HW7

2021321148 INSIK CHO

□

```
def decisiontree( X, Y, method = 'CART'):
  if method == 'CART':
```

method = ‘CART’를 입력하면 CART로 one-level tree가 되고, 그 외 아무거나 입력 시 C4.5가 되도록 함수를 정의했습니다.

□ cart

```
if method == 'CART':
    A=[]
    B = np.unique(Y)
    for i in range(0,len(X.iloc[0])):
      if type(X.iloc[0,i]) == str:
        options = []
        for L in range(1,len(np.unique(X.iloc[:,i]))):
          for subset in itertools.combinations(np.unique(X.iloc[:,i]),L):
            subset = list(subset)
            options.append(subset)
        A.append(len(options))
      else:
        A.append(len(X.iloc[:,i].sort_values().unique()))
    c = np.max(A)
    K = pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    l = pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    number =pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    for i in range(0,len(X.iloc[0])):
      Z=[]
      if type(X.iloc[0,i]) == str:
        options = []
        for L in range(1,len(np.unique(X.iloc[:,i]))):
          for subset in itertools.combinations(np.unique(X.iloc[:,i]),L):
            subset = list(subset)
            options.append(subset)
        for j in range(0,len(options)):
          condition = X.iloc[:,i].isin(options[j])
          l = len(Y[condition==True][Y[condition==True]==np.unique(Y)[1]])
          g = len(Y[condition==True])/len(Y)*gini(Y[condition==True])  + len(Y[condition==False])/len(Y)*gini(Y[condition==False])
          s = gini(Y) - g
          K[j][i] = s
          l[j][i] = np.array([options[j], list(np.delete(np.unique(X.iloc[:,i]),np.where(np.unique(X.iloc[:,i])==options[j])))]).flatten().tolist()
          number[j][i] = [[l,len(Y[condition == True])-l ], [len(Y[Y==np.unique(Y)[1]])-l, len(Y[Y==np.unique(Y)[0]])-(len(Y[condition == True])-l) ]]

      else:
        for j in range(0,len(X.iloc[:,i].sort_values().unique())-1):
          z1 = X.iloc[:,i].sort_values().unique()[j]
          z2 = X.iloc[:,i].sort_values().unique()[j+1]
          Z.append((z1+z2)/2)

          for k in range(0,len(Z)):
            condition = X.iloc[:,i]<=Z[k]
            l = len(Y[condition==True][Y[condition==True]==np.unique(Y)[1]])
            g = len(Y[condition==True])/len(Y)*gini(Y[condition==True])  + len(Y[condition==False])/len(Y)*gini(Y[condition==False])
            s = gini(Y) - g
            K[k][i] = s
            l[k][i] = Z[k]
            number[k][i] = [[l,len(Y[condition == True]-l) ], [len(Y[Y==np.unique(Y)[1]])-l, len(Y[Y==np.unique(Y)[0]])-(len(Y[condition == True])-l) ]]
```

cart로 진행시 위와 같은 코드로 결과가 도출되고,

□C4.5

```
else:
    A=[]
    B = np.unique(Y)
    for i in range(0,len(X.iloc[0])):
      if type(X.iloc[0,i]) == str:
        A.append(1)
      else:
        A.append(len(X.iloc[:,i].sort_values().unique()))
    c = np.max(A)
    K = pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    I = pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    number =pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
    e = len(Y[Y==np.unique(Y)[1]])/len(Y)
    entropyY = -e*np.log2(e) -(1-e)*np.log2(1-e)

    for i in range(0,len(X.iloc[0])):
      if type(X.iloc[0,i]) == str:
        p=[]
        n=[]
        num=[]
        for j in range(0,len(np.unique(X.iloc[:,i]))):
          condition = (X.iloc[:,i]==np.unique(X.iloc[:,i])[j])
          l = len(Y[condition==True][Y[condition==True]==np.unique(Y)[1]])
          p1 = l/len(Y[condition==True])
          p.append((-p1*np.log2(p1) -(1-p1)*np.log2(1-p1))*(len(Y[condition==True])/(len(Y))))
          n.append(-(len(Y[condition==True])/len(Y))*np.log2(len(Y[condition==True])/len(Y)))
          num.append([l,len(Y[condition == True])-l ])
        K[0][i] = (entropyY-np.sum(p))/np.sum(n)
        I[0][i] = np.unique(X.iloc[:,i])
        number[0][i] = num

      else:
        Z=[]
        for j in range(0,len(X.iloc[:,i].sort_values().unique())-1):
          z1 = X.iloc[:,i].sort_values().unique()[j]
          z2 = X.iloc[:,i].sort_values().unique()[j+1]
          Z.append((z1+z2)/2)

          for k in range(0,len(Z)):
            condition = X.iloc[:,i]<=Z[k]
            l = len(Y[condition==True][Y[condition==True]==np.unique(Y)[1]])
            p1 = len(Y[condition==True][Y[condition==True]==np.unique(Y)[1]])/len(Y[condition==True])
            g = len(Y[condition==True])/len(Y)*(-p1*np.log2(p1))  + len(Y[condition==False])/len(Y)*(-(1-p1)*np.log2(1-p1))
            n= -len(Y[condition==True])/len(Y)*np.log2(len(Y[condition==True])/len(Y)) -(1-len(Y[condition==True])/len(Y))*n
            K[k][i] = (entropyY - g)/n
            I[k][i] = Z[k]
            number[k][i] = [[l,len(Y[condition == True]-l) ], [len(Y[Y==np.unique(Y)[1]])-l, len(Y[Y==np.unique(Y)[0]])-(len
```

