# NYCU Pattern Recognition, Homework 3

**311551059, 陳昱丞**

## Part. 1, Coding (70%):

You should type the answer and also screenshot at the same time. Otherwise, no points will be given. The screenshot and the figures we provided below are just examples. **The results below are not guaranteed to be correct.** Please convert it to a pdf file before submission. You should use English to answer the questions. After reading this paragraph, you can delete it.

## Decision Tree

1. (5%) Compute the Entropy and Gini index of the array provided in the sample code.

In [8]:
```
# 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
```

['+' '+' '+' '+' '+' '-']: 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.

   max_depth=3:
   accuracy: 0.7325

In [9]:
```
# 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, sample_weight=None)

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

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

```
Q2-1 max_depth=3:  0.7325
```

max_depth=10:
accuracy: 0.855

In [11]:
```
# 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, sample_weight=None)

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

```
Q2-2 max_depth=10:  0.855
```

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.

criterion='gini':
accuracy: 0.7325

In [12]:
```
# 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, sample_weight=None)

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

```
Q3-1 criterion='gini':  0.7325
```

criterion='entropy':
accuracy: 0.77

In [13]:
```
# 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, sample_weight=None)

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

```
Q3-2 criterion='entropy':  0.77
```

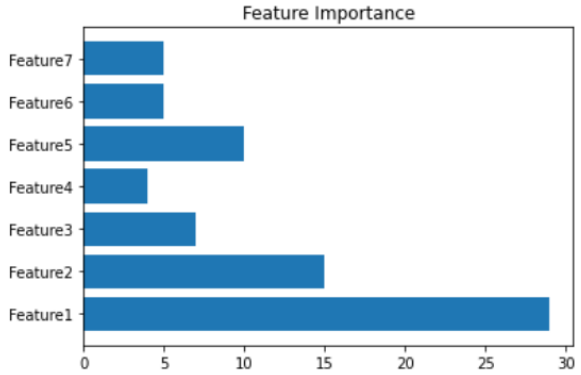
4. (5%) Train your model using criterion='gini', max_depth=10 and max_features=None. Plot the feature importance of your decision tree model.

In [14]:
```
# For Q4

# Use simply counting to get the feature importance: dt_depth10.importance

labelList=['Feature1', 'Feature2', 'Feature3', 'Feature4', 'Feature5', 'Feature6', 'Feature7']

plt.title('Feature Importance')
plt.barh(labelList, dt_depth10.importance)
plt.show()
```




## Random Forest

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.

n_estimators=10:
accuracy: 0.8975

In [15]:
```
# 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_dept
rf_estimators10.fit(X_train, y_train)

print("Q6-1 n_estimators=10: ", accuracy_score(y_val, rf_estimators10.predict(X_val)))
```

```
Q6-1 n_estimators=10:  0.8975
```

n_estimators=50:
accuracy: 0.89625

In [16]:
```
# 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_dept
rf_estimators50.fit(X_train, y_train)

print("Q6-1 n_estimators=50: ", accuracy_score(y_val, rf_estimators50.predict(X_val)))
```

```
Q6-1 n_estimators=50:  0.89625
```

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.

max_features=sqrt(n_features):
accuracy: 0.8975

```
In [17]: # 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_d
         rf_maxfeature_sqrt.fit(X_train, y_train)

         print("Q7-1 max_features='sqrt': ", accuracy_score(y_val,  rf_maxfeature_sqrt.predict(X_val)))

         Q7-1 max_features='sqrt':  0.8975
```

max_features=n_features:
accuracy: 0.875

```
In [18]: # 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("Q7-1 max_features='All': ", accuracy_score(y_val, rf_maxfeature_none.predict(X_val)))

         Q7-1 max_features='All':  0.875
```

## Train your own model

7. (20%) Explain how you chose/design your model and what feature processing you have done in detail. Otherwise, no points will be given.

