Machine Learning for Social Data Science

Problem Set 1

Machine Learning the Identities of Labour's Electorate: Random Forest (Bagged Decision Trees) and XGBoost (Boosted Decision Trees) on BES Wave 30 (1,500 words)

Introduction

Electoral politics in the UK has undergone a profound change in recent decades, nowhere more evident than in the dramatic shift from Labour's worst defeat in 2019 to securing a 174-seat majority on a vote share increase of just 1.7% (Bunting, 2025, p. 14). This conspicuous transformation raises an important question: which social identities underpinned Labour's 2024 electoral coalition? Understanding the identity basis of this support matters not only for interpreting the results, but also for assessing whether traditional class alignments still structure vote choice or whether newer identity cleavages now play a more decisive role. Historically, scholarly accounts of British voting behaviour emphasised social class as the primary structuring force of party choice (Evans, 2000). Prosser et al. find that class-dominated elections from the mass enfranchisement of 1918 through the post-war period, before declining from the 1990s onwards (2024). In the wake of the decline of class-based voting, research has highlighted the growing salience of education, religion, ethnicity, gender, age, and other social identity factors (Johnston, Jones and Manley, 2018; Kolpinskaya and Fox, 2024; Simon, Jennings and Durrant, 2024; Bunting *et al.*, 2025). Such identities may interact in complex and non-linear ways that traditional regression approaches may struggle to capture.

To address this, this essay applies ensemble learning algorithms to the *British Election Study* (*BES*) (Fieldhouse et al., no date) data to predict Labour vote choice using identity variables. The structure of this essay is as follows: a data and methods section, followed by findings, and finally, a conclusion.

Data and Methods

This essay applies two ensemble learning techniques, Random Forest and XGBoost, to model Labour vote choice. Both algorithms are based on decision trees, but they combine them in different ways to improve predictive performance. Random Forest constructs bagged decision trees by training multiple decision trees on bootstrapped samples and averaging their predictions, reducing variance and limiting overfitting through independent trees (Biau and Scornet, 2016). In contrast, XGBoost builds boosted decision trees, sequentially adding new trees that focus on correcting the errors of previous, 'weaker' trees (Sahin, 2020). By emphasising reduced bias, boosting iteratively improves predictive accuracy, often outperforming other approaches on complex datasets. Comparing these methods allows for a methodological assessment of whether sequential boosting or parallel bagging is more effective for predicting vote choice based on social identity factors.

The data is drawn from Wave 28 of the BES internet panel. The BES is a nationally representative panel survey which is widely used in political behaviour research due to its extensive political attitude measures, detailed socio-demographic data collection, and its strong sampling strategy. Wave 28 is selected because it captures 30,342 respondents immediately prior to the 2024 General Election, providing timely measures of vote intention and social identity factors directly relevant to the election outcome.

Together, the data and methods used by this essay address two research questions:

1. Which social identities most strongly predict a respondent voting Labour in the 2024 General Election?
2. Which ensemble approach, bagged or boosted decision trees, achieves superior predictive performance?

The unit of analysis and observation is the individual respondent. The dependent variable and target is Labour voter, a binary indicator of whether the respondent reported voting Labour. Independent variables and features capture social identities: age, gender, ethnicity, religion, sexuality, disability status, education level, subjective class, and region.

Model performance is evaluated using precision, recall, F1 score, and area under the precision-recall curve (PR-AUC). Recall measures the proportion of actual Labour voters correctly identified, while precision captures the proportion of predicted Labour voters who truly voted Labour. The F1 score balances precision and recall, providing a single harmonic mean measure of classification performance. PR-AUC is included because suspected class imbalance makes it more informative than receiver operating characteristic area under the curve, as it focuses specifically on performance in identifying the positive (Labour-voting) class (Cook and Ramadas, 2020).

