HW3

Part. 1, Coding (70%):

1. (5%) Gini Index or Entropy is often used for measuring the "best" splitting of the data. Please compute the Entropy and Gini Index of this array `np.array([1,2,1,1,1,1,2,2,1,1,2])` by the formula below. (More details on page 5 of the hw3 slides, 1 and 2 represent class1 and class 2, respectively)

```
print("Gini of data is ", gini(data))
```
✓ 0.6s

```
Gini of data is  0.46280991735537190
```

```
print("Entropy of data is ", entropy(data))
```
✓ 0.7s

```
Entropy of data is  0.9456603046006401
```

2. (10%) Implement the Decision Tree algorithm (CART, Classification and Regression Trees) and train the model by the given arguments, and print the accuracy score on the test data. You should implement two arguments for the Decision Tree algorithm,

1. **Criterion**: The function to measure the quality of a split. Your model should support "gini" for the Gini impurity and "entropy" for the information gain.
2. **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 split data once

2.1. Using Criterion='gini', showing the accuracy score of test data by Max_depth=3 and Max_depth=10, respectively.

```
clf_depth3_pred = clf_depth3.predict(x_test)
clf_depth10_pred = clf_depth10.predict(x_test)
print(accuracy_score(y_test, clf_depth3_pred))
print(accuracy_score(y_test, clf_depth10_pred))
```
✓ 0.1s

```
0.92
0.9233333333333333
```

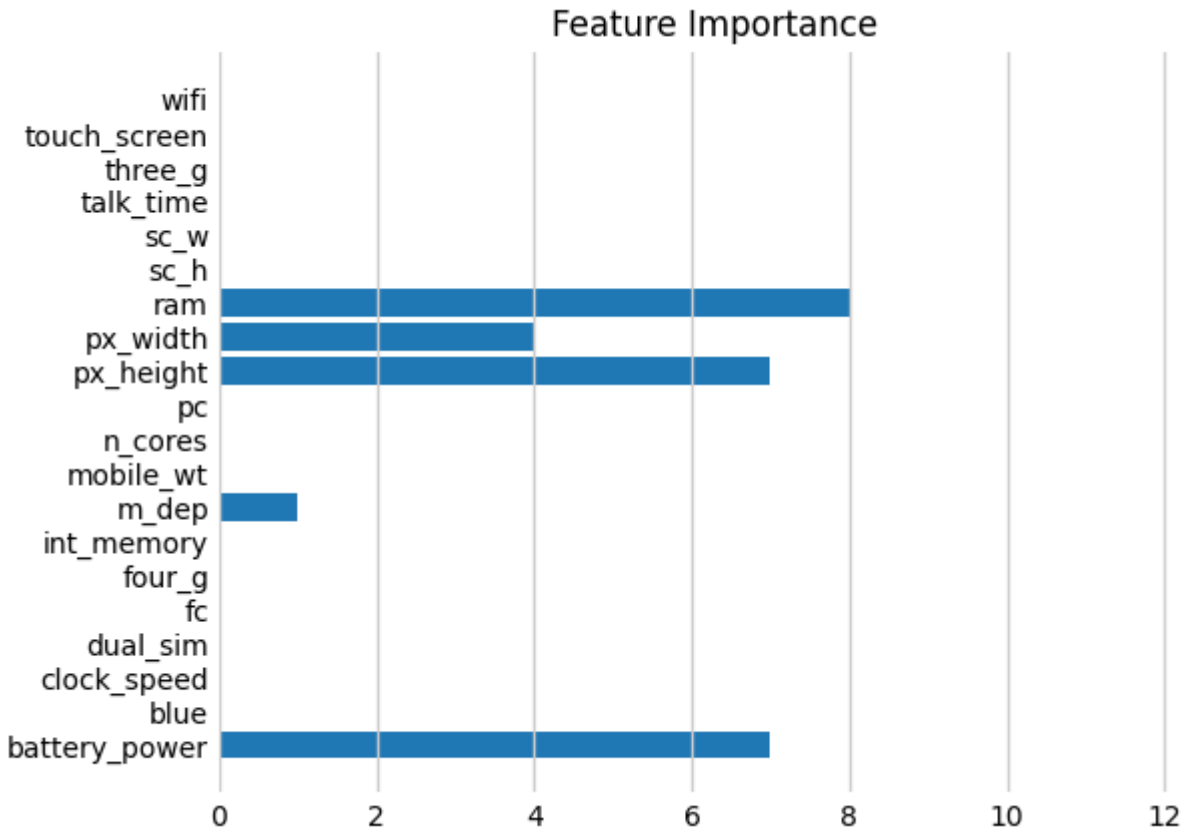
2.2. Using Max_depth=3, showing the accuracy score of test data by Criterion='gini' and Criterion='entropy', respectively.

```
clf_gini_pred = clf_gini.predict(x_test)
clf_entropy_pred = clf_entropy.predict(x_test)
print(accuracy_score(y_test, clf_gini_pred))
print(accuracy_score(y_test, clf_entropy_pred))
```

✓ 0.7s

```
0.92
0.9333333333333333
```

3. (5%) Plot the [feature importance] of your Decision Tree model. You can use the model from Question 2.1, max_depth=10. (You can use simply counting to get the feature importance instead of the formula in the reference, more details on the sample code. Matplotlib is allowed to be used)



4. (15%) Implement the AdaBoost algorithm by using the CART you just implemented from question 2. You should implement one argument for the AdaBoost.

