3. Classification

3.1 MNIST

```
X, y = mnist["data"], mnist["target"]

X.shape
# (70000, 784) # 28*28=784개의 픽셀 (각 픽셀값은 0(흰색)~255(검은색
y.shape
# (70000,)
```

```
# reshape

some_digit = X[36000]
some_digit_image = some_digit.reshape(28, 28)
plt.imshow(some_digit_image, cmap=matplotlib.cm.binary, inter
```

```
# 데이터셋 나누기
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:6
```

```
# 훈련 세트 섞기 (모든 교차 검증 폴드가 비슷해지도록 하기 위해)
# : 데이터가 순서대로 정렬되어 있으면, 교차 검증 폴드의 각각이 비슷한 데이
#   특정 패턴에 과적합, 편향 가능
#   잘못된 모델 평가를 초래함.

import numpy as np
shuffle_index = np.random.permutation(60000)
X_train, y_train = X_train[shuffle_index], y_train[shuffle_in
```

3.2 이진 분류기 훈련

```
# 라벨을 10진수가 아니라 True or False로 바꾸기 (이진 분류)
```

```
y_train = (y_train == 5) # 5는 True고, 다른 숫자는 모두 False
y_test_5 = (y_test == 5)
```

- SGD(Stochastic Gradient Descent): 확률적 경사 하강법
  - 매우 큰 데이터셋을 효율적으로 처리
    - 한 번에 하나씩 훈련 샘플을 독립적으로 처리하기 때문
    - 그래서 “온라인 학습”에 잘 들어맞음 (↔ 일괄 처리 학습)

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(max_iter=5, random_state=42)
sgd_clf.fit(X_train, y_train_5)

sgd_clf.predict([some_digit])
# array([True, dtype=bool])
```

3.3 성능 측정

3.3.1 교차 검증을 사용한 정확도 측정

- cross_val_score()
- k-겹 교차 검증(cross validation): 훈련 세트를 K개의 폴드(fold)로 나누고, 각 폴드에 대해 예측을 만들고 평가하기 위해 나머지 폴드로 훈련시킨 모델을 사용함.

```
from sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf, X_train, y_tarin_5, cv=3, scoring="a
# array([0.9502, 0.96565, 0.96495])
```

3.3.2 오차 행렬(confusion matrix)

```
from sklearn.model_selection import cross_val_predict

# cross_val_predict: k-겹 교차 검증을 수행하지만, 평가 점수를 반환하지
```

```
#                     각 테스트 폴드에서 얻은 '예측'을 반환
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5,
```

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_train_5, y_train_pred)
#array([[54579,   0,],
#       [    0, 5421]])
#위 같은 경우는 '완벽한 분류기'인거임.
```

Actual \ Predicted	N	P
N (negative class)	TN (true negative)	FP (false positive)
P (positive class)	FN (false negative)	TP (true positive)

3.3.3 정밀도와 재현율

- precision(정밀도) : TP / (FP + TP) → 양성 예측의 정확도
  - 정밀도가 중요한 경우: 어린아이에게 안전한 동영상 걸러내기
- recall(재현율) : TP / (FN + TP) → 정확하게 감지한 양성 샘플의 비율
  - 재현율이 중요한 경우: 감시 카메라를 통해 좀도둑을 잡아내는 분류기
  - 민감도(sensitivity)
  - 진짜 양성 비율(true positive rate = TPR)
- F1 score : 정밀도와 재현율의 '조화 평균(harmonic mean)'

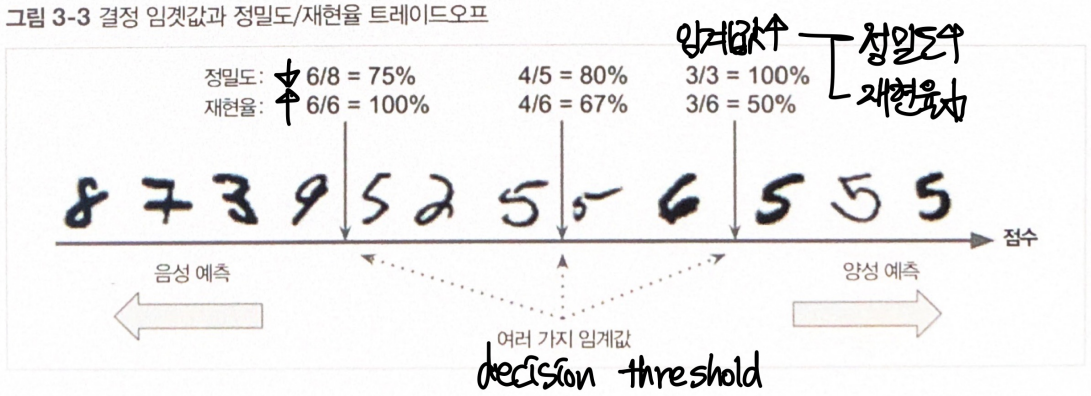
$$F1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = 2 * \frac{precision * recall}{precision + recall}$$

```
from sklearn.metrics import f1_score
f1_score(y_train_5, y_train_pred)
# 0.78
```

3.3.4 정밀도/재현율 트레이드오프

- 정밀도를 올리면 재현율 줄고, 재현율 올리면 정밀도 줄음

그림 3-3 결정 임곗값과 정밀도/재현율 트레이드오프



3.3.5 ROC 곡선

3.4 다중 분류

3.5 에러분석

3.6 다중 레이블 분류

3.7 다중 출력 분류

3.8 연습문제