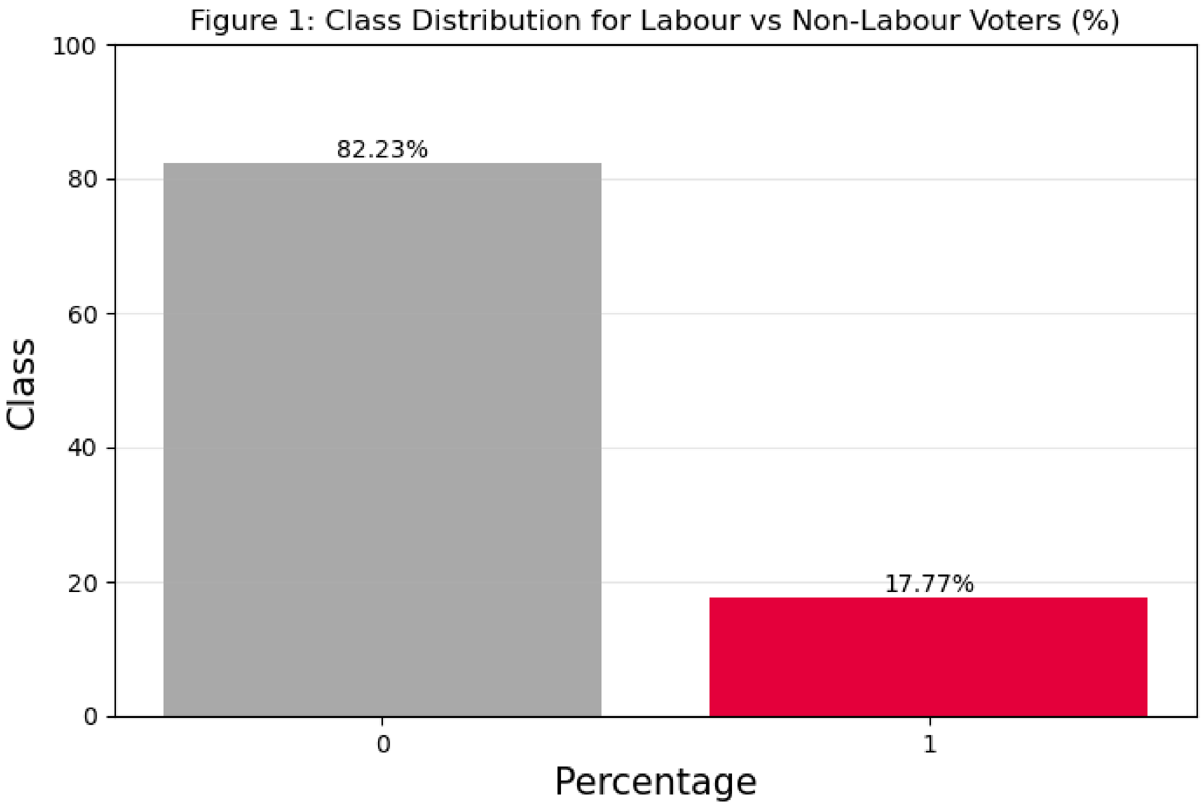
Internal validity is strengthened through splitting the data into training and testing sets and cross-validation, which minimises overfitting and enhances confidence in the models' predictive accuracy. External validity is supported by the BES's representative sample, although generalisation is limited to the British electorate in the period immediately preceding the 2024 General Election.

The ethical considerations for this research are minimal as it relies on anonymised secondary data. To ensure research integrity, code snippets are included, and a GitHub repository link is provided (Appendix A). The dataset was cleaned by removing unnecessary columns, renaming variables for clarity, and ensuring all variables are in a continuous or categorical format to utilise XGBoost's 'enable_categorical' argument.

Findings

The findings section proceeds in three stages: First, it conducts five XGBoost models, each incorporating incremental improvements, before comparing their performance and examining the feature importance of the strongest model. Second, it replicates the same five model specifications using Random Forest for direct comparison. Finally, it evaluates the best-performing model from each specification through cross-validation to ensure the consistency of the results across different train-test splits.

To begin with the first phase, the XGBoost models are introduced sequentially, with each specification designed to address the limitations identified in the previous iteration. Model 1 establishes a baseline using a binary logistic regression objective with log-loss evaluation and an 80/20 train-test split. However, both descriptive inspection of the data and the relatively weak performance of the baseline model indicated a potential class imbalance problem. Figure 1 confirms this imbalance with 82.23% of respondents being non-Labour voters compared to 17.77% Labour voters, a ratio of approximately 1 to 4.6.



Model 2 therefore implements a stratified train-test split to preserve class proportions across both splits. Model 3 further addresses imbalance by introducing class weights, penalising misclassification of Labour voters more heavily. Model 4 adjusts the probability threshold for classification; iterative testing demonstrated that a 0.15 cut-off improved the model results. Finally, Model 5 applies Bayesian optimisation, tuning maximum tree depth, learning rate, number of estimators, sub-sample ratio, column sampling rate, minimum child weight, and the gamma regularisation parameter to identify the optimal configuration.

