# 실습 핵심 코드 설명

1. MNIST 데이터 불러오기

from sklearn.datasets import fetch\_openml

mnist = fetch\_openml('mnist\_784', version=1, as\_frame=False) # 손글씨 숫자 데이터셋 불러옴

X, y = mnist["data"], mnist["target"].astype(np.uint8) # X: 픽셀 데이터, y: 숫자 라벨 (정수로 변환)

2. 훈련/테스트 데이터 나누기

X\_train, X\_test = X[:60000], X[60000:] # 앞 6만개 = 훈련용

y\_train, y\_test = y[:60000], y[60000:] # 뒤 1만개 = 테스트용

3. 이진 분류용 라벨 생성 (숫자 5인지 여부)

y\_train\_5 = (y\_train == 5) # True/False 형태의 이진 라벨 생성

y\_test\_5 = (y\_test == 5)

4. 모델 학습 (확률적 경사 하강법)

from sklearn.linear\_model import SGDClassifier

sgd\_clf = SGDClassifier(random\_state=42) # 모델 생성

sgd\_clf.fit(X\_train, y\_train\_5) # 모델 훈련 (5인지 아닌지 분류)

5. 예측

sgd\_clf.predict([X[0]]) # X[0]: 첫 번째 이미지 (숫자), 예측결과 True/False 반환

6. 정확도 계산 (cross\_val\_score)

from sklearn.model\_selection import cross\_val\_score

cross\_val\_score(sgd\_clf, X\_train, y\_train\_5, cv=3, scoring="accuracy") # 교차검증 정확도 측정

7. 정밀도, 재현율, F1 점수

from sklearn.model\_selection import cross\_val\_predict

from sklearn.metrics import precision\_score, recall\_score, f1\_score

# 결정 점수 얻기 (predict 대신 score 얻기)

y\_scores = cross\_val\_predict(sgd\_clf, X\_train, y\_train\_5, cv=3, method="decision\_function")

# 정밀도-재현율 커브

from sklearn.metrics import precision\_recall\_curve

precisions, recalls, thresholds = precision\_recall\_curve(y\_train\_5, y\_scores)

# 정밀도-재현율 vs 임계값 시각화

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 4))

plt.plot(thresholds, precisions[:-1], "b--", label="Precision")

plt.plot(thresholds, recalls[:-1], "g-", label="Recall")

plt.xlabel("Threshold")

plt.grid(True)

plt.legend()

plt.title("Precision and Recall vs Threshold")

plt.show()

# 특정 임계값 설정 후 재예측

y\_train\_pred\_90 = (y\_scores > 3000) # 결정점수가 3000 이상이면 5라고 판단

# 점수 출력

precision\_score(y\_train\_5, y\_train\_pred\_90)

recall\_score(y\_train\_5, y\_train\_pred\_90)

8. ROC 곡선

from sklearn.metrics import roc\_curve, roc\_auc\_score

fpr, tpr, thresholds = roc\_curve(y\_train\_5, y\_scores) # FPR, TPR 계산

# ROC 곡선 시각화

plt.figure(figsize=(6, 6))

plt.plot(fpr, tpr, linewidth=2, label="ROC curve")

plt.plot([0, 1], [0, 1], "k--") # 랜덤 추정 기준선

plt.axis([0, 1, 0, 1])

plt.xlabel("False Positive Rate")

plt.ylabel("True Positive Rate")

plt.title("ROC Curve")

plt.grid(True)

plt.legend()

plt.show()

roc\_auc\_score(y\_train\_5, y\_scores) # AUC 점수 (1에 가까울수록 좋음)

9. 다중 클래스 분류 (0~9 전체 숫자 예측)

from sklearn.linear\_model import SGDClassifier

sgd\_clf.fit(X\_train, y\_train) # y\_train 전체 숫자 사용 (0~9)

sgd\_clf.predict([X[0]]) # 결과 예: 5

10. 다중 레이블 분류

from sklearn.neighbors import KNeighborsClassifier

import numpy as np

y\_train\_large = (y\_train >= 7)

y\_train\_odd = (y\_train % 2 == 1)

y\_multilabel = np.c\_[y\_train\_large, y\_train\_odd] # 예: [True, False]

knn\_clf = KNeighborsClassifier()

knn\_clf.fit(X\_train, y\_multilabel) # 여러 개의 라벨 동시에 학습

knn\_clf.predict([X[0]]) # 예측 결과: [False, True]

11. 다중 출력 분류 (예: 잡음 제거)

noise = np.random.randint(0, 100, (len(X\_train), 784)) # 훈련 이미지에 잡음 추가

X\_train\_mod = X\_train + noise

noise\_test = np.random.randint(0, 100, (len(X\_test), 784))

X\_test\_mod = X\_test + noise\_test

knn\_clf.fit(X\_train\_mod, X\_train) # 입력: 잡음 이미지, 출력: 원본 이미지

clean\_digit = knn\_clf.predict([X\_test\_mod[0]]) # 잡음 제거 결과 예측

전체 세션 내용 요약: 분류(Classification)  
  
1. 이진 분류  
- 목표: 숫자 '5'인지 아닌지 판단  
- 모델: SGDClassifier  
- 평가: 정확도, 정밀도(precision), 재현율(recall), F1 점수  
- 시각화: Precision-Recall 곡선, ROC Curve  
  
2. 다중 클래스 분류  
- 0부터 9까지 모든 숫자 분류  
- SGDClassifier 자동으로 OvA(One-vs-All) 방식 적용  
  
3. 다중 레이블 분류  
- 예시: (7 이상인가?), (홀수인가?) → 두 조건을 모두 예측  
- 모델: KNeighborsClassifier  
  
4. 다중 출력 분류  
- 예시: 잡음 섞인 이미지에서 원본 이미지 복원  
- 입력: 잡음 이미지 / 출력: 깨끗한 이미지