Data-driven Sports Forecasting: Analyzing the Profitability of Machine Learning in the Football Betting Industry

Francesco Zonaro — francesco.zonaro@gmail.com

March 2024

All research described in this paper was conducted solely for academic purposes and does not involve real-world application. The algorithm developed in this study is intended for research and experimental use only and will not be deployed for actual gambling activities.

1 Introduction

Pure gambling involves the denial of all systems in the appointment of property, pushing the mind into a world of anarchy where things come upon one, and pass from one miraculously. These were John Hobson's thoughts on gambling, contained in an article written by the 19th-century economist for the Journal of Ethics [10] in 1905. The article entitled *The Ethics of Gambling* provided a thorough analysis of the factors contributing to the ethical nature of an act of gambling and discussed the potential social effects of its widespread diffusion. One interesting consideration made by Hobson lies in the difference between *Pure gambling* and *Mixed Gambling*: the first one involves no skill and requires no prior knowledge, hence the need for the help of a miraculous force, while the latter refers to a series of games of chance in which skills or prior knowledge can provide an advantage over other participants. Notably, just a few games of chance are entirely destitute of skill, even if that skill is solely related to speed or accuracy in calculating chances. From the standpoint of those who do have this additional expertise, the operation ceases to be pure gambling and becomes mixed gambling. This work aims to leverage the evolution in machine learning techniques to develop a machine-learning-based betting architecture able to surpass the concept of pure gambling, as described by Hobson, by introducing elements of skill and strategic decision-making. At the core of the architecture, modern machine learning techniques such as XGBoost and Recurrent Neural Networks are used, leveraging Gated Recurrent Units to enhance the network's capacity to retain dependencies over time. One of the key aspects will be the integration with SHAP, an explanation framework that offers insights into the reasoning behind the models' predictions. This not only enhances transparency but also enables the assessment of the underlying logic behind our predictions. After testing the accuracy of the system in comparison with bookmakers, which represent the current state-of-the-art in the field, further testing was performed to determine whether this architecture could translate into tangible profits.

2 Background

2.1 Modern Betting

In the modern betting industry, the gambler is presented with a set of odds for each match, which differ depending on the chosen bookmaker. A bookmaker is an organization or a person that accepts and pays out bets, on sporting and other events, at agreed-upon odds. When setting these odds, bookmakers consider various factors, such as team performance, injuries and historical data, with the main objective of attracting balanced betting on both sides of an outcome. Odds can be showcased in various formats such as European, American or English. Odds will be assumed to be in European format in the course of this work. European odds represent the potential return on a one-unit bet, including the initial stake, obtained by multiplying the wager by the decimal odds. In the context of a football match, the bookmaker will offer several different *betting markets*, where a betting market is a specific type or category of event, with odds related to each outcome. These markets range from broad forecasts on the trend of a match, for example involving the determination of the minimum number of goals scored during the game, to the exact prediction of the final scoreline, with odds typically reflecting the change in probability between each outcome. It is important to mention that the odds that are presented to the participants do not exactly reflect the outcome probabilities computed by the bookmakers, as they usually propose lower odds to ensure their profit margin, where the margin is effectively the percentage of the total payout that a bookmaker keeps as pure profit. Moreover, the bookmaker regularly adjusts the odds based on all incoming bets, a practice known as *balancing.* This strategy assists the bookmaker in maintaining a profitable margin and mitigates the risk of overexposure to any particular outcome. Moreover, bookmakers do not rely solely on margin and on balancing bets to make a profit, but their prediction accuracy and scientific approach are usually what separates them from the average gambler. Paul & Weinbach, in their 2010 study, concluded that the average gambler acts more as a fan than as an investor, meaning their choices are not as impartial as the ones from the bookmaker [13].

2.2 XGBoost

XGBoost, short for eXtreme Gradient Boosting, stands out as a highly sophisticated and versatile implementation of gradient boosting [4], and it has become a popular choice in various machine learning applications. For a given dataset $D = (X, Y)$, a generic tree ensemble model uses T additive weak learners to predict the output $\hat{y_i}$ as follows:

$$\hat{y_i} = \phi(x_i) = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in F \tag{2.1}$$

where x_i is a single example, f_t is the t-th weak learner added to the model, $\hat{y_i}$ is the prediction for the i-th example and $F = \{f(x) = w_{q(x)}\}$ is the tree space, with q representing the structure of each tree that maps an example to the corresponding leaf index. Each f_t corresponds to an independent tree structure q and leaf weights w. In gradient boosting the idea is to add a new function f at each step, with the objective of minimizing the loss function, accounting for the total ensemble model performances. Moreover, XGBoost adds a regularization term that accounts for model complexity. At step t, the following loss value is computed for each candidate function f_t

$$L^{(t)} = \sum_{i=1}^{n} l(y_i, \hat{y_i}^{(t-1)} + f_t(x_i)) + \Omega(f_t) \tag{2.2}$$

with the regularization term $\Omega(f_t)$ being

$$\Omega(f) = \gamma K + \frac{1}{2}\lambda ||w||^2 \tag{2.3}$$

where K is the number of leaves, w contains the output values of the tree and λ, γ being the regularization tuning parameters. The introduction of a penalty function (regularization term) has the primary goal of discouraging the model from becoming too complex or fitting the noise in the training data. Optimizing the first term, which is the training loss, encourages finding predictive models, while optimizing the second term, which is the regularization term, encourages simpler models. At step t the chosen f_t will be the one that minimizes Equation 2.2.

2.3 Recurrent Neural Networks

A recurrent neural network (RNN) is a type of artificial neural network that is suited to work with sequential data or time series data. This is achieved by incorporating loops within its architecture, enabling the network to retain a memory of previous inputs.

In a feedforward neural network, each layer has its own set of weights that are unique to that layer. When the network processes data, each neuron in a layer receives inputs from all neurons in the previous layer, and each connection has its weight. However, in RNNs, the concept of sharing parameters across layers comes naturally from the recurrence mechanism. In an RNN, the same set of weights and biases are utilized at each time step, or in other words, at every layer of the unfolded network. As shown in Figure 2.1, W represents the shared weight matrix, U represents the input-to-hidden layer weight matrix, and V represents the hidden-to-output layer weight matrix. With this notion, the recurrent connection in an RNN can be represented as

$$h_t = \sigma(W \cdot h_{t-1} + U \cdot x_t + b) \tag{2.4}$$

where h_t is the hidden state at time step t, x_t is the input at time step t, b is the bias term and σ represents the activation function. When training RNNs, Backpropagation Through Time (BPTT) is the algorithm used to calculate gradients and update the shared parameters of the network. Although its principles align with traditional backpropagation, there

