CH3. 분류

📖 3.1 MNIST

MINIST dataset

- 고등학생과 미국 인구 조사국 직원들이 손으로 쓴 70,000개의 작은 숫자 이미지를 모은 데이터 셋

- 각 이미지에는 어떤 숫자를 나타내는지 레이블 되어 있음
- 사이킷런에서 제공하는 여러 헬퍼함수를 통해 데이터셋을 다운 받을 수 있음
- 사이킷런에서 제공하는 데이텃 셋들은 비슷한 딕셔너리 구조를 가지고 있음
  - 데이터셋을 설명하는 **DESCR 키**
  - 샘플이 하나의 행, 특성이 하나의 열로 구성된 배열을 가진 **data 키**
  - 레이블 배열을 담은 **target 키**

```
# 배열 살펴보기
X,y=mnist['data'],mnist['target']
print(X.shape)
print(y.shape)

(결과)
# 7만개의 이미지가 있고, 각 이미지에는 784개의 특성이 있다.
# 이미지가 28x28 픽셀이기 때문이다.
(70000, 784)
(70000,)
```

→ 개개의 특성은 단순히 0(흰색)부터 255(검은색)까지의 픽셀 강도를 나타낸다.

- mnist 데이터셋은 이미 훈련세트(앞쪽 6만개 이미지)와 테스트 세트(뒤쪽 1만개)로 나누어 놓았다. 또한 어떤 학습 알고리즘은 훈련 샘플의 순서에 민감해서 많은 비슷한 샘플이 연이어 나타나면 성능이 나빠지기도 하는데, mnist의 훈련 세트는 이미 섞여 있어서 모든 교차 검증 폴드를 비슷하게 만든다.

📖 3.2 이진 분류기 훈련

이진 분류기

- 문제를 단순화해서 하나의 숫자, 예를들면 숫자 5만 식별해보자. 이 '숫자 5 감지기'는 '5맞음'와 '5아님' 두 개의 클래스를 구분할 수 있는 이진 분류기의 한 예이다.
- 1) 분류작업을 위한 타겟 벡터 만들기

```
y_train_5=(y_train == 5)
y_test_5=(y_test == 5)
```

- 2) 분류모델을 하나 선택해 훈련하기
  - 분류 모델 : 확률적 경사 하강법 (**SGD**) 분류기
    - 사이킷런의 SGDClassifier 클래스를 사용해서 가져오기
    - SGD는 한 번에 하나 씩 훈련 샘플을 독립적으로 처리하기 때문에 (그래서 온라인 학습에 잘 들어맞는 것도 있음) 매우 큰 데이터셋을 효율적으로 처리하는 장점을 지니고 있다.
    - SGDClassifier는 훈련하는 데 무작위성을 사용한다. 결과를 재현하고 싶으면 random_state 매개변수를 지정해야 한다.

```
from sklearn.linear_model import SGDClassifier
sgd_clf=SGDClassifier(random_state=42)

# 훈련시키기
sgd_clf.fit(X_train,y_train_5)

# 이미지 감지하기
sgd_clf.predict([some_digit])
```

