Stockholms universitet

A Comparative Study of Predictive Power in Football Match Outcomes

Gabriel Lindqvist

Kandidatuppsats i matematisk statistik
Bachelor Thesis in Mathematical Statistics

Kandidatuppsats 2025:12
Matematisk statistik
Juni 2025

www.math.su.se

Matematisk statistik
Matematiska institutionen
Stockholms universitet
106 91 Stockholm

Matematiska institutionen

Stockholm University

Mathematical Statistics
Stockholm University
Bachelor Thesis **2025:12**
http://www.math.su.se

A Comparative Study of Predictive Power in Football Match Outcomes

Gabriel Lindqvist*

June 2025

Abstract

This thesis investigates the use of machine learning methods to predict outcomes of football matches in the English Premier League. Specifically, the study compares the performance of multinomial logistic regression and random forest classifiers on a three-class prediction task, where the outcomes are either home win, draw or away win. The analysis is based on data that includes prior seasons starting from the 2016/2017 season up to the current 2024/2025 season. Predictive performance is evaluated using accuracy, class-specific F1 scores, and macro-averaged F1. To ensure robust results, each classifier was trained and tested across 10 independently stratified train/test splits. The results show that both classifiers performed similarly on the majority classes, but struggled to accurately classify draws. Further analysis of feature importance and predicted class probability distributions highlights challenges associated with class imbalance and limited separation in the feature space. The findings underline the importance of feature engineering for this particular prediction task.

*Postal address: Mathematical Statistics, Stockholm University, SE-106 91, Sweden. E-mail: gali8113@student.su.se. Supervisor: Johannes Heiny, Ola Hössjer.

Acknowledgements

I would like like to start off with expressing my sincerest gratitude to my supervisors, Ola Hössjer and Johannes Heiny, for their guidance and continuous encouragement throughout this thesis project. Even when I fell behind my original timeplan, their positive attitude helped me stay motivated.

I would also like to acknowledge the use of ChatGPT by Open AI. The tool was used mostly for code debugging, but also for proofreading.

Contents

1 Introduction

Predictions in sports is a common phenomenon that many people have either encountered or engaged in. This can be anything from informal guessing, participation in fantasy leagues, or placing online sport bets. Today, most competitive sports are inherently data-driven, generating large volumes of information, and is often widely accessible for anyone who wants to understand certain outcomes in sports. In the case of football, the supply and demand of data has been extensive. For example, professional football clubs utilize data to gain an edge against its competitors [4].

This study uses data from the English Premier League (abbr., EPL), spanning from the 2016/17 season to the current 2024/2025 season. One season consists of 20 competing teams, where each team plays all other teams twice (home and away). This means that each team will play 19 home games and 19 away games (38 games in total) in a season. A single match has three possible outcomes: home win, draw, or away win.

A known concept used in football is *home advantage*, which refers to the team playing at home has an advantage over its opponent. This was evident when looking at the data set used, with outcomes resulting in home wins being dominant and with draws being the most uncommon outcomes. The nature of this imbalance in the data is one of the biggest challenges for machine learning methods when trying to predict match outcomes, as reported in Choi [3]. There are methods that can be used to try and mitigate imbalance. Two common techniques involve sampling techniques or assigning class weights [11].

The main focus in this thesis is to compare the performance of two machine learning methods when predicting football match outcomes: *multinomial logistic regression* and *random forest*. Both of these methods can handle multiclass classification problems, which are well-suited when trying to predict among three possible outcomes. Multinomial logistic regression is a generalization of the linear classification model logistic regression [9]. Random Forest, introduced by Breiman [2], is an ensemble method which uses multiple non-linear decision trees.

2 Theory

This section serves to present the theoretical background of the methods used in this thesis. Unless otherwise stated, the theory presented is based on Hastie [7] and Lindholm [9].

2.1 Supervised Learning

This section is based on Supervised learning, a category of machine learning that focuses on *learning* a function from labeled data. The goal is to model

the relationship between input $\boldsymbol{x}$ and output y from *training data*, such that the model generalizes well to new unseen data.

Supervised learning problems are typically divided into two categories, depending on the type of y. When y is continuous, we call this a regression problem. On the contrary, when y is categorical, this is commonly known as a *classification problem*. This thesis focuses on *multiclass* classification problems, where the response variable can take on more than two unordered categories. Consequently, the theoretical background presented below will primarily cover the classification case.

Let $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ denote the *training data set*, consisting of n observations. Each pair of observations $(\boldsymbol{x}_i, y_i) \in \mathcal{T}$ includes a *feature vector* $\boldsymbol{x}_i \in \mathbb{R}^p$ of dimension p (also called predictors), and a known label (or class) $y_i \in Y$, where Y denotes the set of possible class labels. These observations are then used to learn a function $f : \mathbb{R}^p \to Y$. This function is known as a *classifier*, which maps input vectors to class labels.

The next step is to evaluate how well the classifier generalizes this relationship. This is commonly done using a *validation set*. A validation set is a set of observations that the classifier has not seen. Let $\mathcal{V} = \{(\boldsymbol{x}_i^*, y_i^*)\}_{i=1}^m$ denote the validation set, consisting of m observations. For each unseen input $\boldsymbol{x}_i^*$, denote the classifier's predictions as $\hat{f}(\boldsymbol{x}_i^*) = \hat{y}_i$. Each prediction is then compared to the true value y_i^*, which is used to assess the predictive power of the classifier.

In summary, the classifier attempts to capture the relationship between the features and the output class, such that it can predict the class label for a new observation $\boldsymbol{x}^*$ with a high accuracy.

2.2 Multinomial Logistic Regression

Multinomial logistic regression is a generalization of binary logistic regression. It belongs to the class of *generalized linear models* (GLMs) and is widely used in multi-class classification problems. The model assumes a linear relationship between input variables and the log-odds of each outcome class relative to a reference class [7].

Let $\boldsymbol{x} \in \mathbb{R}^p$ denote a vector of p input features. For each class $k \leq K$, the model defines a *linear predictor*

$$\eta_k = \beta_{0k} + \boldsymbol{\beta}_k^T \boldsymbol{x}, \tag{1}$$

where $\boldsymbol{\beta}_k^T \in \mathbb{R}^p$ is the coefficient vector for class k and $\beta_{0k} \in \mathbb{R}$ is the corresponding intercept. The predicted class probabilities are obtained by applying the *softmax function* to the linear predictors

$$P(Y = k \mid \boldsymbol{x}) = \frac{e^{\eta_k}}{\sum_{i=1}^K e^{\eta_i}}, \quad k = 1, \ldots, K. \tag{2}$$

This formulation ensures that all class probabilities are non-negative and sum to one [9].

To avoid over parametrization, one class is assigned as the *reference class*. Most commonly, the last class K is set as the reference class. This is done by setting its coefficients and intercept to zero (i.e., $\boldsymbol{\beta}_K = \mathbf{0}$ and $\beta_{0K} = 0$). The remaining parameters are estimated by the MLE (abbr. for maximum likelihood estimation). Given a training set $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=i}^n$, the log-likelihood is given by

$$\ell(\{\beta_{0k}, \boldsymbol{\beta}_k\}) = \sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{I}(y_i = k) \ \log P(Y = k \mid \boldsymbol{x}_i), \tag{3}$$

where $\mathbb{I}(y_i = k)$ is the indicator function. Maximizing the log-likelihood is equivalent to minimizing the negative log-likelihood, which is often referred to as the *loss function*

$$\mathcal{L}(\{\beta_{0k}, \boldsymbol{\beta}_k\}) = -\sum_{i=1}^{n} \sum_{k=1}^{K} \mathbb{I}(y_i = k) \ \log P(Y = k \mid \boldsymbol{x}_i). \tag{4}$$

Since there is no closed-form solution for maximizing the likelihood, numerical optimization methods such as iteratively reweighted least squares (IRLS) or coordinate descent are typically used, depending on the implementation [7].