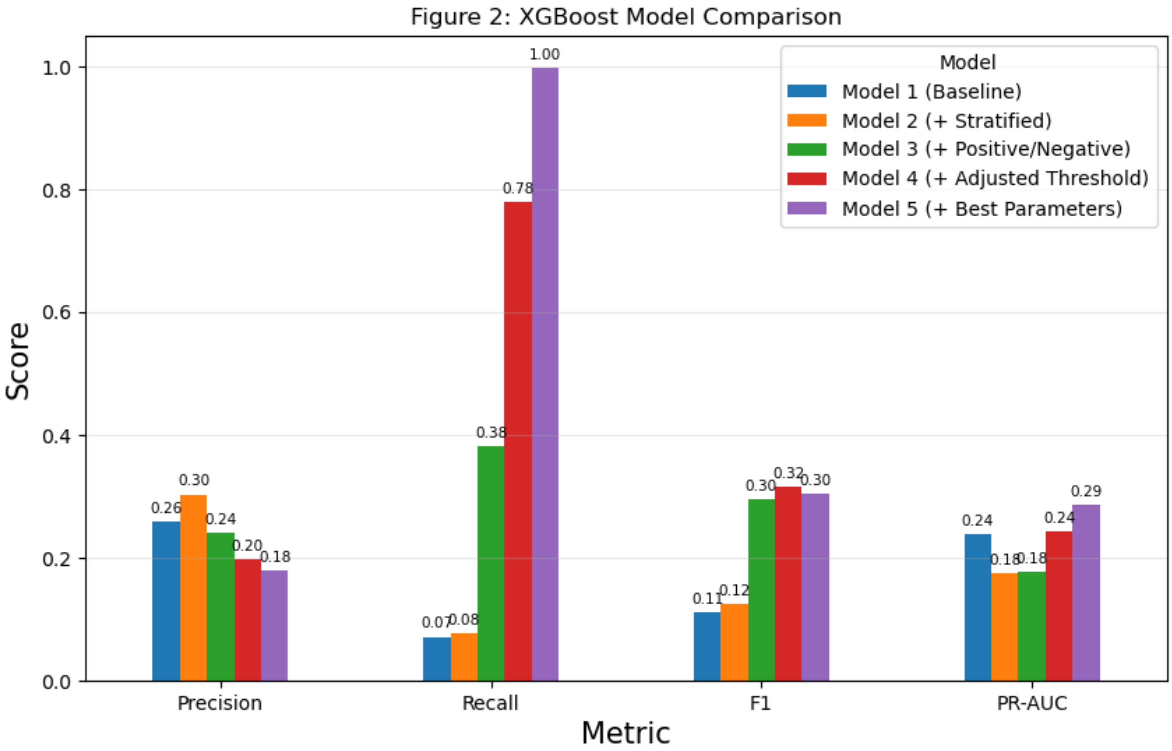
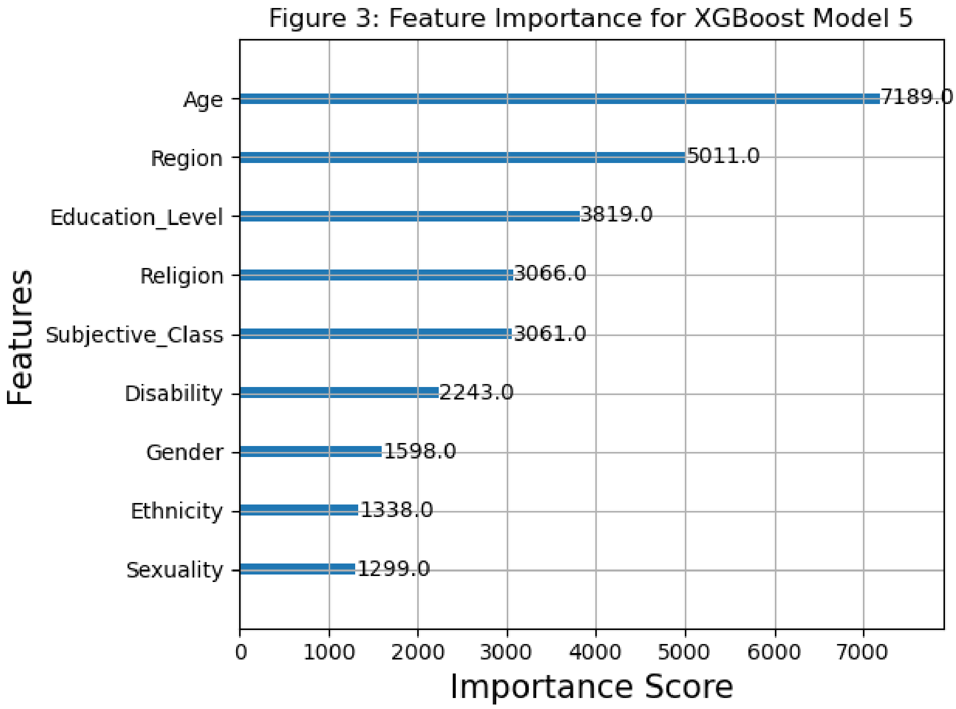


