NYCU Pattern Recognition, Homework 3

[312552056], [鄭璟翰]
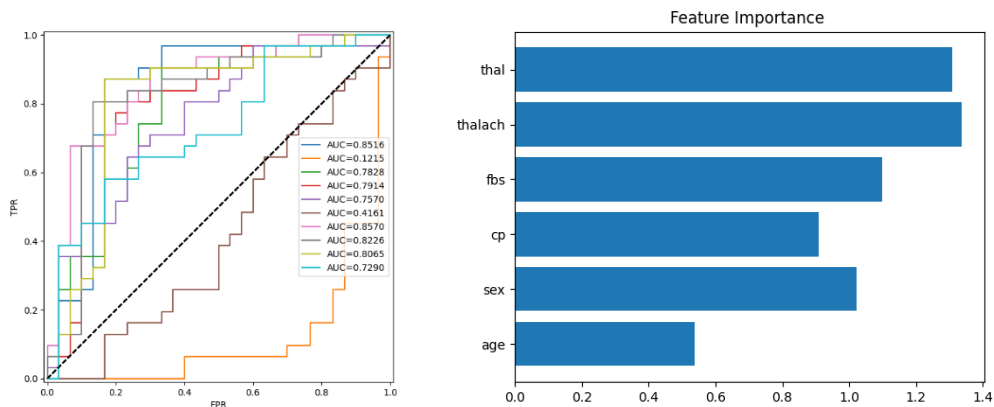
Part. 1, Coding (60%):

(20%) Adaboost

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

```
2024-05-14 20:10:08.517 | INFO     | __main__:main:38 - AdaBoost - Accuracy: 0.7705
```

2. (5%) Plot the AUC curves of <u>each</u> weak classifier.

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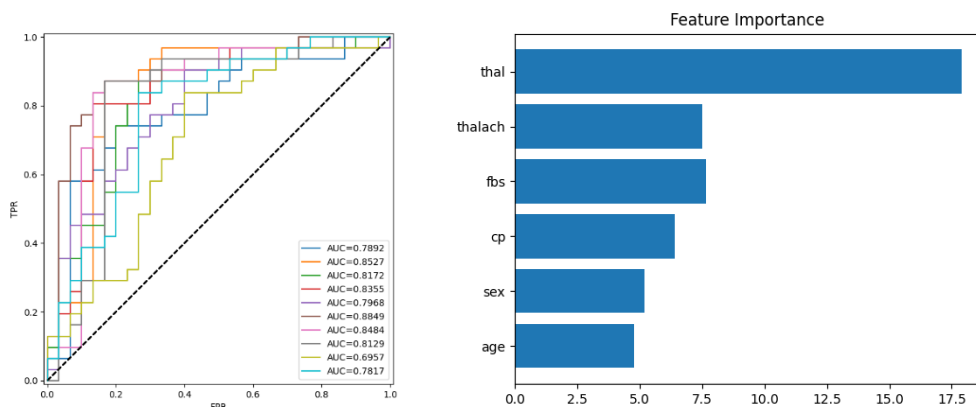


```python
def compute_feature_importance(self) -> t.Sequence[float]:
    """Implement your code here"""
    feature_impoertance = [model.layer.weight.data.abs() * a for model, a in zip(self.learners, self.alphas)]
    feature_impoertance = np.array(feature_impoertance)
    return feature_impoertance.sum(axis=0).ravel()
```

(20%) Bagging

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

```
2024-05-14 20:11:12.360 | INFO     | __main__:main:63 - Bagging - Accuracy: 0.8361
```

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"""
    feature_impoertance = [model.layer.weight.data.abs() for model in self.learners]
    feature_impoertance = np.array(feature_impoertance)
    return feature_impoertance.sum(axis=0).ravel()
```
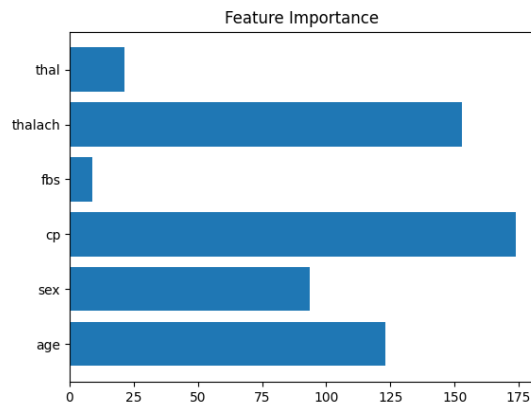
(15%) Decision Tree

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

```
2024-05-14 20:11:55.269 | INFO     | __main__:main:77 - gini index: 0.4628
2024-05-14 20:11:55.270 | INFO     | __main__:main:78 - entropy: 0.9457
```

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

```
2024-05-14 20:11:55.365 | INFO     | __main__:main:86 - 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.

```
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release$ flake8 main.py
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release$ cd src
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release/src$ flake8 adaboost.py
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release/src$ flake8 bagging.py
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release/src$ flake8 decision_tree.py
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release/src$ flake8 utils.py
(pytorch_env) zjinghan@zjinghan-VivoBook-ASUSLaptop-X512JP-X512JP:~/Documents/NYCU/Patte
• rn Recognition/HW3/release/src$ cd ..
```

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?

   In majority voting ensemble classifiers, the classification result is determined by the majority decision of individual weak classifiers. Thus, if two or more weak classifiers yield correct results, the ensemble classifier will also produce a correct result. The probability of all three weak classifiers yielding correct results is 0.6^3, while the probability of exactly two weak classifiers yielding correct results is calculated as 3 *

$C_2^3 * 0.6^2 * 0.4$. Therefore, the final accuracy of the ensemble classifier is the sum of these probabilities, equals to 64.8%.

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.
The Random Forest algorithm combines bagging and multiple decision trees. Bagging involves randomly sampling from the training data to prevent overfitting of the weak classifiers. Hence, decision trees within Random Forest do not require pruning to avoid overfitting.

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 = ?

$$1.\, f(x) = \begin{cases} x, x > 0 \\ 0, x \leq 0 \end{cases}, \frac{df(x)}{dx} = \begin{cases} 1, x > 0 \\ 0, x < 0 \\ undefined, x = 0 \end{cases}$$

$$2.\, f(x) = \begin{cases} x, x > 0 \\ 0.01, x \leq 0 \end{cases}, \frac{df(x)}{dx} = \begin{cases} 1, x > 0 \\ 0.01, x < 0 \\ undefined, x = 0 \end{cases}$$

$$3.\, f(x) = \frac{1}{1 + e^{-x}}, \frac{df(x)}{dx} = \frac{0 * (1 + e^{-x}) - 1 * (-e^{-x})}{(1 + e^{-x})^2} = \frac{e^{-x}}{(1 + e^{-x})^2}$$
$$= \frac{1}{1 + e^{-x}} * \frac{e^{-x}}{1 + e^{-x}} = f(x)(1 - f(x))$$

$$4.\, f(x) = \frac{x}{1 + e^{-x}}, \frac{df(x)}{dx} = \frac{1 * (1 + e^{-x}) - x * (-e^{-x})}{(1 + e^{-x})^2} = \frac{1}{1 + e^{-x}} + \frac{e^{-x}}{(1 + e^{-x})^2}$$

$$5.\, f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}, \frac{df(x)}{dx} = \frac{(e^x + e^{-x}) * (e^x + e^{-x}) - (e^x - e^{-x}) * (e^x - e^{-x})}{(e^x + e^{-x})^2}$$
$$= 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} = 1 - f(x)^2$$