Once the model is trained, predictions is typically done using the decision rule

$$\hat{y}(\boldsymbol{x}^*) = \arg\max_k P(Y = k \mid \boldsymbol{x}^*),$$

which selects the class with the highest predicted probability [9].

Multinomial logistic regression serves as a baseline model in this thesis. Its interpretability and simplicity makes it attractive for prediction, though it may require extensions such as regularization to perform well in practice. The next section introduces LASSO regularization, which can be applied to mitigate overfitting and improve generalization.

2.3 Regularization and the LASSO

Regularization is a technique used to improve the generalization performance of models by preventing overfitting the training data. In particular, LASSO (abbreviated Least Absolute Shrinkage and Selection Operator) introduces a penalty that encourages sparsity in the model coefficients. This leads to simpler models and automatic variable selection [7].

In contrast to the formulation used in Section 2.2, where K is chosen as a reference class, the `glmnet` implementation of multinomial logistic regression excludes this constraint. Instead, the method estimates the complete set of

parameters for each class $k = 1, \dots K$. This means that the number of parameters across all K classes are $K(p+1)$ [5].

Let $\ell(\{\beta_{0k}, \boldsymbol{\beta}_k\})$ denote the log-likelihood defined in Equation 3. Then, we maximize penalized log-likelihood as

$$\max_{\{\beta_{0k}, \boldsymbol{\beta}_k\}_{k=1}^K \in \mathbb{R}^{K(p+1)}} \left[\frac{1}{n} \ell(\{\beta_{0k}, \boldsymbol{\beta}_k\}) - \lambda \sum_{k=1}^{K} \sum_{j=1}^{p} |\beta_{jk}| \right], \tag{5}$$

where $\lambda \geq 0$ is a regularization parameter and β_{jk}, the jth coordinate of $\boldsymbol{\beta}_k$, denotes the coefficient for feature j in class k.

Since this parametrization models all K classes, it is not identifiable without constraints. This is due to the invariance of the softmax function to additive shifts across all η_k (i.e., $\{\beta_{0k}, \boldsymbol{\beta}_k\}_1^K$, $\{\beta_{0k} - c_0, \boldsymbol{\beta}_k - c\}_1^K$ give identical probabilities $P(Y = k \mid \boldsymbol{x})$). However, as shown in [5], explicit constraints are not required for fitting the model and the penalized likelihood still has a unique solution.

2.4 Decision Trees

Decision trees are rule-based prediction models used for both classification and regression problems. For classification, the goal is to partition the feature space into disjoint regions, each assigned to one of the possible class labels. The model takes the form of a tree structure, where each internal node represents a decision based on the value of a single feature, and each terminal node (or leaf) represents a predicted class.

More formally, let $\boldsymbol{x} \in \mathbb{R}^p$ denote the input feature vector. A decision tree partitions the input space $\mathbb{R}^p$ into M disjoint regions $\{R_1, \dots, R_M\}$ such that

$$\mathbb{R}^p = \bigcup_{m=1}^{M} R_m, \quad R_i \cap R_j = \emptyset \text{ for } i \neq j.$$

For each region R_m, the tree assigns a class label $c_m \in \{1, \dots, K\}$. The classifier is given by

$$f(\boldsymbol{x}) = \sum_{m=1}^{M} c_m \mathbb{I}(\boldsymbol{x} \in R_m),$$

where $\mathbb{I}(\cdot)$ is the indicator function. To assign a class label to region R_m, the algorithm considers the subset of training observations that fall into this region. Let $\mathcal{T}_m = \{(\boldsymbol{x}_i, y_i) \in \mathcal{T} : \boldsymbol{x}_i \in R_m\}$ be the subset of training data in the region R_m. For each class $k \in Y$, define the proportion of observations in $\mathcal{T}_m$ that belong to class k as

$$\hat{p}_{mk} = \frac{1}{|\mathcal{T}_m|} \sum_{(\boldsymbol{x}_i, y_i) \in \mathcal{T}_m} \mathbb{I}(y_i = k).$$

Then the predicted class for region R_m is given by

$$\hat{y}_m = \arg\max_{k \in Y} \hat{p}_{mk}.$$

That is, the model assigns the class that appears most frequently among the training observations in that region. This is often referred to as *majority vote*. Additionally, the estimated probabilities $\hat{p}_{mk}$ are often retained for each class k, which enables probabilistic predictions. These are especially useful when combining trees in ensemble methods, such as soft voting in Random Forests.

2.4.1 Recursive Binary Splitting

The structure of a decision tree is built using a greedy algorithm called *recursive binary splitting*. At each internal node of the tree, the algorithm selects a feature x_j and a threshold θ to divide the observations into two disjoint regions

$$R_{\text{left}} = \{\boldsymbol{x} : x_j \leq \theta\}, \quad R_{\text{right}} = \{\boldsymbol{x} : x_j > \theta\}.$$

The objective is to choose the split that results in the purest possible child nodes. That is, regions where observations tend to belong to the same class. This is measured using an impurity criterion, with three common choices being the *misclassification error*, *Gini index* and *entropy*.

Let $\hat{p}_{mk}$ denote the proportion of observations belonging to class k in region R_m. Then, for region R_m, the three impurity measures are defined as

$$\begin{aligned}
\text{Misclassification}(R_{\text{m}}) &= 1 - \max_k \hat{p}_{mk}, \\
\text{Gini}(R_m) &= \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}), \\
\text{Entropy}(R_m) &= -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}.
\end{aligned}$$

For a proposed split that produces the child nodes R_{left} and R_{right}, the total impurity computed as the weighted sum of the impurity in each child

$$I_{\text{split}} = \frac{n_{\text{left}}}{n} I(R_{\text{left}}) + \frac{n_{\text{right}}}{n} I(R_{\text{right}}),$$

where $n = n_{\text{left}} + n_{\text{right}}$ is the number of observations in the parent node, and $I(\cdot)$ denotes the impurity measure. The algorithm searches over all features and split points, selecting the one that minimizes I_{split}. This process is repeated recursively to each resulting child node, growing the tree top-down

until a stopping criterion is met. Some common stopping rules include a minimum number of observations in a node, maximum tree depth, or a threshold on impurity reduction.

One issue with the misclassification rate is that it is less sensitive to changes in the class proportions within a node. In contrast, the Gini index and entropy are more responsive to such changes and typically yield better splits.

As mentioned in the beginning, this algorithm is greedy, meaning it selects the best split at each step without considering the impact of future splits. As such, the final tree is not guaranteed to be globally optimal. In other words, the structure that minimizes total misclassification or impurity over the entire tree is not necessarily achieved. Moreover, fully grown trees tend to overfit and are highly sensitive to small changes in training data. These limitations motivate the use of ensemble methods like *bagging* and *Random Forests*, which will be presented later.

2.5 Ensemble Tree Methods

Ensemble methods refer to the design where multiple machine learning models, known as *base models*, are constructed and combined to produce a final prediction. One common approach is bootstrap aggregation, which is presented in the following.

2.5.1 Bagging

Bootstrap aggregating, also known as *bagging*, is one ensemble method considered for models prone to high variance, such as decision trees. Bagging aims to reduce this variance without increasing its bias. The method works by constructing an ensemble of similar base models, each trained on slightly varied versions of the original training set.