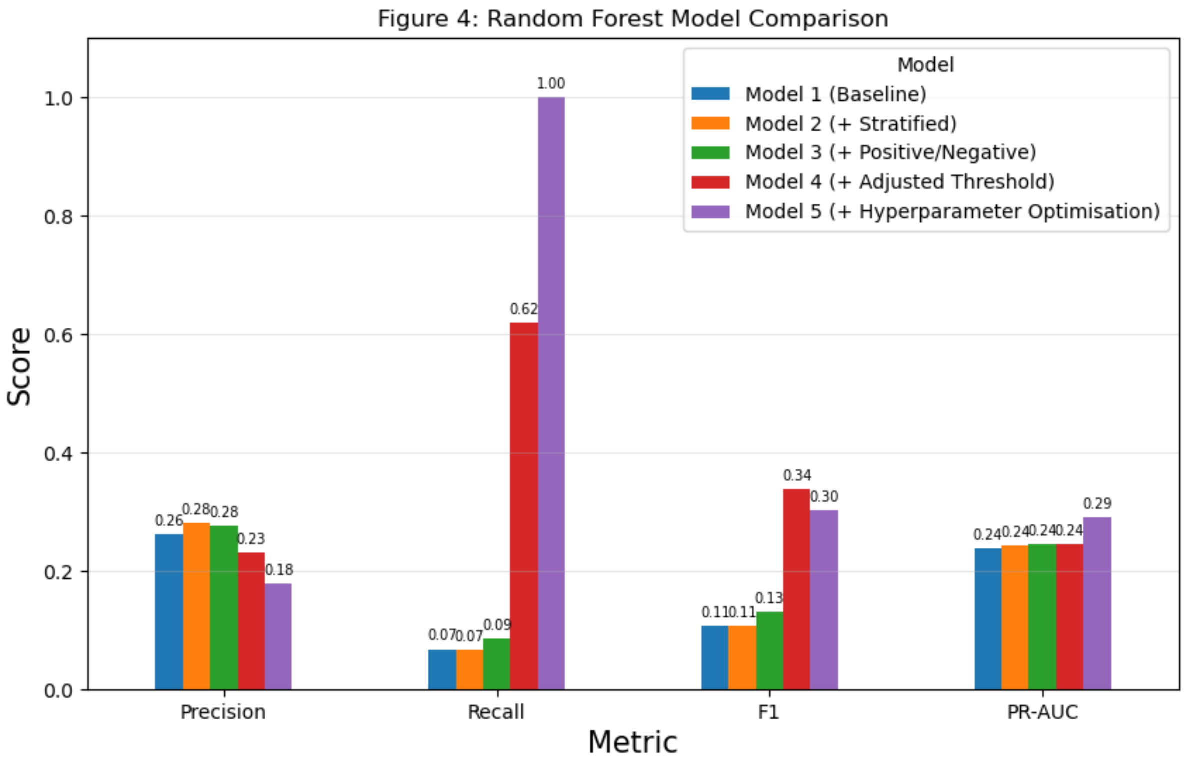
Figure 2 illustrates the results of these XGBoost models across the performance metrics. Precision decreases across models from 0.26 in Model 1, peaking in Model 2 at 0.30, and falling to 0.18 in Model 3; reflecting a growing tendency to predict Labour voters more liberally. In contrast, recall improves substantially from 0.07 in Model 1 to 1.00 in Model 5, indicating that later models successfully identify nearly all Labour voters. The F1 score rises initially, peaking at 0.32 in Model 4 before dropping slightly to 0.30 in Model 5, balancing the trade-off between precision and recall. PR-AUC shows modest improvements, reaching 0.29 in Model 5, suggesting that despite lower precision, the model's ability to rank Labour voters effectively improves with successive enhancements.



The feature importance for XGBoost Model 5 (Figure 3) shows that age is the strongest predictor of Labour vote choice, followed by region and education level. Religion and subjective class also contribute substantially, while disability, gender, ethnicity, and sexuality have smaller but non-substantial influence, highlighting the multifaceted social identity basis of voting behaviour.

The same modelling strategy was then applied using Random Forest, with one-hot encoding applied to all categorical features to ensure computability with the algorithm. The train-test data splits used in the XGBoost specifications to allow for a direct comparison of model performance. Model 1 used 100 trees with the 'entropy' criterion as a baseline. Model 3 applied balanced class weights to address class imbalance and Model 5 performed hyperparameter optimisation through 'RandomizedSearchCV', tuning the number of estimators, maximum features, maximum depth, minimum samples for split and leaf, and criterion.

