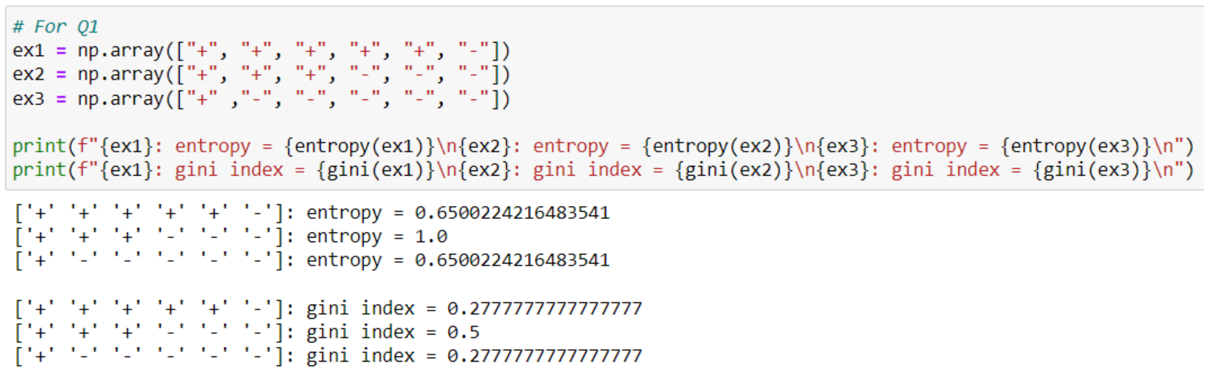
NYCU Pattern Recognition, Homework 3

311553010, 陳姿羽

Part. 1, Coding (70%):

1. (5%) Compute the Entropy and Gini index of the array provided in the sample code, using the formulas on page 6 of the HW3 slide.
   ['+' '+' '+' '+' '+' '-']: entropy = 0.6500224216483541
   ['+' '+' '+' '-' '-' '-']: entropy = 1.0
   ['+' '-' '-' '-' '-' '-']: entropy = 0.6500224216483541

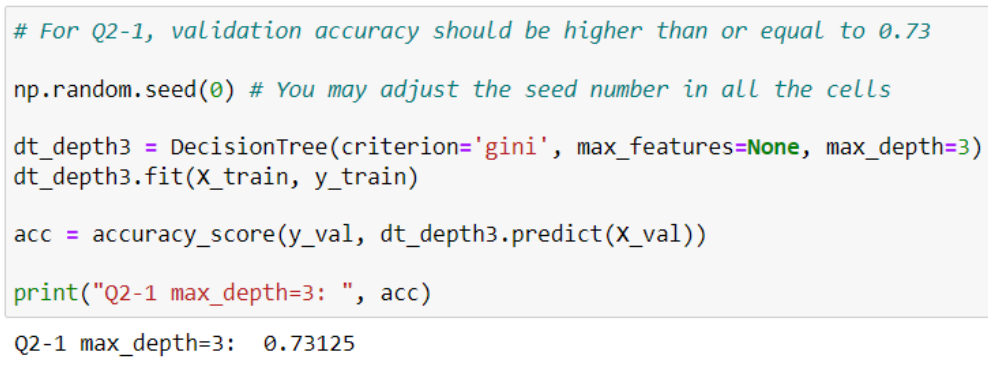
   ['+' '+' '+' '+' '+' '-']: gini index = 0.2777777777777777
   ['+' '+' '+' '-' '-' '-']: gini index = 0.5
   ['+' '-' '-' '-' '-' '-']: gini index = 0.2777777777777777



```
# For Q1
ex1 = np.array(["+", "+", "+", "+", "+", "-"])
ex2 = np.array(["+", "+", "+", "-", "-", "-"])
ex3 = np.array(["+" ,"-", "-", "-", "-", "-"])

print(f"{ex1}: entropy = {entropy(ex1)}\n{ex2}: entropy = {entropy(ex2)}\n{ex3}: entropy = {entropy(ex3)}\n")
print(f"{ex1}: gini index = {gini(ex1)}\n{ex2}: gini index = {gini(ex2)}\n{ex3}: gini index = {gini(ex3)}\n")
```

