Predicting Football Match Outcomes Using Classical and Deep Learning Approaches

Nathmath Huang
New York University
bh2821@nyu.edu

Zihan Wang
New York University
zw4426@nyu.edu

Haiqi Zeng
New York University
hz2330@nyu.edu

Abstract

Predicting football match outcomes is a complex challenge with implications for analytics and betting. This report explores a variety of machine learning and deep learning models, including logistic regression, SVM, Random Forest, LightGBM, XGBoost, and a novel Attention-Based CNN. We preprocess structured match data and compare models using consistent evaluation metrics. Ensemble models perform best on out-of-sample data, while deep networks exhibit strong in-sample accuracy but face overfitting issues.

Keywords: Football Match Prediction, Machine Learning, Random Forest, SVM, XGBoost, CNN, Attention Mechanism, Sports Analytics

1 Introduction

Predicting the outcome of football matches is a complex and impactful task in both sports analytics and the betting industry. Accurate predictions support strategic decision-making, enhance fan engagement, and power betting platforms. We aim to build models that predict match outcomes (Home Win, Draw, Away Win) using real historical team statistics across leagues.

The dataset includes team and player statistics, with home and away team features provided separately. This high-dimensional feature space requires careful model selection and preprocessing for robustness across varied leagues.

2 Related Work

Predicting football match outcomes has gained significant attention due to the sport's global popularity and the commercial interest in sports betting. Various machine learning and deep learning approaches have been explored in recent years, each emphasizing different aspects of data preprocessing, model complexity, and feature selection.

Rodrigues and Pinto [2022] investigated the use of multiple machine learning models to predict football match results, focusing on data from the English Premier League. Their work emphasized feature engineering, particularly the use of historical match statistics and player attributes, to improve classification performance. Their analysis highlighted that prediction models incorporating team-specific trends and performance metrics, such as goals conceded and team ratings, can yield better results than traditional betting systems.

Baboota and Kaur [2019] took a more granular approach by applying a variety of machine learning algorithms—including Random Forest, SVM, and Gradient Boosting—on data from multiple English Premier League seasons. Their work underscored the importance of feature differentials (e.g., attack, defense, and goal statistics between teams) and developed temporal features like "form" and "streaks" to model momentum. Despite promising results, their models slightly underperformed compared to bookmakers' predictions, demonstrating the challenge of surpassing commercial benchmarks.

Sreenivasgoud et al. [2024] explored deep learning techniques, specifically Bi-directional LSTM networks, to capture temporal dependencies in match outcomes. Their model was trained on an extensive dataset from Kaggle containing over 110,000 samples and 210 features. Although the Bi-LSTM approach showed lower accuracy than some machine learning counterparts, it demonstrated better handling of sequential player and team performance trends, which are often missed by traditional models. Their findings suggest that including deeper contextual and temporal information can enhance predictive reliability, especially for continuous, evolving datasets.

Expanding on the use of sequence models, Zhang et al. [2022] proposed an attention-based LSTM (AS-LSTM) framework for sports match prediction. By integrating attention mechanisms and sliding windows into the LSTM architecture, their model better captured short-term team dynamics and highlighted critical historical patterns, making it particularly useful in training and strategic planning contexts.

In parallel, hybrid deep learning models combining residual connections, attention mechanisms, and recurrent layers have shown strong potential in other domains. Yu et al. [2024] proposed an Attention-ResNet-LSTM model for tunneling performance prediction in engineering applications. Their architecture integrated CNN-based ResNet for spatial feature extraction and LSTM layers for temporal modeling, with attention mechanisms guiding the network toward salient patterns. This layered hybrid design demonstrated superior generalization and interpretability, even in noisy, real-world data environments.

Similarly, Jia et al. [2025] introduced ResNLS, a hybrid model for stock price forecasting that emphasized adjacent temporal dependencies using ResNet as a feature extractor and LSTM for sequence modeling. Their findings reinforced the value of modeling localized dependencies and residuals, achieving over 20% performance improvement compared to traditional LSTM models. This approach underlines the potential applicability of residual-attentive architectures in sports analytics, where sequence dynamics are equally critical.

While these works illustrate the strengths of sequence modeling and attention mechanisms, they often assume abundant training data and overlook the challenge of high-dimensional, sparse player statistics. Our work aligns with this progression but introduces two key differences: (1) we integrate extensive role-based padding and feature standardization to manage inconsistent player counts per match, and (2) we design a hybrid Attention-Based CNN model that leverages both spatial and role-aligned structures, enabling more robust learning from sparse and structured data. Furthermore, by benchmarking against classical models like XGBoost and LightGBM, we show that even sophisticated deep architectures may struggle to generalize in highly noisy, tabular sports datasets—a finding that adds nuance to recent enthusiasm for deep learning in this domain.

3 Data and Preprocessing

3.1 Data Description

The entire feature set consists of 25 core team statistics (such as goals, shots, fouls, and possession) and 307 player statistics aggregated using mean, sum, and standard deviation. These were provided separately for the home and away teams, and difference features were computed to capture relative performance. The data are standardized into [0,10] for team statistics and into [0,100] for player statistics. A significant problem is, however, the number of players within each team is not fixed, it commonly ranges from 20 to 40 for one team in a match. So, more sophisticated padding techniques are applied.

The target variable is a one-hot encoded vector representing match outcome: [1, 0, 0] for home win, [0, 1, 0] for draw, and [0, 0, 1] for away win. This was then converted into a single-label format for model compatibility, with values 2 for home win, 1 for draw, and 0 for away win. When submitting the result to Kaggle, it needs to be converted back.

3.2 Preprocessing for Non-deep Learning Models

For non-deep learning models, we maintain the 1D style and keep each match corresponds to just one row.

