CSE472 (Machine Learning Sessional)

Assignment–2

Logistic Regression with Bagging and Stacking

Department of Computer Science and Engineering

Bangladesh University of Engineering and Technology

Prepared by

Wasif Jalal
#1905084

Introduction

This assignment presents us with the task of implementing a logistic regression classifier and experimenting with its variations and parameters on different datasets. It includes requirements for data preprocessing, model training, and evaluation. The experiment compares the performance of a logistic regression classifier, both standalone and in ensemble methods (bagging and stacking), on multiple datasets, in order to understand the effectiveness of these methods in various scenarios, with detailed performance metrics provided for each.

Methods

1. **Fetching and Loading Data**: The first step involves fetching datasets from an external repository or loading them from local files. This step accommodates various formats and handles missing values.
2. **Data Preprocessing**:
   - **Handling Missing Values**: Replaces placeholders with NaN and either drops rows with missing values or imputes them with the most frequent or median value.
   - **Removing Duplicates**: Eliminates duplicate rows from the dataset if specified.
   - **Feature Scaling**: Applies Min-Max Scaling or Standard Scaling based on user selection to standardise feature values.
   - **Feature Selection**: Filters features based on their correlation or information gain with the target variable, retaining only those with significant weight.
3. **Model Training**:
   - **Logistic Regression**: Trains a logistic regression model using gradient descent with L1 or L2 regularisation.
   - **Bagging/Bootstrapping**: Creates multiple bootstrap samples from the training data and trains a separate logistic regression model on each sample to form an ensemble.
   - **Stacking**: Uses predictions from the bagged models as features for a meta-classifier, aiming to enhance overall performance.
4. **Evaluation**:
   - **Performance Metrics**: Computes accuracy, precision, recall, F1-score, specificity, ROC AUC, and Precision-Recall AUC.
   - **Visualisation**: Uses violin plots to visualise and compare performance metrics across the bagged logistic regression classifiers.

Usage Guide

To run the code on a specific dataset, follow these instructions:

1. **Dataset 1**: Uncomment the lines under the `# Dataset 1` section.
2. **Dataset 2**: Uncomment the lines under the `# Dataset 2` section and comment out Dataset 1 and Dataset 3 lines.
3. **Dataset 3**: Uncomment the lines under the `# Dataset 3` section and comment out Dataset 1 and Dataset 2 lines.

To run the script on a new dataset, create a new section similar to the existing dataset sections and uncomment all lines within that section. Adjust the parameters (set as global variables in the script) as needed for the new dataset. The purpose of each variable is described on the next page.

