|  |  |
| --- | --- |
| 교육 제목 | **Regression** |
| 교육 일시 | 21.10.18 (월) |
| 교육 장소 | 온라인 (집) |
| **교육 내용** | |
| 오전 | **▣ Regrssion**  -------------------------------------------------------------------------------------------  **∇ score()**  \* Regression에서 score()는 R^2값{= (1-오류)^2}을 출력하게 됨.  -------------------------------------------------------------------------------------------  **∇ Linear Model**  \* 데이터 범위에서 벗어나는 너무 큰 에러 값 데이터가 적용되면  가장 가까운 거리나 index값이 같은 값을 출력함  \* from sklearn.linear\_model import LinearRegression  lr = LinearRegression()  lr.fit(train\_input, train\_target)  \*  \* KNR Model이 아닌, Linear Model에서 초매개변수는  모형을 multi로 쓸 지, poly로 사용할 지 (변수, 특징을 늘릴지) 결정해야 함.  -------------------------------------------------------------------------------------------  **∇ Polynomial Regression**  \* 데이터가 약간 휘어있다면 제곱항을 추가해주면 더 좋음  \* train\_poly = np.column\_stack((train\_input\*\*2, train\_input))  # 제곱 항 추가, 독립인 경우는 순서가 상관이 없음  test\_poly = np.column\_stack((test\_input\*\*2, test\_input))  \* print(lrp.coef\_, lrp.intercept\_) # 파라미터 구한후 방정식 세워서 plot  x\_new = np.arange(10, 50)  y\_new = 75.14 - 17.87\*x\_new + 0.94\*x\_new\*\*2  ------------------------------------------------------------------------------------------- |
| 오후 | **▣ Regularization**  -------------------------------------------------------------------------------------------  **∇ polynomial transform() -> 단위 맞춰줌**  \* from sklearn.preprocessing import PolynomialFeatures\*  train\_poly = poly.transform(train\_input)  test\_poly = poly.transform(test\_input)  -------------------------------------------------------------------------------------------  **∇ Ridge Penalty**  \* Ridge Regression 사용해서 overitting 방지  \* 스케일링 필요  train\_scaled = ss.transform(train\_poly)  test\_scaled = ss.transform(test\_poly)  \* ridge = Ridge()  ridge.fit(train\_scaled, train\_target)  \* Ridge(alpha=n)  alpha -> penalty에서 람다 값임. (default 값은 1)  \* Ridge에서 Lambda(여기서는 alpha) Plot 해보기  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))  -------------------------------------------------------------------------------------------  **∇ LASSO Penalty**  \* from sklearn.linear\_model import Lasso  lasso = Lasso()  lasso.fit(train\_scaled, train\_target) \*  \* LASSO에서 Lambda(여기서는 alpha) Plot 해보기  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) # max\_iter = 10000 ????????????????????????  lasso.fit(train\_scaled, train\_target)    train\_score.append(lasso.score(train\_scaled, train\_target))  test\_score.append(lasso.score(test\_scaled, test\_target))  \* 알파값 조절해서 넣어줌  lasso = Lasso(alpha = 10)  lasso.fit(train\_scaled, train\_target)  print(lasso.score(train\_scaled, train\_target))  print(lasso.score(test\_scaled, test\_target))  ------------------------------------------------------------------------------------------- |