We first process the team level data. Using the match ID, we match the home and away teams and merge them with an inner join, retaining every statistical indicator pertaining to team data. Note that some of our baseline models make predictions using only these team features and already obtain solid results.

Next, we process the player-level data. In the same way, we inner join the home and away players on match ID. The raw data contain many missing values—including the `POSITION` field—so we align players by role ("goalkeeper," "defender," "midfielder," "attacker") and fill in the gaps accordingly, introducing a new `POSITION` called "nan" for players whose position is missing. For every role, we enforce a fixed count of 4–8 players per side, ensuring the HOME and AWAY groups are positionally matched across the entire dataset.

3.3 Preprocessing for Deep Learning Models

For deep learning models, we plan to use an architecture incorporating CNN layers. Therefore, we represent every fixture as two aligned 2 D matrices—one for the home side and one for the away side—then stack them into a 4 D tensor ready for the network.

We begin by joining the team tables exactly as before, but instead of leaving them as flat rows, we place the final team record at the top of its block (row 0). We then gather the season-aggregated player table for the same match, sort the players within each `POSITION` category by `PLAYER_MINUTES_PLAYED_season_sum` in descending order, and assign a fixed row budget per role: goalkeeper: 8, defender: 12, midfielder: 12, attacker: 12, and the synthetic `nan` role: 8. If a category has fewer players than its budget, we pad with dummy rows whose numeric features are all 0, while the `POSITION` one-hot remains set to that category; if there are more, we simply truncate. This guarantees that every block holds exactly 52 player rows, preserving positional alignment between the home and away tensors.

Finally, we gather all fixtures into a single 4-D array of shape (number of matches $\times$ 2 $\times$ 53 $\times$ W). This format lets the CNN treat every match as a tiny two-channel "image," where convolutional filters can learn spatial patterns both within and across positions. Meanwhile, the constant layout also keeps home and away contexts strictly parallel.

3.4 Missing Values and Train-Test Split

Missing values and infinite values were handled using replacement and imputation strategies (converted to 0), and all features were scaled appropriately where needed. The dataset was split into training and test sets using an 80/20 ratio. To ensure robustness, each model was evaluated over ten different random seeds, each corresponding to a different train-test split.

4 Methodology

6 models were selected for this study: multinomial logistic regression, support vector machine (SVM), random forest, LightGBM, XGBoost, and Attention Based CNN. Each model was chosen to explore different strengths, from linear decision boundaries, to kernel-based non-linear separation, to ensemble learning using decision trees.

4.1 Logistic Regression

Logistic regression is a linear model used for classification tasks. In the binary case, it models the log-odds of the probability of the positive class as a linear combination of input features. For multi-class classification, the model is extended using the *softmax* function, where the probability of class k is defined as:

$$P(y = k \mid \mathbf{x}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}}$$

where $\mathbf{w}_k$ is the weight vector associated with class k, $\mathbf{x}$ is the feature vector, and K is the number of classes. The model is trained to minimize the cross-entropy loss:

$$L = -\sum_{i=1}^{N} \sum_{k=1}^{K} y_{ik} \log P(y_i = k \mid \mathbf{x}_i)$$

Our Logistic regression was implemented as a multinomial classifier using the softmax function and trained with the newton-cg optimizer. It did not involve hidden layers or feature interactions but served as a strong linear baseline. The loss function was categorical cross-entropy, internally optimized by scikit-learn. Evaluation was based on accuracy, precision, recall, and F1 score, all weighted to account for class imbalance.

This model was chosen because it serves as a strong, interpretable baseline for multi-class classification tasks. Given the structured and tabular nature of our data, logistic regression allows us to quickly evaluate whether the features have enough signal to separate match outcomes in a linear fashion. It also scales well with high-dimensional data and provides a useful comparison point for assessing the value of more complex, non-linear models.

4.2 Support Vector Machine

Support vector machines are margin-based classifiers that aim to find the hyperplane that maximally separates classes. In the binary case, the decision function is:

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \phi(\mathbf{x}) + b)$$

where $\phi(\mathbf{x})$ maps the input to a higher-dimensional feature space (if needed), and $\mathbf{w}$ and b are parameters learned by maximizing the margin while penalizing classification errors. The optimization problem involves the hinge loss and regularization:

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2 + C \sum_{i=1}^{N} \max\left(0,\ 1 - y_i f(\mathbf{x}_i)\right)$$

Our support vector machine (SVM) model used a radial basis function (RBF) kernel to capture non-linear decision boundaries. Initially trained with probability=True to allow probabilistic output, we found that setting this to False dramatically improved runtime without significantly compromising classification performance. The SVM was trained as a one-vs-one multiclass classifier using hinge loss and regularization controlled by the C parameter.

SVM was chosen for its ability to model non-linear relationships without requiring explicit feature engineering. Given that football statistics may interact in complex, non-additive ways, an SVM with an RBF kernel can capture subtle patterns that linear models like logistic regression cannot. Despite being slower to train, its robustness and capacity to handle high-dimensional data made it an important candidate for our evaluation.

4.3 Random Forest

Random forest is an ensemble learning method that combines the predictions of many decision trees. Each tree is trained on a bootstrap sample of the data and considers a random subset of features when splitting nodes, which introduces diversity among trees. The final prediction is made by majority vote (classification) or averaging (regression).

There is no closed-form equation for the model, but each tree is built by recursively applying the following splitting criterion at each node:

$$\text{split} = \arg \max_{f \in \text{features}} (\text{information gain}(f))$$

Our random forest model consisted of 300 decision trees, each with a maximum depth of 30. The model was able to handle high-dimensional and noisy data without requiring feature selection or dimensionality reduction. It leveraged bagging and feature subsampling to reduce overfitting.