Figure 4 presents the performance of the five Random Forest specifications. Like XGBoost, precision decreases across the models, while recall improves substantially, rising from 0.07 in Model 1 to 1.00 in Model 5. F1 score peaks at Model 4 before a slight decline in Model 5. PR-AUC also increases to 0.29 in Model 5, indicating improved overall model ranking of Labour voters. Overall, Random Forest and XGBoost display very similar patterns across these metrics, suggesting that both ensemble approaches respond comparably to class imbalance adjustments and probability threshold tuning.



The top 20 feature importance for Model 5 (Figure 5) shows that education and religion emerge as the strongest predictors, with postgraduate and undergraduate qualifications and identifying with no religion contributing the most. Compared to XGBoost, Random Forest places slightly more emphasis on specific education and religion, arising from the one-hot encoding, but the broader patterns of age, class, and social identity factors shaping Labour vote choice is consistent across both models. This highlights the robustness of these predictors irrespective of the ensemble methods used.

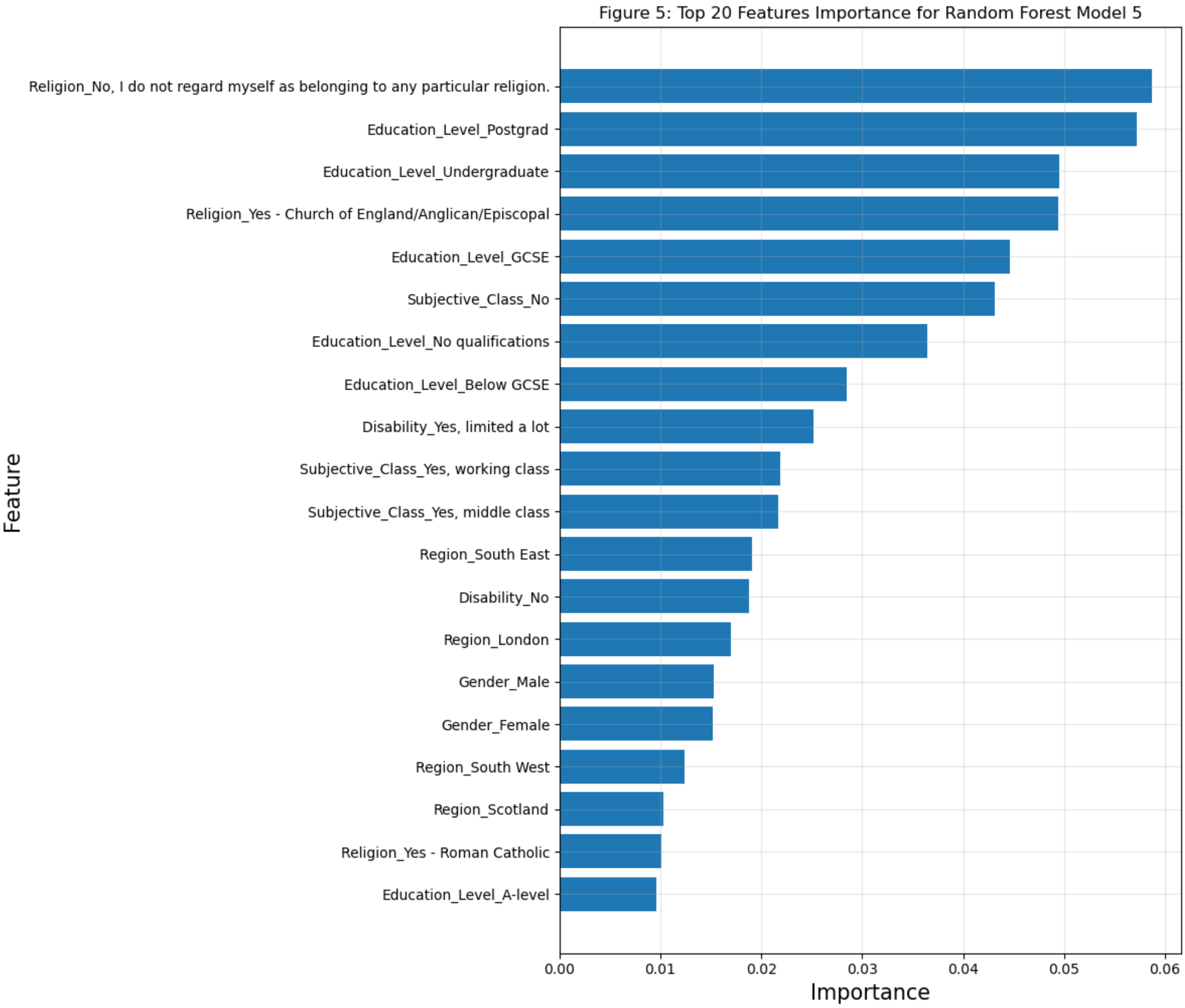
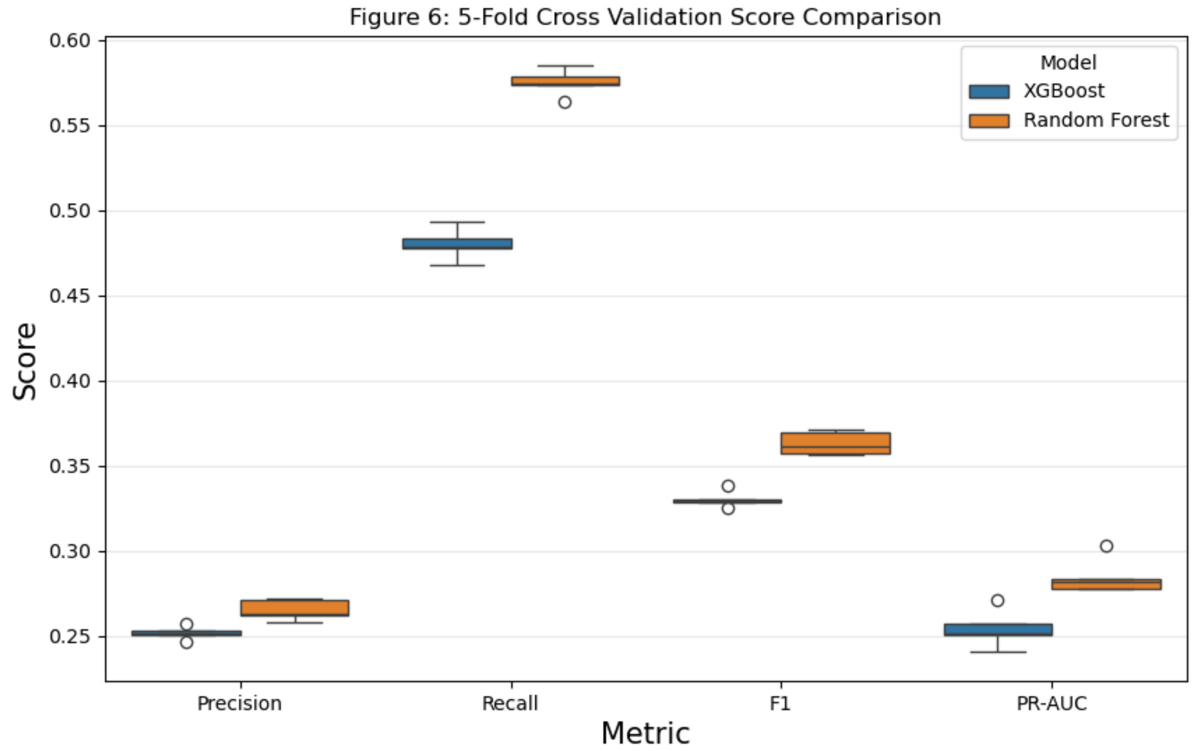
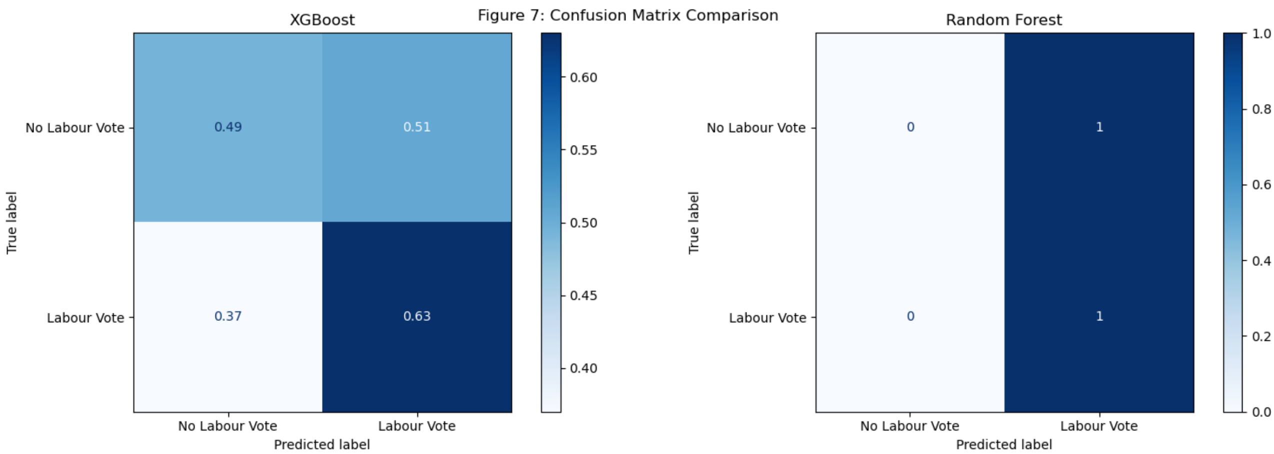


Figure 6 presents a boxplot of five-fold cross-validation results for both models, it shows that Random Forest consistently outperforms XGBoost across all cross-validation folds. It achieves higher median precision, recall, F1, and PR-AUC scores, with relatively tight interquartile ranges, indicating stable performance. While both models display modest predictive power overall, bagged trees demonstrates slightly stronger and more consistent generalisation than boosting.



