Project Report

Artificial Intelligence in Robotics

Name: Dimitrios Kapsos
Project Title: EPL Match Outcome Predictor Using ML Models
E-mail: kapsosdim@gmail.com

1 Project Overview

In this project, three distinct Machine Learning models will be evaluated on a rather difficult task, that is predicting a football match outcome. Specifically, given as information the half time result, the current season, the home and away team strength and some recent form, the models will have to predict whether the home team wins, the away team wins or if it's a draw. This project was inspired from a YouTube video, however the approach taken is not the same. The video can be seen here.

As it is well known, football matches are random and anything can happen at any part of the game. So, even with the information about half time score, there is still plenty of time for an upset, going against all odds. Keeping this in mind, our end expectations are not that high. After all, if we were able to predict the match outcome by building a simple ML model, we could easily make millions by betting on that!

The dataset used can be found in kaggle in this link. It contains English Premier League football matches from season 1993-94 to 2021-22 and it will be further analyzed below.

For the programming language, **Python** was preferred. Alongside Python, various libraries were also included. For example, *pandas* for handling the dataset, *numpy* for arrays and various operations, *PyTorch* and *scikit-learn* for the ML models implementation.

The whole script is written in a jupyter notebook along with a few explanations and comments. The screenshots used here, can also be found in the notebook. Also, the project is uploaded in GitHub here. For the setup instructions, please refer to the README file.

2 Implementation Details

2.1 Overview of the Dataset

As mentioned earlier, the dataset contains data from 29 seasons. In each season, all the games are recorded. In Table 2.1 a single line containing the important for us information can be viewed. The dataset itself contains even more information about each game, like the referee,

the home team corners taken etc, but for simplicity only the information on the Table will be used to generate the predictions.

Season	Home Team	Away Team	HT Score	FT Score	FT Result
2008-09	Liverpool	Everton	0 - 0	1 - 1	D

Table 1: Example taken from the dataset.

Right from the start, the dataset suffers from some minor data loss. In seasons 1993-94 and 1994-95 the half time score is not recorded, while season 2021-22 is incomplete. Even though we could keep the last incomplete season, since all the data are still intact, it was decided to delete these three seasons from the dataset.

The next step involves transforming the categorical data into integers that the models can work with. For the output label, the final result, this is rather simple. The home win will be represented as '1', the draw as '0' and the away win as '2'. The season feature also gets its unique number ranging from '1' for the first season to '27' for the last. Lastly, the teams were given a number based on their total performance throughout all of the seasons. By calculating all their gathered points, strong teams are given bigger numbers while weaker teams smaller. For example, "Chelsea" is given number '46' while "Blackpool" number '2'.

As it is, there are not many features for the model to learn the relationship between input and output. In order to assist with that issue, six more features are being added per game. These are about each team's recent form, the average points collected and the goals scored and conceded from the home and away team. The average is calculated from the five previous games.

Another important aspect of our dataset is that EPL games are heavily home-win biased something that can be calculated easily by our dataset. Specifically, in our 9880 recorded games, 46% are won by the home team, 28% are won by the away team and the rest are draws. To restore the balance, we will need to add class weights, so that the models don't learn to bias the home-win result.

Finally, the dataset is split using 80% for training and the rest 20% for testing purposes.

2.2 Models and Results

The three models tested on the above dataset are a Random Forest Classifier, a XGB Classifier and a MLP with embedding layers. Each of the three offer different advantages with regard to the rest and all together they can sum up the general expected performance on the dataset.

2.2.1 Random Forest Classifier

The simplest model trained is the Random Forest Classifier. It works by taking an instance from the training set and passing it through many decision trees. Each of the tree decides on what class the instance refers to and in the end the majority of the trees' votes wins.