Bagging begins by generating multiple bootstrap samples from the training dataset $\mathcal{T}$, where each sample has the same size as $\mathcal{T}$ and is drawn with replacement. As a result, it is likely that each sample contains duplicates and excludes some original observations. For each bootstrap sample, a base model is trained independently. Due to resampling, each model is exposed to a different subset of the training data, introducing variability into the learned models. For prediction, the final output is determined by aggregating the predictions from all base models. In the classification setting, this aggregation is often performed using either *hard voting* (majority vote of class labels) or *soft voting*, where one averages the predicted class probabilities from each model and selects the class with the highest average probability.

We now present how bagging reduces variance. Specifically, how averaging reduces variance in general and how this relates to bagging. Let

$\{Z_b\}_{b=1}^{B}$ be a sequence of identically distributed random variables with mean $\mathbb{E}[Z_b] = \mu$ and variance $\text{Var}(Z_b) = \sigma^2$. Furthermore, let ρ denote the average correlation between any pair of random variables in the sequence:

$$\rho = \frac{\frac{1}{B(B-1)} \sum_{b \neq c} \mathbb{E}[(Z_b - \mu)(Z_c - \mu)]}{\sigma^2}.$$

Then, the mean and variance when averaging these random variables

$$\mathbb{E}\left[\frac{1}{B}\sum_{b=1}^{B} Z_b\right] = \mu, \tag{6}$$

$$\text{Var}\left(\frac{1}{B}\sum_{b=1}^{B} Z_b\right) = \frac{1-\rho}{B}\sigma^2 + \rho\sigma^2. \tag{7}$$

From Equation 6, we see that the mean does not change when averaging. This means that averaging does not increase the bias. Equation 7 shows that the variance is reduced as B increases, assuming the random variables are not perfectly correlated (i.e., $\rho < 1$). Then variance is reduced by averaging. In fact, the variance approaches $\rho\sigma^2$ as $B \to \infty$, and hence bagging leads to variance reduction if $\rho < 1$. That is, the variance has a lower bound which is dependent on ρ.

This theory applies directly to bagging. Suppose we generate B bootstrap samples from our training data $\mathcal{T}$ and train a separate base model, respectively on each. Let $\{\boldsymbol{g}^{(b)}(\boldsymbol{x})\}_{b=1}^{B}$ denote the ensemble of trained models, each outputting a vector of predicted class probabilities for a new input $\boldsymbol{x}^*$. Then the final ensemble prediction using soft voting is given by

$$\boldsymbol{g}_{\text{bag}}(\boldsymbol{x}^*) = \frac{1}{B}\sum_{b=1}^{B} \boldsymbol{g}^{(b)}(\boldsymbol{x}^*). \tag{8}$$

Although each model is trained on a different bootstrap sample, they are all derived from the same dataset $\mathcal{T}$, which means their predictions are identically distributed, but likely correlated. Consequently, the ensemble prediction achieves lower variance compared to a single model.

2.5.2 Random Forests: Reducing Correlation through Feature Randomization

As seen in the previous section, the effectiveness of bagging is limited by the correlation between the base models. When the base learners are highly correlated, the variance reduction achieved through averaging is less substantial. The Random Forest algorithm improves on this by actively reducing correlation between trees.

Random Forests achieve this decorrelation by introducing additional randomness when growing each tree. Specifically, at each node of each tree, the algorithm randomly selects a subset of the features (of size $q \leq p$) and restricts the split to be chosen only from this subset. This feature subsetting is performed independently at every split and for every tree in the ensemble. By combining bagging with randomized feature selection, the algorithm generates more diverse trees that are less likely to make similar errors.

To give an intuitive understanding as to why feature subsetting reduces the correlation between the trees, recall Section 2.4.1. There, we showed that the variance reduction achieved through bagging is limited by the correlation ρ between the base models. If many trees in the ensemble are trained on similar dominant features, they tend to produce similar splits, leading to high correlation. Instead, by forcing each tree to consider only a random subset of features at each split, Random Forests encourage diversity among trees. As a result, even if one feature dominates the data, not all trees will rely on it at every split. This randomness leads to more variation in tree structure and decisions, reducing ρ and improving the effectiveness of averaging.

In practice, the number of features q considered at each split is often set to $q = \lfloor \sqrt{p} \rfloor$ for classification problems. A more systematic way is by using the *out-of-bag* (OOB) error to find an optimal value of q [**?**].

2.5.3 Final Prediction in Random Forests

Once all trees in the Random Forest have been trained, predictions for new observations are made by aggregating the outputs from each tree in the ensemble. In Section 2.5.1, we briefly mentioned the two alternatives:

1. **Hard voting** (majority vote): Each tree predicts a class label for the new observation, and the final prediction is the class that receives the most votes.

2. **Soft voting** (probability averaging): Each tree outputs a probability distribution across all classes, and the final prediction is the class with the highest average probability across all trees.

In this thesis, soft voting will be implemented.

Let $\boldsymbol{g}^{(b)}(\boldsymbol{x}^*)$ denote the predicted class probabilities from tree b for an input $\boldsymbol{x}^*$, and $|\boldsymbol{g}^{(b)}(\boldsymbol{x}^*)| = \sum_{k=1}^{K} [\boldsymbol{g}^{(b)}(\boldsymbol{x}^*)]_k = 1$. Then, the aggregated (soft) prediction from the Random Forest is given by

$$\boldsymbol{g}_{\text{RF}}(\boldsymbol{x}^*) = \frac{1}{B} \sum_{b=1}^{B} \boldsymbol{g}^{(b)}(\boldsymbol{x}^*). \tag{9}$$

The final predicted class is then obtained via

$$\hat{y}(\boldsymbol{x}^*) = \arg\max_k \left[\boldsymbol{g}_{\text{RF}}(\boldsymbol{x}^*)\right]_k . \tag{10}$$

2.6 Dealing with Class Imbalance

Class imbalance refers to a setting where one or more classes are significantly underrepresented in the training data. This can bias a classifier toward predicting the majority class and lead to misleading performance metrics. Lindholm [9] highlight that in such cases, accuracy becomes an unreliable evaluation criterion and recommend using more discriminative metrics such as precision, recall, and the F_1 score.

A widely used technique for handling class imbalance is *inverse class frequency* weighting. This approach adjusts the influence of each observation during training by assigning class-specific weights inversely proportional to their frequencies in the data. Therefore, it reduces the impact of majority class and increases the penalty for misclassification of minority class examples [11].

Formally, if n_k denotes the number of training observations belonging to class $k = 1, \dots K$ and n the total number of observations, the class weight for class k is given by

$$w_k = \frac{n}{K \cdot n_k},$$

[10]. This weighting ensures that each class contributes equally to the loss function during model training. In practice, this technique can improve model robustness in imbalanced settings [11].

2.7 Evaluation Metrics

Confusion Matrix

The confusion matrix summarizes the performance of a classification model by showing the counts of true and predicted class labels. In a multiclass setting with K classes, the confusion matrix takes the form of a $K \times K$ table, where each row corresponds to the predicted class and each column to the actual class. For example, a confusion matrix for a three-class problem labeled A, B and C would correspond to Table 1 The diagonal elements n_{kk} indicate correct predictions, while off-diagonal elements represent misclassifications.

