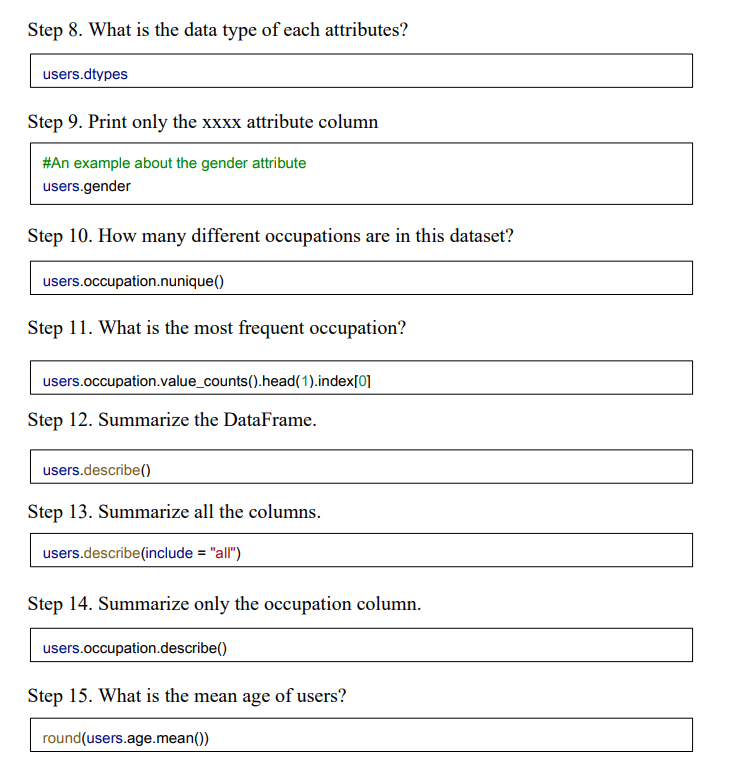
2월 24일

Lecture 1

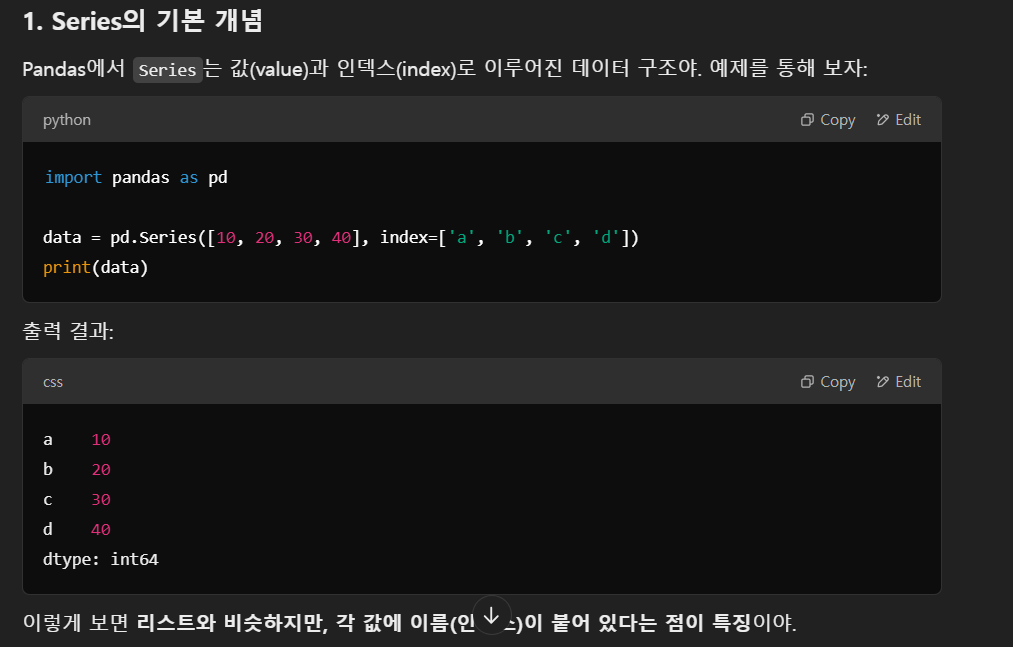
**기본적인 데이터 파악 방법**





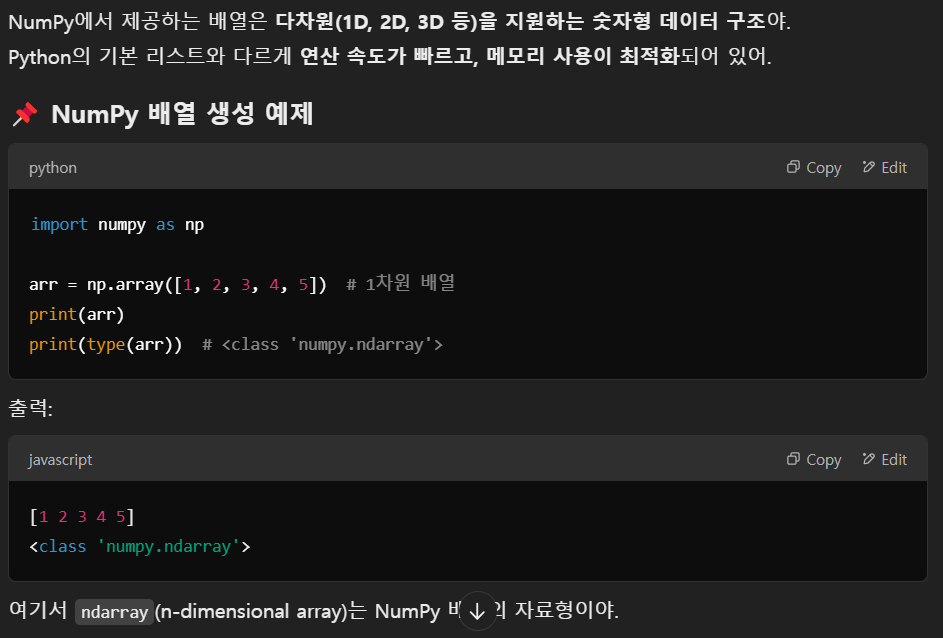
Value\_count()를 사용하면 판다스의 series를 불러옴. 그래서 head() 다음에 index[] 통해서 serise의 인덱스만 불러오는 거임.

**Series란?**



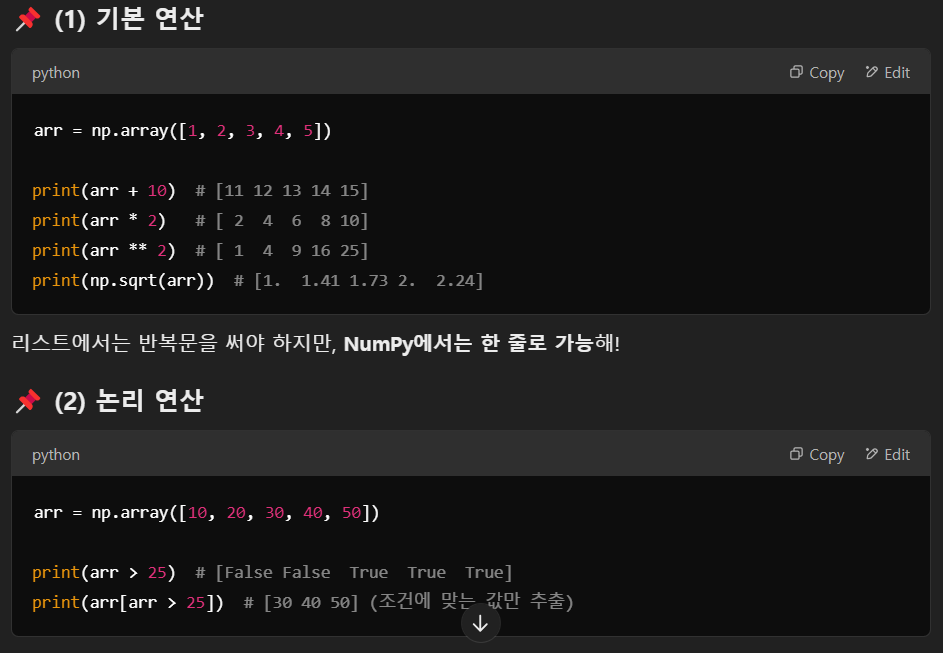
**NumPy 배열(NumPy Array)와 벡터 연산화(vectorisation)란?**

넘파이 배열이 벡터 연산화 특징을 탑재하고 있음.

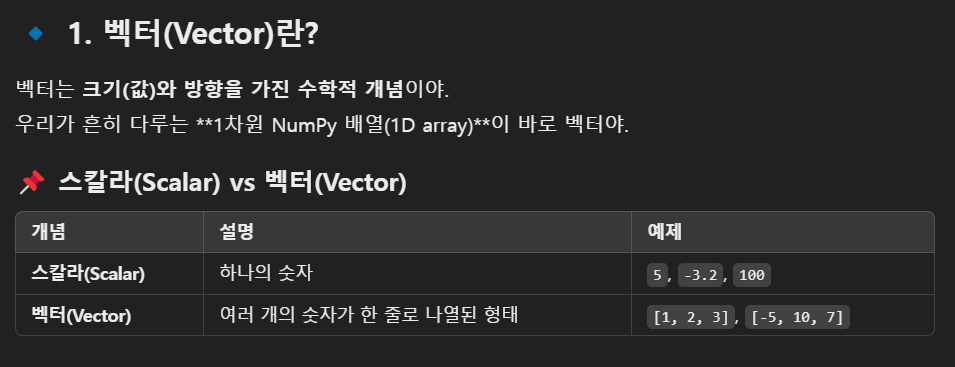




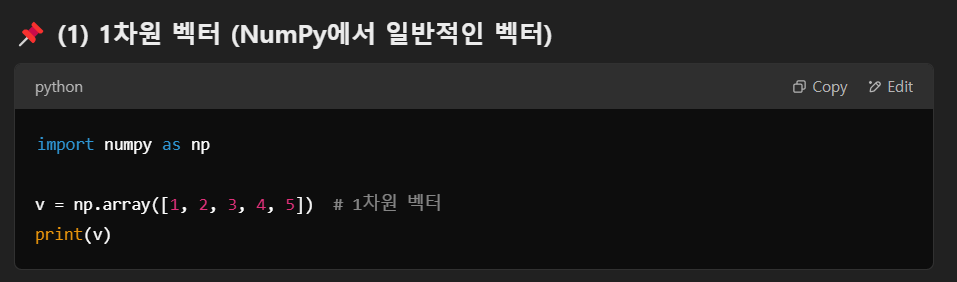
넘파이 추가 예시)



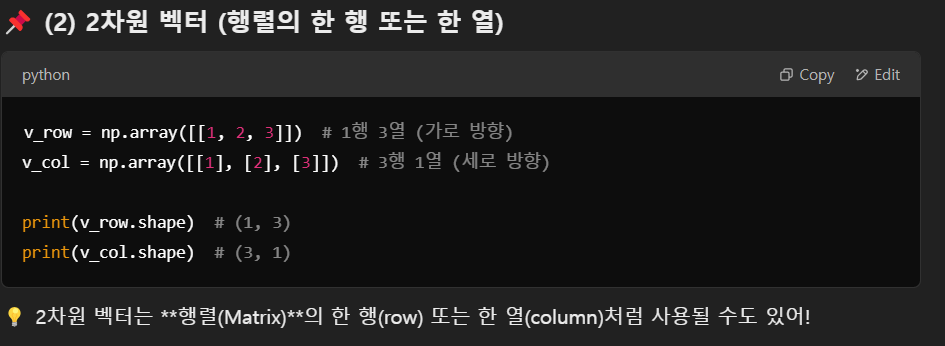
**백터란?**

****

1차원 벡터

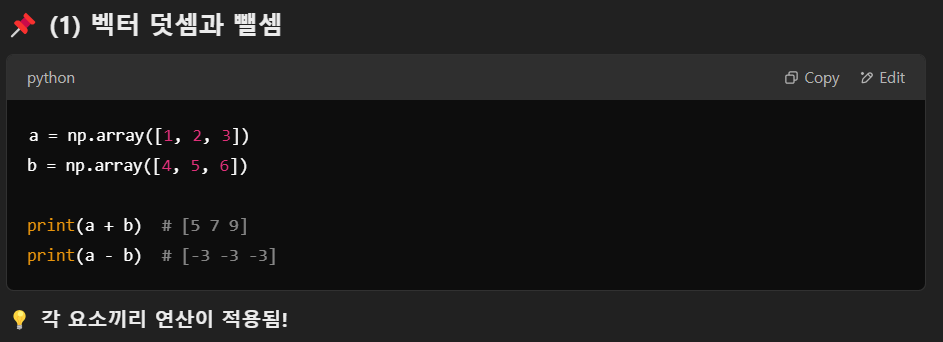
****

2차원 벡터

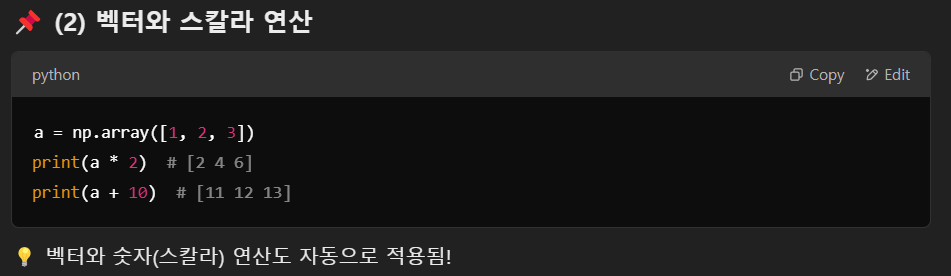
****

**벡터와 벡터 연산 – 각 요소(나열 순서)끼리 연산 됨.**

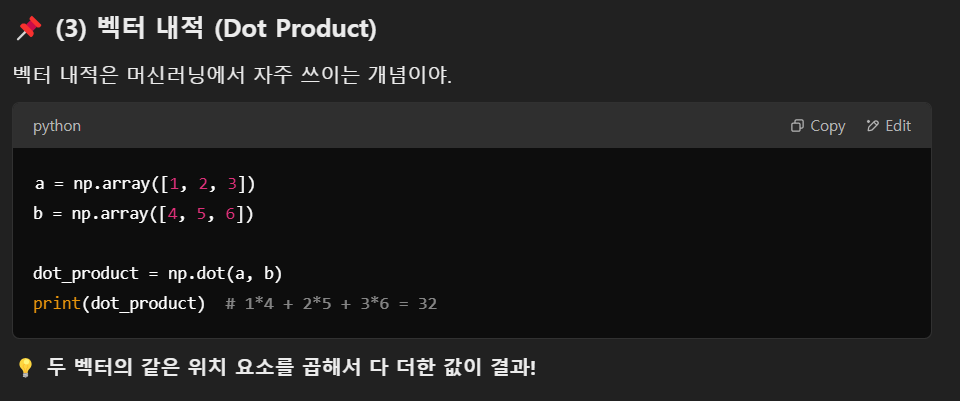
벡터와 덧셈과 뺄셈

****

벡터와 스칼라 연산

****

벡터 내적(Dot Product)