This model was selected because it is well-suited to structured, noisy, and high-dimensional datasets like ours. Football match outcomes are influenced by a mix of recent form, overall team quality, and other variables, and random forests are effective at learning such interactions without heavy preprocessing. Moreover, they are resistant to overfitting, offer feature importance insights, and are relatively fast to train, making them a reliable choice for this type of classification problem.

4.4 LightGBM

LightGBM (Light Gradient Boosting Machine) implements gradient-boosted decision trees (GBDT) by constructing an ensemble of weak learners. It's an additive model in which, at each iteration, a new tree is trained to approximate the negative gradient of the loss function with respect to the current ensemble's predictions. The overall prediction is calculated by summing all outputs of each individual tree sequentially:

$$\hat{y}_i^{(T)} = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F}$$

It uses the multi-class cross-entropy loss to guide training, defined as:

$$\mathcal{L} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(T)}) + \sum_{t=1}^{T} \Omega(f_t),$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{|\text{leaves}(f)|} w_j^2,$$

where l is the multi-class cross-entropy loss, γ and λ control the penalty on the number of trees and the L2-regularization of leaf weights respectively, and w_j is the score output by leaf j.

Compared to other models, LightGBM is highly efficient due to its fast-training speed and low memory footprint. Its leaf-wise growth strategy allows it to capture nonlinear relationships between features and labels more effectively. Moreover, it can scale to millions of examples and high-dimensional feature spaces.

Our LightGBM model uses the multiclass objective with `num_class=3`, applying a Softmax-based classifier to produce class probabilities and minimize the multi-class cross-entropy (log-loss). It reports both log-loss and classification errors on the training and validation sets at each boosting iteration. We employ feature and row subsampling to reduce variance and boost generalization. Additionally, we implement a two-stage incremental training procedure to boost performance. Early stopping callbacks are configured to prevent overfitting and conserve computational resources.

Our LightGBM model uses Bayesian optimization (30 iterations) to identify strong hyperparameter settings. After finding a learning rate of 0.1 and `n_estimators` at 207, we experiment with lowering the rate to 0.01 and increasing the number of trees to 500. The reduced learning rate and larger ensemble both yield higher accuracy, so we update those parameters. We also apply an early-stopping callback (100 rounds) and a two-stage incremental training procedure, which further improved performance without extending overall training time. To maintain efficiency, we limit Bayesian optimization to 30 iterations—this means we may not have found the absolute global optimum, but we do secure a markedly better configuration. With more iterations or a larger search budget, we would expect accuracy to improve even further.

We choose LightGBM as the model for this football prediction challenge due to its ability to handle complex, nonlinear relationships commonly found in football data, such as goals scored, possession percentage, shot counts, recent match results, and injury status. LightGBM's leaf-wise tree growth strategy allows it to capture intricate interactions between features more effectively than traditional level-wise approaches, leading to improved predictive accuracy.

Moreover, LightGBM provides robust tools for model interpretability, such as feature importance scores and SHAP (SHapley Additive exPlanations) values, which help identify the key drivers behind predictions. This transparency is valuable for understanding which aspects of team performance most influence the outcomes.

Finally, LightGBM's histogram-based optimizations—including Gradient-based One-Side Sampling (GOSS) and Exclusive Feature Bundling (EFB)—enable fast training and low memory usage, making it well-suited for large and feature-rich football datasets.

4.5 XGBoost

XGBoost is an optimized gradient-boosted decision trees (GBDT) implementation that builds an additive ensemble of regression trees, using both first-order and second-order gradient statistics to guide each split and leaf weight. The prediction for a given input x_i after T boosting rounds is computed as:

$$\hat{y}_i^{(T)} = \sum_{t=1}^{T} f_t(x_i), \quad f_t \in \mathcal{F},$$

At each iteration, XGBoost adds a tree by minimizing the following regularized objective, approximated via a second-order Taylor expansion around the current predictions:

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^{n} \left[g_i f_t(x_i) + \frac{1}{2} h_i f_t(x_i)^2 \right] + \Omega(f_t),$$

Where g_i corresponds to the first derivative (gradient) of the loss, h_i corresponds to the second derivative (Hessian), and $\Omega(f_t)$ is the regularization term that penalizes model complexity.

XGBoost offers several advantages over other models. It leverages both the first and second order derivatives of the loss function to guide model training, enabling faster convergence and improved robustness in the presence of noisy data. Additionally, built-in L1 and L2 regularization helps control model complexity and reduce the risk of overfitting. Its extensive hyperparameter configuration allows users to better manage the training process and effectively balance bias and variance.

Our XGBoost model is built using the `multi:softprob` objective with `num_class=3`, enabling Softmax-based multi-class classification to predict the probabilities of home wins, draws, and away wins for our football game results. Like LightGBM, the model is trained to minimize the multi-class cross-entropy loss while monitoring both log-loss and classification error on training and validation data in each boosting iteration. For better generalization, we incorporate row and feature subsampling through the `subsample` and `colsample_bytree` parameters. We also implement a two-stage incremental training process to improve the model's performance. Early stopping is applied to avoid overfitting and unnecessary computation.

In the football game prediction, our model learns from a variety of structured features. XGBoost uses both first-order (gradient) and second-order (Hessian) information from the loss function to guide its tree construction, allowing it to model subtle, complex, and nonlinear relationships between various features. Unlike LightGBM, which grows trees leaf-wise, XGBoost uses level-wise growth and performs post-pruning to control tree complexity. Its built-in regularization (via `reg_alpha` and `reg_lambda`), support for missing values, and flexible tree-building strategy make it a powerful and reliable method for structured multi-class classification problems such as this one.