Table 1: Example of a confusion matrix for a three-class classification problem

Predicted \ Actual	A	B	C
A	n_{AA}	n_{AB}	n_{AC}
B	n_{BA}	n_{BB}	n_{BC}
C	n_{CA}	n_{CB}	n_{CC}

Accuracy

Accuracy is the most commonly used evaluation metric and measures the proportion of correctly classified observation

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}}.$$

This corresponds to the sum of the diagonal elements of the confusion matrix divided by the total number of observations. Naturally, this can be extended to class-specific accuracy. In this thesis, we define this as

$$\text{Class Acc}(k) = \frac{\text{Number of correctly predicted class } k}{\text{Total number of true class } k}.$$

This corresponds to the diagonal value in the confusion matrix for class k, divided by the sum of all values in column k.

As mentioned in Section 2.6, relying on accuracy alone can be misleading, especially dealing with class imbalances. Therefore, we present some alternative evaluation metrics that will be used during evaluation.

Precision, Recall and F1 Score

In the multiclass setting, the F1 score for each class is computed using an approach, where the class of interest is treated as the positive class and all others as negative. To illustrate this further recall Table 1, and say we treat A as the the positive class. Then, Table 2 represents the corresponding confusion matrix. TP_A (true positives) denotes the number of observations correctly predicted as class A, FP_A (false positives) refers to observations from classes B or C that were incorrectly predicted as class A, and FN_A (false negatives) represents the number of actual class A observations that were misclassified as either class B or C [6].

Table 2: Illustration of TP, FP, and FN for class A under a one-versus-rest approach.

Predicted \ Actual	A	B	C
A	TP_A	FP_A	FP_A
B	FN_A	–	–
C	FN_A	–	–

Before presenting the F1 score, we define two related evaluation metrics known as: *precision* and *recall*. Precision describes the proportion of predicted observations for a class that are actually correct. That is, it measures how many of the predicted positives are true positives. Recall describes the proportion of actual observations from a class that the classifier correctly

identifies. In other words, precision tells us how many of the predicted positives are true positives, and recall measures how many of the true positives the classifier successfully captures [6].

Following the definitions implemented in scikit-learn [10], the precision and recall for a specific class k is given by:

$$\text{Precision}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FP}_k}, \quad \text{Recall}_k = \frac{\text{TP}_k}{\text{TP}_k + \text{FN}_k}.$$

Then, the corresponding class-specific F1 score is then given by the harmonic mean of precision and recall:

$$\text{F1}_k = \frac{2 \cdot \text{Precision}_k \cdot \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}.$$

To summarize overall performance across all classes, the macro-averaged F1 score is computed as the unweighted mean of the class-specific F1 scores:

$$\text{F1}_{\text{macro}} = \frac{1}{K} \sum_{k=1}^{K} \text{F1}_k.$$

OOB Error Rate

For random forest, model performance can also be assessed using the *out-of-bag (OOB)* error rate. Since each decision tree is trained on a bootstrap sample, about one-third of the training data is left out for each tree. These excluded observations are referred to as *out-of-bag samples*.

The OOB error is computed by predicting the class of each training observation using only the trees that did not include it in their training sample. This results in an unbiased estimate of the generalization error, avoiding the need for a separate validation set [9].

3 Data

The data used in this study consist of match results from the English Premier League (abbr. EPL) for the seasons 2016/2017 to 2024/2025. This was obtained from DataHub (sourced from Football Data UK) [1]. Each season was stored in a separate CSV file, where each row corresponds to a single match and columns represent various match-specific information.

The original data sets included fundamental match information such as match date, teams, full-time results, and other match-specific details. These variables served as the basis for computing the derived features used in the models. It is also worth mentioning that the original data sets included half-time results and their respective match statistics. These were excluded in this thesis, as the objective is to predict outcomes solely on pre-match information.

To prepare the data set for modeling, several pre-processing steps were done. This included converting date strings to proper date formats, and merging all seasonal data sets into a single data frame. Derived features, such as recent team form, goal metrics, and leaderboard standings, were computed based on this merged data set.

3.1 Feature Engineering

To capture relevant information about each team's recent performance and relative strength, multiple features were engineered from the raw match results. These derived features were computed iteratively for each team based on the chronological order of matches within a season. The features can be grouped into the following categories:

- **Performance-based features** (e.g., Form Points, Win/Loss Streaks, Unbeaten Records, Points per Match)
- **Goal-related features** (e.g., Goals per Match, Conceded Goals per Match, Goal Differences)
- **Historical Team Strength** (e.g., Last Season Position, Promoted Status)
- **Relative Strength Features** (i.e., differences between home and away team metrics)

The engineered features were chosen to capture both short-term momentum and long-term team strength. Performance-based metrics are designed to reflect team momentum and form, while goal-based metrics provide insight into a team's offensive and defensive capabilities. Furthermore, differences between competing teams aim to quantify the relative strength in each matchup. Lastly, including previous season standings and promoted status allows the model to incorporate historical team reputation and league experience (see Table 3 for a complete list of included features).

To avoid *data leakage*, each engineered feature only had access to the available information before predicting the match. For example, rolling statistics were computed using data from matches that precede the current fixture. This precaution prevents the model from accidentally accessing future information, which would lead to over-optimistic performance estimates. For further discussion on data leakage and its consequences, see Kaufman et al. [8].

Table 3: Summary of input features. HT = Home Team and AT = Away Team.

Features	Description	Type
HTLBS/ATLBS	Final league position based on previous season (excluding newly promoted teams)	Integer
HT3WS/AT3WS	Has won the previous three games	Binary
HT5WS/AT5WS	Has won the previous five games	Binary
HT3LS/AT3LS	Has lost the previous three games	Binary
HT5LS/AT5LS	Has lost the previous five games	Binary
HT3GU/AT3GU	Unbeaten the previous three games	Binary
HT5GU/AT5GU	Unbeaten the previous five games	Binary
HTP/ATP	Newly promoted team	Binary
LBSD	Difference in LBS between the two playing teams	Integer
HTGPM/ATGPM	Goals scored per match this season	Numeric
GPMD	Difference in GPM between the two playing teams	Numeric
HTCPM/ATCPM	Goals conceded per match this season	Numeric
CPMD	Difference in CPM between the two playing teams	Numeric
HTGDPM/ATGDPM	Difference between GPM and CPM this season	Numeric
GDPMD	Difference in GDPM between the two playing teams	Numeric
HTPPM/ATPPM	Points per Match this season	Numeric
PPMD	Difference in PPM between the two playing teams	Numeric
HTFP/ATFP	Sum of points gained in the previous five matches	Numeric
FPD	Difference in FP between the two playing teams	Numeric

3.2 Data Overview

The final data set consists of match results from nine consecutive EPL seasons, ranging from 2016/2017-2024/2025. Each row corresponds to a single match and includes 34 engineered input features related to the participating teams. Due to the design of some features, the first five matches played by each team were filtered out. This results in a total of 2877 matches. The data set is arranged in descending order with respect to the match date fixture.

The response variable, Full-Time Result (abbr. FTR), is a categorical variable with three classes: home win (`H`), draw `D`, and away win (`A`). Table 4

shows the class distribution of the entire data set. As illustrated, class imbalance is present. Home Win `H` is a majority class, whereas Draw `D` are notably unrepresented.

.

Table 4: Distribution of match outcomes in the final data set.

Outcome	Count	Proportion
Home Win (H)	1308	45.5%
Draw (D)	650	22.6%
Away Win (A)	919	31.9%

4 Methods

This section outlines the implementation and training of the two classifiers discussed in this thesis. To ensure reliable and robust evaluation, each classifier was trained and tested over 10 independently generated stratified train/test splits, each with an 80/20 ratio. This means that for each of the 10 iterations, a new split was drawn from the full data set, while preserving the original class distribution in both training and test sets. This approach helps account for variability in performance due to data partitioning.

