Final Project

Is it possible to accurately predict the outcome of an English Premier League (EPL) match — win, draw, or loss — based on team-level performance metrics?

Introduction

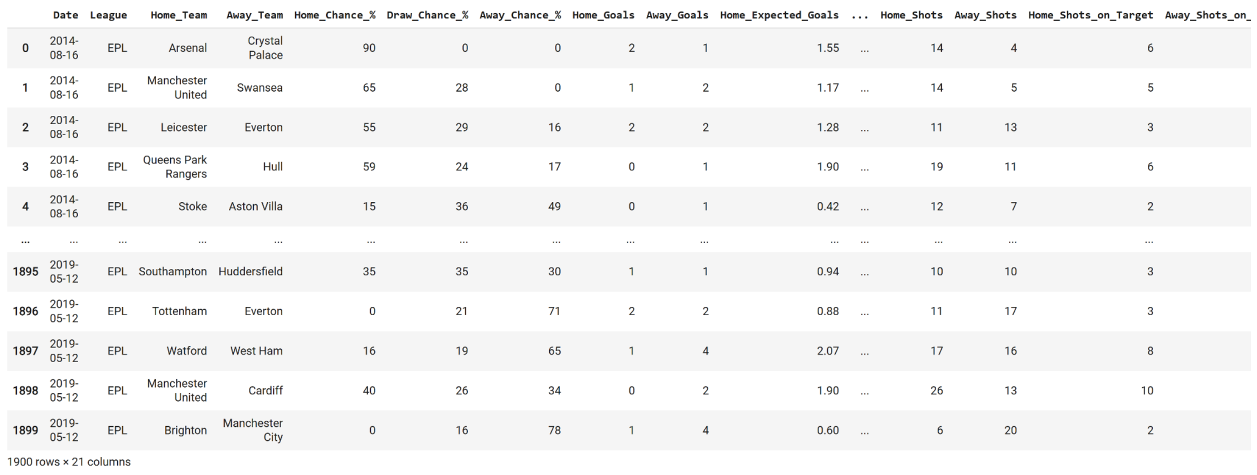
Predicting the outcomes of football matches—specifically wins, draws, or losses—has long been a central challenge in sports analytics. This project investigates whether match outcomes in the English Premier League (EPL) can be accurately predicted using team-level performance metrics derived from historical data. By framing the task as a multiclass classification problem, my research explores four machine learning models of increasing complexity: logistic regression, decision tree, k-nearest neighbors (KNN), and a neural network. The models are trained and evaluated on engineered features that capture differences in offensive performance, shot quality, and tactical pressure between competing teams. Among the models tested, the neural network consistently delivered the most balanced performance across outcome categories, achieving the highest macro F1-score and demonstrating improved prediction of underrepresented results such as losses. These findings suggest that incorporating non-linear modeling and modern deep learning techniques enhances the capacity to capture nuanced match dynamics and provides more equitable outcome predictions—offering actionable insights for analysts and teams alike.

Dataset Description

The dataset used in this project was sourced from OpenML, a widely used platform for sharing open data and machine learning benchmarks. It contains match-level statistics for European football leagues, including detailed metrics on team performance, expected goals (xG), and game outcomes. For the purpose of this study, I focused specifically on the English Premier League

(EPL), selecting data from the 2014/15 through 2018/19 seasons. This five-season window was chosen to ensure consistency across teams, as each club plays the same number of matches in a season, making the data more structurally reliable for predictive modeling.

Figure 1 below shows the original dataset after initial filtering for EPL matches only. The raw data includes a variety of match-specific features such as team names, goals scored, expected goals, number of shots (both total and on target), pressing metrics like PPDA, deep attacking entries, and projected win probabilities (e.g., *Home_Chance_%*). Each row corresponds to a single match, with separate statistics for the home and away teams.



	Date	League	Home_Team	Away_Team	Home_Chance_%	Draw_Chance_%	Away_Chance_%	Home_Goals	Away_Goals	Home_Expected_Goals	...	Home_Shots	Away_Shots	Home_Shots_on_Target	Away_Shots_on_
0	2014-08-16	EPL	Arsenal	Crystal Palace	90	0	0	2	1	1.55	...	14	4	6	
1	2014-08-16	EPL	Manchester United	Swansea	65	28	0	1	2	1.17	...	14	5	5	
2	2014-08-16	EPL	Leicester	Everton	55	29	16	2	2	1.28	...	11	13	3	
3	2014-08-16	EPL	Queens Park Rangers	Hull	59	24	17	0	1	1.90	...	19	11	6	
4	2014-08-16	EPL	Stoke	Aston Villa	15	36	49	0	1	0.42	...	12	7	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1895	2019-05-12	EPL	Southampton	Huddersfield	35	35	30	1	1	0.94	...	10	10	3	
1896	2019-05-12	EPL	Tottenham	Everton	0	21	71	2	2	0.88	...	11	17	3	
1897	2019-05-12	EPL	Watford	West Ham	16	19	65	1	4	2.07	...	17	16	8	
1898	2019-05-12	EPL	Manchester United	Cardiff	40	26	34	0	2	1.90	...	26	13	10	
1899	2019-05-12	EPL	Brighton	Manchester City	0	16	78	1	4	0.60	...	6	20	2	

1900 rows × 21 columns

Figure 1

To prepare this data for modeling, several preprocessing steps were applied. First, I filtered the dataset by date to include only matches played before May 12, 2019—marking the end of the 2018/19 EPL season. I also removed unnecessary columns (e.g., *Unnamed:_0*) and cleaned the team name strings to eliminate inconsistent formatting. Next, I created a new target variable called "*Match_Result*", which classifies each game outcome as a 'Win', 'Draw', or 'Loss' from the home team's perspective. This variable serves as the classification label for the predictive models.

In order to strengthen the model's ability to capture relative differences between competing teams, I engineered a set of new features that compare home and away performance across several dimensions. These include differences in expected goals (*xG_Diff*), total shots and shots on target, pressing intensity (*PPDA_Diff*), deep attacking entries, and chance projections such as "*ExpectedPoints_Diff*" and "*Chance_%_Diff*". This approach allows the model to evaluate each match not just based on isolated team metrics, but on how one team performed relative to its opponent.

