Pattern Recognition HW1

Methods

We employ three classifiers in this study: Logistic Regression, Multi-Layer Perceptron, and Random Forest. The models are trained following the procedure described below, and the final performance is evaluated and compared on the test set.

* 1. Data Preprocessing

Numerical features are standardized and categorical features are one-hot encoded. All preprocessing steps are integrated into a Pipeline to avoid data leakage.

* 1. Train–Test Split

The dataset is split into **80% training** and **20% testing** using a stratified split to preserve class distribution.

* 1. Model Training and Hyperparameter Tuning

Models are trained using 5-fold cross-validation, and hyperparameters are selected via grid search based on balanced accuracy.

* 1. Model Evaluation

Model performance is assessed using classification metrics and the confusion matrix. For binary tasks, ROC curves and AUC scores are additionally reported.  
Learning curves are used to analyze overfitting and data efficiency.

Experiments and settings

* 1. Datasets
     + [Predict Students' Dropout and Academic Success](https://archive.ics.uci.edu/dataset/697/predict+students+dropout+and+academic+success) (3 classes)

This dataset contains academic and demographic features used to classify students into three outcomes: **Dropout**, **Enrolled**, or **Graduate**.

* + - [Wine](https://archive.ics.uci.edu/dataset/109/wine) (3 classes)  
      This dataset consists of chemical composition measurements of wine samples and aims to classify them into **three distinct cultivars**.
    - [Secondary Mushroom](https://archive.ics.uci.edu/dataset/848/secondary+mushroom+dataset) (2 classes)  
      This dataset includes descriptive features of mushroom characteristics and is used to classify mushrooms as **edible** or **poisonous.**
    - [CDC Diabetes Health Indicators](https://archive.ics.uci.edu/dataset/891/cdc+diabetes+health+indicators) (2 classes)  
      This dataset contains health and lifestyle survey data and is used to predict whether a person is **non-diabetic** or **pre-diabetic/diabetic**.

In the following sections, we refer to these datasets as **Dataset 1**, **Dataset 2**, **Dataset 3**, and **Dataset 4**, respectively.

* 1. Hyperparameter Settings

Logistic Regression

* + - * The **solver** parameter specifies the optimization algorithm, where **lbfgs**, **newton-cg**, and **sag** are suitable for L2-regularized models, while **saga** additionally supports L1 regularization and works well with large or sparse datasets.
      * The **penalty** parameter determines the type of regularization applied to the model, with **L2** providing smooth coefficient shrinkage and **L1** encouraging sparsity by pushing some feature coefficients to zero.
      * The **C** parameter controls the inverse strength of regularization, where **smaller values** imply stronger regularization to reduce overfitting, and **larger values** allow more flexible model fitting.

Multi-layer Perceptron

* + - * **The hidden\_layer\_sizes parameter controls the architecture of the neural network.**  
        Each tuple specifies the number of neurons in each hidden layer.
      * The **activation** parameter determines the nonlinear transformation applied at each neuron. The **relu** activation helps training deep networks efficiently by alleviating the vanishing gradient problem, while **tanh** is smoother and can be beneficial when the data distribution is centered around zero.  
        Different activations may lead to different learning dynamics and decision boundary shapes.
      * **The alpha parameter specifies the L2 regularization strength.**  
        This term penalizes large weight values during optimization to reduce overfitting.  
        Smaller alpha values allow more flexible model fitting, while larger values enforce smoother weight magnitudes and improve generalization stability.
      * **The batch\_size parameter indicates the number of samples processed before each update of the model weights.**  
        Smaller batch sizes introduce more stochasticity and may help escape poor local minima, while larger batch sizes provide smoother gradient estimates but may converge to less optimal solutions.

Random Forest

* + - * **The n\_estimators parameter specifies the number of decision trees in the forest.**A larger number of trees generally improves model stability and performance by reducing variance.
      * **The max\_depth parameter controls the maximum depth of each tree.**  
        Setting None allows trees to grow fully, capturing complex patterns but risking overfitting.
      * The **min\_samples\_leaf** parameter specifies the minimum number of samples required to form a leaf node.
      * **The min\_samples\_split parameter determines the minimum number of samples required to split an internal node.**  
        Increasing this value limits the tree’s tendency to grow too deep and complex, thereby improving model robustness and reducing sensitivity to noise.
      * The **max\_samples** controls the number of samples drawn from the training set to train each tree in the Random Forest

Using **smaller values** increases tree diversity but may slightly reduce individual tree accuracy — this can improve generalization and reduce overfitting.