It should also be noted that the implementation was done in R. Furthermore, all non-categorical features were centered and scaled prior to training and evaluation of the two classifiers.

LASSO-Penalized Multinomial Logistic Regression

The LASSO-penalized multinomial logistic regression classifier was implemented using the `glmnet` package [5]. A full set of coefficients was estimated for each class by setting `type.multinomial = "ungrouped"`. Inverse class frequency weights were calculated (as described in Section 2.6). Each observation in the training split were assigned corresponding weight using the argument `weights`.

For each trained classifier, predictions on unseen data were obtained using the `predict()` function from the same package. The penalty parameter λ used was obtained by specifying `s = lambda.min`, which corresponds to the minimum value of λ across the 10-fold cross-validation. Final classification was performed by selecting the class with the highest predicted probability.

Random Forest

Random forest was implemented using the `ranger` package [12]. Each were trained with the default settings of 1000 trees and the Gini index as the

impurity criterion. Furthermore, the argument `probability = TRUE` was passed. This ensures that each tree outputs a probability distribution over classes for each observation. These distributions are then averaged across the ensemble, which aligns with the concept of soft voting described in Section 2.5.3.

Like in the multinomial logistic regression modeling, inverse class frequency weights were computed and applied to each observation. These weights specified in the `case.weights` argument, which adjusts the probability of each observation being selected in the bootstrap samples used to train individual trees. However, it should be noted that this alone does not directly impose penalties during tree growing. There is also the `class.weights` argument that does directly influence the splitting rule. Both the `case.weights` and the `class.weights` argument were initially considered for handling class imbalance. The final implementation presented only uses the former. A table comparing performance metrics can be found in Appendix A.

5 Results

This section presents and compares the predictive performance of the two classifiers in focus. Results are reported as averages across the 10 random train/test splits with an 80/20 ratio. For each split, evaluation metrics such as accuracy, macro F1 score, and class-specific performance were computed to asses both overall and per-class predictive performance.

5.1 Performance and Evaluation Metrics

Multinomial Logistic Regression

Table 5 displays the performance metrics for the multinomial logistic regression classifier. The classifier achieved an average accuracy of 0.480, and a macro-averaged F1 score was 0.447. Furthermore, the classifier performed moderate on the Home Win and Away Win classes, achieving F1 scores of 0.573 and 0.515, respectively. However, performance on the Draw class was noticeably lower, reporting a F1 score of 0.267 and a class-specific accuracy of just 0.252.

Table 5: Performance metrics for the multinomial logistic regression classifier

Metric	Mean	Standard Deviation
Overall Performance		
Accuracy	0.480	0.021
F1 Macro	0.447	0.021
Class-Specific Performance		
Class Acc (Home Win)	0.539	0.025
Class Acc (Draw)	0.252	0.037
Class Acc (Away Win)	0.556	0.026
F1 (Home Win)	0.573	0.023
F1 (Draw)	0.267	0.038
F1 (Away Win)	0.515	0.002

Random Forest

Table 6 shows the corresponding performance for the random forest classifier trained using inverse class frequency weights via `case.weights`. The classifier reports similar results in both overall and class-specific performance metrics. As with the logistic regression classifier, the random forest performed better on the Home Win and Away Win classes, while performance on the Draw class remained the weakest.

Table 6: Performance metrics for the Random Forest classifier

Metric	Mean	Standard Deviation
Overall Performance		
Accuracy	0.497	0.013
F1 Macro	0.446	0.012
OOB Error	0.385	0.002
Class-Specific Performance		
Class Acc (Home Win)	0.610	0.026
Class Acc (Draw)	0.184	0.029
Class Acc (Away Win)	0.552	0.042
F1 (Home Win)	0.599	0.014
F1 (Draw)	0.219	0.027
F1 (Away Win)	0.519	0.033

Stability and Performance Comparison

To assess the robustness of the two classifiers, the standard deviation of each metric were calculated. The relatively low standard deviations reported in Tables 5 and 6 suggest that both classifiers performed consistently across different data splits. This implies that their predictive behavior is not sensitive to specific samples. Overall, while the random forest classifier displayed slight improvements in overall accuracy and log-loss, both classifiers demonstrated similar performance patterns. In particular, both showed relatively strong performance on the Home Win and Away win classes, but consistently struggled to correctly classify Draw outcomes, as reflected in both F1 the scores and the class-specific accuracies.

5.2 Comparison of Predicted Probability Distributions

To examine how the two classifiers assign probabilities to different outcome classes density plots along with summary statistics, and calibration plots are presented.

The density plots in Figure 1 illustrates how the predicted probabilities are distributed for each outcome class, regardless of the true label. Each curve represents the distribution of predicted probabilities assigned to a particular class across all test folds. A supplement summary is displayed in Table 7.

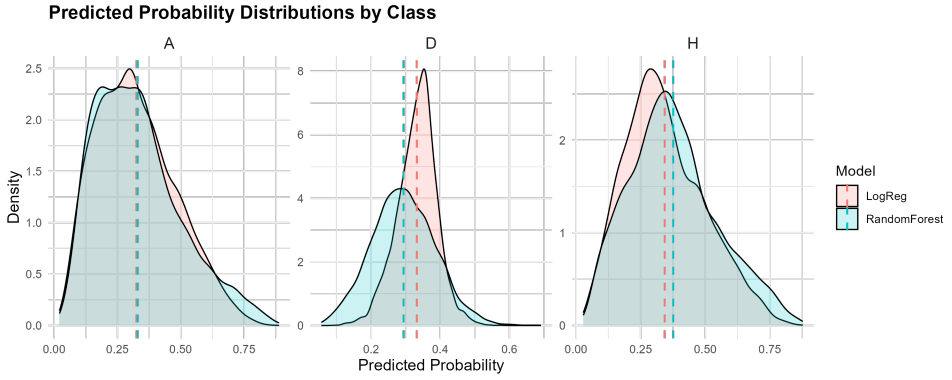

Figure 1: Density plots of predicted probabilities assigned to each class across all test folds

Table 7: Summary statistics of predicted probabilities assigned to each outcome class, regardless of the true label. The table reports the mean predicted probability, the standard deviation (abbr. SD), and the 10th and 90th percentiles across all test folds.

Class	Classifier	Mean	10th %	90th %	SD
A	LogReg	0.325	0.131	0.546	0.156
A	RandomForest	0.330	0.132	0.571	0.169
D	LogReg	0.332	0.256	0.399	0.058
D	RandomForest	0.294	0.178	0.411	0.091
H	LogReg	0.343	0.154	0.564	0.154
H	RandomForest	0.376	0.160	0.621	0.170

For the Draw class, both classifiers assign relatively low probabilities overall, with most values concentrated below 0.5. This is reflected in the reported lower means and narrow percentile ranges. In contrast, predictions for Home Win and Away Win are more broadly distributed. Both classifiers shows noticeable right tails for these classes, consistent with the higher standard deviations and broader percentile ranges. These results indicate that the classifiers tend to assign higher probabilities more frequently to the more common Home and Away outcomes, while being conservative towards the minority class.

Figure 2 shows calibration plots for each class and classifier, illustrating the relationship between predicted probabilities and observed frequencies, aggregated over all test folds. Predicted probabilities were binned into ten intervals with equal widths, where for each bin, the observed frequency equals the proportion of times the predicted class matched the true class. Perfect calibration is indicated by the dashed diagonal line.