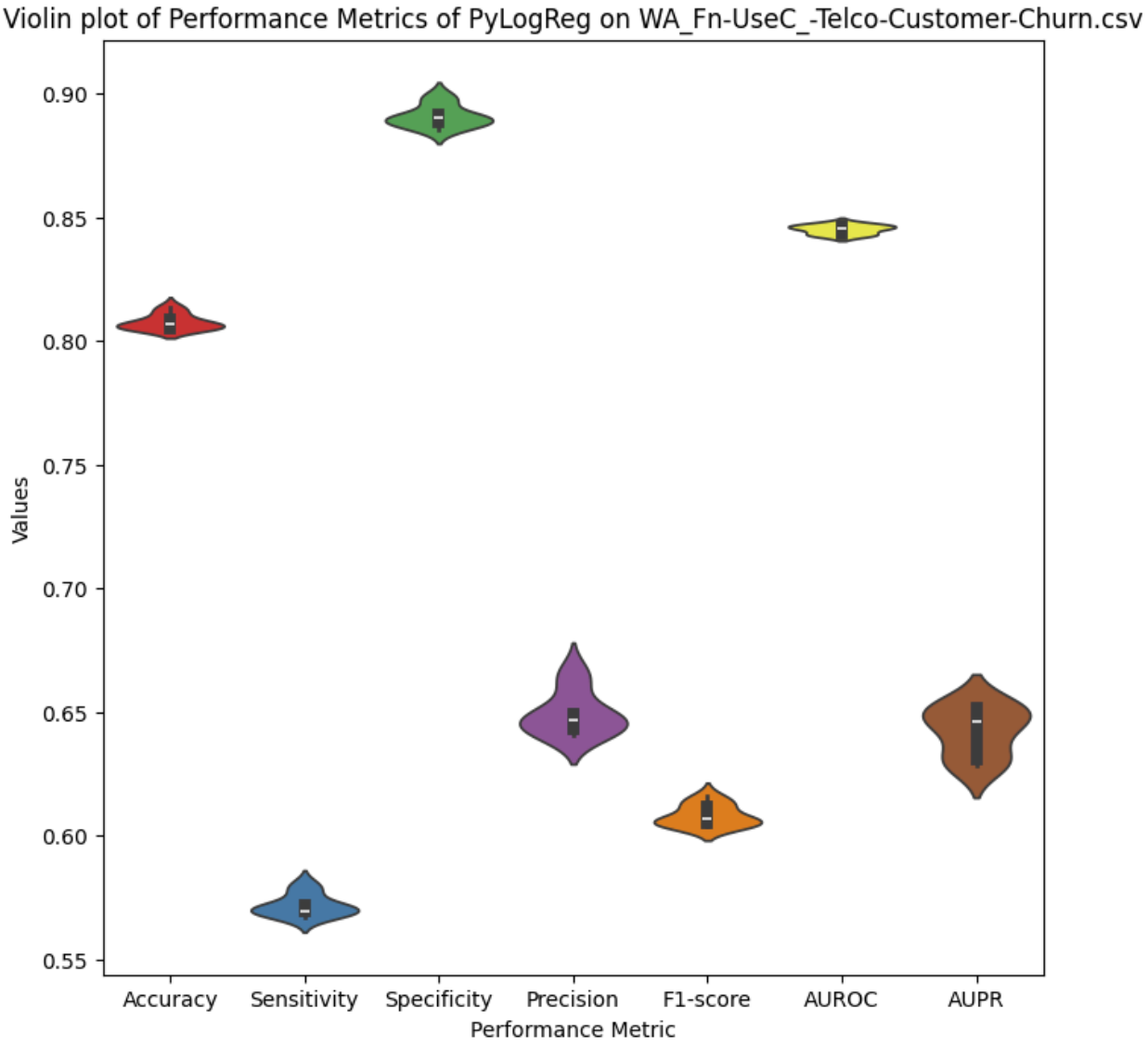
- **`remote`**: Indicates whether to fetch the dataset from an external repository (`True`) or use a local file (`False`). Setting as `False` will print out a direct link to the CSV upon successfully importing it from UCI's repository. That can be downloaded and used locally for faster access.
- **`data_filepath`**: Path to the dataset file or the name of the dataset in the repository.
- **`target_label`**: The name of the column that contains the target variable for classification.
- **`index_col`**: Specifies the column to use as the DataFrame index, if applicable.
- **`has_index`**: Indicates whether the dataset includes an index column (`True`) or not (`False`).
- **`missing_placeholders`**: List of values that represent missing data in the dataset.
- **`dropna`**: Whether to drop rows with missing values (`True`) or impute them (`False`).
- **`drop_dup`**: Whether to remove duplicate rows from the dataset (`True`) or not (`False`).
- **`scaler_type`**: The type of scaler to use for feature scaling: `'m'` for MinMaxScaler or `'s'` for StandardScaler.
- **`feature_sel`**: Method for feature selection. Can be `'corr'` for correlation-based selection or `'ig'` for information gain-based selection.
- **`corr_threshold`**: The threshold for filtering features based on their absolute correlation with the target variable when using correlation-based feature selection.
- **`ig_threshold`**: The threshold for selecting features based on information gain when using information gain-based feature selection.
- **`drop_cols`**: List of columns to remove from the dataset.
- **`prune`**: Indicates whether to perform class pruning to balance the dataset (`True`) or not (`False`). If `prune` is `True`, the following parameters are required:
    - **`neg_pos[0]`**: Number of negative samples to retain. Positive integer limits the number of negative samples; `0` or negative integer means no pruning for negative samples.
    - **`neg_pos[1]`**: Number of positive samples to retain. Positive integer limits the number of positive samples; `0` or negative integer means no pruning for positive samples.
    - **`neg_pos[2]`**: Value in the target variable column that represents the negative class. This value is used to differentiate between negative and positive samples for pruning.
- **`alpha`**: Learning rate for the gradient descent algorithm used in logistic regression.
- **`max_iter`**: Maximum number of iterations for the gradient descent algorithm.
- **`penalty`**: Type of regularisation to use: `'l1'` for L1 regularisation or `'l2'` for L2 regularisation.
- **`Lambda`**: Regularization parameter that controls the strength of regularisation.
- **`random_state`**: Seed for random number generation to ensure reproducibility.
- **`meta_alpha`**: Learning rate for the meta-classifier in stacking.
- **`meta_max_iter`**: Maximum number of iterations for the meta-classifier.
- **`meta_penalty`**: Type of regularisation to use for the meta-classifier: `'l1'` for L1 regularisation or `'l2'` for L2 regularisation.
- **`meta_Lambda`**: Regularisation parameter for the meta-classifier.