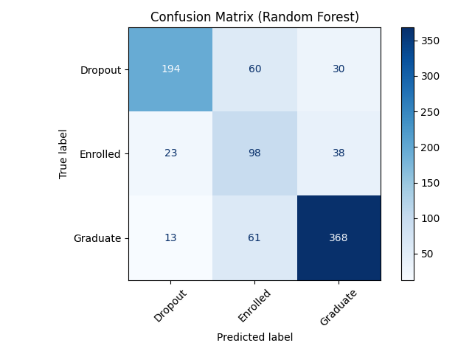
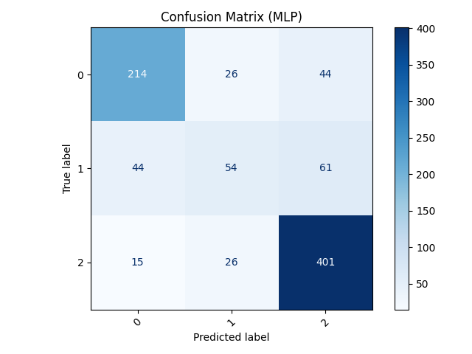
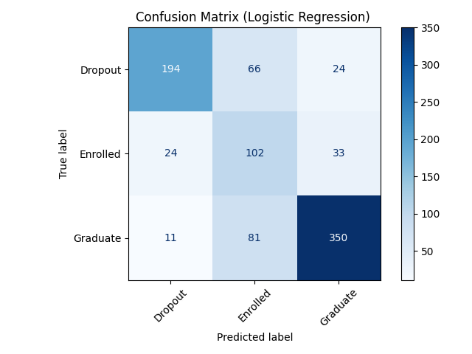
Using **larger values** allows each tree to learn from more data, which reduces bias but may increase correlation among trees and risk overfitting.

* 1. Results of Dataset 1

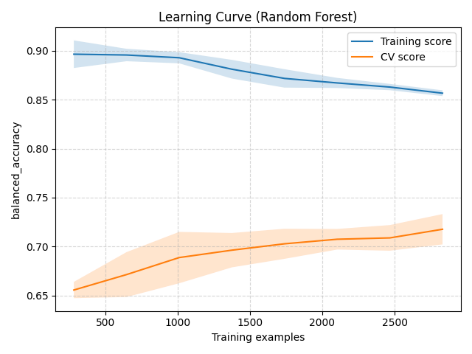
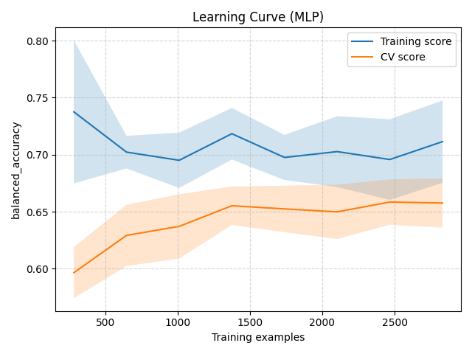
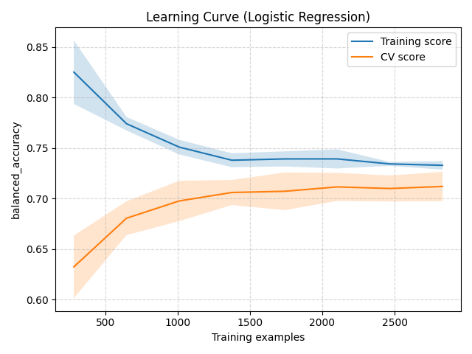
Balanced accuracy & Best parameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic regression | MLP | Random forest |
| Balanced accuracy | 0.712 | 0.673 | 0.719 |
| Best parameters | Solver : newton-cg  C : 100  Penalty : l2 | Hidden layer : (64)  Activation : tanh  Alpha :  Batch size : 32  Learning rate : | Estimators : 600  Max depth : 10  Min sample leaf : 4  Min sample split : 2  Max samples : 0.9 |