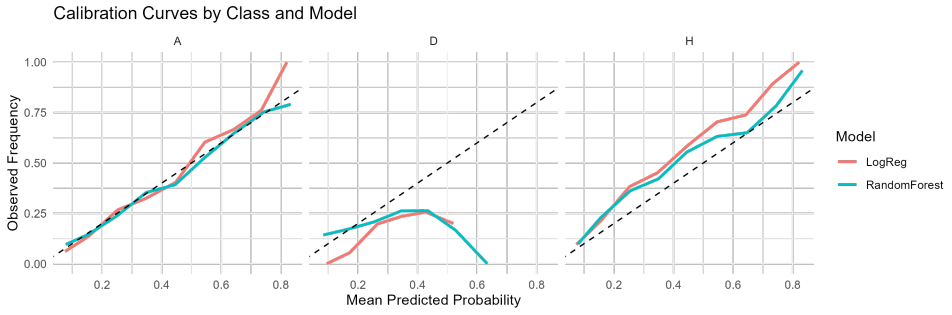

Figure 2: Calibration curves for each outcome class and classifier

For the Away Win class, random forest displays near-perfect calibration,

closely following the diagonal across the entire probability range. Logistic regression tends to under-predict the probability of Away Wins at higher predicted probability levels, as its calibration curve lies above the diagonal for predicted probabilities ≤ 0.5. This indicates that the predicted probabilities for Away Wins are systematically lower than the observed frequencies. For the Home Win class, random forest comes closest to the diagonal at higher predicted probability levels, though both classifiers tend to under-predict Home Win outcomes across the full range.

Both classifiers display poor calibration for the Draw class. Logistic regression consistently over-predicts Draw outcomes, with its calibration curve lying below the diagonal across all probability bins. Random forest shows slight improved calibration at lower probability levels but displays the same overall trend of over-prediction. These patterns are consistent with earlier findings based on the performance metrics and probability distributions, confirming that the Draw class remains the most challenging one to classify correctly.

5.3 Confusion Matrix Analysis

To further analyze class-specific prediction patterns, Figure 3 presents the normalized confusion matrices for both classifiers, averaged across all test folds. Each column represents a true class label, and cell values equal the proportion of predictions assigned to each each predicted class, normalized by the number of instances in that true class. Rows correspond to predicted classes.

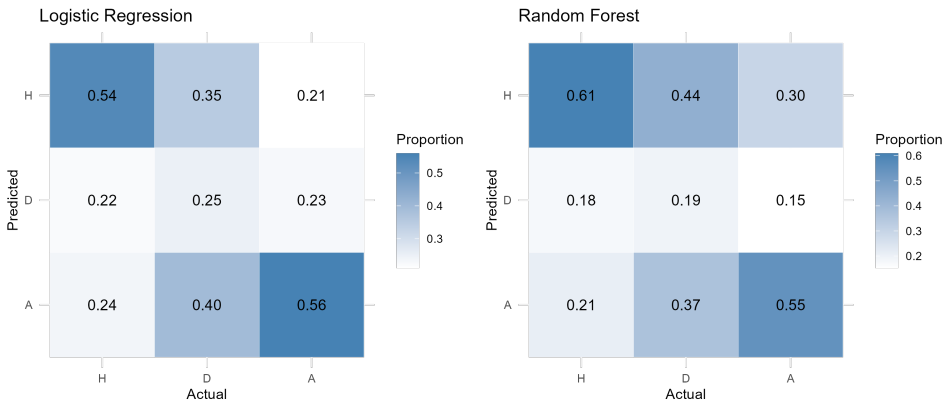

Figure 3: Confusion matrix heat maps averaged across all folds. Each cell represents the proportion of predictions for each true class label.

Both classifiers display high misclassification rates when the true class label is Draws, with the random forest struggling the most. The distributed proportions in the second column shows that logistic regression tends to misclassify Draw with Away Win slightly more than with Home Win. Conversely, the random forest algorithm tends to misclassify with Home Win slightly more often when the true class label is Draw.

When the true class is either Home Win or Away Win, both classifiers demonstrate comparable results when predicting the correct class label. However, the distribution of the misclassification proportions shows that random forest tends to misclassify Away Wins more often with Home Wins, suggesting minor confusion when distinguishing the two. On the contrary, the misclassification proportions obtained from the logistic regression classifier are relatively evenly distributed for each true class label.

5.4 Feature Importance

To understand which input features contributed most to the predictions, a bar plot was made for both classifiers. Figure 4 displays the top 10 features for each classifier, where the importance scores were standardized and averaged across the 10 stratified train/test folds. Shared features between the two classifiers are highlighted in color, while model-specific features are shown in gray.

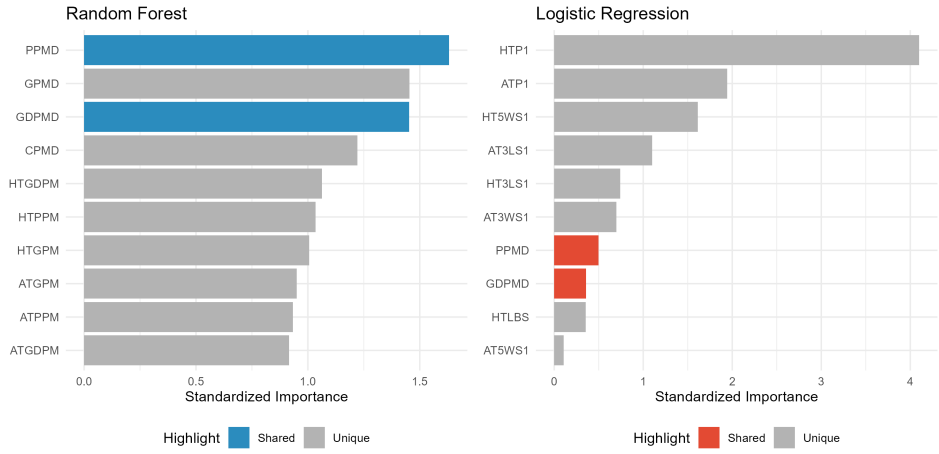

Figure 4: Bar charts illustrating the top 10 features for each classifier. Highlighted bars means that both classifiers included these features.

Both classifiers consistently ranked the feature GPMD (Goals per Match Difference) and GDPMD (Goal Difference per Match Difference) among the most important. This agreement suggests that relative goal-based features

between teams is a strong predictor of match outcomes. However, the classifiers also reveal key differences. The classifier obtained from the logistic regression emphasized features tied to prior season rankings and recent form (e.g., HTP, HT5WS). This suggests that the classifier relied more on intuitive indicators of team strength. In contrast, the random forest prioritized on goal-based and defensive features (e.g., CPMD, ATGPM), suggesting that the random forest may have learned more complex patterns involving team performance.

6 Discussion

6.1 Most Challenging Prediction Task

The results in this comparison study clearly outline the difficulty both methods had in correctly classifying draw outcomes. Across all evaluation metrics presented, draws were associated with the weakest performance.

Random forest assigned the Draw label less frequently than logistic regression, as seen in Figure 3. This tendency is further supported by the predicted probability distributions and summary statistics presented in Section 5.2. Although the random forest classifier showed larger variation in predicted probabilities for Draw outcomes (i.e., wider percentile ranges), compared to the logistic regression classifier, it generally assigned lower probabilities to draws overall. This is reflected in its lower mean and lower minimum percentile for the Draw class. Since soft voting only selects the class with the highest predicted probability, these lower values likely contributed to the classifier assigning Draw as the predicted class label less frequently than logistic regression.