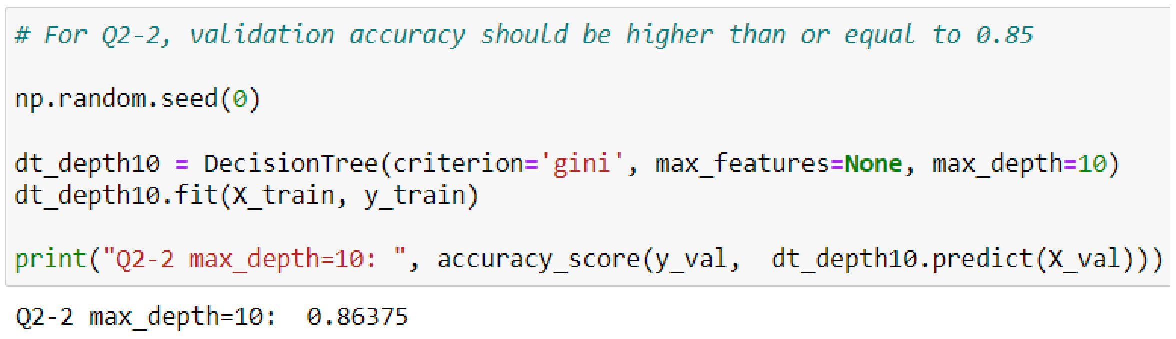
```
['+' '+' '+' '+' '+' '-']: entropy = 0.6500224216483541
['+' '+' '+' '-' '-' '-']: entropy = 1.0
['+' '-' '-' '-' '-' '-']: entropy = 0.6500224216483541

['+' '+' '+' '+' '+' '-']: gini index = 0.2777777777777777
['+' '+' '+' '-' '-' '-']: gini index = 0.5
['+' '-' '-' '-' '-' '-']: gini index = 0.2777777777777777
```

2. (10%) Show the accuracy score of the validation data using criterion='gini' and max_features=None for max_depth=3 and max_depth=10, respectively.
   Q2-1 max_depth=3: 0.73125
   Q2-2 max_depth=10: 0.86375



```
# For Q2-1, validation accuracy should be higher than or equal to 0.73

np.random.seed(0) # You may adjust the seed number in all the cells

dt_depth3 = DecisionTree(criterion='gini', max_features=None, max_depth=3)
dt_depth3.fit(X_train, y_train)

acc = accuracy_score(y_val, dt_depth3.predict(X_val))

print("Q2-1 max_depth=3: ", acc)
```

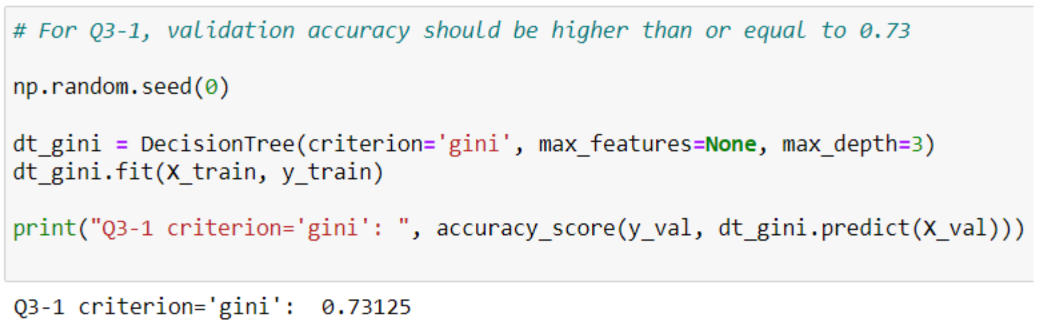
```
Q2-1 max_depth=3:  0.73125
```



```
# For Q2-2, validation accuracy should be higher than or equal to 0.85

np.random.seed(0)

dt_depth10 = DecisionTree(criterion='gini', max_features=None, max_depth=10)
dt_depth10.fit(X_train, y_train)

print("Q2-2 max_depth=10: ", accuracy_score(y_val,  dt_depth10.predict(X_val)))
```

