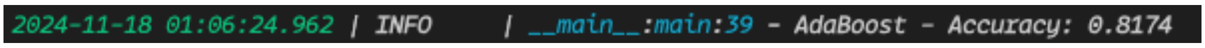
NYCU Introduction to Machine Learning, Homework 3

111550035, 蔡昀錚

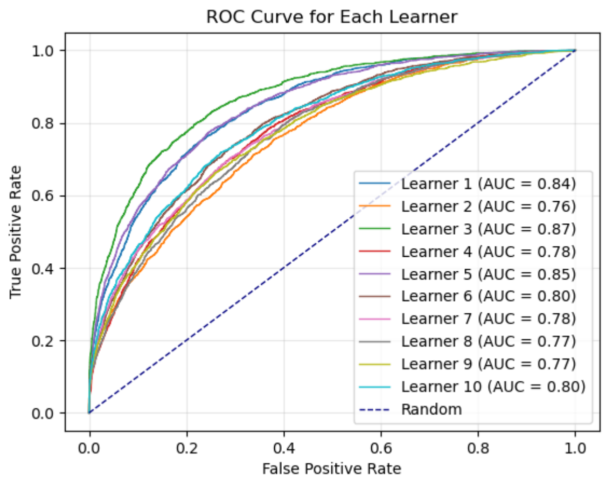
Part. 1, Coding (60%):
(20%) Adaboost

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

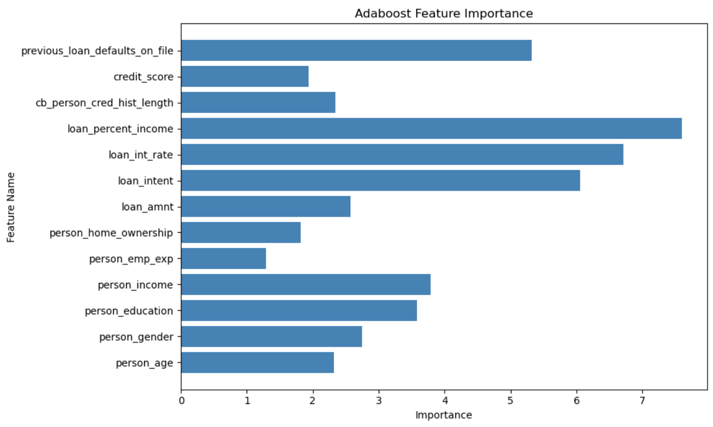


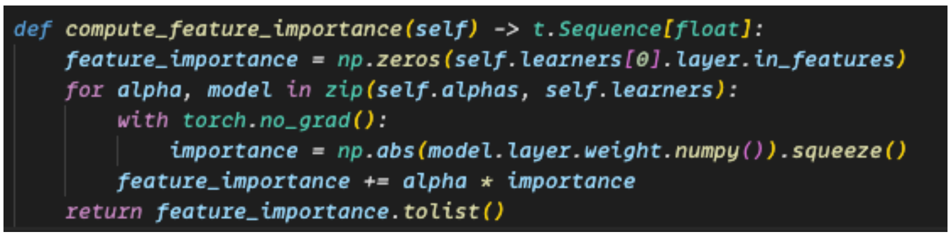
```
2024-11-18 01:06:24.962 | INFO     | __main__:main:39 - AdaBoost - Accuracy: 0.8174
```

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.

