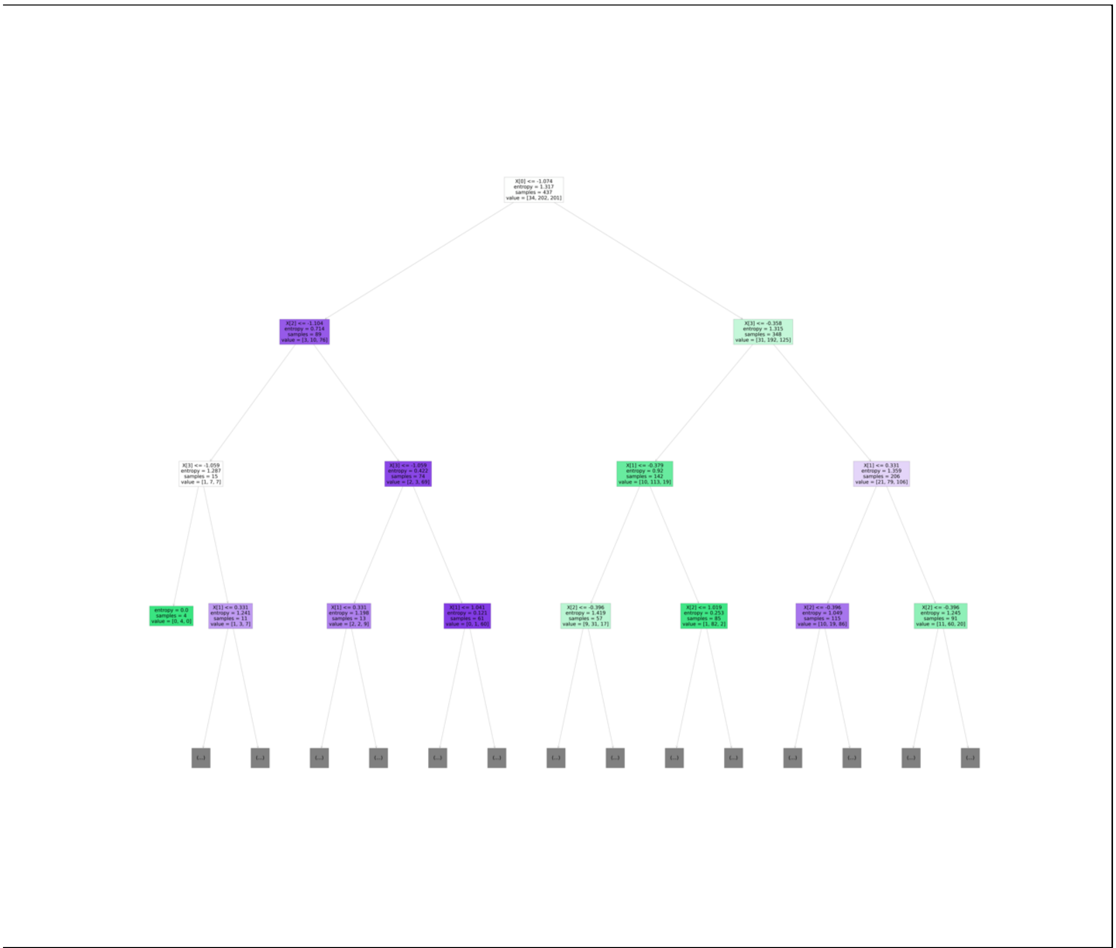
Exercise 2

109526016 鄭博謙

首先我先把相同的資料集放在 Jupyter 上利用 plot_tree 套件將資料轉換成以下的圖片，決策樹的執行順序是以 preorder 的方式建立起決策樹，以下會再依序介紹程式的執行順序及功能說明。



Function 實作

Implement the “_gini” function

```python
def _gini(self, sample_y, n_classes):
    total_num_sample = sample_y.size
    elements, counts = np.unique(sample_y, return_counts=True)
    gini = 0
    for i in counts:
        gini = gini + (i/total_num_sample)*(1-i/total_num_sample)
    return gini
```

