NYCU Introduction to Machine Learning, Homework 3

111550129, 林彥亨

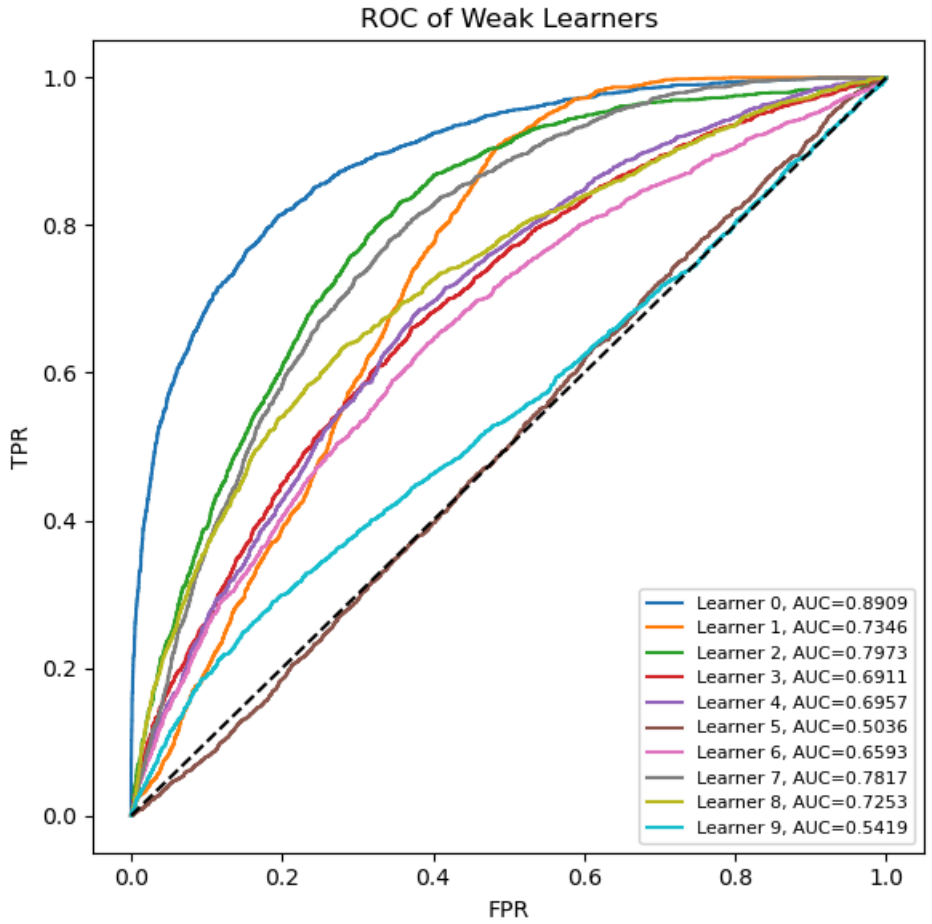
Part. 1, Coding (60%):

