5.2 Results Discussion

Strategy	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score	Log Loss
Random Forest	0.67	0.62	0.67	0.64	0.74752
Logistic Regression	0.68	0.64	0.68	0.65	0.74772
XGBoost	0.67	0.63	0.67	0.64	0.74405
Neural Network	**0.68**	**0.64**	**0.68**	**0.65**	**0.74228**

Table 27: Models' Performance Comparison.

The experimental results present a consistent pattern across model families: (1) a shorter temporal aggregation window ($t = 6$) yields modestly better predictive performance than a longer window ($t = 9$); (2) omitting SMOTE produces better-calibrated probability estimates than applying SMOTE; and (3) feature selection yields stable, modest improvements across learners. The best observed log-loss value is 0.74228 (Neural Network, $t = 6$, 25% ANOVA filter). XGBoost attains the second-best value (0.74437, $t = 6$, feature-importance filter). The Logistic Regression baseline achieves 0.74772 under its optimal configuration ($t = 6$, no SMOTE, feature-importance filter), a figure that is comparable to the best Random Forest result (0.74752, $t = 6$, no SMOTE, full feature set).

At first, the strong performance of Logistic Regression was surprising, particularly given its simplicity relative to other approaches like Random Forest, which is commonly used in football modeling.

One of the reasons might hinge upon correlated features, which could lead to multicollinearity. Although tree-based models are often robust to prediction accuracy under multicollinearity, strongly correlated features can induce instability in variable importance scores, making them harder to interpret. This also leads to overfitting on noisy patterns of the training data, adversely affecting the model's ability to generalize to new data. Logistic Regression, through the application of L1 or L2 regularization, effectively addresses multicollinearity, setting the coefficients of irrelevant features to zero (L1 Regularization - Lasso) or pulling the coefficients of irrelevant variables towards one another, improving model stability and generalizability (L2 Regularitazion - Ridge).

Another explanation may lie in the linearity of the output. Nonetheless, this work has consistently noted that although shot dominance appears more linear than match outcomes or goals, it is still influenced by numerous nonlinear factors, supported by the effectiveness of complex models like neural networks.

Finally, the dataset's high dimensionality might, paradoxically, benefit more from a simpler model in this instance. Logistic regression—a model with significantly fewer parameters—tends to overfit less, particularly when regularization is applied. It exhibits higher bias but lower variance, potentially improving performance on test data.

Random Forest, instead, demands vast datasets to create strong, independent trees. In high-dimensional settings with limited samples, these trees might be excessively correlated, diminishing the "forest" effect and elevating the risk of overfitting.

It is important to remark that, eventually, more advanced and complex models can outperform simpler models. XGBoost can surpass logistic regression because it effectively captures nonlinearities and interactions within the dataset and inherently manages feature sparsity and heavy-tailed distributions. Comparing XGBoost with Random Forest, XGBoost's boosting mechanism is suggested to reduce bias by sequentially addressing errors, as opposed to Random Forest's averaging approach of high-variance trees, which often leads to higher AUC and PR-AUC on structured tabular data. XGBoost's incorporation of regularization methods, such as learning rate adjustments, L1/L2 penalties, and column subsampling, is anticipated to improve generalization in high-dimensional contexts. Additionally, XGBoost's additive framework might better highlight feature importance and accommodate subtle monotonic effects. Ultimately, Neural Networks excel at learning detailed high-order interactions across many continuous features compared to tree ensembles. Although overfitting is a common risk in this context, effective regularization techniques like dropout, weight decay, and early stopping have been applied to mitigate this issue, preserving optimal generalization capacity and performance.

5.2.1 Baseline Strategies

The following baseline strategies are used to assess the predictive performance of the trained machine learning model. Baselines serve two principal purposes: they establish minimal performance thresholds that any useful model must exceed, and they provide interpretable points of comparison that help to contextualize improvements achieved by more complex methods. The baselines presented here are intentionally simple, transparent, and reproducible, enabling straightforward evaluation, while still offering a level of sophistication beyond that of a naive random strategy.

Naive Predictor. The Naive Predictor always predicts the same outcome for every match: the majority class observed in the training set, *Home More Shots*. This strategy is the simplest conceivable classifier, supplying a lower bound on classification accuracy and related metrics; any model that cannot outperform this trivial rule is of limited practical value, since it is not able to beat a strategy that does not exploit any contextual information.