Evaluation Metrics

In this project, the task of predicting football match outcomes (Win, Draw, Loss) constitutes a multiclass classification problem with a moderately imbalanced class distribution. Although accuracy is a widely used evaluation metric, it can be misleading in this context. A model may achieve high accuracy by consistently predicting the most frequent class—typically wins—yet perform poorly on less frequent but equally critical outcomes such as draws or losses. This imbalance poses a challenge for fair performance assessment.

To ensure a more meaningful and balanced evaluation, I adopt the macro-averaged F1-score as the primary metric across all models. The F1-score integrates both precision (the proportion of predicted positives that are correct) and recall (the proportion of actual positives that are correctly predicted), offering a more nuanced view of classification performance. The macro-average further enhances fairness by treating each class equally, regardless of its frequency. This is particularly relevant in sports analytics, where identifying less common but impactful results—such as unexpected losses or tightly contested draws—can yield valuable insights for teams, coaches, and analysts.

By prioritizing macro F1-score, I establish a consistent and equitable standard for comparing model performance across my models. While supporting metrics such as accuracy, confusion matrices, and per-class precision and recall are also reported to aid interpretation, the macro F1-score remains the most appropriate measure for this imbalanced, multiclass setting. Its emphasis on class-level balance ensures that model effectiveness is not overstated by majority-class dominance and better reflects real-world predictive utility.

Model 1: Logistic Regression

Overview and Implementation

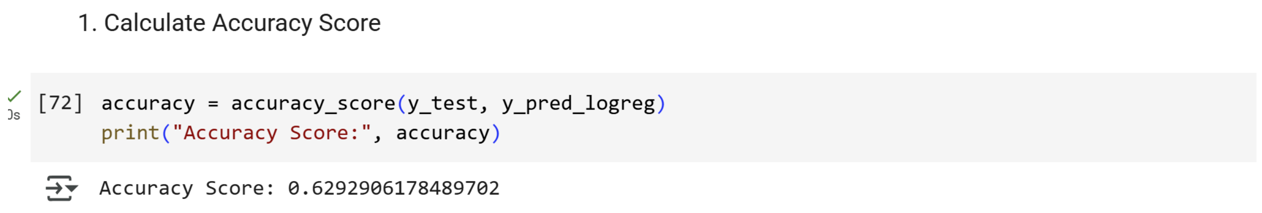
To initiate the modeling process, I implemented a multinomial logistic regression model to predict match outcomes—categorized as win, draw, or loss. Logistic regression was selected as a baseline classifier due to its interpretability and suitability for multiclass classification problems. It provides a clear starting point for understanding how well structured, linear relationships between match features and outcomes can perform.

The model was trained using seven engineered features, each representing a performance difference between the home and away teams. These features were selected to capture disparities in shot quality, offensive pressure, chance creation, and overall match dominance. Before training, the feature set was standardized using z-score normalization to ensure consistent scaling, which is critical for optimizing logistic regression performance.

The dataset was split into training and testing subsets using an 80-20 split. The model was implemented using the LogisticRegression class from Scikit-learn, with the *lbfgs solver* and *multi_class='multinomial'* setting. A fixed *random_state* was used to ensure reproducibility.

Results and Interpretation

To assess the performance of the logistic regression model in predicting match outcomes, I evaluated the model using multiple classification metrics, with particular emphasis on the macro-averaged F1-score as the primary indicator of model effectiveness. While the model achieved an overall accuracy of approximately 63% (Figure 2)—a solid result for a baseline classifier—accuracy alone does not capture class-specific disparities, especially given the class imbalance present in the dataset.



1. Calculate Accuracy Score

```
[72] accuracy = accuracy_score(y_test, y_pred_logreg)
     print("Accuracy Score:", accuracy)
```

```
Accuracy Score: 0.6292906178489702
```

Figure 2

The confusion matrix (Figure 3) and classification report (Figure 4) offer a deeper view into how well the model performs across individual outcome categories. The model was most effective at predicting wins, correctly classifying 171 out of 200 win cases. This is reflected in the strong recall (0.85), precision (0.65), and F1-score (0.74) for the *Win* class. The *Draw* class also showed moderate performance, with a recall of 0.73 and F1-score of 0.68, indicating the model's ability to detect balanced outcomes reasonably well. In contrast, the model struggled significantly with the *Loss* class, predicting only 7 out of 105 cases correctly. The resulting recall of 0.07 and F1-score of 0.11 highlight the model's limited ability to recognize losing outcomes.

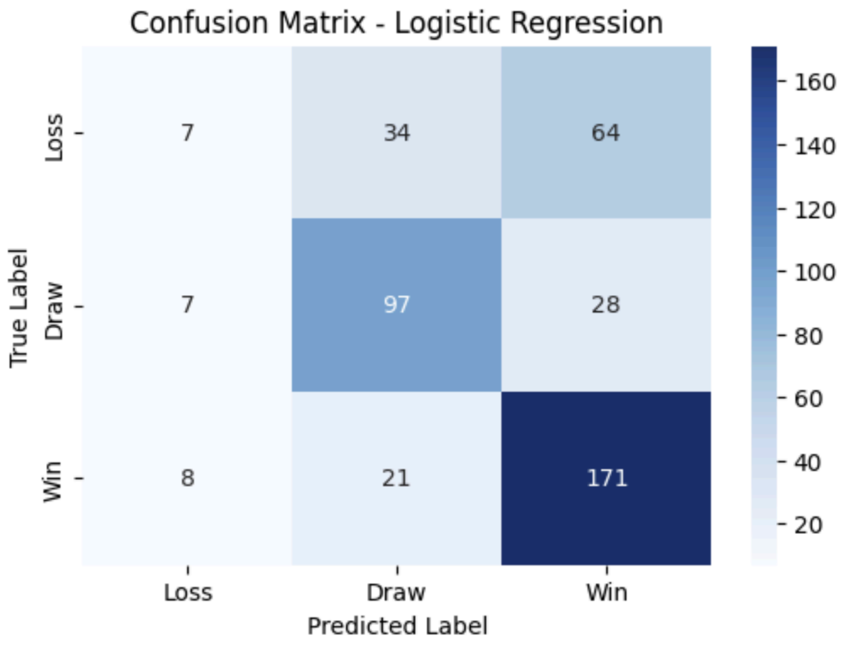
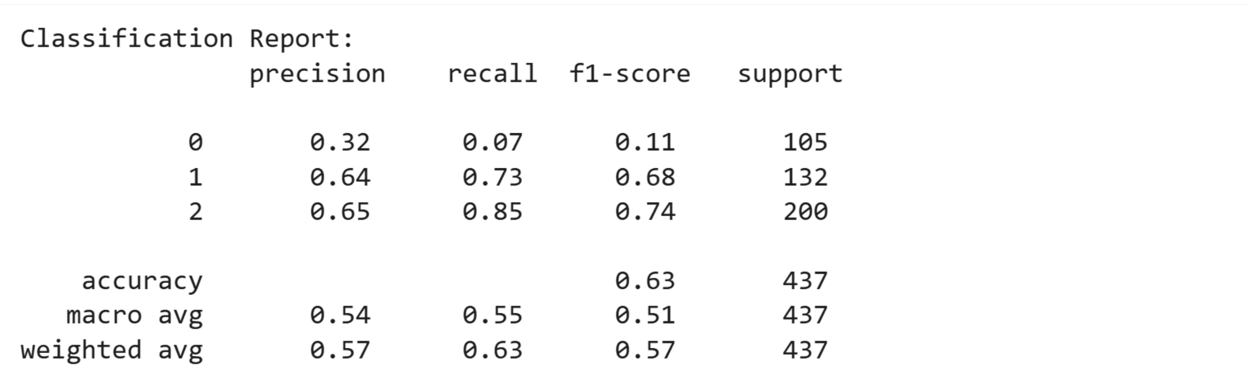