****

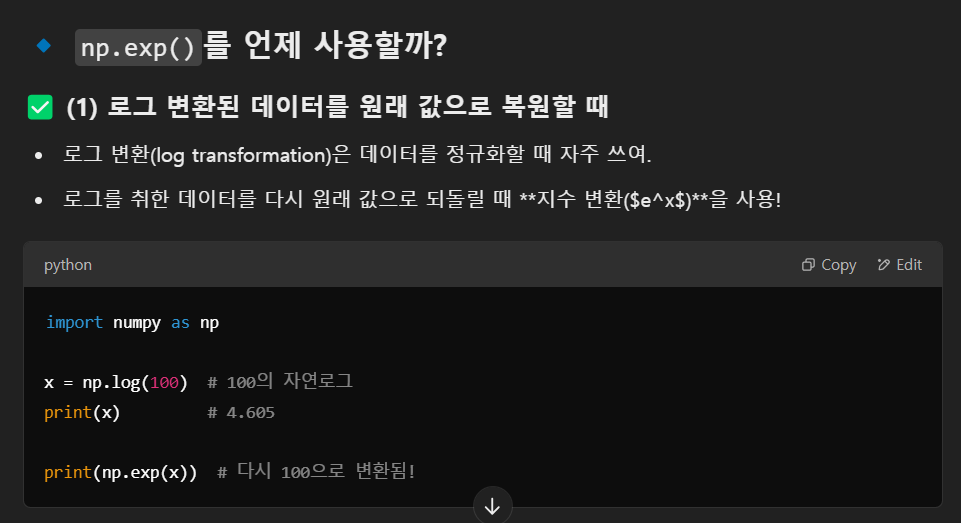
**np.exp()가 하는 일**

지수로 나타낸다는 것은 자연상수 $e$(2.718)를 밑(base)으로 하는 지수 함수(Exponential Function)를 적용한다는 뜻이야.

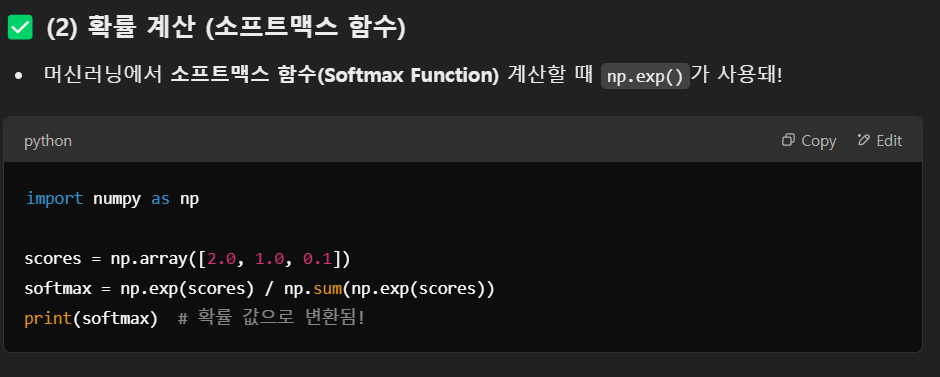
예를 들어 np.exp(x)는 자연상수에 제곱인 거야. 즉, 위에서 말했듯이 지수로 나타내주는 거지.

**np.exp()를 언제 사용할까?**

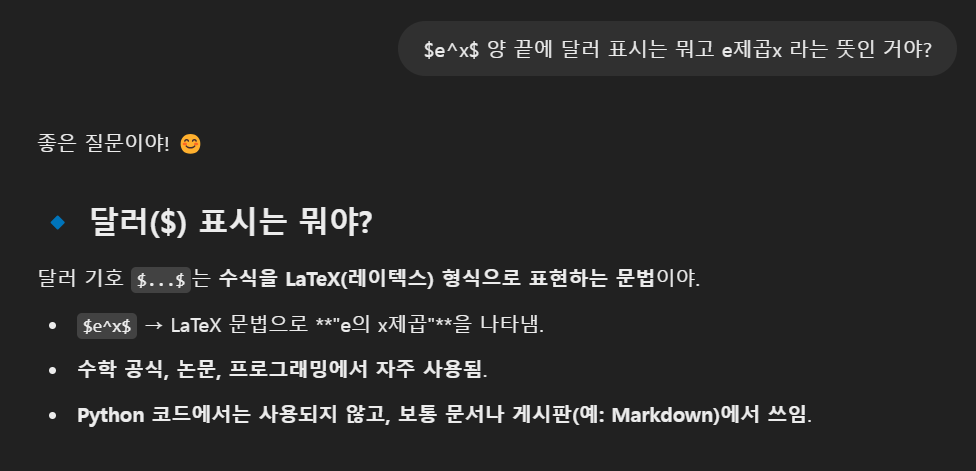
로그 변환을 원래 값으로 변환할 때

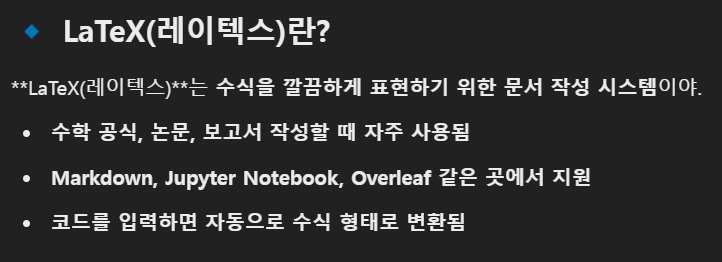
****

확률 계산(소프트맥스 함수)

****

**달러 표시(레이텍스)**

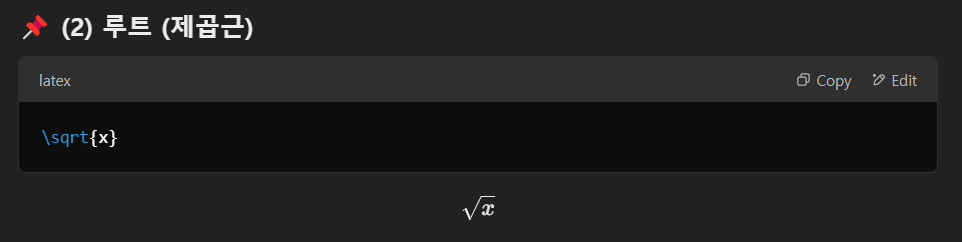




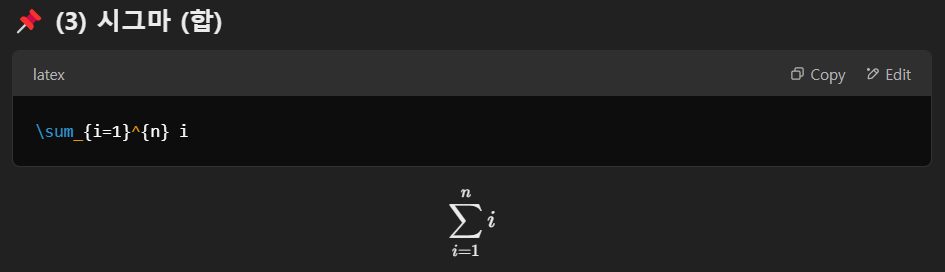
레이텍스 분수



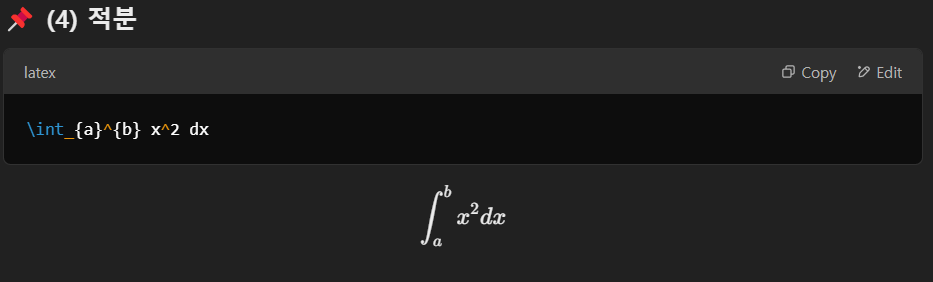
레이텍스 루트



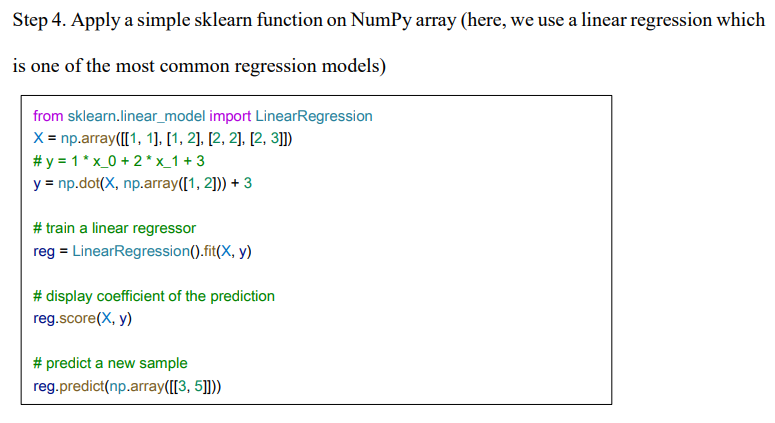
레이텍스 시그마



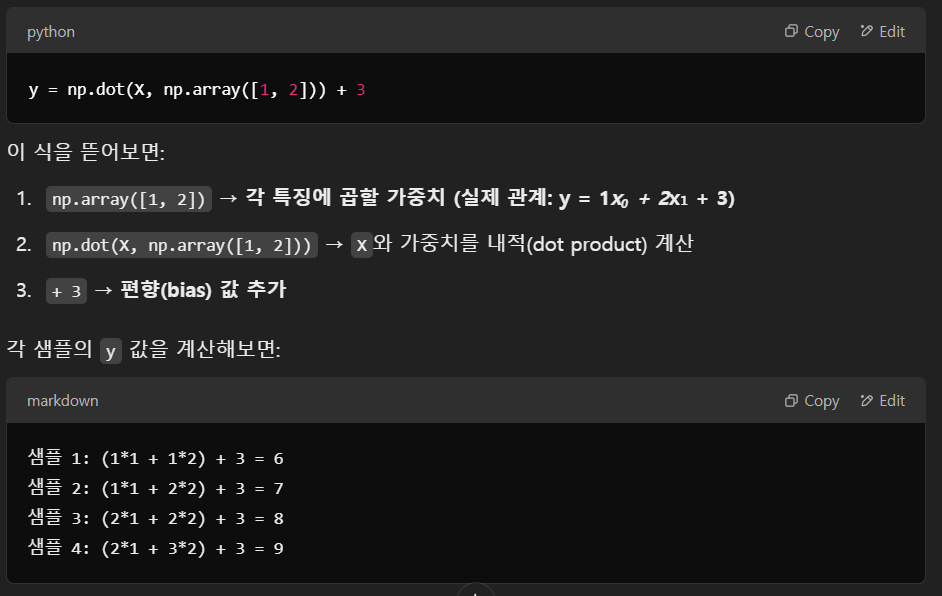
레이텍스 적분



**Sklearn.linear\_model (선형 회귀 모델)**



X는 2차원 배열 형식을 띄고 있어 1행과 2열, 그리고 y는 벡터 내적을 통해서 X의 각 순서 열에 1과 2를 각각 곱해서 +3(bias, 편향 값)한 값을 구할 수 있어.



