|  |  |
| --- | --- |
| 교육 제목 | Machine Learning |
| 교육 일시 | 2021. 10. 18 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  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]  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]  #plt.scatter(bream\_length, bream\_weight, label="")  plt.scatter(bream\_length, bream\_weight, label="bream=1")  plt.scatter(smelt\_length, smelt\_weight, label="bream=0")  plt.title("Bream vs Smelt")  plt.xlabel("Length")  plt.ylabel("Weight")  plt.legend()  plt.show()  # Data format  length=bream\_length+smelt\_length  weight=bream\_weight+smelt\_weight  print(np.shape(length))  print(np.shape(bream\_length))  print(np.shape(smelt\_length))  fish\_data=[[l,w] for l, w in zip(length, weight)]  print(type(fish\_data))  print(np.shape(fish\_data))  fish\_target=[1]\*35+[0]\*14  print(fish\_target)  fish\_data[: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)  print(np.shape(fish\_data))  print(np.shape(train\_input), np.shape(test\_input))  print(train\_target)  print(test\_target)  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_input)  train\_scaled=ss.transform(train\_input)  test\_scaled=ss.transform(test\_input)  # knn  from sklearn.neighbors import KNeighborsClassifier  # 객체 생성  kn=KNeighborsClassifier(n\_neighbors=5)  kn.fit(train\_scaled, train\_target)  perch\_length = np.array(  [8.4, 13.7, 15.0, 16.2, 17.4, 18.0, 18.7, 19.0, 19.6, 20.0,  21.0, 21.0, 21.0, 21.3, 22.0, 22.0, 22.0, 22.0, 22.0, 22.5,  22.5, 22.7, 23.0, 23.5, 24.0, 24.0, 24.6, 25.0, 25.6, 26.5,  27.3, 27.5, 27.5, 27.5, 28.0, 28.7, 30.0, 32.8, 34.5, 35.0,  36.5, 36.0, 37.0, 37.0, 39.0, 39.0, 39.0, 40.0, 40.0, 40.0,  40.0, 42.0, 43.0, 43.0, 43.5, 44.0]  )  perch\_weight = np.array(  [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0, 85.0, 85.0,  110.0, 115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0, 110.0,  130.0, 150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0,  197.0, 218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0,  514.0, 556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0,  820.0, 850.0, 900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0,  1000.0, 1000.0]  )  plt.scatter(perch\_length, perch\_weight)  plt.xlabel('length')  plt.ylabel('weight')  plt.show()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  perch\_length, perch\_weight#, random\_state=42)  )  print(np.shape(train\_input))  print(np.shape(test\_input))  train\_input=train\_input.reshape(-1, 1)  test\_input=test\_input.reshape(-1, 1)  print(np.shape(train\_input))  print(np.shape(test\_input))  from sklearn.neighbors import KNeighborsRegressor  knr=KNeighborsRegressor()  knr.fit(train\_input, train\_target)  print(knr.score(train\_input, train\_target))  print(knr.score(test\_input, test\_target))  r2\_train=[]  r2\_test=[]  neighbors\_n=[]  for n in range(1, 21):  knr.n\_neighbors=n  knr.fit(train\_input, train\_target)  r2\_train.append(knr.score(train\_input, train\_target))  r2\_test.append(knr.score(test\_input, test\_target))  neighbors\_n.append(n)    # print(r2\_train)  # print(r2\_test)  # print(neighbors\_n)  plt.plot(neighbors\_n, r2\_train, label='train')  plt.plot(neighbors\_n, r2\_test, c="r", label='test')  plt.xlabel('neighbors number')  plt.ylabel('R^2')  plt.legend()  plt.show()  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.neighbors import KNeighborsClassifier  # 객체 생성  kn=KNeighborsClassifier(n\_neighbors=5)  perch\_length = np.array(  [8.4, 13.7, 15.0, 16.2, 17.4, 18.0, 18.7, 19.0, 19.6, 20.0,  21.0, 21.0, 21.0, 21.3, 22.0, 22.0, 22.0, 22.0, 22.0, 22.5,  22.5, 22.7, 23.0, 23.5, 24.0, 24.0, 24.6, 25.0, 25.6, 26.5,  27.3, 27.5, 27.5, 27.5, 28.0, 28.7, 30.0, 32.8, 34.5, 35.0,  36.5, 36.0, 37.0, 37.0, 39.0, 39.0, 39.0, 40.0, 40.0, 40.0,  40.0, 42.0, 43.0, 43.0, 43.5, 44.0]  )  perch\_weight = np.array(  [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0, 85.0, 85.0,  110.0, 115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0, 110.0,  130.0, 150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0,  197.0, 218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0,  514.0, 556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0,  820.0, 850.0, 900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0,  1000.0, 1000.0]  )  plt.scatter(perch\_length, perch\_weight)  plt.xlabel('length')  plt.ylabel('weight')  plt.show()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  perch\_length, perch\_weight)  train\_input=train\_input.reshape(-1, 1)  test\_input=test\_input.reshape(-1, 1)  from sklearn.neighbors import KNeighborsRegressor  knr=KNeighborsRegressor(n\_neighbors=3)  knr.fit(train\_input, train\_target)  