Historical Averages Strategy. The Historical Averages Strategy constructs match-specific point estimates of expected shots for each side from team-specific historical statistics. In practice, the predicted number of shots for the home team is computed as

$$S_h = \tfrac{1}{2}\big(\bar{s}_{h,\text{home}} + \bar{c}_{a,\text{away}}\big),$$

where $\bar{s}_{h,\text{home}}$ denotes the home team's empirical average shots when playing at home and $\bar{c}_{a,\text{away}}$ denotes the away team's empirical average shots conceded when playing away. Analogously, the predicted number of shots for the away team is

$$S_a = \tfrac{1}{2}\big(\bar{s}_{a,\text{away}} + \bar{c}_{h,\text{home}}\big).$$

The classifier issues the label *Home More Shots* if $S_h > S_a$, *Away More Shots* if $S_h < S_a$, and a *Draw* label when $S_h = S_a$. As a benchmark, it assesses whether a model trained on a richer feature set meaningfully outperforms heuristics based on simple historical aggregates.

Betting Odds Strategy. The Betting Odds Strategy exploits pre-match market odds data for match outcomes, on the assumption that (1) bookmakers offer good proxies for match expectations betting markets and (2) match outcome information provides valuable context for understanding shot outcome due to the correlation between these variables. In practice, the strategy compares the two teams' decimal win odds and predicts that the team with the lower decimal odds (i.e., the market favorite to win the match) will also be the team to take more shots. This strategy is appealing for its simplicity while simultaneously offering a strategy that a typical bettor could employ and an integration of insights from bookmakers' technology.

It is important to note that these strategies generates deterministic point predictions rather than calibrated probability estimates; consequently, assessment of probabilistic calibration and uncertainty quantification will be treated seperately.

5.2.2 Final Comparison

Before presenting the final assessment of the best-performing model (neural network), it is useful to review the performance of the baseline strategies. (Table 28).

Strategy	Accuracy	Weighted Precision	Weighted Recall	Weighted F1-Score
Naive Predictor	0.59	0.35	0.59	0.44
Historical Averages	0.65	0.61	0.65	0.61
Betting Odds	0.65	0.63	0.65	0.63

Table 28: Comparison of baseline strategies performance.

The naïve strategy attains an accuracy of 59%. At first glance it may appear acceptable, but the strategy is unable to discriminate between classes: it exploits the dominant *Home More Shots* class and therefore reflects the home advantage bias discussed throughout this work.

The *Historical Averages* and *Betting Odds* strategies, leveraging more contextual information, perform markedly better: each achieves an overall accuracy of 65%.

These values are comparable to the lower end of the model accuracies observed in this study. In particular, the *Betting Odds* strategy slightly reduces the number of predicted *draws* (the rarest class) compared to the *Historical Averages* one, which marginally improves its empirical accuracy, since this class remain difficult to predict reliably given its low occurrence (support = 118).

These baselines can be used then to assess whether the neural network model extracts additional predictive value from the contextual features.

Strategy	**Accuracy**	**Weighted Precision**	**Weighted Recall**	**Weighted F1-Score**
Naive Predictor	0.59	0.35	0.59	0.44
Historical Averages	0.65	0.61	0.65	0.61
Betting Odds	0.65	0.63	0.65	0.63
Neural Network	**0.68**	**0.64**	**0.68**	**0.65**

Table 29: Neural Network v Baseline Strategies.

Table 29 summarizes the comparison between the neural network and the baselines. The neural network's best configuration ($t = 6$, 25% ANOVA filter, no SMOTE) attains an accuracy of 68%, outperforming the two informed baselines by 3 percentage points. In absolute terms, on the test set of 2,268 matches this corresponds to approximately 68 additional correctly predicted matches ($2,268 \times 0.03 \approx 68$).

Beyond accuracy, the neural network shows improvements in the weighted aggregate metrics. Compared to the strongest baseline (*Betting Odds*), the neural network yields a weighted precision of 0.64 (+0.01), a weighted recall of 0.68 (+0.03), and a weighted F1-score of 0.65 (+0.02). Against the *Historical Averages* baseline, the neural network improves weighted precision by $\approx +0.03$, weighted recall by +0.03, and weighted F1 by +0.04. These gains indicate that the neural network is not only more accurate overall, but also produces more reliable class-level predictions on average.

5.3 Calibration

Model calibration refers to the process of adjusting the predicted probabilities of a machine learning model so that they accurately reflect the true likelihood of an event. The primary goal of calibration is to ensure that, for instance, among all instances assigned a predicted probability of 0.7, approximately 70% actually exhibit the target outcome.

Within the literature relevant to this research, calibration is rarely evaluated explicitly. Forecast performance is typically evaluated using aggregate metrics like the Brier score or ranked probability score, while often overlooking the confidence level of the probabilities generated by the models. A few works inspect prediction calibration: (Hubáček et al., 2021) present calibration curves for models predicting soccer

match outcomes; (Wheatcroft & Sienkiewicz, 2021) examine the calibration of machine learning predictions, demonstrating a potential correlation between forecast probabilistic accuracy and average profit in betting. For instance, forecasts generated with the neural networks exhibit both optimal negative log loss and the greatest profit (Wheatcroft & Sienkiewicz, 2021). Nonetheless, there is a broad research gap since there is no previous study that systematically compares calibration methods in this domain.