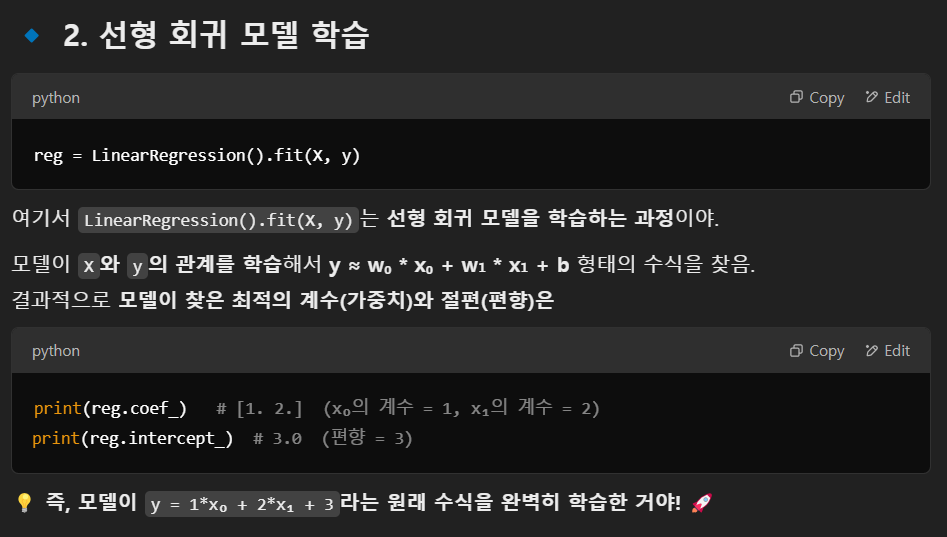
즉, y = np.array([6, 7, 8, 9])

이제 모델을 학습할 데이터 (x,y)가 준비 됨!

**선형 회귀 모델 학습**

reg = LinearRegression().fit(X, y)

계수란?

****

**R^2 결정계수 모델 평가**

reg.score(X, y)

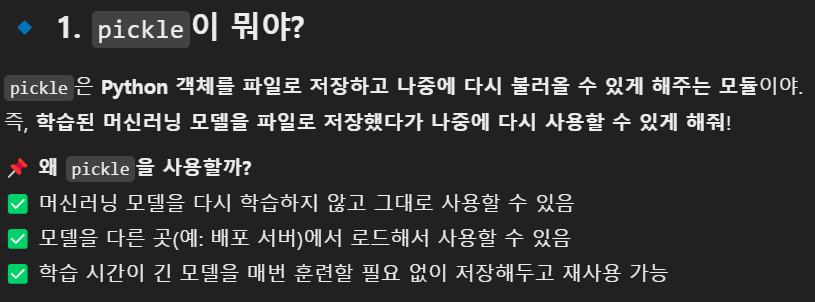
****

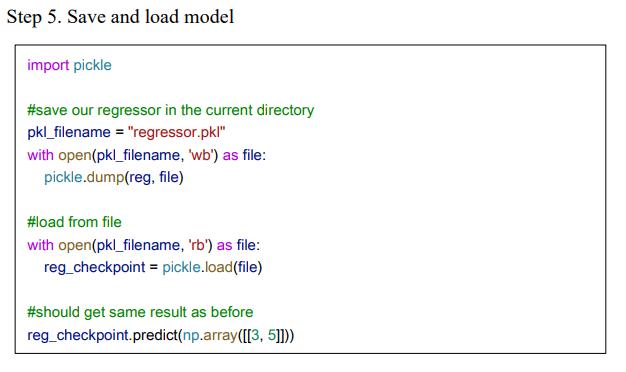
**새로운 샘플 예측 (predict)**

****

**Save and load model**

Pickle

****

****

모델 저장

****

모델 불러오기

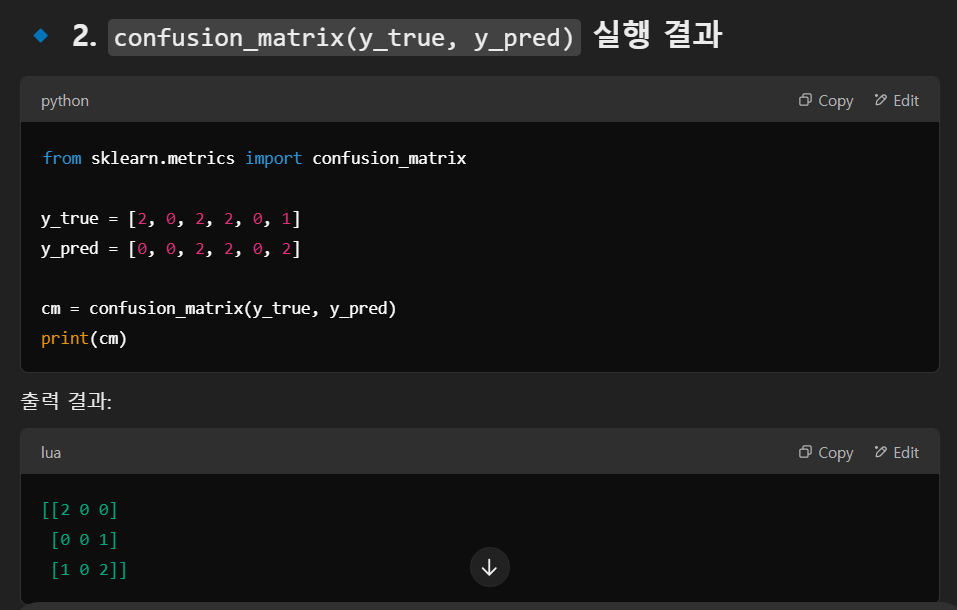


Matplotlib 시각화

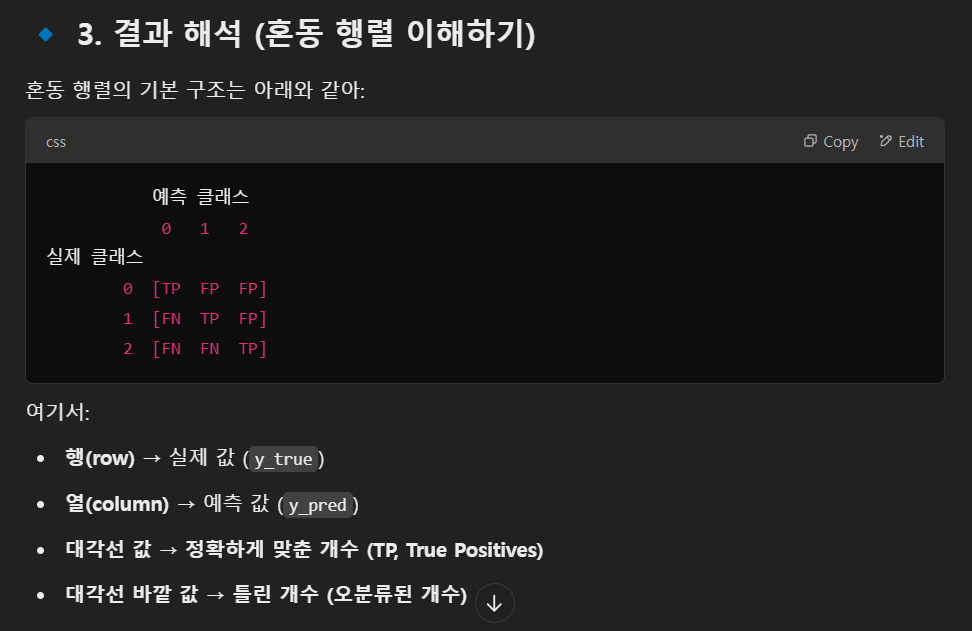


**Confusion Matrix**

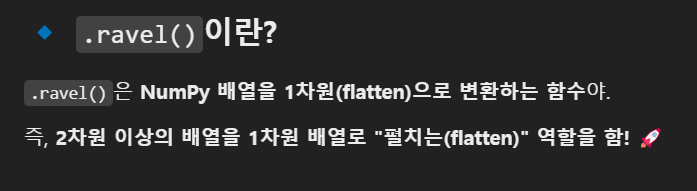
실행 결과

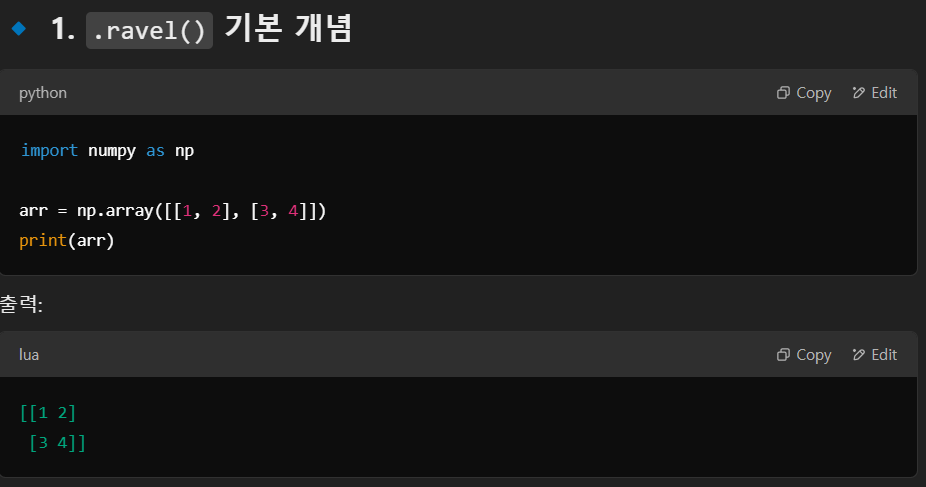


해석 결과



**.ravel()**







**.ravel() Confusion Matrix에 사용하기**

****

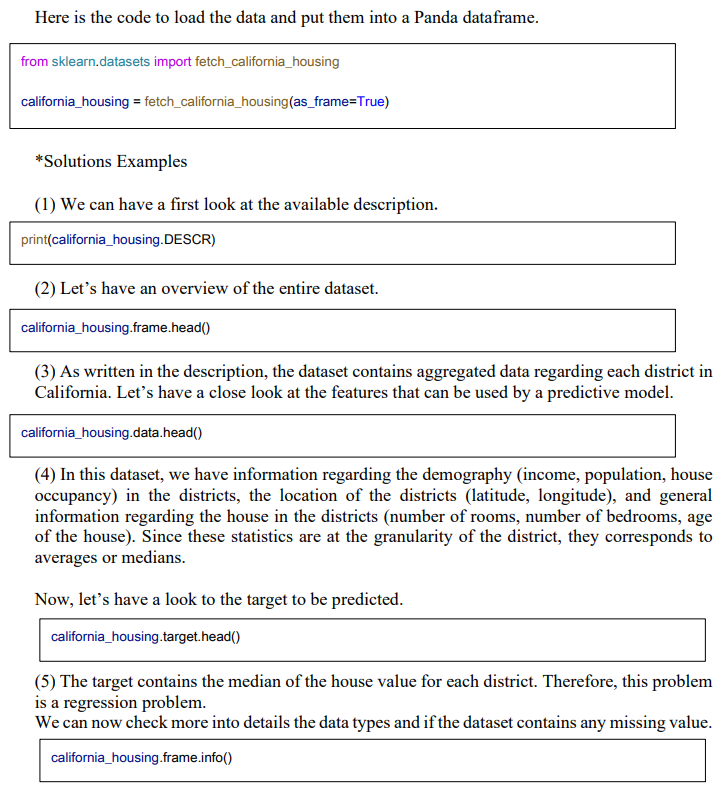
적용 후



근데 여기서 ravel()을 통해 나오는 순서가 TN, FP, FN, TP이기 때문에 이 순서에 맞게 쓴 거임. 만약 순서 다르게 하면 값과 이름이 불일치함.

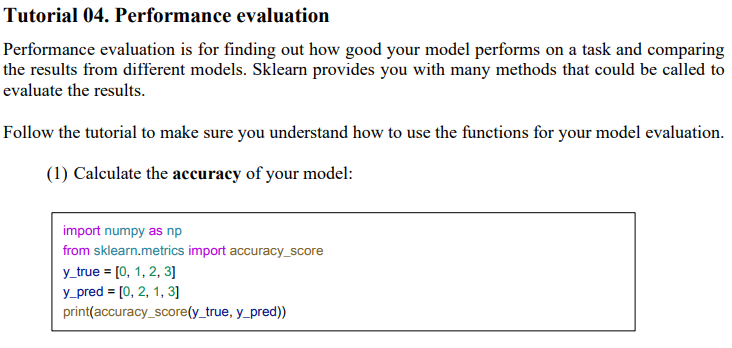
**데이터 사전 작업 - preliminary analysis of datasets**

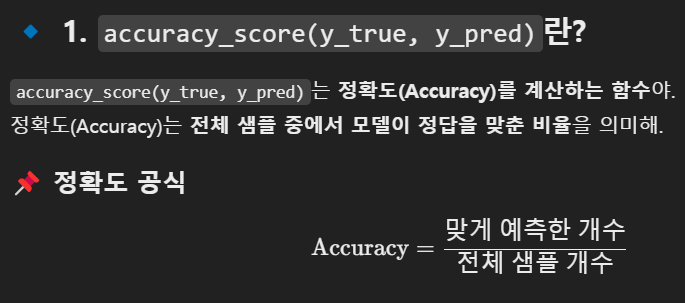
1페이지의 “기본적인 데이터 파악 방법”에 더불어 분석의 첫 번째 단계.

