NYCU Introduction to Machine Learning, Homework 3

110550130, 劉秉驊

Part. 1, Coding (50%):

For this coding assignment, you are required to implement the Decision Tree and Adaboost algorithms using only NumPy. After that, train your model on the provided dataset and evaluate the performance on the testing data.

(30%) Decision Tree

Requirements:

- Implement **gini index** and **entropy** for measuring the best split of the data.
- Implement the decision tree classifier (CART, Classification and Regression Trees) with the following two arguments:
- **criterion**: The function to measure the quality of a split of the data. Your model should support "gini" and "entropy".
- **max_depth**: The maximum depth of the tree. If max_depth=None, then nodes are expanded until all leaves are pure. max_depth=1 equals to splitting data once.

Tips:

- Your model should produce the same results when rebuilt with the same arguments, and there is no need to prune the trees.
- You can use the recursive method to build the nodes.

Criteria:

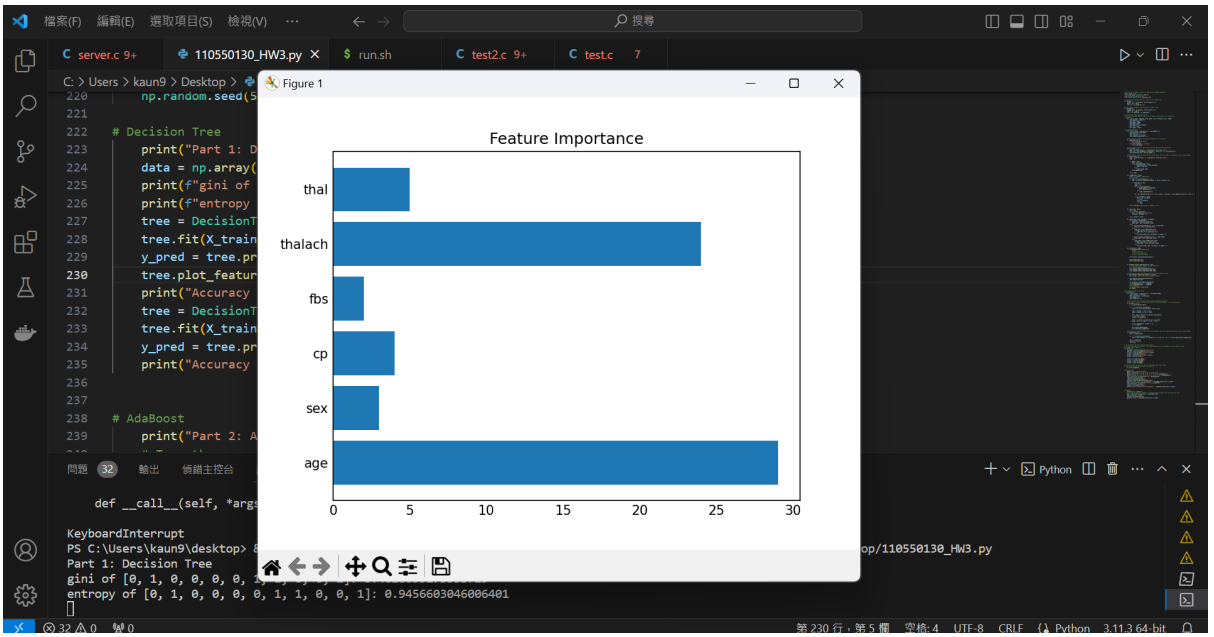
1. (5%) Compute the gini index and the entropy of the array [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1].
2. (10%) Show the accuracy score of the testing data using criterion="gini" and max_depth=7. Your accuracy score should be higher than 0.7.
3. (10%) Show the accuracy score of the testing data using criterion="entropy" and max_depth=7. Your accuracy score should be higher than 0.7.
4. (5%) Train your model using criterion="gini", max_depth=15. Plot the feature importance of your decision tree model by simply counting the number of times each feature is used to split the data. Your answer should look like the plot below:

(next page)

Answer 1 2 3 :

```
root@MSI:/mnt/c/Users/user/Desktop/Class/3up/ml/hw3# python3 110550130_HW3.py
Part 1: Decision Tree
gini of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.4628099173553719
entropy of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.9456603046006401
Accuracy (gini with max_depth=7): 0.7049180327868853
Accuracy (entropy with max_depth=7): 0.7213114754098361
Part 2: AdaBoost
Accuracy: 0.8032786885245902
root@MSI:/mnt/c/Users/user/Desktop/Class/3up/ml/hw3#
```

Answer 4 :



(20%) Adaboost

Requirements:

- Implement the Adaboost algorithm by using the decision tree classifier (max_depth=1) you just implemented as the weak classifier.
- The Adaboost model should include the following two arguments:
- **criterion**: The function to measure the quality of a split of the data. Your model should support "gini" and "entropy".
- **n_estimators**: The total number of weak classifiers.

