FYS 4155 Project 3 - Predicting Premier League Match Results with Machine Learning

Hakan Abediy

December 18, 2023

Abstract

This project report aims to predict the full time results(home win, away win or draw) of Premier League matches using three machine learning algorithms: Logistic Regression, Feedforward Neural Network (FFNN), and XGBoost.The report is strucutred into the sections; introduction, theory and methods, results and discussions and conclusion.The report involves two cases. Case i includes only basic features and uses the dataset of Premier League matches from 2019/2020 to 2020/2021 season. Case ii includes a dataset from 2019/2023 to 2023/2024 and betting odds and elaborate preprocessing.Performance is measured using metrics like F1 , accuracy and precision scores. SKlearn's FFNN and especially XGBoost consistently shows superior performance in both cases. XGBoost and Sklearn's FFNN was able to achieve an F1 score of 0.62. Logistic regression could not give a coherent model in case i due to no feature engineering and class imbalance. The report shows that feature selection and data preprocessing are important to predict model performance. The results in this report aligns with results from existing literature.

1 Introduction

This project report is a supervised machine learning classification task focused on predicting the full time results of English Premier League football matches. The dataset spans the period from 2019 to 2023 and includes various features related to team statistics, match details, and betting odds. The primary goal is to predict the Full-Time Result (FTR) of each match, which can be 'H' (Home Win), 'A' (Away Win), or 'D' (Draw).

Predicting match outcomes in sports analytics holds significant importance across various domains, including sports management, betting and for recruitment or transfers.

The dataset provides match details, team statistics, and betting odds for every Premier League Match over several seasons.

The motivation for using machine learning is to further understand and analyze the complexities of football dynamics with all its randomness and non-linear relations. Machine learning models can capture these non-linear relations in a better way to further understand this global sport.

All the codes and results in this report are available on the GitHub repository:
https://github.com/hakanoa/FYS-4155-Project-3

2 Theory and Methods

2.1 Data Preprocessing

2.1.1 Dataset overview

The dataset is gathered from https://www.football-data.co.uk/ [4] and it provides detailed information about football matches in the English Premier League over several seasons. Each row represents a specific match. Each column represents a variety of attributes from the date of the match, team names; Full time results, half time results,team statistics, card information and even several betting odds.

The betting odds have been gathered the day before the matches are played.

The dataset consists of 113 columns/attributes and 760 rows/matches per season, and the dataset is updated after every round. There are 38 rounds in a season.

In this report there were to run cases where case i was the simple case and case ii was the more advanced case. Case i consists of match data for two seasons, from the 2019/2020 season to the 2021/2022 season. The only features that were used were the Home Team name, Away Team name and the day of the game. The data was not scaled. Case ii consists of match data for 3.5 seasons, from the 2019/2020 season to the 2023/2024 season. The features that were used were the features from case i plus betting odds. Additionally, form columns were added as a feature to capture the current strength of each team before a match. Form and confidence is a crucial predictor of sports performance [6]. Also, the data was scaled in this, and this is important when the dataset consists of many different features, each with a different range of values. The scaling will give meaningful comparisons between features.

2.1.2 Data cleaning and handling missing values

The features were converted from objects or strings to numbers to fullfill the algorithmic requirements and for computation efficiency.In this dataset there were no missing values.

The dataset was already neatly setup with one value in each cell.

2.1.3 Features and target variable

The main features used are the home team, away team, day of the game and betting odds.

The target variable is the full time results (FTR) and have three classes/outcomes, which are Home win (H or 0), Away win(A or 1) and Draw(D or 2). Therefore, this is a multiclass classification problem which differs from the binary one studied in Project 2.

2.2 Machine Learning Algorithms

Logistic Regression, FFNN and XGBoost were used in this report to predict the full time results of Premier League football matches based on certain features. This section describes the three algorithms.

2.2.1 Logisitc regression

Logistic regression is mostly suitable for binary classification. The code for Logistic Regression was reused from Project 2. The theory and algorithm implementation is still valid from Project 2. See the Theory and Methods Chapter in Project 2 [1].

2.2.2 FFNN

The code for FFNN was reused from Project 2. The theory for FFNNs in general, back prorogation, activation function and algorithm implementation is still valid from Project 2. See the Theory and Methods Chapter in Project 2 [1].