****

**Performance Evaluation (평가 방법)**

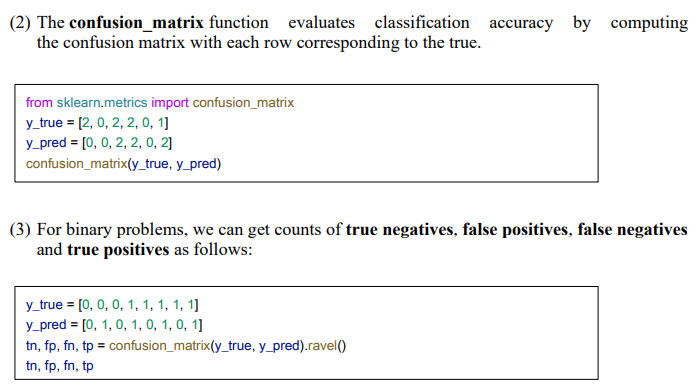
정확도(Accuracy score)

****

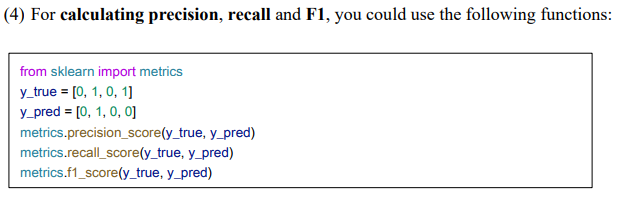
****

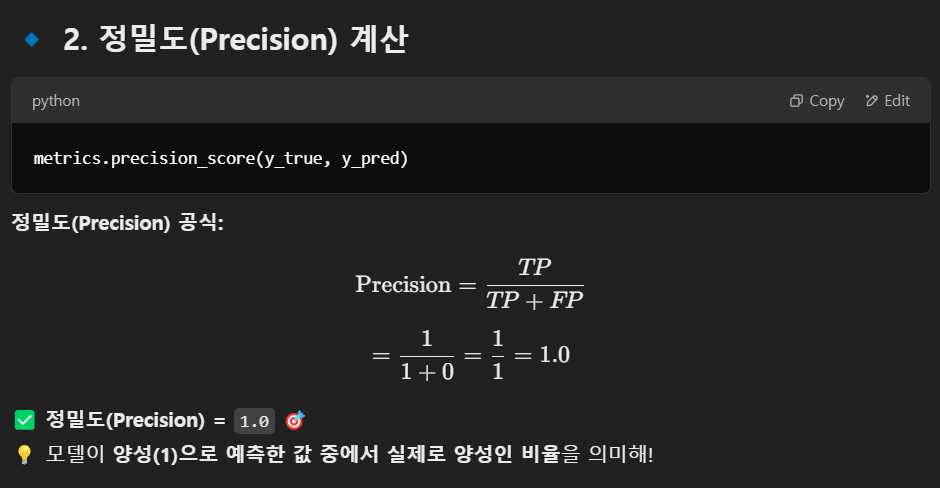
Confusion Matrix (Evaluate Accuracy)

정확도를 평가하는 방법 중 하나로 위에 보다 자세한 설명이 있음

****

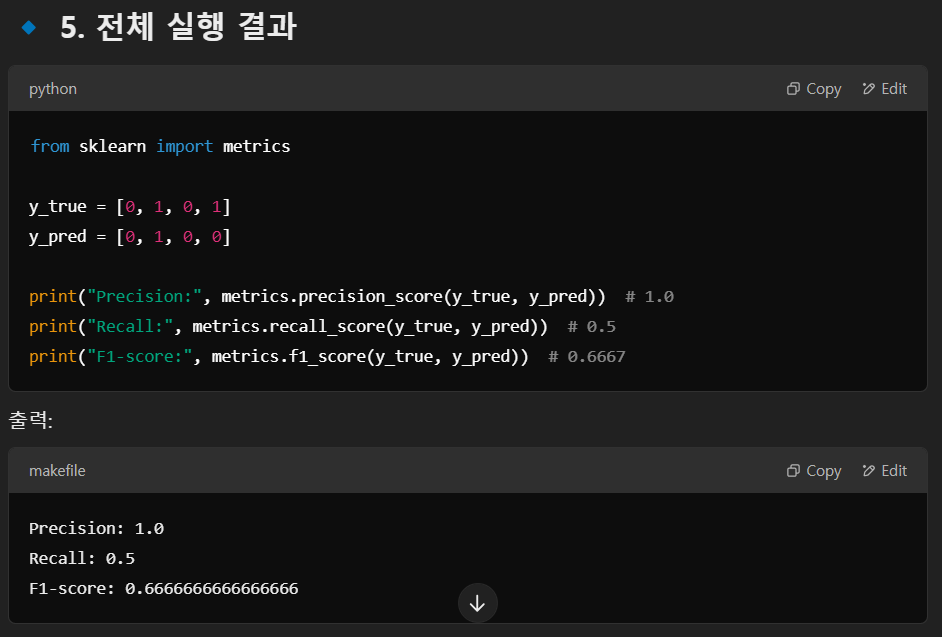
정밀도(Precision), 재현율(Recall), F1 점수(F1-score)

****

****

****

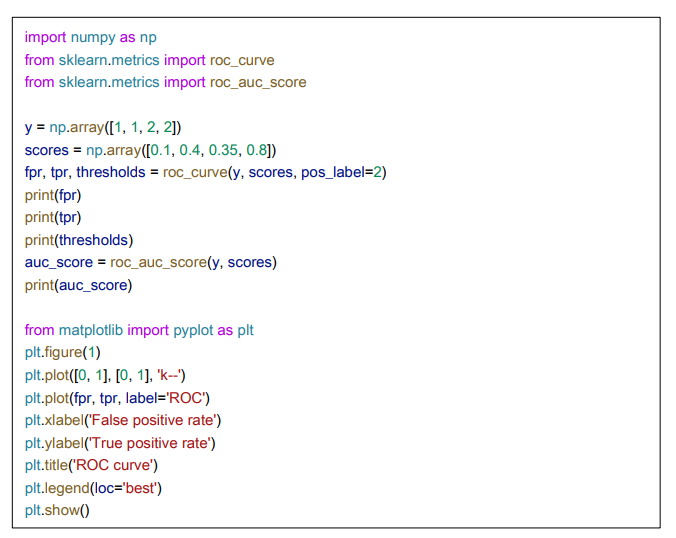
****

****

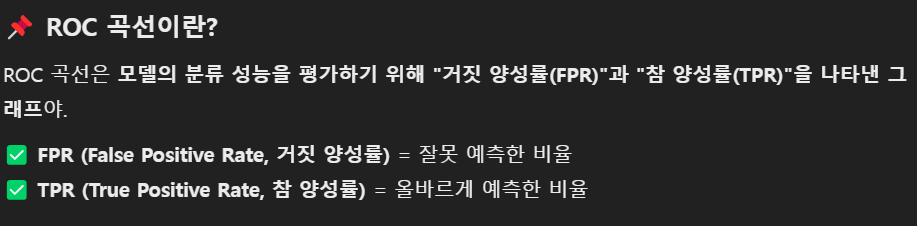
ROC(Receiver Operating Characteristic) 곡선을 그리고 AUC(Area Under the Curve) 점수를 계산

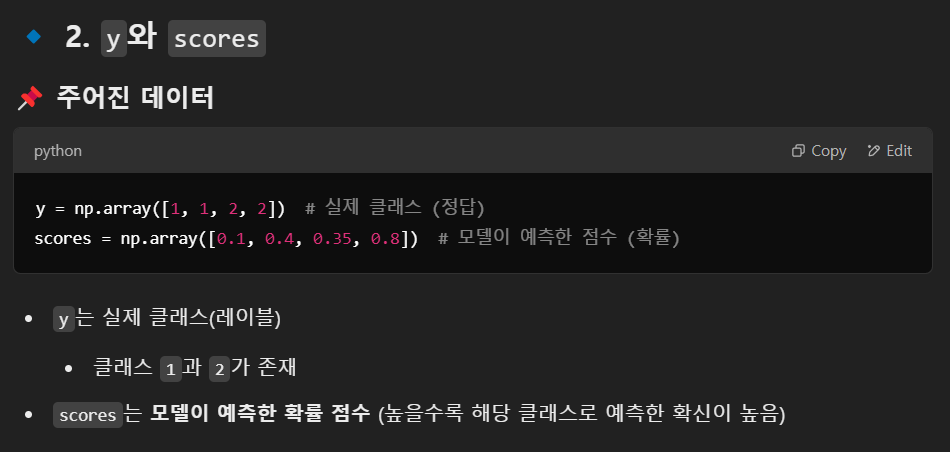
The function roc\_curve computes the receiver operating characteristic curve or ROC

curve.



ROC 곡선은 이진 분류(binary classification) 모델의 성능을 평가하는 방법

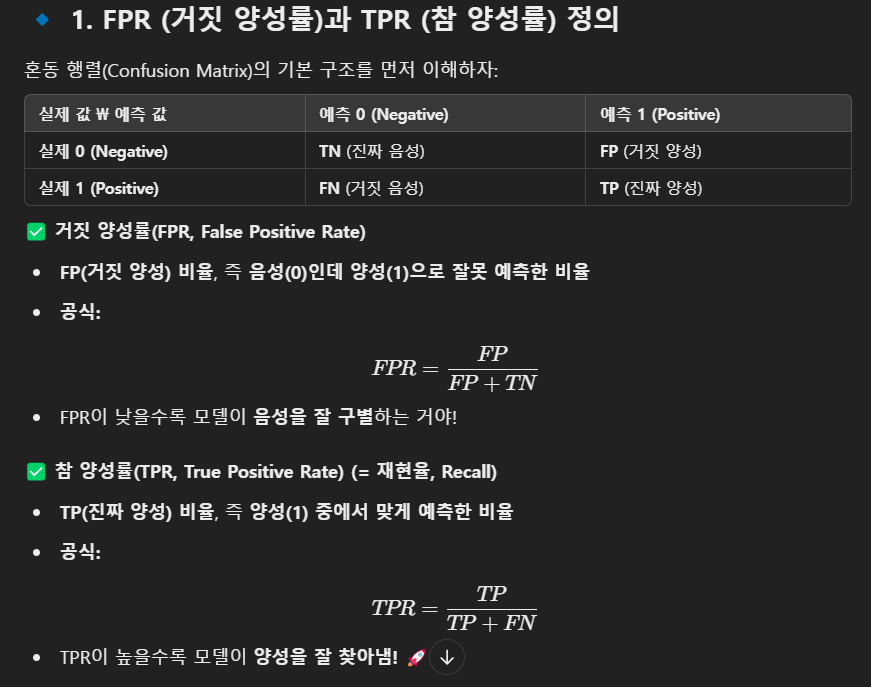




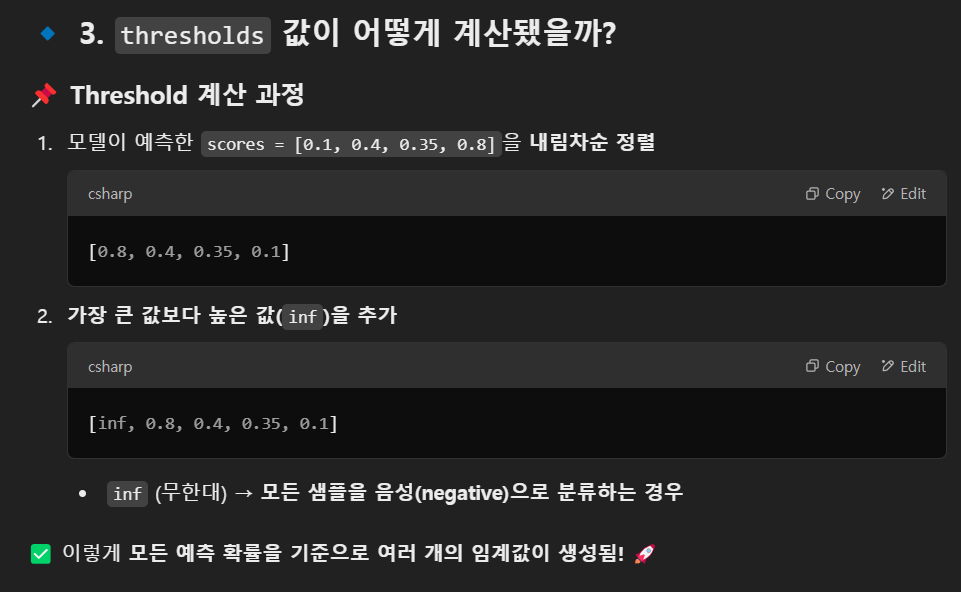




FPR,TPR 계산하기

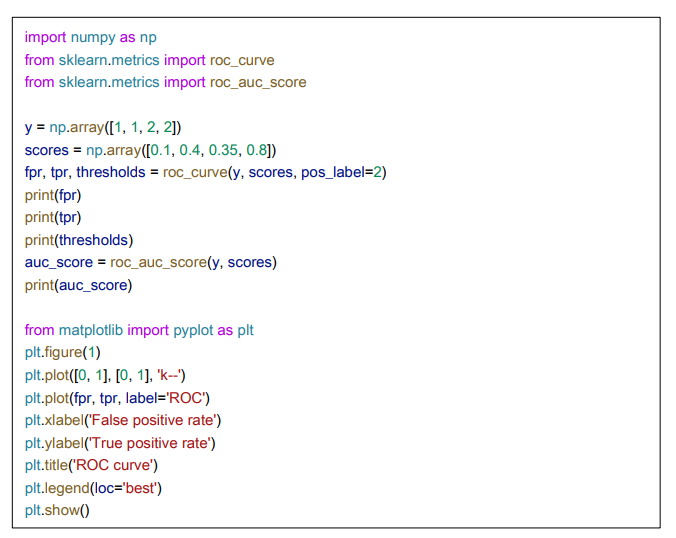


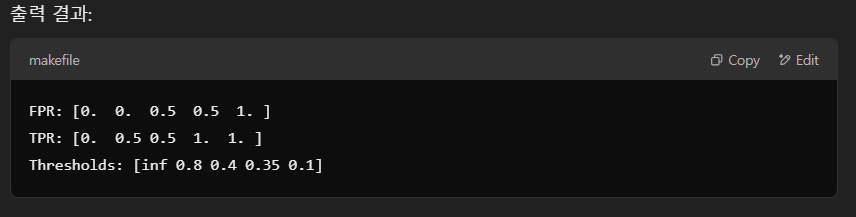
임계값(thresholds) 계산하기



다시 예제로 돌아와서, ROC 곡선에서는 thresholds 값들을 기준으로 모든 샘플을 분류하면서 FPR(거짓 양성률)과 TPR(참 양성률)을 계산해.