print(knr.score(train\_input, train\_target))  print(knr.score(test\_input, test\_target))  knr.predict([[50]])  dist, indx=knr.kneighbors([[50]])  print(indx)  plt.scatter(train\_input, train\_target)  plt.scatter(50, 1033, marker="D")  plt.scatter(train\_input[indx], train\_target[indx], marker="^")  plt.show()  # 선형회귀  from sklearn.linear\_model import LinearRegression  lr=LinearRegression()  lr.fit(train\_input, train\_target)  print(lr.score(train\_input, train\_target))  print(lr.score(test\_input, test\_target))  lr.predict([[50]])  print(lr.coef\_, lr.intercept\_)  x\_new=np.arange(12,60)  y\_new=x\_new\*lr.coef\_+lr.intercept\_  plt.scatter(train\_input, train\_target)  plt.scatter(50, 1252, marker="^")  plt.plot(x\_new, y\_new)  plt.show()  train\_input[:10]  # print(train\_input.shape)  train\_poly=np.column\_stack((train\_input\*\*2, train\_input))  test\_poly=np.column\_stack((test\_input\*\*2, test\_input))  train\_poly[:10]  lrp=LinearRegression()  lrp.fit(train\_poly, train\_target)  print(lrp.coef\_, lrp.intercept\_)  print(lrp.score(train\_poly, train\_target))  print(lrp.score(train\_poly, train\_target))  print(lrp.score(test\_poly, test\_target))  x\_new=np.arange(10, 50)  y\_new=112.14-20.93\*x\_new+0.99\*x\_new\*\*2  plt.scatter(train\_input, train\_target)  plt.show() |
| 오후 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  df=pd.read\_csv('https://bit.ly/perch\_csv')  print(type(df))  print(df.shape)  # print(df.shape)  # df.head()  perch\_full=df.to\_numpy()  print(type(perch\_full))  print(np.shape(perch\_full))  perch\_weight = np.array(  [5.9, 32.0, 40.0, 51.5, 70.0, 100.0, 78.0, 80.0, 85.0, 85.0,  110.0, 115.0, 125.0, 130.0, 120.0, 120.0, 130.0, 135.0, 110.0,  130.0, 150.0, 145.0, 150.0, 170.0, 225.0, 145.0, 188.0, 180.0,  197.0, 218.0, 300.0, 260.0, 265.0, 250.0, 250.0, 300.0, 320.0,  514.0, 556.0, 840.0, 685.0, 700.0, 700.0, 690.0, 900.0, 650.0,  820.0, 850.0, 900.0, 1015.0, 820.0, 1100.0, 1000.0, 1100.0,  1000.0, 1000.0]  )  print(np.shape(perch\_weight))  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  perch\_full, perch\_weight, random\_state=42)  # polynomial transform  from sklearn.preprocessing import PolynomialFeatures  poly=PolynomialFeatures(degree=3)  poly.fit([[2,3]])  poly.transform([[2,3]])  poly=PolynomialFeatures(degree=5, include\_bias=False)  poly.fit(train\_input)  train\_poly=poly.transform(train\_input)  test\_poly=poly.transform(test\_input)  train\_input[:5]  train\_poly  # print(train\_poly[:10])  poly.get\_feature\_names()  from sklearn.linear\_model import LinearRegression  lr=LinearRegression()  lr.fit(train\_poly, train\_target)  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target))  lr=LinearRegression()  lr.fit(train\_poly, train\_target)  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target))  # Rigde regression  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_poly)  train\_scaled=ss.transform(train\_poly)  test\_scaled=ss.transform(test\_poly)  from sklearn.linear\_model import Ridge  ridge=Ridge()  ridge.fit(train\_scaled, train\_target)  print(ridge.score(train\_scaled, train\_target))  print(ridge.score(test\_scaled, test\_target))  # Ridge lambda plot  train\_score=[]  test\_score=[]  alpha\_list=[0.001, 0.01, 0.1, 1, 10, 100]  for alpha in alpha\_list:  ridge=Ridge(alpha=alpha)  ridge.fit(train\_scaled, train\_target)    train\_score.append(ridge.score(train\_scaled, train\_target))  test\_score.append(ridge.score(test\_scaled, test\_target))  train\_score  test\_score  plt.plot(np.log10(alpha\_list), train\_score, label="Train")  plt.plot(np.log10(alpha\_list), test\_score, label="Test")  plt.xlabel("log10(alpha)")  plt.ylabel("R^2")  plt.legend()  plt.show()  from sklearn.linear\_model import Lasso  lasso=Lasso()  lasso.fit(train\_scaled, train\_target)  print(lasso.score(train\_scaled, train\_target))  print(lasso.score(test\_scaled, test\_target))  train\_score=[]  test\_score=[]  alpha\_list=[0.001, 0.01, 0.1, 1, 10, 100]  for alpha in alpha\_list:  # 라쏘 모델을 만듭니다.  lasso=Lasso(alpha=alpha, max\_iter=10000)  # 라쏘 모델을 훈련합니다.  lasso.fit(train\_scaled, train\_target)  # 훈련 점수와 테스트 점수를 저장합니다.  train\_score.append(lasso.score(train\_scaled, train\_target))  test\_score.append(lasso.score(test\_scaled, test\_target))  plt.plot(np.log10(alpha\_list), train\_score, label="Training data")  plt.plot(np.log10(alpha\_list), test\_score, label="Test data")  plt.xlabel("log10(alpha)")  plt.ylabel("R^2")  plt.legend()  plt.show()  lasso=Lasso(alpha=10)  lasso.fit(train\_scaled, train\_target)  print(lasso.score(train\_scaled, train\_target))  print(lasso.score(test\_scaled, test\_target))  np.sum(lasso.coef\_ !=0) |