Figure 3



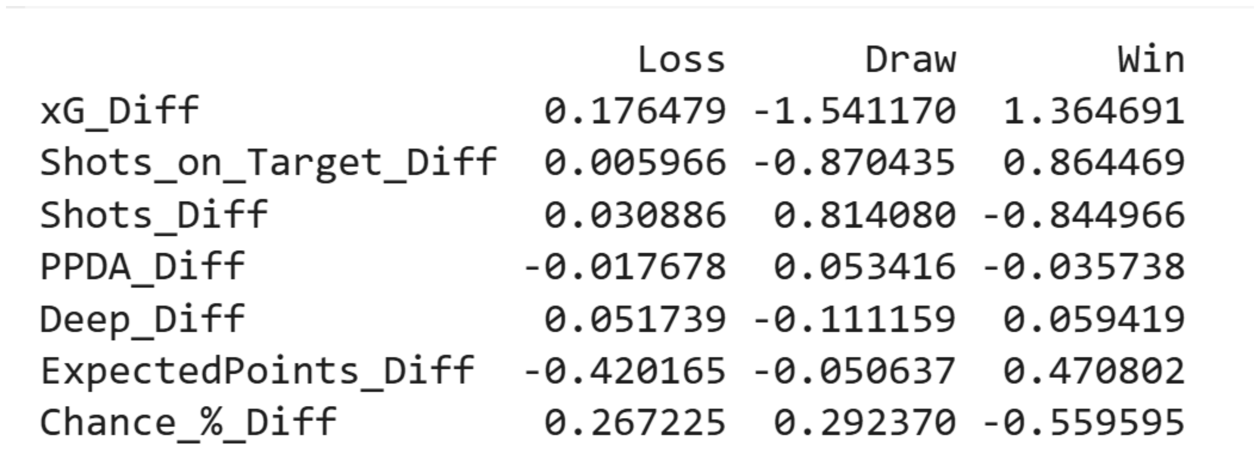
```
Classification Report:
              precision    recall  f1-score   support

           0       0.32      0.07      0.11       105
           1       0.64      0.73      0.68       132
           2       0.65      0.85      0.74       200

    accuracy                           0.63       437
   macro avg       0.54      0.55      0.51       437
weighted avg       0.57      0.63      0.57       437
```

Figure 4

One plausible explanation for the model's difficulty in accurately predicting losses is that the *Loss* class may be underrepresented in the training data. Additionally, match outcomes labeled as losses are often influenced by highly contextual and unpredictable real-world events, such as defensive errors, red cards, or tactical shifts late in the game—none of which are directly captured by the team-level statistical features used here. This may partially explain the model's limited sensitivity in identifying such outcomes.

These class-level disparities clearly illustrate why the macro F1-score is a more informative metric in this context. By averaging performance equally across all three classes, macro F1-score avoids the misleading optimism that often accompanies accuracy in imbalanced settings. The performance imbalance observed here strongly justifies the exploration of more flexible, nonlinear models that may better capture the interactions between features and improve the detection of underrepresented classes like losses.

To better understand how specific features influenced the logistic regression model's predictions, I analyzed the learned model coefficients (Figure 5). Each coefficient reflects the direction and strength of a feature's impact on the likelihood of a particular outcome. Notably, features like *xG_Diff* and *Shots_on_Target_Diff* had strong positive coefficients for the *Win* class and negative coefficients for *Draw*, suggesting that teams with superior expected goals and shot quality are more likely to secure a win. On the other hand, features like *ExpectedPoints_Diff* and *Chance_%_Diff* contributed more heavily toward *Draw* or *Loss* predictions. This coefficient analysis complements the performance metrics by providing interpretability—one of the key advantages of logistic regression—and reinforces the idea that offensive dominance plays a decisive role in match outcomes.



```
                         Loss      Draw       Win
xG_Diff              0.176479 -1.541170  1.364691
Shots_on_Target_Diff 0.005966 -0.870435  0.864469
Shots_Diff           0.030886  0.814080 -0.844966
PPDA_Diff           -0.017678  0.053416 -0.035738
Deep_Diff            0.051739 -0.111159  0.059419
ExpectedPoints_Diff -0.420165 -0.050637  0.470802
Chance_%_Diff        0.267225  0.292370 -0.559595
```

Figure 5

Model 2: Decision Tree

Overview and Implementation

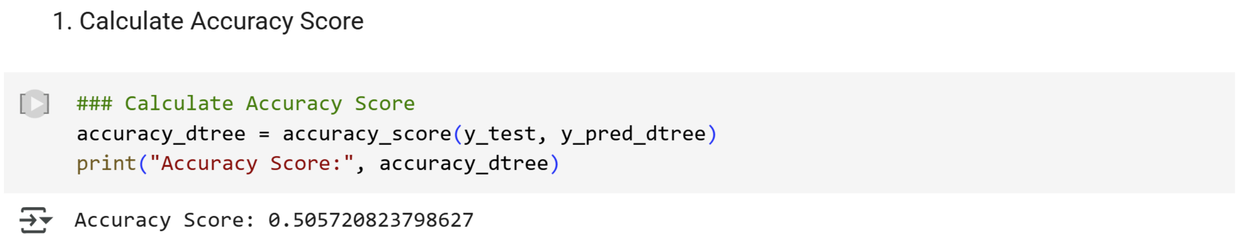
Following the logistic regression model, I implemented a Decision Tree classifier as the second predictive model. Decision trees are well-suited for classification tasks involving structured tabular data and are capable of modeling nonlinear relationships and interactions between features. Unlike logistic regression, decision trees do not assume a linear boundary and are able to handle complex splitting rules, which makes them a strong candidate for capturing team behavior patterns in football match data.