(20%) Adaboost

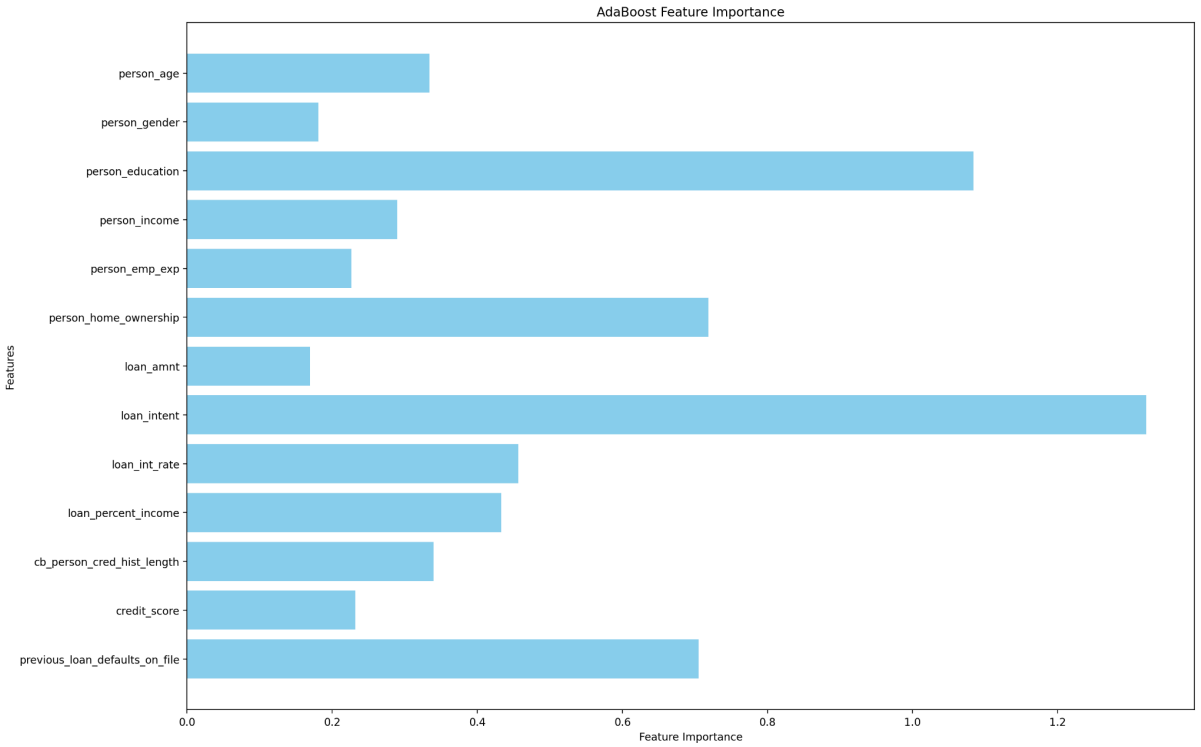
1. (5%) Show your accuracy of the testing data (n_estimators = 10)
   num_epochs = 700 learning_rate = 0.002

```
2025-11-16 21:30:12.354 | INFO     | __main__:main:68 - AdaBoost - Accuracy: 0.8579
```

2. (5%) Plot the AUC curves of each weak classifier.



3. (10%)
   a. Plot the feature importance of the AdaBoost method.



b. Paste the snapshot of your implementation of the feature importance estimation.

```python
def compute_feature_importance(self) -> t.Sequence[float]:
    """
    TODO: Implement the feature importance calculation
    """

    # feature_importance_j = sum_t alpha_t * |w_j^(t)|
    if not self.alphas:
        return []
    n_features = self.input_dim
    importance = np.zeros(n_features, dtype=np.float64)

    for alpha, learner in zip(self.alphas, self.learners):
        # 1-layer linear
        w = learner.linear.weight.detach().numpy().reshape(-1)  # (D,)
        importance += alpha * np.abs(w)

    return importance.tolist()
```

c. Explain how you compute the feature importance for the Adaboost method shortly (within 100 words as possible)