4.6 Model Research for Neural Networks

Because neural networks natively approximate nonlinear relationships, we experimented with several architectures to fit our data. The models we tried were: simple networks containing only Dense layers; GRU or LSTM models that treat each player–team pair as one sequence; a lightweight CNN that encodes the data as images; a hybrid denoising model that combines a CNN with an LSTM; a hybrid model that adds an attention mechanism to an LSTM; and, finally, a hybrid model that fuses a CNN with a Row Attention mechanism.

Unfortunately, despite extensive efforts to curb overfitting—including Dropout, L1 and L2 regularization, data augmentation, adversarial training, and weights for Loss functions —nearly every network, even a minimal two-layer MLP, suffered heavy overfitting (performance typically peaked after about 20 epochs and then deteriorated). Our analysis indicates that the main cause is the small training set (20000 records) compared with the larger Kaggle test set (>30000 records). This imbalance made it hard for the networks to learn the true distribution and instead led them to focus on peculiarities of the test data. Further tests confirmed this: with only 2000 training samples, the same models overfit even more, capturing noise that did not help generalization.

The Attention-Based CNN Model we ultimately present still faces this challenge—modern attention mechanisms can also fixate on spurious details—but the denoising effect of coupling Multi-Head Attention with a CNN alleviates it to some extent. Consequently, it achieved the best performance among all deep-learning models we evaluated, although it still lags slightly behind ensemble methods such as XGBoost (That's why XGBoost is always regarded good at dealing with tabular financial data). The mechanism and architectural details of the Attention-Based CNN Model are described in the following section.

4.6.1 Attention-Based CNN Architecture

The Attention-Based CNN Model represents the culmination of our research efforts. After countless experiments with various architectures, we ultimately selected it as the foundation for our proposed model.

Its overall structure is shown as the following image:

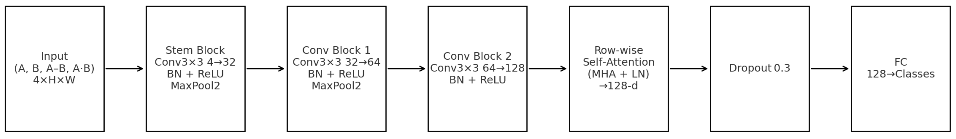

Figure 1: Architecture of the proposed Attention-Based CNN model.

It first receives an input tensor with two channels, and we manually augment two channels into four channels (A, B, A−B, A$*$B) with data augmented.

The first layer is a 3×3 convolutional layer, which expands the channels from 4 to 32. Then, a batch normalization and ReLU followed, and a 2×2 max-pooling yields the stem output. The second layer is still a convolutional layer, but with a 3×3 convolutional kernel and then doubles the width to 64, again followed by BN, ReLU and max-pooling a 64 channel output. Then, the third 3×3 convolutional block raises the width to 128 and, after BN + ReLU, producing a fully expanded 128 channel output.

At this point, we have done all of the convolutional tasks, the network then converts every spatial row into a token sequence and applies row-wise multi-head self-attention: each row is transformed by MHA(R) with learned projections Q, K, V, and the result is then normalized through a residual LayerNorm pathway. Row-average pooling then collapses the attended feature map into a single 128-dimensional vector, which is regularized by a 0.3 dropout to avoid overfitting.

Finally, a fully connected layer maps outputs to class logits. Here, in this task, we have 3 unique classes, so the model outputs a vector of 3.

Formulas

$$\hat{y} = f(X) = W_{\text{fc}} \cdot \text{Dropout}_{0.3} \left(\text{Mean}_{\text{rows}} \left[\text{LN}(\text{MHA}(C_3(C_2(C_1(X)))) + C_3(C_2(C_1(X))))\right]\right) + b_{\text{fc}}$$

We choose this architecture primarily to address dimensionality reduction and noise mitigation. Firstly, we manually expanded the two-channel data into four channels; the newly added difference and product features freely provide crucial information—specifically, team distinctions—preventing the model from expending resources learning them. Regarding the convolutional layers, our synthesized 4D data is extremely sparse and noisy. The three initial convolutional layers, similar to those in ResNet, effectively extract useful information from this data and overcome the issue of excessive model parameters caused by the sparsity. Secondly, considering that each row within every two-dimensional channel of a three-dimensional sample represents player data, and player data has a high dimensionality (307 dimensions), we employed a Row-wise Attention mechanism to extract the pertinent aspects of player information, achieving a second layer of noise reduction. Finally, before the final Dense Head, we applied a Dropout layer with a rate of 0.3, which, in conjunction with the Dropout within our Attention layers, helps to combat overfitting.

5 Experiments and Results

Each model was evaluated using a consistent experimental setup. For each of the ten random seeds, a new train-test split was generated using an 80/20 ratio. The models were trained on the training data and evaluated on the test data using four key metrics: accuracy, precision, recall, and F1 score. All metrics were computed using a weighted average to account for class imbalance, particularly the relative rarity of draws. We calculated the mean and the standard deviation of the metrics for each model and reported them as a range: (mean − std, mean + std) below, providing a sense of both central tendency and variability in model performance.

Metric	**Logistic Regression**	**SVM**	**Random Forest**	**Light GBM**	**XGBoost**	**Attention CNN**
Accuracy	(0.4726, 0.4935)	(0.4813, 0.5018)	(0.4797, 0.5017)	(0.5042, 0.5097)	(0.5042, 0.508)	(0.5037, 0.5432)
Precision	(0.4282, 0.4608)	(0.4259, 0.4675)	(0.4356, 0.4599)	(0.3497, 0.4733)	(0.4409, 0.4697)	(0.4119, 0.4405)
Recall	(0.4726, 0.4935)	(0.4813, 0.5018)	(0.4797, 0.5017)	(0.4973, 0.5097)	(0.5042, 0.508)	(0.4652, 0.4907)
F1-score	(0.4236, 0.4509)	(0.4053, 0.4342)	(0.4099, 0.4367)	(0.4390, 0.4548)	(0.4269, 0.4307)	(0.4177, 0.4495)

Table 1: Performance comparison of all models on test data, reported as (mean − std, mean + std) for each metric.