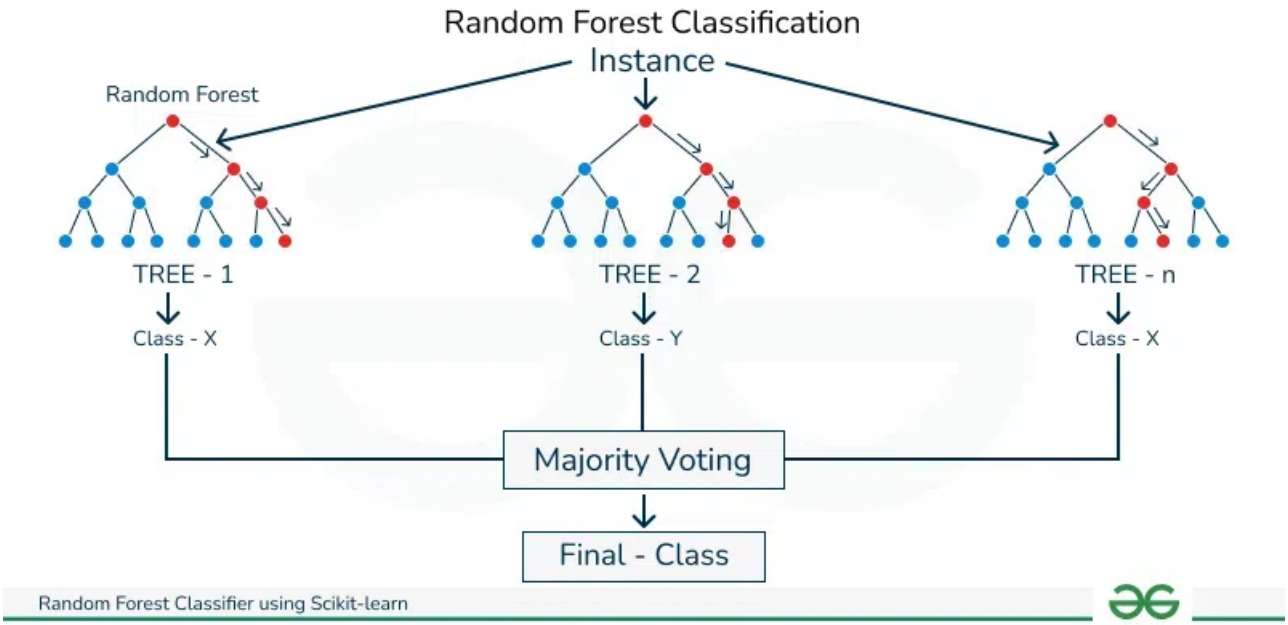

Figure 1: Random Forest Classifier

By backpropagation, the way the trees decide is improved. The image from GeeksforGeeks summarizes this brilliantly and can be viewed in Fig. 1.

In our case, there are going to be 300 decision trees with a maximum depth of 20. These numbers can be modified to achieve better performance but after experimentation, the difference in the error is negligible. Also, class weights are added to balance the dataset and prevent biases.

After fitting and testing the model, the achieved accuracy is 62.2%. By the term accuracy in out task we refer to how many labels were correctly predicted divided by the total number of predictions made. The accuracy however does not tell the whole story. To better view the results, the confusion matrix is provided in Fig. 2. The precision metric is also provided, by the precision term we refer to the number of correct predictions made for one class divided by the total number of predictions for the same class (e.g how many predicted draws were actually draws):

1. Draw : 38.8%
2. Home Win : 73.9%
3. Away Win : 64.0%

To better visualize which feature is the most important, the list of feature importance can be calculated after the model has been trained. For brevity the matrix is not presented here but can be viewed in the source code.