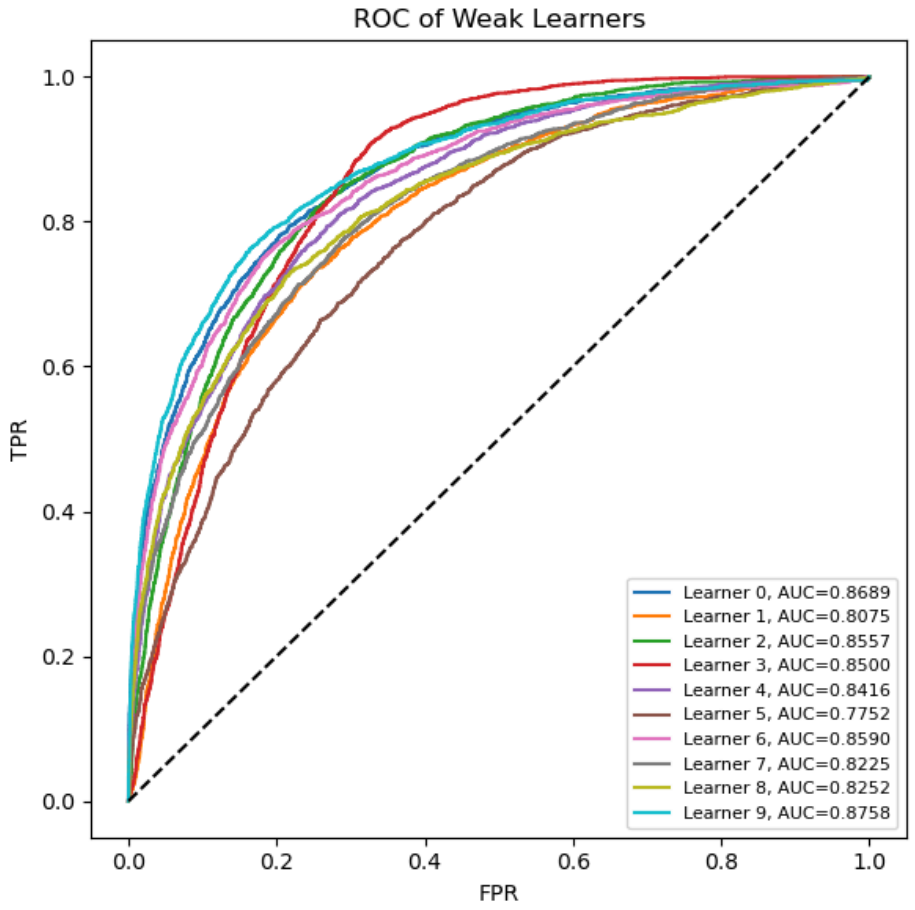
In Adaboost, each weak classifier is a linear model with weight vector **w**. After training, every learner obtains a coefficient **α**□, which reflects its performance. The feature importance is computed by combining all weak learners. That is, for each feature j, we take its absolute weight in each weak classifier, multiply by the classifier's α□, and sum over all learners.

(20%) Bagging

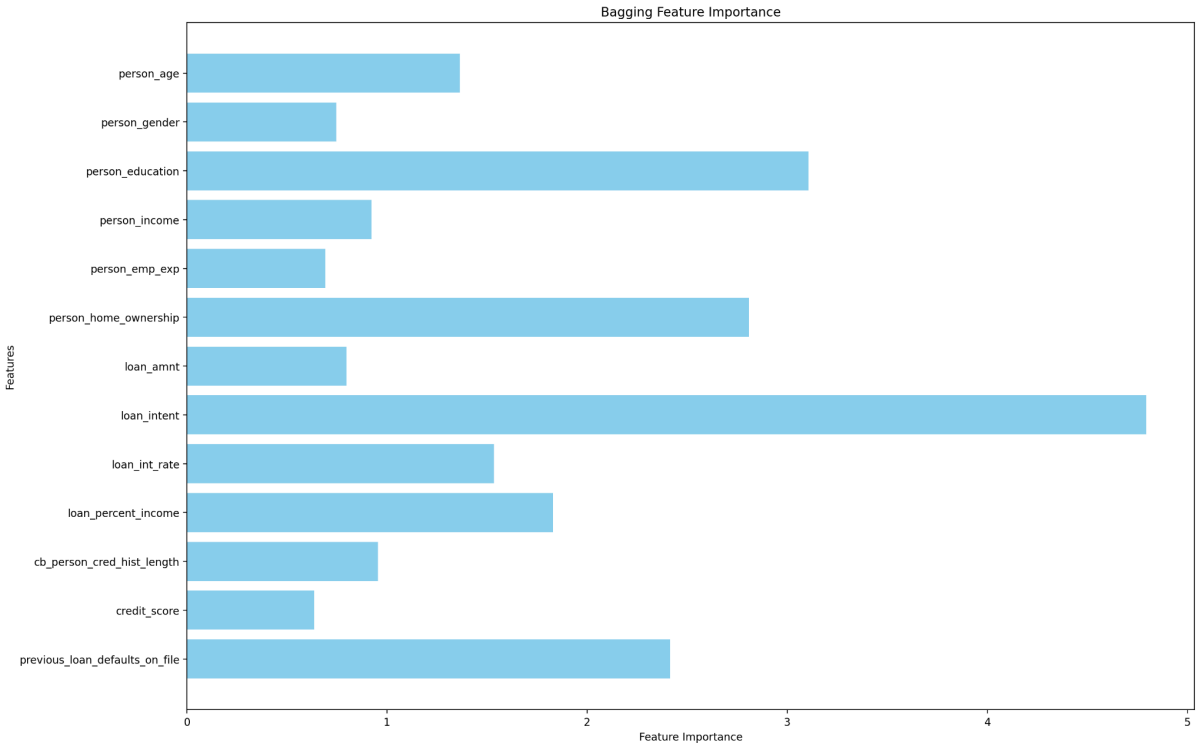
4. (5%) Show the accuracy of the test data using 10 estimators. (n_estimators=10) num_epochs = 700, learning_rate = 0.002

```
2025-11-16 21:32:01.693 | INFO     | __main__:main:97 - Bagging - Accuracy: 0.8650
```

