HW6 Report

인대영 (2022311154)
2022 - 10 - 17

1. We first prompt the user to enter the data sets. We then define a function for calculating the GINI impurity according to the CART splitting rule.

$$imp(t) = 1 - \sum_j p_j^2$$

GINI Impurity

In [8]:
```python
response=np.unique(y_train)
```

In [9]:
```python
def GINI_impurity(data):
    n=len(data)
    p1 = sum(data == response[0])/n
    p2 = sum(data == response[1])/n
    gini = 1 - (p1**2+p2**2)

    return gini
```

In [10]:
```python
gini_train = GINI_impurity(y_train)
np.round(gini_train,4)
```

Out[10]: 0.4082

In [11]:
```python
gini_test = GINI_impurity(y_test)
np.round(gini_test,4)
```

Out[11]: 0.4735

We then implement the greedy search algorithm as shown below. We create lists to store the best goodness of split values and their respective indices. The weights for updating the splits are updated for each iteration of the variables.

$$\Delta i(s^*, i) = \max_{s \in S} \Delta i(s, 1) \quad \text{where } s: \text{goodness of split}$$

In [16]:
```python
for i in range(len(x_train.T)):
    xmeans = [] #List for storing splitting points
    for j in range(len(np.unique(x_train[i]))-1):
        xmeans.append((np.unique(x_train[i])[j] + np.unique(x_train[i])[j+1])/2) #Caloulate splitting points

    #Caloulate goodness of split
    goodness = [] #Store goodness of split values
    for k in range(len(xmeans)):
        #apply split rule
        left = x_train[i] <= xmeans[k]
        right = x_train[i] > xmeans[k]

        #weights for GINI impurity
        w1 = sum(left)/len(x_train)
        w2 = sum(right)/len(x_train)

        splits=GINI_impurity(y_train) - (w1*GINI_impurity(y_train[left]) + w2*GINI_impurity(y_train[right]))
        goodness.append(splits)

    #Save values
    max_goodness.append(max(goodness))
    max_index.append(np.argmax(goodness))

    #Finding variable with best goodness of split
    variable = str(np.argmax(max_goodness)+1)

    #Coresponding split point
    split_point = (np.unique(x_train[np.argmax(max_goodness)])[max_index[np.argmax(max_goodness)]] + np.unique(x_train[np.argmax(max_goo
```

In [18]:
```python
#Best goodness of split values
np.round(max_goodness,4)
```

Out[18]: array([0.055 , 0.0953, 0.0137, 0.022 , 0.026 , 0.0337, 0.0471])

In [19]:
```python
#Corresponding indices of observations
max_index
```

Out[19]: [6, 68, 19, 11, 46, 151, 7]

In [20]:
```python
#Splitting variable
variable
```

Out[20]: '2'

In [21]:
```python
#Final splitting point
split_point
```

Out[21]: 146.5

We can see from the result that the splitting point is at x2 = 146.5. Using the model, we fit the test data and compare the predictions with the actual response variables of the test data via confusion matrix. The accuracy is calculated as 0.73. The tree structure along with the confusion matrix and accuracy of predictions are summarized in the output file as shown below.

HW6_output - 메모장

파일 편집 보기

```
Tree Structure
      Node 1: 2 (72, 180)
          Node 2: x2 <= 146.5, 2 (34, 164)
          Node 3: x2 > 146.5, 1 (38, 16)

Confusion Matrix (Test)
------------------------------
      Predicted Class
           1    2
Actual  1  37   50
Class   2  11  128

Model Summary (Test)
----------------------------
Overall accuracy = 0.73
```