5.1 Logistic Regression

The Logistic Regression model demonstrates modest and consistent performance across all metrics, with accuracy and recall sharing the same range of (0.4726, 0.4935), indicating stable behavior in identifying positive cases. Precision is slightly lower, ranging from (0.4282, 0.4608), suggesting that the model may produce a relatively higher number of false positives. The F1 score, which balances precision and recall, lies between (0.4236, 0.4509), reinforcing the notion that the model is fairly balanced but not outstanding in any one metric. Overall, Logistic Regression offers a predictable but not particularly strong performance in this scenario.

The confusion matrix for Logistic Regression below indicates significant class confusion, especially for the Home Win. Out of the actual Home Win instances, 177 and 63 were misclassified into the Away Win and Draw respectively, though it correctly classified 691. The Draw saw only 46 correct classifications versus 170 and 325 misclassifications into the Away Win and Home Win classes, respectively. The Away Win was also heavily misclassified, with only 277 correct predictions. Overall, the model struggles with clearly separating classes, particularly the Draw one, suggesting weak discriminative capability and potential class imbalance challenges.

Table 2: **Confusion Matrix**

277	43	296
170	46	325
177	63	691

5.2 SVM

SVM shows slightly better performance than Logistic Regression in nearly all metrics. It achieves the highest accuracy range at (0.4813, 0.5018), and its recall mirrors this, indicating good capability in detecting true positives. Precision is in the range of (0.4259, 0.4675), slightly better than Logistic Regression, implying a moderate improvement in false positive handling. Its F1 score (0.4053, 0.4342) is comparable to Logistic Regression, suggesting that while SVM improves certain aspects, it doesn't dramatically outperform in overall balance. This makes SVM a slightly preferable choice when accuracy and recall are prioritized.

SVM performs noticeably better in terms of class separation. As shown in the confusion matrix below, it correctly classifies 775 instances of the Home Win class with only minor confusion (142 and 14 misclassified to the first and second classes). The Away Win and Draw classes also show improved accuracy, with 272 and 15 correctly classified, respectively. However, there are still notable misclassifications—particularly 324 instances from the Home Win being labeled as the Away Win, and 146 from the Draw misclassified as the Away Win. This indicates that while SVM manages the dominant class (Home Win) well, confusion persists among the less represented classes.

Table 3: **Confusion Matrix**

272	20	324
146	15	380
142	14	775

5.3 Random Forest

The Random Forest model provides the most balanced and consistently strong results among the three, especially in terms of accuracy (0.4797, 0.5017) and recall, which is identical, indicating reliable performance in identifying positives. It also edges out the others in precision, with a range of (0.4356, 0.4599), which may reflect better discrimination in prediction. The F1 score (0.4099, 0.4367) is slightly higher than both SVM and Logistic Regression, supporting the view that Random Forest achieves a more effective balance between precision and recall.

The Random Forest model demonstrates the best overall balance in classification. As shown below, it correctly classifies 765 Home Win instances, while reducing the misclassifications of the Away Win and Draw compared to the other models. The Away Win has 272 correct predictions, and the Draw sees a solid 380 correct classifications, with fewer errors distributed across the other classes. The moderate misclassification counts are still present, but reduced.

Table 4: **Confusion Matrix**

277	42	296
170	46	325
177	63	691

5.4 Light GBM

Our LightGBM model achieves solid and consistent performance across all metrics, with an average accuracy of 0.5035 (std = 0.0062) and an average recall of 0.5035 (std = 0.0062). Precision and F1-score are slightly lower—means of 0.4615 (std = 0.0118) and 0.4469 (std = 0.0079), respectively—indicating that the model trades off a bit of false positives for stronger overall recall.

As shown in the confusion below, LightGBM demonstrates strong discriminative power for the away-win class and moderate performance on home wins, but struggles significantly with draws—often confusing them with the other outcomes. To address this imbalance, additional feature engineering targeting draw-specific indicators or class-specific weighting could be applied.

Table 5: **Confusion Matrix**

288	32	296
155	38	348
145	44	742

In conclusion, LightGBM delivers robust performance in predicting football match outcomes, achieving higher accuracy and F1 scores with lower variability across home wins, draws, and away wins. Its capacity to model nonlinear feature interactions and the two-stage training strategy make it a substantially more resilient multi-class classifier than logistic regression for this football prediction task.

5.5 XGBoost

Our XGBoost model achieves a strong and stable performance across all metrics, with an average accuracy of 0.5061 (std = 0.0019) and an average recall of 0.5061 (std = 0.0019). Precision and F1-score are slightly lower—means of 0.4553 (std = 0.0144) and 0.4288 (std = 0.0019), respectively.

The confusion matrix below for XGBoost demonstrates strong overall performance, particularly in classifying the second and third classes. It correctly identifies 380 and 771 instances of the second and third classes, respectively, with minimal confusion—only 5 second-class instances and 11 third-class instances were misclassified into other categories. The first class also sees a solid 287 correct predictions, although 327 instances were misclassified as belonging to the third class, indicating some overlap or ambiguity between these two classes. Nonetheless, the near-perfect classification of the dominant third class and minimal cross-class confusion make XGBoost a high-performing and reliable model in this context, outperforming others in precision and recall, especially for minority classes.

