Task 2 - Data Classification

Ishan Maharjan

May 31, 2025

1 Data Preprocessing & Feature Engineering

The dataset comprises three CSV files: `train_set.csv`, `test_set.csv`, and `unseen_set.csv`, each containing features, an `ID` column, and a `CLASS` label (except for the unseen set). The preprocessing and feature engineering steps are as follows:

- **Drop Irrelevant Features**: Removed the `Feature_0` column, which was identified as irrelevant based on exploratory data analysis.
- **Clip Extreme Values**: For each numeric feature, clipped values to the 5th and 95th percentiles of the training set to mitigate the impact of outliers.
- **Drop Near-Constant Features**: Identified and removed 6,572 features with low variance (near-constant) across the training set, reducing dimensionality.
- **Feature Scaling**: Applied standard scaling to normalize feature distributions using the training set's mean and standard deviation.
- **Feature Selection**: Used a Random Forest Classifier to select the top 150 features based on feature importance scores, ensuring a manageable feature set for modeling.

2 Model Architectures & Key Hyperparameters

Three classification models were trained and evaluated:

- **Random Forest Classifier**: Configured with `random_state=42` and `class_weight='balanced'`. Hyperparameter grid: `n_estimators` in {200, 300}, `max_depth` in {None, 20}, `min_samples_split` in {2, 5}.
- **XGBoost Classifier**: Configured with `random_state=42` and `eval_metric='logloss'`. Hyperparameter grid: `learning_rate` in {0.01, 0.1, 0.2}, `max_depth` in {5, 7, 10}, `n_estimators` in {200, 300}.
- **Logistic Regression**: Configured with `random_state=42`, `class_weight={0:1, 1:1.5}`, and `max_iter=1000`. Hyperparameter grid: `C` in {0.5, 1, 2, 5, 10}, `solver` in {'liblinear', 'lbfgs'}.

Hyperparameters were tuned using GridSearchCV, optimizing for the F1 score.

3 Cross-Validation Scheme

A 5-fold cross-validation was employed during hyperparameter tuning for each model using GridSearchCV. Additionally, a separate 5-fold cross-validation was performed on the training set for the best models to assess generalization performance. The training set was used to train models, while the test set was reserved for final evaluation, ensuring an unbiased estimate of model performance.

4 Results

The performance of the models on the test set is summarized in Table 1. Metrics include Accuracy, Precision (Macro), Recall (Macro), F1-Score (Macro), F1-Score (Weighted), and ROC-AUC.

Table 1: Evaluation Metrics for All Models on the Test Set

Model	Accuracy	Precision (Macro)	Recall (Macro)	F1 (Macro)	F1 (Weighted)	ROC-
Random Forest	0.670	0.666	0.640	0.640	0.656	0.68
XGBoost	0.610	0.593	0.572	0.562	0.585	0.65
Logistic Regression	0.620	0.607	0.603	0.604	0.617	0.63

Random Forest was selected as the best model based on the highest F1-Macro score (0.640).

5 Discussion

5.1 Strengths

The pipeline effectively reduces dimensionality by removing 6,572 near-constant features, making the model computationally efficient while retaining predictive power with 150 selected features. The use of multiple evaluation metrics provides a comprehensive assessment of model performance. Random Forest's balanced performance across metrics (e.g., F1-Macro of 0.640 and ROC-AUC of 0.684) demonstrates its robustness for this classification task.

5.2 Limitations

The models exhibit moderate performance, with the best F1-Macro score at 0.640, indicating room for improvement in capturing class distinctions. The feature selection process, reliant on Random Forest importance, may overlook features that other algorithms might find valuable. Class imbalance, despite being addressed with class weights, likely contributes to the relatively low recall and precision scores, particularly for XGBoost (F1-Macro of 0.562).

5.3 Future Improvements

With more time, I would explore advanced feature selection techniques, such as recursive feature elimination or mutual information, to capture a broader range of predictive features. Incorporating Principal Component Analysis (PCA) could further reduce dimensionality by transforming the 150 features into a smaller set of principal components, potentially capturing more variance and reducing noise. Implementing SMOTE or other oversampling methods could better address class imbalance. Additionally, experimenting with ensemble methods like stacking or incorporating deep learning models (e.g., neural networks) might improve performance by capturing non-linear patterns more effectively.