NYCU Pattern Recognition, Homework 3

312553024 江尚軒

Part. 1, Coding (60%):

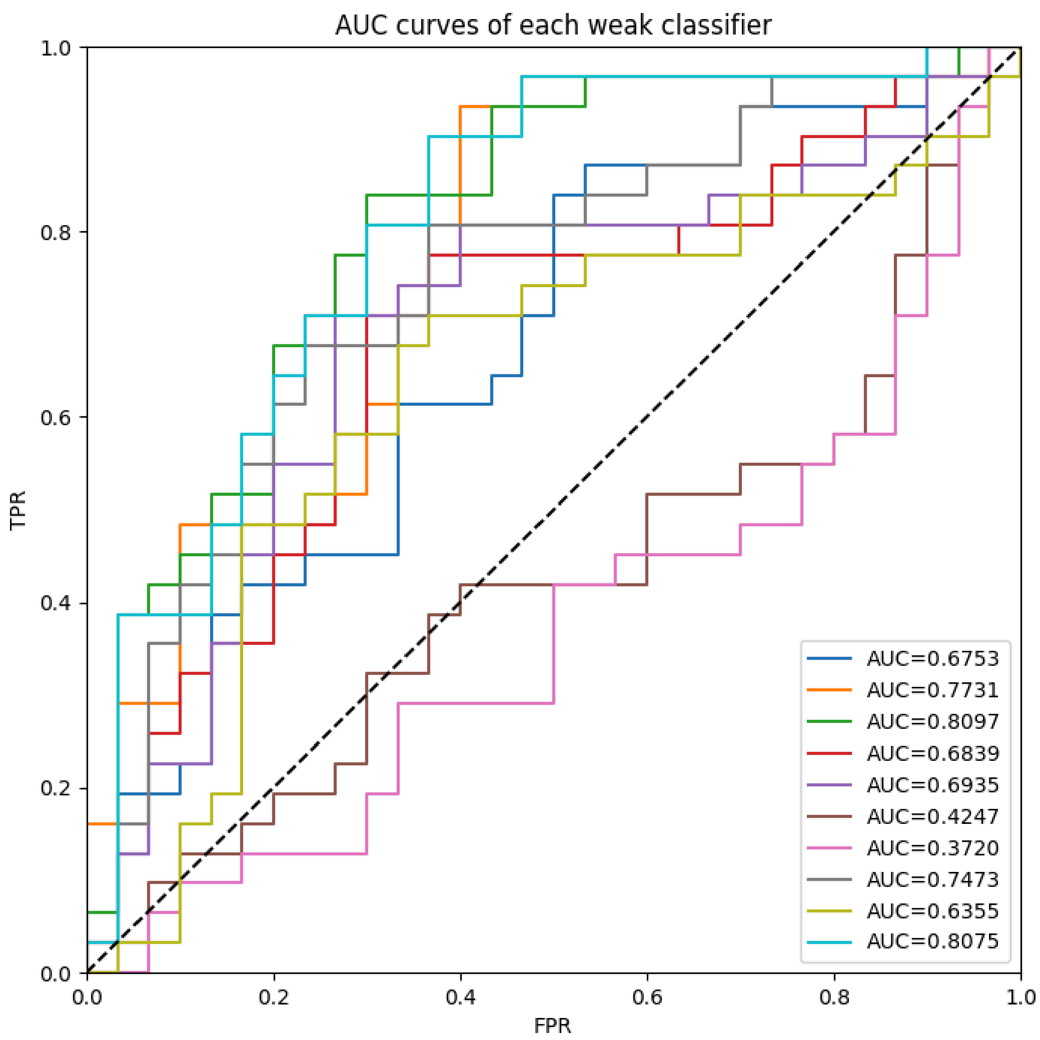
(20%) Adaboost

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

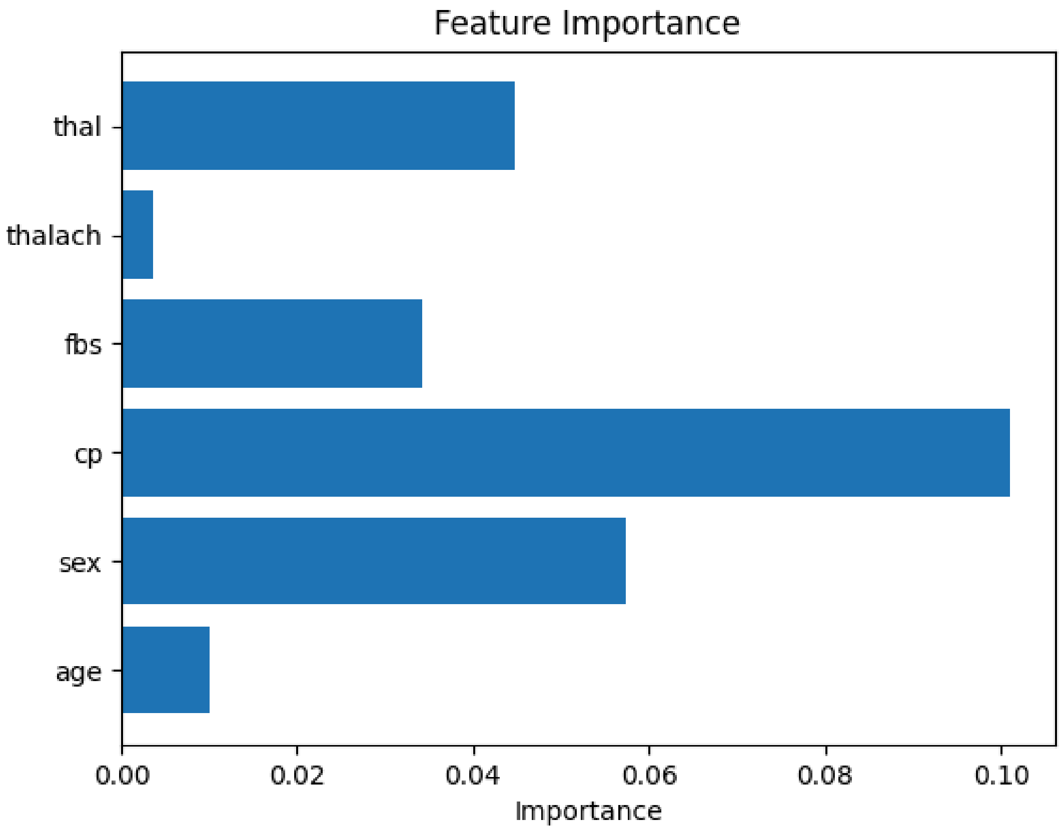
```
clf_adaboost.fit(X_train, y_train, num_epochs=1000, learning_rate=0.001)
```

```
2024-05-15 11:35:02.401 | INFO     |    main    :main:41 - AdaBoost - Accuracy: 0.7377
```

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.