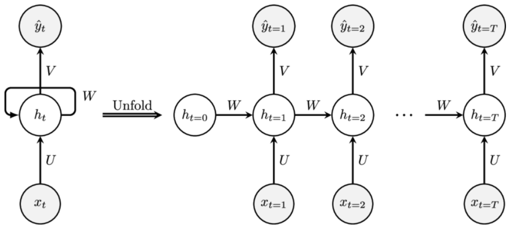

Figure 2.1: The architecture of a folded and unfolded recurrent neural network

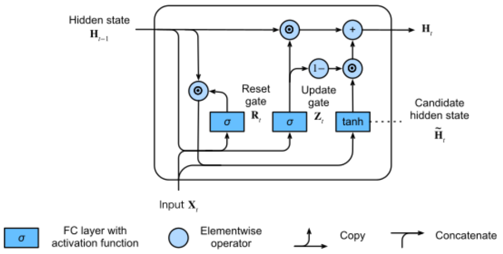

Figure 2.2: Diagram of the gated recurrent unit

are some differences due to the different structure of the network. During the forward pass of BPTT, the RNN processes the input sequence one element at a time, updating its hidden state at each time step based on the current input and the previous hidden state. The final hidden state is then used to generate predictions and compute the loss. During the backward pass, gradients are propagated backward through time to update the parameters of the network. Since the same parameters are shared across all time steps, the gradients from each time step are accumulated and applied to the parameters accordingly.

Standard RNNs suffer from the vanishing gradient problem, which limits their memory capacity and their ability to capture long-term dependencies. Consequently, innovative architectures have emerged: one proposed solution is the Gated Recurrent Units (GRUs) architecture, which incorporates the concept of *gating* to better regulate the flow of information.

2.3.1 Gated Recurrent Units

Gated Recurrent Units (GRUs) are a type of RNN architecture designed to address some of the limitations of traditional RNNs, such as difficulties in learning long-term dependencies. Introduced by Cho et al. in 2014 [5], GRUs have gained popularity due to their simplicity and effectiveness in capturing sequential dependencies in data. At their core, GRUs function similarly to standard RNNs by processing sequential information in a step-by-step manner. However, they incorporate a gating mechanism that enables them to selectively update and reset their internal state, allowing for better handling of long-range dependencies in the input sequence. A GRU unit, shown in Figure 2.2, consists of two main gates: the update gate z_t and the reset gate r_t. These gates control the flow of information within the unit, facilitating the selective updating and resetting of the hidden state. The hidden state h_t at each time step is calculated as a linear combination of the previous hidden state h_{t-1} and the current input x_t, with the update and reset gates determining the influence of each component. The update gate is responsible for determining how much of the previous hidden state should be retained and how much of the new candidate state should be included. It is calculated using a sigmoid activation function:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

where σ is the sigmoid function, W_z is the weight matrix for the update gate, and $[h_{t-1}, x_t]$ represents the concatenation of the previous hidden state and the current input. The reset gate, on the other hand, decides how much of the previous hidden state should be forgotten. It is computed similarly to the update gate:

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

Once the reset and update gates are determined, a candidate hidden state $(\tilde{h}_t)$ is computed using the hyperbolic tangent $(tanh)$ activation function:

$$\tilde{h}_t = tanh(W_h \cdot [r_t \odot h_{t-1}, x_t])$$

where W_h is the weight matrix for the candidate hidden state, $\odot$ denotes element-wise multiplication, and $[r_t \odot h_{t-1}, x_t]$ is the concatenation of the reset-gated previous hidden state and the current input. Finally, the new hidden state (h_t) is a combination of the previous hidden state and the candidate hidden state, controlled by the update gate:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t.$$

GRUs, thanks to their gating mechanism, are capable of selectively retaining or discarding information at each time step. Additionally, their simpler structure compared to other gated architectures like Long Short-Term Memory (LSTM) networks often leads to faster training times and reduced computational requirements [6].

2.4 SHAP

Understanding why a model makes a certain prediction can be as crucial as the prediction itself in many applications, especially those involving medical treatments or financial transactions, as decision-makers cannot blindly rely on a model without insight into its decision-making process. While interpreting simple models like decision trees is straightforward and can be done by following their branches, with complex models a direct interpretation is usually not possible. Among model-agnostic explanation techniques, SHAP, a post-hoc model introduced by Lundberg and Lee in 2017 [12], stands out. SHAP is categorized under the class of additive feature attribution methods and aims to understand the underlying mechanics of the model by performing a series of perturbations. These perturbations involve adjusting features based on a binary mask, where some features remain unaltered, while others are replaced with a neutral background value, typically 0 or the average value from the training dataset. The SHAP model is seen as a linear function of binary variables:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i \tag{2.5}$$

where $z' \in \{0,1\}^M$, M is the number of simplified input features, and $\phi_i \in \mathbb{R}$. In this definition, $g(z')$ represents the explanation model, where z' is a binary vector representing the perturbated input features. In Equation 2.5 ϕ_0 represents the bias term, and $\phi_i z'_i$ represents the contribution of the i-th feature on the model's output. In the system proposed by SHAP, ϕ_i is the Shapley's value associated with the i-th feature. This value directly represents the importance of the i-th feature in determining the final prediction. The higher its associated Shapley's value is, the more important and influential the feature is. Given that the effect of adding or removing a feature depends on the presence of other features as well, all feature subsets $S \subseteq F$, where F is the set of all features, must be explored. As this is infeasible, SHAP approximates this by calculating the contribution of each feature across various combinations of feature subsets. By sampling and averaging the contributions over these subsets, SHAP can accurately estimate the impact of each feature without exhaustively evaluating every possible combination. This approach balances computational feasibility with the need for a comprehensive analysis of feature interactions.

2.5 Related Work

Many studies investigated the strategies of bookmakers and gamblers in sports. Forrest & Simmons in 2000 [8], focused on determining if a statistical model could be more precise than a seasoned expert in determining the outcome of sport events. As expected, experts were found less able to process publicly available data with respect to a statistical model and cases in which experts could be advantaged due to independent knowledge were rare. Continuing this work in 2005 Forrest, Goddard & Simmons [7] determined that bookmakers were on par with statistical models, strengthening the idea of bookmakers acting as investors rather than simply as domain experts. This work was followed by Angelini and De Angelis in 2018 [2], which examined the effectiveness of 41 bookmakers in 11 European major leagues over 11 years. Some of the bookmaker's markets turned out to be inefficient since a trivial strategy of betting on opportunities with odds in a certain range led to positive profit.

Many machine-learning techniques were tested during the two editions of the Soccer Prediction Challenge, an international machine-learning competition that invites the machine-learning community to predict the outcomes of a set of football matches from leagues worldwide. The majority of the techniques were found to perform similarly, with Gradient Boosting and Neural Networks working slightly better than other approaches. The organizers concluded that the most important area of focus should be the modeling of existing knowledge into the ML model [3]. Notably, the dataset used in these competitions contained a large amount of data, from hundreds of leagues worldwide, but for every match only high-level statistics were present, meaning plenty of in-depth information was overlooked. Recently, Hubacek [11] introduced a forecasting system designed to profit from sports-betting markets - specifically in NBA - using machine learning, investigating the need for decorrelation with the bookmaker's odds.