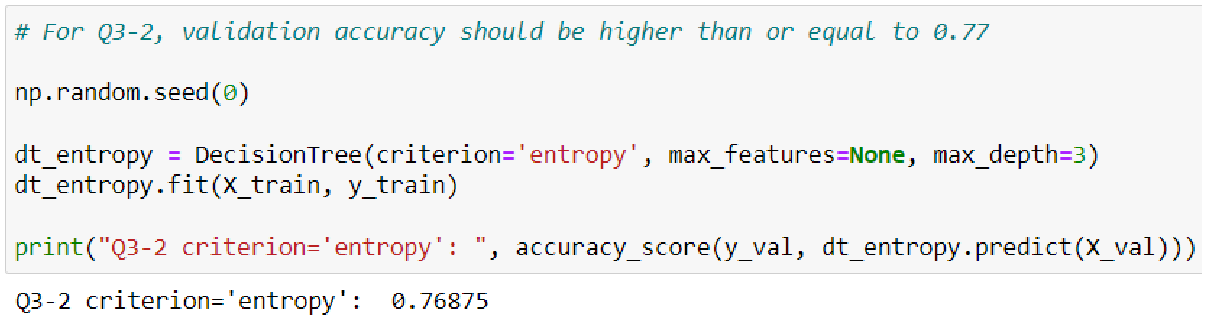
```
Q2-2 max_depth=10:  0.86375
```

3. (10%) Show the accuracy score of the validation data using max_depth=3 and max_features=None, for criterion='gini' and criterion='entropy', respectively.
   Q3-1 criterion='gini': 0.73125
   Q3-2 criterion='entropy': 0.76875



```
# For Q3-1, validation accuracy should be higher than or equal to 0.73

np.random.seed(0)

dt_gini = DecisionTree(criterion='gini', max_features=None, max_depth=3)
dt_gini.fit(X_train, y_train)

print("Q3-1 criterion='gini': ", accuracy_score(y_val, dt_gini.predict(X_val)))
```

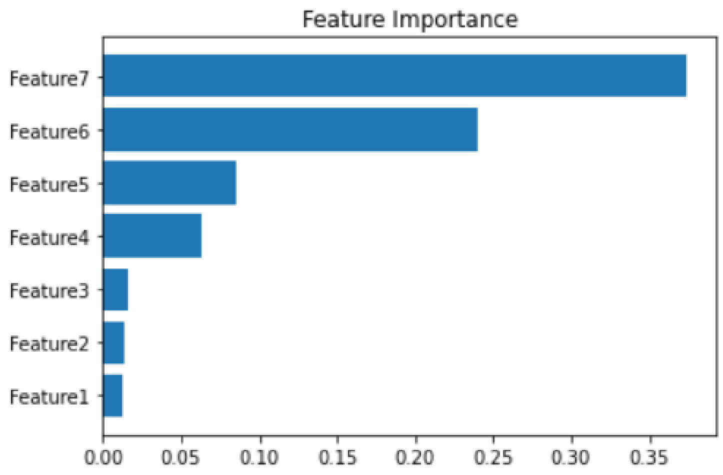
```
Q3-1 criterion='gini':  0.73125
```



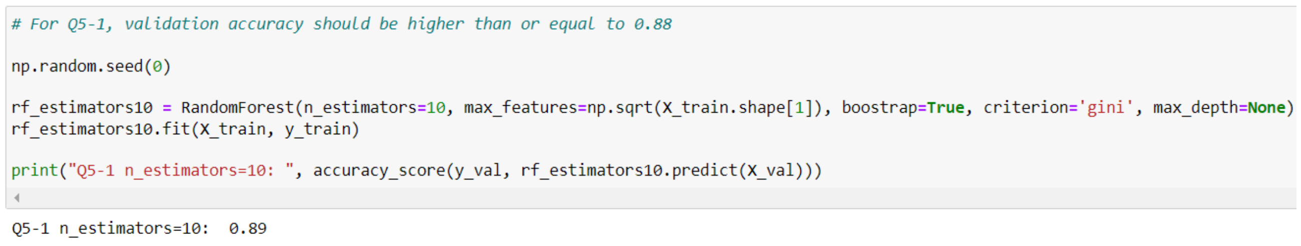
```
# For Q3-2, validation accuracy should be higher than or equal to 0.77

np.random.seed(0)

dt_entropy = DecisionTree(criterion='entropy', max_features=None, max_depth=3)
dt_entropy.fit(X_train, y_train)

print("Q3-2 criterion='entropy': ", accuracy_score(y_val, dt_entropy.predict(X_val)))
```

```
Q3-2 criterion='entropy':  0.76875
```

4. (5%) Train your model using criterion='gini', max_depth=10 and max_features=None. Plot the feature importance of your decision tree model by simply counting the number of times each feature is used to split the data.