(Niculescu-Mizil & Caruana, 2005) highlight Platt scaling and isotonic regression as default calibrations for calibration methods. Platt scaling applies a sigmoid (logistic) function to a model's decision scores (originally designed for SVMs), essentially converting raw scores to calibrated probabilities. Isotonic regression, on the other hand, learns a general non-parametric mapping to transform predicted scores to observed outcome frequencies. The Platt mapping has a nearly sigmoidal bias, while isotonic regression can remove spurious monotonic distortion. The two calibrators are learned from hold-out data and then used to adjust the classifier outputs so that empirical event frequencies are more accurately estimated.

The following section will evaluate the best-performing model from each of the previously identified algorithms using calibration curves. This analysis aims to assess the reliability of predicted probabilities in a practical setting. Subsequently, Platt scaling and isotonic regression will be utilized to evaluate whether the calibration of the trained models can be enhanced.

5.3.1 Calibration Curves

A calibration curve in machine learning represents a visual depiction that gauges how well a model's predicted probabilities line up with the actual observed outcome frequencies, offering a graphical evaluation of model calibration (Lane, 2025). These plots generally feature the predicted probability on the x-axis and the frequency of actual outcomes on the y-axis.

Below are represented the calibration curves for the most important classes (*Home More Shots* and *Away More Shots*) for each of the best model for each algorithm.