Confusion matrix



Learning curve

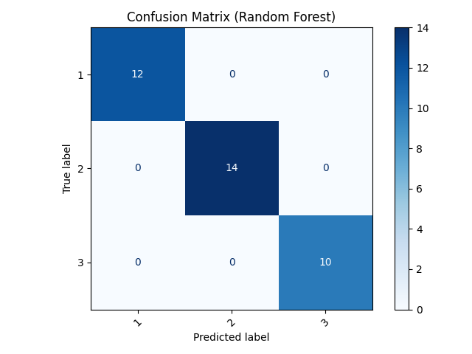
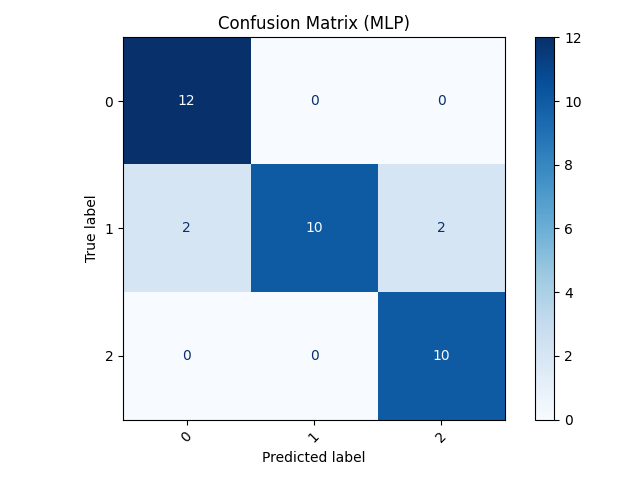
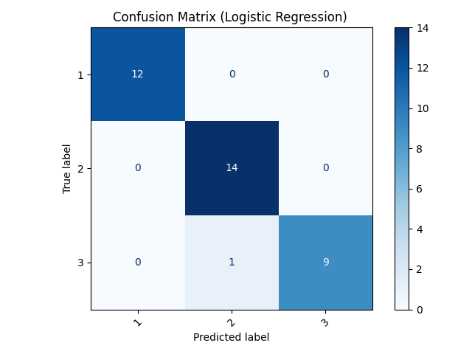


1. Results of Dataset 2

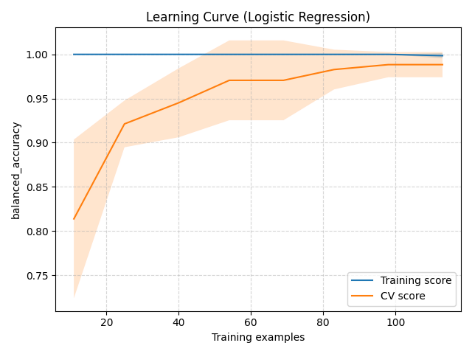
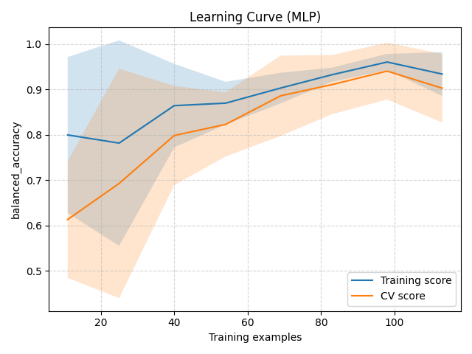
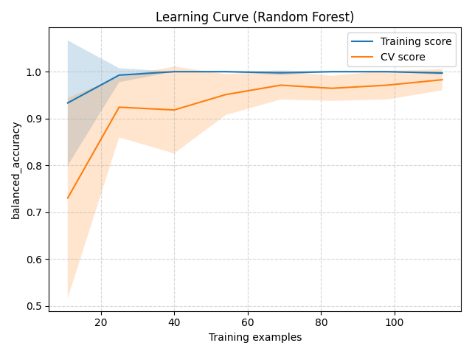
Balanced accuracy & Best parameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic regression | MLP | Random forest |
| Balanced accuracy | 0.988 | 0.946 | 0.983 |
| Best parameters | Solver : saga  C : 1  Penalty : l1 | Hidden layer : (128,64)  Activation : tanh  Alpha :  Batch size : 64  Learning rate : | Estimators : 300  Max depth : None  Min sample leaf : 2  Min sample split : 2  Max samples : 0.5 |