Finally, Figure 7 displays the normalised results of a confusion matrix which shows that XGBoost correctly identifies 63% of Labour voters but misclassifies 51% of non-Labour voters as Labour, reflecting high recall but weak specification. Random Forest predicts all cases as Labour, achieving perfect recall but zero specificity, indicating severe overprediction of Labour and limited discriminative capacity.



Conclusion

Overall, across both models this essay found that age, education, region, religion, region, and subjective class consistently emerged as the strongest predictors, indicating that Labour's 2024 coalition is structured more by educational and cultural cleavages then by class alone. In terms of performance, Random Forest demonstrated slightly greater consistency and overall predictive strength than XGBoost under cross-validation. Future research could examine intersectional identity interactions more explicitly and formally compare models using statistical tests such as McNeamer's test to assess the significant differences in classification importance (Rainio, Teuho and Klén, 2024).

References

Biau, G. and Scornet, E. (2016) 'A random forest guided tour', *TEST*, 25(2), pp. 197–227. Available at: https://doi.org/10.1007/s11749-016-0481-7.

Bunting, H. *et al.* (2025) 'Political Identities in Britain During Brexit and Covid: Their Construction and Impact on Preferences and Behaviour', *Reflections on Polarisation and Inequalities in Brexit Pandemic Times*. Routledge.

Bunting, H. (2025) 'THE RESULTS: HOW BRITAIN VOTED IN 2024', *Parliamentary Affairs* [Preprint]. Available at: https://doi.org/10.1093/pa/gsaf017.

Cook, J. and Ramadas, V. (2020) 'When to consult precision-recall curves', *The Stata Journal: Promoting communications on statistics and Stata*, 20(1), pp. 131–148. Available at: https://doi.org/10.1177/1536867X20909693.

Evans, G. (2000) 'The Continued Significance of Class Voting', *Annual Review of Political Science*, 3(Volume 3, 2000), pp. 401–417. Available at: https://doi.org/10.1146/annurev.polisci.3.1.401.

Johnston, R., Jones, K. and Manley, D. (2018) 'Age, sex, qualifications and voting at recent English general elections: an alternative exploratory approach', *Electoral Studies*, 51, pp. 24–37. Available at: https://doi.org/10.1016/j.electstud.2017.11.006.

Kolpinskaya, E. and Fox, S. (2024) 'Religion and National Identity', *Insights into Political Identity*, p. 25. Available at: chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.understandingsociety.ac.uk/wp-content/uploads/insights/Insights_2024.pdf.

Prosser, C. (2024) 'Fragmentation revisited: the UK General Election of 2024', *West European Politics*, 0(0), pp. 1–13. Available at: https://doi.org/10.1080/01402382.2024.2430915.

Rainio, O., Teuho, J. and Klén, R. (2024) 'Evaluation metrics and statistical tests for machine learning', *Scientific Reports*, 14(1), p. 6086. Available at: https://doi.org/10.1038/s41598-024-56706-x.

Sahin, E.K. (2020) 'Assessing the predictive capability of ensemble tree methods for landslide susceptibility mapping using XGBoost, gradient boosting machine, and random forest', *SN Applied Sciences*, 2(7), p. 1308. Available at: https://doi.org/10.1007/s42452-020-3060-1.

Simon, E., Jennings, W. and Durrant, G. (2024) ‘The geography of educational voting: Understanding where individuals with similar qualifications vote differently across Britain’, *Political Geography*, 112, p. 103113. Available at: https://doi.org/10.1016/j.polgeo.2024.103113.

Appendices

Appendix A – GitHub Repository Link

https://github.com/RyanWhitehead-04/MLSDS-Problem-Set-1.git

Appendix B – Google Antigravity AI Agent Chat History

Chat Conversation

Note: _This is purely the output of the chat conversation and does not contain any raw data, codebase snippets, etc. used to generate the output._

User Input

I have British Election study wave data loaded in a pandas dataframe called BES.
The target variable is 'Labour_vote' (0 or 1) and all the other columns are binary features (0/1).
I have already split the data into training (80%) and testing (20%), and stratified this to overcome class imbalance, the training and test datasets are called BES_training_stratified and BES_testing_stratified, respectively.