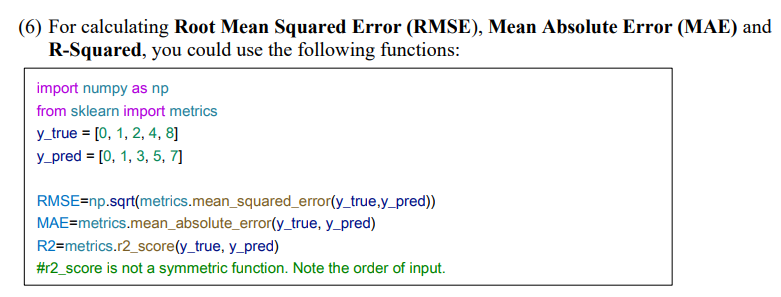
그렇다면 아래 코딩에서 임계값을 통해서 FPR 과 TPR 은 어떻게 자동 계산이 되어서 아래의 출력 결과로 산출되는 걸까?



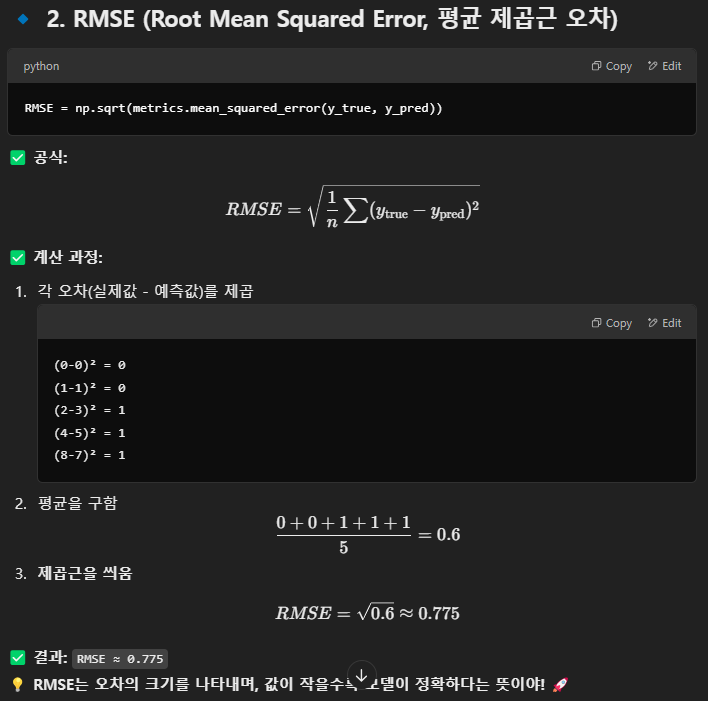


먼저 위에서 설명이 되어 있듯이 먼저 모델 예측 확률을 내림차순을 사용해 임계값을 구함. 그리고 무한대부터 0.1일까지를 각 기준으로 삼아서 코딩의 score와 비교. 무한대의 경우 무조건 음수이기 때문에 예측값이 모두 1임. 그래서 confusion matrix에 TP 가 0, FP 가 0, TN 가 2 FN 이 2 임. 이 경우에 위에 FPR, TPR 공식으로 계산을 해보면 각각 0이 나오는 거임 이렇게 임계값 마다 반복하면 위의 출력 결과 속의 가 생성이 됨. 그리고 이 값을 바탕으로 코딩의 ROC 커브가 그려지는 거임. 즉, ROC 커브는 여러 임계값을 바탕으로 FPR, TPR 을 계산해서 그리는 거임.

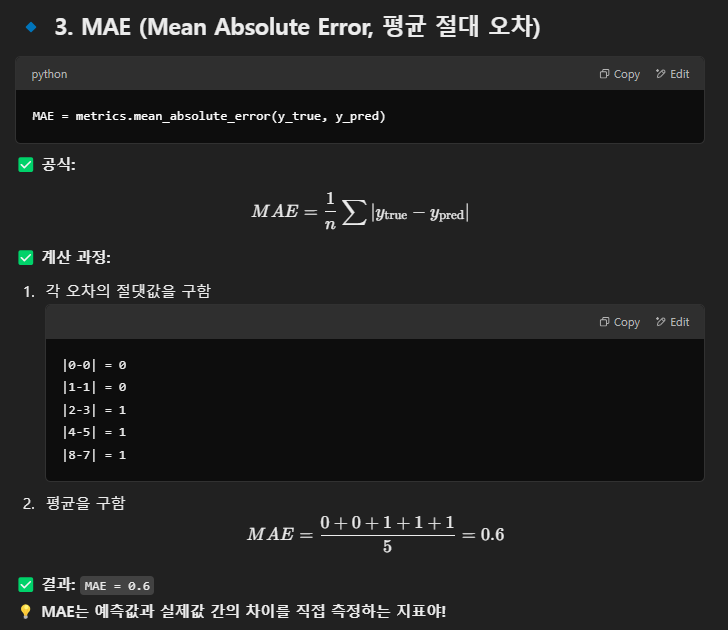
회귀 모델 성능을 평가하는 세 가지 지표



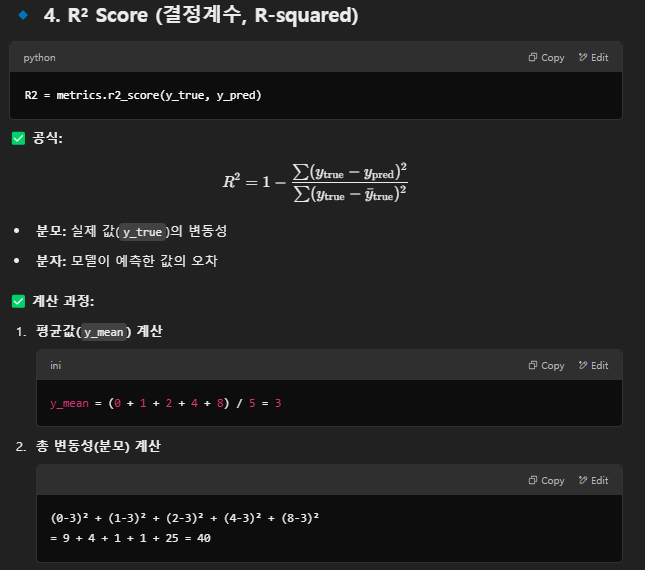
RMSE (평균 제곱근 오차)

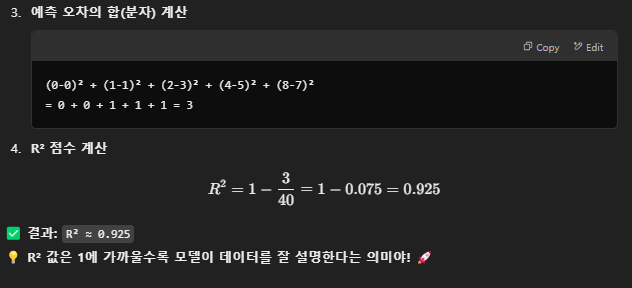


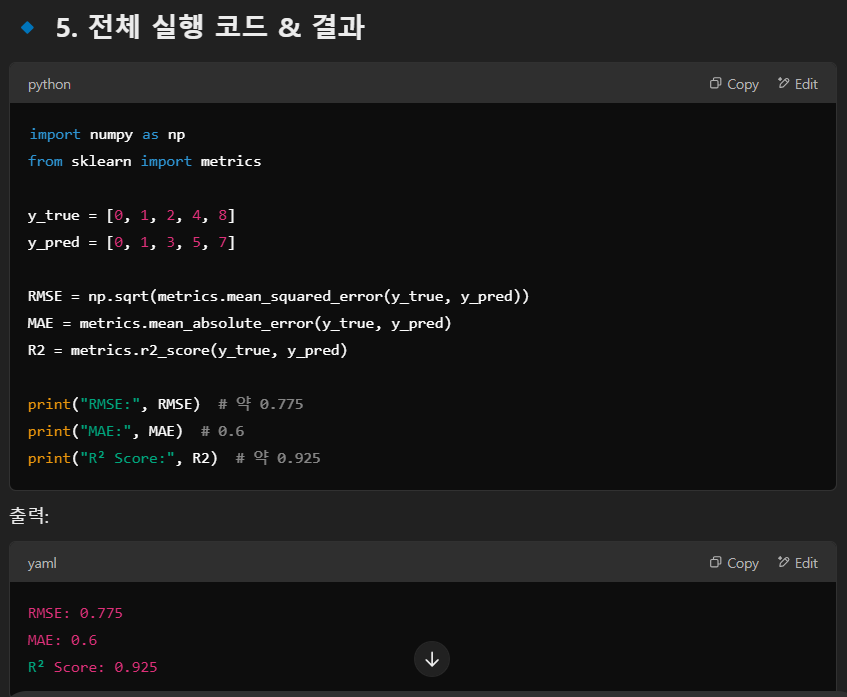
MAE (평균 절대 오차)



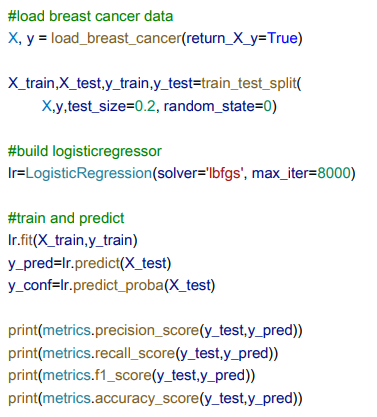
R2(R^2) Score (결정계수)







Lab 1 예제 2

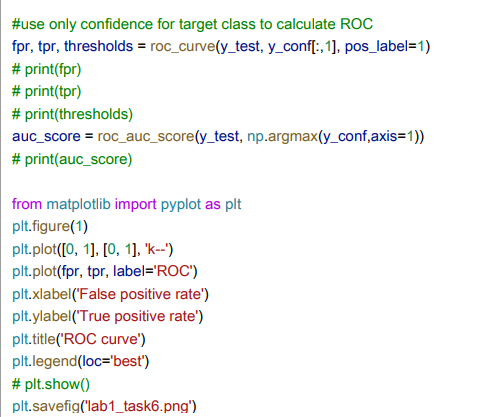


먼저, 왜 모델을 트레이닝 할 때 X\_train, y\_train을 사용할까?

X\_train을 이진화 한게 y\_train인데 단순히 y\_train 데이터로 훈련을 시키면 안되나 생각했었다. 존나 멍청한 생각이다. 모델의 목적이 원본 데이터 X의 30개의 열 중 29개를 사용해 target 열을 예측하는 거기 때문에 필수적인 데이터인 것이다.

둘째, 왜 여기서 precision, recall, f1 score, accuracy score을 구하는데 y\_test와 y\_pred를 쓸까? 위의 질문의 연장선인 부분이다. X\_test 데이터를 사용해서 맞춰야 할 정답이 y\_test이다. 그렇기 때문에 여측값인 y\_pred와 답안지 y\_test를 비교해서 네 가지의 평가값을 구하는 것이다.

