NYCU Pattern Recognition, Homework 2

Deadline: Apr. 26, 23:59

Part. 1, Coding (80%):

For this coding assignment, you are required to implement the Decision Tree and Random Forest a lgorithms using only NumPy. Afterward, you will need to train your model on the provided dataset and evaluate its performance on the validation data.

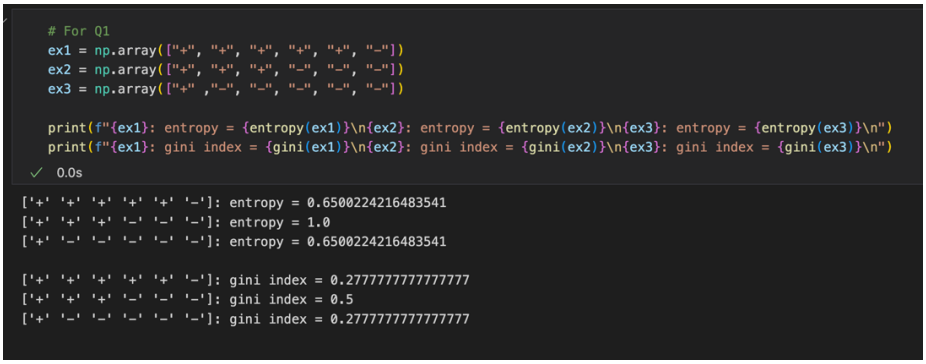
(30%) Decision Tree

Requirements:

- Implement the Gini and Entropy for measuring the "best" splitting of the data.
- Implement the Decision Tree algorithm (CART, Classification and Regression Trees) with the following 3 arguments:
- Criterion: The function to measure the quality of a split of the data. Your model should su pport "gini" for the Gini impurity and "entropy" for the information gain.
- Max_depth: The maximum depth of the tree. If Max_depth=None, then nodes are expande d until all leaves are pure. Max_depth=1 equals splitting data once.
- Max_features: The number of features to consider when looking for the best split. If None, then max_features=n_features.
- For more detailed descriptions of the arguments, please refer to Scikit-learn.
- Your model should produce the same results when rebuilt with the same arguments, and th ere is no need to prune the trees.
- You can use the recursive method to build the nodes.

Criteria:

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.



```
# 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")
```

✓ 0.0s

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

```
# 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)
```

✓ 0.3s

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

```
# 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)))
```

✓ 0.8s

```
Q2-2 max_depth=10:  0.8525
```

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.

```
# 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)))
```

✓ 0.3s

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

```
# 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)))
```

✓ 0.3s

```
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 by simply counting the number of time s each feature is used to split the data.

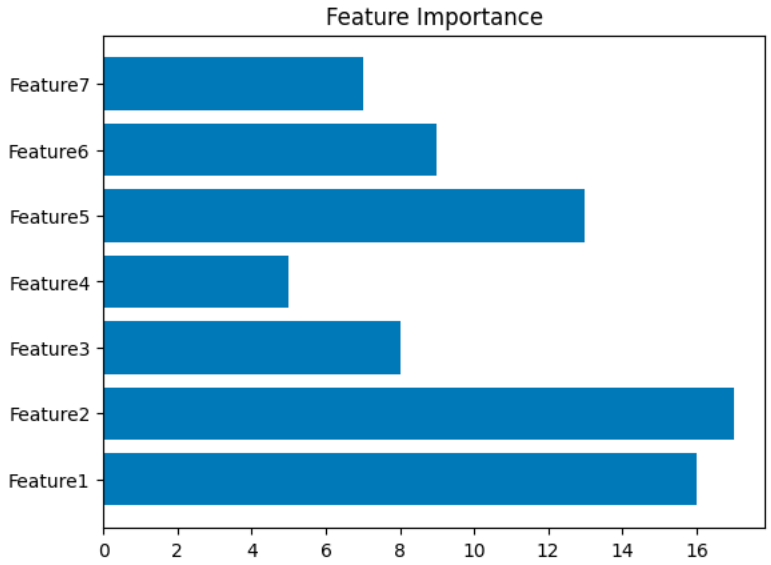
```
# For Q4

# 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.values())
plt.show()
```

✓ 0.1s



(20%) Random Forest

Requirements:

- Fix the random seed.
- Implement the Random Forest algorithm by using the CART you just implemented.
- The Random Forest model should include the following three arguments:
- N_estimators: The number of trees in the forest.
- Max_features: The number of features to consider when looking for the best split using the decision tree.
- Bootstrap: Whether to use bootstrap samples when building trees.
- For more detailed descriptions of the arguments, please refer to Scikit-learn.
- Use majority voting to obtain the final prediction.

Criteria:

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.