Performance Evaluation

Dataset-1 (Telco Customer Churn)

Performance on test set

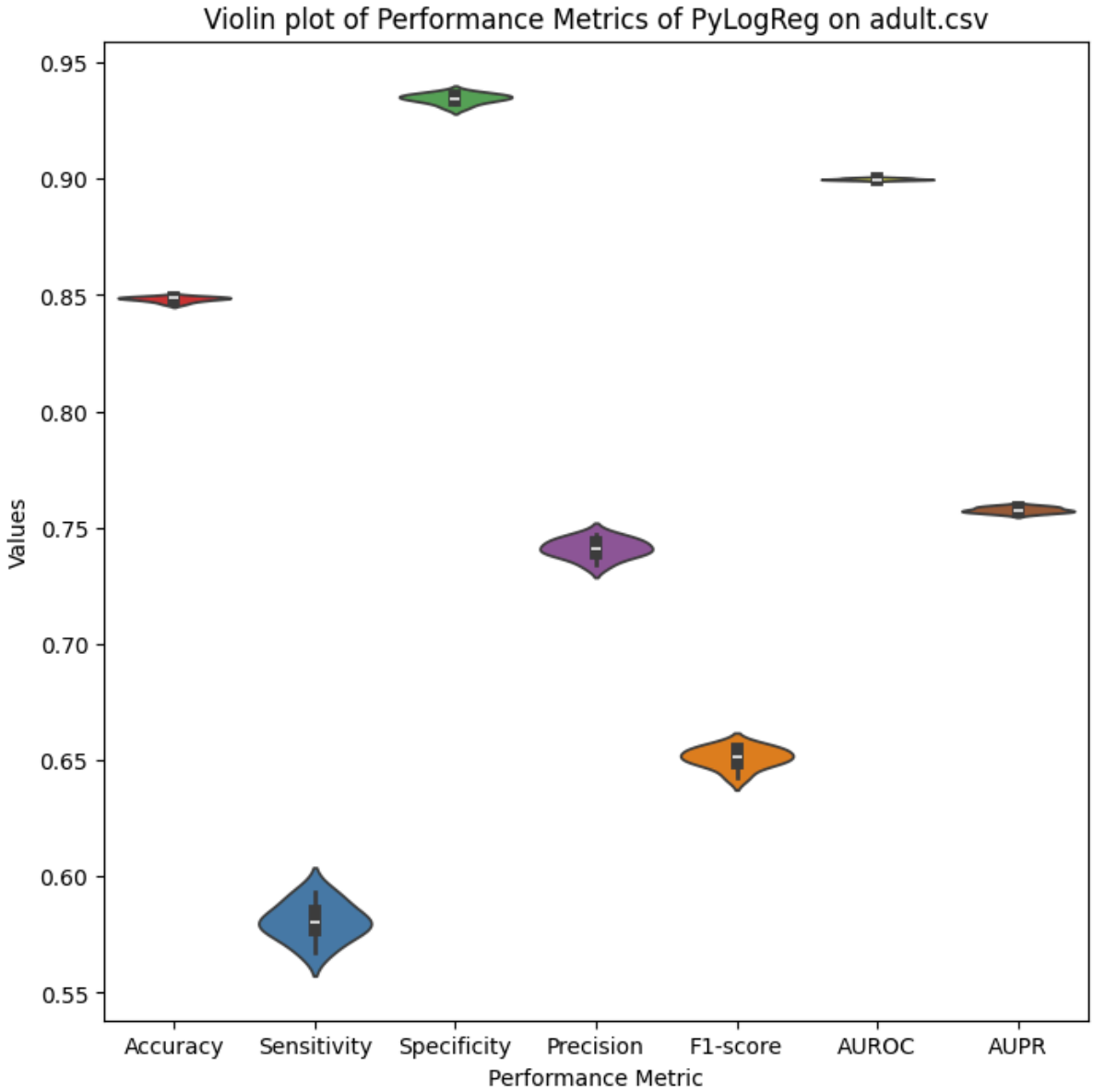
Method	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR	0.807786 ± 0.0031	0.571904 ± 0.0044	0.891026 ± 0.0044	0.649478 ± 0.0089	0.608191 ± 0.0043	0.845295 ± 0.0018	0.642395 ± 0.0099
Voting ensemble	0.812367	0.572207	0.897115	0.662461	0.614035	0.845496	0.641287
Stacking ensemble	0.814499	0.564033	0.902885	0.672078	0.613333	0.845549	0.642708