1. **N_estimators**: The number of trees in the forest.

4.1. Showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.

```
clf_n_estimators10_pred = clf_n_estimators10.predict(x_test)
clf_n_estimators100_pred = clf_n_estimators100.predict(x_test)
print(accuracy_score(y_test, clf_n_estimators10_pred))
print(accuracy_score(y_test, clf_n_estimators100_pred))
```

✓ 0.1s

```
0.9533333333333334
0.9566666666666667
```

5. (15%) Implement the Random Forest algorithm by using the CART you just implemented from question 2. You should implement three arguments for the Random Forest.

1. **N_estimators**: The number of trees in the forest.
2. **Max_features**: The number of features to consider when looking for the best split
3. **Bootstrap**: Whether bootstrap samples are used when building trees

5.1. Using Criterion='gini', Max_depth=None, Max_features=sqrt(n_features), Bootstrap=True, showing the accuracy score of test data by n_estimators=10 and n_estimators=100, respectively.

```
clf_10tree_pred = clf_10tree.predict(x_test)
clf_100tree_pred = clf_100tree.predict(x_test)
print(accuracy_score(y_test, clf_10tree_pred))
print(accuracy_score(y_test, clf_100tree_pred))
```

✓ 0.1s

```
0.9266666666666666
0.9566666666666667
```

5.2. Using Criterion='gini', Max_depth=None, N_estimators=10, Bootstrap=True, showing the accuracy score of test data by Max_features=sqrt(n_features) and Max_features=n_features, respectively.

```
clf_random_features_pred = clf_random_features.predict(x_test)
clf_all_features_pred = clf_all_features.predict(x_test)
print(accuracy_score(y_test, clf_random_features_pred))
print(accuracy_score(y_test, clf_all_features_pred))
```

✓ 0.9s

```
0.9466666666666667
0.97
```

6. (20%) Tune the hyperparameter, perform feature engineering or implement more powerful ensemble methods to get a higher accuracy score. Please note that only the ensemble method can be used. The neural network method is not allowed.

```
my_model = MyModel()
my_model.fit(x_train, y_train)
y_pred = my_model.predict(x_test)
print(accuracy_score(y_test, y_pred))
```

✓ 1m 41.3s

```
0.9566666666666667
```

Part. 2, Questions (30%):

1. Why does a decision tree have a tendency to overfit to the training set? Is it possible for a decision tree to reach a 100% accuracy in the training set? please explain. List and describe at least 3 strategies we can use to reduce the risk of overfitting of a decision tree.

因為 decision tree 一直長下去，會針對 train data 的資料進行細分，直到每個點都是相同類別的資料導致每個 leaf node 的資料過度細分而導致 overfit，也因此拿 train data 進行 predict 的話可以到達 100%。

1. 增加 trainning data: 有夠樣的數據，就不會導致 decision tree 過度偏向某一種類的數據
2. 當 node 的 data 數過小就不再 split: 避免 decision tree 深度到太深，將每個 leaf node 都細分成極少量的數據
3. 交叉驗證: 將資料切分成 k 份，用 k - 1 份 data 進行 train ，再用剩下那份進行 test ，進行 k 次，評估是否 overfit ，再針對超參數進行調整。

2. This part consists of three True/False questions. Answer True/False for each question and briefly explain your answer.

a. In AdaBoost, weights of the misclassified examples go up by the same multiplicative factor.

True. $D_{t+1}(i) = \frac{D_t(i)\exp(-a_t y_i h_t(x_i))}{Z_t}$ 這個 weak classifier predict 錯的資料 $y_i h_t(x_i)$ 會為 -1, 正確的資料為 1。因此同輪的 weight ，正確的都乘 $\exp(-a_i)$ ，錯誤的都乘 $\exp(a_i)$ ，最後再 normalization。Hence weights of the misclassified examples go up by the same multiplicative factor.