Confusion matrix



Learning curve

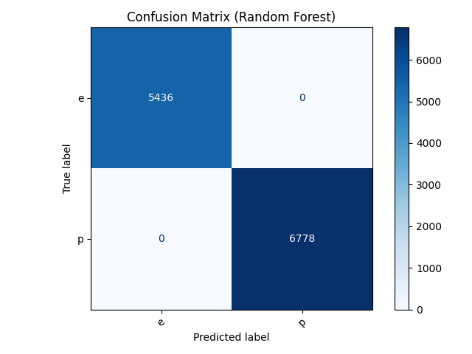
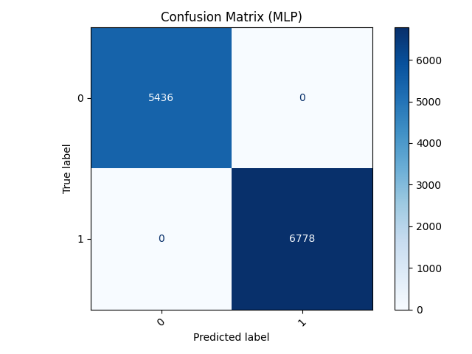
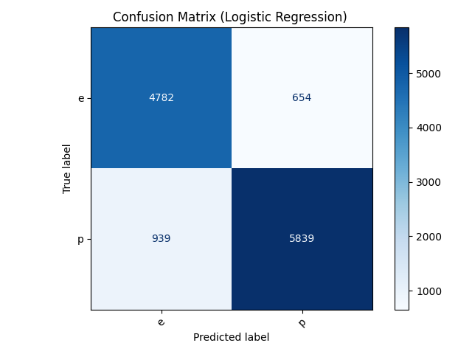


1. Results of Dataset 3

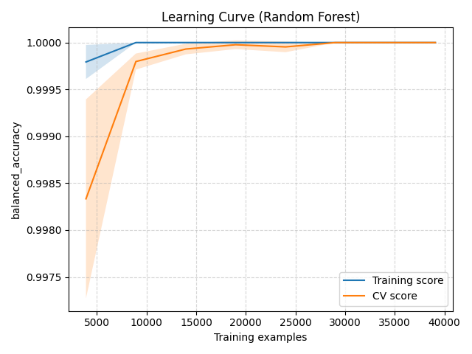
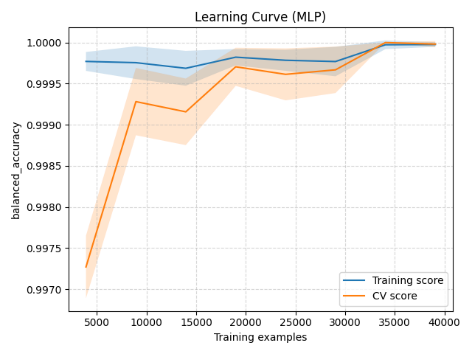
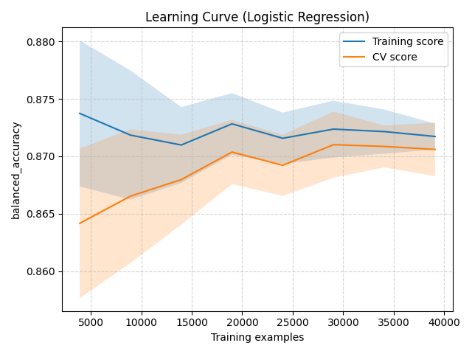
Balanced accuracy & Best parameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic regression | MLP | Random forest |
| Balanced accuracy | 0.871 | 1.000 | 1.000 |
| Best parameters | Solver : lbfg  C : 100  Penalty : l2 | Hidden layer : (128,64)  Activation : tanh  Alpha :  Batch size : 64  Learning rate : | Estimators : 300  Max depth : None  Min sample leaf : 2  Min sample split : 2  Max samples : 0.5 |