The confusion matrices of Figure 3 further illustrate the challenges associated with predicting Draws. Both classifiers displayed high proportions of misclassification when the true class was Draw, with random forest performing slightly worse. Logistic regression distributed its errors more evenly, with a minor tendency to assigning Away Win. In contrast, random forest had a tendency to misclassify Draw outcomes more often as Home Wins, showing its bias toward the majority class.

This predictive difficulty may in part be explained by the underlying feature distributions. The top-ranking features presented in Section 5.4 were visualized in bar plots and box plots. These can be found in Appendix B.

The bar plots reveal that the categorical features do not provide clear separation for Draw outcomes. Only minor shifts are observed between the levels, and the proportion of Draws remains consistently low. Moreover, the box plots reveal that the numeric features associated with Draw outcomes tend to display lower variability compared to those associated with Home and Away outcomes. There is also the issue where the Home and Away

classes clearly overlap the Draw class. In other words, the numeric features also suffer from lack of separation for Draw outcomes.

In summary, the visualization of feature distributions and predicted probabilities offers an insight as to why both methods struggle in classifying Draws. The lack of separation in the top-ranking features may explain the predicted probability distribution and the poor calibration for the Draw class.

6.2 Method Comparison and Evaluation

The two methods displayed comparable predictive performance based on the evaluation metrics presented. Overall accuracy and macro F1 scores were similar, and both classifiers demonstrated moderate class-specific performance for the Home and Away Win classes. However, neither method performed well on the Draw class. Although, compared to random forest, the logistic regression classifier displayed better results in terms of correctly classifying Draw.

While the overall performance was similar, both classifiers showed some differences in behavior. Random forest displayed a tendency to favor Home Win when the true class label was either Draw or Home Win. In contrast, logistic regression displayed a slight tendency to misclassify Draw more often as Home Win. However, when the true class label was Away Win, the proportions of misclassifications were evenly distributed. In terms of probability estimates, the calibration plots showed that the random forest was better aligned with observed outcomes for the Home and Away classes.

Finally, both classifiers demonstrated stable performance across the 10 stratified train/test. The standard deviations of accuracy, macro F1, and class-specific metrics were consistently low, and the predictive behavior of the two classifiers was not sensitive to data partitioning. This stability reflects the robustness of both methods, further suggesting that the limitations lie in the data.

6.3 Reflection and Looking Forward

Feature engineering seems to play a major role for this particular classification problem, which is also highlighted in Choi [3]. As discussed earlier, the current feature set lacked clear separability from other outcomes. One possible improvement could be to include contextual or external data, such as team ratings, match importance, or even betting odds. Other probabilistic metrics such as xG (expected goals) could possibly improve performance if included.

Implementing inverse class frequency weighting helped to boost the predictive performance for the Draw class (see Table 9). However, exploring different alternatives in dealing with class imbalance would be preferable.

Finally, an interesting take would be to compare whether the two classifiers would perform better given half-time results.

7 Conclusion

The aim of this thesis was to compare the performance of classifiers obtained from multinomial logistic regression and random forest for predicting football match outcomes based on the English Premier League. The results showed that the two classifiers perform similarly overall, achieving comparable accuracy and macro F1 scores. Both methods were relatively effective in classifying Home and Away Wins, but consistently struggled to correctly predict draws. Analysis of predicted probabilities, calibration, and feature distributions suggested that the challenges lie more in the input data, rather than in the performance of the two classifiers.

References

[1] English premier league dataset. https://datahub.io/core/english-premier-league, 2024. Accessed: January 2024. Dataset no longer publicly available.

[2] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 10 2001.

[3] Bing Choi, Lee-Kien Foo, and Sook-Ling Chua. Predicting football match outcomes with machine learning approaches. *MENDEL*, 29:229–236, 12 2023.

[4] Paolo Cintia, Fosca Giannotti, Luca Pappalardo, Dino Pedreschi, and Marco Malvaldi. The harsh rule of the goals: Data-driven performance indicators for football teams. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10, 2015.

[5] Jerome H. Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1â22, 2010.

[6] Margherita Grandini, Enrico Bagli, and Giorgio Visani. Metrics for multi-class classification: an overview, 2020.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Vol. 12)*. Springer, New York, 2nd edition, 2017.

[8] Shachar Kaufman, Saharon Rosset, and Claudia Perlich. Leakage in data mining: Formulation, detection, and avoidance. volume 6, pages 556–563, 01 2011.

[9] Andreas Lindholm, Niklas Wahlström, Fredrik Lindsten, and Thomas B. Schön. *Machine Learning - A First Course for Engineers and Scientists*. Cambridge University Press, 2022.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[11] Philipp Thölke, Yorguin-Jose Mantilla-Ramos, Hamza Abdelhedi, Charlotte Maschke, Arthur Dehgan, Yann Harel, Anirudha Kemtur, Loubna Mekki Berrada, Myriam Sahraoui, Tammy Young, Antoine Bellemare Pépin, Clara El Khantour, Mathieu Landry, Annalisa Pascarella, Vanessa Hadid, Etienne Combrisson, Jordan O' Byrne, and Karim Jerbi. Class imbalance should not throw you off balance: Choosing the right classifiers and performance metrics for brain decoding with imbalanced data. *NeuroImage*, 277:120253, 2023.

[12] Marvin N. Wright and Andreas Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77(1):1–17, 2017.

A Appendix

A.1 Performance Comparison in Random Forest

Table 8: Performance comparison of Random Forest using only `case.weights` vs. both `case.weights` and `class.weights`. Results are averaged across 10 stratified train/test splits.

Metric	**Only** `case.weights`	**Both Weights**	**Difference**
Accuracy	0.497 (0.013)	0.493 (0.014)	-0.06
OOB Error	0.385 (0.002)	0.385 (0.003)	0.00
F1 (Home)	0.599 (0.014)	0.592 (0.013)	-0.070
F1 (Draw)	0.219 (0.027)	0.217 (0.032)	-0.002
F1 (Away)	0.519 (0.026)	0.523 (0.033)	+0.004
F1 Macro	0.443 (0.016)	0.444 (0.015)	+0.001
Class Acc (Home)	0.610 (0.026)	0.595 (0.023)	-0.015
Class Acc (Draw)	0.184 (0.029)	0.197 (0.034)	+0.013
Class Acc (Away)	0.552 (0.042)	0.555 (0.042)	+0.003

A.2 Performance Comparison Using Weights Versus No Weights

Table 9: Comparison of predictive performance with and without inverse class frequency weighting, highlighting the impact in Draw class. Results are averaged across 10 stratified test splits. NA is reported due to the classifier not predicting Draw class.