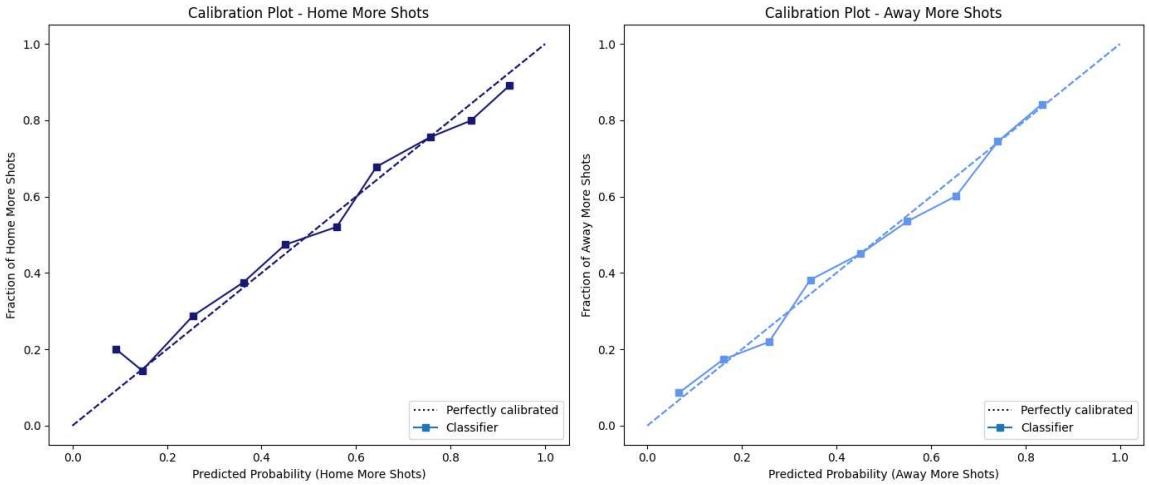

Figure 14: Random Forest Calibration Curve

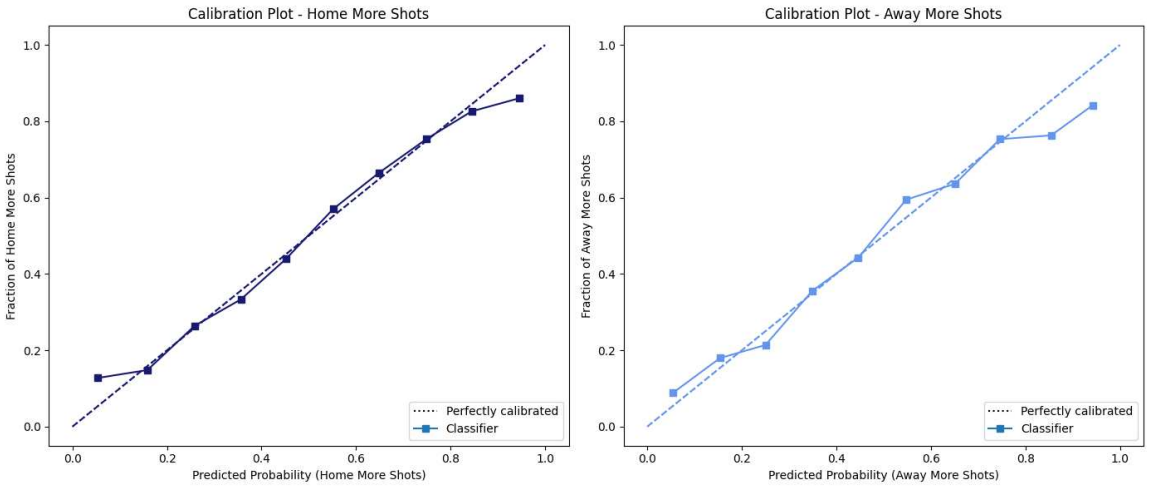

Figure 15: Logistic Regression Calibration Curve

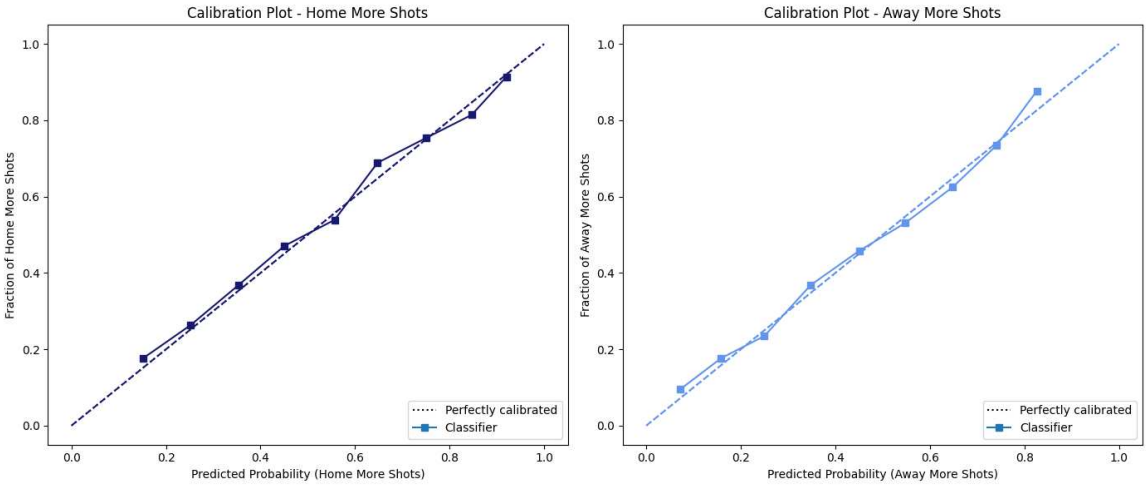

Figure 16: XGBoost Calibration Curve

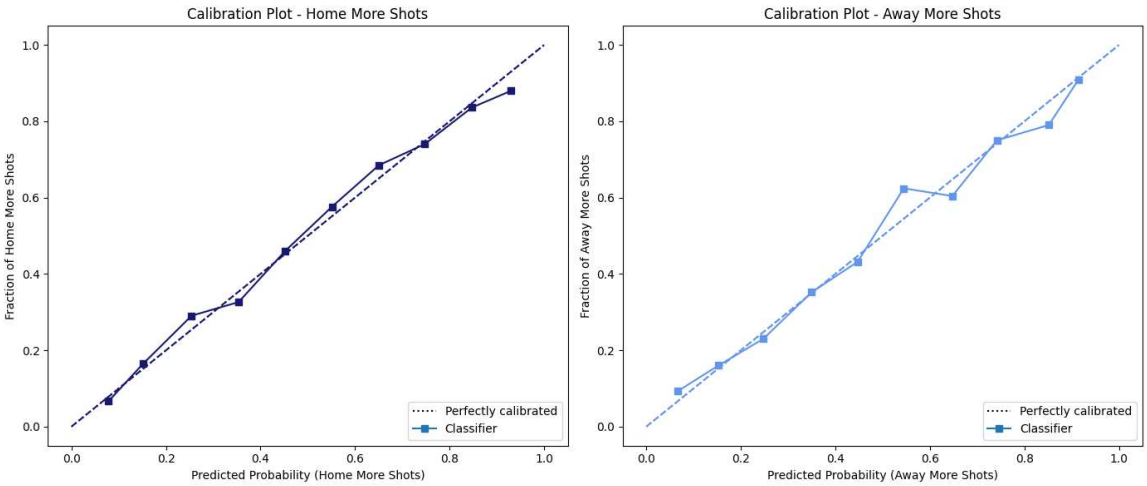

Figure 17: Neural Network Calibration Curve

The visual representation indicates that all models appear to be well calibrated with respect to the majority class, while XGBoost demonstrates superior performance compared to the other models in relation to the *Away More Shots Class*.

