HW8

2021321148
Insik Cho

1. Regression

```
# Decision Regressor
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))
I = pd.DataFrame(index = range(0,len(X.iloc[0])), columns = range(0,c))
M = 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 is_numeric_dtype(X.iloc[:,i]) == False :
    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])
      bary1= np.sum(Y[condition == True])/len(Y[condition==True])
      bary2= np.sum(Y[condition == False])/len(Y[condition==False])
      mse1 = np.sum((Y[condition==True] - bary1)**2)/len(Y[condition==True])
      mse2 = np.sum((Y[condition==False] - bary2)**2)/len(Y[condition==False])
      mse = mse1 * (len(Y[condition==True])/len(Y)) + mse2 * (len(Y[condition==False])/len(Y))
      K[j][i] = -mse
      I[j][i] = np.array([options[j], list(np.delete(np.unique(X.iloc[:,i]),np.where(np.unique(
      number[j][i] = len(Y[condition==True])
      M[j][i] = [bary1, bary2]

  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]
      bary1= np.sum(Y[condition == True])/len(Y[condition==True])
      bary2= np.sum(Y[condition == False])/len(Y[condition==False])
      mse1 = np.sum((Y[condition==True] - bary1)**2)/len(Y[condition==True])
      mse2 = np.sum((Y[condition==False] - bary2)**2)/len(Y[condition==False])
      mse = mse1 * (len(Y[condition==True])/len(Y)) + mse2 * (len(Y[condition==False])/len(Y))
      K[k][i] = -mse
      I[k][i] = Z[k]
```

```
K = K.astype(float)
idx = []
col = K.idxmax(axis=1)
for i in range(0,len(X.iloc[0])):
    idx.append(K[col[i]][i])
idx_r = np.argmax(idx) # 첫 노드를 나누는 변수
idx_c = col[idx_r]


print('tree structure ')
print( 'node 1 :', 'n = ',len(Y), 'mean = ', np.mean(Y) )
if is_numeric_dtype(X.iloc[:,idx_r]) == True:
  print('node 2: ', X.columns[idx_r], '<=', l[idx_c][idx_r], ', n = ', number[idx_c][idx_r], ', mean = ', M[idx_c][idx_r][0])
  print('node 3: ', X.columns[idx_r], '>', l[idx_c][idx_r], ', n = ', len(Y)- number[idx_c][idx_r], ', mean = ', M[idx_c][idx_r][1])

else:
  print('node 2: ', X.columns[idx_r], '=', l[idx_c][idx_r][0], ', n = ', number[idx_c][idx_r], ', mean = ', M[idx_c][idx_r][0])
  print('node 3: ', X.columns[idx_r], '=', l[idx_c][idx_r][1], ', n = ', len(Y)- number[idx_c][idx_r], ', mean = ', M[idx_c][idx_r][1])
```

위와 같이 Decision tree regression을 하는 코드를 작성하였고, 주어진 boston_tr 데이터를 이용한 결과는 다음과 같습니다.

```
tree structure
node 1 : n =  343 mean =  21.74344023323615
node 2:  lstat <= 10.14 , n =  150 , mean =  27.656666666666666
node 3:  lstat > 10.14 , n =  193 , mean =  17.147668393782382
```

2. fitting test data

```
# applying to test data

def tst(X,Y):
  if is_numeric_dtype(X.iloc[:,idx_r]) == True:
    condition = X.iloc[:,idx_r] <= l[idx_c][idx_r]
    bary1= np.sum(Y[condition == True])/len(Y[condition==True])
    bary2= np.sum(Y[condition == False])/len(Y[condition==False])
    mse1 = np.sum((Y[condition==True] - bary1)**2)/len(Y[condition==True])
    mse2 = np.sum((Y[condition==False] - bary2)**2)/len(Y[condition==False])
    mae1 = np.sum(np.abs(Y[condition==True] - bary1))/len(Y[condition==True])
    mae2 = np.sum(np.abs(Y[condition==False] - bary2))/len(Y[condition==False])
    mape1 = np.sum(np.abs(Y[condition==True] - bary1)/Y[condition == True])/len(Y[condition==True])
    mape2 = np.sum(np.abs(Y[condition==False] - bary2)/Y[condition == False])/len(Y[condition==False])
    mse = mse1 * (len(Y[condition==True])/len(Y)) + mse2 * (len(Y[condition==False])/len(Y))
    mae = mae1 * (len(Y[condition==True])/len(Y)) + mae2 * (len(Y[condition==False])/len(Y))
    mape = mape1 * (len(Y[condition==True])/len(Y)) + mape2 * (len(Y[condition==False])/len(Y))
    print('rMSE = ', np.sqrt(mse))
    print('MAE = ', mae)
    print('MAPE = ', mape)
  else:
    condition = X.iloc[:,idx_r].isin(l[idx_c][idx_r][0])
    bary1= np.sum(Y[condition == True])/len(Y[condition==True])
    bary2= np.sum(Y[condition == False])/len(Y[condition==False])
    mse1 = np.sum((Y[condition==True] - bary1)**2)/len(Y[condition==True])
    mse2 = np.sum((Y[condition==False] - bary2)**2)/len(Y[condition==False])
    mae1 = np.sum(np.abs(Y[condition==True] - bary1))/len(Y[condition==True])
    mae2 = np.sum(np.abs(Y[condition==False] - bary2))/len(Y[condition==False])
    mape1 = np.sum(np.abs(Y[condition==True] - bary1)/Y[condition == True])/len(Y[condition==True])
    mape2 = np.sum(np.abs(Y[condition==False] - bary2)/Y[condition == False])/len(Y[condition==False])
    mse = mse1 * (len(Y[condition==True])/len(Y)) + mse2 * (len(Y[condition==False])/len(Y))
    mae = mae1 * (len(Y[condition==True])/len(Y)) + mae2 * (len(Y[condition==False])/len(Y))
    mape = mape1 * (len(Y[condition==True])/len(Y)) + mape2 * (len(Y[condition==False])/len(Y))
    print('rMSE = ', np.sqrt(mse))
    print('MAE = ', mae)
    print('MAPE = ', mape)
```

위와 같이 rmse, mae, mape를 도출하는 코드를 작성하였고,

```
tst(tstX,tstY)
```

```
rMSE =  6.2942365503065405
MAE =  5.0444363459669574
MAPE =  0.29468029638740356
```

다음의 결과를 얻었습니다.