**Week 9 class: Programming Homework – Best Solution**

**이준희, 조현식, 차원우, 이민아**

**# Homework 1 - Decision Tree**

**# source code**

import numpy as np

import pandas as pd

# entropy calculator (target\_feature: this node feature\_name / target\_value: one of category value in target\_feature / parent\_feature: parent node feature\_name / parent\_value: one of category value in parent\_feature)

def entr(target\_feature, target\_value=None, parent\_feature=None, parent\_value=None):

# if root node

if target\_feature == 'Outcome':

# calculate total, responded count, not responded count

total = df[target\_feature].count()

responded = df[target\_feature][df['Outcome']=='Responded'].count()

notResponded = df[target\_feature][df['Outcome']=='Not responded'].count()

# not root node

else:

# extract dataframe

if parent\_feature != None and parent\_value != None:

ext\_df = df[df[parent\_feature]==parent\_value]

else:

ext\_df = df

ext\_df = ext\_df[ext\_df[target\_feature]==target\_value]

# calculate total, responded count, not responded count

total = ext\_df[target\_feature][ext\_df[target\_feature]==target\_value].count()

responded = ext\_df[target\_feature][ext\_df['Outcome']=='Responded'].count()

notResponded = ext\_df[target\_feature][ext\_df['Outcome']=='Not responded'].count()

# calculate entropy

# check np.log2(0)

if responded == 0 or notResponded == 0:

entr = 0

else:

entr = -(responded/total\*np.log2(responded/total) + notResponded/total\*np.log2(notResponded/total))

return round(entr,2),total

# add dataset

df = pd.DataFrame({'District':['Suburban','Suburban','Rural','Urban','Urban','Urban','Rural','Suburban','Suburban','Urban','Suburban','Rural','Rural','Urban'],

'House Type':['Detached','Detached','Detached','Semi-detached','Semi-detached','Semi-detached','Semi-detached','Terrace','Semi-detached','Terrace','Terrace','Terrace','Detached','Terrace'],

'Income':['High','High','High','High','Low','Low','Low','High','Low','Low','Low','High','Low','High'],

'Previous Customer':['No','Yes','No','No','No','Yes','Yes','No','No','No','Yes','Yes','No','Yes'],

'Outcome':['Not responded','Not responded','Responded','Responded','Responded','Not responded','Responded','Not responded','Responded','Responded','Responded','Responded','Responded','Not responded']})

# data exploration

print(df.info())

print(df,end="\n\n")

# 1. Calculate information gain (first attribute(root))

# calculate root entropy

root\_entr,root\_total = entr('Outcome')

print('Entropy(root):',round(root\_entr,2))

print()

# calculate 'District' (Suburban,Rural,Urban)

suburban\_entr,suburban\_total = entr('District','Suburban')

rural\_entr, rural\_total = entr('District','Rural')

urban\_entr, urban\_total = entr('District','Urban')

print('Entropy(Suburban/Rural/Urban):',suburban\_entr,'/',rural\_entr,'/',urban\_entr)

print('\*Rural entropy is 0. So Stop split in Rural')

# calculate information gain

district\_info = root\_entr-(suburban\_total\*suburban\_entr + rural\_total\*rural\_entr + urban\_total\*urban\_entr)/root\_total

print('Information gain(District):',round(district\_info,2))

print()

# calculate 'House Type' (Detached,Semi-detached,Terrace)

detached\_entr,detached\_total = entr('House Type','Detached')

semi\_detached\_entr, semi\_detached\_total = entr('House Type','Semi-detached')

terrace\_entr, terrace\_total = entr('House Type','Terrace')

print('Entropy(Detached/Semi-detached/Terrace):',detached\_entr,'/',semi\_detached\_entr,'/',terrace\_entr)

# calculate information gain

houseType\_info = root\_entr-(detached\_total\*detached\_entr + semi\_detached\_total\*semi\_detached\_entr + terrace\_total\*terrace\_entr)/root\_total

print('Information gain(House Type):',round(houseType\_info,2))

print()