However, visual representation might not be enough to assess the performance. In the previous sections, log loss has been used as the discriminant metric to assess the performance and calibration of the different classifiers. Here an additional metric will be introduced to further verify the calibration of the models, namely the Brier Score, which is a scoring rule that quantifies the accuracy of probabilistic forecasts by measuring the mean squared difference between predicted probabilities and observed outcomes (Brier, 1950). In the multi-class setting, the Brier Score is formulated as:

$$BS = \frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{K} (p_{i,k} - y_{i,k})^2, \tag{7}$$

where K is the number of classes, $p_{i,k}$ is the predicted probability for class k in the i-th instance, and $y_{i,k}$ is the indicator (1 if class k is observed, 0 otherwise) (Gneiting & Raftery, 2007).

Both the Brier Score and log loss aim to minimize the expected score under the true distribution. However, the Brier Score emphasizes quadratic differences, which promotes calibration by considering errors based on their squared size. On the other hand, log loss uses a logarithmic penalty, making it more sensitive to errors in the probability tails, severely penalizing overconfident incorrect predictions and encouraging both precision and calibration.

Mathematically, the Brier Score's quadratic form yields a bounded range, akin to mean squared error, which enhances interpretability and robustness to extreme probabilities. The log loss is, instead, unbounded and can diverge to infinity for predictions approaching 0 or 1 incorrectly, implying greater emphasis on avoiding absolute certainties unless justified and facilitating gradient-based optimization.

These mathematical traits translate to practical differences: the log loss might be more useful in a training context, where the goal is to optimize the probabilistic calibration of the machine learning models. On the other hand, the Brier Score could prove more beneficial for assessing and interpreting these models.

Model	Brier Score
Random Forest	0.4474
XGBoost	0.4453
Logistic Regression	0.4448
Neural Network	**0.4437**

Table 30: Brier Score Comparison

Table 30 shows, once more, that the neural network model proves itself as the top performer. It is crucial to acknowledge that both log loss and Brier Score are adversely affected by the *Draw* class, which is typically assigned to very low probabilities. This can lead to skewed results in those uncommon instances when the outcome is indeed a draw.

The subsequent task is to assess whether the model's calibration can be further enhanced.

5.3.2 Calibration Improvement

This section will explore the application of Platt scaling and isotonic regression to determine if the calibration of models discussed in the preceding section can be enhanced. To evaluate any potential improvements in calibration, the log loss and Brier score of the calibrated models will be compared to those of the uncalibrated ones. A decrease in both log loss and Brier Score will indicate that the calibration improvement process has been successful.

The validation set will be used to fit both the Platt scaling and isotonic regression models to calibrate the models.

Model	Uncalibrated Brier Score	Calibrated Brier Score	Uncalibrated Log Loss	Calibrated Log Loss
Random Forest	**0.4474**	0.6667	**0.7475**	1.0986
XGBoost	**0.4453**	0.4456	**0.7441**	0.7449
Logistic Regression	**0.4448**	0.6667	**0.7477**	1.0986
Neural Network	**0.4437**	0.445	**0.7423**	0.7445

Table 31: Effects on calibration after Platt scaling.

Model	Uncalibrated Brier Score	Calibrated Brier Score	Uncalibrated Log Loss	Calibrated Log Loss
Random Forest	**0.4474**	0.6667	**0.7475**	1.0986
XGBoost	**0.4453**	0.4468	**0.7441**	0.8465
Logistic Regression	**0.4448**	0.6667	**0.7477**	1.0986
Neural Network	**0.4437**	0.4438	**0.7423**	0.75

Table 32: Effects on calibration after isotonic regression.

Table 31 and Table 32 show the result of this procedure. As observed, the performance of all models declines upon implementing both Platt scaling and isotonic regression.

The comparative analysis reveals that Platt Scaling demonstrates superior robustness across models. As a parametric sigmoid-based method, it avoids the overfitting risks of non-parametric isotonic regression, which often degrades performance when sample sizes are limited. This pattern is evident in the results, where Platt Scaling maintains or slightly improves calibration, whereas isotonic regression frequently worsens predictive reliability, especially for more flexible learners.

Logistic Regression and Random Forest suffer the greatest degradation under calibration, with sharp increases in both log loss and Brier score. This is consistent with Logistic Regression being inherently well-calibrated, leaving little room for improvement, and with Random Forest's discrete voting mechanism, which produces probability distributions less suited to parametric or non-parametric recalibration. XGBoost and Neural Network experience a degradation as well, but mostly comparable to the original results.

Overall, these findings indicate that calibration is not universally advantageous. Thus, the in-built calibration of the different models might be considered already optimal.