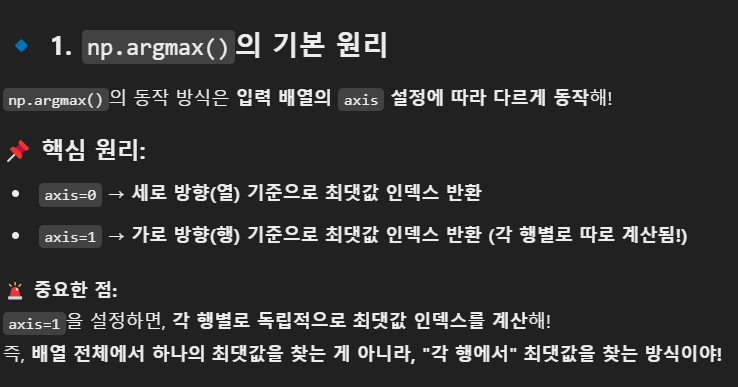
이어서 ->

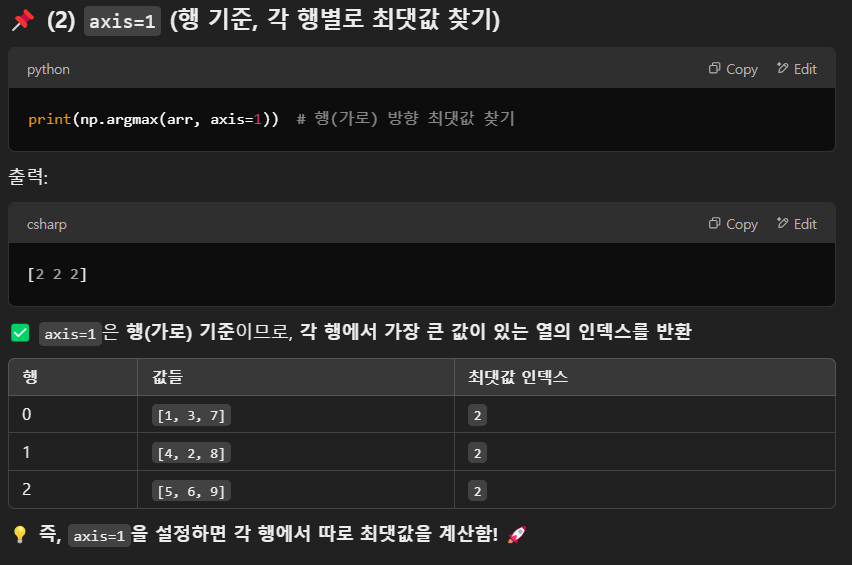


세 번째, FPR, TPR, thresholds 구하기

fpr, tpr, thresholds = roc\_curve(y\_test, y\_conf[:,1], pos\_label=1) // 이 코드는 roc 커브를 그리는 거자나. roc\_curve() 안에 들어가는 값들을 보면 y\_test (정답), 모든 예측중에 값이 1로 양성일 확률, 그리고 1이 양성이라는 설정. 이때, 모든 예측중에 값이 1로 양성일 확률 통해 thresholds를 구하고 자동 계산으로 fpr, tpr 구하는 거임.

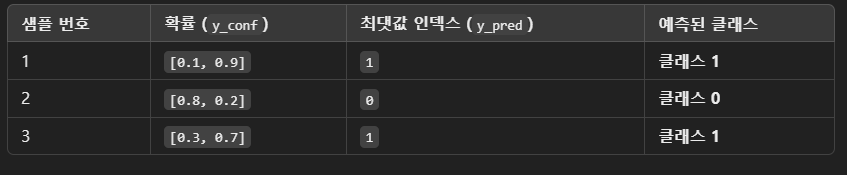
네 번째, AUC 스코어는 위에서는 코딩이 roc\_auc\_score(y, score(확률)) 이라고 되어 있지만 여기서는 np.argmax(y\_conf, axis=1)) 사용함. 이거 대신 y\_conf[:,1]를 쓰면 안되나 싶음!

np.argmax() 는



그래서 행을 기준으로 최댓값인 열을 결과값이 되는 거임.

위 코딩을 사용한 예시)



**2월 25일**

**Lecture 2**

**Data Division (데이터 디비젼) 코딩 예시**

import numpy as np

from sklearn.model\_selection import Kfold

X = np.array([[1, 2], [3, 4], [1, 2], [3, 4]])

y = np.array([1, 2, 3, 4])

kf = KFold(n\_splits=2)

kf.get\_n\_splits(X)

print(kf)

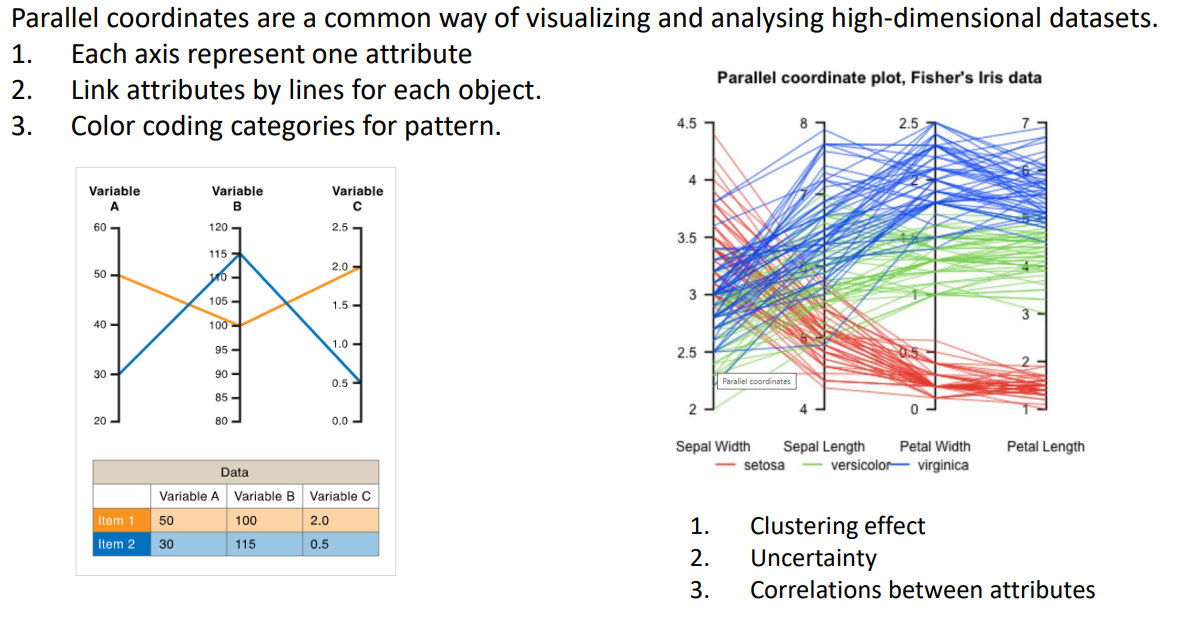
for train\_index, test\_index in kf.split(X):

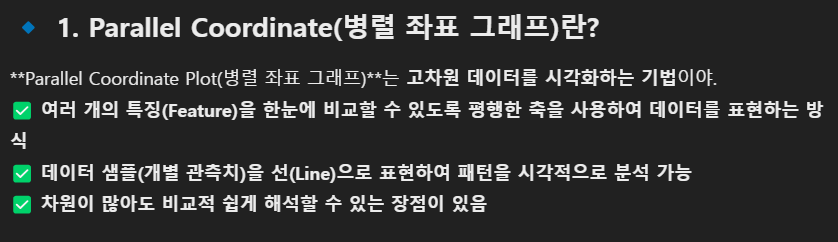
print("TRAIN:", train\_index, "TEST:", test\_index)

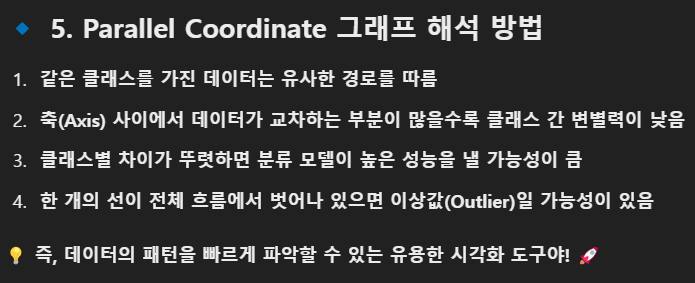
X\_train, X\_test = X[train\_index], X[test\_index]

y\_train, y\_test = y[train\_index], y[test\_index]

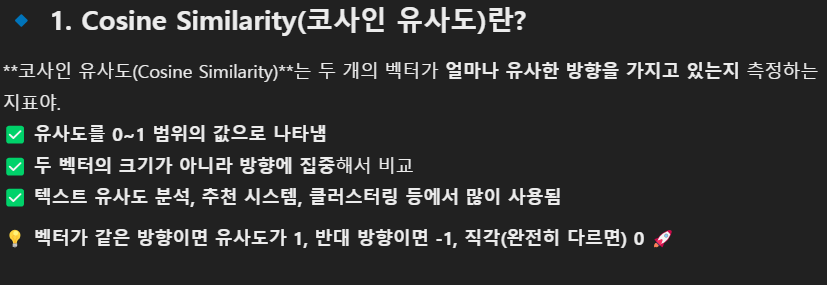
**Parallel Coordinate**

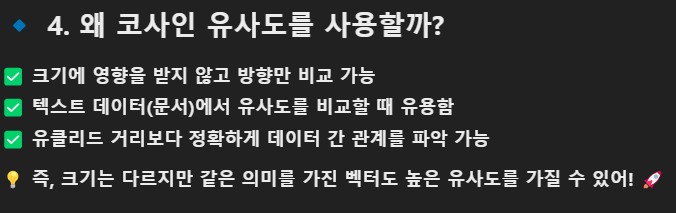




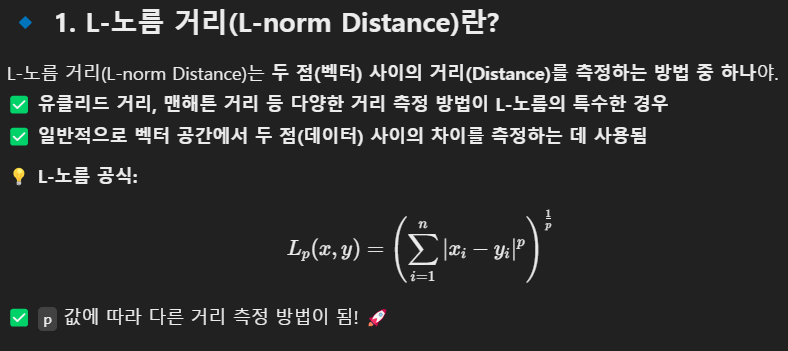


**Cosine Similarity (코사인 유사도)**

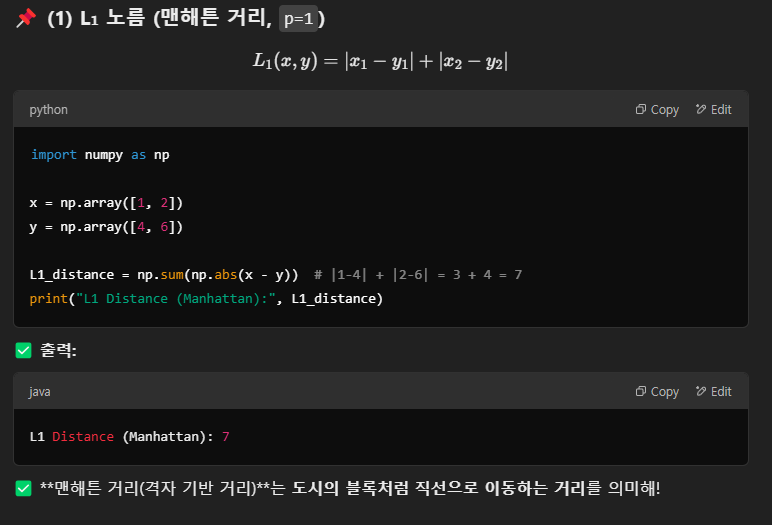




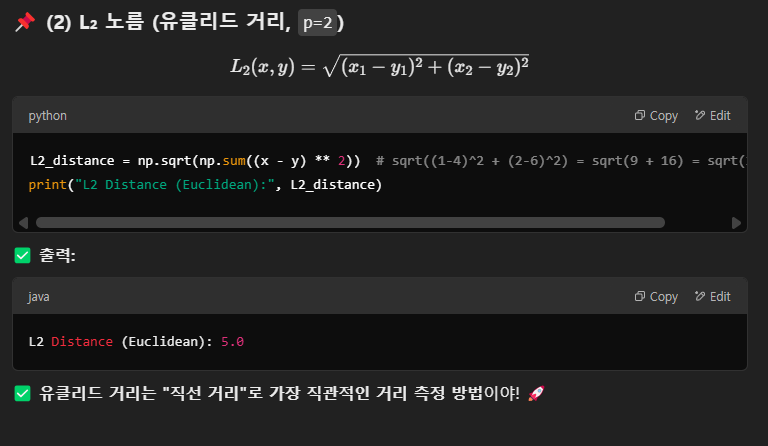
L norm distances (L-노름 거리)



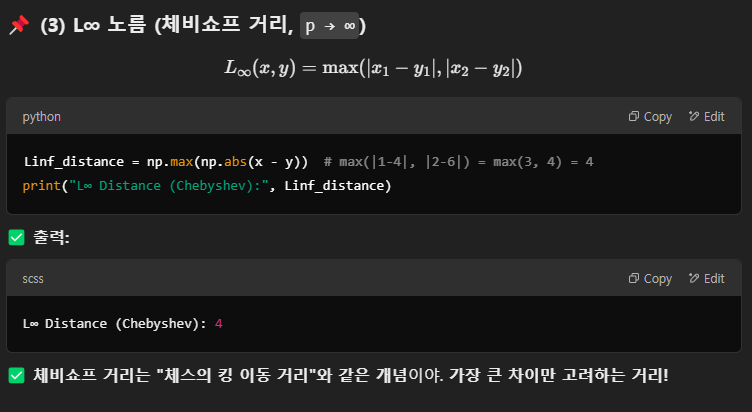
맨해튼 거리



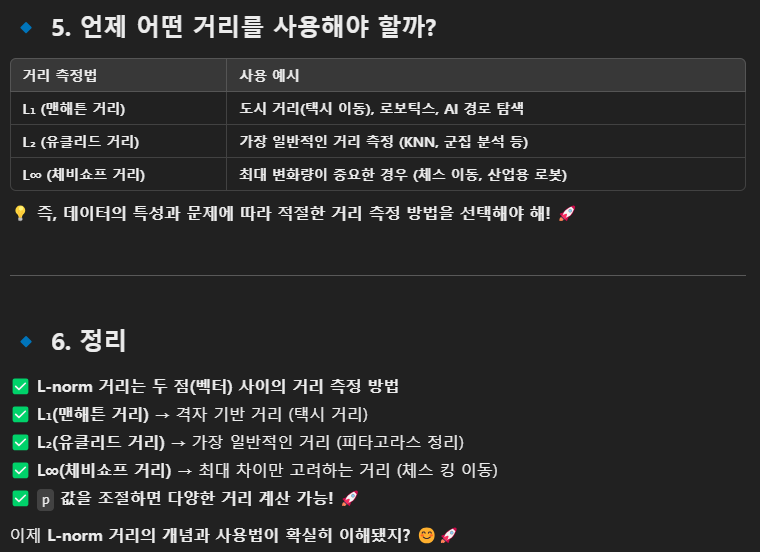
유클리드 거리



체비쇼프 거리



그래서 이 개념을 언제, 왜?