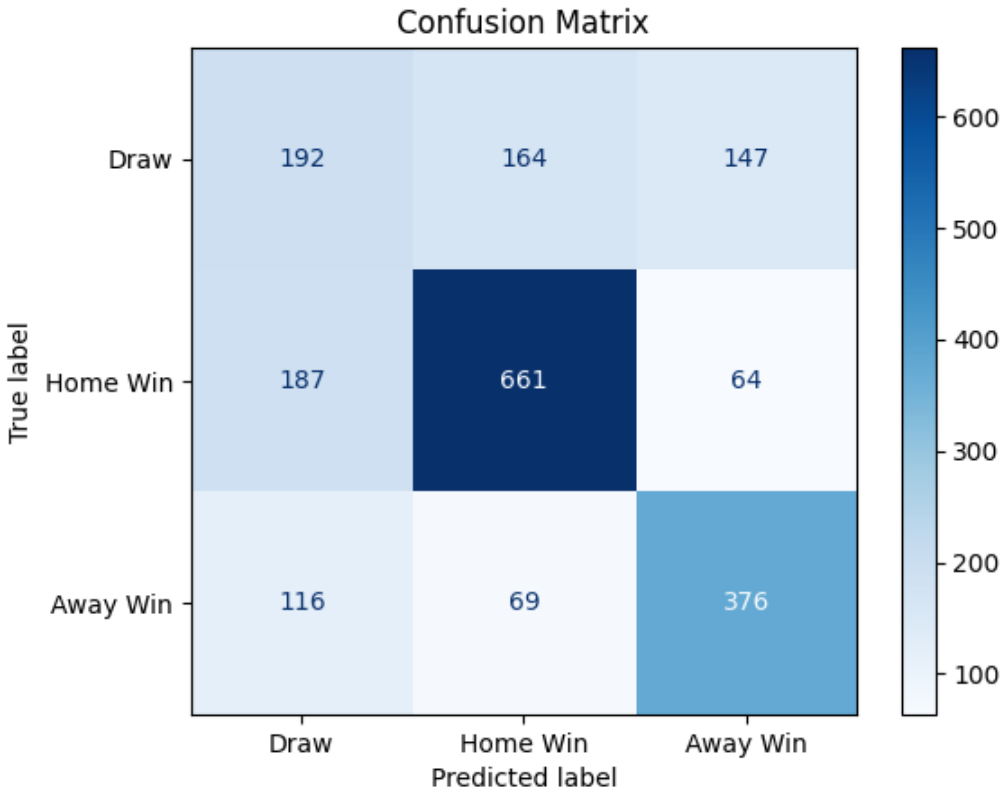

Figure 2: Confusion Matrix for Random Forest Classifier

2.2.2 XGB Classifier

Extreme Gradient Boosting or in short XGB is a type of ensemble learning method that combines multiple weak models to form a stronger model. As GeeksforGeeks mention, XGB uses decision trees, which are combined in an effort to improve the end result.

After instantiating the model, tailoring it for multi-class classification and computing the class weights we are ready to set it in motion. The accuracy score in this case is 61.3%, the confusion matrix can be viewed in Fig. 3 while the precision score is:

1. Draw : 38.2%
2. Home Win : 72.2%
3. Away Win : 63.6%

As with the Random Forest Classifier, we can once again tweak all the parameters (i.e number of estimator, max depth etc.) to achieve better performance. Also, the feature importance is available here as well but is excluded for brevity.