Upon final evaluation, Neural Networks and XGBoost emerge as the most effective models. The former exhibits superior performance with respect to log loss and Brier score, while the latter displays better visual calibration in the calibration curves. Both models successfully surpass the established baseline strategies, indicating their ability to extract valuable insights from the features and generate credible and practical outcomes.

Nevertheless, this study has not adequately addressed the problem posed by the *Draws* class. This class has consistently posed a significant challenge in football modeling research, since it constitutes a minority class, even in different classification problems with other target variables. Specifically, in this thesis, the challenges associated with this class are further exacerbated, given that the occurrence of draws is less frequent in shot outcomes than in final results, for instance. This rarity renders the outcome exceptionally difficult to capture and accurately predict.

The presence of draws also significantly impairs the capacity to objectively evaluate the validity of the log loss and Brier score metrics. While the raw numerical assessments suggest that the model is not perfectly calibrated and therefore not wholly dependable for betting strategies—where nearly perfect calibration is crucial—the calibration curves reveal a different narrative. They demonstrate that the more prevalent classes are, in fact, reasonably well-calibrated. Therefore, the scores are adversely affected by the poor performance and calibration associated with draws, to which probabilities close to zero are typically assigned, resulting in significant spikes in both log loss and Brier score.

Chapter 6

Final Remarks

This thesis has addressed the task of predicting shot dominance in football matches, framed it as a multiclass classification problem.

By leveraging a rich dataset of historical match statistics and advanced metrics, the study developed and evaluated a set of several machine learning models.
Even though the study has found the presence of a significant home advantage in shot generation, a structural bias that might hinder the scope of the thesis, the work systematically demonstrated that data-driven models can extract meaningful information from pre-match features, outperforming established baseline strategies.

The research concludes that sophisticated modeling techniques can yield well-calibrated probabilistic forecasts, despite the inherent stochasticity present in football.

Several promising directions emerge from this research:

Expanding the dataset The expansion of the dataset can be approached in two ways: by increasing the sample size, thereby augmenting the number of matches available for training, and by broadening the feature set. While the former is feasible due to the availability of additional data from *Fbref*, the latter presents more challenges.

The current work already includes an impressive array of advanced metrics. Extending the feature set further might not yield substantial benefits and could lead the models to overfit, as shown in this study, where pruning the feature set resulted in noticeable performance improvements. Therefore, the introduction of new metrics should aim at two objectives: (i) utilizing feature engineering to create valuable metrics that capitalize on existing data. As an example, the *Historical Averages* strategy can serve as an inspiration. The feature space could be simplified by aggregating statistics to provide a singular expected value for each metric, such as merging home and away statistics, as well as taken and conceded metrics, to derive an expected

metric for a team's shots in a match, akin to this approach. (ii) The focus should shift from discovering new metrics to developing quantitative methods to assess aspects of the game that have traditionally been deemed "subjective" or difficult to quantify. Factors such as fatigue, morale, injuries, and lineups could potentially offer measurable positive contributions to the dataset, enhancing the reliability of the models.

Optimizing Dataset Composition Future revisions of this research should aim to scrutinize the existing feature set more rigorously. A thorough analysis of feature importance or ANOVA filtering, along with an examination of multicollinearity, could yield more effective models or, at a minimum, models that are more interpretable. Different versions of the dataset may be tested to evaluate their performances. For instance, future studies might exclude numerical historical statistics and focus solely on the differential ones or vice versa to eliminate a strong multicollinearity component, and to determine if this adjustment leads to improved results.

Evaluating Alternative Frameworks An alternative approach may involve implementing a different framework for the prediction task. The aim of this research can be achieved not solely through a classification solution, but also through a regression model, whereby the precise number of shots for each team is forecasted and then compared. While a classification solution has been deemed more efficacious and practical, exploring its confrontation with an alternative configuration warrants consideration.

Assessing Model Profitability The primary practical application of the models developed in this study is, undeniably, within the domain of betting. However, evaluating the reliability of these models for betting purposes necessitates an analysis of their profitability in a real-world simulation. Unfortunately, this goal cannot be reached in the present work. Despite the existence of freely accessible online data concerning odds for match outcomes or popular betting markets such as Over/Under 2.5 Goals, there is currently a lack of publicly available data for newer markets, such as shot outcome. Consequently, it cannot be certaintly asserted that the models developed, although well-calibrated and outperforming designated baseline strategies, will be profitable in practical application.
Future research, contingent upon the availability of historical data for shot dominance odds, should concentrate on formulating practical betting strategies based on these models, implementing them, and subsequently evaluating their profitability. In scenarios where odds data for shot dominance is still unavailable, one hypothetical approach would be to infer these odds from other accessible odds that exhibit moderate to strong correlations, such as match outcomes.