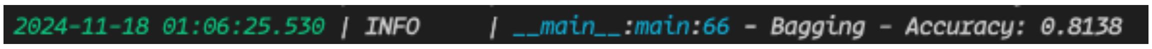




```python
def compute_feature_importance(self) -> t.Sequence[float]:
    feature_importance = np.zeros(self.learners[0].layer.in_features)
    for alpha, model in zip(self.alphas, self.learners):
        with torch.no_grad():
            importance = np.abs(model.layer.weight.numpy()).squeeze()
        feature_importance += alpha * importance
    return feature_importance.tolist()
```

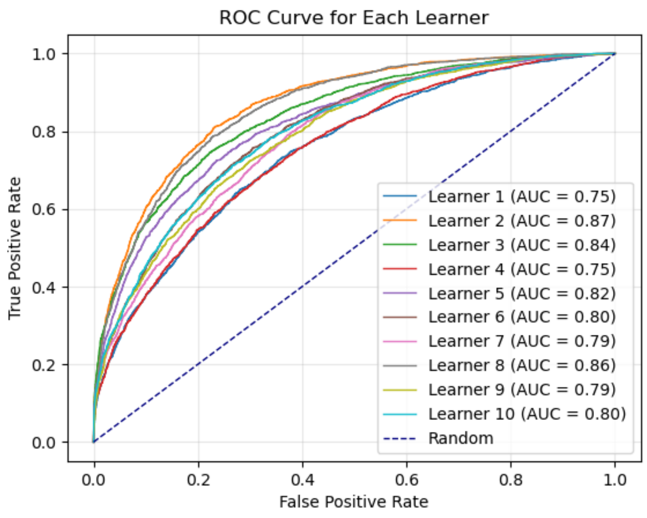
(20%) Bagging

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

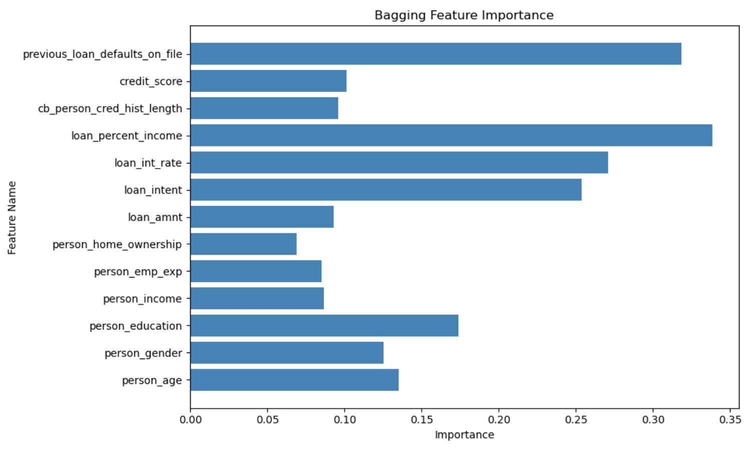


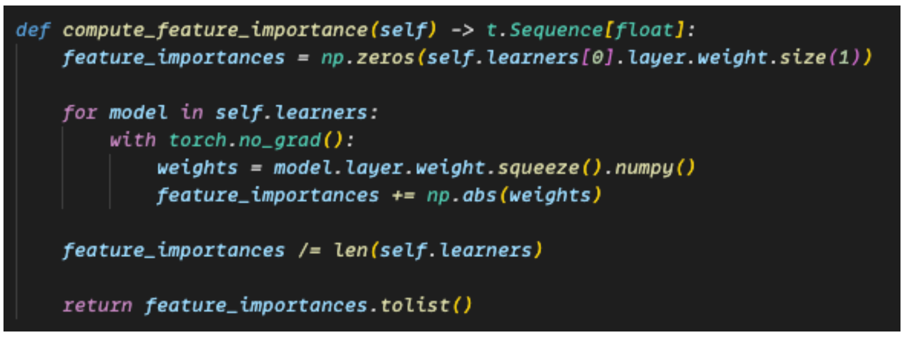
```
2024-11-18 01:06:25.530 | INFO     | __main__:main:66 - Bagging - Accuracy: 0.8138
```

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.

