|  |  |
| --- | --- |
| 교육 제목 | **Classification KNN & Regression KNN** |
| 교육 일시 | 21.10.15 |
| 교육 장소 | 오프라인 (영우글로벌러닝) |
| **교육 내용** | |
| 오전 | **▣ Classification**  -------------------------------------------------------------------------------------------  **∇ training set, test set 으로 데이터 나누기 + target 생성**  \* fish\_data = np.column\_stack((fish\_length, fish\_weight))  \* fish\_target = [1]\*35 + [0]\*14  \* np.ones(35  \* np.zeros(14  \* fish\_target = np.concatenate((np.ones(35), np.zeros(14)  -------------------------------------------------------------------------------------------  **∇ Feature Scaling**  \* mean = np.mean(train\_input)  \* mean = np.mean(train\_input, axis = 0)  \* mean = np.mean(train\_input axis = 1)  \* std = np.std(train\_input, axis = 0)  \* train\_scaled = (train\_input - mean)/std  \* new = ([25, 150] - mean) / std  \* mean\_t = np.mean(test\_input, axis = 0)  \* std\_t = np.std(test\_input, axis = 0)  \* test\_scaled = (test\_input - mean\_t) / std\_t    \* kn.score(test\_scaled, test\_target)  -------------------------------------------------------------------------------------------  **∇ KNN**  \* from sklearn.neighbors import KNeighborsClassifier  \* kn = KNeighborsClassifier()  \* n\_neighbors = 5 k값 설정. default 값은 5  \* weights -> 거리에따라 가중치를 어케할지, uniform이면 거리가 멀수록, distance면 거리가 가까울 수록 더 큰 영향력  \* metric(미터법)은 default 값이 minkowski, p 가 1이면 맨하튼 거리 , 2면 유클리디안 거리 사용  \* leaf\_size = 나눈 영역 개수  \* algorithm -> default 값은 auto임. 사용자가 알아서 해라, brute이면 거리 다 계산함  \*  \* kn.fit(데이터 값 리스트명, 타겟 리스트명) ->  \* kn.score(fish\_data, fish\_target) ->  \* kn.predict([[임의의 x축 값, 임의의 y축 값]]) -> 예측해 볼 값 입력, ★ 반드시 차원을 맞춰줘야함  ------------------------------------------------------------------------------------------- |
| 오후 | **▣ Regression**  -------------------------------------------------------------------------------------------  **∇ 회귀에서 score()는 R^2을 나타냄.**  \* knr.score(test\_input, test\_target)  -------------------------------------------------------------------------------------------  **∇ R^2 리스트 생성하기**  knr = KNeighborsRegressor()  r2\_train = [] # train의 R^2  r2\_test = [] # test의 R^2  neighbor\_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))  neighbor\_n.append(n)  -------------------------------------------------------------------------------------------  **∇ Grid 해줘서 초매개변수 설정해주기**  plt.scatter(neighbor\_n, r2\_train, label ="train")  plt.scatter(neighbor\_n, r2\_test, label = "test")  plt.xlabel("neighbor\_n")  plt.ylabel("R^2")  plt.legend()  plt.show()\*  -------------------------------------------------------------------------------------------  **∇ KNR**  \* from sklearn.model\_selection import train\_test\_split  \*  \* train\_test\_split()  \* array -> 우리가 가지고 있는 데이터  \* test\_size, train\_size -> 따로 넣어주지 않으면 임의로 0.7대 0.3으로 분할해줌  \* random\_state -> 무작위로 섞어줌  \* shuffle -> 섞어줌  \* stratify -> 계층화 해서 섞어줌  \* train\_input, test\_input, train\_target, test\_target = train\_test\_split(  fish\_data, fish\_target, stratify = fish\_target, random\_state = 42)------------------------------------------------------------------------------------------- |