2.2.3 XGBoost

Boosting methods generally work by training predictors in a sequential manner, where each predictor attempts to rectify the mistakes made by its predecessor. There's a variety of these boosting methods available for use. One of them is called Gradient Boosting. An optimized implementation of Gradient boosting is available in the Python library XGBoost (extreme gradient boosting). [5]

XGBoost is an efficient and widely used implementation of gradient boosted trees, that adds a few more improvements to gradient tree boosting. One of the key enhancements is the addition of a regularizer on the tree complexity. This helps to control over-fitting, making the model more general and less likely to overreact to noise in the training data. Another significant improvement is the use of a second-order approximation of the loss function, rather than just a linear approximation. This allows XGBoost to capture more complex patterns in the data, leading to more accurate predictions.

The XGBoost optimizes the following objective:

$$L(f) = \sum_{i=1}^{N} l(y_i, f(x_i))) + \Omega(f) \tag{1}$$

Where

$$\Omega(f) = \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{J} \omega_j^2 \tag{2}$$

is the regularizer, where J is the number of leaves, and ≤ 0 and $\lambda \leq 0$ are regularization coefficients

2.3 Implementation and testing

This subchapter will explain the implementation and testing of Logisitc regression, FFNN and XGBoost which was implemented to recieve the results in 3.

2.3.1 Logisitc Regression

Logistic regression was implemented from scratch and using SKlearn. The code used to this was from project 2. The following describes the implementation and training of the

from scratch logistic regression algorithm. The logistic regression method includes a sigmoid activation function even though it's usually for binary classification problems.

The logistic-regression function takes the design matrix X, target variable Y, learning rate, batch size and the cost function as arguments. The logistic regression model is trained using stochastic gradient descent with mini-batch updates. It iterates over different combinations of learning rates and batch sizes, and for each combination, it shuffles the data and performs mini-batch updates to optimize the model parameters. The performance of each combination is evaluated based on the specified cost function (accuracy, precision, or F1 score).

The predict-proba function takes the design matrix X and model paramters beta as arguments. The function predicts the probability of the positive class using the sigmoid function.

The gradient function calculates the gradient of the cost function with respect to the model parameters and is used to update the gradient.

The training loop iterates over the different hyperparameters: learning rates, batch sizes to perform stochastic gradient descent with mini-bacthes for an N number of iterations. The performance of each combination is measured using a specified score type.

For Sklearn's logistic regression grid search using `GridSearchCV` is used to find the best hyperparamters which gives the best F1 score.

The model with the best hyperparameters is evaluated on the test data, and the metric are printed for both the logistic regression from scratch and SKlearn's algorithm.

2.3.2 FFNN

A FFNN was implemented from scratch and using SKlearn's MLPCLassifier.

The following hyperparamters were explored: activation functions, number of hidden neurons, learning rate, and number of epochs. They were found by bruteforcing several combinations of hyperparamters within a range. The range was defined by experimentation. The weights are set randomly.

The number of hidden neurons refers to the number of neurons in the hidden layers. It can significantly impact the performance of the model. Too few neurons can result in underfitting, while too many can lead to overfitting. Learning rate determines the step size at each iteration while moving toward a minimum of a loss function

Number of epochs refers to the number of times the learning algorithm will work through the entire training dataset

Only Sigmoid is the activations function which was tested for in the hidden layer. Only Sigmoid, ReLU and Leaky ReLU are activations functions which were choices to be used in the output layer.

2.3.3 XGBoost

The model is created by using XGBClassifier. It is created with the following initial hyperparameters: max depth, learning rate, gamma, n estimators, and reg lambda. These hyperparameters control the complexity of the model, the learning rate,the minimum loss reduction, the number of gradient boosted trees, and the L2 regularization term on weights,

respectively. Then, the model is trained using the fith method. A grid search is performed to find the optimal hyperparameters,that gives the highest F1-score. Thereafter, the model performance is evaluated using f1 score, accuracy score and precision.

The range of the hyperparamters when doing grid search was selected based on experimentation to strike the balance between not overfitting and not being too conservative.

Max depth determines the max depth of a tree. Deeper trees can model more complex relationships by adding more nodes, but as depth increases, the model becomes more likely to overfit. The choice of learning rate can should not to be too large to avoid overfitting and too small/conservative to avoid non-convergence.

Gamma is the minimum loss reduction required to make a further partition on a leaf node of the tree. Larger gamma-values will make the algorithm more conservative. [3]

reg lambda is the L2 regularization term where larger values make the model more conservative as a larger penalty is added to the loss function which is used to prevent overfitting. [3]

2.3.3.1 Choice of relevant activation functions and hyperparameters

GridSearchCV is a tool in the Scikit-learn library that automates the process of finding the optimal hyperparamters and activation functions.