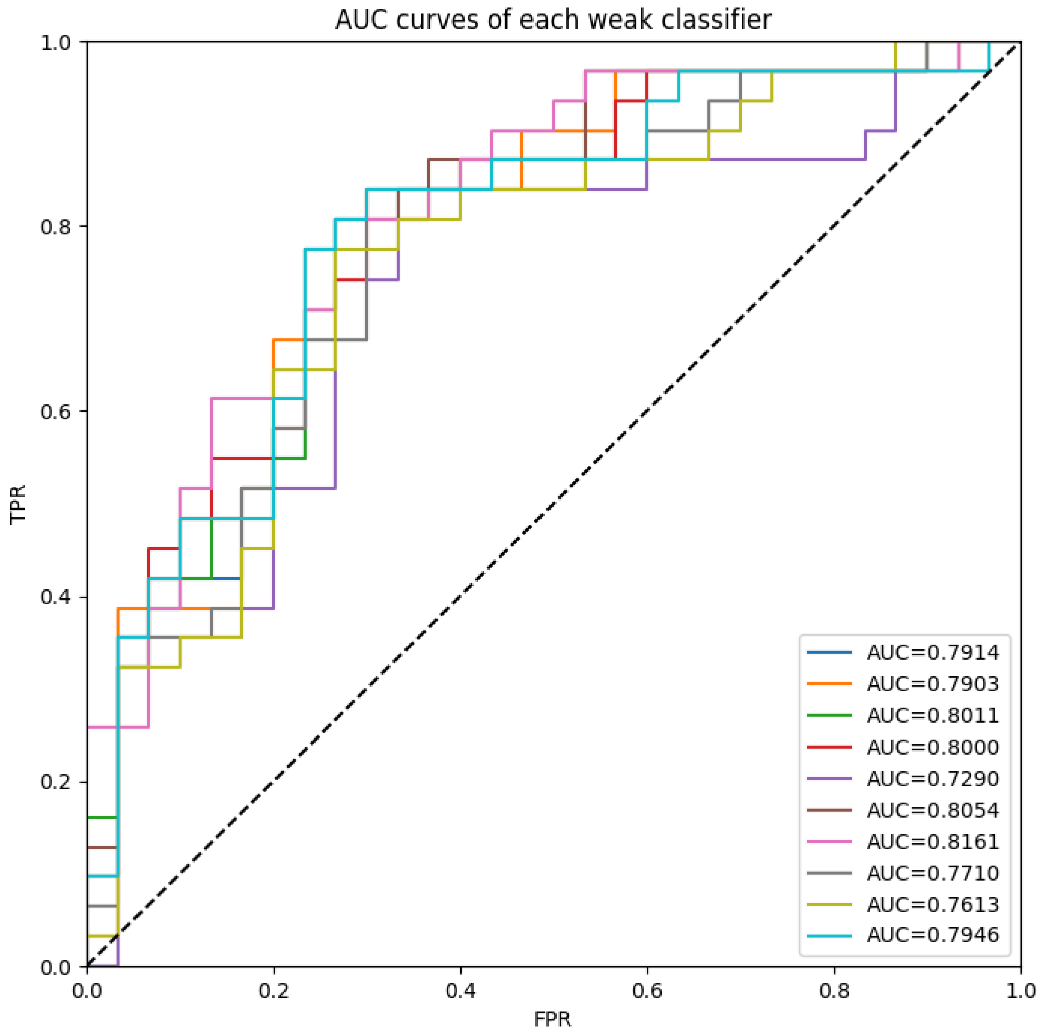
(20%) Bagging

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

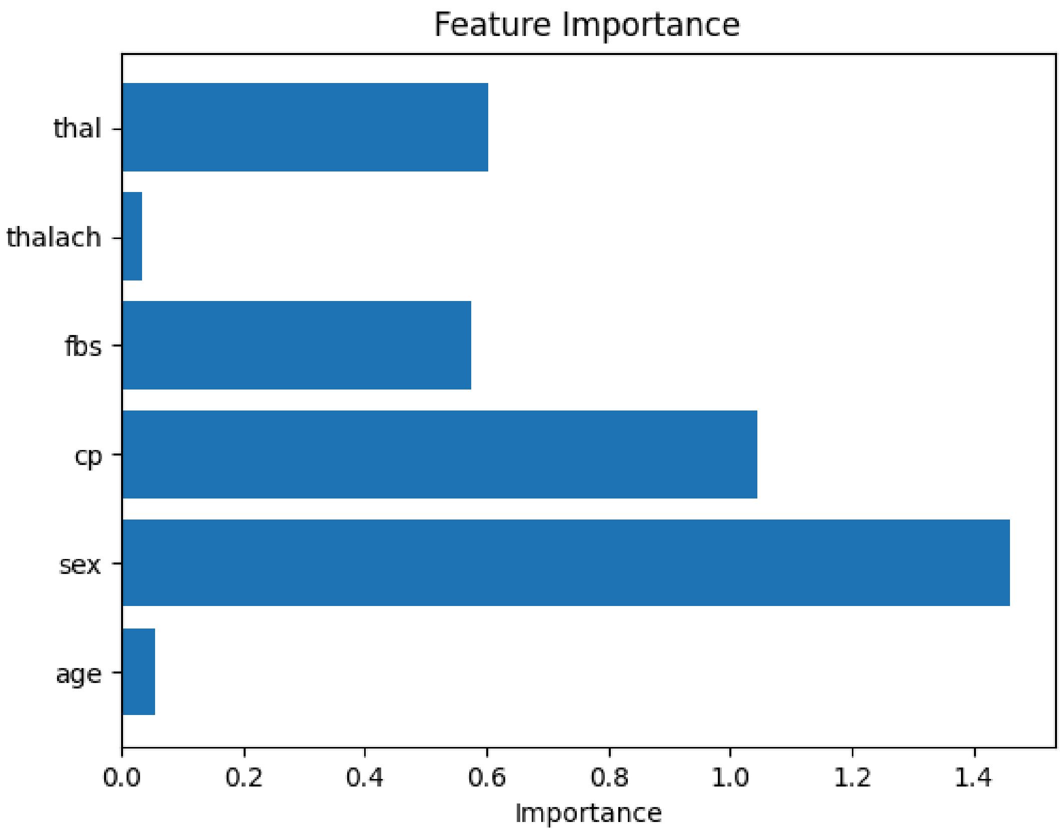
```
clf_bagging.fit(X_train, y_train, num_epochs=1000, learning_rate=0.004)
```

```
2024-05-15 12:10:18.639 | INFO     |   main   :main:49 - Bagging - Accuracy: 0.7705
```