3 ML-based system for outcome prediction

The objective of this work was to train a machine learning model able to predict whether the total number of goals scored in a match would exceed or remain below the 2.5 goals threshold. This outcome, which is usually referred to as the Under 2.5/Over 2.5 (U/O 2.5) betting market by bookmakers, was a sensible choice due to several key factors. Firstly it is widely popular, making it a highly relevant area of study. At the same time, current literature does not explore it in depth, preferring the more captivating question of determining the winning side of a match, rather than the number of goals scored. Furthermore, despite its apparent simplicity, this market presents significant complexity. Bookmakers frequently struggle to accurately predict these outcomes and usually rely on balancing odds to ensure their profit margin rather than leaning heavily towards a specific outcome. One of the central challenges of this task, both for bookmakers and gamblers, lies in the great number of factors that can influence the final result: team dynamics, player injuries or form, weather conditions, opponent tactics and many other unpredictable variables can influence the probability of the two teams scoring more or less goals in a given matchday. Throughout a season, numerous matches anticipated to be high-scoring end up as tedious 0-0 tactical battles, while seemingly dull matches defy expectations with goals being scored from the start. As much as these unexpected twists make the beautiful game so exciting, they make it also extremely unpredictable hence complicating the task at hand.

3.1 Data and Features

Historical data from twelve major football leagues was analyzed over five seasons (2018-2023), namely the English Premier League, EFL Championship, La Liga, Segunda División, Bundesliga, 2. Bundesliga, Serie A, Serie B, Ligue 1, Ligue 2, Eredivisie, and Liga Portuguesa. In general, the number of matches within the data pool was reduced to maintain a high level of quality in all instances. This decision followed from the suggestions of the participants of the 2017 Soccer Prediction Challenge [3], an international machine learning competition that required participants to predict the result of a set of future matches, using the provided set of past match events, containing over 250'000 games, as training data. In the dataset provided by the organizers, each match only contained high-level data: a temporal ID, team names and the number of goals scored by each team. One of the key findings of the competition has been the difficulty of the prediction model in understanding complex football patterns, prompting participants to advocate for enhancing the depth of data available as a potential solution.

On top of the historical match data, an ELO value, obtained from ClubELO and meticulously calculated according to the methodology outlined on their website [1], was associated with each team in each match. The ELO rating provided the analysis with a stable measure of a team's historical quality within its league, allowing to evaluate not only the team itself but also the quality of the opponents that were faced in previous matches. Maintaining a positive form during a tough run of opponents adds more value to the obtained results. Moreover, while recent data is crucial, historical trends are difficult to invert: established teams with a strong record are more likely to maintain consistency and gain momentum as the season unfolds; conversely, newly promoted teams or those with a history of underperformance may struggle to maintain a steady performance level throughout the season. Lastly, utilizing the U/O 2.5 odds provided by the pool of bookmakers tracked by Football-Data.co.uk enabled a comparison of the models' predictions with the current state of the art, as represented by the bookmakers. The task of matching standard names utilized by various sources to a unique team identifier posed a notable challenge during data processing. To address this issue, the TheFuzz python package [2] was used. This package employs a fuzzy string-matching algorithm to obtain the likeness between strings, offering a numerical score indicative of their similarity. In the few instances where the package failed to identify the correct mapping, an additional dictionary was manually created.

3.2 Data Preprocessing

The first modeling choice was related to the size of the window of past matches to consider when computing statistics for the match in focus. When determining the optimal window size for match selection, there are three main factors to keep in mind. The first factor is that the most recent performances are the most important and impactful information that can be given to the model, making a shorter window suitable. On the contrary, though, the more matches the model can observe, directly or indirectly, the better it will be its understanding of the team's dynamics. This is particularly true when considering models working with sequential input, which will contain all the previous match data, rather than the tabular input, where averaging data over many matches may lead to losing sight of the recency importance. The third factor is the *start of season* problem: if the model needs a large number of matches to calculate the input vector, then it will not be usable for a large portion of the season. This problem could be solved by extending the concept of a season, considering the final matches of the previous season as the first matches of the new season, but this option was discarded as the divergence in team performances and overall team quality at the start of a new season can be too large with respect to the previous season. Taking everything into account, a window size of five matches was chosen, as it is flexible in terms of the start of the season problem, yet contains enough information to perform an informed analysis.

3.2.1 Tabular Preprocessing

XGBoost requires data in a tabular format. Let's suppose it is required to compute match data with a five-match window size to obtain the preview data usable for predicting match 8. Due to the start of the season problem, predicting this match would not be possible if a ten-match window size were required. Each performance feature value is computed as the mean of the values from matches 7, 6, 5, 4, and 3, producing a single aggregated row containing the average values.

[1] http://clubelo.com/System

[2] https://github.com/seatgeek/thefuzz

3.2.2 Sequential Preprocessing

The preprocessing steps for the GRU-based system necessitate a fundamentally different approach. Instead of aggregating data into a single row, a matrix of matches is constructed, with each row representing a distinct match. Additionally, all categorical features are transformed into numerical representations. Finally, it is essential to ensure that the numerical values in the data are on a similar scale. This involves normalizing them between -1 and 1 to prevent any disproportionate influence on the model. Note that during tabular preprocessing, the values were not scaled, as XGBoost handles normalization automatically. Once the input data is processed, the matrix must be converted into an appropriate data structure, typically a Tensor. At this point, the preprocessed input sequence is ready to be fed into the neural network model.

3.3 High-Level Training Architecture

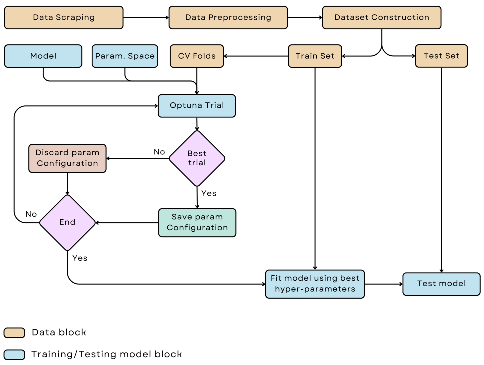

Figure 3.1: High-level view of the training architecture

The training process for the models follows a systematic and iterative approach designed to optimize performance, as described in Figure 3.1. After loading the data, a series of preprocessing steps, are applied to prepare it for compatibility with the selected model architecture. After obtaining the preprocessed data, the dataset is partitioned into distinct training and testing subsets. The training subset is further partitioned into ten folds to facilitate cross-validation. Central to the optimization strategy is Optuna [3], a sophisticated hyperparameter optimization framework [1]. Starting from the cross-validation folds, the chosen model architecture and a predefined parameter space, Optuna performs a series of trials to determine the optimal configuration of hyperparameters. Throughout this iterative process, Optuna dynamically refines its search, iteratively narrowing down the parameter space to identify configurations that result in superior training performances. When a trial surpasses the performance of its predecessors, the corresponding parameter configuration is preserved as the prevailing optimal solution. This iterative process continues until the predefined number of Optuna trials is completed. At this point, Optuna concludes its exploration and the model undergoes retraining using the entire training dataset. The retraining phase serves to achieve peak efficacy, leveraging all the training data. Ultimately, the model performances are evaluated against the previously unseen test dataset.