The chosen hyperparameters and activation functions are given in Tables 1 and 2 and which were described in 2.3.1 to 2.3.3.

Table 1: Hyperparameters for Case i

Model	**Hyperparameters**
XGBoost	$\gamma = 0.5$, learning rate $= 0.1$, max depth $= 5$, reg lambda $= 0.005$
Logistic Regression (SKlearn)	learning rate $= 0.1$
Own Logistic Regression	learning rate $= 1$, batch size $= 16$
MLPClassifier	activation $=$ relu, $\alpha = 0.1$, hidden layer sizes $= (15,)$, learning rate init $= 0.01$, solver $=$ sgd

Table 2: Hyperparameters for Case ii

Model	**Hyperparameters**
XGBoost	$\gamma = 0.3$, learning rate = 0.3, max depth = 11, reg lambda = 0.1
Logistic Regression (SKlearn)	learning rate = 2
Own Logistic Regression	learning rate = 0.001, batch size = 32
MLPClassifier	activation = relu, $\alpha = 0.001$, hidden layer sizes = $(15,)$, learning rate init = 0.1, solver = adam

2.4 Performance metrics

The sections decribes the performance metrics, accuracy score, precision score and F1 socre, which were used in this report.

2.4.1 Accuracy score

We measured the performance of our data set by using accuracy score. The accuracy score is the number of correctly guessed targets t_i divided the number of targets n. It can be written as:

$$Accuracy = \frac{\sum I(t_i = y_i)}{n}$$

Where I is the indicator function and is 1 if $y_i = t_i$ and 0 otherwise. y_i is the output. [7]

2.4.2 Precision score

Precision score is a metric that measures how often a model correctly predicts the positive class, and it measures the proportion of true positive predictions among all positive predictions.

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

Precision is concerned with the correctness of positive predictions, while Accuracy considers the overall correctness of all predictions.Therefore, precision is a good measure when the cost of False Positive is high, while accuracy is a good measure when the prediction classes in the data are nearly balanced.

2.4.3 F1 score

F1 is a metric that combines precision and recall scores. F1 score can be interpreted as the mean of the precision and recall and their contributions are therefore equal to the score. F1 score can be expressed as:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

The F1 score is particularly useful when the dataset is imbalanced and there is a large difference in samples between the different classes. z

In summary, while accuracy is a good measure when the prediction classes in the data are nearly balanced, precision is more useful when the cost of false positives is high. The F1 score is a robust measure for imbalanced datasets, as it considers both precision and recall.

2.5 Confusion Matrix

Confusion matrix is a table layout that helps us visualize the performance of a specific algorithm. In a binary/2 classes classification problem the matrix would be 2 x 2. In a 3-class classification problem the confusion matrix is 3 x 3. Confusion matrices can be important to understand the nature of the predictions.

Each row of the matrix represents the instances of the predicted class. Each column represents the the instances of the actual class.The diagonal represents the the number of points where the predicted value is equal to the true value (i.e where we predict H and the true results was H). Elements that are not on the diagonal are values which are mislabeled by the classifier (i.e where we predict H and the true result was D or A). The higher the diagonal values of the confusion matrix, the better, indicating many correct predictions.Confusion matrices help us see how many predictions were correct and where the model tends to make mistakes. Confusion matrices can be important to understand the nature of the predictions.

The confusion matrices in Figure 3 and 1 were plotted using sklearn.metric's confusion metric method.

2.6 Class imbalance

A significant challenge in classification analysis is the potential issue of class imbalance. Class imbalance occurs when the classes in the target variable are not represented equally. For instance, most matches result in a home win, fewer in an away win, and even fewer end in a draw. The reason why home teams win more often are for example due to more fan support, familiarity and travel fatigue. This causes the dataset to be imbalanced. If the model is trained on this imbalanced data, it might become biased towards predicting a home win, simply because that's the most common outcome in the training data. This could lead to poor performance when predicting away wins or draws, even when the match conditions favor these outcomes. It's important to address this imbalance to have an accurate model.

One way to address class imbalance and balance the classes is to use Random Oversampler and apply it to X_{train} and y_{train}, which oversamples the minority classes A/1 and D/2. Extra draws and Away wins are generated by randomly duplicating examples from the original

minority classes. However, oversampling can also lead to overfitting of the training data, so it's important to be careful.

3 Results

This Chapter will show an overview of the results of Case i and Case ii. The types of results which are displayed in this Chapter are performance scores for Logsitic Regression, FFNN and XGBoost, overview of feature importance and confusion matrices for the algorithms for both cases.