Above all, my hope is that this work will serve as an inspiration for authors that love to explore the fascinating world of football modeling. Over the years, the complexity of accessible data and developed models has increased, along with the

opportunities, tools, and markets that bettors can utilize to implement profitable strategies.
I encourage to try to diverge from the traditional paradigm of football analytics, which often fixates on repeatedly predicting a single variable without acknowledging that it is very hard to fully grasp the complexity of these outcomes, a factor that perhaps represents the main reason why supporters around the world are so fascinated by this sport.

Exploring new possibilities and pioneering new frontiers in this field might reveal an approach which may be more efficient and, at the same time, engaging.

Bibliography

Angelini, G., & De Angelis, L. (2017). Parx model for football match predictions. *Journal of Forecasting*, *36*(7), 795–807.

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research*, *13*, 281–305.

Berrar, D., Lopes, P., & Dubitzky, W. (2019). Incorporating domain knowledge in machine learning for soccer outcome prediction. *Machine Learning*, *108*. https://doi.org/10.1007/s10994-018-5747-8

Berrar, D., Lopes, P., & Dubitzky, W. (2024). A data- and knowledge-driven framework for developing machine learning models to predict soccer match outcomes. *Machine Learning*, *113*, 8165–8204. https://doi.org/10.1007/s10994-024-06625-9

Bishop, C. (2007, October). *Pattern recognition and machine learning (information science and statistics)*.

Blagus, R., & Lusa, L. (2013). Smote for high-dimensional class-imbalanced data. *BMC Bioinformatics*, *14*, 106. https://doi.org/10.1186/1471-2105-14-106

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

Breiman, L., Friedman, J., Olshen, R. A., & Stone, C. J. (1984). *Classification and regression trees* (1st ed.). Chapman; Hall/CRC. https://doi.org/10.1201/9781315139470

Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, *78*(1), 1–3. https://doi.org/10.1175/1520-0493(1950)078<0001:VOFEIT>2.0.CO;2

Bunker, R., & Susnjak, T. (2022). The application of machine learning techniques for predicting match results in team sport: A review. *Journal of Artificial Intelligence Research*, *73*, 1285–1322. https://doi.org/10.1613/jair.1.13509

Bystroński, M., Hołysz, M., Piotrowski, G., Chawla, N., & Kajdanowicz, T. (2025, May). *Smotext: Smote meets large language models*. https://doi.org/10.48550/arXiv.2505.13434

Castellano, J., Casamichana, D., & Peñas, C. (2012). The use of match statistics that discriminate between successful and unsuccessful soccer teams. *Journal of human kinetics*, *31*, 139–47. https://doi.org/10.2478/v10078-012-0015-7

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357. https://doi.org/10.1613/jair.953

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

Cranmer, S. J., & Desmarais, B. A. (2016). What can we learn from predictive modeling? https://arxiv.org/abs/1612.05844

Deloitte. (2025, June 12). *Annual review of football finance 2025*.

Dip, A. D., Rahman, N., & Ahmed, M. (2024). Predicting football match results: An analysis of feature selection and machine learning techniques using a curated dataset. *2024 IEEE International Conference on Power, Electrical, Electronics and Industrial Applications (PEEIACON)*, 927–932. https://doi.org/10.1109/PEEIACON63629.2024.10800599

Dixon, M. J., & Coles, S. G. (1997). Modelling association football scores and inefficiencies in the football betting market. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, *46*(2), 265–280. https://doi.org/https://doi.org/10.1111/1467-9876.00065

Elreedy, D., Atiya, A., & Kamalov, F. (2023). A theoretical distribution analysis of synthetic minority oversampling technique (smote) for imbalanced learning. *Machine Learning*, *113*. https://doi.org/10.1007/s10994-022-06296-4

Emerson, R. W. (2022). Anova assumptions. *Journal of Visual Impairment & Blindness*, *116*(4), 585–586. https://doi.org/10.1177/0145482X221124187

FIFA. (2020). Fifa vision 2020–2023.

Friedman, J. (2000). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, *29*. https://doi.org/10.1214/aos/1013203451

Galekwa, R. M., Tshimula, J. M., Tajeuna, E. G., & Kyandoghere, K. (2024). A systematic review of machine learning in sports betting: Techniques, challenges, and future directions. https://arxiv.org/abs/2410.21484

Gneiting, T., & Raftery, A. E. (2007). Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, *102*(477), 359–378. https://doi.org/10.1198/016214506000001437

Goodfellow, I. J., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Groll, A., Schauberger, G., & Tutz, G. (2015). Prediction of major international soccer tournaments based on team-specific regularized poisson regression. *Statistical Modelling*, *15*(6), 485–503.