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.



(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].

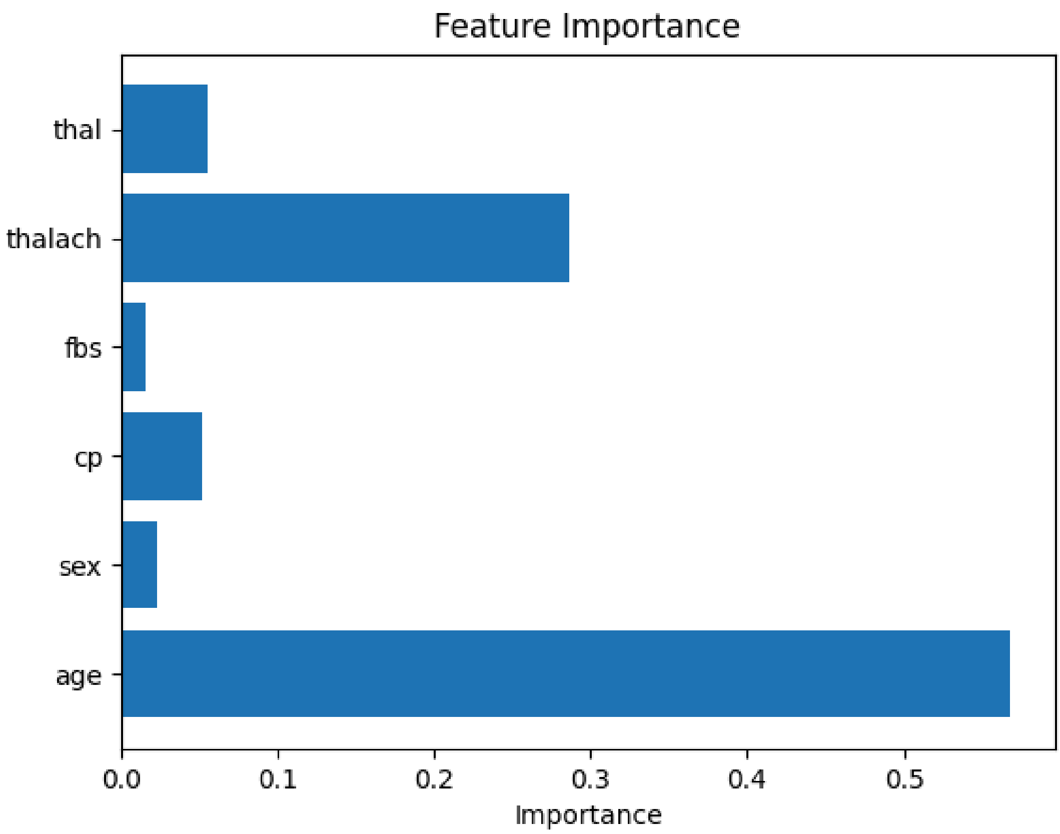
```
2024-05-15 12:10:18.803 | INFO     |   main   :main:57 - DecisionTree - Gini index: 0.4628
2024-05-15 12:10:18.803 | INFO     |   main   :main:58 - DecisionTree - Entropy: 0.9457
```

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

```
clf_tree = DecisionTree(max_depth=7)
```

```
2024-05-15 12:14:39.662 | INFO     |   main   :main:63 - DecisionTree - Accuracy: 0.7213
```