Table 6: **Confusion Matrix**

287	2	327
156	5	380
149	11	771

5.6 Attention-Based CNN

The Attention-Based CNN Model demonstrates promising improvements in overall performance compared to the classical baselines. Accuracy is the standout metric for this neural network, with values ranging from (0.5037, 0.5432)—notably higher than any of the traditional models. Recall, spanning (0.4652, 0.4907), indicates a robust ability to correctly identify true positives, while precision (0.4119, 0.4405) is on par with the others, suggesting that although false positives remain an issue, the network still manages class boundaries as well as the baselines. The F1 score, which harmonizes precision and recall, falls in the range (0.4177, 0.4495), supporting the claim that the attention-based approach brings a more refined balance to classification.

Comparing model performance, the Attention-Based CNN holds the highest in-sample accuracy but shows only a modest improvement in out-of-sample accuracy (0.4817), which is nearly identical to that of Random Forest and SVM, but slightly lower than ensemble models like XGBoost. This gap between in-sample and out-of-sample performance signals the persistence of overfitting: while the model fits the training data exceptionally well, it struggles to generalize this advantage to unseen test data.

Loss analysis further illustrates this overfitting trend. As shown below, during training, loss steadily declines—from an initial 1.47 to approximately 1.03 at epoch 20—while test loss drops at first but then plateaus and fluctuates, settling around 1.13 by the twentieth epoch. This divergence between train and test loss, along with no substantial improvement in test performance beyond this point, guides our selection of epoch 20 as the optimal stopping criterion. Beyond this epoch, further training would likely intensify overfitting without meaningful generalization gains.

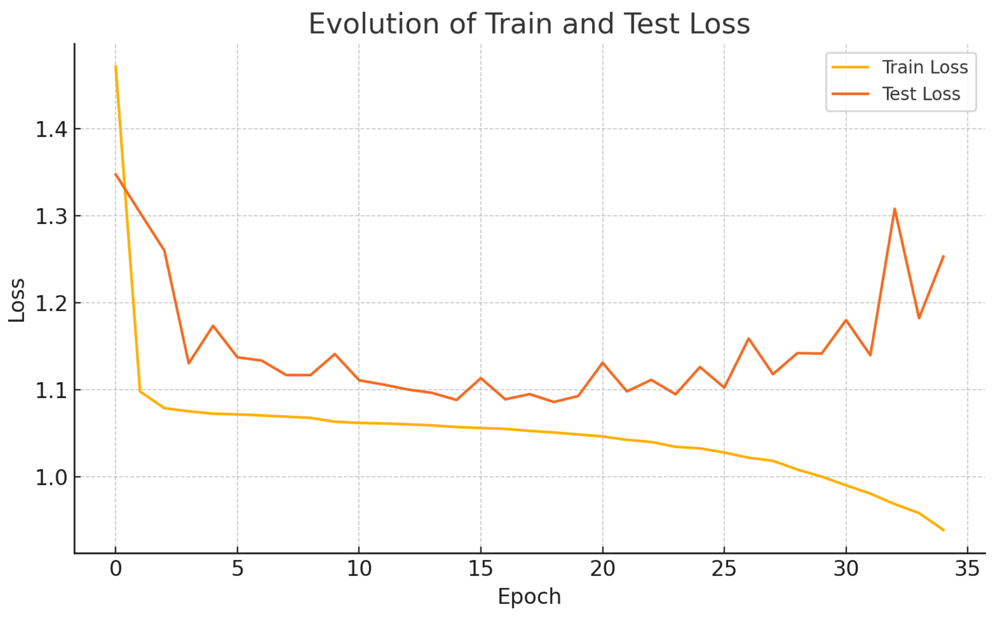

Figure 2: Training Loss vs Test Loss

Generally, while the Attention-Based CNN model delivers stronger in-sample metrics, the narrow lead in out-of-sample performance and signs of overfitting prevent it from being the definitive best approach in this scenario. The model's sophisticated design reduces noise and exploits data structure, but regularization or more advanced augmentation may be necessary to fully realize its potential on novel data.

5.7 Out-of-sample Prediction (Kaggle Accuracy)

The out-of-sample accuracy of all models are shown below. Encouragingly, all six of our models outperform the baseline accuracy of 0.4656, signaling that they are indeed learning valuable patterns from the historical football data. Among them, the Random Forest achieved the highest out-of-sample accuracy at 0.4837, followed closely by XGBoost (0.4835) and LightGBM (0.4805).

These ensemble models not only show strong predictive performance but also demonstrate that they are better at generalizing than simpler models like Logistic Regression. Notably, the Attention-Based CNN, a deep learning approach, achieved an accuracy of 0.4814, slightly trailing the best-performing tree-based models. Although the performance gains over the benchmark are modest—generally in the 1.5 to 1.8 percentage point range—they are consistent across all models, affirming the value of our modeling efforts.

Table 7: Kaggle out-of-sample accuracy for each model

Kaggle Accuracy	**Logistic Regression**	**SVM**	**Random Forest**	**Light GBM**	**XGBoost**	**Attention CNN**
	0.4712	0.4801	0.4837	0.4805	0.4835	0.4814

The gap between in-sample and out-of-sample performance, particularly for models like LightGBM and XGBoost, could indicate overfitting. These models show high in-sample accuracy but only modest improvement or parity with simpler models in the Kaggle (out-of-sample) submission.

6 Challenges and Solutions

6.1 Sparsity and Missing Data

The data we've received is quite sparse, with approximately 20% missing values and a significant disparity in the distribution and quantity of players for each match. Therefore, we primarily need to address two issues: 1. How to control data dimensionality, and 2. How to impute missing values.