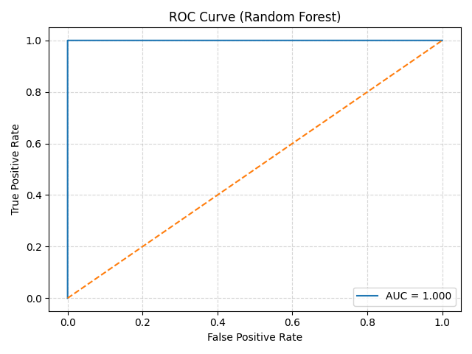
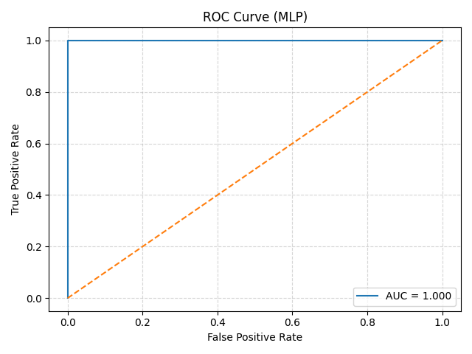
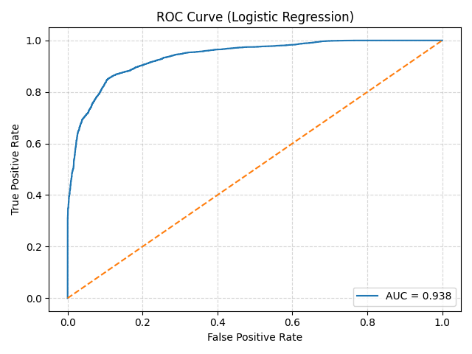
Confusion matrix



Learning curve



ROC & AUC

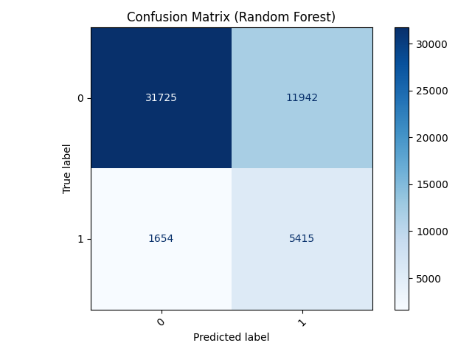
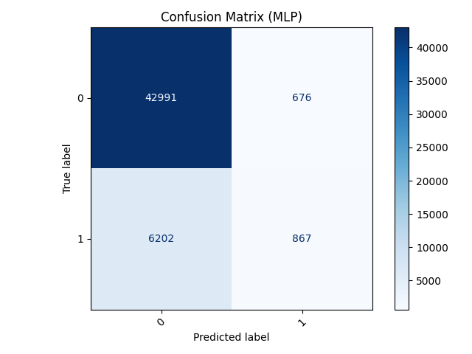
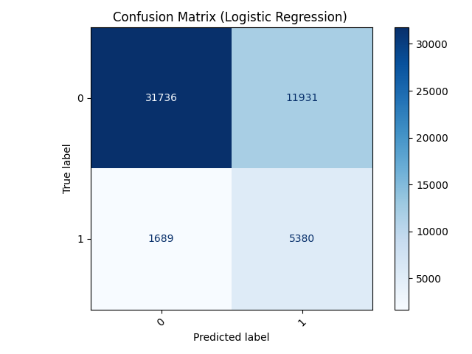


1. Results of Dataset 4

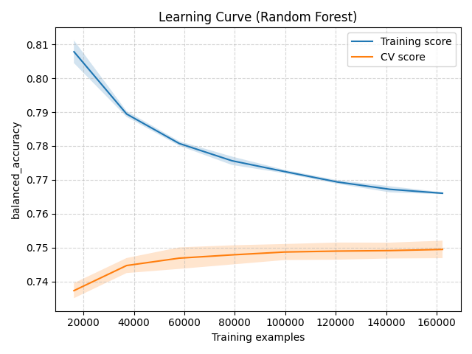
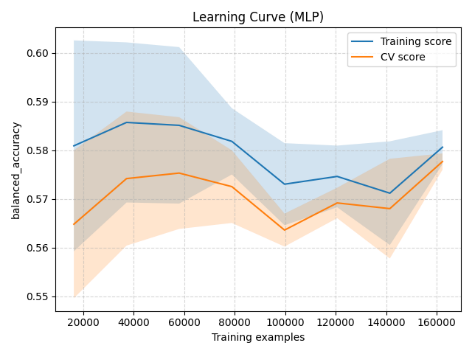
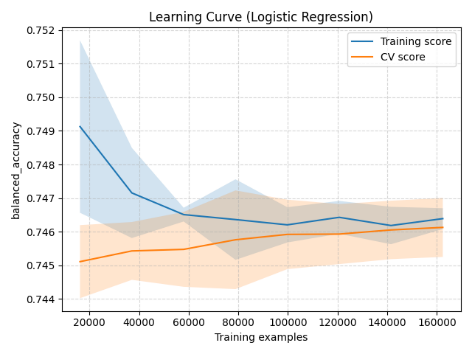
Balanced accuracy & Best parameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Logistic regression | MLP | Random forest |
| Balanced accuracy | 0.746 | 0.582 | 0.750 |
| Best parameters | Solver : saga  C : 100  Penalty : l1 | Hidden layer : (128,64)  Activation : relu  Alpha :  Batch size : 128  Learning rate : | Estimators : 300  Max depth : 10  Min sample leaf : 6  Min sample split : 2  Max samples : 0.9 |