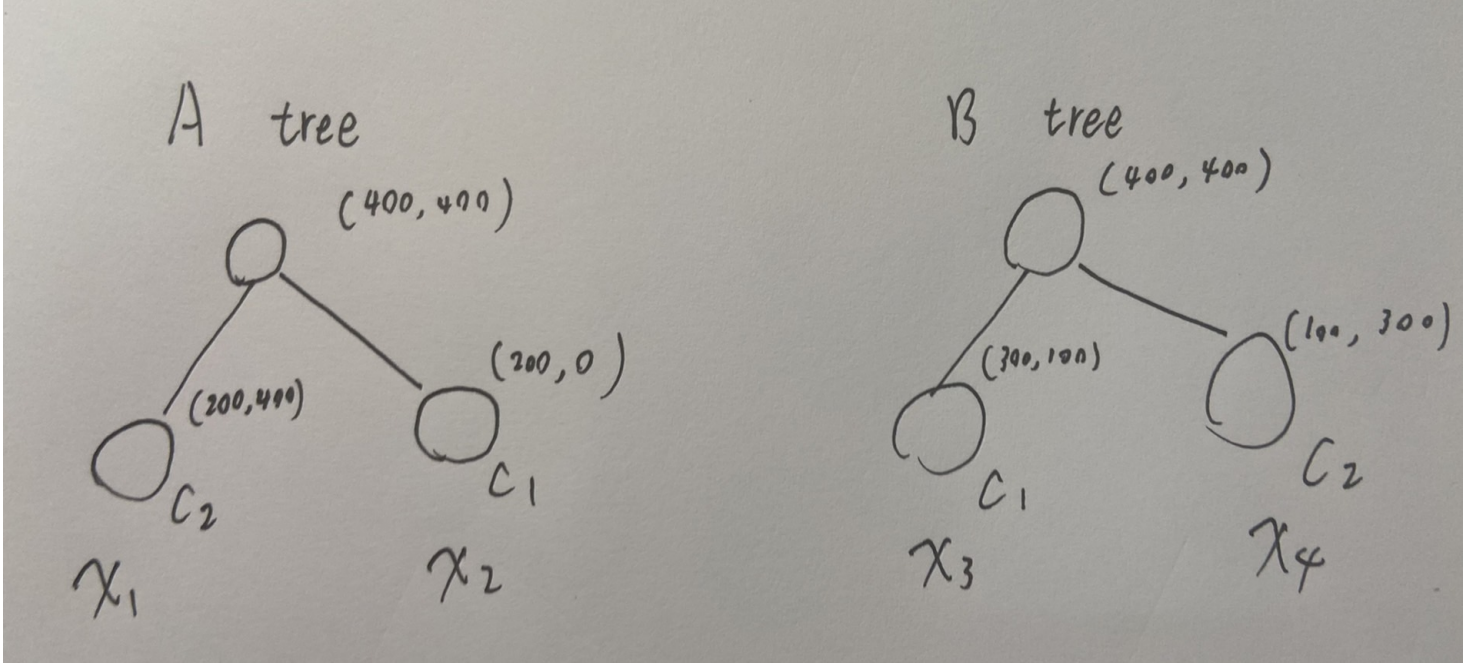
b. In AdaBoost, weighted training error ε_t of the t_{th} weak classifier on training data with weights D_t tends to increase as a function of t.

True. 增加錯誤資料的權重，會導致錯誤資料一直被重視，但這些資料本身就是很難被分類的資料，會導致一直分一直錯，因而增加權重 error。

c. AdaBoost will eventually give zero training error regardless of the type of weak classifier it uses, provided enough iterations are performed.

False. weak classifier 需要比 random 亂猜來得好。

3. Consider a data set comprising 400 data points from class C_1 and 400 data points from class C_2. Suppose that a tree model A splits these into $(200, 400)$ at the first leaf node and $(200, 0)$ at the second leaf node, where (n, m) denotes that n points are assigned to C_1 and m points are assigned to C_2. Similarly, suppose that a second tree model B splits them into $(300, 100)$ and $(100, 300)$. Evaluate the misclassification rates for the two trees and hence show that they are equal. Similarly, evaluate the cross-entropy $Entropy = -\sum_{k=1}^{K} p_k log_2 p_k$ and Gini index $Gini = 1 - \sum_{k=1}^{K} p_k^2$ for the two trees. Define p_k to be the proportion of data points in region R assigned to class k, where $k = 1, \ldots, K$.



建立 decision tree 時，leaf node 的 class 會被判定為較多資料的 class 因此 x_1, x_4 為 C_2 ， x_2, x_3 為 C_1 ，而對於 A tree 而言有 200 個 C_1 data 被誤分到 x_1 ，對於 B tree 而言有 100 個 C_2 data 誤分到 x_3 ， 100 個 C_1 data 誤分到 x_4 ，兩個 tree 的 misclassification rates 都是 0.25 。

x_1 :

$$Entropt = -(400/600)\log_2(400/600) - (200/600)\log_2(200/600) = 0.918$$

$$Gini = 1 - (400/600)^2 - (200/600)^2 = 4/9 = 0.444$$

x_2 :

$$Entropt = -(200/200)\log_2(200/200) - (0/200)\log_2(0/200) = 0$$

$$Gini = 1 - (200/200)^2 - (0/200)^2 = 0$$

x_3 :

$$Entropt = -(300/400)\log_2(300/400) - (100/400)\log_2(100/400) = 0.811$$

$$Gini = 1 - (300/400)^2 - (100/400)^2 = 3/8 = 0.375$$

x_4 :

$$Entropt = -(100/400)\log_2(100/400) - (300/400)\log_2(300/400) = 0.811$$

$$Gini = 1 - (100/400)^2 - (300/400)^2 = 3/8 = 0.375$$