In feature processing, I combined the training and validation dataset together as the input of my random forest model.

```
In [20]: X_all = np.concatenate((X_train, X_val))
         y_all = np.concatenate((y_train, y_val))
         print(X_all.shape)
         print(y_all.shape)

         (1600, 7)
         (1600,)
```

In RandomForest model, I set n_estimators=50, max_features=sqrt(#features), boostrap=True, criterion='gini', max_depth=None. It can achieve 0.99625 accuracy on the validation dataset.

```
In [22]: my_model = RandomForest(n_estimators=50, max_features=np.sqrt(X_all.shape[1]), boostrap=True, criterion='gini', max_depth=None)
         my_model.fit(X_all, y_all)
         print("my model n_estimators=50: ", accuracy_score(y_val, my_model.predict(X_val)))

         test_pred = my_model.predict(X_test)
         test_pred = np.expand_dims(test_pred, axis=1)
         print("test_pred shape: ", test_pred.shape)

         my model n_estimators=50:  0.99625
         test_pred shape:  (800, 1)
```

## Part. 2, Questions (30%):

1. Answer the following questions in detail:

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

Because a decision tree has a tendency to create complex and highly specific decision rules that perfectly fit the training data, even if they are not representative of the underlying patterns in the data. This can lead to poor generalization performance on new, unseen data.

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

Yes, it is possible for a decision tree to achieve 100% accuracy on the training set. As above mentioned, decision trees tend to create complex and highly specific decision rules that perfectly fit the training data.

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

1. Fixed depth: Fixed the depth of the tree so that we can limit the tree from growing too deep.
2. Minimum number of data points per leaf: Set a minimum number of data points that must be present in each leaf node. This means that a split is only allowed if the resulting subsets have more than a certain number of data points.
3. Minimum information gain: The decision tree is only allowed to split the data if the resulting subsets are significantly more pure than the original data.

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.

True.
For each misclassified example i, the weight is updated as w_i * exp(alpha).

b. In AdaBoost, weighted training error $\varepsilon t$ of the t-th weak classifier on training data with weights Dt tends to increase as a function of t.

False.
Because subsequent weak classifiers are trained to focus on the difficult-to-classify examples that were misclassified by previous weak classifiers. Therefore, the weighted training error of the t-th weak classifier tends to 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.
If the chosen weak classifier is not expressive enough to capture the underlying patterns in the data, then AdaBoost may not be able to achieve zero training error.

3. Consider a data set comprising 400 data points from class C1 and 400 data points from class C2. 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 C1 and m points are assigned to C2. 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 and Gini index for the two trees. Define Pk to be the proportion of data points in region R assigned to class k, where k = 1, . . . , K.

In tree A, the majority class in the first leaf node is C2, and in the second leaf node, the majority is C1. So the number of misclassification points in A is 200 in the first leaf node; In tree B, the majority class in the first leaf node is C1, and in the second leaf node, the majority is C2. So the number of misclassification points in B is 200 (100 in the first leaf node and 100 in the second leaf node). Therefore, the misclassification rate of the two models are equal, both 200 / 800 = 0.25.

cross-entropy of A = 3/4 * (entropy of first leaf) + 1/4 * (entropy of second leaf)
= 3/4 * (1/3*log2(1/3)+2/3*log2(2/3)) + 1/4 * 0 = 3/4 * 0.918 = 0.6885
cross-entropy of B = 1/2 * (entropy of first leaf) + 1/2 * (entropy of second leaf)
= 1/2 * (3/4*log2(3/4)+1/4log2(1/4)) * 2 = 0.811
gini-index of A = 3/4 * (gini of first leaf) + 1/4 * (gini of second leaf)
= 3/4 * (1-(1/3)^2-(2/3)^2) + 1/4 * (1-1) = 0.333
gini-index of B = 1/2 * (gini of first leaf) + 1/2 * (gini of second leaf)
= 1/2 * (1-(3/4)^2-(1/4)^2) * 2 = 0.375