1. Decision tree source code

import numpy as np

import pandas as pd

# create dataset

data = 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']})

print(data)

# descriptive features

features = data[['District', 'House Type', 'Income', 'Previous Customer']]

# target feature

target = data['Outcome']

# calculate entropy

def entropy(target\_col):

    elements, counts = np.unique(target\_col, 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

# calculate information gain by using entropy

def InfoGain(data, split\_attribute\_name, target\_name):

    total\_entropy = entropy(data[target\_name])

    print('Entropy(D) = ', round(total\_entropy, 2))

    vals, counts = np.unique(data[split\_attribute\_name], return\_counts=True)

    Weighted\_Entropy = np.sum([(counts[i]/np.sum(counts))\*entropy(data.where(data[split\_attribute\_name]==vals[i]).dropna()[target\_name]) for i in range(len(vals))])

    print('H(', split\_attribute\_name, ') = ', round(Weighted\_Entropy, 2))

    Information\_Gain = total\_entropy - Weighted\_Entropy

    return Information\_Gain

# find root node

District\_Information\_Gain = round(InfoGain(data, "District", "Outcome"), 2)

House\_Type\_Information\_Gain = round(InfoGain(data, "House Type", "Outcome"), 2)

Income\_Information\_Gain = round(InfoGain(data, "Income", "Outcome"), 2)

Previous\_Customer\_Information\_Gain = round(InfoGain(data, "Previous Customer", "Outcome"), 2)

print('---------------------------------------------------')

print("Information Gain(District) = ", District\_Information\_Gain, '\n')

print("Information Gain(House Type) = ", House\_Type\_Information\_Gain, '\n')

print("Information Gain(Income) = ", Income\_Information\_Gain, '\n')

print("Information Gain(Previous Customer) = ", Previous\_Customer\_Information\_Gain)

print('Root attribute : District')

# create root node's child node dataset

data\_suburban = data[data['District'] == 'Suburban']

data\_rural = data[data['District'] == 'Rural']

data\_urban = data[data['District'] == 'Urban']

# find child node for suburban

sub\_House\_Type\_Information\_Gain = round(InfoGain(data\_suburban, "House Type", "Outcome"), 2)

sub\_Income\_Information\_Gain = round(InfoGain(data\_suburban, "Income", "Outcome"), 2)

sub\_Previous\_Customer\_Information\_Gain = round(InfoGain(data\_suburban, "Previous Customer", "Outcome"), 2)

print("Information Gain(House Type) = ", sub\_House\_Type\_Information\_Gain, '\n')

print("Information Gain(Income) = ", sub\_Income\_Information\_Gain, '\n')

print("Information Gain(Previous Customer) = ", sub\_Previous\_Customer\_Information\_Gain, '\n')

print("District's left child(suburban) attribute : Income")

# find child node for urban

urban\_House\_Type\_Information\_Gain = round(InfoGain(data\_urban, "House Type", "Outcome"), 2)

urban\_Previous\_Customer\_Information\_Gain = round(InfoGain(data\_urban, "Previous Customer", "Outcome"), 2)

print("Information Gain(House Type) = ", urban\_House\_Type\_Information\_Gain, '\n')

print("Information Gain(Previous Customer) = ", urban\_Previous\_Customer\_Information\_Gain, '\n')

print("District's right child(urban) attribute : Previous Customer")

Output screen

텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명



텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

1. K nearest neighbor source code

import numpy as np

import pandas as pd

# create dataset

data = pd.DataFrame({'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']})

print(data)

# target customer data to predict t shirt size

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

output screen

# calculate & predict target data's t shirt size

def KNN(df, target\_data, k):

    distance\_arr = []

    distance = 0.0

    for height, weight, size in zip(df['HEIGHT(cm)'], df['WEIGHT(kg)'], df['T SHIRT SIZE']):

        distance = np.sqrt((height-target\_data['HEIGHT(cm)'])\*\*2 + (weight-target\_data['WEIGHT(kg)'])\*\*2)

        distance\_arr.append([height, weight, distance, size])

    distance\_arr.sort(key=lambda x: x[2])

    # find nearest data

    k\_rank = distance\_arr[:k]

    m = 0

    l = 0

    for size in k\_rank:

        if size[3] == 'M':

            m += 1

        elif size[3] == 'L':

            l += 1

    if m>l:

        return 'M'

    else:

        return 'L'

target\_size = KNN(data, target, 3)

print(target\_size)

텍스트, 모니터, 화면, 검은색이(가) 표시된 사진

자동 생성된 설명