3.4 XGBoost Based System

When selecting an appropriate model for working with tabular data, XGBoost was chosen for its renowned efficiency in handling such data, thanks to its highly optimized algorithms and parallelization capabilities. XGBoost offers an extensive array of tunable parameters, as listed in its documentation [4], explored using Optuna's automatic hyperparameter search. Notably, the most important parameter is the number of estimators, which corresponds to the number of trees that will compose the final ensemble. As can be seen in the high-level XGBoost-based prediction architecture shown in Figure 3.2, the prediction process for a single instance is straightforward. It's worth noting that instead of producing a single binary outcome (0 or 1), XGBoost offers the possibility of obtaining a more interesting result. Instead of returning discrete labels, the model can provide

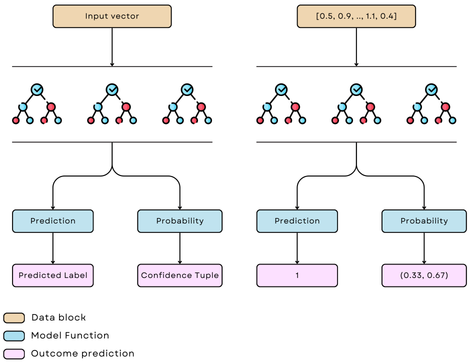

Figure 3.2: High-level view of the XGBoost-based architecture

[3] https://optuna.org/

[4] https://xgboost.readthedocs.io/en/stable/parameter.html

confidence levels for each possible outcome, which are expressed as a tuple. For instance, consider the figure example: the predicted label is 1, but simply knowing this label does not convey the model's level of confidence. By examining the $(0.33, 0.67)$ confidence tuple, where the first value represents the Under 2.5 probability and the second value denotes the likelihood of Over 2.5, it can be concluded that the model reasonably leans towards the latter. A higher confidence level indicates greater certainty, helping to determine when to trust the model's prediction.

3.5 GRU Based System

GRUs, as recurrent neural networks, excel in capturing temporal dependencies and patterns within sequential data. However, this comes with the trade-off of increased computational complexity and time consumption, especially when dealing with large volumes of data. The GRU network was implemented using Keras, a high-level neural networks API running on top of TensorFlow. In the GRU-based system, the prediction process is more convoluted, as can be seen in Figure 3.3. In this scenario, the input represents a sequence of match descriptions organized in a matrix format. Each row of the matrix represents a single match description. The processing involves recurrent steps where the GRU unit handles one match description at a time. As it progresses through the sequence, the GRU updates its hidden state using the current input and its previous hidden state, thereby capturing the temporal dependencies within the match sequence. After the recurrence, a dense layer followed by a softmax function generates the confidence tuple, mirroring the same output obtained with XGBoost. To increase the performance of the GRU model, parameters such as Hidden Size, Dropout, and Recurrent Dropout were optimized, a description of which can be found in the Keras documentation [5].

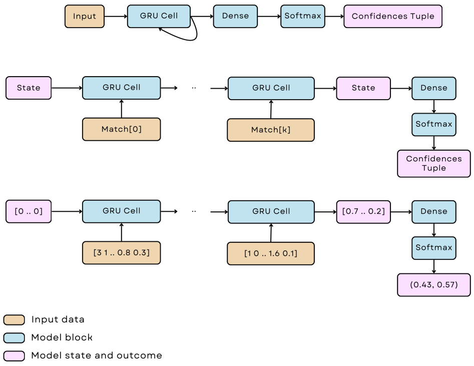

Figure 3.3: High-level view of the GRU-based architecture

3.6 Threshold Algorithm

A properly functioning predictive model should be able to maintain a high level of accuracy on all instances. Therefore, the classic way of assessing its accuracy would involve testing and considering predictions on all available matches. On the other hand, given that the intended focus of this work is on participating in the betting and forecasting game as the gambler rather than organizing it as the bookmaker, there is the advantage of being able to selectively choose which events to concentrate on. Given the financial nature of the task, system reliability is key, therefore the final system will discard predictions supported by a low degree of confidence. This is achieved using a confidence threshold τ under which the system considers the prediction uncertain. Responsible for the filtering is an algorithm called `FINDUNCERTAINTHRESHOLD`, developed in 2023 by Gerevini et Al. [9], as a key component of their methodology to estimate the prognosis of hospitalized patients with COVID-19. The algorithm aims to find an optimal threshold for determining uncertain predictions based on probability scores. Note that using an excessively high τ value means that too many samples (matches) are classified as uncertain, making the model much less useful in generating profit. To avoid this, during the search for an adequate τ value, a maximum fraction of training samples considerable as uncertain is specified. This is controlled with a parameter, called `max_u`. In the experimental analysis, the parameter has been kept at a high value, specifically 0.8 and 0.9, meaning up to respectively 80% and 90% of all predictions could be discarded in each run, as the focus was on getting as many correct predictions as possible rather than relying on beating the bookmaker in terms of general accuracy power, which would be preposterous given the profit margin mechanism explained in Section 2.1, their much longer experience in the domain and higher data availability.

[5] https://keras.io/api/layers/recurrent_layers/gru/

4 Approaches to Sports Wagering

In the world of sports wagering, finding the right strategy is a complex task influenced by factors like risk tolerance or budget availability. This chapter explores various strategies and investigates their relationship with the prediction architecture.

4.1 Problem Definition

Considering a single league, n matches are played each round. In the context of a single match, multiple different markets are offered by the bookmaker: Result, Under/Over, Both teams to score and many others. As it is reasonable to think that a gambler bets on a single market on each match, the choice of the market is itself a problem, as there may be a different degree of profitability between all the favorable odds identified by the gambler. To simplify the problem definition, the betting market is predetermined and coincides with the Under/Over 2.5 market. Supposing each gambler will only bet on a single outcome of the identified market and considering the binary nature of the select U/O 2.5 market, the probability of winning the wager is assigned to p_i and the probability of losing the wager as $1-p_i$. In a generic market, with a multi-class outcome, p_i would be equal to the probability of the wagered outcome happening and $1-p_i$ would be the sum of the probabilities of all the other outcomes happening. In the general formulation, the profit on each bet can then be defined as

$$P_i = \begin{cases} o_i s_i - s_i & \text{with probability } p_i \\ -s_i & \text{with probability } 1-p_i \end{cases} \tag{4.1}$$

where o_i is the odd proposed by the bookmaker for the chosen outcome and s_i is the amount (stake) that the gambler decided to bet. If the gambler wins the bet, then the profit is calculated as the total payout minus the wagered stake. On the other hand, if the gambler loses the bet, the profit is negative and corresponds to the lost stake. Defining as B the budget of the gambler, as min_{bet} the minimum bet accepted by the bookmaker and as s_i the stake for bet i, where i identifies the match on which the bet is placed, the first constraint to which the stake is subject to is