**코사인 유사도 코딩 예시**

#Importing required libraries

from sklearn.datasets import load\_iris

from sklearn.cluster import AgglomerativeClustering

import numpy as np import matplotlib.pyplot as plt

#Getting the data ready

data = load\_iris() df = data.data

#Selecting certain features based on which clustering is done

df = df[:,1:3]

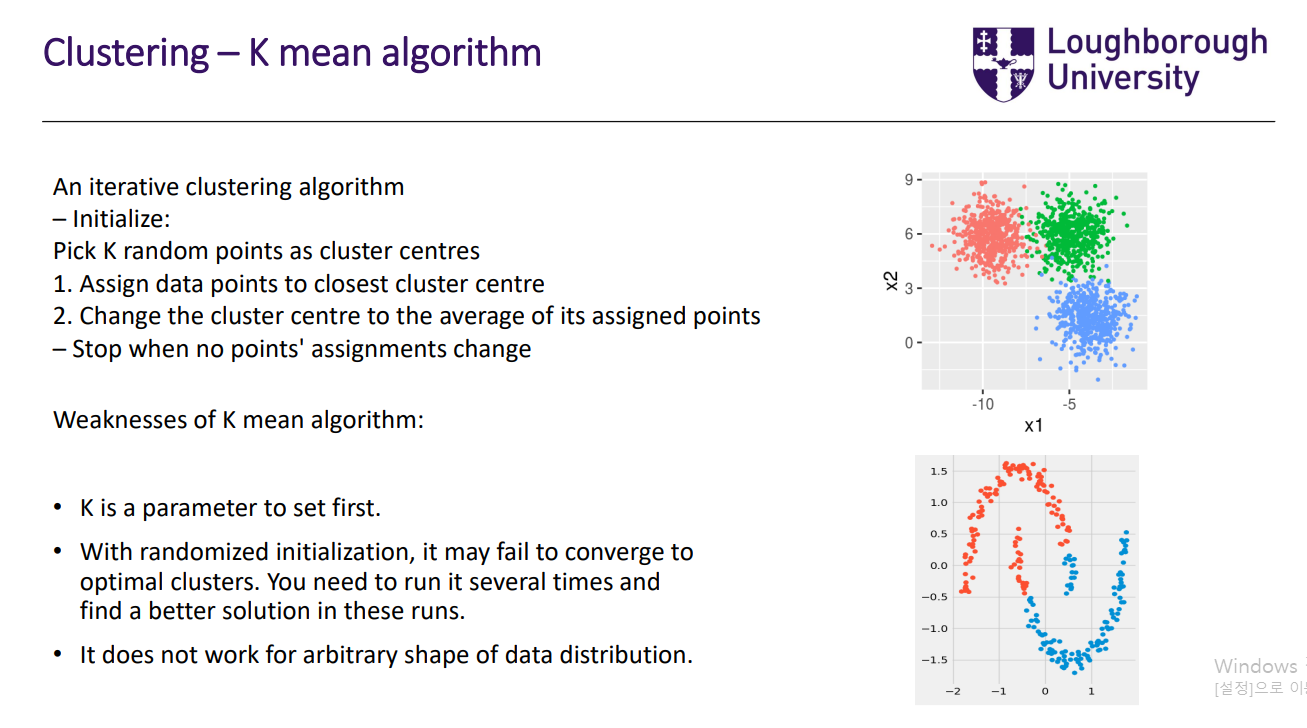
from sklearn.metrics import DistanceMetric

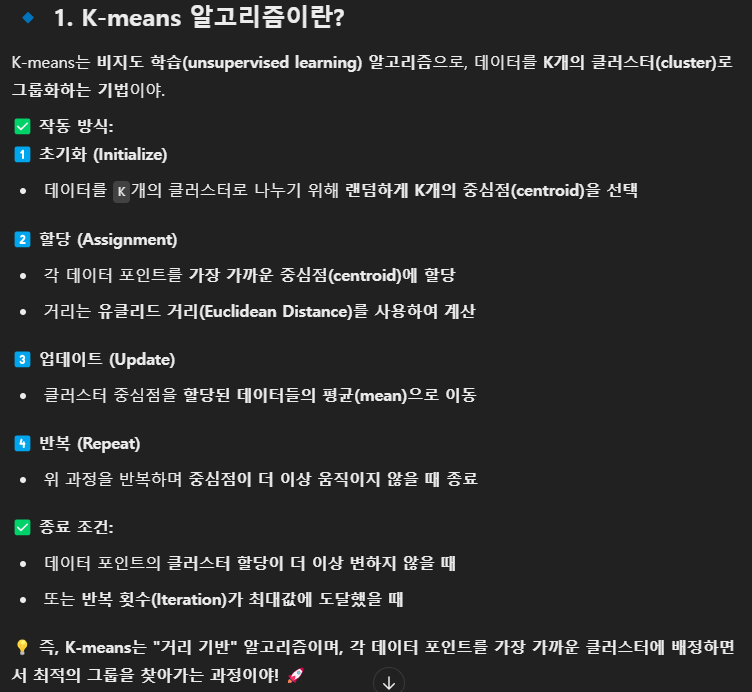
dist = DistanceMetric.get\_metric('euclidean')

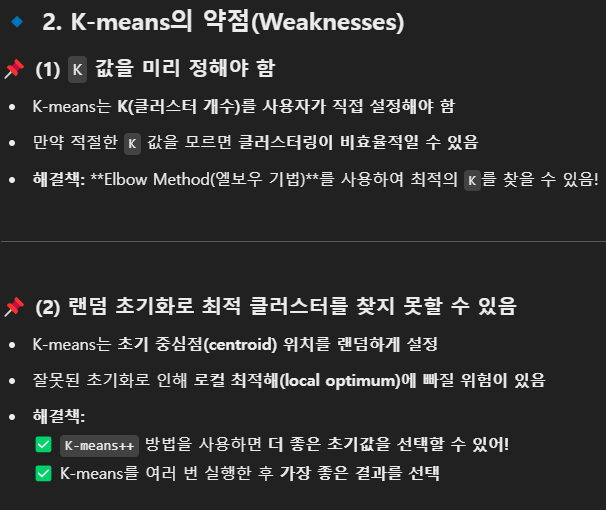
print(dist.pairwise(df))

