NYCU Pattern Recognition, Homework 3

Deadline: May 4, 23:59

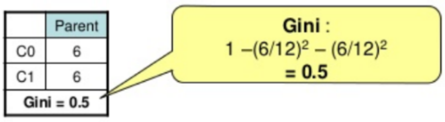
Part. 1, Coding (80%):

In this coding assignment, you need to implement the Decision Tree, AdaBoost and Random Forest algorithm by using only NumPy, then train your implemented model by the provided dataset and test the performance with testing data. Find the sample code and data on the GitHub page
https://github.com/NCTU-VRDL/CS_AT0828/tree/main/HW3

Please note that only NumPy can be used to implement your model, you will get no points by simply calling sklearn.tree.DecsionTreeClassifier.

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)

$$Gini = 1 - \sum_j p_j^2$$



$$Entropy = -\sum_j p_j \log_2 p_j$$

- If all classes are the same in one node

$$entropy = -1\log_2 1 = 0$$

- If the classes are half-and-half

$$entropy = -0.5\log_2 0.5 - 0.5\log_2 0.5 = 1$$

```
print("Gini of data is ", gini(data))

Gini of data is  0.4628099173553719
```

```
[10] print("Entropy of data is ", entropy(data))

Entropy of data is  0.9456603046006402
```

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.

```
[14] clf_depth3 = DecisionTree(criterion='gini', max_depth=3)
     clf_depth3.fit(train_df_data, train_df_target)

     pred_target = clf_depth3.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)

     print("Accuracy:", acc)

Accuracy: 0.79
```

```
[15] clf_depth10 = DecisionTree(criterion='gini', max_depth=10)
     clf_depth10.fit(train_df_data, train_df_target)

     pred_target = clf_depth10.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)

     print("Accuracy:", acc)

Accuracy: 0.74
```

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

```
[16] clf_gini = DecisionTree(criterion='gini', max_depth=3)
     clf_gini.fit(train_df_data, train_df_target)

     pred_target = clf_gini.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)

     print("Accuracy:", acc)

Accuracy: 0.79
```

```
[17] clf_entropy = DecisionTree(criterion='entropy', max_depth=3)
     clf_entropy.fit(train_df_data, train_df_target)

     y_pred = clf_entropy.predict(test_df_data)
     acc = accuracy_score(test_df_target, y_pred)

     print("Accuracy:", acc)

Accuracy: 0.75
```

Note: Your decisition tree scores should over ***0.7****. It may suffer from overfitting, if so, you can tune the hyperparameter such as `max_depth`*

Note: You should get the same results when re-building the model with the same arguments, ***no need to prune the trees***

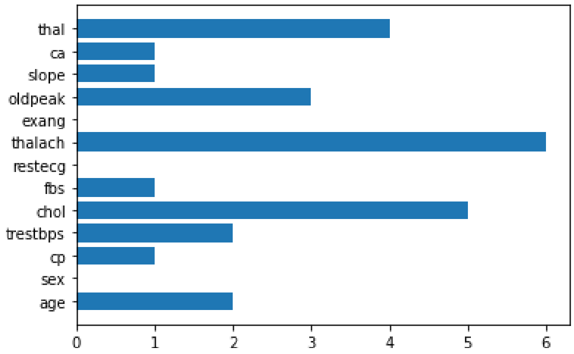
Note: You can find the best split threshold by both methods. First one: 1) Try N-1 threshold values, where the i-th threshold is the average of the i-th and (i+1)-th sorted values. Second one: Use the unique sorted value of the feature as the threshold to split

Hint: You can use the recursive method to build the nodes

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 importanc

e instead of the formula in the reference, more details on the sample code. **Matplotlib** is allowed to be used)

Ans:



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.

```
[25] # Fit model
     clf_10ab = Adaboost( n_estimators = 10 )
     clf_10ab.fit(train_df_data, train_df_target)

     from sklearn.metrics import roc_auc_score
     # Predict on test set
     pred_target = clf_10ab.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)

     print("AdaBoost Accuracy:", acc)

     AdaBoost Accuracy: 0.83
```

```
[27] # Fit model
     clf_100ab = Adaboost( n_estimators = 100 )
     clf_100ab.fit(train_df_data, train_df_target)

     from sklearn.metrics import roc_auc_score
     # Predict on test set
     pred_target = clf_100ab.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)

     print("AdaBoost Accuracy:", acc)

     AdaBoost Accuracy: 0.81
```