The same feature set used in the previous model was applied here. The data was preprocessed using feature scaling, even though decision trees do not inherently require scaling—this was done to maintain consistency across models.

To build the decision tree model, I used the DecisionTreeClassifier from *scikit-learn* with gini and a fixed random_state for reproducibility. The model was trained on the scaled training data (*X_train_scaled, y_train*) and used to make predictions on the test set (*X_test_scaled*).

Results and Interpretation

The Decision Tree model achieved an overall accuracy (Figure 6) of approximately 50.6%, which is lower than that of the Logistic Regression model. However, the model's macro F1-score (Figure 7) was 0.47, indicating moderate overall balance across the three outcome classes.

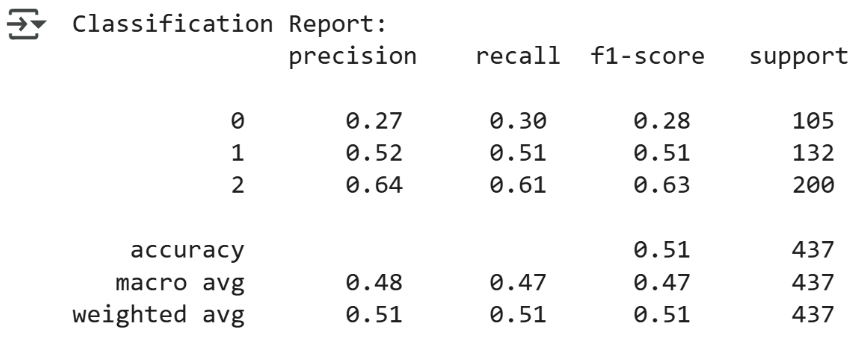


1. Calculate Accuracy Score

```
### Calculate Accuracy Score
accuracy_dtree = accuracy_score(y_test, y_pred_dtree)
print("Accuracy Score:", accuracy_dtree)
```

```
Accuracy Score: 0.505720823798627
```

Figure 6



```
Classification Report:
              precision    recall  f1-score   support

           0       0.27      0.30      0.28       105
           1       0.52      0.51      0.51       132
           2       0.64      0.61      0.63       200

    accuracy                           0.51       437
   macro avg       0.48      0.47      0.47       437
weighted avg       0.51      0.51      0.51       437
```

Figure 7

A closer look at the per-class metrics shows that the model struggled the most with predicting "Loss" outcomes, which had the lowest F1-score (0.28). This underperformance is likely due to both class imbalance—losses are less frequent in the dataset—and the model's difficulty in capturing complex match dynamics such as red cards, goalkeeping errors, or late-game tactical shifts, none of which are well-represented in the structured team-level statistics used here. In contrast, the model performed substantially better on "Win" predictions, achieving an F1-score of 0.63, and showed moderate effectiveness at identifying draws (F1-score of 0.51). These findings suggest that while the model can learn strong patterns for dominant outcomes like wins, it has limitations in generalizing to rarer or more context-specific outcomes.

The confusion matrix (Figure 8) further supports these observations: misclassifications of losses into other categories are frequent, and draws are often confused with either wins or losses. The bar plot (Figure 9) comparing class-level precision, recall, and F1-score visualizes this disparity clearly, with the "Loss" class having significantly shorter bars, reinforcing its relative underperformance. This class-level imbalance again highlights the importance of using macro F1-score over accuracy, as it treats each outcome equally and reveals where the model struggles.