3.1 Case i

Table 3 shows an overview of the algorithm performance for XGBoost, SKlearn's FFNN/MLPClassifier, our own FFNN ,our own Logisitc Regression and SKlearn's Logistic Regression, as seen in Section 2.1.

In case i the XGBoost algorithm outperforms the others with the highest F1 score of 0.44 and a precision score of 0.47. The SKlearn FFNN and Own FFNN show similar performance with F1 scores of 0.37 and 0.36, respectively. However, the logistic regression models, both the SKlearn and the custom implementation, performs the worst with lower F1 and precision scores.

Table 3: Algorithm Performance Sorted by F1 Score for Case i

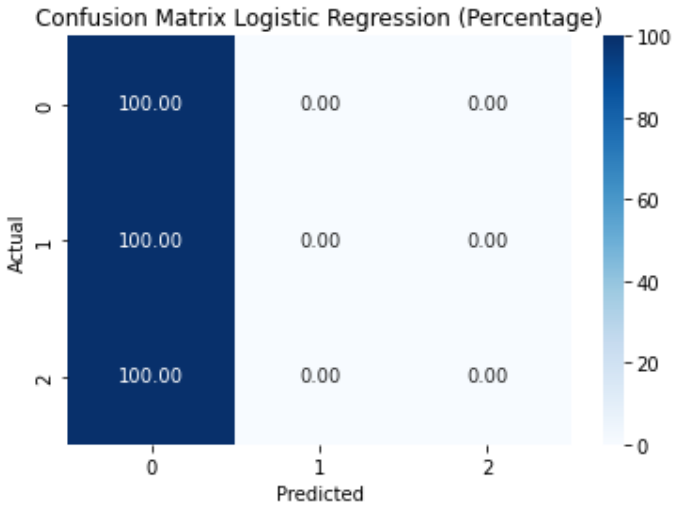
Algorithm	F1 Score	Accuracy Score	Precision Score
XGBoost	0.44	0.44	0.47
SKlearn FFNN	0.37	0.44	0.38
Own FFNN	0.36	0.40	0.32
Own Logistic Regression	0.28	0.44	0.24
SKlearn Logistic Regression	0.21	0.41	0.19

Figure 1 shows the confusion matrices of Scikit-learn's Logistic Regression, Scikit-learn's MLPClassifier, and XGBoost for Case i in percentage.

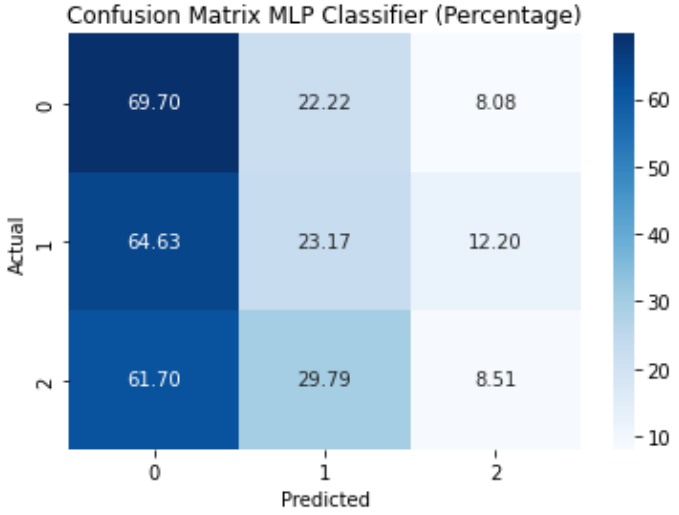
For the confusion matrix of Scikit-learn's Logistic Regression, as seen in Figure 1(a), the algorithm predicts all values to be 0/H. The actual values of all instances also belong to H.

Figure 1(b) shows that Scikit-learn's FFNN correctly predicts the H, A, and D classes 69.70%, 23.17%, and 8.51% of the time, respectively. By looking at the mean of the columns, the H class is predicted over 60%, A 23%, and D 10% of the time.

