|  |  |
| --- | --- |
| 교육 제목 | Machine Learning |
| 교육 일시 | 2021. 10. 19 |
| 교육 장소 | 영우글로벌러닝 |
| **교육 내용** | |
| 오전 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  from sklearn.neighbors import KNeighborsClassifier  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]  )  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.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))  print(lr.coef\_, lr.intercept\_)  x\_new=np.arange(15, 51)  y\_new=x\_new\*37.48-651.99  plt.scatter(train\_input, train\_target)  plt.plot(x\_new, y\_new, color="r")  plt.show()  train\_poly=np.column\_stack((train\_input, train\_input\*\*2))  test\_poly=np.column\_stack((test\_input, test\_input\*\*2))  lr=LinearRegression()  lr.fit(train\_poly, train\_target)  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target))  print(lr.coef\_, lr.intercept\_)  x\_new=np.arange(15, 51)  y\_new=-19\*x\_new+0.97\*x\_new\*\*2+92.7  plt.scatter(train\_input, train\_target)  plt.plot(x\_new, y\_new, color="r")  plt.show()  ------------------------------------------------------------------------  df=pd.read\_csv('https://bit.ly/perch\_csv')  print(type(df))  # df.head()  perch\_full=df.to\_numpy()  print(type(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]  )  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  perch\_full, perch\_weight)  from sklearn.preprocessing import PolynomialFeatures  # poly=PolynomialFeatures()  poly=PolynomialFeatures(degree=5, include\_bias=False)  poly.fit(train\_input)  train\_poly=poly.transform(train\_input)  test\_poly=poly.transform(test\_input)  poly.get\_feature\_names()  lr=LinearRegression()  lr.fit(train\_poly, train\_target)  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target))  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(alpha=1)  ridge.fit(train\_scaled, train\_target)  print(ridge.score(train\_scaled, train\_target))  print(ridge.score(test\_scaled, test\_target))  train\_score=[]  test\_score=[]  alpha\_list=[0.001, 0.01, 0.1, 1, 10, 100]  # alpha\_list=np.arange(1, 20)  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))  plt.plot(np.log10(alpha\_list), train\_score, label='train')  plt.plot(np.log10(alpha\_list), test\_score, label='test')  plt.legend()  plt.show() |
| 오후 | import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish=pd.read\_csv('https://bit.ly/fish\_csv')  fish.head()  # print(fish.shape)  fish['Species'].value\_counts()  fish\_input=fish[["Weight", "Length", "Diagonal", "Height", "Width"]].to\_numpy()  # fish\_input.head()  # fish\_input[:5]  fish\_target=fish["Species"].to\_numpy()  # fish\_target[:10]  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  print(np.shape(train\_input))  print(np.shape(test\_input))  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_input)  train\_scaled=ss.transform(train\_input)  test\_scaled=ss.transform(test\_input)  z=np.arange(-5, 5, 0.1)  prob\_y=1/(1+np.exp(-z))  plt.plot(z, prob\_y)  plt.axhline(1, linestyle="--", color="r")  plt.axhline(0, linestyle="--", color="r")  plt.axhline(0.5, linestyle="--", color="b")  plt.show()  bream\_smelt\_index=(train\_target=="Bream") | (train\_target=="Smelt")  bream\_smelt\_index  # train\_target  train\_bream\_smelt=train\_scaled[bream\_smelt\_index]  target\_bream\_smelt=train\_target[bream\_smelt\_index]  print(np.shape(train\_scaled))  print(np.shape(train\_bream\_smelt))  from sklearn.linear\_model import LogisticRegression  lr=LogisticRegression()  lr.fit(train\_bream\_smelt, target\_bream\_smelt)  print(lr.score(train\_bream\_smelt, target\_bream\_smelt))  print(train\_bream\_smelt[:5])  print(lr.predict\_proba(train\_bream\_smelt[:5]))  print(lr.predict(train\_bream\_smelt[:5]))  print(lr.coef\_, lr.intercept\_)  lr=LogisticRegression(max\_iter=1000)  lr.fit(train\_scaled, train\_target)  print(lr.score(train\_scaled, train\_target))  print(lr.score(test\_scaled, test\_target))  print(lr.predict(test\_scaled[:10]))  print(test\_target[:10])  lr.predict\_proba(test\_scaled[:5]).round(3)  print(lr.classes\_)  print(lr.coef\_)  print(lr.intercept\_)  ------------------------------------------------------------------------------  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  fish=pd.read\_csv('https://bit.ly/fish\_csv')  fish['Species'].value\_counts()  fish.head()  fish\_input=fish[["Weight", "Length", "Diagonal", "Height", "Width"]].to\_numpy()  fish\_target=fish["Species"].to\_numpy()  from sklearn.model\_selection import train\_test\_split  train\_input, test\_input, train\_target, test\_target=train\_test\_split(  fish\_input, fish\_target, stratify=fish\_target)  from sklearn.preprocessing import StandardScaler  ss=StandardScaler()  ss.fit(train\_input)  train\_scaled=ss.transform(train\_input)  test\_scaled=ss.transform(test\_input)  from sklearn.linear\_model import SGDClassifier  sc=SGDClassifier(loss="log", max\_iter=100)  sc.fit(train\_scaled, train\_target)  print(sc.score(train\_scaled, train\_target))  print(sc.score(test\_scaled, test\_target))  # Partial fit  sc.partial\_fit(train\_scaled, train\_target)  print(sc.score(train\_scaled, train\_target))  print(sc.score(test\_scaled, test\_target))  classes=np.unique(train\_target)  classes  sc=SGDClassifier(loss="log")  train\_score=[]  test\_score=[]  for i in range(0, 300):  sc.partial\_fit(train\_scaled, train\_target, classes=classes)  train\_score.append(sc.score(train\_scaled, train\_target))  test\_score.append(sc.score(test\_scaled, test\_target))  train\_score  plt.plot(train\_score, label="Train")  plt.plot(test\_score, label="Test")  plt.ylim(0.5, 1.0)  plt.legend()  plt.show() |