📖 3.3 성능측정

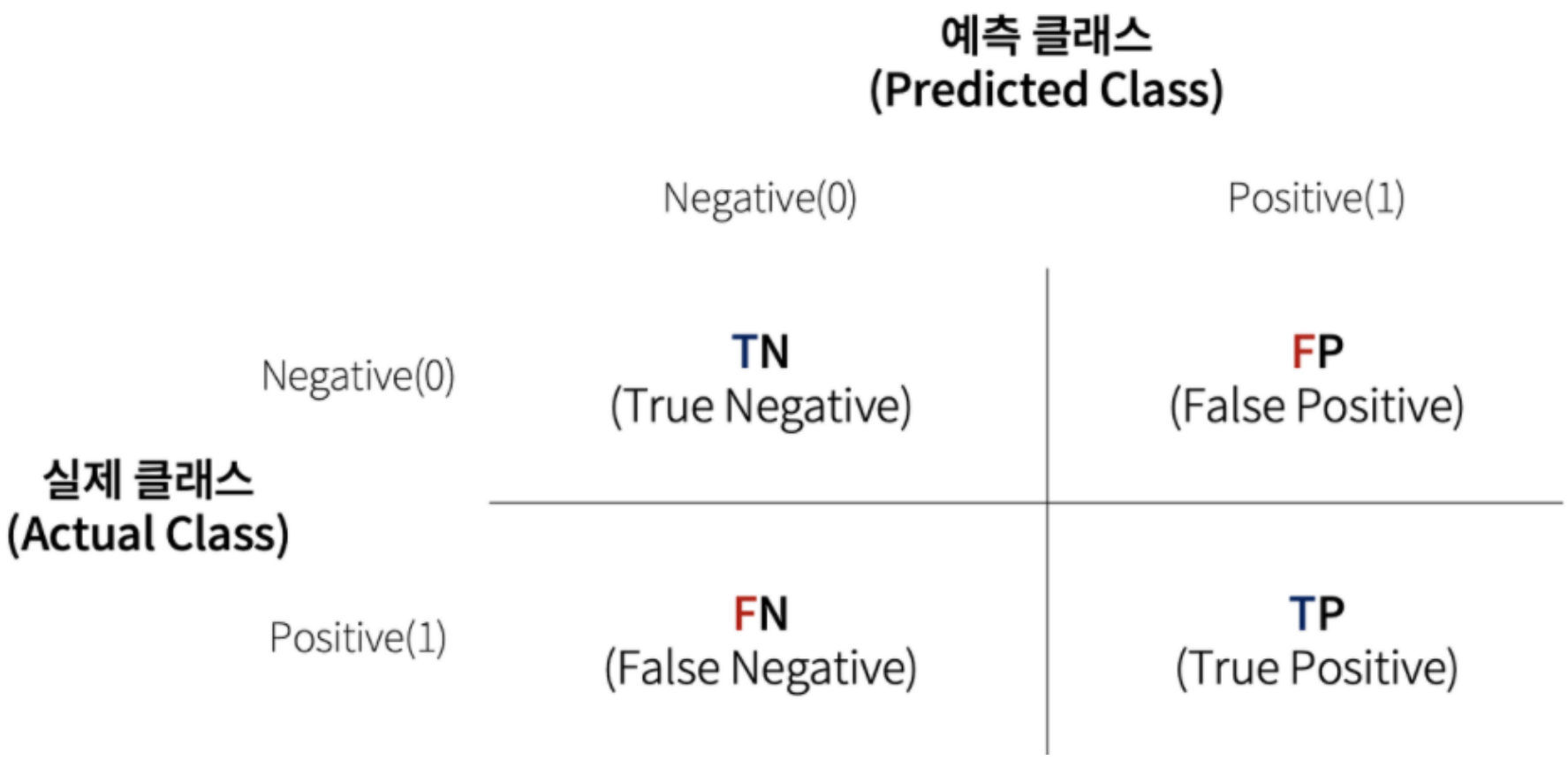
▼ 3.3.1 교차 검증을 사용한 정확도 측정

- `cross_val_score(모델,X_train,y_train,cv='fold수',scoring='accuracy')`
- 정확도는 분류기의 성능 측정 지표로 선호 되지 않는다. 특히 불균형한 데이터셋 (어떤 클래스가 다른 것보다 월등히 많은 경우) 을 다룰 때 더욱 그러하다.

▼ 3.3.2 오차행렬

오차행렬(confusion matrix)

- 아이디어 : 클래스 A의 샘플이 클래스 B로 분류된 횟수를 세는 것
- (ex) 분류기가 숫자 5의 이미지를 3으로 잘못 분류한 횟수를 알고 싶다면 오차 행렬의 5행 3열을 보면 된다.



`cross_val_predict()`

- 오차 행렬을 만들려면 실제 타깃과 비교할 수 있게, 먼저 예측값을 만들어야 한다. `cross_val_predict()` 함수를 사용해 훈련데이터로부터 예측값을 만든다.
- cross_val_score() 함수처럼 k-겹 교차검증을 수행하지만 평가점수를 반환하지 않고, 각 테스트 폴드에서 얻은 예측값을 반환한다. 즉, 훈련 세트의 모든 샘플에 대해 깨끗한 예측을 얻게 된다. (여기서 깨끗하다는 뜻은 모델이 훈련하는 동안 보지 못했던 데이터에 대해 예측했다는 뜻이다)
- `confusion_matrix()` 함수를 사용해 타깃 클래스와 예측 클래스를 넣고 오차행렬을 만들기 : `confusion_matrix(y_train_5,y_train_pred)`
- 행 : 실제 클래스 , 열 : 예측한 클래스

```
confusion_matrix(y_train_5,y_train_pred)
```

```
array([[53892,   687],
       [ 1891,  3530]], dtype=int64)
```