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.

```
[29] clf_10tree = RandomForest(n_estimators=10, max_features=np.sqrt(train_df_data.shape[1]))
     clf_10tree.fit(train_df_data, train_df_target)
     pred_target = clf_10tree.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)
     print('RandomForest accuracy: ',acc)

     RandomForest accuracy: 0.79
```

```
[37] clf_100tree = RandomForest(n_estimators=100, max_features=np.sqrt(train_df_data.shape[1]))
     clf_100tree.fit(train_df_data, train_df_target)
     pred_target = clf_100tree.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)
     print('RandomForest accuracy: ',acc)

     RandomForest accuracy: 0.8
```

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.

```
[38] clf_random_features = RandomForest(n_estimators=10, max_features=np.sqrt(train_df_data.shape[1]))
     clf_random_features.fit(train_df_data, train_df_target)
     pred_target = clf_random_features.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)
     print('RandomForest accuracy: ',acc)

     RandomForest accuracy: 0.74
```

```
[39] clf_all_features = RandomForest(n_estimators=10, max_features=train_df_data.shape[1])
     clf_all_features.fit(train_df_data, train_df_target)
     pred_target = clf_all_features.predict(test_df_data)
     acc = accuracy_score(test_df_target, pred_target)
     print('RandomForest accuracy: ',acc)

     RandomForest accuracy: 0.78
```

Note: Use majority votes to get the final prediction, you may get different results when re-building the random forest model

6. (30%) Tune the hyperparameter, perform feature engineering or implement more powerful ensemble methods to get a higher accuracy score. Screenshot your tests score on the report. Please note that only the ensemble method can be used. The neural network method is not allowed.

Accuracy	Your scores
acc > 0.85	30 points
0.8 < acc <= 0.85	25 points
0.7 < acc <= 0.8	20 points
acc < 0.7	0 points

51秒

```
[53] from sklearn.metrics import accuracy_score

     clf_gbc = GradientBoosting(n_estimators=100, learning_rate=0.1, max_depth=3)
     clf_gbc.fit(train_df_data, train_df_target)
     test_preds = clf_gbc.predict(test_df_data)

     print('Gradient Boosting Accuarcy score: ', accuracy_score(test_df_target, test_preds))
```

```
Gradient Boosting Accuarcy score:  0.8
```

4分鐘

```
[54]                 best_max_depth = max_depth

     best_learning_rate = 0.01
     for learning_rate in parameters["learning_rate"]:
         clf_gbc = GradientBoosting(n_estimators=best_n_estimators, learning_rate=learning_rate, max_depth=best_max_depth)
         clf_gbc.fit(train_df_data, train_df_target)
         test_preds = clf_gbc.predict(test_df_data)
         acc = accuracy_score(test_df_target, test_preds)
         if acc > max_acc_score:
             max_acc_score = acc
             best_learning_rate = learning_rate
     print('best Gradient Boosting accuracy: ',max_acc_score)
     print('best n estimators: ',best_n_estimators)
     print('best max depth: ',best_max_depth)
     print('best learning rate',best_learning_rate)
```

```
best Gradient Boosting accuracy:  0.81
best n estimators:  10
best max depth:  1
best learning rate 0.01
```

Part. 2, Questions (20%): 0819823 陳子祈

1. (10%) 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 (300, 100) at the first leaf node and (100, 300) 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 (200, 400) and (200, 0). **Evaluate the <u>misclassification rates</u> 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 and show that they are both lower for tree B than for tree A.** Define p_k to be the proportion of data points in region R assigned to class k, where k = 1, . . . , K

(A)

50% 50%
(400, 400)
C1 ↙ ↘ C2
75% 25% 25% 75%
(300, 100) (100, 300)

50% 50%
(400, 400)
C2 ↙ ↘ C1
33% 67% 100% 0%
(200, 400) (200, 0)