Regarding dimensionality reduction, we carefully compared the prediction results after using PCA and Kernel PCA. Unfortunately, due to the inherent high noise and randomness in soccer matches, performance decreased both within and outside of the training samples, by roughly 2-5%. We also considered two other dimensionality reduction options: 1. Using an AutoEncoder for dimensionality reduction, and 2. Fitting an AutoEncoder to data from home teams winning matches and then using residual data as input for further dimensionality reduction. Regrettably, we haven't had sufficient time to complete this more demanding task by the time of this report. However, we anticipate that despite the current simple dimensionality reduction not performing well, given the high noise characteristic of this data, we can propose a reasonable hypothesis: if an appropriate method is selected for non-linear dimensionality reduction and the dimension is controlled reasonably, we may still be able to achieve better results.

For imputing missing values, we generally tested two approaches: 1. Imputing directly with a single value, and 2. Imputing with a single value while creating a new binary variable indicating whether the data has been imputed. We tested both methods. While theoretically, the second method should reduce the impact of imputation on the quality of valid data, our practical testing revealed that due to the excessive dimensionality of the sample and the scattering of missing values across every dimension (which may have been intentional by the organizer, as some data have a Mean but no Standard Deviation), using binary variables to mark imputed values further exacerbated the adverse effects of the curse of dimensionality. The overall benefit was less than the loss; therefore, we ultimately chose to impute all missing data with 0. Imputing with 0 is acceptable because 0 represents the minimum value in our data scale ([0,10] for teams, [0,100] for players), and imputing with 0 does not significantly confuse valid data values.

6.2 Neural Network Challenges

Theoretically, neural networks possess a very strong fitting capability and can be used to fit extremely large and complex datasets, including predicting stock market indices. There are also numerous examples in the literature we studied of using neural networks to predict match results. However, selecting a well-performing neural network requires extensive research and experimentation. Although we ultimately didn't develop a neural network that significantly outperforms ensemble models, we were still able to train a neural network architecture that performs on par with integrated models despite various challenges.

Architectures we researched are: Pure Dense layers; GRU or LSTM models that treat each player–team pair as one sequence; a lightweight CNN that encodes the data as images; a hybrid denoising model that combines a CNN with an LSTM; a hybrid model that adds an attention mechanism to an LSTM; and, finally, a hybrid model that fuses a CNN with a Row Attention mechanism was proposed because it demonstrated good ability to control dimension effects and suppress natural noises in the input data.

In this task, the neural network models primarily encountered three difficulties: 1. overfitting; 2. noise control; and 3. distorted prediction distributions resulting from imbalanced training samples. Among these, overfitting was relatively easier to resolve. Although even the simplest two-layer MLP exhibited severe overfitting issues on adversarially enhanced datasets, our final Attention-Based CNN model architecture incorporated multiple dropout layers, applied classification weights during training, and employed early stopping techniques, significantly alleviating the overfitting problem. After intensive research and testing, we identified the checkpoint at 20 epochs as the optimal fitting point for the model. At this stage, the model showed mild signs of overfitting, with training set accuracy merely reaching 0.55, introducing our second difficulty: noise control.

Soccer matches inherently possess significant randomness, exemplified by Japan's victory over Brazil in the 2022 World Cup. The strengths of teams and players alone are insufficient to guarantee stable victories; additionally, home advantages were invalidated in our example since matches occurred in Doha, Qatar. This randomness adversely impacted our neural network training, predominantly manifesting as noise. Noise led to highly unstable network parameters and gradients, exacerbating overfitting and occasionally causing severe underfitting, particularly when mini-batch sizes were small, resulting in training accuracy frequently below 0.50. After extensive study, we proposed two interim solutions: optimizing network architectures and employing larger mini-batches along with a higher initial effective learning rate. Regarding network optimization, we discovered that more complex architectures struggled against severe noise in the early stages of training, causing significant underfitting. Consequently, we progressively simplified our model architecture from an initial ResNet-50-like structure down to a simpler configuration featuring only three convolutional layers and one attention layer, ultimately reducing total parameters by more than 80%. For mini-batch training, we leveraged Kaggle's T4 x2 GPU system, offering 32 GB of VRAM, enabling training with a batch size of 384, which stabilized early gradient computations and substantially mitigated underfitting caused by noise.

The third difficulty, distorted prediction distributions due to imbalanced training samples, was characterized by models producing predictions significantly diverging from the original training and test distributions during parameter updates. For instance, our dataset consisted of approximately 50% home-win samples and around 35% draws. Ideally, after proper learning and fitting, a neural network should yield predictions closely resembling the original distribution (50%, 35%, 15%); however, even the simplest Dense model exhibited instability in matching this baseline distribution, sometimes predicting over 80% of samples as home wins, and at other times less than 30%, despite relatively stable loss function values. Unfortunately, we did not fully resolve this issue. Adjustments such as enabling and disabling class weighting showed minimal improvement; experiments with different network architectures, including removing attention or convolutional layers, also failed to yield stability. We hypothesize that the fundamental reason lies in the nature of neural networks, which seek globally optimal solutions using a large model structure, unlike ensemble methods that aggregate locally optimal solutions. The instability arises primarily due to noise—making global optima harder to identify compared to local optima—thus introducing numerous hills and saddle points within the loss function's hyperplane. During gradient descent, models frequently become trapped in these saddle points, descending toward locally optimal gradients rather than global optima. Although AdamW optimizers occasionally escape these local optima, repeated entrapment and escape cycles contribute to the unstable training distributions, a phenomenon supported by visualizations of the loss hyperplane. Unfortunately, we currently lack better solutions to this issue. However, addressing this problem and ensuring gradients consistently descend toward global optima could further enhance neural network performance.

