Pattern Recognition HW1

Methods

* 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.
  1. Results

Dataset 1