Metric	**Logistic Regression**		**Random Forest**	
	With Weights	**No Weights**	**With Weights**	**No Weights**
Accuracy	0.480 (0.021)	0.537 (0.013)	0.497 (0.013)	0.523 (0.016)
F1 Macro	0.447 (0.021)	0.566 (0.064)	0.446 (0.012)	0.400 (0.0161)
F1 (Draw)	0.267 (0.038)	NA	0.219 (0.027)	0.049 (0.025)
Class Acc (Draw)	0.252 (0.037)	NA	0.184 (0.029)	0.029 (0.037)

B Appendix

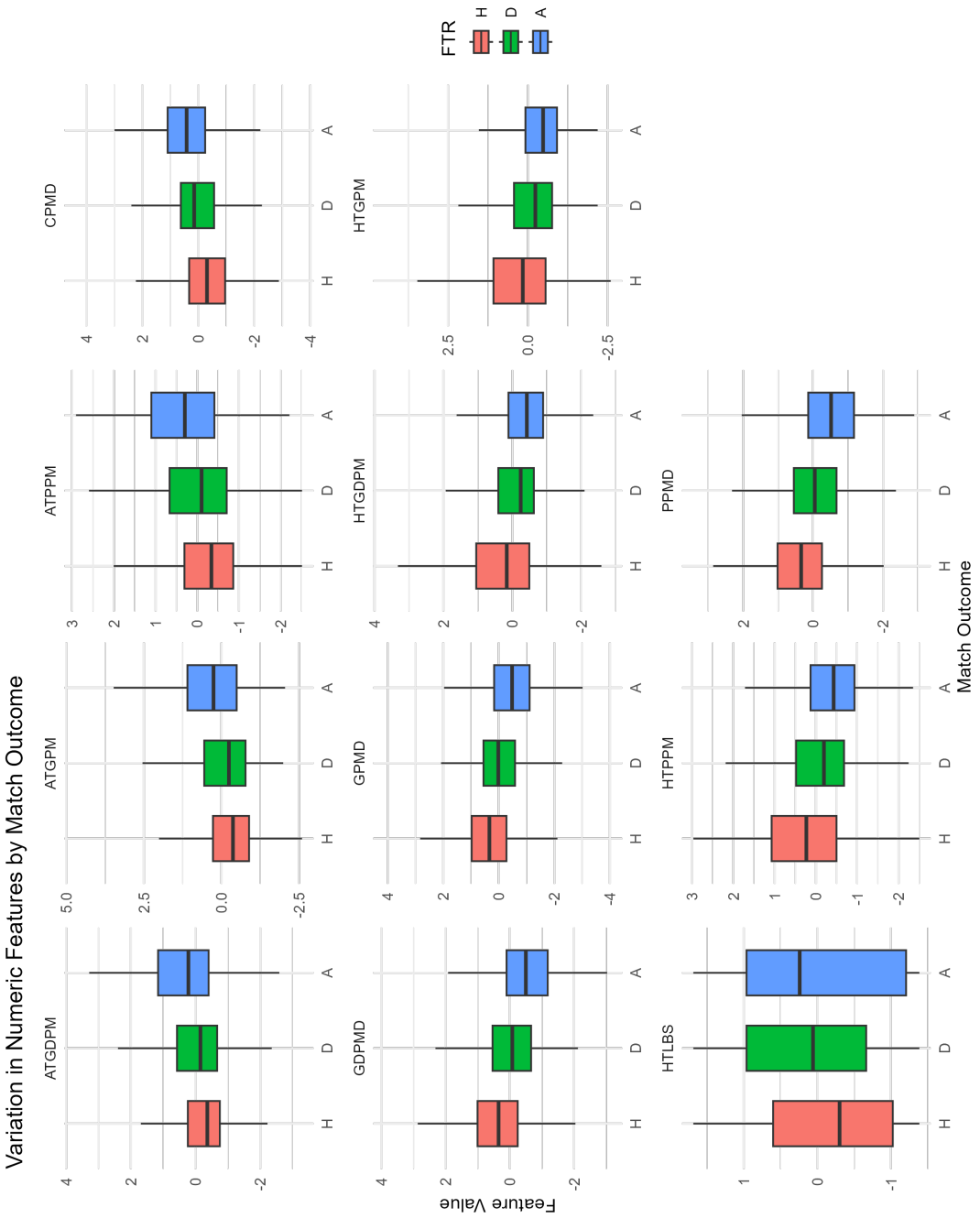

Figure 5: Box plots of scaled numeric feature distributions by match outcomes

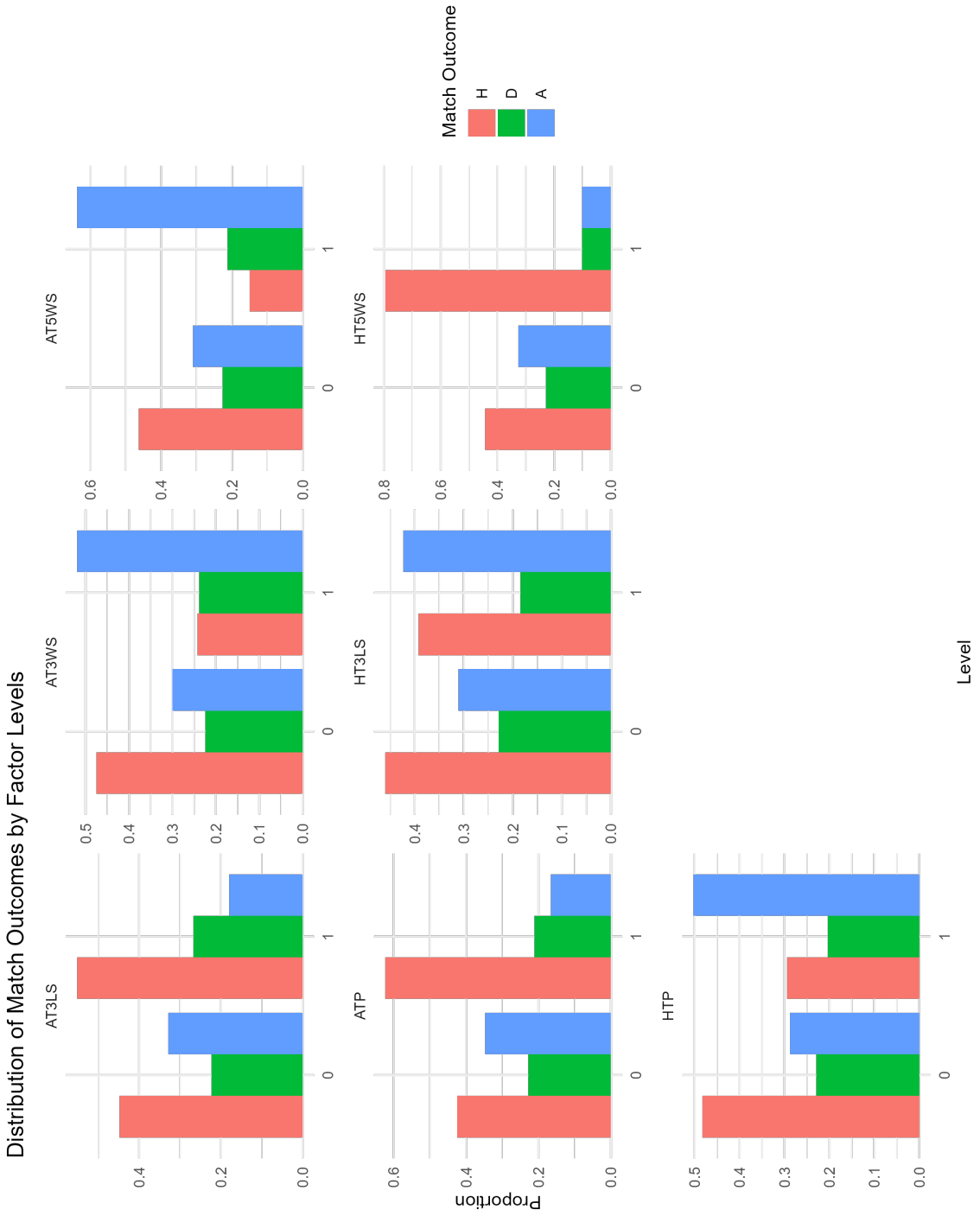

Figure 6: Bar plots of match outcome proportions by factor level for selected dummy variables