C4.5는 위와 같은 코드를 따라 결과를 도출합니다.

□

```
print('tree structure (', method, ')')
if len(Y[Y==np.unique(Y)[1]]) >= len(Y)/2:
  print( 'node 1 : yes  (', len(Y[Y==np.unique(Y)[1]]),',', len(Y) - len(Y[Y==np.unique(Y)[1]]), ')')
else:
  print( 'node 1 : no  (', len(Y[Y==np.unique(Y)[1]]),',', len(Y) - len(Y[Y==np.unique(Y)[1]]), ')')
for i in range(0,len(l[idx_c][idx_r])):
  if number[idx_c][idx_r][i][0]>= number[idx_c][idx_r][i][1]:
    print(f'node {i+2} :' 'class = ', l[idx_c][idx_r][i], ' yes (',number[idx_c][idx_r][i][0], ',' , number[idx_c
  else:
    print(f'node {i+2} :' 'class = ', l[idx_c][idx_r][i], ' no (',number[idx_c][idx_r][i][0], ',' , number[idx_c]

Y2= []
if number[idx_c][idx_r][0][0]>= number[idx_c][idx_r][0][1]:
  for i in range(0,len(Y)):
    if X.iloc[:,idx_r][i] == l[idx_c][idx_r][0]:
      Y2.append(np.unique(Y)[1])
    else:
      Y2.append(np.unique(Y)[0])
else:
    if X.iloc[:,idx_r][i] == l[idx_c][idx_r][1]:
      Y2.append(np.unique(Y)[1])
    else:
      Y2.append(np.unique(Y)[0])

confusion_tst = confusion_matrix(Y,Y2)
accu_tst = 0
for i in range(len(np.unique(Y))):
  accu_tst = accu_tst + confusion_tst[i][i]
accuracy_tst = accu_tst / X.shape[0]

print('\n\nconfusion matrix (train)')
print('----------------------------------')
print('          predicted class \n Actual 1 ' ,confusion_tst[0], '\n class  2 ', confusion_tst[1])
for i in range(2,len(np.unique(Y)) ):
  print(f'        {i+1} ', confusion_tst[i])
print('model summary')
print('----------------------------------')
print('Overall accuracy = ' ,accuracy_tst)
```

위의 코드를 통해 결과도출 까지 이루어지도록 작성했습니다.

출력 결과는 다음과 같습니다.

□cart

```
tree structure ( CART )
node 1 : no  ( 711 , 1490 )
node 2 :class =  Female  yes ( 344 , 126 )
node 3 :class =  Male  no ( 367 , 1364 )


confusion matrix (train)
----------------------------------
| | | | | |predicted class
 Actual 1  [1364  126]
 class  2  [367 344]
model summary
----------------------------------
Overall accuracy =  0.7760109041344844
```

□C4.5

```
tree structure ( C4_5 )
node 1 : no  ( 711 , 1490 )
node 2 :class =  Female  yes ( 344 , 126 )
node 3 :class =  Male  no ( 367 , 1364 )


confusion matrix (train)
----------------------------------
| | | | | |predicted class
 Actual 1  [1364  126]
 class  2  [367 344]
model summary
----------------------------------
Overall accuracy =  0.7760109041344844
```

둘 다 동일하게, 성별을 기준으로 분류가 되는 결과가 나왔습니다.