Dataset-2 (Adult – UCI)

Performance on test set

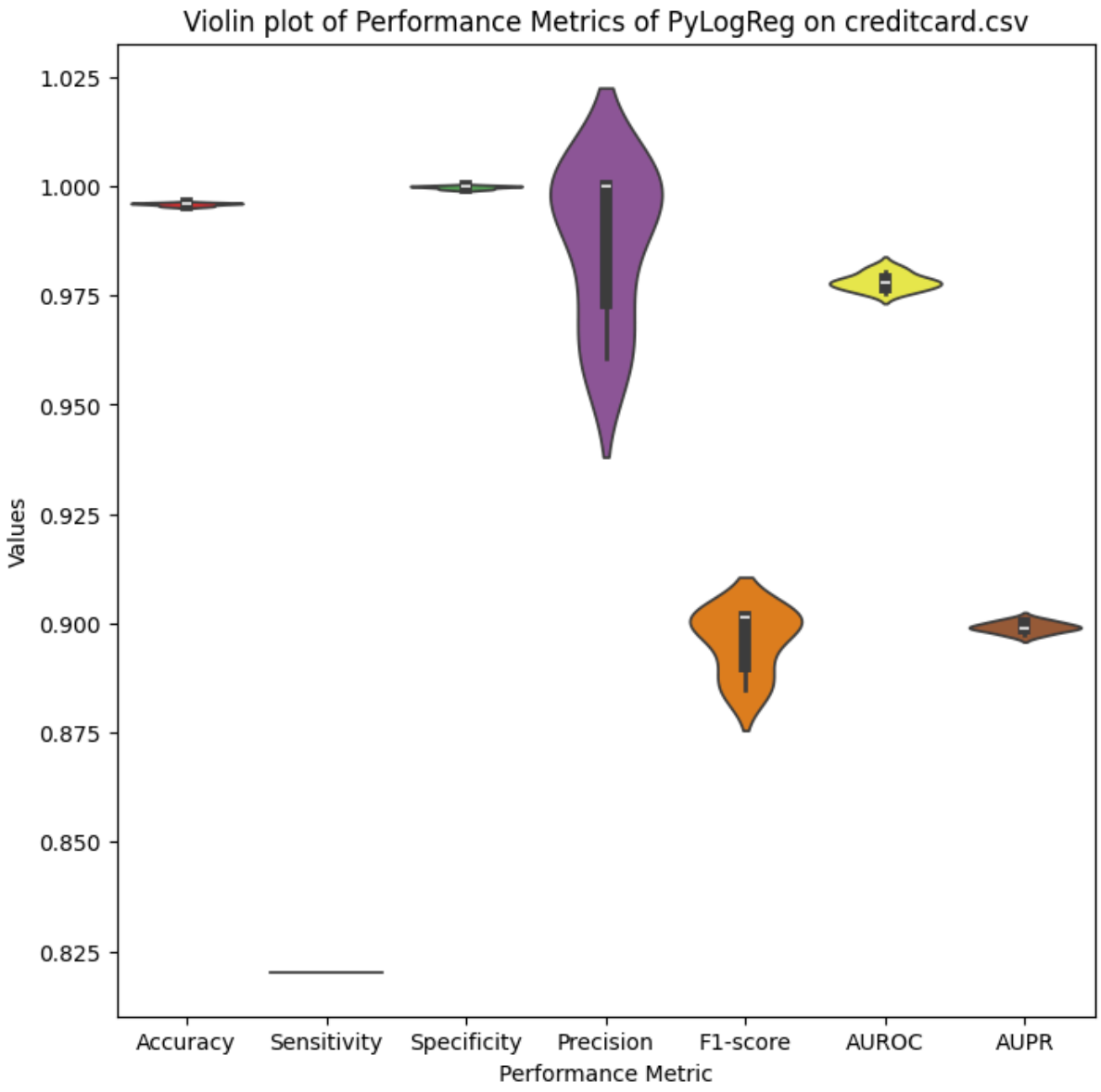
Method	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR (avg.)	0.848284 ± 0.0010	0.580744 ± 0.0080	0.934555 ± 0.0020	0.741076 ± 0.0040	0.651143 ± 0.0042	0.899897 ± 0.0004	0.757691 ± 0.0012
Voting ensemble	0.84891	0.580605	0.935427	0.743548	0.652051	0.900112	0.758188
Stacking ensemble	0.852493	0.58984	0.937187	0.751739	0.661021	0.902699	0.767091



