NYCU Introduction to Machine Learning, Homework 3

111550088, 張育維

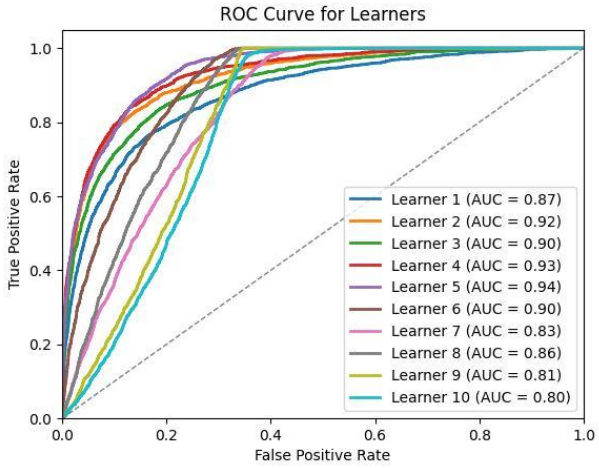
Part. 1, Coding (60%):

(20%) Adaboost

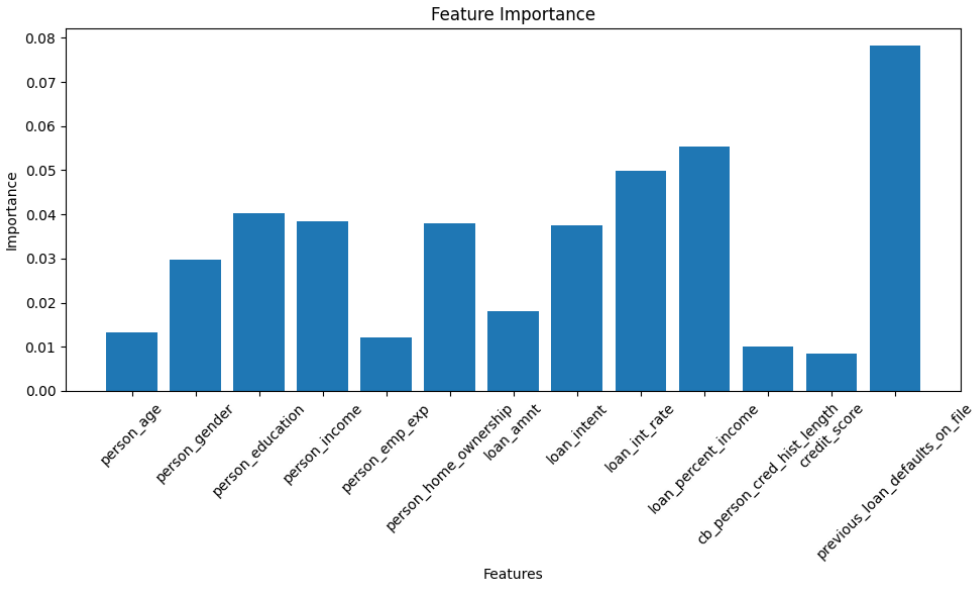
1. (10%) Show your accuracy of the testing data (n_estimators = 10)

```
2024-11-19 17:14:56.760 | INFO     | __main__:main:52 - AdaBoost - Accuracy: 0.8904
```

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



3. (5%) Plot the feature importance of the AdaBoost method. Also, you should snapshot the implementation to calculate the feature importance.