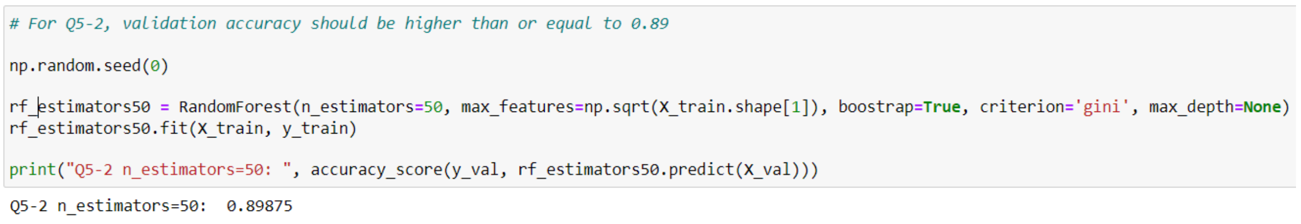


5. (10%) Show the accuracy score of the validation data using criterion='gini', max_depth=None, max_features=sqrt(n_features), and bootstrap=True, for n_estimators=10 and n_estimators=50, respectively.
   Q5-1 n_estimators=10: 0.89
   Q5-2 n_estimators=50: 0.89875



```
# For Q5-1, validation accuracy should be higher than or equal to 0.88

np.random.seed(0)

rf_estimators10 = RandomForest(n_estimators=10, max_features=np.sqrt(X_train.shape[1]), boostrap=True, criterion='gini', max_depth=None)
rf_estimators10.fit(X_train, y_train)

print("Q5-1 n_estimators=10: ", accuracy_score(y_val, rf_estimators10.predict(X_val)))
```

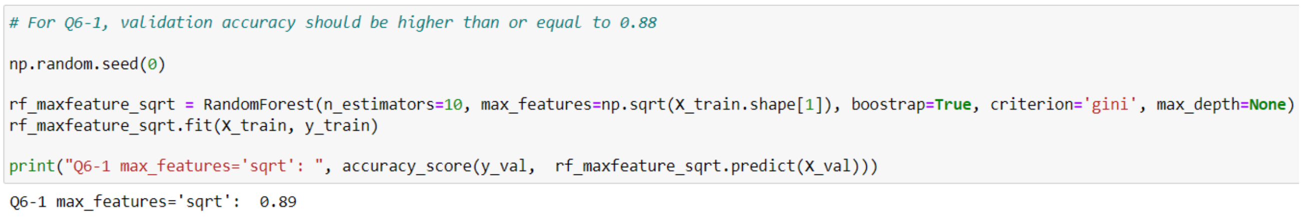
```
Q5-1 n_estimators=10:  0.89
```



```
# For Q5-2, validation accuracy should be higher than or equal to 0.89

np.random.seed(0)

rf_estimators50 = RandomForest(n_estimators=50, max_features=np.sqrt(X_train.shape[1]), boostrap=True, criterion='gini', max_depth=None)
rf_estimators50.fit(X_train, y_train)

print("Q5-2 n_estimators=50: ", accuracy_score(y_val, rf_estimators50.predict(X_val)))
```

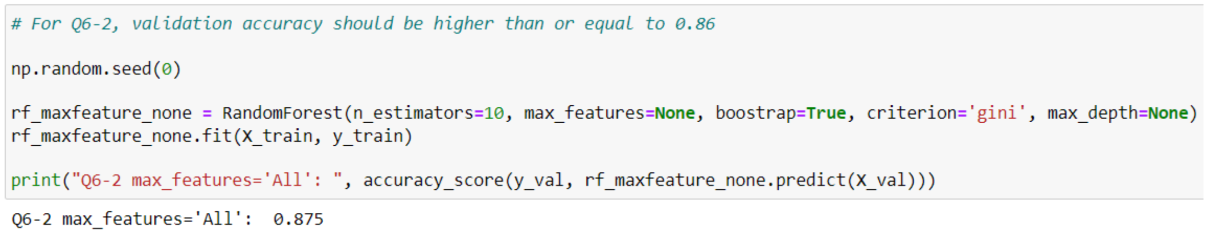
```
Q5-2 n_estimators=50:  0.89875
```

6. (10%) Show the accuracy score of the validation data using criterion='gini', max_depth=None, n_estimators=10, and bootstrap=True, for max_features=sqrt(n_features) and max_features=n_features, respectively.
   Q6-1 max_features='sqrt': 0.89
   Q6-2 max_features='All': 0.875



```
# For Q6-1, validation accuracy should be higher than or equal to 0.88

np.random.seed(0)

rf_maxfeature_sqrt = RandomForest(n_estimators=10, max_features=np.sqrt(X_train.shape[1]), boostrap=True, criterion='gini', max_depth=None)
rf_maxfeature_sqrt.fit(X_train, y_train)

print("Q6-1 max_features='sqrt': ", accuracy_score(y_val,  rf_maxfeature_sqrt.predict(X_val)))
```

```
Q6-1 max_features='sqrt':  0.89
```