# calculate 'Income' (High,Low)

high\_entr,high\_total = entr('Income','High')

low\_entr,low\_total = entr('Income','Low')

print('Entropy(High/Low):',high\_entr,'/',low\_entr)

# calculate information gain

income\_info = root\_entr-(high\_total\*high\_entr + low\_total\*low\_entr)/root\_total

print('Information gain(Income):',round(income\_info,2))

print()

# calculate 'Previous Customer' (No,Yes)

no\_entr,no\_total = entr('Previous Customer','No')

yes\_entr,yes\_total = entr('Previous Customer','Yes')

print('Entropy(No/Yes):',no\_entr,'/',yes\_entr)

# calculate information gain

previousCustomer\_info = root\_entr-(no\_total\*no\_entr + yes\_total\*yes\_entr)/root\_total

print('Information gain(Previous Customer):',round(previousCustomer\_info,2))

print()

# root selection result

print('Root attribute: District (',round(district\_info,2),')')

print('-'\*50,end="\n\n")

# 2-1. Calculate information gain (Suburban -> second attribute)

# calculate root entropy (Suburban)

root\_entr,root\_total = entr('District','Suburban')

print('Entropy(root):',round(root\_entr,2))

print()

# calculate 'House Type' (Detached,Semi-detached,Terrace)

detached\_entr,detached\_total = entr('House Type','Detached','District','Suburban')

semi\_detached\_entr,semi\_detached\_total = entr('House Type','Semi-detached','District','Suburban')

terrace\_entr,terrace\_total = entr('House Type','Terrace','District','Suburban')

print('Entropy(Detached/Semi-detached/Terrace):',detached\_entr,'/',semi\_detached\_entr,'/',terrace\_entr)

print('\*Detached entropy is 0. So Stop split in Detached')

print('\*Semi-detached entropy is 0. So Stop split in Semi-detached')

# calculate information gain

houseType\_info = root\_entr-(detached\_total\*detached\_entr + semi\_detached\_total\*semi\_detached\_entr + terrace\_total\*terrace\_entr)/root\_total

print('Information gain(House Type):',round(houseType\_info,2))

print()

# calculate 'Income' (High,Low)

high\_entr,high\_total = entr('Income','High','District','Suburban')

low\_entr,low\_total = entr('Income','Low','District','Suburban')

print('Entropy(High/Low):',high\_entr,'/',low\_entr)

print('\*High entropy is 0. So Stop split in High')

print('\*Low entropy is 0. So Stop split in Low')

# calculate information gain

income\_info = root\_entr-(high\_total\*high\_entr + low\_total\*low\_entr)/root\_total

print('Information gain(Income):',round(income\_info,2))

print()

# calculate 'Previous Customer' (No,Yes)

no\_entr,no\_total = entr('Previous Customer','No','District','Suburban')

yes\_entr,yes\_total = entr('Previous Customer','Yes','District','Suburban')

print('Entropy(No/Yes):',no\_entr,'/',yes\_entr)

# calculate information gain

previousCustomer\_info = root\_entr-(no\_total\*no\_entr + yes\_total\*yes\_entr)/root\_total

print('Information gain(Previous Customer):',round(previousCustomer\_info,2))

print()

# root selection result

print('Second attribute(Suburban->): Income (',round(income\_info,2),')')

print('-'\*50, end="\n\n")

# 2-2. Calculate information gain (Urban -> second attribute)

# calculate root entropy (Urban)

root\_entr,root\_total = entr('District','Urban')

print('Entropy(root):',round(root\_entr,2))

print()

# calculate 'House Type' (Detached,Semi-detached,Terrace)

detached\_entr,detached\_total = entr('House Type','Detached','District','Urban')

semi\_detached\_entr,semi\_detached\_total = entr('House Type','Semi-detached','District','Urban')

terrace\_entr,terrace\_total = entr('House Type','Terrace','District','Urban')

print('Entropy(Detached/Semi-detached/Terrace):',detached\_entr,'/',semi\_detached\_entr,'/',terrace\_entr)

print('\*Detached entropy is 0. So Stop split in Detached')