Implement the “_entropy” function

```python
def _entropy(self, sample_y, n_classes):
    elements, counts = np.unique(sample_y, return_counts=True)
    entropy = np.sum([(-counts[i]/np.sum(counts))*np.log2(counts[i] /
    np.sum(counts)) for i in range(len(elements))])
    return entropy
```

以上兩個 function 依照公式去寫出來的

Implement the “_feature_split” function

此功能為找出最佳的 idx & thr 的值 再另外切割原資料的數據並放在左右子樹繼續執行。

```python
def _feature_split(self, X, y, n_classes, node):
        m = y.size
        if m <= 1:
            return None, None

        # Gini or Entropy of current node.
        if self.criterion == "gini":
            best_criterion = self._gini(y, n_classes)
        else:
            best_criterion = self._entropy(y, n_classes)

        best_idx, best_thr = None, None

        information = []
        for j in range(4):
```

```
            tem_train_scale = X
            index = np.argsort(tem_train_scale[:, j])
            sort_x = tem_train_scale[:, j][index]
            sort_y = y[index]

            find_midpoint = []
            for i in range(len(index)-1):
                if(sort_x[i] < sort_x[i+1]):
                    midpoint = (sort_x[i]+sort_x[i+1])/2
                    find_midpoint.append([round(midpoint, 3), i+1])

            if(len(find_midpoint) == 0):
                continue
            for i in range(len(find_midpoint)):
                left = sort_y[:find_midpoint[i][1]]
                right = sort_y[find_midpoint[i][1]:]
                if self.criterion == "gini":
                    left_criterion_value = round(
                        self._gini(left, n_classes), 3)
                    right_criterion_value = round(
                        self._gini(right, n_classes), 3)
                    information.append(
                        [left_criterion_value, right_criterion_value,
find_midpoint[i][1], j,find_midpoint[i][0]])
                else:
                    left_criterion_value = round(
                        self._entropy(left, n_classes), 3)
                    right_criterion_value = round(
                        self._entropy(right, n_classes), 3)
                    information.append(
                        [left_criterion_value, right_criterion_value,
find_midpoint[i][1], j,find_midpoint[i][0]])
        # 找最小的entropy or gini index
        return_information = []
        data = []
        sample = m
        for i in range(len(information)):
            left_sample = information[i][2]
```

```
            right_sample = sample - left_sample
            left_entropy = information[i][0]
            right_entropy = information[i][1]
            result = (left_sample/sample)*left_entropy +
(right_sample/sample)*right_entropy
            data.append(result)

        for i in range(len(information)):
            if min(data) == data[i]:
                for j in range(len(information[i])):
                    return_information.append(information[i][j])

        best_idx, best_thr = return_information[3], return_information[4]

        index_result = np.argsort(tem_train_scale[:, return_information[3]])
        sort_x_result = tem_train_scale[index_result]
        sort_y_result = y[index_result]

        # 返回左右子樹的 X_train_scale && y_train data
        node.left_X_train_scale = sort_x_result[:return_information[2]]
        node.right_X_train_scale = sort_x_result[return_information[2]:]
        node.left_y_train = sort_y_result[:return_information[2]]
        node.right_y_train = sort_y_result[return_information[2]:]

        if best_thr == 0 or (return_information[0] ==0 and
return_information[1] ==0):
            print('This node is leaf')
            print('return pre node')
            node.leaf = True
            return None, None
        else:
            print('find idx is : {0}, thr is : {1}'.format(best_idx,
best_thr))
            print('Left node ' + self.criterion + ' is : '+
str(return_information[0]))
            print('Right node ' + self.criterion + ' is : '+
str(return_information[1]))
        return best_idx, best_thr
```