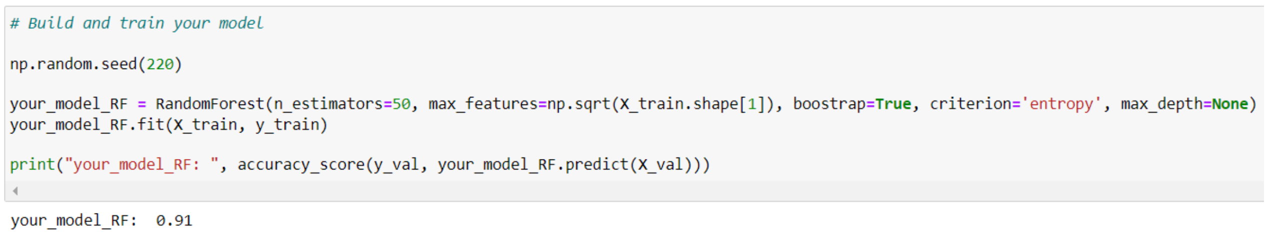
```
# For Q6-2, validation accuracy should be higher than or equal to 0.86

np.random.seed(0)

rf_maxfeature_none = RandomForest(n_estimators=10, max_features=None, boostrap=True, criterion='gini', max_depth=None)
rf_maxfeature_none.fit(X_train, y_train)

print("Q6-2 max_features='All': ", accuracy_score(y_val, rf_maxfeature_none.predict(X_val)))
```

```
Q6-2 max_features='All':  0.875
```

7. (20%) Explain how you chose/design your model and what feature processing you have done in detail. Otherwise, no points will be given.
   random seed=220
   n_estimators=50
   max_features=np.sqrt(X_train.shape[1])
   boostrap=True
   criterion='entropy'
   max_depth=None
   your_model_RF accuracy_score: 0.91

   First, I used the models of decision tree and random forest respectively to choose the appropriate model. However, I found that it was difficult for the decision tree to achieve an accuracy higher than 0.9, so I chose random forest model. Next, I began adjusting the parameters, referencing the parameters used in Q5-2. I found that the

accuracy score was higher when using Entropy as the criterion rather than Gini index, and that the accuracy score did not change significantly when n_estimators was greater than 50. At this point, the accuracy score had reached 0.905. Finally, I adjusted the random seed based on the discussions in the forum, and the accuracy score reached 0.91 when using a random seed of 220.
And I also found that the accuracy score using for loop to compute impurity is higher than using np.sum().



```
# Build and train your model

np.random.seed(220)

your_model_RF = RandomForest(n_estimators=50, max_features=np.sqrt(X_train.shape[1]), boostrap=True, criterion='entropy', max_depth=None)
your_model_RF.fit(X_train, y_train)

print("your_model_RF: ", accuracy_score(y_val, your_model_RF.predict(X_val)))
```

```
your_model_RF:  0.91
```

Part. 2, Questions (30%):

1. Answer the following questions in detail:

a、 Why does a decision tree tend to overfit the training set?

When the deeper and more complex the tree, the greater the likelihood of overfitting. Because it captures too much noise data and creates the rules that only apply to the training set.

b、 Is it possible for a decision tree to achieve 100% accuracy on the training set?

Yes. If we keep add branches until each instance has its own leaf, it is possible to achieve 100% accuracy on the training set. But the result would be severe overfitting and poor generalization to new data.

c、 List and describe at least three strategies we can use to reduce the risk of overfitting in a decision tree.

i. Setting a maximum depth: Setting a maximum depth to avoid learning complex and creating the rules that only apply to the training set.

ii. Ensemble methods: Combine multiple decision trees to make predictions, such as Random Forest. By combining the predictions of multiple trees, to reduce overfitting and improve the generalization ability of the model.

iii. Pruning: Removing some nodes to avoid the tree growing to its full depth. It can improve the ability of the validation set tree.