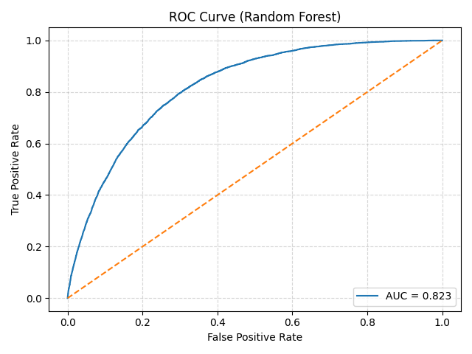
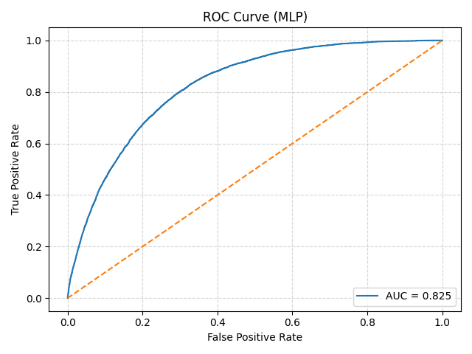
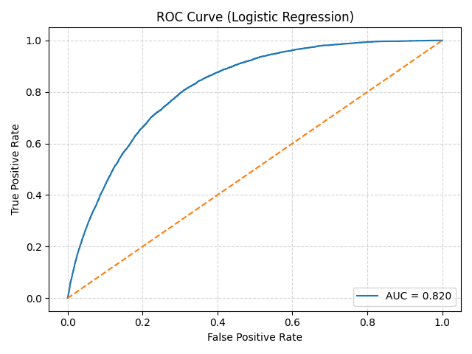
Confusion matrix



Learning curve



ROC & AUC



Analysis

1. Dataset 1

Based on the confusion matrices and learning curves, the dataset shows noticeable variation in class separability, with the **Enrolled**class being consistently the most difficult to classify across all three models. Logistic Regression tends to underfit due to its limited model capacity and linear decision boundaries. Although the MLP model is capable of capturing nonlinear patterns, the limited amount of training data leads to a large gap between training and validation performance, indicating overfitting. In contrast, the Random Forest model demonstrates more stable performance, effectively capturing nonlinear relationships while maintaining relatively high validation accuracy without large fluctuations. Therefore, among the three models, Random Forest achieves the best overall performance in this study.

1. Dataset 2

For the small-scale dataset, all three classifiers demonstrate strong performance, but their learning behaviors differ. Logistic Regression achieves stable and accurate results, suggesting that the class boundaries in this dataset are close to linearly separable. The MLP model shows higher variance during training and validation, indicating mild overfitting due to the limited amount of training data. In contrast, the Random Forest model obtains the most stable and consistent results across both training and validation, benefiting from its ability to capture non-linear feature interactions without requiring large data volumes. Therefore, Random Forest is the most suitable model for this dataset

1. Dataset 3

In this dataset, all three models achieve strong classification performance, while the MLP and Random Forest models reach near-perfect results (AUC = 1.00). This indicates that the dataset is highly separable, with clear distinctions between the classes. Logistic Regression also performs well (AUC ≈ 0.94), but it still misclassifies some boundary samples, reflecting the limitation of linear models when dealing with more complex decision boundaries. Overall, MLP and Random Forest demonstrate the best performance on this dataset.

1. Dataset 4

All three models show similar AUC values around 0.82 but vary in balanced accuracy.  
Logistic Regression and Random Forest achieve more stable results, while the MLP fails to converge effectively.