$$min_{bet} \leq s_i \leq B \tag{4.2}$$

which means the stake has to be included between the minimum accepted bet and the maximum budget at the gambler's disposal. Deciding on the actual stake to bet involves a complex balancing act. If the gambler is confident in making a profit, betting the entirety of the budget on the most lucrative odd each time might seem like the logical choice. However, in reality, perfect predictions are rare. Therefore, it is crucial to calibrate each stake with both potential profitability and risk in mind. Several strategies were implemented to test the system in various scenarios. While the model may identify matches where it is confident about the outcome, it does not guarantee that every one of these matches presents a worthwhile betting opportunity. For example, if the odds offered by the bookmaker on a particular match are too low, it may be wiser to avoid betting on that match. This means that a strategy is, in its base form, a function

$$f : (\hat{c_i}, o_i) \rightarrow \{0, 1\} \tag{4.3}$$

that for each pair of confidence estimate $\hat{c_i}$ and odd o_i determines if it is favorable (1) or not favorable (0) to bet on that outcome in that match. The second task that the strategy should perform is managing the gambler's budget to determine the stake of each bet. In many of the tested strategies, both tasks are accomplished simultaneously, proposing a stake that can vary from 0 to a certain maximum amount. Finally, note that in the system the number of simultaneous wagers is limited, due to the generally low number of selected matches. Nevertheless, acknowledging the occasional necessity for placing bets on multiple matches, we have implicitly placed a constraint on the betting strategies: in any case, the strategy is restricted to allocating no more than 10% of its budget to any single bet. This precautionary measure also helps to reduce the potential risk of significant losses resulting from a single model error.

4.2 Betting Architecture

Upon reaching this point, the prediction model, in conjunction with the threshold algorithm, has gen-

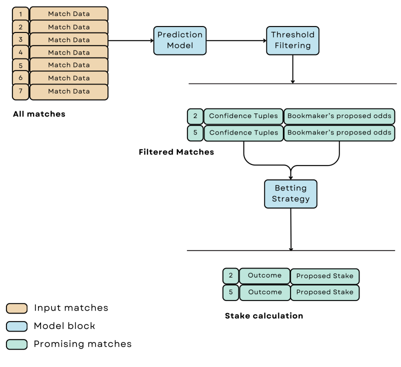

Figure 4.1: High-level view of the betting architecture

erated a collection of matches for which the model should possess a good understanding. The confidence tuple information along with the corresponding odds provided by the bookmaker for each match is given as input to the betting strategy, which produces the stake to bet. The high-level view of this part of the architecture is depicted in Figure 4.1.

4.3 Strategies

Several different strategies were tested, with the objective of determining the effectiveness of the system in various scenarios. Ultimately, each strategy has its advantages and disadvantages, and the choice of strategy may depend on the gambler's risk tolerance, budget size and betting objectives. For each strategy, apart from flat betting where the bet remains fixed, three different betting amounts were determined: `SMALL_BET`, `NORMAL_BET`, `LARGE_BET` all expressed as percentages of the budget. Specifically, `SMALL_BET` corresponds to 3% of the current budget, `NORMAL_BET` to 6% of the current budget and `LARGE_BET` to 9% of the current budget.

Flat Betting

Flat betting is the simplest strategy, where the gambler places the same amount on every bet regardless of the confidence level or the odds proposed by the bookmaker. Other than being extremely simple and easy to manage, it offers an implicit form of risk management. The wagered amount will be typically low, to account for all types of bets, thus mitigating the potential losses resulting from errors made by the prediction model. Of course, it has many limitations. By wagering the same amount across all matches, gamblers may miss out on opportunities to capitalize on particularly favorable odds. While this approach may not always be optimal, partial effectiveness is anticipated due to the thresholded selection process.

Proportional Betting

In its simpler implementation, the concept of proportional betting is similar to flat betting, with the distinction that to be fixed is not the amount itself, but rather the percentage of the budget being wagered. This strategy offers the advantage of automatically adjusting bet sizes according to the gambler's budget. This not only reduces the risk of losing the entire budget but also allows for increased stakes when profits have been realized. In this work, the gambler is supposed to bet 6% of their budget on every bet, corresponding to the `NORMAL_BET` size in the notation. The key idea is that the budget B will vary between matches if they are not concurrent, thereby changing the amount wagered on each match.

Odd based

Another potential approach involves adjusting the betting amount based on the odds proposed by the bookmaker. This strategy allows the gambler to decide whether to allocate more resources to lower odds or higher odds, depending on their chosen risk factor. For example, participants inclined towards lower-risk bets may opt to allocate larger sums on lower odds. This decision derives from the belief that such bets carry a higher probability of success, based on the double assurance provided by the bookmaker's preferred outcome coinciding with the model assessment. Conversely, if participants are inclined to take on more risk in pursuit of potentially higher rewards, they might allocate larger amounts to bets with higher odds. This strategy is based on the notion of seizing opportunities when the model is confident about a match ending with a specific outcome, while the bookmaker may not share the same level of certainty. Both the high-risk and low-risk strategies were tested together with the prediction model.

Confidence Intervals

This strategy involves adjusting the stake size based on the model's confidence for each prediction. Higher confidence levels will lead to larger bets, while lower confidence levels will result in smaller bets. The more reliable the model is in its assessment, the better this strategy will work. To implement it, bets are categorized into different confidence intervals based on their assigned confidence level. These intervals are determined by the minimum and maximum confidence values present in the dataset. For example, if the minimum confidence value is 0.6 and the maximum is 0.8, three gates are established at 0.65, 0.7, and 0.75. A confidence level lower than 0.65 would correspond to a `NO_BET`, confidence between 0.65 and 0.7 would correspond to a `SMALL_BET`, confidence between 0.7 and 0.75 would correspond to a `NORMAL_BET` and confidence higher than 0.75 would correspond to a `LARGE_BET`. In a real-world scenario, when deploying this model, it would be crucial to save the interval levels along with the model and the threshold value, to ensure consistency. Note that in the threshold implementations, roughly 10% or 20% of bets are kept, indicating a conservative selection criterion where all chosen bets hold some degree of value. Consequently, the allocation of betting amounts has been designed with this in mind. Should a larger proportion of bets be retained, it may become necessary to modify the confidence levels, increasing the scope of the `NO_BET` or `SMALL_BET` zone.

Value betting

The final strategy utilizes both odds and confidence information together. This strategy adjusts bets not only based on the perceived likelihood of an event occurring (confidence) but also on the potential return on investment (odds). For bets inside the lowest confidence level (`LOWEST_CONFIDENCE`), no bet is placed regardless of the odds. This conservative approach avoids the risk of less certain outcomes. As confidence increases (`MODEST_CONFIDENCE`), bets are placed but only if the odds are substantial (`HIGH_ODD`). At the next level (`GOOD_CONFIDENCE`), the strategy becomes more aggressive: normal bets are placed on medium size odds and large bets are placed on higher odds. At the highest level (`HIGH_CONFIDENCE`), the strategy suggests placing bets across all odds categories, with the size of the bet increasing with more favorable odds. This approach can be viewed as a hybrid of the opportunity-based and confidence-level approaches, ensuring that bets are only placed when there is a justified expectation of winning and the stake is adjusted for the potential return indicated by the odds.