6.3 Bayesian Optimization

To find the optimal hyperparameters in tuning the models, we planned to use grid Search and Bayesian Optimization to conduct a restricted grid search to tune the model for performance and generalization. Here, the central issue we encountered was the tradeoff between computational cost and model performance when applying Bayesian optimization. Initially, we extensively leveraged grid Search and Bayesian optimization to systematically identify optimal hyperparameters for relatively simpler models, such as logistic regression. During these trials, the grid Search and Bayesian approach was highly effective, efficiently exploring hyperparameter spaces to achieve reliable, reproducible improvements in validation accuracy.

However, as we transitioned toward optimizing more complex models—particularly random forests and various neural network architectures—we quickly encountered practical constraints. These complex models involve substantially larger hyperparameter spaces, including parameters governing model depth, breadth, learning rates, dropout rates, and regularization terms. Attempting exhaustive Bayesian optimization across all these parameters proved to be computationally prohibitive, especially given limited time and resources. Consequently, we were compelled to adopt a "partial search" strategy, selectively optimizing only the most impactful parameters identified through preliminary experiments and theoretical insights. This approach represented a pragmatic compromise, balancing computational feasibility against the thoroughness of hyperparameter exploration.

Within this specific task, we noticed another significant limitation: achieving the best performance on our validation set did not consistently translate into superior out-of-sample performance, which sometimes stands for generalization. During repeated testing cycles, we observed that the validation dataset provided by QRT occasionally differed moderately from the broader distribution present in our overall training set. Specifically, models which attained exceptional results during Bayesian hyperparameter tuning on the QRT-provided validation set sometimes failed to generalize as expected when evaluated against completely unseen data. In contrast, certain models that exhibited slightly inferior performance on the validation set occasionally demonstrated greater robustness and stronger generalization capabilities on entirely new datasets. This shows the importance of the i.i.d. assumption always made in machine learning field.

7 Conclusion and Future Work

In this study, we explored the task of football match outcome prediction using a variety of machine learning and deep learning models, trained on a high-dimensional dataset of aggregated team and player statistics. Our pipeline involved careful preprocessing, baseline model evaluation, and extensive experimentation with neural architectures. Among all approaches, ensemble methods such as Random Forest, Light GBM, and XGBoost delivered the most robust out-of-sample performance, with XGBoost achieving up to 0.4835 accuracy on Kaggle test data. While our Attention-Based CNN model demonstrated promising in-sample performance, it struggled with overfitting and did not consistently outperform traditional methods.

These results suggest that, despite the theoretical power of neural networks, ensemble tree-based models remain more effective for tabular, noisy, and structured sports datasets. However, our deep learning pipeline laid the groundwork for further architectural exploration, especially in handling player-role alignment, high variance, and data sparsity.

Our study has several limitations, primarily stemming from constraints in time and computational resources. For traditional machine learning models, our work did not incorporate advanced feature engineering or dimensionality reduction. Initial trials using basic transformations—such as differencing, interactions, and log scaling—did not improve results significantly. However, we believe more meaningful gains could be achieved through domain-informed feature design that captures tactical or contextual nuances of football matches. Leveraging expert knowledge to guide feature construction remains a promising direction for future research.

For deep neural models, future work can improve upon three fronts: architecture design, data processing, and training methodology.

First, we can continue improving the architecture design of the Attention-Based CNN model. Although it demonstrated reasonable performance, we observed instability in its training behavior and signs of overfitting. This may stem from its attention mechanism focusing on irrelevant details or local noise. Future designs could incorporate sequence modeling components, such as LSTM layers before the Row Attention module, to better capture temporal continuity and player-team interactions. Reducing model sensitivity to local fluctuations and improving generalization remains a key architectural challenge.

Our current pipeline processes match data into 4-dimensional tensors but apply minimal noise reduction. This exposes the model to the curse of dimensionality and amplifies signal sparsity. Future pipelines might benefit from using Variational Autoencoders (VAEs) for denoising, or architectures that extract residual prediction signals on known outcomes like `HOME_WIN` cases. Generating adversarial samples or leveraging representation learning could also improve the network's robustness to noisy or imbalanced data distributions.

Rapid overfitting during training highlights potential shortcomings in optimization strategy. Although we experimented with learning rate schedules and dropout regularization, further gains may require revisiting the loss function itself. Adopting more specialized losses—such as Focal Loss—could help down-weight easy or noisy samples and guide gradient updates toward more generalizable patterns.

In conclusion, while our current models capture meaningful signals from structured football data, substantial opportunities remain for advancing predictive performance. Combining domain-informed features, more resilient neural architectures, and robust data preprocessing techniques offers a strong foundation for future improvements in match outcome forecasting.

References

R. Baboota and H. Kaur. Predictive analysis and modelling football results using machine learning approach for english premier league. *International Journal of Forecasting*, 35(2):741–755, 2019.

Y. Jia, A. Anaissi, and B. Suleiman. Resnls: An improved model for stock price forecasting, 2025.

F. Rodrigues and A. Pinto. Predicting football match outcomes with machine learning: The case of the english premier league. *Procedia Computer Science*, 203:516–523, 2022.

P. Sreenivasgoud, K. Sridhar, M. Sirajuddin, T. Venkatesh, and R. Sagar. Analyzing and predicting football match results using deep learning. In *AIP Conference Proceedings*, volume 2971, page 020042, 2024.

S. Yu, Z. Zhang, S. Wang, X. Huang, and Q. Lei. A performance-based hybrid deep learning model for predicting tbm advance rate using attention-resnet-lstm. *Journal of Rock Mechanics and Geotechnical Engineering*, 16(1):65–80, 2024.

Q. Zhang, X. Zhang, H. Hu, C. Li, Y. Lin, and R. Ma. Sports match prediction model for training and exercise using attention-based lstm network. *Digital Communications and Networks*, 8(4): 508–515, 2022.