|  |  |
| --- | --- |
| 교육 제목 | Market & Machine Learning |
| 교육 일시 | 2021. 10. 15 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import numpy as np # 매트릭스 연산  import pandas as pd # 데이터프레임  import matplotlib.pyplot as plt # 플롯  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(bream\_length)  print(type(bream\_length))  print(np.shape(bream\_length))  plt.scatter(bream\_length, bream\_weight)  plt.show()  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)  plt.scatter(smelt\_length, smelt\_weight)  plt.title("Bream, Smelt")  plt.xlabel("Length (cm)")  plt.ylabel("Weight (g)")  plt.legend()  plt.show()  # knn을 이용한 분류  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]  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)  # knn 모델 fitting  kn.fit(fish\_data, fish\_target)  # knn model metrics  kn.score(fish\_data, fish\_target)  # prediction  kn.predict([[30, 600]])  plt.scatter(bream\_length, bream\_weight, s= 100, label="bream=1")  plt.scatter(smelt\_length, smelt\_weight, s= 100, label="bream=0")  plt.scatter(30, 600, s=100, label="New")  plt.title("Bream vs Smelt")  plt.xlabel("Length")  plt.ylabel("Weight")  plt.legend()  plt.show()  kn.\_fit\_X  kn.\_y  print(np.shape(length))  print(np.shape(weight))  # fish\_data  fish\_data=np.column\_stack((length, weight))  print(np.shape(fish\_data))  fish\_data[:5]  train\_input=fish\_data[:35]  test\_input=fish\_data[35:]  train\_target=fish\_target[:35]  test\_target=fish\_target[35:]  # 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=KNeighborsClassifier()  kn.fit(train\_input, train\_target)  kn.score(test\_input, test\_target)  print(kn.predict(test\_input))  print(test\_target)  fish\_data=np.column\_stack((length, weight))  #fish\_target=[1]\*35+[0]\*14  #np.ones(35)  #np.zeros(14)  fish\_target=np.concatenate((np.ones(35), np.zeros(14)))  # print(np.shape(fish\_data))  # print(fish\_target)  fish\_data[:5]  fish\_target[:5] |
| 오후 | from sklearn.model\_selection import train\_test\_split  #train\_test\_split(  # \*arrays,  # test\_size=None,  # train\_size=None,  # random\_state=None,  # shuffle=True,  # stratify=None  #)  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_data, fish\_target, stratify=fish\_target, random\_state=42)  print(np.shape(fish\_data))  print(np.shape(train\_input))  print(np.shape(test\_input))  print(test\_target)  # 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]])  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')  plt.xlabel("Length")  plt.ylabel("Weight")  plt.show()  train\_input[inds]  # Feature scaling  mean=np.mean(train\_input, axis=0)  std=np.std(train\_input, axis=0)  print(mean)  print(std)  print(train\_input[:5])  # 표준화  train\_scaled=(train\_input-mean)/std  train\_scaled[:5]  new=([25,150]-mean)/std  print(new)  plt.scatter(train\_scaled[:, 0], train\_scaled[:, 1])  plt.scatter(-0.23, -0.94, marker="^")  plt.show()  kn=KNeighborsClassifier()  kn.fit(train\_scaled, train\_target)  # mean\_t=np.mean(test\_input, axis=0)  # std\_t=np.std(test\_input, axis=0)  # test\_scaled=(test\_input-mean\_t)/std\_t  test\_scaled=(test\_input-mean)/std  kn.score(test\_scaled, test\_target)  kn.predict([new])  dlst, indx=kn.kneighbors([new])  print(indx)  plt.scatter(train\_scaled[:, 0], train\_scaled[:, 1])  plt.scatter(new[0], new[1], marker="^")  plt.scatter(train\_scaled[indx, 0], train\_scaled[indx, 1], marker="D")  plt.show() |