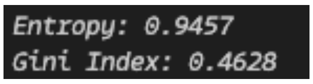




```python
def compute_feature_importance(self) -> t.Sequence[float]:
    feature_importances = np.zeros(self.learners[0].layer.weight.size(1))

    for model in self.learners:
        with torch.no_grad():
            weights = model.layer.weight.squeeze().numpy()
            feature_importances += np.abs(weights)

    feature_importances /= len(self.learners)

    return feature_importances.tolist()
```

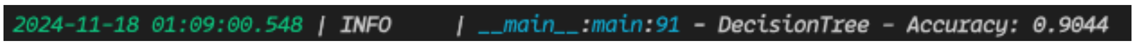
(15%) Decision Tree

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



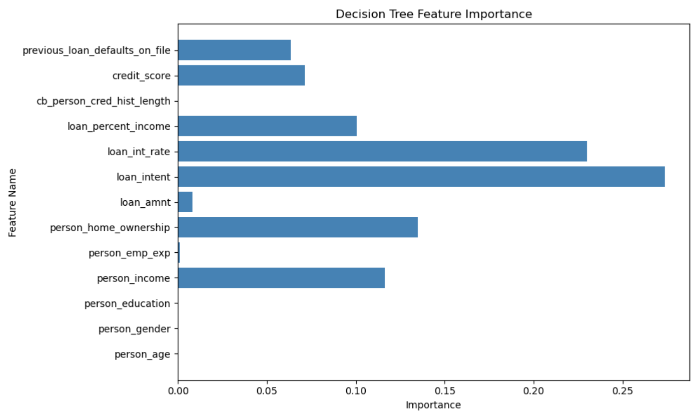
```
Entropy: 0.9457
Gini Index: 0.4628
```

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



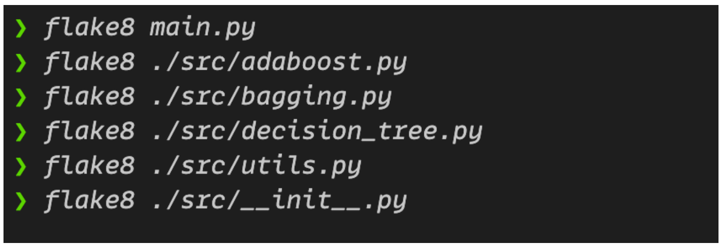
```
2024-11-18 01:09:00.548 | INFO     | __main__:main:91 - DecisionTree - Accuracy: 0.9044
```

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).



```
❯ flake8 main.py
❯ flake8 ./src/adaboost.py
❯ flake8 ./src/bagging.py
❯ flake8 ./src/decision_tree.py
❯ flake8 ./src/utils.py
❯ flake8 ./src/__init__.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.

Difference	Bagging	Boosting
Training	Parallel training each weak classifier	Sequential training each weak classifier
Goal	Reduce variance, prevent overfitting	Reduce bias and ensemble weak learners
Sample weight	Uniform	Adjust weight based on correction

In summary, Bagging aims to reduce variance and overfitting by aggregating multiple independent models, while Boosting builds a strong predictive model by focusing on correcting errors of previous models in a sequential manner.

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.

(1) Maximum Depth of the Tree:
The tree stops growing when a pre-specified maximum depth is reached.

(2) Minimum Number of Samples per Node (Minimum Split Size):
A node is only split if it contains at least a minimum number of samples (a leaf node is created if further splitting would result in fewer sample than this minimum threshold).

(3) Minimum Information Gain (Impurity Reduction):
The splitting stops if the gain in information from a potential split is below a threshold. This ensures that further splits provide a significant contribution to model performance.

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.

No. Although increasing diversity among trees is beneficial, it helps to reduce the correlation between individual trees, which can reduces the variance of the overall model. However, diversity must be balanced with the individual strength of each tree, or the overall performance of the forest can suffer.

Also, the extreme randomness can lead to very weak trees, especially in datasets with a large number of features where only a few features strongly affect the result.

In summary, Although setting m to 1, it maximizes diversity of the forest, it is likely to make individual trees weaker. As a result, setting m to 1 may not always be a good idea.