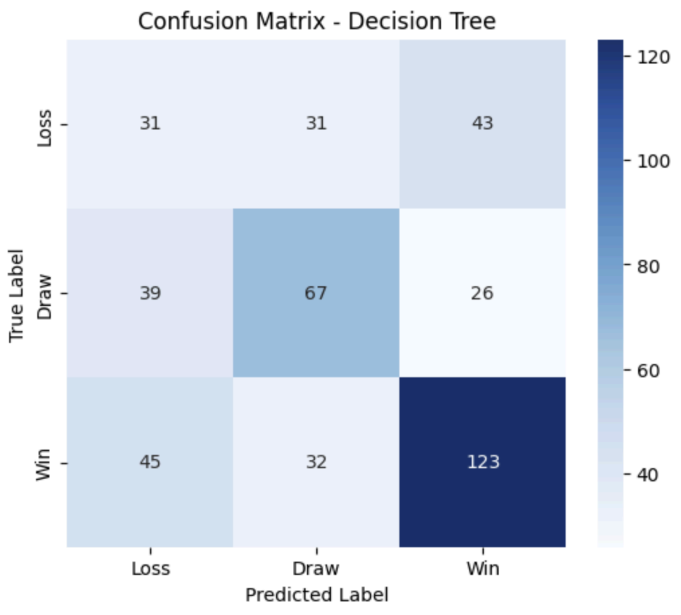


Figure 8

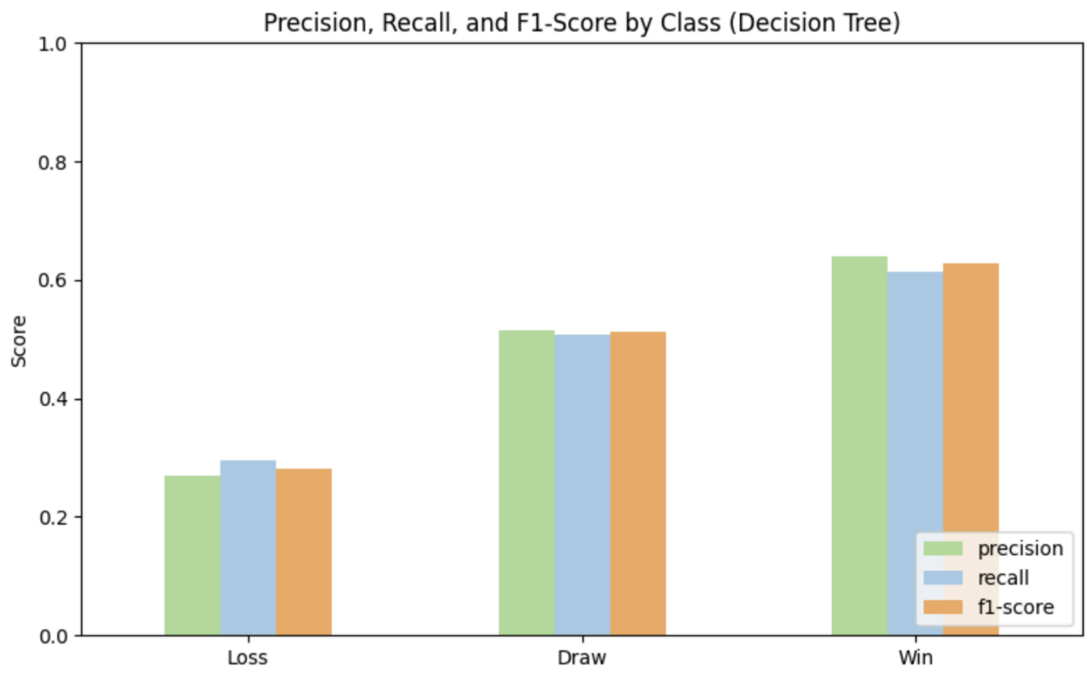


Figure 9

To further understand how the model makes its decisions, the truncated decision tree visualization (Figure 10) offers interpretability into which features are most influential. The root node splits on *ExpectedPoints_Diff*, indicating it is the most decisive feature in the model. Features such as *Chance_%_Diff, Shots_on_Target_Diff,* and *xG_Diff* appear frequently in upper-level splits, especially in paths that lead to confident "Win" classifications. This structural view supports the earlier metric-driven findings by showing that the tree tends to form strong rules for identifying wins, while loss-related branches are more diffuse and fragmented. The interpretability of the decision tree adds qualitative support to the conclusion that while this model is more transparent than logistic regression, it still exhibits limited predictive power for underrepresented match outcomes.

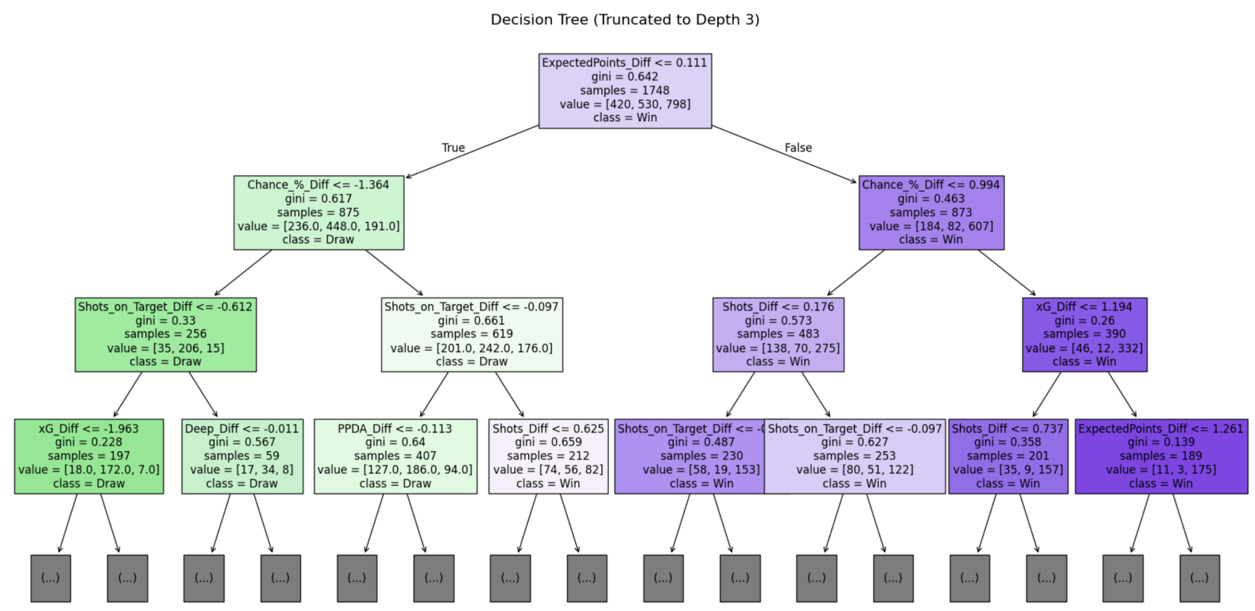


Figure 10

Overall, although the Decision Tree model provides clear interpretability and moderate success in predicting wins and draws, it underperforms compared to Logistic Regression based on macro F1-score. These findings align with the broader research goal of identifying effective models for predicting football match outcomes and highlight the need to explore more flexible algorithms—such as ensemble methods or neural networks—that can better capture non-linear relationships and improve class-level balance.

Model 3: k-Nearest Neighbors (KNN)

Overview and Implementation

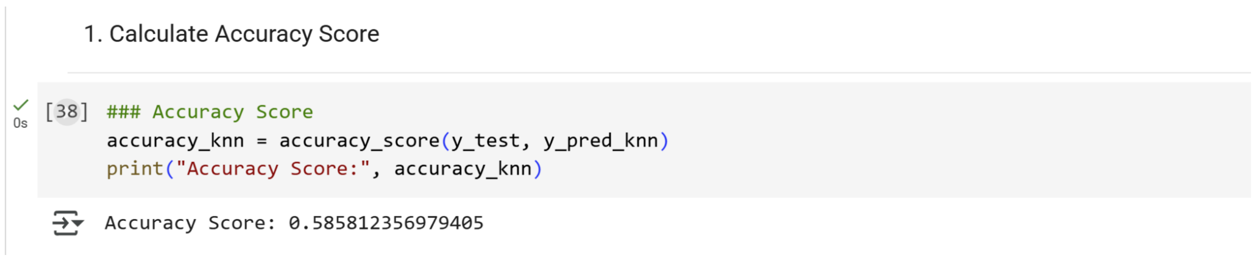
The third predictive model implemented is the K-Nearest Neighbors (KNN) classifier, a non-parametric algorithm that makes predictions based on the majority class among the *k* most similar training examples. Given its intuitive logic and flexibility, KNN is particularly useful for evaluating local similarity patterns in match-level statistics.