5 Results

This chapter explores the effectiveness of the models and discusses the efficacy of the threshold algorithm. Specifically, the architecture's performance was evaluated by simulating betting scenarios for the 2022-2023 season. All strategies outlined in Chapter 4 were employed to test the model's effectiveness in conjunction with different approaches. This simulation provided a measure of the model's performance against bookmaker benchmarks, representing the current state of the art.

5.1 Configurations

As already discussed, different configurations were experimented during the testing phase. Across these configurations, one constant element was the use of 10 cross-validation splits for each model evaluation. However, due to the distinct characteristics of decision trees and neural networks, the number of trials was different between XGBoost and the GRU-based model. To ensure fairness, the same amount of time was allocated to train each model, specifically 5 hours. The decision to use a random search for GRUs derives from the relatively low number of executed trials with respect to XGBoost, which would not have allowed Optuna to work properly. Furthermore, tests were conducted on both models with and without the implementation of the thresholding filter. When activating the threshold filter, thresholds of 0.8 and 0.9 were applied, which means the model was capable of excluding up to 80% and 90% of uncertain predictions, respectively. Lower threshold levels were deemed uninteresting for this research.

5.2 Prediction Results

Table 5.1 presents the accuracy results for both the XGBoost and GRU models. It is evident that the thresholded systems significantly outperform the non-thresholded results, as expected. Each result is compared with the bookmaker accuracy on the same set of matches. Models were also trained on different feature subsets to further investigate the necessity for in-depth information. The consistent outperformance of models trained on medium-sized datasets in comparison to those trained on smaller datasets, and their proximity to those trained on full-sized datasets, highlights the effectiveness of utilizing a medium-sized dataset, which strikes the right balance between granularity of information and system efficiency. Notably, this distinction diminishes in the absence of a threshold, where model performances converge. When compared against bookmakers, models without the thresholding filter exhibit a substantial deviation from the bookmaker's accuracy. Conversely, thresholded systems generally match or closely approach the bookmaker's performance, often within a margin of less than 1%. Moreover, within both the XGBoost-based and GRU-based models, at least one configuration demonstrates superior performance compared to the bookmaker's predictions,

Model	**Small**		**Medium**		**Full**	
	Model	**Bookie**	**Model**	**Bookie**	**Model**	**Bookie**
XGB - No Threshold	56.64%	**59.21%**	56.28%	**59.21%**	56.27%	**59.21%**
XGB - Threshold 0.8	64.24%	**64.37%**	64.48%	**65.36%**	64.14%	**64.95%**
XGB - Threshold 0.9	65.19%	**65.49%**	67.99%	**68.84%**	**68.46%**	67.45%
GRU - No Threshold	55.80%	**59.21%**	56.18%	**59.21%**	56.15%	**59.21%**
GRU - Threshold 0.8	63.57%	**63.94%**	**67.06%**	66.04%	**65.02%**	64.33%
GRU - Threshold 0.9	61.84%	**62.26%**	**68.51%**	68.51%	**67.64%**	67.37%

Table 5.1: The accuracy performance of each model, compared with the bookmaker's accuracy on the same set of matches

Model	**Accuracy**	**Precision**	**Recall**	**F1 Score**	**Retained**
XGB - No Thresh	56.28%	54.00%	61.00%	57.00%	100%
XGB - Thresh. 0.8	64.48%	64.00%	85.00%	73.00%	17.50%
XGB - Thresh. 0.9	67.99%	66.00%	**93.00%**	**77.00%**	9.00%
GRU - No Thresh.	56.18%	56.00%	38.00%	45.00%	100%
GRU - Thresh. 0.8	67.06%	69.00%	55.00%	61.00%	15.00%
GRU - Thresh. 0.9	**68.51%**	**73.00%**	51.00%	60.00%	7.86%

Table 5.2: The performance of all the models trained on the medium-size dataset in terms of accuracy, precision, recall, f1 score and the percentage of matches retained after the threshold filter

achieving a margin of up to one percentage point.

In Table 5.2 more in-depth metrics were explored for models trained on the always reliable medium-size dataset. The quality varies notably across different threshold levels. When no threshold is applied, the models demonstrate moderate accuracy, precision, recall, and F1 score. However, as the threshold increases, there is a consistent improvement in every metric, indicating better overall prediction quality. The final column of the table underscores the influence of the threshold system on the number of matches deemed worthwhile for wagering. It is evident that the performance gains achieved through higher thresholds come at the expense of significantly reducing the number of matches on which the system suggests to bet. This reduction is particularly noticeable in the best-performing models, where the percentage of matches kept drops as low as 7.86% on the test set. In terms of accuracy, GRUs appear to have a small advantage over XGBoost. However, delving into deeper metrics, XGBoost emerges as the more reliable option overall. This discrepancy could be attributed to the architecture of the GRU model, which likely necessitates more time and a larger volume of data for adequate training.

5.3 Betting Simulation

To assess the potential profits the models could have yielded during the 2022/2023 season, a simple simulation was designed. Firstly, the starting balance is set to 100 units. Next, the algorithm begins iterating through each match, representing a betting opportunity. For each game, it checks if the current balance is positive. If not, it stops further betting and declares the simulation ended with a complete loss of the budget. For each match, it calculates the amount to wager, which will depend on the current tested betting strategy, the confidence of the model and the proposed odd. Subsequently, the algorithm checks if the model's prediction matches the true outcome of the game. If it does, it calculates the profit based on the wagered stake and the bookmaker's odd. Conversely, if the outcome does not match the model prediction, the profit is determined as the negative value of the wagered amount. After processing each match, it updates the balance based on the calculated profit or loss. Finally, it returns the final balance minus the starting balance: if the returned value is positive, it indicates a profit, while a negative value indicates a loss.

Table 5.3 shows the effectiveness of each strategy in optimizing the final profit, expressed as a percentage of the initial budget. A profit of -100% indicates a complete loss of the budget, while a profit of 100% would signify a doubling of the initial budget and 50% would signify a half increase, raising the budget to 1.5 times its original amount. The absence of the threshold filter is immediately reflected by a lack of profitability across most strategies employed on both models, primarily due to the low accuracy of the models and their significant disparity with the bookmaker's prediction accuracy. Not only are the models inaccurate, but they are also playing against a superior opponent. Among the strategies, the value strategy stands out as the only one demonstrating resilience, managing to generate substantial profits when applied to the XGBoost model and almost breaking even when applied to the GRU model. It's worth noting that this strategy inherently incorporates a form of thresholding by determining bet sizes based on confidence levels and bookmaker's odds, which explains its ability to generate a profit. The importance of the threshold is confirmed when observing the performance of the Confidence strategy as well, which is somewhat able to avoid a complete loss of the budget employing a similar mechanism. This hints at the necessity for a higher threshold, as both strategies demonstrate effectiveness by discarding bets in the lower confidence interval while wagering on those on which the model seems more confident. Testing the XGBoost model trained with a 0.9 threshold reveals that almost all strategies yield a positive profit or manage to break even, which indicates the robustness of the overall architecture. However, it is intriguing to observe that in this scenario, the most effective strategy is the one centered on flat betting. While simple, this approach proves logical when dealing with pre-filtered, advantageous bets. By consistently wagering a fixed amount, the need for additional assessments is avoided. At the same time, the Value and Confidence strategy struggle, mainly due to the excessive removal of bets resulting from the application of a double threshold.