Hastie, T., Tibshirani, R., & Friedman, J. (2009, February). *The elements of statistical learning: Data mining, inference, and prediction, second edition (springer series in statistics)*.

Ho, T. K. (1995). Random decision forest. *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278–282.

Hochkirchen, T. (2008). Design and analysis of experiments, vol. i, introduction to experimental designs. *Journal of the Royal Statistical Society Series A: Statistics in Society*, *172*(1), 282–282. https://doi.org/10.1111/j.1467-985X.2008.00571_2.x

Hubáček, O., Šír, G., & železný, F. (2021). Forty years of score-based soccer match outcome prediction: An experimental review. *IMA Journal of Management Mathematics*, *33*. https://doi.org/10.1093/imaman/dpab029

Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., & Tang, P. T. P. (2017). On large-batch training for deep learning: Generalization gap and sharp minima. https://arxiv.org/abs/1609.04836

Khamsan, M. M., & Maskat, R. (2019). Handling highly imbalanced output class label. *MALAYSIAN JOURNAL OF COMPUTING*. https://api.semanticscholar.org/CorpusID:214255869

Kingma, D. P., & Ba, J. (2017). Adam: A method for stochastic optimization. https://arxiv.org/abs/1412.6980

Kohavi, R. (2001). A study of cross-validation and bootstrap for accuracy estimation and model selection. *14*.

Lane, R. O. (2025). A comprehensive review of classifier probability calibration metrics. https://arxiv.org/abs/2504.18278

Lauron, M. L. C., & Pabico, J. P. (2016). Improved sampling techniques for learning an imbalanced data set. https://arxiv.org/abs/1601.04756

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. In G. Montavon, G. B. Orr, & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade: Second edition* (pp. 9–48). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-35289-8_3

Louppe, G. (2015). Understanding random forests: From theory to practice. https://arxiv.org/abs/1407.7502

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

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines, 807–814.

Ng, A. Y. (2004). Feature selection, l1 vs. l2 regularization, and rotational invariance. *Proceedings of the Twenty-First International Conference on Machine Learning*, 78. https://doi.org/10.1145/1015330.1015435

Niculescu-Mizil, A., & Caruana, R. (2005). Predicting good probabilities with supervised learning. *Proceedings of the 22nd International Conference on Machine Learning*, 625–632. https://doi.org/10.1145/1102351.1102430

Prechelt, L. (2000). Early stopping - but when? *Lecture Notes in Computer Science*. https://doi.org/10.1007/3-540-49430-8_3

Schumaker, R. P., Solieman, O. K., & Chen, H. (2010). *Sports data mining* (Vol. 26). Springer Science & Business Media.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(60). https://doi.org/10.1186/s40537-019-0197-0

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. https://arxiv.org/abs/1206.2944

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, *15*(1), 1929–1958.

Stübinger, J., Mangold, B., & Knoll, J. (2020). Machine learning in football betting: Prediction of match results based on player characteristics. *Applied Sciences*, *10*(1). https://doi.org/10.3390/app10010046

Sutskever, I., Martens, J., Dahl, G., & Hinton, G. (2013, 17–19 Jun). On the importance of initialization and momentum in deep learning. In S. Dasgupta & D. McAllester (Eds.), *Proceedings of the 30th international conference on machine learning* (pp. 1139–1147, Vol. 28). PMLR. https://proceedings.mlr.press/v28/sutskever13.html

Tibshirani, R., Hastie, T., & Friedman, J. (2010). Regularized paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, *33*. https://doi.org/10.1163/ej.9789004178922.i-328.7

Toloşi, L., & Lengauer, T. (2011). Classification with correlated features: Unreliability of feature ranking and solutions. *Bioinformatics*, *27*(14), 1986–1994. https://doi.org/10.1093/bioinformatics/btr300

Varotto, L. (2023). *Applicazione di modelli di classificazione per la valutazione delle prestazioni nel calcio: Gli expected goals e gli expected points* [Bachelor's thesis]. Dipartimento di Scienze Statistiche, Università degli Studi di Padova [Supervised by Manuela Cattelan, with contribution from Mauro Bernardi].

Wheatcroft, E., & Sienkiewicz, E. (2021). Calibration and hyperparameter tuning in football forecasting with machine learning.

Wheatcroft, E. (2020a). Forecasting football matches by predicting match statistics. https://arxiv.org/abs/2001.09097

Wheatcroft, E. (2020b). A profitable model for predicting the over/under market in football. *International Journal of Forecasting*, *36*(3), 916–932. https://doi.org/https://doi.org/10.1016/j.ijforecast.2019.11.001

Yip, S., Zou, Y., Hung, R. T. H., & Yiu, K. F. C. (2023). Forecasting number of corner kicks taken in association football using compound poisson distribution. https://arxiv.org/abs/2112.13001