2. For each statement, answer True or False and provide a detailed explanation:

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

False. The misclassified examples will go up, but not by the same multiplicative factor.

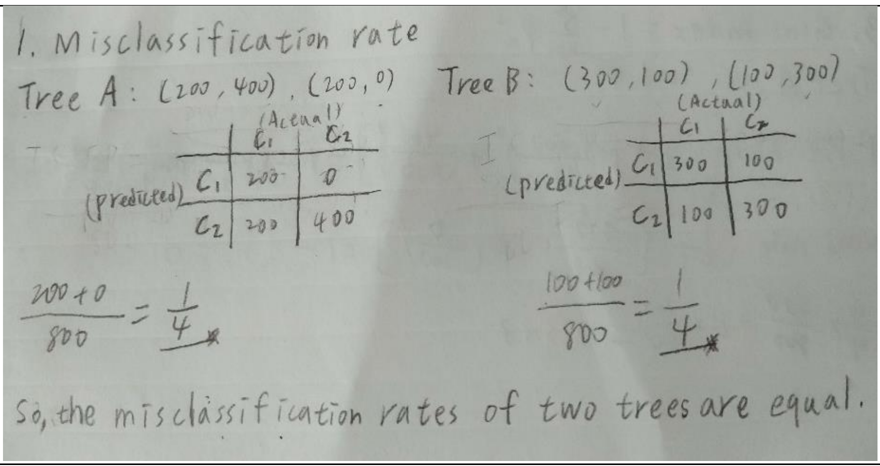
b、 In AdaBoost, weighted training error ε_t of the t_{th} weak classifier on training data with weights D_t tends to increase as a function of *t*.

False. The weighted training error ε_t will decrease as a function of *t*.

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

False. We cannot ensure that AdaBoost will eventually give zero training error. If the weak classifier isn't fit the training data or isn't strong, AdaBoost will not give zero training error in any iterations.

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 <u>misclassification</u> 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, . . . , K.

Misclassification rates


1. Misclassification rate

Tree A : (200, 400) , (200, 0)

		(Actual) C_1	C_2
(predicted)	C_1	200	0
	C_2	200	400

$\frac{200+0}{800} = \frac{1}{4}$ *

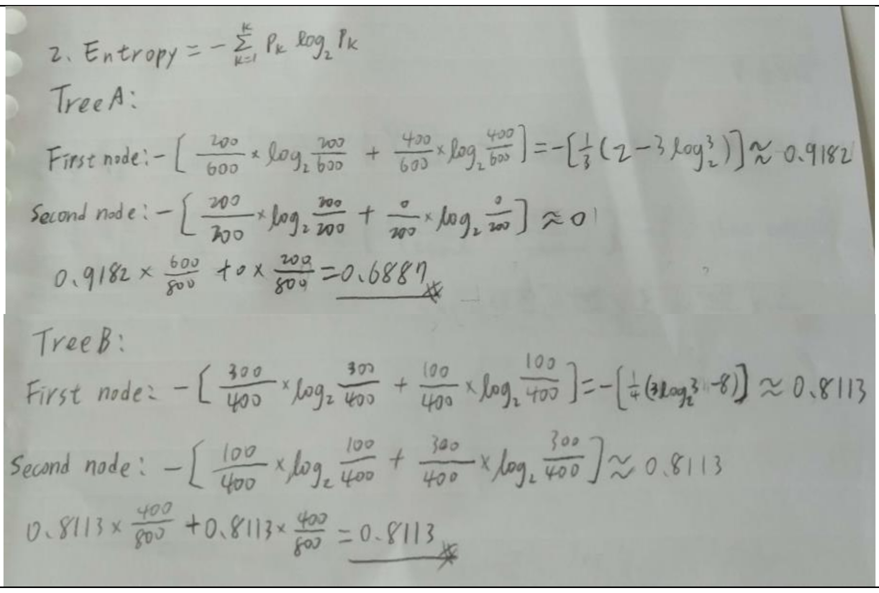
Tree B : (300, 100) , (100, 300)

		(Actual) C_1	C_2
(predicted)	C_1	300	100
	C_2	100	300

$\frac{100+100}{800} = \frac{1}{4}$ *

So, the misclassification rates of two trees are equal.