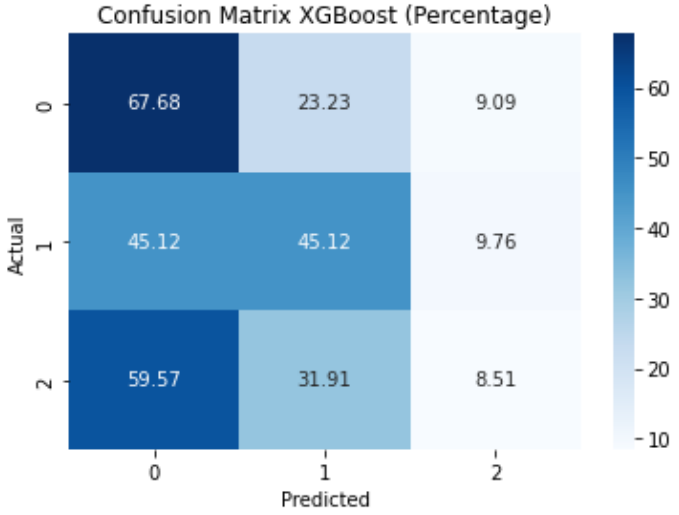
Figure 1(c) shows that XGBoost correctly predicts the H, A, and D classes 67.68%, 45.12%, and 8.51% of the time, respectively. By looking at the mean of the columns, the H class is predicted approximately 58%, A 33%, and D 9% of the time.



((a)) Confusion matrix for Case i using SK's Logistic Regression



((b)) Confusion matrix for Case i using SK's FFNN/MLPclasifier



((c)) Confusion matrix for Case i using XGBoost

Figure 1: Confusion matrices for Case ii for the different algorithms

3.2 Case ii

In case ii the class distribution in the train set Table 4 shows the results for the more advanced case ii, as seen in Section 2.1.

Table 4 shows the performance of XGBoost, SKlearn's FFNN/MLPClassifier, our own FFNN ,our own Logisitc Regression and SKlearn's Logistic Regression by their F1 scores for Case ii. The XGBoost and SKlearn FFNN algorithms have the highest F1 scores of 0.62, with accuracy scores of 0.61 and precision scores of 0.63 and 0.65, respectively. The Own FFNN algorithm has a lower F1 score of 0.55 but a higher accuracy score of 0.63 and a precision score of 0.49. The SKlearn Logistic Regression algorithm has an F1 score of 0.58, an accuracy score of 0.60, and a precision score of 0.62. The Own Logistic Regression algorithm has the lowest scores with an F1 score of 0.51, an accuracy score of 0.50, and a precision score of 0.35.

Table 4: Algorithm Performance Sorted by F1 Score for Case ii

Algorithm	**F1 Score**	**Accuracy Score**	**Precision Score**
XGBoost	0.62	0.61	0.63
SKlearn FFNN	0.62	0.61	0.65
Own FFNN	0.55	0.63	0.49
SKlearn Logistic Regression	0.58	0.60	0.62
Own Logistic Regression	0.51	0.50	0.35

Figure 2 shows a bar plot of the most important features for Case ii. The range of the values is from 0 to 1. The most important features are `AvgH`, `AT_Avg_FTAG`, `HT_Avg_FTHG`, `HT_Avg_FTAG`, `AT_Avg_FTHG`, as well as other betting odds indicators. Only the 10 most important features are used to create the model. `AvgH` is the Market average home win odds and is clearly the most important feature. `AT_Avg_FTAG` is the Away team's average away goals over the 3 previous matches. `HT_Avg_FTHG` is the home team's average home goals over the 3 previous matches.

