**SVM & 인공신경망**

**SVM(Support Vector Machine)**

**Theory**

서포트 벡터 머신 알고리즘은 주어진 데이터 집합을 바탕으로 하여 새로운 데이터가 어떤 범주에 속할 것인지를 판단하는 비확률적 이진 선형 분류 모델을 생성한다. Svm의 분류 모델은 데이터가 사상된 공간에서 경계로 표현되며, 공간상에 존재하는 여러 경계 중 가장 큰폭을 가진 경계를 찾습니다. 반대로 svm 회귀 모델은 일정한 마진 오류 안에서 두 클래스 간의 도로 폭이 가능한 한 최대가 되도록 하는 대신, 제한된 마진 오류 안에서 도로 안에 가능한 많은 샘플이 들어가도록 학습합니다.

SVM은 데이터 그룹을 나누는 초평면을 찾으면서 이 폭이 가장 넓은 것을 찾는 방법이다. 따라서 데이터의 크기에 민감한 특성을 가지고 있다. 따라서, 데이터 정제를 위해 정규화 혹은 표준화를 수행해 주어여 한다.

sklearn.svm.SVC(\*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache\_size=200, class\_weight=None, verbose=False, max\_iter=- 1, decision\_function\_shape='ovr', break\_ties=False, random\_state=None)

sklearn.svm.SVR(\*, kernel='rbf', degree=3, gamma='scale', coef0=0.0, tol=0.001, C=1.0, epsilon=0.1, shrinking=True, cache\_size=200, verbose=False, max\_iter=- 1)

**C(cost)** : 이론에서 배운 주요 파라미터로써 **어느 정도의 오차를 허용할지에 대한 파라미터입니다.**

**kernel** : 어떤 커**널함수를 사용할지에 대한 파라미터입니다.** 'linear', 'sigmoid', 'rbf', 'poly'가 활용됩니다.

**degree** : **어느 차수까지의 다항차수로 분류를 할 지에 대한 파라미터입니다.** 커널함수가 'poly' 일 때 사용됩니다.

**gamma** : **곡률 경계에 대한 파라미터입니다.** 'rbf', 'poly', 'sigmoid'일 때 튜닝하는 값입니다.

**coef0** : 상수값으로써 'poly', 'sigmoid'일 때 튜닝을 진행합니다.

▷ 결측치, 이상치 -> 표준화 or 정규화 or 로그변환 -> SVM

**인공신경망(MLPClassifier)**

**Theory**

인공신경망 모형은 동물의 뇌신경계를 모방하여 분류 또는 예측하기 위해 만들어진 모형이다. 인공신경망은 가중치를 반복적으로 조정하며 학습하며 뉴런들은 링크로 연결되어 있고, 각 링크에는 수치적인 가중치가 있다. 학습을 통해 본 가중치를 업데이트하는 방향으로 진행된다.

class sklearn.neural\_network.MLPClassifier(hidden\_layer\_sizes=100, activation='relu', \*, solver='adam', alpha=0.0001, batch\_size='auto', learning\_rate='constant', learning\_rate\_init=0.001, power\_t=0.5, max\_iter=200, shuffle=True, random\_state=None, tol=0.0001, verbose=False, warm\_start=False, momentum=0.9, nesterovs\_momentum=True, early\_stopping=False, validation\_fraction=0.1, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, n\_iter\_no\_change=10, max\_fun=15000)

class sklearn.neural\_network.MLPRegressor(hidden\_layer\_sizes=100, activation='relu', \*, solver='adam', alpha=0.0001, batch\_size='auto', learning\_rate='constant', learning\_rate\_init=0.001, power\_t=0.5, max\_iter=200, shuffle=True, random\_state=None, tol=0.0001, verbose=False, warm\_start=False, momentum=0.9, nesterovs\_momentum=True, early\_stopping=False, validation\_fraction=0.1, beta\_1=0.9, beta\_2=0.999, epsilon=1e-08, n\_iter\_no\_change=10, max\_fun=15000)

주요 파라미터

* hidden\_layer\_sizes: tuple, length = n\_layers - 2, default=(100,) The ith element represents the number of neurons in the ith hidden layer.
* activation: {‘identity’, ‘logistic’, ‘tanh’, ‘relu’}, default=’relu’ Activation function for the hidden layer.
* alpha: float, default=0.0001 L2 penalty (regularization term) parameter.
* batch\_size: int, default=’auto’ Size of minibatches for stochastic optimizers. If the solver is ‘lbfgs’, the classifier will not use minibatch. When set to “auto”, batch\_size=min(200, n\_samples)
* learning\_rate{‘constant’, ‘invscaling’, ‘adaptive’}, default=’constant’ Learning rate schedule for weight updates.
* learning\_rate\_init: double, default=0.001 The initial learning rate used. It controls the step-size in updating the weights.