When considering the GRU architecture, applying a 0.8 threshold already significantly enhances the overall results, ensuring positive profits across almost all strategies. Notably, the flat betting strategy maintains its robust performance and the opportunity-based strategy also yields promising outcomes. The superiority of this specific model aligns with the model's accuracy results, considering it surpasses the bookmaker's prediction accuracy by 1.5%

Model	**Fix**	**Prop**	**Conf**	**Cons**	**Opp**	**Value**
XGB - No Threshold (Mid)	-100%	-100%	-80%	-100%	-100%	**101%**
XGB - Threshold 0.8 (Mid)	2%	-49%	**36%**	-58%	-47%	20%
XGB - Threshold 0.9 (Mid)	**42%**	19%	-6%	35%	-2%	-13%
GRU - No Threshold (Mid)	-100%	-100%	-36%	-100%	-100%	**-11%**
GRU - Threshold 0.8 (Mid)	**82%**	51%	0%	14%	80%	-12%
GRU - Threshold 0.9 (Mid)	**21%**	-1%	4%	-5%	-2%	3%

Table 5.3: The performance of the models trained on the medium-size dataset in terms of generated profit

and is the only model trained on the medium size dataset to do so. Using a 0.9 threshold leads instead to diminished performances, due to the reduced volume of bets, even though the overall results remain positive. As observed with XGBoost, employing a flat betting strategy still emerges as the most effective approach with this threshold. Ultimately, these findings highlight the adaptability and effectiveness of the overall architecture, showcasing its ability to generate profits across various scenarios and strategies.

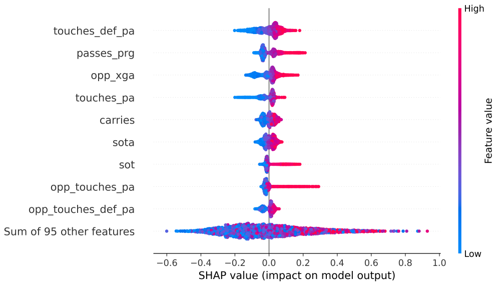

Figure 6.1: SHAP beeswarm plot of the top influential features on average

6 Model Interpretation and Practical Usage

This chapter discusses the use of model interpretability techniques during both the development and usage phases. Additionally, it provides an overview of the process followed when applying the architecture in practice.

6.1 Model Intepretation

One of the distinguishing features of the XGBoost-based architecture is its integration with SHAP (SHapley Additive exPlanations) [1] [12]. By leveraging concepts from game theory, SHAP provides insights into each feature's contribution to the final prediction as explained in Section 2.4. It is crucial to remember that Shapley's values do not directly assess the model's performance but rather provide insights into its decision-making process. However, understanding the rationale behind the model's predictions is as invaluable as having a reliable model. The information offered by SHAP allowed us to discern whether the model's reasoning aligned with human intuition, offering reassurance during the development phase and valuable insights during practical use. This transparency opens the door for the architecture to serve also as a supportive tool rather than operating autonomously, as gamblers can combine their knowledge with the model's output.

Insightful information that was often examined during development can be obtained using SHAP's beeswarm plot, depicted in Figure 6.1. This plot is designed to display an information-dense summary of how the top features in a dataset impact the model's output. Each instance's explanation is represented by a single dot positioned along each feature row, with the x-coordinate of the dot representing the importance of that feature for that instance. The density of dots along each row illustrates the distribution of Shapley's values. Additionally, color is utilized to quickly convey an intuition of the original value of the feature. By default, the features are arranged based on their mean absolute Shapley's value. Upon examination, it becomes apparent that for one of the best performing XGBoost models - specifically the one trained with the threshold set at 0.9 on the medium-sized dataset - the metrics associated with events in the penalty area - the features ending with *pa* - carry substantial importance, whether they are performed or conceded to opponents. This observation is in line with logical reasoning, as a greater amount of time spent in either penalty area generally indicates more potential for goal-scoring opportunities. However, it is important to note that this ordering prioritizes broad average impact over features with rare yet high-magnitude impacts.

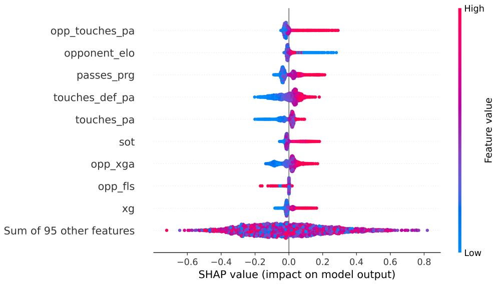

Figure 6.2: SHAP beeswarm plot of the features with the highest magnitude impact

The plot in Figure 6.2 showcases the features that result in the most significant influence on the model's decision when they assume either high or low values. The frequency of touches within the penalty area remains consistently represented, underscoring its sig-

[1] https://github.com/shap/shap

nificance. As previously highlighted within this work, a notable correlation emerges between the frequency of fouls and the likelihood of a match concluding with fewer goals. This association is rooted in the premise that highly physical matches often result in fewer scoring opportunities. There is an interesting relationship between the opponent's ELO and the final outcome, as the model suggests that if the opponent has a particularly low ELO value, there is a higher likelihood of the match resulting in more goals. This assertion aligns with the common observation that weaker teams are typically more susceptible to conceding multiple goals, thus leading to higher-scoring matches. Additionally, xG is featured in both explanations: in the first plot as the opponent's conceded xG (`opp_xga`), and in the second plot as the xG produced by the home team (`xG`). The dependency plot in Figure 6.3 highlights how Shapley's values change based on the xG produced by the team (on the x-axis) and the xG conceded by the opponent, represented by the color of the dot. When analyzing this relationship, distinct intervals can be observed. Essentially, they can be divided into three categories: xG below 1.5, between 1.5 and 2, and above 2. In the range where xG is below 1.5, the `xg` feature importance tends to be either zero or negative and it is interesting to note that in this range the importance is lower when the `opp_xga` is higher. This could be attributed to teams with lower xG struggling to convert opportunities, hence failing to capitalize on the conceded opponent xGA. Additionally, a low team xG and high opponent xGA suggest weakness on both sides, potentially leading to a low-scoring match. As the xG value increases, the dependency on the opponent xGA stabilizes and the importance of the feature increases.