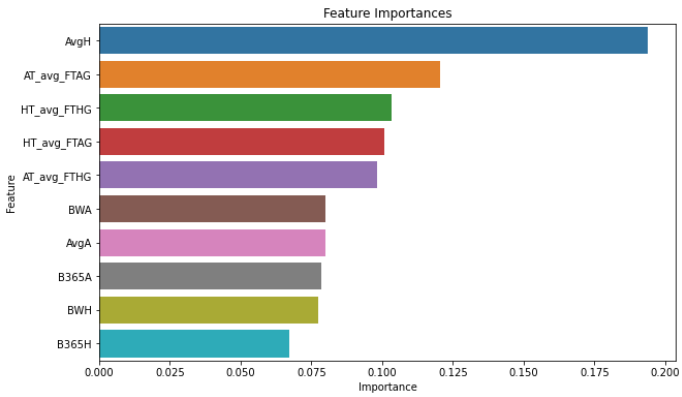


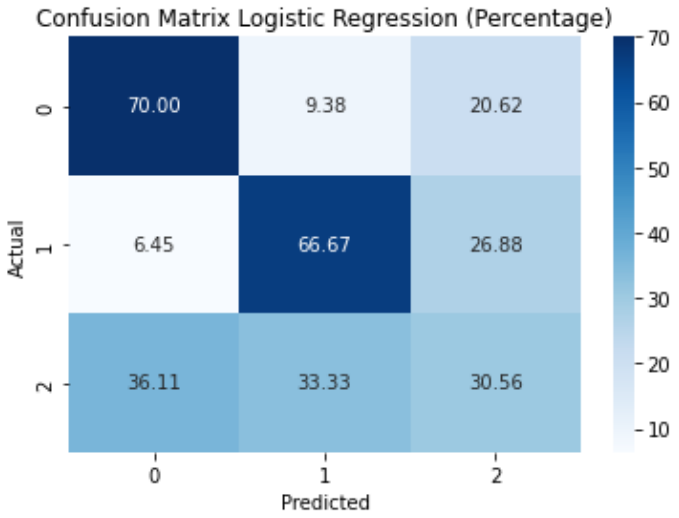
Figure 2: Feature importance for Case ii when using XGBoost

Figure 3 shows the confusion matrix of Sklearn's Logistic Regression, SKlearn's MLP-Classifier and XGBoost for Case ii.

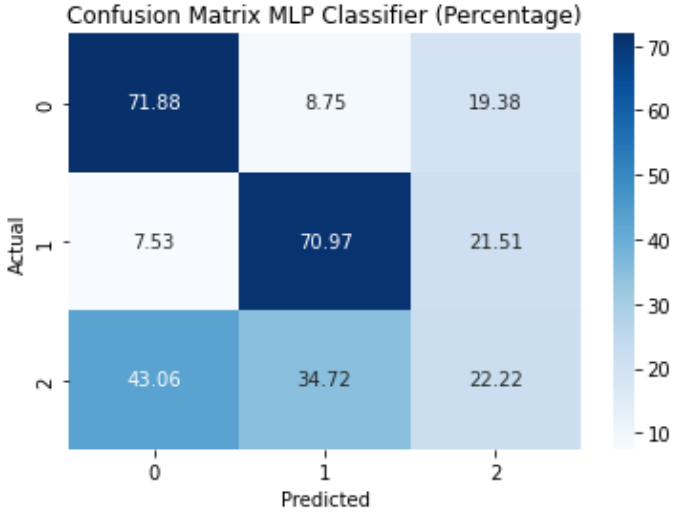
Subfigure 3(a) shows that Scikit-learn's Logistic Regression correctly predicts the H, A, and D classes 70.00%, 66.67%, and 31.56% of the time, respectively. By looking at the mean of the columns, the H class is predicted approximately 38%, A 37%, and D 26% of the time.

Subfigure 3(b) shows that Scikit-learn's Feedforward Neural Network (FFNN) correctly predicts the H, A, and D classes 71.88%, 70.97%, and 22.2% of the time, respectively. By looking at the mean of the columns, the H class is predicted approximately 41%, A 38%, and D 21% of the time.

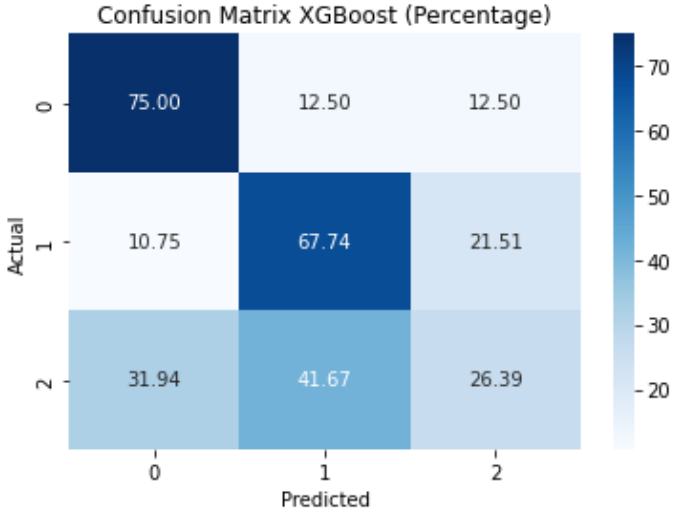
Subfigure 3(c) shows that XGBoost correctly predicts the H, A, and D classes 75.00%, 67.74%, and 26.39% of the time, respectively. By looking at the mean of the columns, the H class is predicted approximately 39%, A 41%, and D 20% of the time.



((a)) Confusion matrix for Case ii using SK's Logistic Regression



((b)) Confusion matrix for Case ii using SK's FFNN/MLPclasifier



((c)) Confusion matrix for Case ii using XGBoost

Figure 3: Confusion matrices for Case ii for the different algorithms

4 Discussions

This section will discuss the Results section 3.1 by focusing on the feature importance analysis, confusion matrices, performance metric, and the problem of class distribution Furthermore, the results will be linked with existing literature and a critical assessment of the algorithms and their limitations.

