|  |  |
| --- | --- |
| 교육제목 | 데이터 기반 인공지능 시스템 엔지니어 양성 과정 |
| 교육일시 | 211018 |
| 교육장소 | 자택(디스코드 사용 온라인 학습) |
| **교육내용** | |
| 1. K-최근접 이웃 회귀   예측하려는 샘플에 가장 가까운 샘플 k개를 선택, 그다음 이 샘플들의 클래스를 확인하여 다수 클래스를 새로운 샘플의 클래스로 예측한다.  Ex) k=3(샘플이 3개)이라 가정하면 최근접이웃 중 가장 많은 클래스의 ‘평균’을 따라감.  최근접이웃 예제)  import numpy as np  import pandas as pd  import matplotlib.pyplot as plt  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)  #데이터 나누기  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]]) # length가 50이면 weight가 1033.333일것이다  dist, indx = knr.kneighbors([[100]])  print(indx)  plt.scatter(train\_input, train\_target)  plt.scatter(100, 1033, marker = "D" , color = 'r')  plt.scatter(train\_input[indx], train\_target[indx], marker = "^", color = 'y', s=100)  plt.show()  < 선형회귀 >  from sklearn.linear\_model import LinearRegression  lr = LinearRegression()  lr.fit(train\_input, train\_target) # x축 : 트레이닝인풋  # ^length ^weight  print(lr.score(train\_input, train\_target))  print(lr.score(test\_input, test\_target))  # R^2 = SSR/SSt = 1- SSE(에러의합)/SST  # 에러의 합이 적을수록 R^2값이 커진다  lr.predict([[50]])  # y햇(^)은 세타0+세타1x = -621.5 + 36.3x(여기에서 x에 50넣음) = 1196.7  print(lr.coef\_, lr.intercept\_) # 기울기 36, 절편 -621  x\_new = np.arange(12, 60)  y\_new = x\_new\*lr.coef\_ + lr.intercept\_  plt.scatter(train\_input, train\_target)  plt.scatter(50, 1196, marker='^', color = 'black')  plt.plot(x\_new, y\_new, color = 'r') # 회귀직선  plt.show()  # underfitting, 데이터가 휘니까 x제곱이 들어간 다중회귀모양으로 다시 그려야함  train\_input[:10] # 제곱해줘야됨  #print(train\_input.shape)  train\_poly = np.column\_stack((train\_input\*\*2, train\_input)) #옆으로 붙이기  test\_poly = np.column\_stack((test\_input\*\*2, test\_input))  # train\_poly[:10]  test\_poly[:10]  lrp = LinearRegression()  # parameter구하기 : 세타0,1,2  lrp.fit(train\_poly, train\_target)  print(lrp.coef\_, lrp.intercept\_)  # 세타2(제곱) 세타1  # 결과 : [세타2, 세타1], 세타0  # y= 121.83-21x+1.01x^2  print(lrp.score(train\_poly, train\_target))  print(lrp.score(test\_poly, test\_target))  # 구간별 직선, 15에서 49까지 정수 배열을 만들기  x\_new = np.arange(15, 50)  y\_new = 121.83 - 21.6\*x\_new + 1.01\*x\_new\*\*2  plt.scatter(train\_input, train\_target)  # 15에서 49까지 2차 방정식 그래프  plt.plot(x\_new, y\_new)  plt.xlabel('length')  plt.ylabel('weight')  plt.show()   1. 특성공학과 규제   훈련 세트보다 테스트 세트의 점수가 높은 문제를 해결하기 위해 제곱보다 더 고차항을 넣어야 함.(x^2까지 있는 다항식을 x^3이상으로).  예)  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  # 중회귀분석] y= 세타0 + 세타1\*x1 + 세타2\*x2 + 세타3\*x3  # 이차항으로 만들기 + 세타4\*x1^2 + 세타5\*x2^2 + 세타6\*x3^2...  # 복잡하니 패키지가 해주도록 만든다.  df = pd.read\_csv("https://bit.ly/perch\_csv\_data")  print(type(df)) # 56x3 데이터프레임. 넘파이타입으로 바꿔야함  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)) # 56x1  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, 항을 늘려주는녀석 가져옴  from sklearn.preprocessing import PolynomialFeatures  poly = PolynomialFeatures(degree = 2) # degree 3차항  poly.fit([[2,3]]) # 2 = x1, 3 = x2  poly.transform([[2,3]]) # x,x1,x2,x1x2,x1^2,x2^2 세제곱까지 다 나옴  poly = PolynomialFeatures(degree = 5, include\_bias=False)  # 자동으로 들어가는 절편 1이 빠짐    poly.fit(train\_input)  train\_poly = poly.transform(train\_input)  test\_poly = poly.transform(test\_input)  print(train\_poly[:10])  print('\n',poly.get\_feature\_names()) # 요 순서대로 나오는거랍니다  train\_input[:10]  #중회귀분석  from sklearn.linear\_model import LinearRegression  lr = LinearRegression()  lr.fit(train\_poly, train\_target) # target은 움직이는게 아님  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target))  # 2차항을 5차항으로 바꾸니까 마이너스..오버피팅이었음   1. Regularization(규제) 목적   과적합 방지  모델 파라미터 수를 조정(단순한 모델)  <주로 사용하는 penalty parameters>   * Ridge(L2 Norm) * Lasso(L1 Norm, Least absolute shrinkage and selection operator) * Elastic net(L1 + L2 Norm)   페널티 텀을 넣으면 함수 결과값이 작아짐  예)  poly = PolynomialFeatures(degree = 5, include\_bias=False)  poly.fit(train\_input)  train\_poly = poly.transform(train\_input)  test\_poly = poly.transform(test\_input)  print(np.shape(train\_input))  print(np.shape(train\_poly))  train\_input[:5]  train\_poly  lr = LinearRegression()  lr.fit(train\_poly, train\_target)  print(lr.score(train\_poly, train\_target))  print(lr.score(test\_poly, test\_target)) # 5차항으로 만드니까 마이너스가 나옴. 과적합  # Ridge 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))  # 5차항인데도 릿지페널티 거니까 잘들어맞음  # 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) # (ridge함수안에있는 alpha = 위 반복문 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(alpha\_list, train\_score, label = 'Train')  plt.plot(alpha\_list, test\_score, label = 'Test')  plt.legend()  plt.show()  #단위가 0.001부터 100까지라서 그래프가 묘하게 나옴  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') # 알스퀘어값  plt.legend()  plt.show() # 모양 이쁘게나옴, -1이 베스트지점이다. log10X = -1 => X=0.1  <Lasso>  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:  lasso = Lasso(alpha=alpha, max\_iter=10000) # 라쏘 만들기  lasso.fit(train\_scaled, train\_target)  # 훈련 점수와 테스트 점수 저장  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)  plt.plot(np.log10(alpha\_list), test\_score)  plt.xlabel('alpha')  plt.ylabel('R^2')  plt.show()  # -1 또는 1이 후보임. 다시한번 알아볼필요가 있음.  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) # 55개중에 40개가 사라짐 | |