# calculate information gain

houseType\_info = root\_entr-(detached\_total\*detached\_entr + semi\_detached\_total\*semi\_detached\_entr + terrace\_total\*terrace\_entr)/root\_total

print('Information gain(House Type):',round(houseType\_info,2))

print()

# calculate 'Income' (High,Low)

high\_entr,high\_total = entr('Income','High','District','Urban')

low\_entr,low\_total = entr('Income','Low','District','Urban')

print('Entropy(High/Low):',high\_entr,'/',low\_entr)

# calculate information gain

income\_info = root\_entr-(high\_total\*high\_entr + low\_total\*low\_entr)/root\_total

print('Information gain(Income):',round(income\_info,2))

print()

# calculate 'Previous Customer' (No,Yes)

no\_entr,no\_total = entr('Previous Customer','No','District','Urban')

yes\_entr,yes\_total = entr('Previous Customer','Yes','District','Urban')

print('Entropy(No/Yes):',no\_entr,'/',yes\_entr)

print('\*No entropy is 0. So Stop split in No')

print('\*Yes entropy is 0. So Stop split in Yes')

# calculate information gain

previousCustomer\_info = root\_entr-(no\_total\*no\_entr + yes\_total\*yes\_entr)/root\_total

print('Information gain(Previous Customer):',round(previousCustomer\_info,2))

print()

# root selection result

print('Second attribute(Urban->): Previous Customer (',round(previousCustomer\_info,2),')')

print('-'\*50, end="\n\n")

#Tree result

print('<Tree Result>')

print(' '\*30,'District')

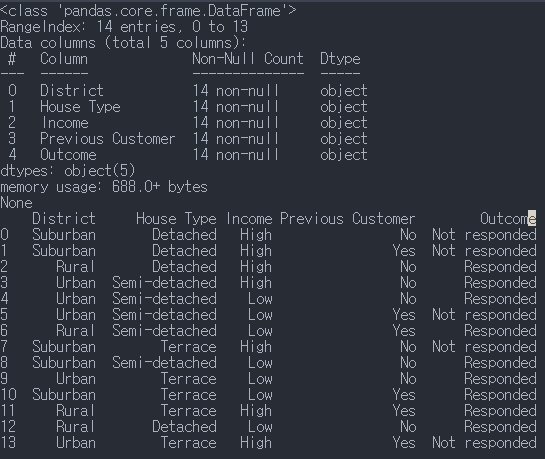
print(' '\*10,'(Suburban)\t\t(Rural)\t\t(Urban)')

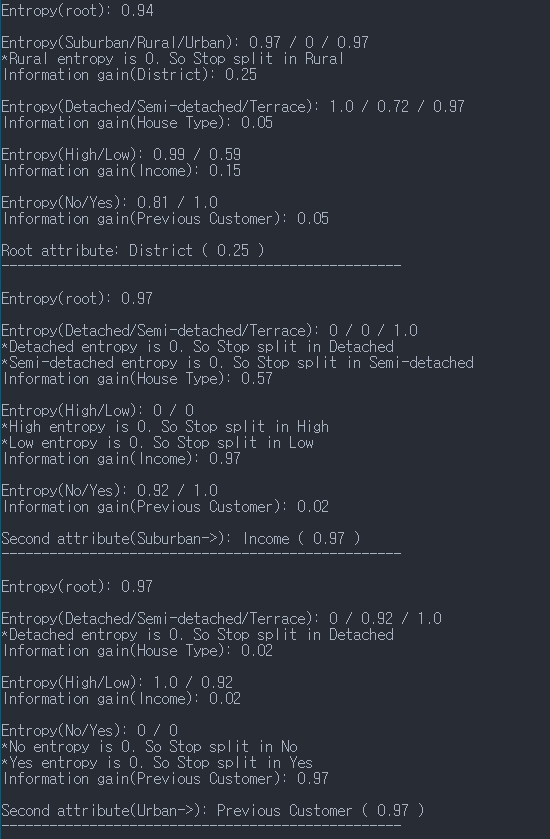
print(' '\*10,'Income\t\tResponded\tPrevious Customer')