5. (5%) Plot the AUC curves of each weak classifier.



6. (10%)
    a. Plot the feature importance of the Bagging method.



b. Paste the snapshot of your implementation of the feature importance estimation.

```python
def compute_feature_importance(self) -> t.Sequence[float]:
    """
    TODO: Implement the feature importance calculation
    """
    # feature_importance_j = sum_t |w_j^(t)|
    n_features = self.input_dim
    importance = np.zeros(n_features, dtype=np.float64)

    for learner in self.learners:
        # 1-layer linear
        w = learner.linear.weight.detach().cpu().numpy().reshape(-1)  # (D,)
        importance += np.abs(w)

    return importance.tolist()
```

c. Explain how you compute the feature importance for the Bagging method shortly (within 100 words as possible)

In Bagging, each weak classifier is trained on a bootstrap sample. Since every learner is a linear model, we use its weight vector **w** to measure feature contribution. After training all learners, the feature importance is computed by averaging the absolute weights across all models.

(15%) Decision Tree

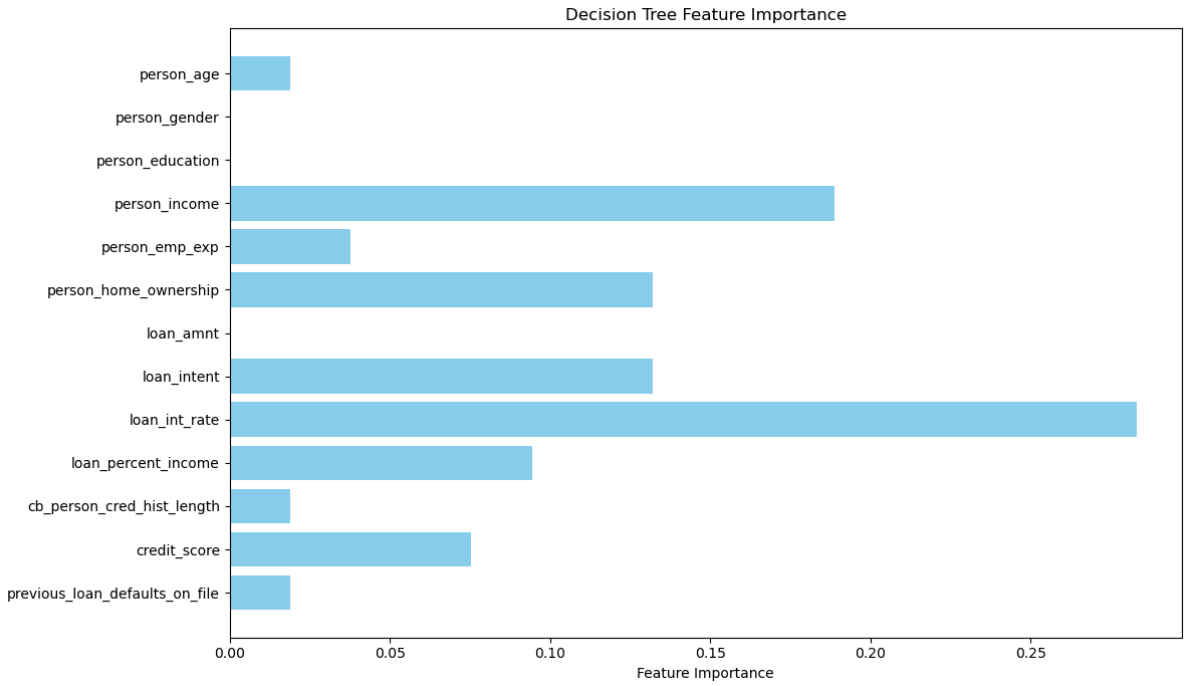
7. (5%) Compute the Gini index and the entropy of the array [0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 1].

```
2025-11-16 23:58:38.648 | INFO     | __main__:main:125 - Gini index of arr = 0.495868
2025-11-16 23:58:38.648 | INFO     | __main__:main:126 - Entropy of arr = 0.994030
```