Please help me:
1. Set up a hyperopt search space for XGBclassifier hyperparamters (max_depth, learning_rate, n_estimators, subsample, colsample_bytree, min_child_weight, and gamma).
2. Define a objective function that minimises negative F1 score.
3. Run Bayesian optimisation with 50 evaluations.
4. Add comments explaining each step.

Provide the code so I can copy and paste it into my notebook.

Planner Response

Here is the complete code to set up and run the Bayesian optimization for your XGBoost model using `hyperopt`. I have structured it sequentially so you can copy and paste it directly into your Jupyter Notebook.

```python
import numpy as np
from xgboost import XGBClassifier
from sklearn.metrics import f1_score
from hyperopt import fmin, tpe, hp, STATUS_OK, Trials
from hyperopt.pyll.base import scope

# --- PREPARATION ---
```

```
# First, separate your features (X) and target variable (y) from your datasets
X_train = BES_training_stratified.drop('Labour_vote', axis=1)
y_train = BES_training_stratified['Labour_vote']

X_test = BES_testing_stratified.drop('Labour_vote', axis=1)
y_test = BES_testing_stratified['Labour_vote']


# --- STEP 1: Set up the hyperopt search space ---

# Define the boundaries for the XGBClassifier hyperparameters you want to optimize
# - hp.quniform returns values rounded to the nearest given step (useful for integers)
# - hp.uniform returns continuous floats within the range
space = {
    'max_depth': hp.quniform('max_depth', 3, 10, 1),              # Depth of the tree
(integers roughly 3 to 10)
    'learning_rate': hp.uniform('learning_rate', 0.01, 0.3),      # Step size shrinkage
(continuous 0.01 to 0.3)
    'n_estimators': hp.quniform('n_estimators', 50, 500, 10),     # Number of boosting
rounds (trees)
    'subsample': hp.uniform('subsample', 0.5, 1.0),               # Fraction of
observations to safely sample per tree
    'colsample_bytree': hp.uniform('colsample_bytree', 0.5, 1.0), # Fraction of columns to
sample per tree
    'min_child_weight': hp.quniform('min_child_weight', 1, 10, 1),# Minimum instance
weight needed in a child
    'gamma': hp.uniform('gamma', 0.0, 5.0)                        # Minimum loss reduction
required for split
}


# --- STEP 2: Define objective function minimizing negative F1 score ---

def objective(params):
    # hyperopt passes parameters as floats by default if using uniform/quniform.
    # We must cast the hyperparameters that XGBoost expects to be integers.
    params['max_depth'] = int(params['max_depth'])
    params['n_estimators'] = int(params['n_estimators'])

    # Initialize the XGBClassifier with the current set of parameters proposed by hyperopt
    clf = XGBClassifier(
        **params,
```

```
        random_state=42,       # Ensures reproducibility
        eval_metric='logloss'  # Prevents unnecessary iteration warnings in newer XGBoost versions
    )

    # Train the model on the stratified training data
    clf.fit(X_train, y_train)

    # Predict on the stratified test data
    preds = clf.predict(X_test)

    # Calculate the positive F1 score (harmonic mean of precision and recall)
    f1 = f1_score(y_test, preds)

    # hyperopt is designed to strictly MINIMIZE the return value of the objective function.
    # Therefore, to maximize the F1 score, we return the negative F1 score.
    # STATUS_OK tells hyperopt that the function executed successfully without errors.
    return {'loss': -f1, 'status': STATUS_OK}


# --- STEP 3: Run Bayesian optimization with 50 evaluations ---

# The Trials object will keep a record of all the hyperparameters and corresponding losses evaluated
trials = Trials()

# fmin is the main optimization function built into hyperopt
best = fmin(
    fn=objective,                    # The objective function we just defined above
    space=space,                     # The hyperparameter search space
    algo=tpe.suggest,                # The optimization algorithm (Tree-structured Parzen Estimator)
    max_evals=50,                    # The number of models/evaluations to test (50, as requested)
    trials=trials,                   # The history dictionary to record metrics
    rstate=np.random.default_rng(42) # Random state for consistent/reproducible optimization behavior
)

# Output the best hyperparameters discovered
print("Best hyperparameters found during the Bayseian Search:")
```

```
print(best)
```

A Brief Note on the Final Output
When you print the `best` dictionary at the end, remember that hyperopt returns exact float values for all `quniform` searches. If you decide to train a final, permanent model on these best parameters, remember to cast the integers where necessary, for example: `max_depth = int(best['max_depth'])`.