To optimize performance, I performed hyperparameter tuning using GridSearchCV to identify the best value for the k parameter. The grid search tested multiple values for n_neighbors and selected the configuration that achieved the best cross-validated performance. This added depth to the KNN implementation by ensuring that the model is not arbitrarily fixed to a suboptimal *k* value. Additionally, I used 5-fold cross-validation to increase robustness and generalizability of the selected parameter.

After identifying the optimal value of *k*, the model was trained on the scaled training set and tested on the scaled test set. Feature scaling was necessary for KNN since distance-based models are sensitive to feature magnitude.

Results and Interpretation

The KNN model with hyperparameter tuning achieved an overall accuracy (Figure 11) of approximately 58.6%, comparable to the logistic regression and slightly higher than the decision tree model. However, as discussed earlier, accuracy alone is insufficient in multiclass classification problems with class imbalance. Therefore, we evaluate performance primarily using the macro-averaged F1-score, which provides a balanced view of model performance across all classes.


1. Calculate Accuracy Score

```
[38] ### Accuracy Score
     accuracy_knn = accuracy_score(y_test, y_pred_knn)
     print("Accuracy Score:", accuracy_knn)
```

```
Accuracy Score: 0.585812356979405
```

Figure 11

The macro F1-score (Figure 12) for the KNN model was 0.51, slightly better than the decision tree (0.47) and on par with logistic regression (0.51). The class-specific F1-scores show a pattern consistent with previous models: the model performed best on the "Win" category (F1 = 0.71), moderately well on "Draw" (F1 = 0.61), and struggled the most with "Loss" (F1 = 0.22). This indicates that while KNN is strong at detecting wins, it faces similar challenges in identifying losses—a likely result of data imbalance and the more complex or variable conditions under which teams lose, which are harder to capture with team-level features.

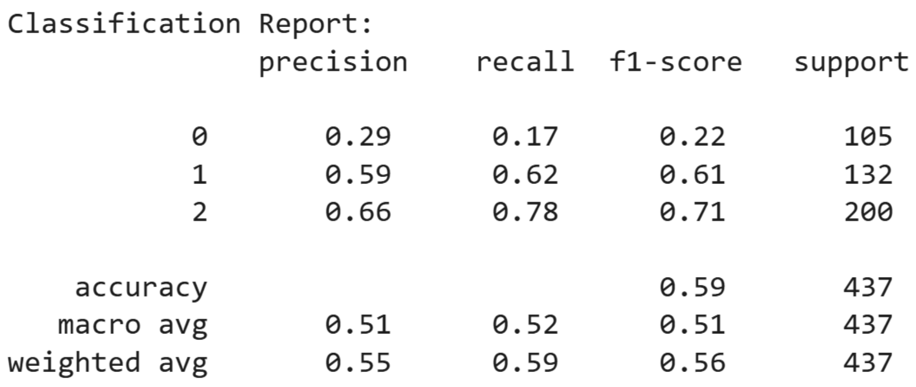

```
Classification Report:
              precision    recall  f1-score   support

           0       0.29      0.17      0.22       105
           1       0.59      0.62      0.61       132
           2       0.66      0.78      0.71       200

    accuracy                           0.59       437
   macro avg       0.51      0.52      0.51       437
weighted avg       0.55      0.59      0.56       437
```

Figure 12

The confusion matrix (Figure 13) reinforces these observations: the model correctly predicted 156 out of 200 wins, but only 18 of 105 losses, frequently misclassifying them as wins or draws. The draw category had more balanced predictions, with 82 correct out of 132.

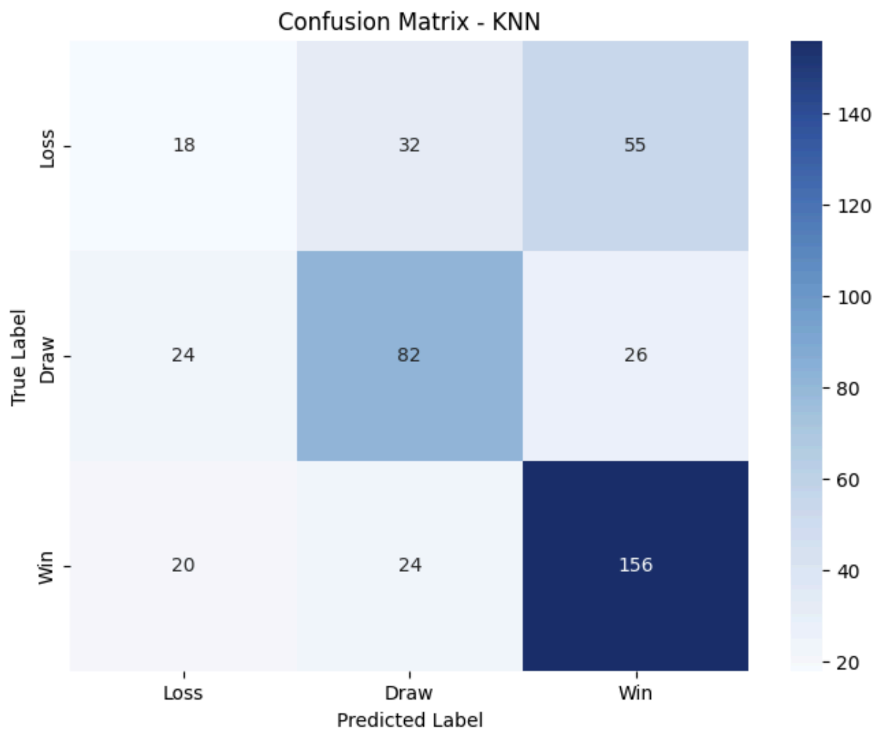


Figure 13

Overall, while KNN provided solid performance in recognizing wins and moderate improvement in draws, it did not significantly address the challenge of predicting losses. This model is competitive, and its performance highlights both the strengths and persistent limitations of team-level metrics for multiclass match outcome prediction.

Model 4: Neural Network

To further enhance the modeling framework and explore whether more complex, non-linear relationships exist within the feature set, I decide to include a neural network classifier as a fourth model. While earlier models like logistic regression and decision trees provided interpretability and baseline predictive power, their structural simplicity may limit their ability to capture deeper interactions among features. A neural network offers greater flexibility in learning from patterns that are not linearly separable or easily defined by hard thresholds, potentially improving predictive performance—especially for underrepresented classes like "Loss" Including this model provides a more complete comparative landscape and allows for the evaluation of whether added model complexity translates into tangible performance gains.