Dataset-3 (Credit Card Fraud)

Performance on test set

Method	Accuracy	Sensitivity	Specificity	Precision	F1-score	AUROC	AUPR
LR	0.995849 ± 0.0003	0.820225 ± 0.0000	0.999750 ± 0.0003	0.986764 ± 0.0174	0.895758 ± 0.0072	0.978192 ± 0.0019	0.899132 ± 0.0012
Voting ensemble	0.996093	0.820225	1.0	1.0	0.901235	0.978442	0.900347
Stacking ensemble	0.996093	0.820225	1.0	1.0	0.901235	0.975346	0.893753



Observations

1. Dropping Rows and Columns:

- **Column Pruning**: Dropping redundant columns, such as in *Dataset-2*, improves model performance by reducing noise.
- **Row Pruning**: In *Dataset-3*, pruning rows was essential due to the sparse distribution of positive samples, which otherwise caused the dataset to be biased towards negative classes. While this balancing approach reduced overall accuracy, it significantly improved the F1-score, sensitivity, and AUPR. This suggests that accuracy is a poor metric for evaluating performance on skewed datasets.

2. Feature Selection:

- **Datasets 1 and 3**: Feature selection boosted performance by removing columns with low correlation to the target variable. However, setting the correlation threshold too high had a negative effect, as it led to discarding relevant features along with noise.
- **Dataset-1**: *Information gain* worked as a better metric for feature selection in this dataset, slightly improving the maximum accuracy and F1-score compared to correlation, because after encoding of columns, we are left mostly with discrete-value features. Information gain-based filtering led to poor performance in *Dataset-3*, because all of its features had continuous values. Calculation of information gain also proved to be far more time-consuming due to the large size of the dataset, thus suggesting correlation to a better metric for selecting features in a dataset with primarily continuous values.
- **Dataset-2**: This dataset performed best with no feature selection (i.e., when the minimum correlation threshold was set to zero). Given the overall weak correlation between the target and features, treating any feature as noise proved counterproductive.

3. Regularisation in Gradient Descent:

- **Datasets 1 and 2**: Adding regularisation to the model improved performance, though the effect was subtle in *Dataset-3*. L2 regularisation proved optimal for the meta-classifier across all datasets. The choice between L1 and L2 regularisation for base logistic regression predictors depended on the nature of the dataset.

4. Decaying Learning Rate:

- Applying a decaying learning rate in gradient descent did not yield noticeable performance improvements in any dataset.

5. Effect of Random Seeds:

- Random seed variation had a noticeable impact on performance metrics. Combined with hyperparameters such as α and λ, different random seed and hyperparameter combinations could cause performance to fluctuate within a range of approximately 5%.

6. Ensembling Methods:

- **Bagging**: Adjusting hyperparameters to maximise performance across methods revealed that bagging consistently resulted in as good or better performance than standalone logistic regressions.
- **Stacking**: Stacking further boosted performance, resulting in at least as good outcomes as bagging, or better.