→ 행렬의 첫번째 행은 '5아님' 이미지(음성클래스=negative class)에 대한 것으로 53892개를 5 아님으로 정확하게 분류했다. (진짜음성=negative class)

→ 나머지 687개는 5가 아닌것을 5로 잘못 분류했다. (거짓 양성 = false positive)

→ 두번째 행은 '5맞음' 이미지 (양성클래스 = positive class)에 대한 것으로 1891개를 5아님 으로 잘못 분류했고 (거짓음성 = false negative), 나머지 3530개를 정확히 5라고 분류했다. (진짜 양성 = true positive)

- 완벽한 분류기라면 진짜 양성과 진짜 음성만 가지고 있을 것이므로 오차 행렬의 주 대각선 (왼쪽 위에서 오른쪽 아래로)만 0이 아닌 값이 된다.
- 오차행렬으로부터 얻어지는 요약된 지표
  - **정밀도**(precision) = TP/(TP+FP)
    - TP=진짜 양성, FP=거짓양성
    - 확실한 양성 샘플 하나만 예측한다면 간단히 완벽한 정밀도를 얻을 수 있으나, 이는 분류기가 다른 모든 양성 샘플을 무시하기 때문에 그리 유용하지 않다.
  - **재현율**(recall) = TP/(TP+FN)
    - 분류기가 정확하게 감지한 양성샘플의 비율
    - 민감도(sensitivity) 또는 진짜 양성비율이라고 한다.
    - FN=거짓 음성의 수

▼ 3.3.3 정밀도와 재현율

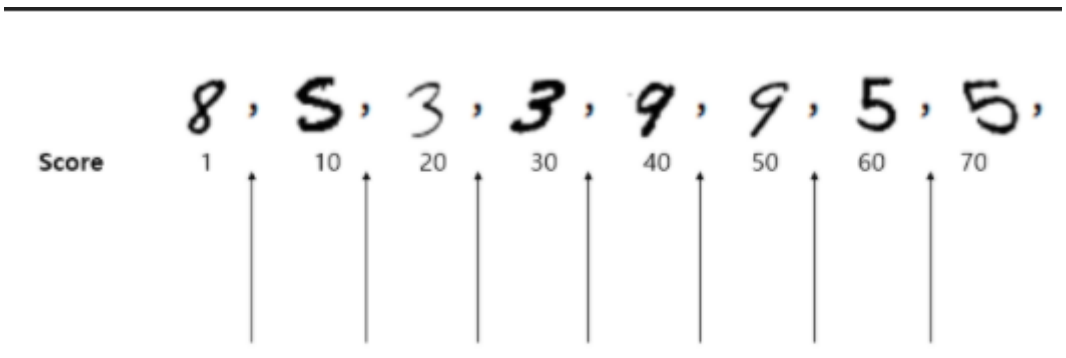
사이킷런은 정밀도와 재현율을 포함하여 분류기의 지표를 계산하는 여러 함수를 제공한다.

- 정밀도 : `precision_score(y_train_5,y_train_pred)`
- 재현율 : `recall_score(y_train_5,y_train_pred)`

F1 점수

- 정밀도와 재현율의 조화평균으로, 두 분류기를 비교하고자 만든 지표이다.
- F1= 2 / ((1/정밀도) + (1/재현율))
- `f1_score()` 함수를 호출하면 된다.
- 정밀도와 재현율이 비슷한 분류기에서는 F1 점수가 높다. 하지만 이게 항상 바람직한 것은 아니다. 상황에 따라 정밀도가 중요할 수도 있고, 재현율이 중요할 수도 있다.
- 정밀도를 올리면 재현율이 줄고 그 반대도 마찬가지이다. 이를 정밀도/재현율 트레이드오프라고 한다.