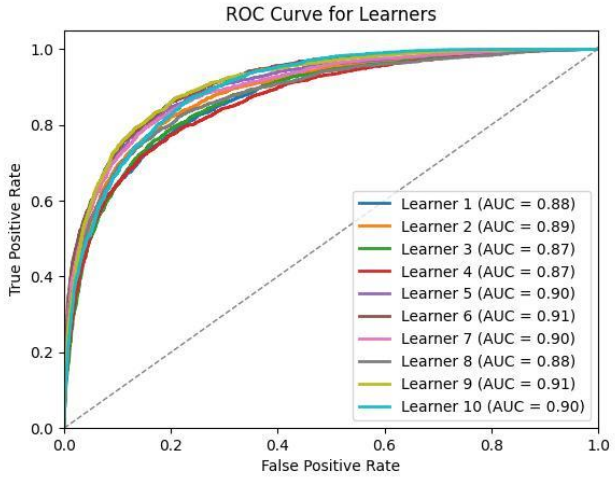
```python
def compute_feature_importance(self) -> t.Sequence[float]:
    """Implement your code here"""
    n_features = self.learners[0].fc1.in_features
    feature_importance = torch.zeros(n_features, dtype=torch.float32)

    for alpha, learner in zip(self.alphas, self.learners):
        with torch.no_grad():
            weights = learner.fc1.weight.abs().sum(dim=0)
            feature_importance += alpha * weights
    feature_importance /= feature_importance.sum()
    return feature_importance.tolist()
    # raise NotImplementedError
```

(20%) Bagging

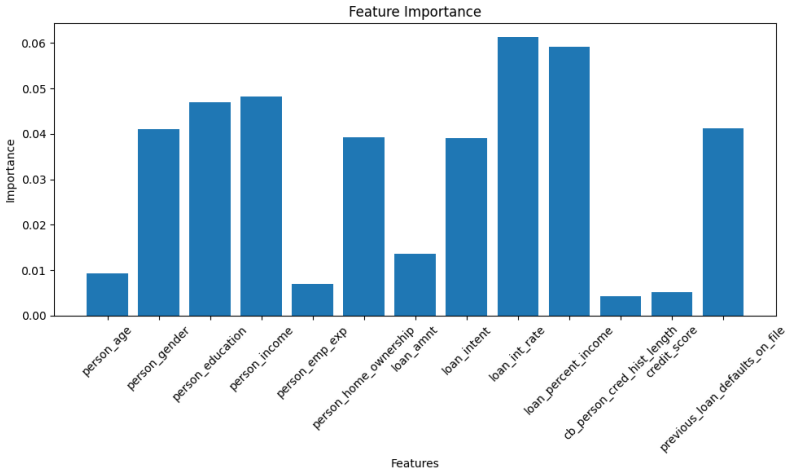
4. (10%) Show your accuracy of the testing data with 10 estimators. (n_estimators=10)

```
2024-11-19 17:17:40.432 | INFO     | __main__:main:85 - Bagging - Accuracy: 0.8288
```

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



6. (5%) Plot the feature importance of the Bagging method. Also, you should snapshot the implementation to calculate the feature importance.



```python
def compute_feature_importance(self) -> t.Sequence[float]:
    """Implement your code here"""

    n_features = self.learners[0].fc1.in_features
    feature_importance = torch.zeros(n_features, dtype=torch.float32)

    for learner in self.learners:
        with torch.no_grad():
            weights = learner.fc1.weight.abs().sum(dim=0)
            feature_importance += weights

    feature_importance = feature_importance / feature_importance.sum()
    return feature_importance.tolist()

    # raise NotImplementedError
```