Tips:

- You can set any random seed to make your result reproducible.

Criteria:

5. (20%) Tune the arguments of AdaBoost to achieve higher accuracy than your Decision Trees.

Points	Testing Accuracy
20 points	0.8 <= acc
15 points	0.78 <= acc < 0.8
10 points	0.76 <= acc < 0.78
5 points	**0.74 <= acc < 0.76**
0 points	**acc < 0.74**

(next page)

On last line, accuracy is 0.8032786

```
root@MSI:/mnt/c/Users/user/Desktop/Class/3up/ml/hw3# python3 110550130_HW3.py
Part 1: Decision Tree
gini of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.4628099173553719
entropy of [0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 1]: 0.9456603046006401
Accuracy (gini with max_depth=7): 0.7049180327868853
Accuracy (entropy with max_depth=7): 0.7213114754098361
Part 2: AdaBoost
Accuracy: 0.8032786885245902
root@MSI:/mnt/c/Users/user/Desktop/Class/3up/ml/hw3#
```

Part. 2, Questions (50%):

1. (10%) True or False. If your answer is false, please explain.
   a. (5%) In an iteration of AdaBoost, the weights of misclassified examples are increased by adding the same additive factor to emphasize their importance in subsequent iterations.
   b. (5%) AdaBoost can use various classification methods as its weak classifiers, such as linear classifiers, decision trees, etc.

a.
False
We increase the weights of misclassified examples by multiplying to adjust individually with each performance of the weak classifier, and then train next iteration.

b.
True

2. (10%) How does the number of weak classifiers in AdaBoost influence the model's performance? Please discuss the potential impact on overfitting, underfitting, computational cost, memory for saving the model, and other relevant factors when the number of weak classifiers is too small or too large.

Speaking of overfitting and underfitting, performance of AdaBoost depends on the strong classifier that consists of sufficient weak classifiers. If the number of weak classifiers is too small that makes a poor strong classifier hard to fit complex data, that is underfitting. If the number of weak classifiers is too large that causes overemphasize the training data, noise, and outliers so that we can't generalize new data well, that is overfitting.
Speaking of computational cost, we can see the result from few weak classifiers that means we compute little or not enough, which brings a light workload for computers but we might get an unsatisfactory result. When having a large number of weak classifiers, we may get a high computational cost that increases training time and prediction time. It combines more weak classifiers to a strong classifier.
With relative to computational cost, few weak classifiers bring light computation that also uses fewer memory to store. On the other hand, larger weak classifiers need more computation that requires more memory.
Additionally, few weak classifiers might have high bias and low variance, which is not keen to the changes of training data. And large weak classifiers might have low bias but high variance, which is keen to data and overfitting.

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 don't agree that setting m to 1 can improve the diversity for some reasons that first, if m is equal to 1, we have only 1 feature at each node making the trees become more and more similar, that is strongly correlated, which results in low effectiveness of ensemble. And second, few features makes much keen to noise and outlier, so that it performs badly to generalize new data and might cause overfitting. Thus, the smaller the number of m is, the correlated the trees are, that is not diverse, and we get poor performance of random forest model.

(next page)

4. (15%) The formula on the left is the forward process of a standard neural network while the formula on the right is the forward process of a modified model with a specific technique.
    a. (5%) According to the two formulas, describe what is the main difference between the two models and what is the technique applied to the model on the right side.
    b. (10%) This technique was used to deal with overfitting and has many different explanations; according to what you learned from the lecture, try to explain it with respect to the ensemble method.

$$z^{(l+1)} = w^{(l+1)} y^l + b^{(l+1)}$$

$$y^{(l+1)} = f(z^{(l+1)})$$

$$r^l = Bernoulli(p)$$

$$\tilde{y}^l = r^l y^l$$

$$z^{(l+1)} = w^{(l+1)} \tilde{y}^l + b^{(l+1)}$$

$$y^{(l+1)} = f(z^{(l+1)})$$

a.
On the left side is the standard neural network. And on the right side is the model using dropout technique. We multiply y with Bernoulli distribution of probability p before it enters the next layer, rather than the left side.

b.
Ensemble method generalizes several predictions of models to improve performance, and dropout can be seen as an ensemble of subnetworks that makes our model go through different paths in the network and averages the different results at test time. Thus dropout can prevent our network from overemphasizing some nodes when we are training, which reduces the sensitivity of noise and reduces overfitting