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



(5%) Code Linting

10. Show the snapshot of the flake8 linting result.

```
adsl-1-2@adsl-1-2:~/Andy/NYCU/碩一/碩一下/圖形識別/1122-pattern-recognition/Homework/HW3/code$ flake8 main.py
adsl-1-2@adsl-1-2:~/Andy/NYCU/碩一/碩一下/圖形識別/1122-pattern-recognition/Homework/HW3/code$ 
```

Part. 2, Questions (40%):

1. (10%) We have three distinct binary classifiers, and our goal is to leverage them in creating an ensemble classifier through the majority voting strategy to make decisions.
   Assuming each individual binary classifier operates independently of the others with an accuracy of 60%, what would be the accuracy of the ensemble classifier?

   To find the accuracy of an ensemble classifier using majority voting where each of three independent binary classifiers has an accuracy of 60%, you can calculate the probability that at least two classifiers are correct.

   Firstly, we count the probability of the correct and incorrect.

   Probability each classifier is correct (p): 0.6
   Probability each classifier is incorrect (q): 0.4

Secondly, we calculate the probability of two cases.

Two classifiers are correct and one is incorrect:

$$\binom{3}{2} \cdot p^2 \cdot q = 3 \cdot (0.6)^2 \cdot 0.4$$

All three classifiers are correct:

$$\binom{3}{3} \cdot p^3 = (0.6)^3$$

Finally, adding these probabilities gives the overall probability that the ensemble classifier makes a correct prediction:

$$P(\text{correct}) = 3 \cdot (0.6)^2 \cdot 0.4 + (0.6)^3 = 0.432 + 0.216 = 0.648$$

In conclusion, the accuracy of the ensemble classifier using majority voting is 64.8%. This demonstrates how combining multiple classifiers can potentially improve accuracy over individual classifiers.

2. (15%) For the decision tree algorithm, we can use the "pruning" technique to avoid overfitting. Does the random forest algorithm also need pruning?
Please explain in detail.

Pruning is not typically necessary for Random Forest algorithms, unlike in single decision trees. Here are the reasons:

- **Built-in Diversification**: Random Forests reduce the risk of overfitting through two key features:
  - **Bootstrap Sampling**: Each tree is trained on a different sample of the data.
  - **Feature Randomness**: A random subset of features is used to split nodes.
- **Averaging Reduces Overfitting**: Overfitting in individual trees is mitigated by averaging their predictions. This ensemble method effectively reduces variance without significantly increasing bias, even if the individual trees are deep and complex.
- **Control via Parameters**: Instead of pruning, Random Forests control overfitting through parameters like the number of trees, the maximum number of features considered for splits, and the minimum samples required to split a node.

3. (15%) Activation functions are core components of neural networks. They need to be differentiable to ensure backpropagation works correctly. Please calculate the derivatives of the following commonly used activation functions.
(For questions 1. and 2., consider the cases where $x > 0$ and $x \leq 0$)

1. f(x) = relu(x),	df(x)/dx = ?
2. f(x) = leaky_relu(x) with negative_slope=0.01,	df(x)/dx = ?
3. f(x) = sigmoid(x),	df(x)/dx = ?
4. f(x) = silu(x),	df(x)/dx = ?
5. f(x) = tanh(x),	df(x)/dx = ?

1. ReLU
- Function: $f(x) = \max(0, x)$
- Derivative:
  - $f'(x) = 1 \quad \text{if } x > 0$
  - $f'(x) = 0 \quad \text{if } x \leq 0$

2. Leaky ReLU
- Function: $$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0.01x & \text{if } x \leq 0 \end{cases}$$
- Derivative:
  - $f'(x) = 1 \quad \text{if } x > 0$
  - $f'(x) = 0.01 \quad \text{if } x \leq 0$

3. Sigmoid
- Function: $$f(x) = \frac{1}{1 + e^{-x}}$$
- Derivative: $f'(x) = f(x)(1 - f(x))$

4. SiLU
- Function: $f(x) = x \cdot \sigma(x)$ where $\sigma(x)$ is the sigmoid function
- Derivative: $f'(x) = f(x) + \sigma(x)(1 - f(x))$

5. Hyperbolic Tangent (tanh)
- Function: $f(x) = \tanh(x)$
- Derivative: $f'(x) = 1 - \tanh^2(x) = 1 - f(x)^2$