Implement the “_build_tree” function

建立樹用遞迴的概念建立樹的 node，並且當深度超過預設值後將停止建立 node

```
def _build_tree(self, X, y, depth=2):

    print('bulid new node')
    num_samples_per_class = [np.sum(y == i)for i in range(self.n_classes_)]
    print('num_samples_per_class', num_samples_per_class)
    # 暫且不明白要做啥
    predicted_class = np.argmax(num_samples_per_class)

    node = Node(
        gini=self._gini(y, self.n_classes_),
        entropy=self._entropy(y, self.n_classes_),
        num_samples=y.size,
        num_samples_per_class=num_samples_per_class,
        predicted_class=predicted_class,
    )

    if depth < self.max_depth:
        node.leaf = False
        self.n_classes_ = len(np.unique(y))
        idx, thr = self._feature_split(
            X, y, self.n_classes_, node)
        node.feature_index = idx
        node.threshold = thr

        if idx is not None:
            print('\n')
            print('bulid left tree <-----')
            node.left = self._build_tree(
                node.left_X_train_scale, node.left_y_train, depth+1)
            print('\n')
            print('bulid right tree ----->')
```

```
            node.right = self._build_tree(
                node.right_X_train_scale, node.right_y_train, depth+1)
      else:
          print('depth >= max_depth')

      return node
```

Implement the “predict” function

實作預測的 function，最後返回 pred[] 再透過 sklearn.metrics.accuracy_score 來計算準確度

```
def predict(self, X):
      predicted_class = self.predict_class_fun(X)
      pred = predicted_class[:,0]

      return pred
   def predict_class_fun(self,X):
      predicted_class = np.empty((X.shape[0],self.n_classes_))
      for i in range(X.shape[0]):
         predicted_class[i] = self.predict_row(X[i,:],self.tree_)
      return predicted_class
   def predict_row(self,row,tree_):
      """Predict single row"""
      if tree_.leaf:
         return tree_.predicted_class
      else:
         if row[tree_.feature_index]<=tree_.threshold:
            return self.predict_row(row,tree_.left)
         else:
            return self.predict_row(row,tree_.right)
```

我有調整 Class Node 的初始值方便我利用物件導向的概念

```
def __init__(self, gini, entropy, num_samples,
              num_samples_per_class, predicted_class):
      self.gini = gini
```

```
        self.entropy = entropy
        self.num_samples = num_samples
        self.num_samples_per_class = num_samples_per_class
        self.predicted_class = predicted_class
        self.feature_index = 0
        self.threshold = 0
        self.left = None
        self.right = None
        ## 以下是新增的部分存取左右子樹所需要的 X_train_scale & y_train 的 data 以及判
斷此節點是否是 leaf
        self.leaf = bool
        self.left_X_train_scale = []
        self.right_X_train_scale = []
        self.left_y_train = []
        self.right_y_train = []
```

