|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211015 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| Train set :  어떠한 데이터와 타깃을 주고 훈련하는 것.  Test set:  머신러닝 알고리즘의 성능을 평가하기 위해 훈련 데이터와 평가에 사용할 데이터가 각각 달라야 하기 때문에 평가에 쓰이는 데이터 세트  샘플 :  여러 데이터 중 하나의 데이터  이미 준비된 데이터 중에서 일부를 떼어 내어 활용하는 경우가 많은데, 평가에 사용하는 데이터를 Test set, 훈련에 사용되는 데이터를 Train set라고 부름.  예제) 도미와 빙어의 데이터로 제시된 샘플이 도미인지 빙어인지 맞추기  import numpy as np # 매트릭스 연산  import pandas as pd  import matplotlib.pyplot as plt # plot  < 도미 데이터 입력 >  bream\_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0, 31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0, 35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0]  bream\_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0, 500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0, 700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0]  print(type(bream\_length))  print(np.shape(bream\_length), '<- 35\*1 matrix') #np.shape 디멘션 확인할때 사용하는거  plt.scatter(bream\_length, bream\_weight, s = 100)  # 인과성, 길이가 길어서 무게가 더 나간다. 무게가 많이 나가서 길이가 길다  plt.title("Bream, Length vs Weight")  plt.xlabel("Length (cm)")  plt.ylabel("Weight (g)")  plt.show()  < 빙어 데이터 입력 >  smelt\_length = [9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2, 12.4, 13.0, 14.3, 15.0]  smelt\_weight = [6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]  print(np.shape(smelt\_length))  plt.scatter(bream\_length, bream\_weight, label = "Bream")  plt.scatter(smelt\_length, smelt\_weight, label = "Smelt")  plt.title("Bream, Smelt")  plt.xlabel("Length (cm)")  plt.ylabel("Weight (g)")  plt.legend()  plt.show()  < knn을 이용한 분류>  # bream weight 밑에 length 붙여넣기 ( n \* 1 모양으로),  # smelt도 붙여줌, 그리고 그 옆에 target 붙이기 (1,0)으로 도미 빙어 구분  length = bream\_length + smelt\_length  weight = bream\_weight + smelt\_weight  print(np.shape(length))  print(length)  fish\_data = [[l, w]for l, w in zip(length, weight)]  print(type(fish\_data))  print(np.shape(fish\_data))  fish\_data[:5] # 사이킷런이 원하는 2차원 리스트  fish\_target = [1]\*35 + [0]\*14  print(fish\_target)  # sklearn knn module import  from sklearn.neighbors import KNeighborsClassifier  # 객체 생성  kn = KNeighborsClassifier(n\_neighbors = 5) # default K = 5  # knn 모델 fitting  kn.fit(fish\_data, fish\_target) # kn.fit : 오메가를 구해준다  # knn model metrics  kn.score(fish\_data, fish\_target)  # prediction : 예측  kn.predict([[30, 600]]) <- 길이 30에 무게 600인 물고기는 뭘까? = 1(도미 나옴)  plt.scatter(bream\_length, bream\_weight, s = 100 , label = "bream = 1")  plt.scatter(smelt\_length, smelt\_weight, s = 100, label = "smelt = 0")  plt.scatter(30, 600, s = 100, label = "New")  plt.title("Bream vs Smelt")  plt.xlabel("Length")  plt.ylabel("Weight")  plt.legend()  plt.show()  # k = 9 이면 모든 데이터가 도미가 됨  print('kn.\_fit\_X = \n', kn.\_fit\_X)  print("kn.\_y = \n", kn.\_y)  <물고기 길이와 무게로 머신러닝 해보기>  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish\_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,  31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,  35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0, 9.8,  10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2, 12.4, 13.0, 14.3, 15.0]  fish\_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,  500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,  700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0, 6.7,  7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]  #fish\_data = [[l,w] for l, w in zip(fish\_length, fish\_weight)]  fish\_data = np.column\_stack((fish\_length, fish\_weight)) #위랑 같이 합치는 메소드  fish\_target = [1]\*35 + [0]\*14 #0이랑 1 바뀌어도 상관없음 해석만 할 수 있으면 됨  print(np.shape(fish\_data))  fish\_data[:5]  # 35 - 트레이닝, 나머지 - 테스트  train\_input = fish\_data[:35]  test\_input = fish\_data[35:]  train\_target = fish\_target[:35]  test\_target = fish\_target[35:]  < 샘플링 편향 >  from sklearn.neighbors import KNeighborsClassifier  