Overview and Implementation

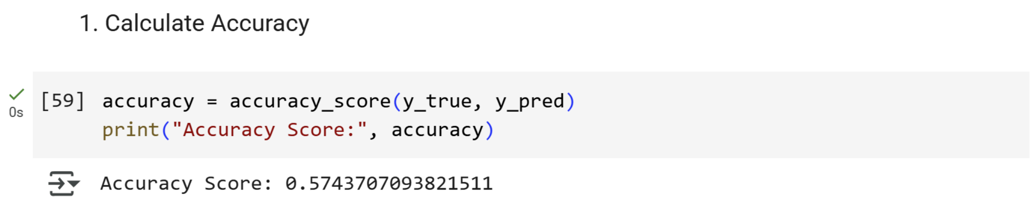
I implemented a neural network classifier using TensorFlow and Keras. The input features were standardized prior to training to facilitate more stable optimization. The architecture consists of several dense layers with ReLU activations, interleaved with dropout layers to reduce overfitting and batch normalization layers to stabilize learning. The final output layer uses a softmax activation function to assign probabilities across the three outcome classes.

To handle class imbalance and ensure fair training across all outcomes, I applied class weighting using *compute_class_weight* from scikit-learn. I also implemented *early stopping* based on validation loss to prevent overfitting and retain the best-performing model. The model was trained with the Adam optimizer and categorical cross-entropy loss, using a validation split of 20% and a batch size of 32 for up to 100 epochs.

This neural network model incorporates several modern deep learning techniques—regularization, normalization, and adaptive learning—to build a more expressive and robust classifier.

Results and Interpretation

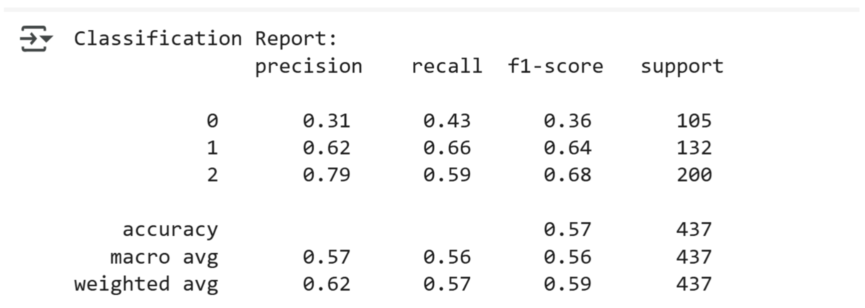
The neural network model achieved an accuracy (Figure 14) of approximately 57.4%, which, while slightly lower than that of the original logistic regression and KNN models, must be interpreted with caution given the multiclass nature and imbalance in the dataset. More importantly, the macro-averaged F1-score (Figure 15), which remains the primary evaluation metric in this project, rose to 0.56, matching or slightly exceeding the performance of other baseline models and confirming the neural network's capability to generalize across outcome categories.



1. Calculate Accuracy

```
[59] accuracy = accuracy_score(y_true, y_pred)
     print("Accuracy Score:", accuracy)

Accuracy Score: 0.5743707093821511
```

Figure 14



```
Classification Report:
              precision    recall  f1-score   support

           0       0.31      0.43      0.36       105
           1       0.62      0.66      0.64       132
           2       0.79      0.59      0.68       200

    accuracy                           0.57       437
   macro avg       0.57      0.56      0.56       437
weighted avg       0.62      0.57      0.59       437
```

Figure 15

The confusion matrix (Figure 16) shows that the model correctly predicted 119 win outcomes, 87 draws, and 45 losses. These figures indicate a clear improvement over earlier models in capturing less frequent results, particularly the "Loss" class, which was previously the weakest point in prediction accuracy. Although there is still some misclassification between loss and draw outcomes, the model displays a significantly more balanced prediction profile than the decision tree and KNN classifiers.

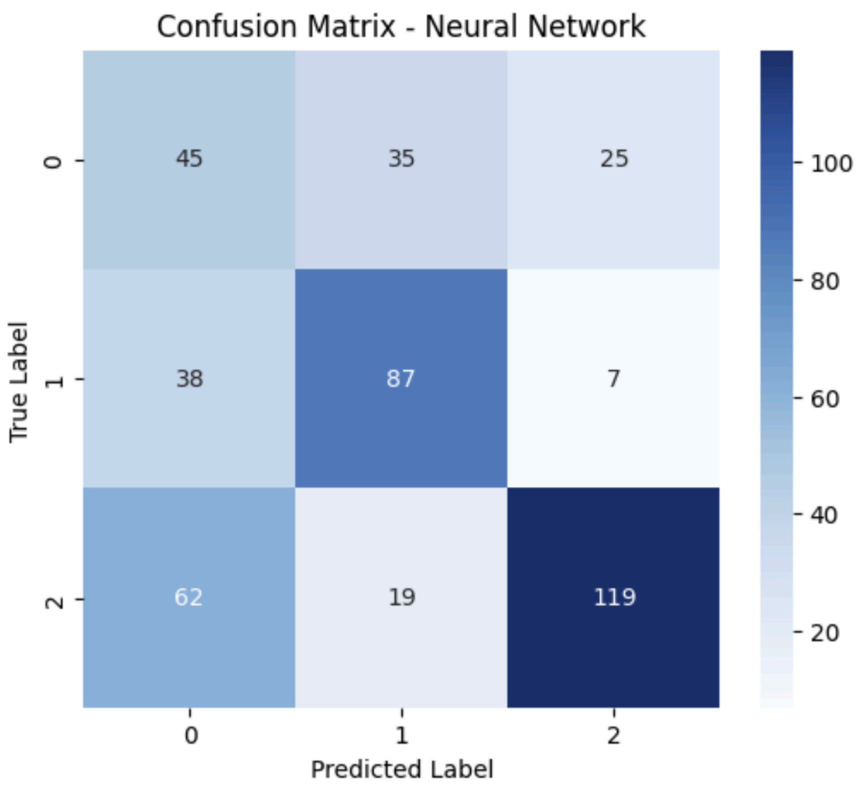


Figure 16

This conclusion is further supported by the classification report. The F1-score for "Loss" increased to 0.36, the highest among all models for this underrepresented class. The model also preserved relatively strong predictive power for "Draw" (F1 = 0.64) and "Win" (F1 = 0.68) outcomes. Precision and recall values across all classes remained consistent, indicating that the model is not overly biased toward any single outcome. Additionally, the grouped bar chart (Figure 17) highlights that the three-class performance is notably more even than in prior models, especially when comparing to the sharp imbalance of KNN and logistic regression.