4.1 Feature Importance Analysis

Figure 2 shows the most important features in Case ii. It's logical that betting odds and the form of the teams represented by the `HT/AT_avg_xxx` features are important features. Anyone who watches sport knows that form and confidence are important aspects of the athlete's performance. Form represents the current strength level of a team. A team with a high current strength level but with a poor historical strength level can be a team with a low current strength level but with a poor historical strength level. This is especially true for a league like the Premier League, which is famous for everyone being able to beat everyone. Average betting odds are based on the opinion of many bookmakers and are the market consensus. However, the role of the bookmaker is to manage risk to ensure a profit, but they also have a strong incentive to ensure accurate odds.

In Case ii, Scikit-learn's `SelectKBest` was employed to select only the top 10 features. It's plausible that using fewer features could yield better full-time results predictions for certain algorithms, as an excess of features can introduce noise that degrades the model's performance. However, this aspect was not thoroughly investigated.

Interestingly, none of the most important features of Case ii are included in Case i. Case i solely relies on data about the home team, away team, and match day, without involving any feature engineering. Consequently, the algorithm struggles to optimally interpret the provided information and their relationships. The significantly poorer performance metrics for Case i compared to Case ii underscore the importance of feature engineering. It's possible that more feature engineering should have been implemented in Case ii) also to allow the algorithms to better understand the relationships between the features, thereby improving predicting performance.

4.2 Performance metrics analysis

The F1 score was chosen as the main performance metric as in the prediction of full-time results there will be a need to handle class imbalance as every outcome is not equally probable. The F1 score which focuses on precision and recall can provide good insight. Accuracy on the other hand can be more misleading and is less robust than F1 score.Accuracy score also can struggle to handle imbalance. F1 score is also a good metric to use when false positives and negatives are significant and this is important in sports betting where you want to a false positive and negative to make you lose money.

In comparison of the algorithm's performance on the datasets, it is observed that that the algorithms' performance vary significantly between the two cases as seen in Tables 3 and 4.

By summarizing the results of Case i) and ii), XGBoost and SKlearn FFNN show superior performance in terms of F1 score and precision. Logistic regression performs the worst in terms of F1 score and accuracy score for both cases, but it performs significantly worse compared to the other algorithms in Case i) than Case ii). For case i) the mean F1 score, accuracy score and precision score for all the algorithms is 0.34, 0.43, 0.32. For case ii) the mean F1 score, accuracy score and precision score for all the algorithms is 0.58, 0.61, 0.57. From case i) to case ii) the F1 score, accuracy score and precision score increased with 71%, 42%, 78%, respectively.

4.3 Class distribution

In Case i), based on our testing set the class distribution is H: 43.4 %, A: 36.0 % and D: 20.6 %. In Case i) in Figure 1 the three algorithms tested severely over predicts Home wins/ H and under predicts draws/D compared to the training data. MLPClassifier and XGBoost predicts home win 59.0 % and draw 9.5 % of the time, on average.

Furthermore, Logistic regression predicts 100 % instances to be a home win and and actual instances to be a home win. This method is not able to give a coherent model. Reasons for this might be that especially Logsitic Regression struggles when there is class imbalance due to the nature of football matches were especially Home wins are over twice as likely as draws. The class imbalance makes the model have a bias towards predicting the home team to win. Also, scaling can play a role in the inchorent model by Logistic Regression.

In Case ii),based on our testing set the class distribution is H: 49.2 %, A: 28.6 % and D: 22.1 %. Logsitic Regression, MLPClassifier and XGBoost often underpredicts home wins and over predicts away wins. The three algorithms predict home win 39.3 %, away win 38.7 % and draw 21.1 % of the time, on average.

The Logistic Regression model manages to predict the full time results in a coherent manner compared to in Case i). This most likely is because of the oversampling of the minority classes which balances the class distribution. The oversampling of the minority classes might also have lead to the all the three algorithms to tend to under predict home wins and over predict away wins.

Both XGboost and MLPclassifer are almost equally able to predict the actual home wins and away wins with approximately 71 % of the time. However, Logistic Regression are are most able to predict the actual draws 30.56 % of the time.

4.4 Results and existing literature

This section investigates the results and if they are in line with existing literature.