(15%) Decision Tree

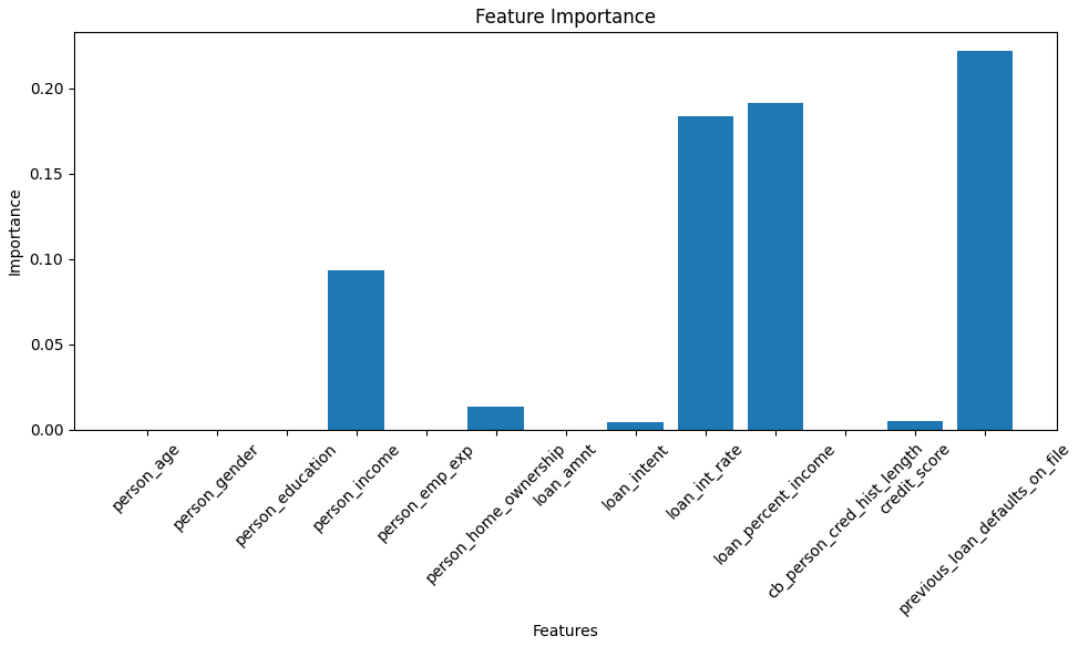
7. (5%) Compute the Gini index and the entropy of the array [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1].

```
Gini Index: 0.4628
Entropy: 0.9457
```

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

```
2024-11-19 20:50:34.762 | INFO     | __main__:main:119 - DecisionTree - Accuracy: 0.9013
```

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).

```
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 utils.py
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 ../main.py
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 decision_tree.py
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 bagging.py
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 adaboost.py
(machine_learning) D:\user\Desktop\for_school\machine_learning\hw3\src>flake8 utils.py
```

Part. 2, Questions (40%):

1. (10%) What are Bagging and Boosting, and how are they different? Please list their difference and compare the two methods in detail.

   Bagging: Train multiple models independently on different subsets of data, and aggregate their predictions by majority voting(if the problem is classification).
   Boosting: Train multiple models sequentially, which means that each model tries to correct the errors of its predecessor.
   Different: First is that bagging's models are trained independently, but the models in boosting will affect another one. Second, Second, in bagging the final prediction is made by averaging across all models. However, in boosting, predictions combine with the weighted sum, which depends on its accuracy.

2. (15%) What criterion do we use to decide when we stop splitting while performing the decision tree algorithm? Please list at least three criteria.

   Maximum depth of the tree: By the pre-defined variable, we can control the max-depth of the tree. This can control the complexity of the tree and prevents overfitting.
   Minimum number of samples per leaf node: Splitting stops if the number of samples in a leaf node fall below a specified threshold after the split. This can ensure that leaf nodes contain a sufficient number of samples, avoid overly specific splits that lead to overfitting.
   Minimum information gain: It evaluates the improvement in the quality of a split, such as decrease in Gini impurity or entropy, and stops if the improvement is below a specified threshold. This can prevent further splits that don't significantly enhance the predictive power of the tree.

3. (15%) A student claims to have a brilliant idea to make random forests more powerful: since random forests prefer trees which are diverse, i.e., not strongly correlated, the student proposes setting m = 1, where m is the number of random features used in each node of each decision tree. The student claims that this will improve accuracy while reducing variance. Do you agree with the student's claims? Clearly explain your answer.

I disagree the student’s claims because his claim is not correct. First, although setting m=1 would maximize diversity among the trees, it would reduce their quality, which would lead to a decrease of overall accuracy. The reduction in variance achieved by averaging many weak trees cannot compensate for their reduced predictive power. Therefore, I don’t agree the student’s claim.