執行過程以下以 entropy 作為範例

```
Bulid Decision Tree by entropy
bulid new node
num_samples_per_class [34, 202, 201]
find idx is : 0, thr is : -1.074
Left node entropy is : 0.714
Right node entropy is : 1.315


bulid left tree <-----
bulid new node
num_samples_per_class [3, 10, 76]
find idx is : 2, thr is : -1.104
Left node entropy is : 1.287
Right node entropy is : 0.422


bulid left tree <-----
bulid new node
num_samples_per_class [1, 7, 7]
find idx is : 3, thr is : -1.059
Left node entropy is : 0.0
Right node entropy is : 1.241


bulid left tree <-----
bulid new node
num_samples_per_class [0, 4, 0]
This node is leaf
return pre node
```

```
bulid right tree ----->
bulid new node
num_samples_per_class [1]
find idx is : 1, thr is : 0.331
Left node entropy is : 0.592
Right node entropy is : 0.811


bulid left tree <-----
bulid new node
num_samples_per_class [1, 0, 6]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [0, 3, 1]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [2, 3, 69]
find idx is : 3, thr is : -1.059
Left node entropy is : 1.198
Right node entropy is : 0.121


bulid left tree <-----
bulid new node
num_samples_per_class [2, 2, 9]
find idx is : 1, thr is : 0.331
Left node entropy is : 0.722
Right node entropy is : 0.918


bulid left tree <-----
bulid new node
num_samples_per_class [2, 0, 8]
depth >= max_depth
```

```
bulid right tree ----->
bulid new node
num_samples_per_class [0, 2, 1]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [0, 1, 60]
find idx is : 1, thr is : 1.041
Left node entropy is : 0.0
Right node entropy is : 0.439


bulid left tree <-----
bulid new node
num_samples_per_class [0, 0]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [0, 1]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [31, 192]
find idx is : 3, thr is : -0.358
Left node entropy is : 0.92
Right node entropy is : 1.359


bulid left tree <-----
bulid new node
num_samples_per_class [10, 113, 19]
find idx is : 1, thr is : -0.379
Left node entropy is : 1.419
Right node entropy is : 0.253
```

```
bulid left tree <-----
bulid new node
num_samples_per_class [9, 31, 17]
find idx is : 2, thr is : -0.396
Left node entropy is : 0.832
Right node entropy is : 1.483


bulid left tree <-----
bulid new node
num_samples_per_class [3, 18, 1]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [6, 13, 16]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [1, 82, 2]
find idx is : 2, thr is : 1.019
Left node entropy is : 0.109
Right node entropy is : 0.544


bulid left tree <-----
bulid new node
num_samples_per_class [1, 68, 0]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [0, 14, 2]
depth >= max_depth
```

```
bulid right tree ----->
bulid new node
num_samples_per_class [21, 79, 106]
find idx is : 1, thr is : 0.331
Left node entropy is : 1.049
Right node entropy is : 1.245


bulid left tree <-----
bulid new node
num_samples_per_class [10, 19, 86]
find idx is : 2, thr is : -0.396
Left node entropy is : 1.465
Right node entropy is : 0.471


bulid left tree <-----
bulid new node
num_samples_per_class [6, 17, 16]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [4, 2, 70]
depth >= max_depth


bulid right tree ----->
bulid new node
num_samples_per_class [11, 60, 20]
find idx is : 2, thr is : -0.396
Left node entropy is : 0.292
Right node entropy is : 1.489


bulid left tree <-----
bulid new node
num_samples_per_class [2, 37, 0]
depth >= max_depth
```

```
bulid right tree ----->
bulid new node
num_samples_per_class [9, 23, 20]
depth >= max_depth

---start predict----

entropy tree train accuracy: 0.656751
entropy tree test accuracy: 0.638298


--------finish--------
```

以上執行過程分別對應到最上方的決策樹，直到 depth>=max_depth or 已經分類完無法再繼續分類下去，最後輸出 accuracy。

以下範例更改 criterion & max_depth 的值都會影響到準確度。

```
print('\n--------start--------\n')
# gini tree
print('Bulid Decision Tree by gini')
accuracy_report(X_train_scale, y_train, X_test_scale,
                y_test, criterion='gini', max_depth=2)
# entropy tree
print('Bulid Decision Tree by entropy')
accuracy_report(X_train_scale, y_train, X_test_scale,
                y_test, criterion='entropy', max_depth=2)

print('\n--------finish--------\n')
```

```
---start predict----

gini tree train accuracy: 0.462243
gini tree test accuracy: 0.457447

Bulid Decision Tree by entropy
bulid new node
num_samples_per_class [34, 202, 201]
depth >= max_depth

---start predict----

entropy tree train accuracy: 0.462243
entropy tree test accuracy: 0.457447


--------finish--------
```

```
print('\n--------start--------\n')
# gini tree
print('Bulid Decision Tree by gini')
accuracy_report(X_train_scale, y_train, X_test_scale,
                y_test, criterion='gini', max_depth=3)
# entropy tree
print('Bulid Decision Tree by entropy')
accuracy_report(X_train_scale, y_train, X_test_scale,
                y_test, criterion='entropy', max_depth=3)

print('\n--------finish--------\n')
```

```
---start predict----

gini tree train accuracy: 0.613272
gini tree test accuracy: 0.537234
```

```
---start predict----

entropy tree train accuracy: 0.613272
entropy tree test accuracy: 0.537234


--------finish--------
```