A study at Mumbai University that used machine learning to predict the full time results of Premier League matches found that XGBoost was one of the best performing algorithms. The F1 score was used as the performance metric. [8]

The performance of FFNN is consistent with literature. Another study on predicting the outcome of football matches found neural networks to be a efficient way of predicting full time results. [9]

A study that used a custom Logistic regression model to predict the accuracy of premier league football matches achieved an accuracy of 69.5 %. The accuracy score in my model was 60 %. However, accuracy is not an robust metric in this case and I have not investigated what kind of feature engineering and dataset which was used in that study. [2]

This shows that the results are in line with existing literature and that these algorithms are used in predicting football match outcomes. It also shows the importance of feature engineering and the quality of the data.

4.5 Critical assessment of the algorithms and their limitations in predicting Premier League outcomes

This part of the discussion will consider an assessment of the the algortihm's strengths and limitations.

Logistic regression is a simple and interpretable model. However, its simplicity is also its limitation. It seems to do fine with binary classification but struggles more with multi-class classification. Logistic regression does not fit well for complex datasets with non-linear relationships, such as team strategy Furthermore, it doesn't handle interactions between features unless they are manually created. This restricts its effectiveness. However, logistic regression performed fairly well for case ii.

The FFNNs are capable of modeling complex, non-linear relationships and can also automatically learn feature interactions. This makes it efficient in predicting football match outcomes. However FFNNs, require more careful hyperparamter tuning than Logistic Regression and lack interpretability. This is because it's not transparent in its decision making process. Furthermore, the weights are set randomly in our FFNN models and therefore the results will differ from one run to the next.

Like with FFNN, XGBoost is capable of modelling non-linear relationships, automatic feature interactions learning, but need careful tuning and lack intepretability. But it also have built-in regularization to prevent overfitting.

Sigmoid was chosen instead of softmax as the activation function as it gave better performance scores than softmax, but the results of the softmax run are not included in this report. This was the case even though Sigmoid is often used for binary classification and softmax is used for multi-class classification.

In summary, a high-performing model doesn't only depend on the choice of model and hyperparameters. These results highlight the importance of preprocessing steps such as feature engineering, oversampling, and scaling in improving the performance of machine learning algorithms. These include the interpretability of the model, the computational resources available, and the specific requirements of the project. For instance, despite their lower performance metrics, simpler models like logistic regression might be preferred to achieve an okay and simple to interpret the result.

5 Conclusion

The report examined the application of Logistic Regression, FFNN, and XGBoost in predicting outcomes of English Premier League matches. The analysis was conducted in two

cases: a simpler approach (Case i) with fewer features and a more advanced approach (Case ii) with additional features like betting odds and team form.

The key findings were that XGBoost and Sklearn's FFNN was superior in terms of F1 score and precision across both cases. Logistic Regression, while simple and interpretable, showed lower performance, highlighting its limitations in handling complex, non-linear data.

The importance of data preprocessing such as feature engineering, oversampling of minority classes and scaling was vital in the performance of the model, as seen in the performance increase from case i to case ii. Betting odds was seen as the most important features and especially average betting odds over several bookmakers.

Furthermore, addressing class imbalance by oversampling the minority classes was implemented to reduce model bias. This especially reduced the bias in predicting home wins, but also slightly increased the bias in predicting away wins.

The recommendations for future work are to explore more sophisticated features and feature engineering such as player performance, real time data or to implement sentiment analysis. Furthermore, the datatset can be tested on other algorithms such as SVG.

References

[1] Hakan Abediy. Fys 4155 project 2 - classification and regression, from linear and logistic regression to neural networks. https://github.com/hakanoa/FYS-4155---Project-2/blob/main/FYS%204155%20Project%202%20-%20hoabediy.pdf, 2023.

[2] Devansh Malhotra Rachna Jain Preeti Nagrath Ashutosh Ranjan, Vishesh Kumar. Predicting the result of english premier league matches. *Springer Professional*, 2021.

[3] XGBoost Contributors. Xgboost documentation, 2023.

[4] Football-Data.co.uk. Football data, 2023. Accessed on: 2023-11-22.

[5] Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2019.

[6] Kate Hays, Owen Thomas, Ian Maynard, and Mark Bawden. The role of confidence in world-class sport performance. *Journal of Sports Sciences*, 27(11):1185–1199, 2009.

[7] Morten Hjorth-Jensen. *Applied Data Analysis and Machine Learning*. 2021. Jupyter Book.

[8] Shubham Patil. Predicting football match results using machine learning. *International Journal of Creative Research Thoughts (IJCRT)*, 23(4):StartPage–EndPage, 2023.

[9] Muntaqim Ahmed Raju, Md. Solaiman Mia, Md. Abu Sayed, and Md. Riaz Uddin. Predicting the outcome of english premier league matches using machine learning. In *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, pages 1–6, 2020.