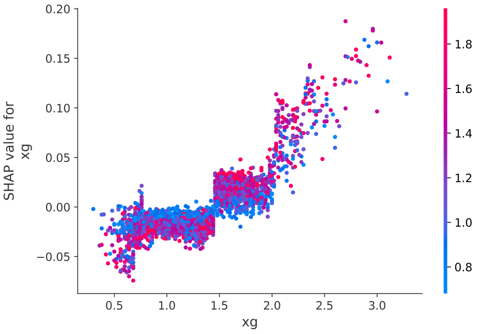

Figure 6.3: SHAP dependency plot of the correlation between the xg and the opp_xga features

These plots played a fundamental role in both the dataset construction phase and the evaluation of the models. During the dataset construction, they guided the selection process by highlighting features consistently utilized by the models, allowing us to retain only relevant and impactful features in the smaller datasets. Furthermore, by observing the distribution of feature importances across instances, it was verified that the model was not identifying any anomalous or illogical pattern within the data. Furthermore, to enrich the gambler's decision-making process, SHAP was used to generate a visual representation of the rationale behind a prediction, specifically constructing a local explanation through the use of a waterfall plot delineating the impact of the features on the predicted outcome, as in Figure 6.4, providing a clear understanding of how the model arrives at its conclusion. In the depicted plot, it becomes evident that the model leans towards predicting an Over 2.5 outcome, driven by a sequence of influential factors. Notably, attributes such as the opponent's significant goal-scoring opportunities conceded (`opp_xga`) and shots on target conceded (`sota`), in combination with the substantial xG created by the home's team (`xG`) suggest an open and entertaining game. Additionally, the high count of touches in the penalty area implies quick ball movement from one goal area to another, suggesting a dynamic match where play predominantly occurs within the penalty areas rather than being concentrated in the midfield. This visualization will be produced together with each prediction made by the model on future matches.

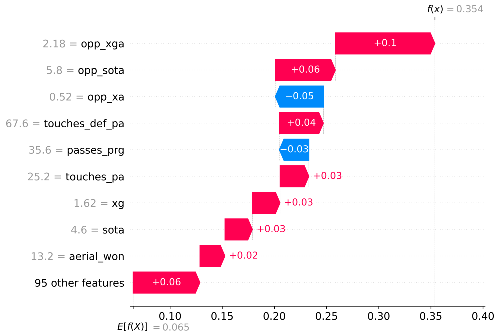

Figure 6.4: SHAP local waterfall plot explaining the reasoning behind a prediction

Ultimately, using both their expertise and the insights provided by the model, the gambler can make informed decisions about whether to act upon the model's recommendations when placing bets or exercise caution.

7 Conclusions and Future Works

These findings highlight the significant utility of machine learning in the field of sports betting and forecasting, in accordance with the current literature on the argument. Both approaches are able to closely match and sometimes exceed the performance of the bookmaker's state-of-the-art predictions when employing a threshold filtering technique, which selectively retains predictions in which the models exhibit higher levels of confidence. This suggests that the models have the potential to generate profits, as confirmed by the outcomes of the simulation conducted using data from the 2022/2023 season. During this period, both the flat betting strategy, where a fixed amount is wagered on the selected matches, and the value betting strategy, aimed at adjusting stake amounts based on the confidence shown by the model and the available odds, demonstrated the ability to generate profit. In comparing the two explored approaches, XGBoost demonstrated greater flexibility in terms of implementation, testing efficiency and compatibility with additional technologies such as SHAP. Conversely, GRUs, with their capacity to autonomously analyze and aggregate sequential data, demonstrated an advantage in accuracy, resulting in slightly superior performances even with limited resources. On this matter, the quantity of publicly available data is fairly limited. However, with access to regular data feeds from sports data providers, there is a reasonable potential for enhancing each model's performance. Additionally, the highly productive football analytics community continually develops groundbreaking metrics, further contributing to potential improvements.

Bibliography

[1] Takuya Akiba et al. "Optuna: A Next-Generation Hyperparameter Optimization Framework". In: *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2019, pp. 2623–2631.

[2] Giovanni Angelini and Luca De Angelis. "Efficiency of online football betting markets". In: *International Journal of Forecasting* 35.2 (2019), pp. 712–721. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2018.07.008. URL: https://www.sciencedirect.com/science/article/pii/S0169207018301134.

[3] Daniel Berrar, Philippe Lopes, and Werner Dubitzky. "Incorporating domain knowledge in machine learning for soccer outcome prediction". In: *Machine Learning* 108.1 (Jan. 2019), pp. 97–126. ISSN: 1573-0565. DOI: 10.1007/s10994-018-5747-8. URL: https://doi.org/10.1007/s10994-018-5747-8.

[4] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *CoRR* abs/1603.02754 (2016). arXiv: 1603.02754. URL: http://arxiv.org/abs/1603.02754.

[5] Kyunghyun Cho et al. "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078 (2014). arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

[6] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014). arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

[7] David Forrest, John Goddard, and Robert Simmons. "Odds-setters as forecasters: The case of English football". In: *International Journal of Forecasting* 21.3 (2005), pp. 551–564. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2005.03.003. URL: https://www.sciencedirect.com/science/article/pii/S0169207005000300.

[8] David Forrest and Robert Simmons. "Forecasting sport: the behaviour and performance of football tipsters". In: *International Journal of Forecasting* 16.3 (2000), pp. 317–331. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(00)00050-9. URL: https://www.sciencedirect.com/science/article/pii/S0169207000000509.

[9] Alfonso Emilio Gerevini et al. "Machine Learning Techniques for Prognosis Estimation and Knowledge Discovery From Lab Test Results With Application to the COVID-19 Emergency". In: *IEEE Access* 11 (2023), pp. 83905–83933. DOI: 10.1109/ACCESS.2023.3296260. URL: https://doi.org/10.1109/ACCESS.2023.3296260.

[10] John A. Hobson. "The Ethics of Gambling". In: *International Journal of Ethics* 15.2 (1905), pp. 135–148. ISSN: 1526422X. URL: http://www.jstor.org/stable/2376380 (visited on 01/05/2024).

[11] Ondřej Hubáček, Gustav Šourek, and Filip Železný. "Exploiting sports-betting market using machine learning". In: *International Journal of Forecasting* 35.2 (2019), pp. 783–796. ISSN: 0169-2070. DOI: https://doi.org/10.1016/j.ijforecast.2019.01.001. URL: https://www.sciencedirect.com/science/article/pii/S016920701930007X.

[12] Scott M. Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *CoRR* abs/1705.07874 (2017). arXiv: 1705.07874. URL: http://arxiv.org/abs/1705.07874.

[13] Rodney J. Paul and Andrew P. Weinbach. "The determinants of betting volume for sports in North America: evidence of sports betting as consumption in the NBA and NHL". English. In: *International Journal of Sport Finance* 5 (May 2010). Article, pp. 128+. ISSN: 15586235. URL: https://link.gale.com/apps/doc/A323349952/AONE?u=googlescholar&sid=bookmark-AONE&xid=c17bfd5c.