kn = KNeighborsClassifier()  kn.fit(train\_input, train\_target)  kn.score(test\_input, test\_target) # 0 = 안맞네  print(type(fish\_data))  print(type(fish\_target))  fish\_target = np.array(fish\_target) # list를 array로  print(fish\_data)  print(fish\_target)  np.arange(1, 11, 2)  index = np.arange(49)  print(index)  np.random.shuffle(index)  print(index)  train\_input = fish\_data[index[:35]]  train\_target = fish\_target[index[:35]]  test\_input = fish\_data[index[35:]]  test\_target = fish\_target[index[35:]]  plt.scatter(train\_input[:,0], train\_input[:,1])  plt.scatter(test\_input[:,0], test\_input[:,1])  plt.show()  kn.fit(train\_input, train\_target)  kn.score(test\_input, test\_target) # 1.0 = 잘맞았다  print(kn.predict(test\_input))  print(test\_target)  < 데이터 전처리, knn >  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish\_length = [25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7, 31.0, 31.0,  31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5, 34.0, 34.0, 34.5, 35.0,  35.0, 35.0, 35.0, 36.0, 36.0, 37.0, 38.5, 38.5, 39.5, 41.0, 41.0, 9.8,  10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2, 12.4, 13.0, 14.3, 15.0]  fish\_weight = [242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0, 500.0, 475.0, 500.0,  500.0, 340.0, 600.0, 600.0, 700.0, 700.0, 610.0, 650.0, 575.0, 685.0, 620.0, 680.0,  700.0, 725.0, 720.0, 714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0, 6.7,  7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2, 13.4, 12.2, 19.7, 19.9]  print(np.shape(fish\_length))  print(np.shape(fish\_weight))  fish\_data = np.column\_stack((fish\_length, fish\_weight))  fish\_target = [1]\*35 + [0]\*14  np.ones(35) # 1로 35개 채운 리스트  np.zeros(14) # 0으로 14개 채운 리스트  fish\_target = np.concatenate((np.ones(35), np.zeros(14)))  # print(np.shape(fish\_data))  # print(fish\_target)  fish\_data[:5]  fish\_target[:5]  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target = train\_test\_split(  fish\_data, fish\_target, stratify = fish\_target, random\_state = 42)  # knn  from sklearn.neighbors import KNeighborsClassifier  kn = KNeighborsClassifier()  kn.fit(train\_input, train\_target)  kn.score(test\_input, test\_target)  #kn.score(train\_input, train\_target)  kn.predict([[25, 150]]) # 이 길이와 무게로 봤을때 빙어냐 도미냐. 0 = 빙어래  distances, inds = kn.kneighbors([[25, 150]])  print(distances)  print(inds)  plt.scatter(train\_input[:,0],train\_input[:,1])  plt.scatter(25, 150, marker = '^')  plt.scatter(train\_input[inds,0],train\_input[inds,1], marker = 'D', color = 'r')  # plt.xlim(0,1000)  plt.xlabel("Length")  plt.ylabel("Weight")  plt.show()  < Feature scaling >  mean = np.mean(train\_input, axis = 0)  # mean = np.mean(train\_input)  # -> 무게에서만 평균낸게아니라 길이 무게 전체의 평균을 낸거임. axis= 해줘야함  std = np.std(train\_input, axis = 0)  print('mean =',mean.round(2))  print('std =', std.round(2))  print(train\_input[:5])  # 표준화(일일히 계산)  # (X1-X바)/S = Z1  # (29.7-27.3)/9.98, (500-454.1)/323.3  # (12.2-27.3)/9.98, (12.2-454.1)/323.3  # 표준화  train\_scaled = (train\_input - mean) / std  train\_scaled[:5]  new = ([25, 150] - mean) / std  print(new) # 25, 150을 표준화 한 데이터  plt.scatter(train\_scaled[:,0], train\_scaled[:,1])  plt.scatter(new[0],new[1],marker='^',color='r')  kn = KNeighborsClassifier()  kn.fit(train\_scaled, train\_target) # target은 scale하는거 아님  # mean\_t = np.mean(test\_input, axis = 0)  # std\_t = np.std(test\_input, axis = 0)  test\_scaled = (test\_input - mean) / std  kn.score(test\_scaled,test\_target)  kn.predict([new]) # 아깐 빙어였던게 이젠 도미가 됨 만세  dist, indx = kn.kneighbors([new]) # new = 25, 150  print(indx) # 가까운 5개가 누구냐? 21번째샘플 14번째샘플 34번째샘플...  plt.scatter(train\_scaled[:,0],train\_scaled[:,1])  plt.scatter(new[0],new[1], marker = '^', color = 'g')  plt.scatter(train\_scaled[indx, 0],train\_scaled[indx, 1],marker = 'D', color = 'r')  plt.show() | |