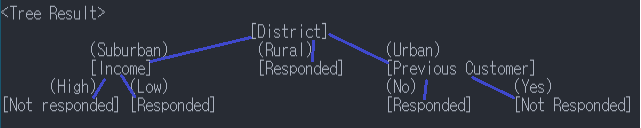
print(' '\*5,'(High)\t(Low)\t\t\t\t(No)\t\t(Yes)')

print('Not responded\tResponded\t\t\tResponded\tNot Responded')

**# Result screen**







**# Homework 2 - KNN**

**# Source code**

import numpy as np

import pandas as pd

# make KNN algorithm

# data: dataset(DataFrame) / target: target value {height,weight} / k: hyperparameter

def KNN(df,target,k):

# 0. Scaling (but we have only numpy, pandas library. So, we will pass this phase.)

# 1. Calculate distance

distance=[]

for h,w,s in zip(df['HEIGHT(cm)'],df['WEIGHT(kg)'],df['T SHIRT SIZE']):

cal\_result=np.sqrt(np.power(h-target['HEIGHT(cm)'],2)+np.power(w-target['WEIGHT(kg)'],2))

distance.append([cal\_result,s])

# 2. Search for the top 5 ranked records in the dataset

# sort distance list

distance.sort()

# select k ranked values

ranked=distance[:k]

# 3. return the most frequency value (M or L)

M\_count = 0

L\_count = 0

for item in ranked:

if item[1] == 'M':

M\_count+=1

elif item[1] == 'L':

L\_count+=1

if M\_count > L\_count:

return 'M'

else : return 'L'

# add dataset

data = {'HEIGHT(cm)':[158,158,158,160,160,163,163,160,163,165,165,165,168,168,168,170,170,170],

'WEIGHT(kg)':[58,59,63,59,60,60,61,64,64,61,62,65,62,63,66,63,64,68],

'T SHIRT SIZE':['M','M','M','M','M','M','M','L','L','L','L','L','L','L','L','L','L','L']}

df = pd.DataFrame(data,columns=['HEIGHT(cm)','WEIGHT(kg)','T SHIRT SIZE'])

# data exploration

print(df.info())

print(df,end="\n\n")

# test KNN algorithm using the dataset (k=3)

target = {'HEIGHT(cm)':161,'WEIGHT(kg)':61}

result = KNN(df,target,3)

# show result (k=3)

print('Result of T-shirt size with KNN(k=3):',result)

# test KNN algorithm using the dataset (k=5)

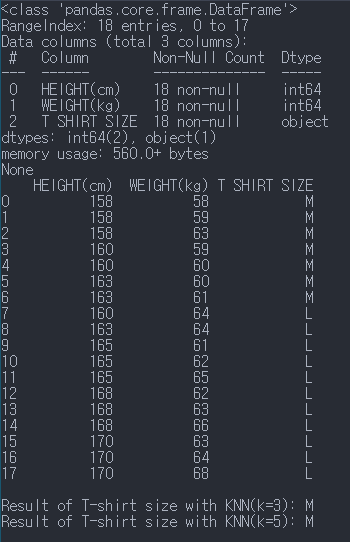
target = {'HEIGHT(cm)':161,'WEIGHT(kg)':61}

result = KNN(df,target,5)

# show result (k=5)

print('Result of T-shirt size with KNN(k=5):',result)

**# Result screen**



**# Team Discussion**

While we doing homework individually and together, we learn that there can be numerous code that have similar output. In this homework each of team member’s codes are all different. So we can learn different kind of coding about decision tree and KNN by each other’s code.

And we feel that programming some algorithm such as decision tree, KNN without library is pretty difficult but by this homework we can learn about that algorithm’s task more certainly.