NYCU Introduction to Machine Learning, Homework 3

Part. 1, Coding (50%):

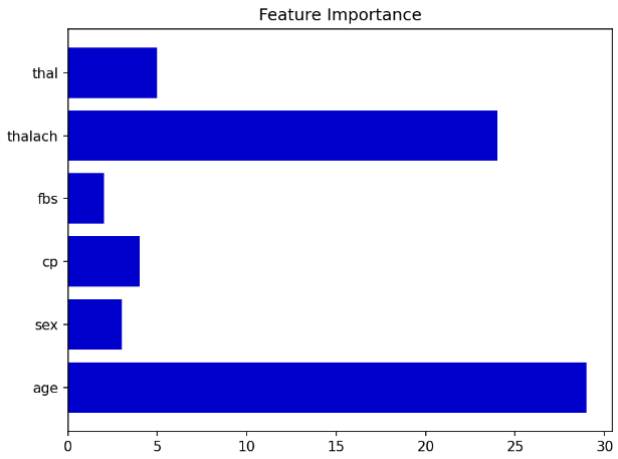
Output:

```
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: 0.8360655737704918
```

Arguments of Adaboost:

```
ada = AdaBoost(criterion='gini', n_estimators=12)
```

Feature importance:



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.

   False.

   The weights of misclassified examples are increased by multiplying the same multiplicative factor, not by adding. The formula is: $D_{t+1}(i) = \frac{D_t(i)\exp[-\alpha_t y_i h_t(x_i)]}{Z_t}$

   We see that if one example is misclassified, its $y_i h_t(x_i)$ becomes -1. Therefore, all misclassified examples in the current iteration are increased by the same multiplicative factor, $exp(\alpha_t)$.

b. (5%) AdaBoost can use various classification methods as its weak classifiers, such as linear classifiers, decision trees, etc.

True.
AdaBoost can use various classification methods as its weak classifiers. This flexibility is also an advantage of it.

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.

- too small:

The performance will be poor due to underfitting, as the number of classifiers is not enough to make effective predictions. The model fails to capture the complexity of the underlying data patterns.

- too large:

The performance will also be poor. One of the problems is overfitting, where the model performs well on the training data but fails on other general, unseen data. Other problems include high memory demand. The process needs more space to store the large number of weak classifiers, which is impractical in some resource-constrained environments. Computational cost and execution time are also problems. Training and applying numerous classifiers require more processing power and time.

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.
Using only one feature per node reduces the complexity of each decision tree. This means that the splitting decision at each node is based on a single feature,

leading to similar tree structures across the forest. As a result, the trees will not be diverse; instead, they will be highly correlated because they are learning the same underlying patterns.

For example, if a feature F is selected for a split, it will consistently lead to the same splitting result in each tree, making the trees' predictions highly similar.

In summary, setting m=1 would not improve the accuracy or reduce the variance of the random forest. Instead, it would likely have the opposite effect by reducing the necessary diversity among the trees.

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.

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

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

a. (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.

The difference is that the right model introduces randomness through the Bernoulli distribution. The variable r^l works as a mask that randomly neglect some values in y^l.

By doing so, the network will see varying inputs in each iteration, improving the model's generalization. This technique is called "Dropout".

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.

The idea of the ensemble method is to combine multiple models to improve overall performance. Since this technique randomly drops out some inputs in each iteration, it can be regarded as each model being trained with varying datasets, resulting in multiple models with different architectures.

The ensemble of predictions from these different models helps deal with overfitting because it introduces diversity. That is, each model focuses on different

patterns within the data, and the combination of their predictions contributes to a more generalized model.