Cross-entropy



2、Entropy $= -\sum_{k=1}^{K} p_k \log_2 p_k$

Tree A:

First node: $-\left[\frac{200}{600}\times\log_2\frac{200}{600}+\frac{400}{600}\times\log_2\frac{400}{600}\right]=-\left[\frac{1}{3}\left(2-3\log_2^3\right)\right]\approx 0.9182$

Second node: $-\left[\frac{200}{200}\times\log_2\frac{200}{200}+\frac{0}{200}\times\log_2\frac{0}{200}\right]\approx 0$

$0.9182\times\frac{600}{800}+0\times\frac{200}{800}=\underline{0.6887}$

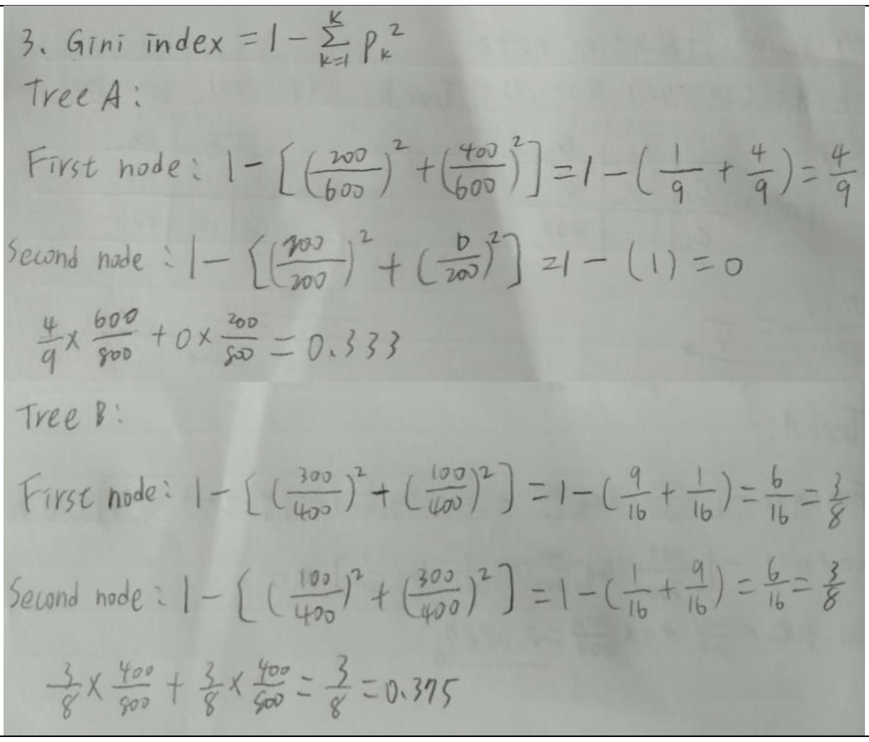
Tree B:

First node: $-\left[\frac{300}{400}\times\log_2\frac{300}{400}+\frac{100}{400}\times\log_2\frac{100}{400}\right]=-\left[\frac{1}{4}\left(3\log_2^3-8\right)\right]\approx 0.8113$

Second node: $-\left[\frac{100}{400}\times\log_2\frac{100}{400}+\frac{300}{400}\times\log_2\frac{300}{400}\right]\approx 0.8113$

$0.8113\times\frac{400}{800}+0.8113\times\frac{400}{800}=\underline{0.8113}$

Gini index



3、Gini index $=1-\sum_{k=1}^{K} p_k^2$

Tree A:

First node: $1-\left[\left(\frac{200}{600}\right)^2+\left(\frac{400}{600}\right)^2\right]=1-\left(\frac{1}{9}+\frac{4}{9}\right)=\frac{4}{9}$

Second node: $1-\{[\left(\frac{200}{200}\right)^2+\left(\frac{0}{200}\right)^2]=1-(1)=0$

$\frac{4}{9}\times\frac{600}{800}+0\times\frac{200}{800}=0.333$

Tree B:

First node: $1-[\left(\frac{300}{400}\right)^2+\left(\frac{100}{400}\right)^2]=1-\left(\frac{9}{16}+\frac{1}{16}\right)=\frac{6}{16}=\frac{3}{8}$

Second node: $1-\{\left(\frac{100}{400}\right)^2+\left(\frac{300}{400}\right)^2]=1-\left(\frac{1}{16}+\frac{9}{16}\right)=\frac{6}{16}=\frac{3}{8}$

$\frac{3}{8}\times\frac{400}{800}+\frac{3}{8}\times\frac{400}{800}=\frac{3}{8}=0.375$