8. (5%) Show your accuracy of the testing data with a max-depth = 7

```
2025-11-16 23:58:38.648 | INFO     | __main__:main:119 - DecisionTree - Accuracy: 0.9147
```

9. (5%) Plot the feature importance of the decision tree.



(5%) Code Linting

10. Show the snapshot of the flake8 linting result (paste the execution command even when there is no error).

```
(ml) heng@heng:~/Desktop/ml_hw3$ flake8 main.py
(ml) heng@heng:~/Desktop/ml_hw3$
```

Part. 2, Questions (40%):

1. (15%)

In the AdaBoost algorithm, each selected weak classifier, h_t is assigned a weight:

$$\alpha_t = \frac{1}{2} ln \frac{1-\epsilon_t}{\epsilon_t}$$

(a) What is the definition of ϵ_t in this formula?

(b) If a weak classifier h_t performs only as well as "random guessing" (i.e., $\epsilon_t = 0.5$), what will α_t be?

(c) Based on your answer in (b), briefly explain how this α_t weight formula ensures the robustness of AdaBoost.

Ans:

(a) ϵ_t represents the weighted error rate of the t - th weak classifier, h_t.

(b) If $\epsilon_t = 0.5$, then

$$\alpha_t = \frac{1}{2} ln(\frac{1-\epsilon_t}{\epsilon_t}) = \frac{1}{2} ln(\frac{1-0.5}{0.5}) = \frac{1}{2} ln(\frac{0.5}{0.5}) = \frac{1}{2} ln(1)$$

we can get $\alpha_t = 0$

(c) As we calculated in (b), when a weak classifier's performance is no better than random guessing $\epsilon_t = 0.5$, its weight α_t becomes 0. Then the AdaBoost automatically ignores classifiers that have no predictive power, ensuring that only classifiers with real predictive ability contribute to the model, which improves the robustness and accuracy of the overall "strong classifier."

2. (15%) Random Forest is often considered an improvement over Bagging (Bootstrap Aggregating) when used with Decision Trees as the base learners.

(a) Briefly explain the potential problem that can exist among the T classifiers (C_1, ..., C_T) produced by Bagging when using decision trees.

(b) To mitigate the problem from (a), Random Forest introduces a second source of randomness in addition to Bagging. Specifically describe what this second source of randomness is and how it works.

Ans:

(a) The potential problem is high correlation among the T trees.

(b) The second source of randomness is “Random input vector”.At each decision node, best split is chosen from a random sample of *m* attributes/features instead of all attributes.

3. (10%) Is it always possible to find the smallest decision tree that codes a dataset with no error in polynomial time? If not, what algorithm do we usually use to search a decision tree in a reasonable time? Also, list the criterion that we stop splitting the decision tree and leaf nodes are created (List two).

Ans:

(a) No, the problem is NP-complete.
(b) Greedy algorithms based on local search.
(c) This data subset is pure and Maximum tree depth is reached.