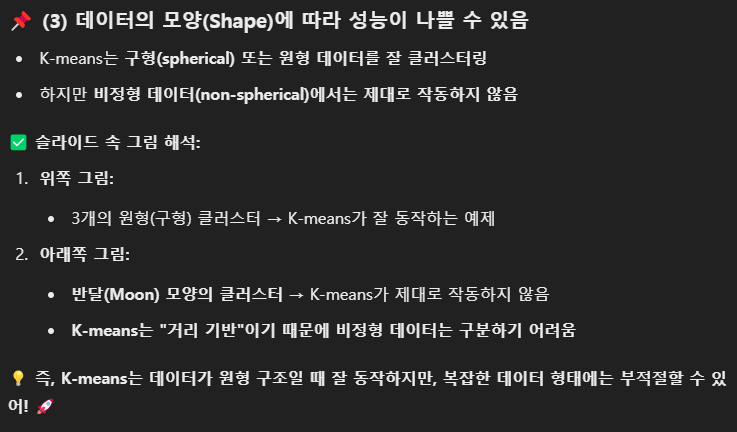
**Clustering – K mean algorithm**

개념과 약점

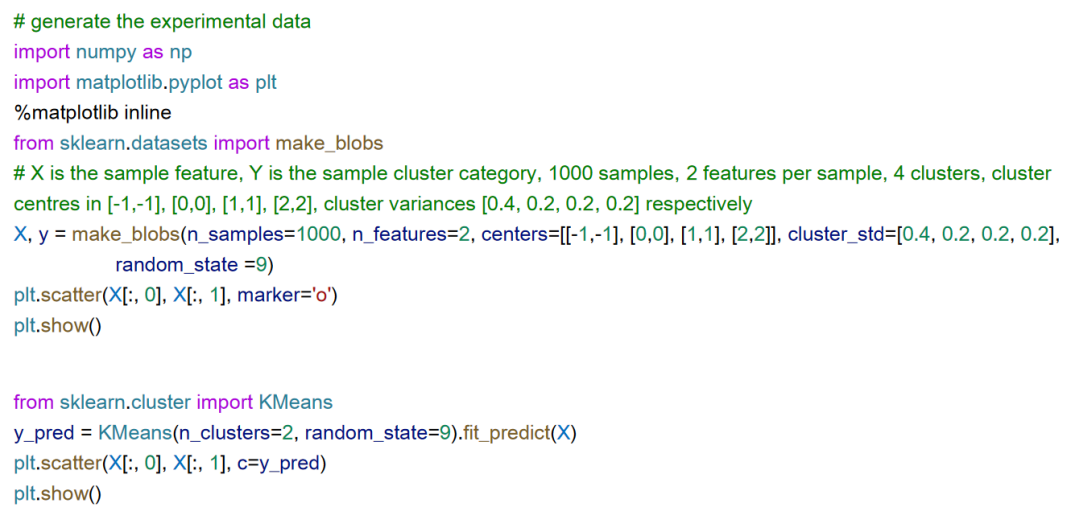




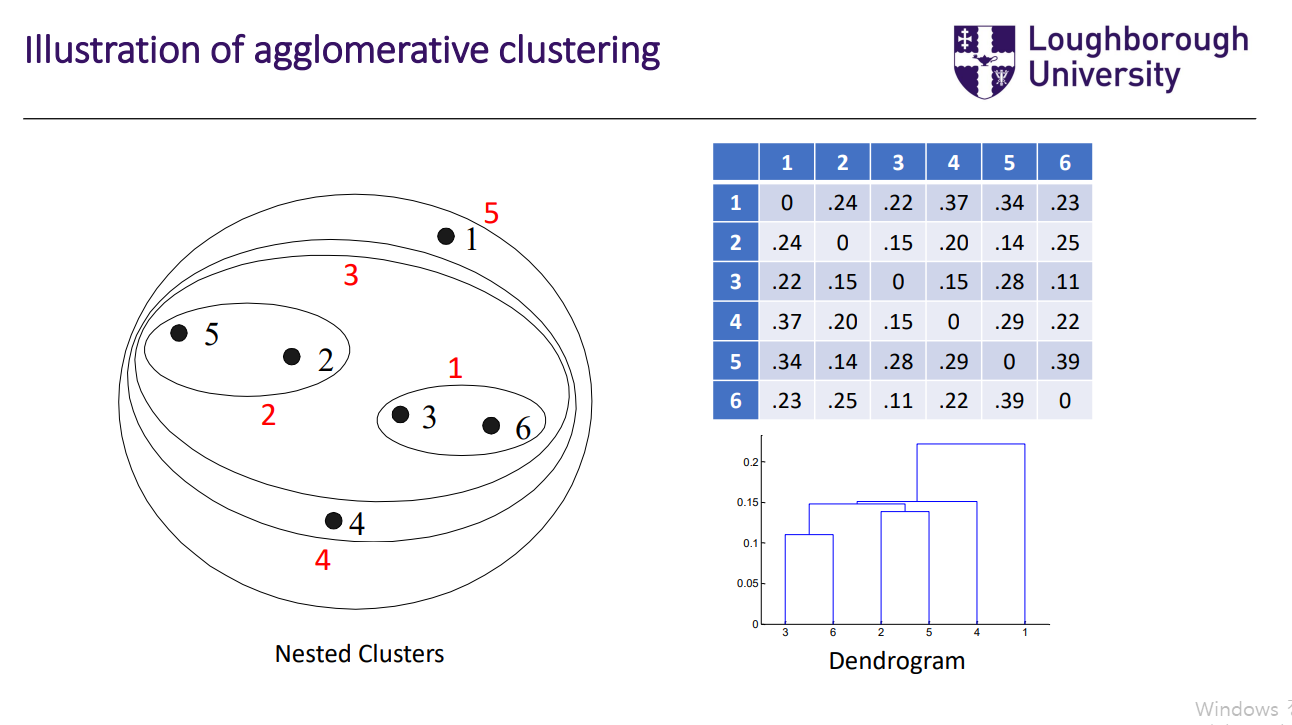


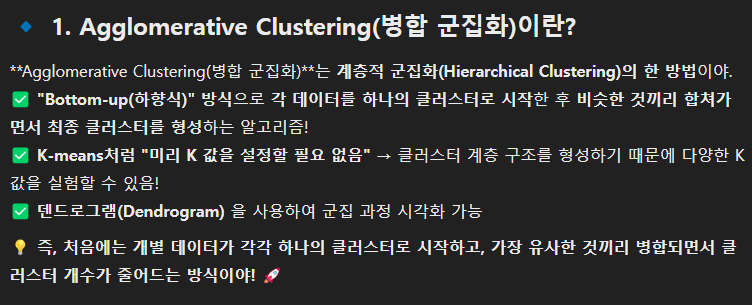


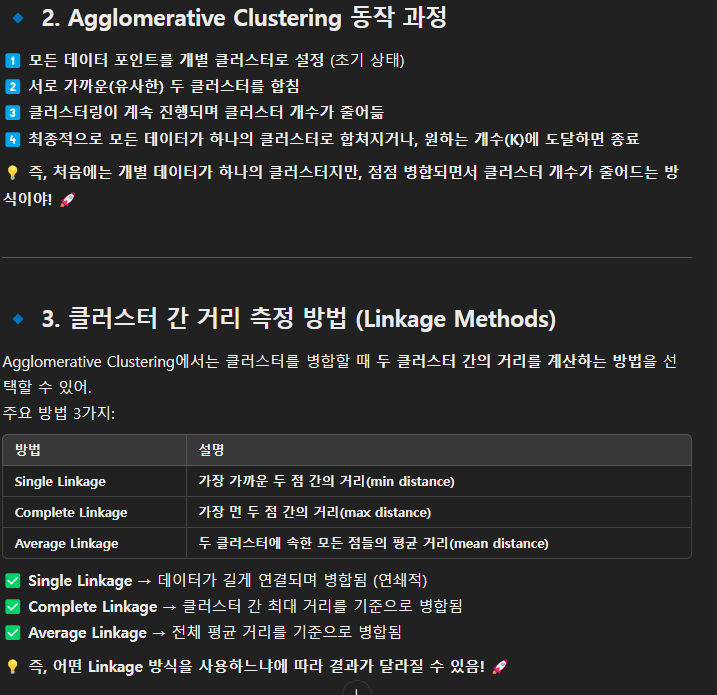
K mean algorithm (K mean 알고리즘) 코딩 예시

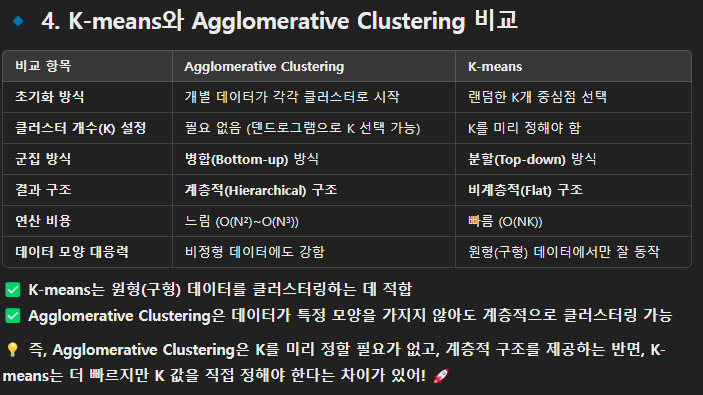


**Clustering – Agglomerative clustering algorithm**

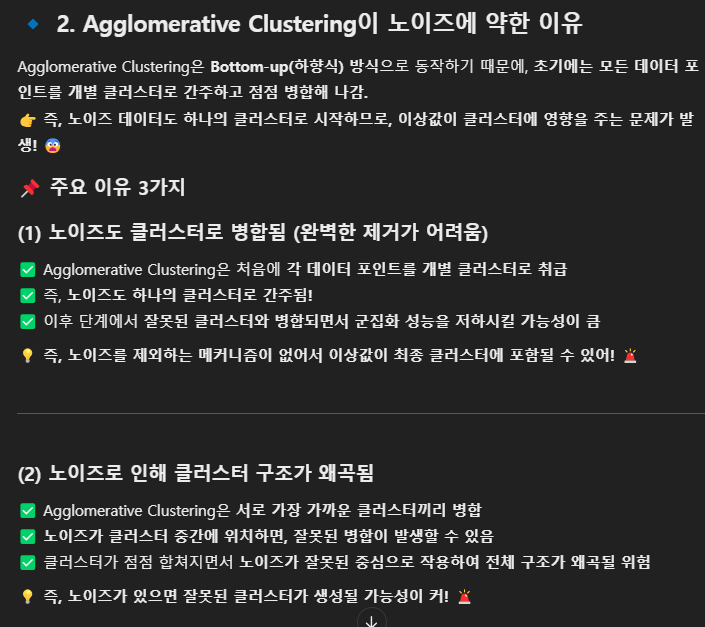
****

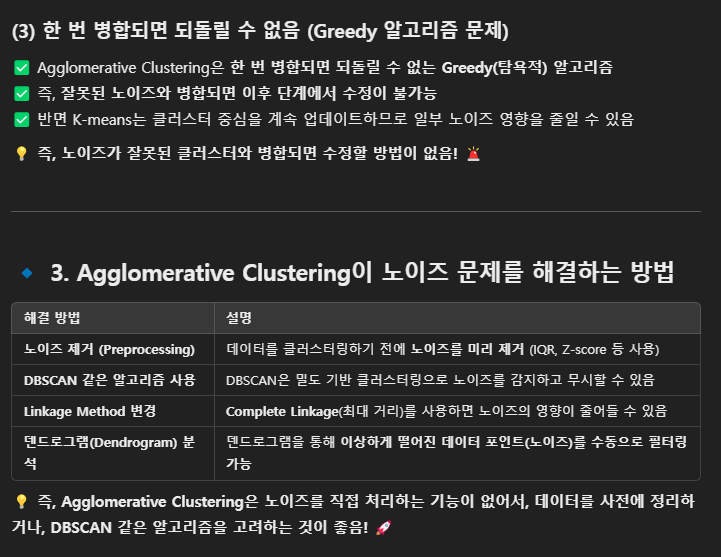






Agglomerative clustering (병합 군집화)의 약점





**Agglomerative clustering (병합 군집화)의 코딩 예시**

#Importing required libraries

from sklearn.datasets import load\_iris

from sklearn.cluster import AgglomerativeClustering

import numpy as np

import matplotlib.pyplot as plt

#Getting the data ready

data = load\_iris()

df = data.data

#Selecting certain features based on which clustering is done

df = df[:,1:3]

#Creating the model

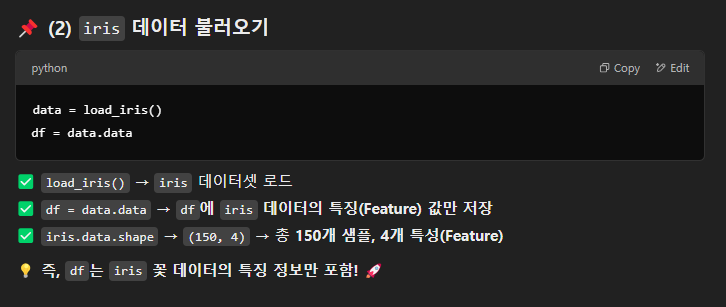
agg\_clustering = AgglomerativeClustering(n\_clusters = 3, affinity = 'euclidean', linkage =

‘average')

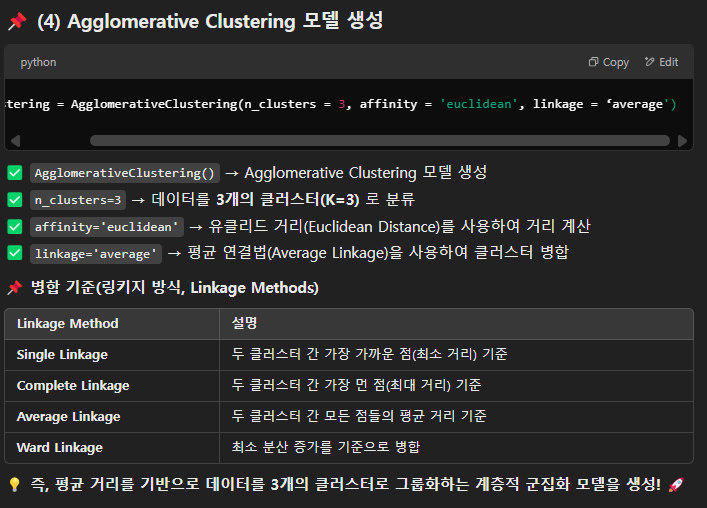
#predicting the labels

labels = agg\_clustering.fit\_predict(df)

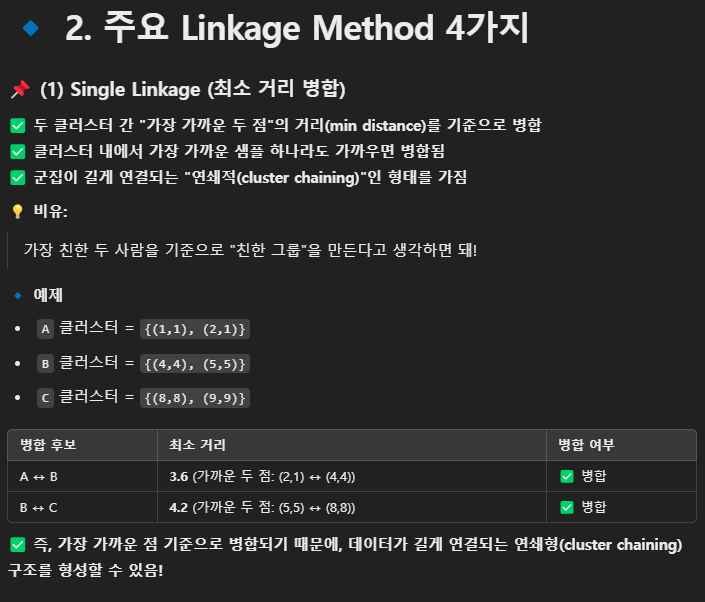
**코딩 설명**

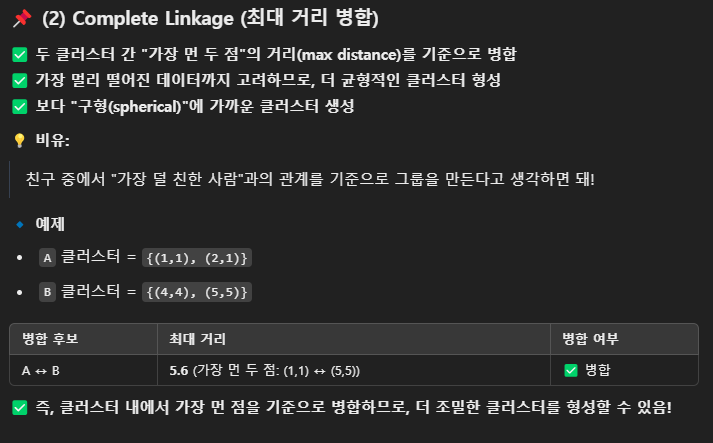


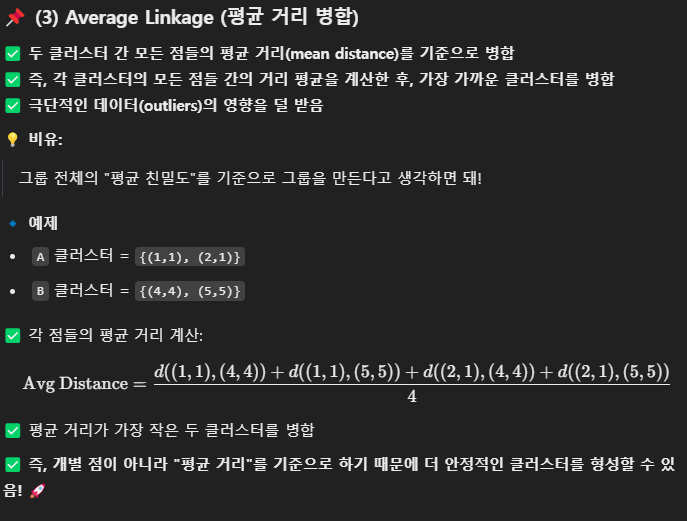


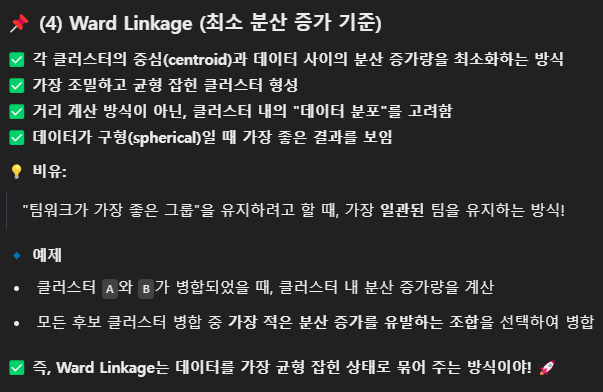


Linkage Method 설명









코드 설명 이어서 ->