Misclassification rate

A left leaf node misclassification rate = $\frac{100}{800}$, A right leaf node misclassification rate = $\frac{100}{800}$

$$P_m(A) = \frac{100}{800} + \frac{100}{800} = \frac{1}{4}$$

B left leaf node misclassification rate = $\frac{200}{800}$, B right leaf node misclassification rate = 0

$$P_m(B) = \frac{200}{800} = \frac{1}{4} \quad \Rightarrow \quad P_m(A) = P_m(B)$$

Entropy information

$$I_E(D_{parent}) = -\left(0.5\, lg(0.5) + 0.5\, lg(0.5)\right) = 1$$

A: $I_E(D_{left}) = -\left(\frac{3}{4} lg\left(\frac{3}{4}\right) + \frac{1}{4} lg\left(\frac{1}{4}\right)\right) \approx 0.81$

$I_E(D_{right}) = -\left(\frac{1}{4} lg\left(\frac{1}{4}\right) + \frac{3}{4} lg\left(\frac{3}{4}\right)\right) \approx 0.81$

$I_{EA} = 0.81 + 0.81 = 1.62$

B: $I_E(D_{left}) = -\left(\frac{2}{6} lg\left(\frac{2}{6}\right) + \frac{4}{6} lg \frac{4}{6}\right) \approx 0.92$

$I_E(D_{right}) = 0$

$I_{EB} = 0.92 \quad \Rightarrow \quad I_{EB} < I_{EA}$

Gini information

$$I_G(D_{parent}) = 1-(0.5^2+0.5^2) = 0.5$$

A: $I_G(D_{left}) = 1-\left[\left(\frac{3}{4}\right)^2+\left(\frac{1}{4}\right)^2\right] = \frac{3}{8} = 0.375$

$I_G(D_{right}) = 1-\left[\left(\frac{1}{4}\right)^2+\left(\frac{3}{4}\right)^2\right] = \frac{3}{8} = 0.375$

$I_{GA} = 0.375 + 0.375 = 0.75$

B: $I_G(D_{left}) = 1-\left[\left(\frac{2}{6}\right)^2+\left(\frac{4}{6}\right)^2\right] = \frac{4}{9} = 0.\overline{4}$

$I_G(D_{right}) = 1-(1^2+0^2) = 0$

$I_{GB} = 0.\overline{4}$

$\Rightarrow \quad I_{GB} < I_{GA}$

2. (10%) By making a variational minimization of the expected exponential error function given by (1) with respect to all possible functions $y(x)$, show that the minimizing function is given by (2). Define t **is target variable** $\in$ **{−1, 1}, x is input vector.**

$$E_{x,t}\left[e^{-ty(x)}\right] = \sum_t \int e^{-ty(x)} p(t|x)p(x)\,dx \quad (1)$$

$$y(x) = \frac{1}{2} ln \frac{p(t=1|x)}{p(t=-1|x)} \quad (2)$$

$$\frac{\partial}{\partial y} E_{x,t}\left[e^{-ty}\right] = \frac{\partial}{\partial y} \sum_t \int e^{-ty} p(t|x)p(x)dx$$

$$= \frac{\partial}{\partial x} \int \left[e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x)\right] dx \frac{\partial x}{\partial y}$$

$$= \left[e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x)\right] \frac{\partial x}{\partial y}$$

令 $\left[e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x)\right] \frac{\partial x}{\partial y} = 0$

左右同時對 x 微分

$$\frac{\partial}{\cancel{\partial x}} \left[e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x)\right] \frac{\cancel{\partial x}}{\partial y} = 0$$

$$\frac{\partial}{\partial y} \left[e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x)\right] = 0$$

$$\Rightarrow -e^{-y} p(t=1|x)p(x) + e^{y} p(t=-1|x)p(x) = 0$$

$$\Rightarrow e^{y} p(t=-1|x)\cancel{p(x)} = e^{-y} p(t=1|x)\cancel{p(x)}$$

$$\Rightarrow e^{2y} = \frac{p(t=1|x)}{p(t=-1|x)}$$

$$\Rightarrow 2y = ln \frac{p(t=1|x)}{p(t=-1|x)} \rightarrow y = \frac{1}{2} ln\left(\frac{p(t=1|x)}{p(t=-1|x)}\right)$$

※