```
# 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]), bo
rf_estimators10.fit(X_train, y_train)

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

✓ 1.8s

```
Q6-1 n_estimators=10:  0.88625
```

```
# 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
rf_estimators50.fit(X_train, y_train)

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

✓ 8.6s

```
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.

```
# 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]), boostr
rf_maxfeature_sqrt.fit(X_train, y_train)

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

✓ 1.7s

```
Q7-1 max_features='sqrt':  0.88625
```

```
# 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_dep
rf_maxfeature_none.fit(X_train, y_train)

print("Q7-1 max_features='All': ", accuracy_score(y_val, rf_maxfeature_none.predict(X_val)))
```

✓ 5.2s

```
Q7-1 max_features='All':  0.86625
```

(20%) Train your own model

Requirements:

- Train your model (either Decision Tree or Random Forest).

- Try different parameters and feature engineering to beat the baseline.
- Save your test predictions in a CSV file.

Criteria:

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

Points	Testing Accuracy
20 points	acc >= 0.90625
15 points	acc > 0.89
10 points	acc > 0.85
5 points	acc > 0.8
0 points	acc <= 0.8

```python
def varible_extend( df, expend = ['Feature1','Feature2','Feature3','Feature4','Feature5','Feature6','Feature7'] ) :
    X = df.drop('Target', axis=1)

    for i in range(len(expend)) :
        for j in range(i,len(expend)) :
            name = f'{expend[i]}*{expend[j]}'
            X[name] = X[expend[i]]*X[expend[j]]
            # X[name] = (X[name] - X[name].mean())/X[name].std()