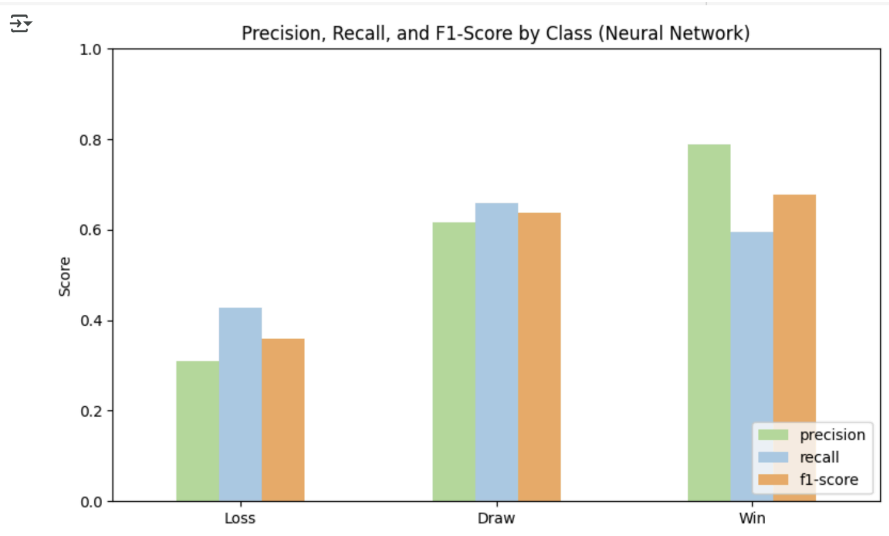


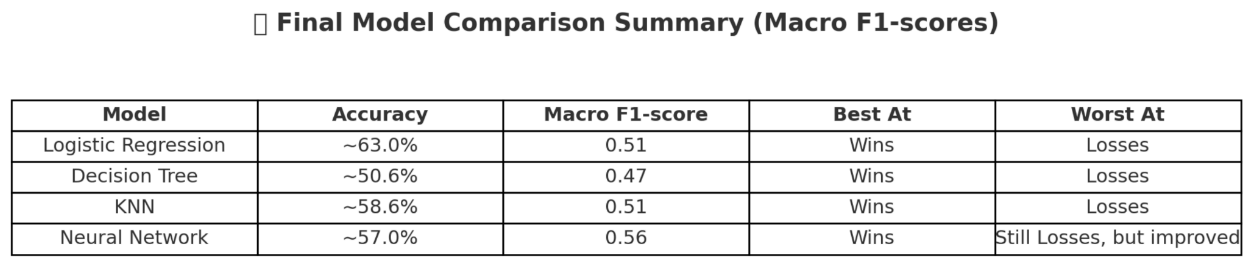
Figure 17

Overall, these results affirm that incorporating class weights, deeper architecture, dropout regularization, and batch normalization helped improve generalization and robustness, especially in minority classes. Although the neural network does not reach the highest accuracy, its macro F1-score, balanced class performance, and interpretability improvements make it a strong candidate for predicting Premier League match outcomes. Compared with the baseline models, it demonstrates the most equitable trade-off across all three outcome categories, offering valuable insights for predictive sports analytics.

Model Summary and Conclusion

To evaluate which model best predicts football match outcomes (Win, Draw, Loss) based on team-level statistics, four classification models were developed and compared: Logistic Regression, Decision Tree, K-Nearest Neighbors (KNN), and a Neural Network. Each model was assessed primarily using macro-averaged F1-score, a metric that ensures equal consideration for each outcome class, making it especially appropriate for our moderately imbalanced dataset. Complementary metrics such as accuracy and class-wise F1-scores were also examined.

The figure below (Figure 18) provides a consolidated view of each model's overall performance, including its accuracy, macro F1-score, and which outcome classes it performed best and worst at:



□ Final Model Comparison Summary (Macro F1-scores)

Model	Accuracy	Macro F1-score	Best At	Worst At
Logistic Regression	~63.0%	0.51	Wins	Losses
Decision Tree	~50.6%	0.47	Wins	Losses
KNN	~58.6%	0.51	Wins	Losses
Neural Network	~57.0%	0.56	Wins	Still Losses, but improved

As the summary table shows, the Neural Network model achieved the highest macro F1-score (0.56), outperforming other models in overall balanced predictive performance. While all models were most successful at identifying wins, the Neural Network showed relative improvement in identifying losses, which had been the weakest class across earlier models. This result suggests that the non-linear capabilities and deeper structure of neural networks allow them to better capture complex patterns in the input features.

From a strategic standpoint, this reinforces the value of using more flexible, non-linear models when predicting nuanced outcomes in competitive sports, especially when interest extends beyond majority class outcomes (i.e., wins) to rarer but equally important results like draws and losses.

Thus, based on both fairness across classes and predictive quality, the **Neural Network model** is selected as the best-performing classifier for this football match prediction task.

Limitations and Future Steps

While the neural network model achieved the best performance among all classifiers, several limitations in the current approach suggest room for improvement. The dataset primarily consists of team-level aggregated statistics such as xG difference and PPDA, which, although informative, do not capture critical contextual factors like red cards, player injuries, or tactical substitutions that may drastically influence match outcomes. Moreover, the models do not account for temporal patterns or seasonal momentum—treating matches as independent observations rather than part of a dynamic sequence. Additionally, while class imbalance was partially addressed through the use of macro F1-score and class weighting, predicting losses remained a persistent weakness, especially in simpler models.

Looking forward, future work could integrate richer data sources, including play-by-play actions, player-level stats, and contextual features, to enhance model inputs. Incorporating time-series modeling could also help capture trends in team performance over time. More extensive hyperparameter tuning and the use of ensemble methods might further optimize predictive accuracy. Lastly, applying interpretability tools such as SHAP or LIME to the neural network would help bridge the gap between model performance and transparency—an important consideration in practical sports analytics applications.