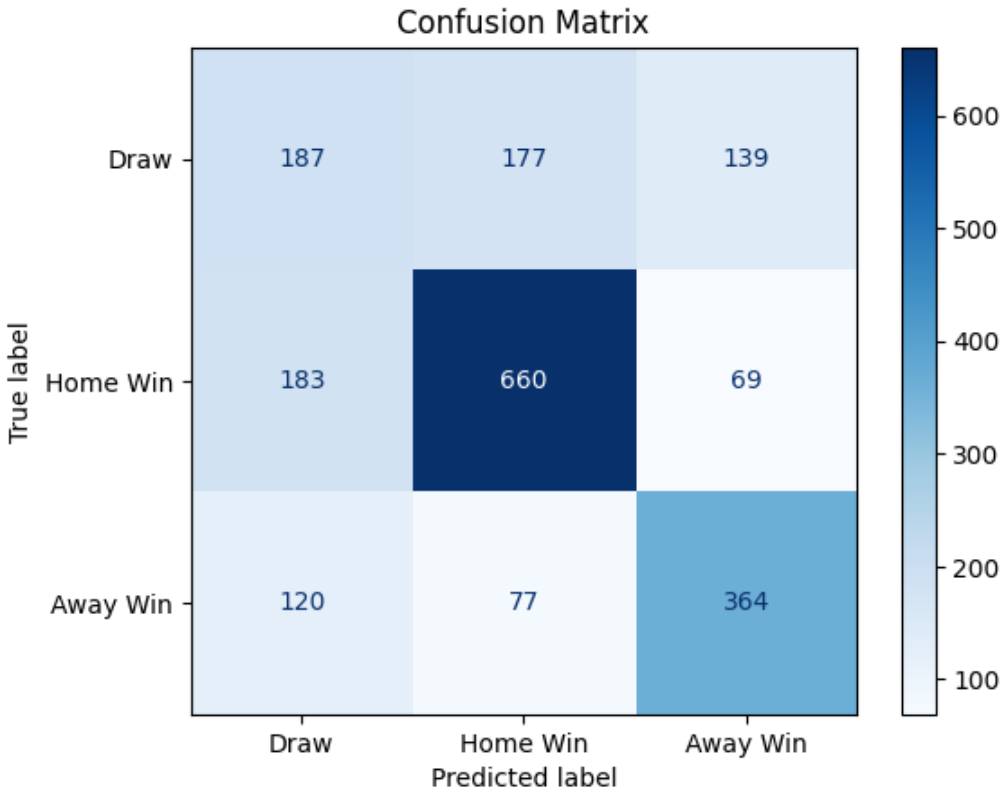

Figure 3: Confusion Matrix for XGB Classifier

2.2.3 Multi-Layer Perceptron

The last model is the most interesting one. The MLP model, gives the developer full control of everything. For starters, we will implement all the training and testing functions, which will enable the model to train smoothly. These functions are designed for batch training, since it has proven to be more effective. Additionally, an early stopping class has been implemented to stop training when no progress has been made and a function to plot the training and testing curves of loss and accuracy.

For the MLP, we will not use the season and teams' ranks as pure integers. Instead, we will create three trainable embedding vectors (one for each feature) which will enable the model itself to learn the patterns in the dataset. Other than that, the model consists of three linear layers containing (32 - 16 - 8) hidden neurons. The activation function used is the Rectified Linear Unit. Between each layer, we also use batch normalization and dropout to prevent overfitting and improve regularization.

After creating the training and testing dataset, we also calculate the class weights. These are then passed to the loss function, which in our case makes sense to be the CrossEntropyLoss. The optimizer used is AdamW and the learning rate scheduler is the ReduceLROnPlateau. The batch size is 32, the initial learning rate is 10^{-3} and the model is trained for a maximum of 100 epochs.

The training progress can be viewed in Fig. 4. The accuracy achieved is 62.9%, the confusion matrix can be viewed in Fig. 5 and the precision score is:

1. Draw : 40.4%
2. Home Win : 77.3%
3. Away Win : 70.1%

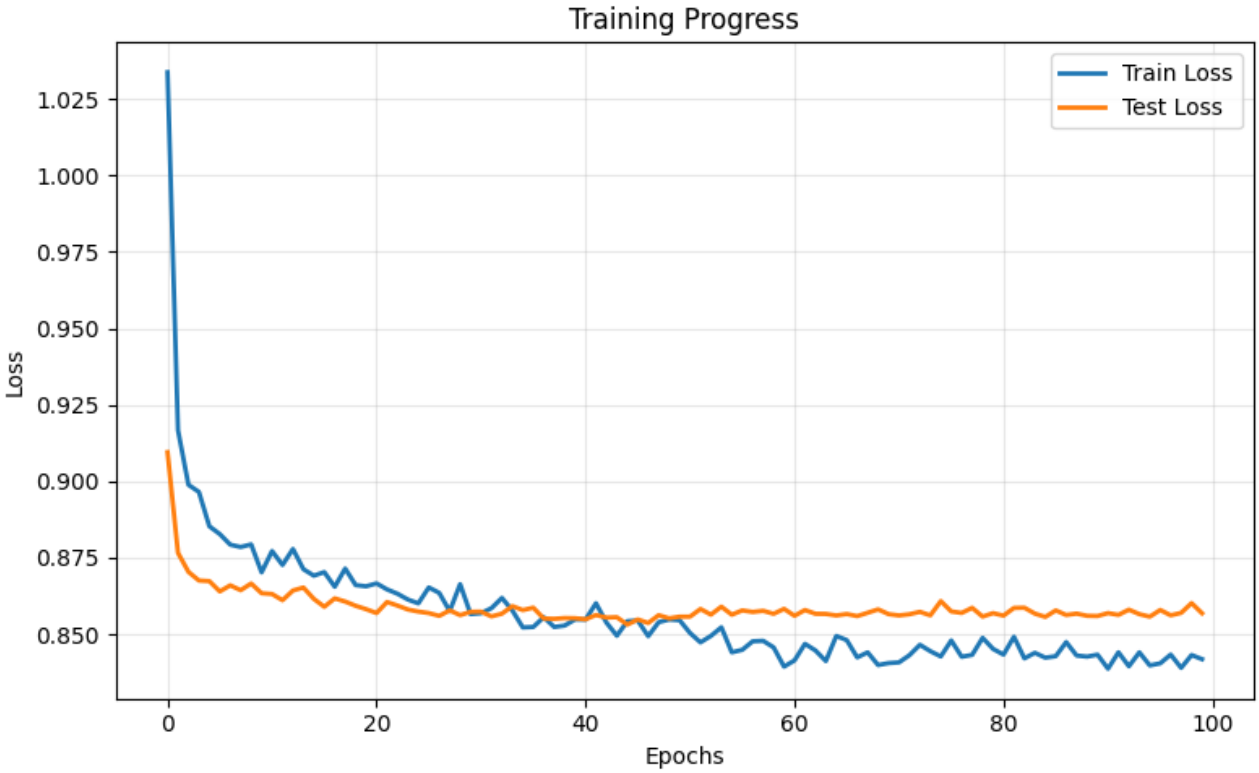

Figure 4: Training and Testing Loss

3 Results and Conclusion

The three models presented similar performance, which can be further improved provided the right parameters are chosen. Since the task includes a great amount of randomness, the accuracy score of $\approx 63\%$ is to be expected. Further work might include performing a grid search to find the optimal hyperparameters, or implementing cross validation mechanisms to ensure robustness of the models.

As mentioned earlier, all the code outputs along with some extra metrics and plots (i.e. recall, f1-score, feature importance, accuracy plot for the MLP) can be viewed in the notebook.

To conclude this project, the three models were successfully trained to predict the outcome of EPL football games using as input various statistics. This project aimed to show understanding of fundamental ML and Artificial Intelligence concepts. All the references used are marked within this review with a link to the corresponding site.

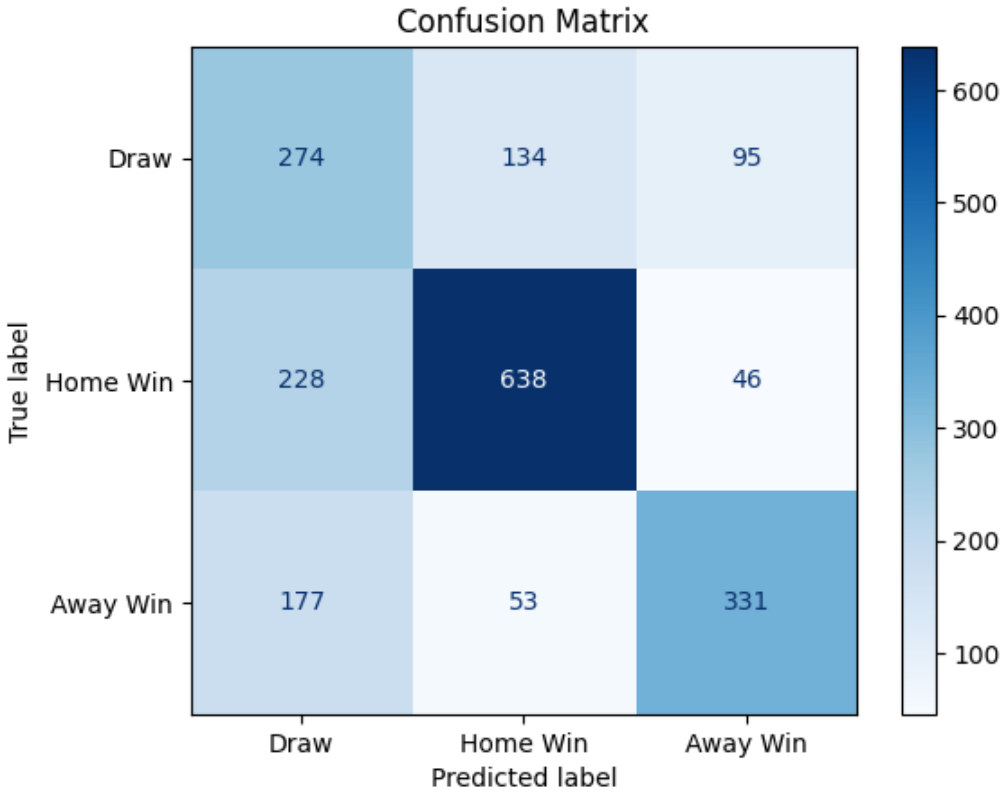

Figure 5: Confusion Matrix for Multi-Layer Perceptron Model