    # for i in range(len(expend)) :
    #     X[expend[i]] = (X[expend[i]] - X[expend[i]].mean())/X[expend[i]].std()

    return X
```

✓ 0.0s Python

For this project, I also used a feature expend to expand my feature, and because the decision tree would ovetfitting, I used random forest as my fit model, and finally, ac cording to the above experimental results, I used "gini" to as my criterion

Part. 2, Questions (30%):

1. Answer the following questions in detail:
    a. Why does a decision tree tend to overfit the training set?

    A decision tree is a supervised machine learning algorithm that recurses the input s pace into subsets based on the values of the input features to create a set of if-then rules for prediction. Each partition in the decision tree is designed to maximize the information gain, i.e., the difference between the impurity of the parent node and t

he weighted impurity of the child nodes. However, if the decision tree is allowed t o grow too deep or too wide, it can become too overlapping and too close to the n oise in the training data. This can lead to overfitting, i.e., a phenomenon where the model performs well on the training data but poorly on the new data.

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

   It is possible, especially if the decision tree is allowed to grow deep enough to fit the noise in the data perfectly. However, this does not necessarily mean that the m odel will perform well on new data, as it may have learned to classify training inst ances based on random fluctuations or anomalies that do not represent the true und erlying patterns in the data. In practice, it is often better to have the model achieve higher accuracy on the validation set, which is a separate set of examples that the model does not see during training.

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

There are three strategies that can be used to reduce the risk of overfitting in decision t rees:

1. Pruning: Pruning is a technique that involves removing nodes or branches from the t ree that do not contribute to the overall accuracy of the model. This is achieved by mea suring the decrease in accuracy caused by the removal of each node or branch, and by removing those nodes or branches that cause the least decrease.

Limiting the depth of the tree: Limiting the depth of the tree prevents it from becoming overly cluttered and over-fitting the data. This is achieved by setting the maximum dep th of the tree or by requiring the minimum number of samples present in each leaf nod e.

Using ensemble methods: Ensemble methods combine multiple decision trees to create a more robust model that is less prone to overfitting. A common ensemble approach is random forest, which builds multiple decision trees on a randomly selected subset of tr aining data and then aggregates their predictions. Another ensemble method is gradient

boosting, which superimposes decision trees on the model to correct the errors of previ ous trees.

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 multiplicati ve factor.

      True, follows from the update equation

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

   True.

   In the course of boosting iterations the weak classifiers are forced to try to classify more diffic ult examples. The weights will increase for examples that are repeatedly misclassified by the w eak classifiers. The weighted training error $\boldsymbol{\varepsilon}_t$ of the t(th) weak classifier on the training data therefore tends to increase.

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

      False,

      Not if the data in the training set cannot be separated by a linear

      combination of the specific type of weak classifiers we are using.

3. Consider a data set comprising 400 data points from class C_1 and 400 data points from clas s C_2. Suppose that a tree model A splits these into (200, 400) at the first leaf node and (20 0, 0) at the second leaf node, where (n, m) denotes that n points are assigned to C_1 and m p oints are assigned to C_2. Similarly, suppose that a second tree model B splits them into (30 0, 100) and (100, 300). Evaluate the misclassification rates for the two trees and hence sho w that they are equal. Similarly, evaluate the cross-entropy $\boldsymbol{Entropy} = -\sum_{k=1}^{K} \boldsymbol{p_k log_2 p_k}$ and Gini index $Gini = 1 - \sum_{k=1}^{K} p_k^2$ for the two tre

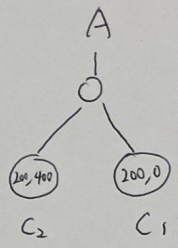
es. Define p_k to be the proportion of data points in region R assigned to class k, where k = 1, . . . , K.

Model A:

Leaf Node 1: p= 200/600 , q= 400/600

Cross entropy = $-1/3 * \log_2(1/3) - 2/3 * \log_2(2/3) \approx 0.9183$

Gini index = $1 - (1/9 + 4/9) = 4/9 \approx 0.4444$.



Leaf Node 2: p= 1, q=0.

Cross-entropy = $-1 * \log_2(1) - 0 * \log_2(0) = 0$.

Gini index = $1 - (1^2 + 0^2) = 0$.

↳ Miss = 200

∴ Miss Rate = $\frac{200}{800} = 0.25$

Entropy A: $\frac{3}{4}\left(-\frac{1}{3} * \log_2(\frac{1}{3}) + \frac{-2}{3}\log_2(\frac{2}{3})\right) = -\frac{1}{2} + \frac{3}{4}\log_2 3$ #

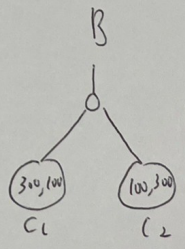
Gini A: $\frac{3}{4}\left(\frac{4}{9}\right) = \frac{1}{3}$ #

Model B:

Leaf Node 1: p= 300/400 , q= 100/400

Cross entropy: $-\left(\frac{3}{4}\cdot\log_2\frac{3}{4} + \frac{1}{4}\log_2\frac{1}{4}\right) = 2 - 3/4\log_2 3$

Gini : $1 - \left(\left(\frac{3}{4}\right)^2 + \left(\frac{1}{4}\right)^2\right) = \frac{3}{8}$



Leaf Node 2: p= $\frac{100}{400}$ q= $\frac{300}{400}$

Cross entropy: $-\left(\frac{1}{4}\cdot\log_2\frac{1}{4} + \frac{3}{4}\log_2\frac{3}{4}\right)$

Gini : $\left(1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right)\right) = \frac{3}{8}$

↳ Miss = 100 + 100 = 200

∴ Miss Rate = $\frac{200}{800} = 0.25$

Entropy B: $\frac{1}{2}(2 - 3/4\log_2 3) + \frac{1}{2}(2 - \frac{1}{4}\log_2 3) = 2 - \frac{3}{4}\log_2 3$ #

Gini B: $\frac{1}{2} * \frac{3}{8} + \frac{1}{2} * \frac{3}{8} = \frac{3}{8}$ #