Lab 2

Classification

In data mining, **classification** is an important task which used to categorize data into classes. The objective is to assign a label to each input sample based on its attributes (also its features). This lab focuses on implementing widely used classification algorithms:

1. **Decision Tree**
2. **Ensemble Model: Random Forest**
3. **Naive Bayes**
4. **Support Vector Machine**
5. **Neural Network: Multilayer Perceptron**

Students will explore these algorithms with a real world dataset and compare the results using various evaluation metrics:

1. **Confusion Matrix**
2. **Accuracy**
3. **Precision**
4. **Recall (Sensitivity or True Positive Rate)**
5. **Specificity (True Negative Rate)**
6. **F1 Score**
7. **ROC Curve and AUC Score**

By completing this lab, students will:

1. Familiar with libraries that support classification tasks for implementing above algorithms.
2. Experiment with hyper-parameter tuning to optimize model performance.
3. Visualize and analyze the performance of the models using evaluation metrics.
4. Gain insights into the strengths and weaknesses of different classification algorithms.

1 Dataset

The *Adult Census Income* dataset, also known as the *Census Income dataset* or the *Adult dataset*, is sourced from the U.S. Census Bureau Database and is commonly used in classification tasks. The goal of this dataset is to predict if an individual's annual income exceeds $50,000 based on demographic characteristics.

The dataset contains 48,842 records and 14 attributes (features) as follows:

Attribute	**Description**
`age`	The age of the individual.
`workclass`	The type of employment (e.g., Private, Government, Self-employed, etc.).
`fnlwgt`	The final weight of the survey.
`education`	The level of education.
`education-num`	The number of years of education, in integer form.
`marital-status`	Marital status (single, married, divorced, etc.).
`occupation`	Occupation (Managerial, Clerical, Service, etc.).
`relationship`	Relationship to the household (spouse, child, other relative, etc.).
`race`	Race.
`sex`	Gender.
`capital-gain`	Income from capital (not from wages).
`capital-loss`	Loss from capital (from investments).
`hours-per-week`	The number of hours worked per week.
`native-country`	Country of origin.

The dataset also contains a target variable **income** with two values, including `<=50K` and `>50K`, representing the individual's income.

To get the dataset, please visit: Adult - UCI Machine Learning.

You probably must be familiar with this dataset through the Project 1 - Data Preprocessing. You can reuse the results of that assignment to continue developing classification algorithms, or preprocess them yourself again to suit the problem.

Students need to split the dataset into two subsets: a training set (70%) and a test set (30%) using the `train_test_split` function from the `scikit-learn` library. To ensure reproducibility, the `random_state` parameter should be set. You need to shuffle the data before splitting and split it in a stratified fashion. Other parameters (if any) should be left at their default settings.

2 Requirements

2.1 Classification algorithms implementation

Students will implement the following classification algorithms. They may use any Python library (e.g., `scikit-learn`, `tensorflow`, `keras`).

1. **Decision Tree**
   - Use a decision tree classifier with the parameter tuning for `max_depth` and `criterion` (e.g., `gini`, `entropy`).
2. **Ensemble Model: Random Forest**
   - Build an ensemble of decision trees.
   - Experiment with `n_estimators`, `max_features`, and `bootstrap`.
3. **Naive Bayes**
   - Implement Gaussian Naive Bayes (or any other variants like Bernoulli or Multinomial if applicable to the dataset).
   - Experiment with `var_smoothing` for Gaussian Naive Bayes or `alpha` for Bernoulli or Multinomial Naive Bayes.
4. **Support Vector Machine (SVM)**
   - Train an SVM model with different kernels (`linear`, `poly`, `rbf` or `sigmoid`).
   - Experiment parameter tuning with `C` or `gamma`.
5. **Neural Network: Multilayer Perceptron (MLP)**
   - Build a feedforward neural network.
   - Experiment with `hidden_layer_sizes`, `activation functions`, and `learning rates`.

To perform hyperparameter optimization for each algorithm to identify the best parameters:

- You can research for the best parameters online, manual tuning, or for more professional, use **grid search** or **random search** with cross-validation.
- Document the parameters tested and the best combination found.

2.2 Evaluation

Evaluate each model using the following metrics, categorized by type:

- **Performance Overview:**
  - **Confusion Matrix**: Visualize using a heatmap for a detailed summary of predictions.
  - **Accuracy**: Overall percentage of correct predictions.
- **Metrics for Positive and Negative Classes:**
  - **Precision**: Proportion of correctly predicted positives out of all predicted positives.
  - **Recall (Sensitivity or True Positive Rate)**: Proportion of actual positives correctly identified.
  - **Specificity (True Negative Rate)**: Proportion of actual negatives that correctly identified.
  - **F1 Score**: Harmonic mean of Precision and Recall, balancing their trade-off.
- **Model Discrimination Ability:**
  - **ROC Curve and AUC Score**: Plot ROC curves for all models on the same graph and compute the Area Under the Curve (AUC) to compare performance.

Use these metrics to analyze the trade-offs between different algorithms.

2.3 Comparison and Analysis

After implementing and evaluating all classification algorithms, analyze their performance across all metrics and provide detailed insights:

- Compare on evaluation metrics, such as accuracy, etc., to identify the best-performing model. Use the confusion matrix to analyze misclassifications and discuss potential behind reason (e.g., class imbalance).
- Compare runtime and scalability, noting trade-offs between speed and prediction quality.
- Recommend the best model(s) based on evaluation metrics, computational cost, and dataset characteristics.

3 Report (Jupyter Notebook)

The source code, result will be reported in a Jupyter Notebook with the following requirements:

- Student information (Student ID, full name, etc.).
- Self-evaluation of the assignment requirements.
- Detailed explanation of each step. Illustrative images, diagrams and equations are required.
- Each processing step must be fully commented, and results should be printed for observation.
- The report needs to be well-formatted.
- Before submitting, re-run the notebook (`Kernel` → `Restart & Run All`).
- References (if any).

4 Assessment

No.	Details	Score
1	Classification algorithms implementation.	50%
2	Evaluation.	25%
3	Comparison and Analysis.	25%

5 Notices

Please pay attention to the following notices:

- This is an **INDIVIDUAL** assignment.
- Duration: about 2 weeks.
- Any plagiarism, any tricks, or any lie will have a 0 point for the course grade.

The end.