▼ 3.3.4 정밀도/재현율 트레이드오프



Threshold	5	15	25	35	45	55	65
Precision	3/7	2/6	2/5	2/4	2/3	2/2	2/2
Recall	3/3	2/3	2/3	2/3	2/3	2/3	1/3

Threshold : 결정 임계값

분류기는 결정함수를 사용해 각 샘플의 점수(Score)를 계산하고 이 점수가 임계값보다 크면 샘플을 양성 클래스에 할당, 그렇지 않으면 음성 클래스에 할당한다. 진짜 양성, 거짓양성, 진짜 음성, 거짓 음성의 개수를 세고, 그에 따른 정밀도와 재현율을 계산하게 되면, 임계값의 기준에 따라 한 값은 높아지고 한 값은 낮아진다.

- 사이킷런의 각 샘플의 점수를 얻을 수 있는 메서드 : `decision_function()`

→ 이 점수를 기반으로 원하는 임계값을 정해 예측을 만들 수 있다.

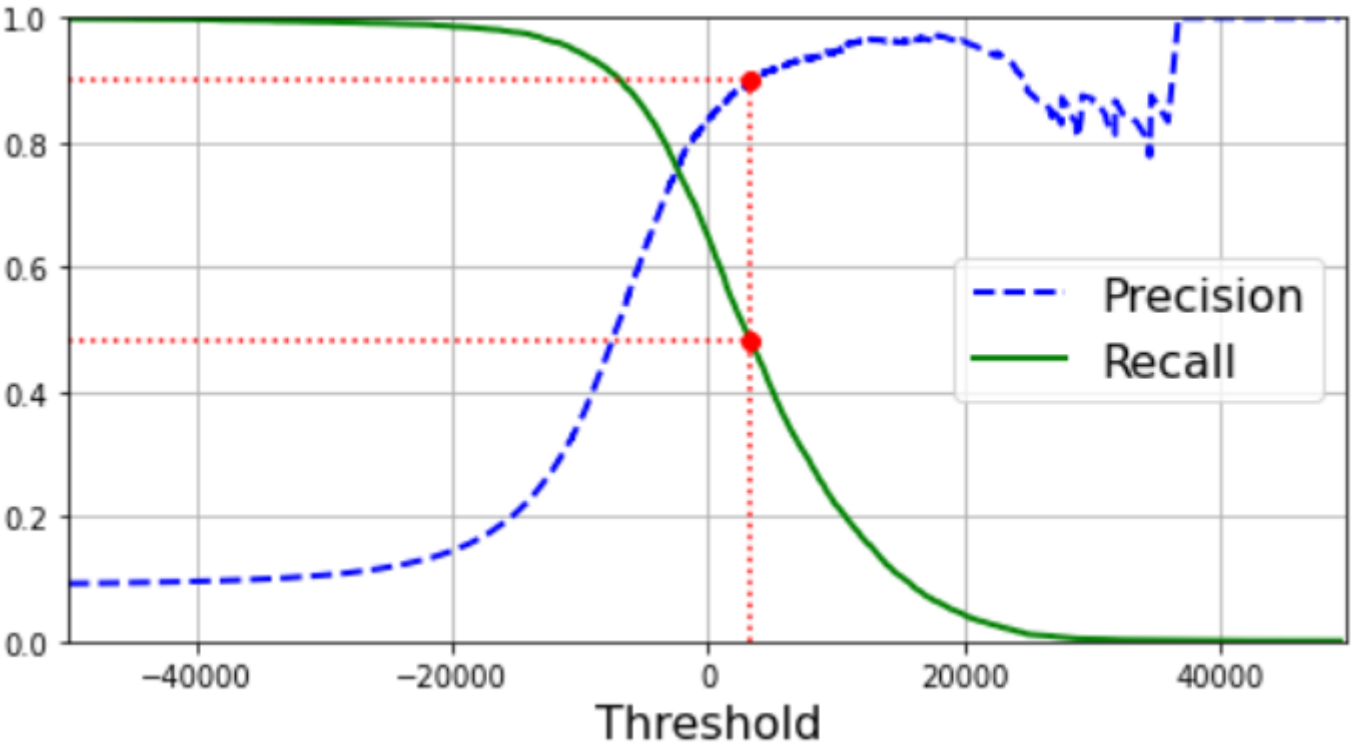
- 적절한 임계값 정하기

1) `cross_val_predict()` 함수를 사용해 훈련 세트에 있는 모든 샘플의 점수를 구하기

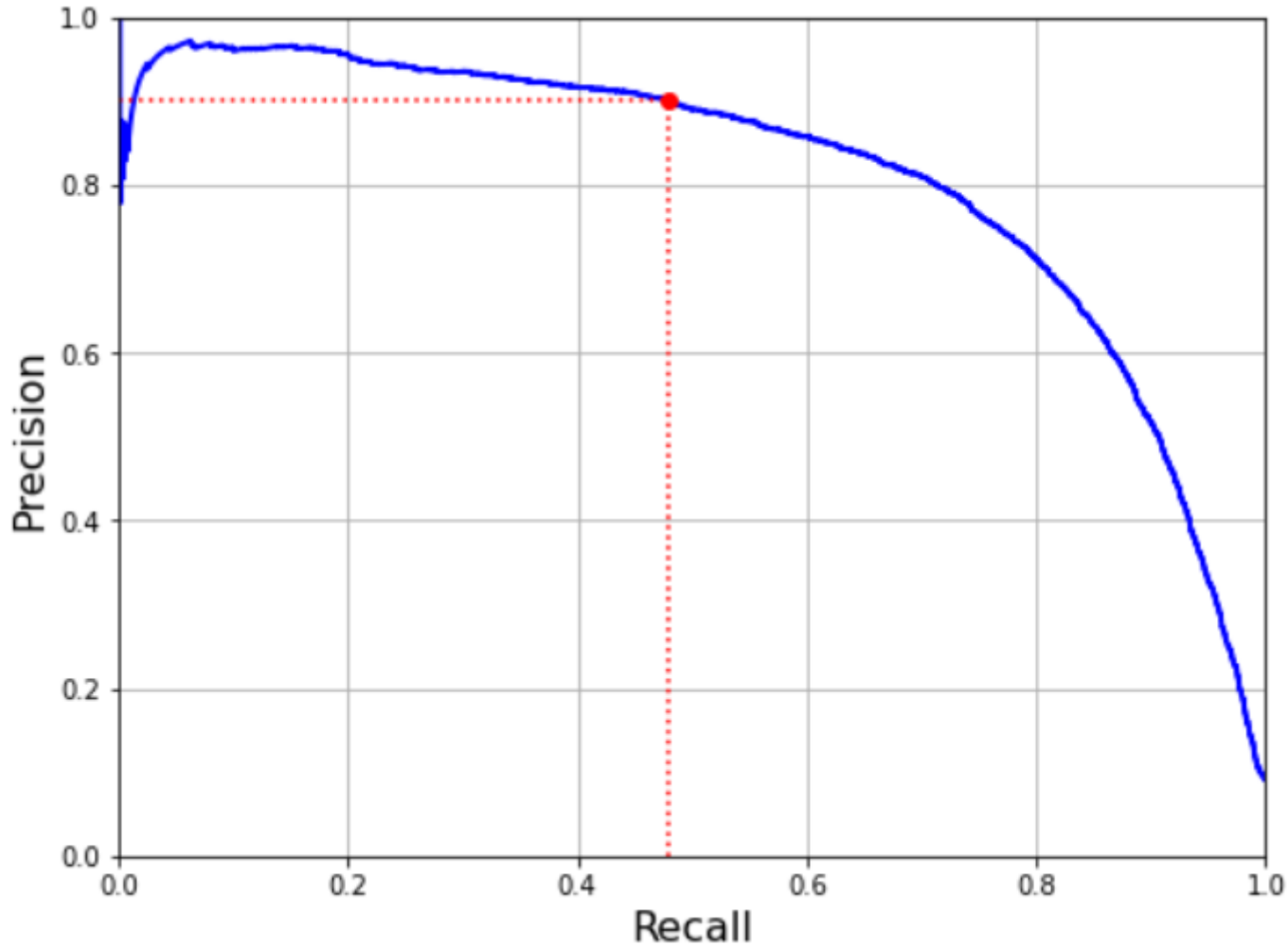
2) `precision_recall_curve()` 함수를 사용해 가능한 모든 임곗값에 대해 정밀도와 재현율 계산하기

3) 정밀도와 재현율을 그려 임곗값 찾기

방법1 : 임곗값에 대한 그래프 그리기



방법2 : 재현율에 대한 정밀도 곡선을 그리기

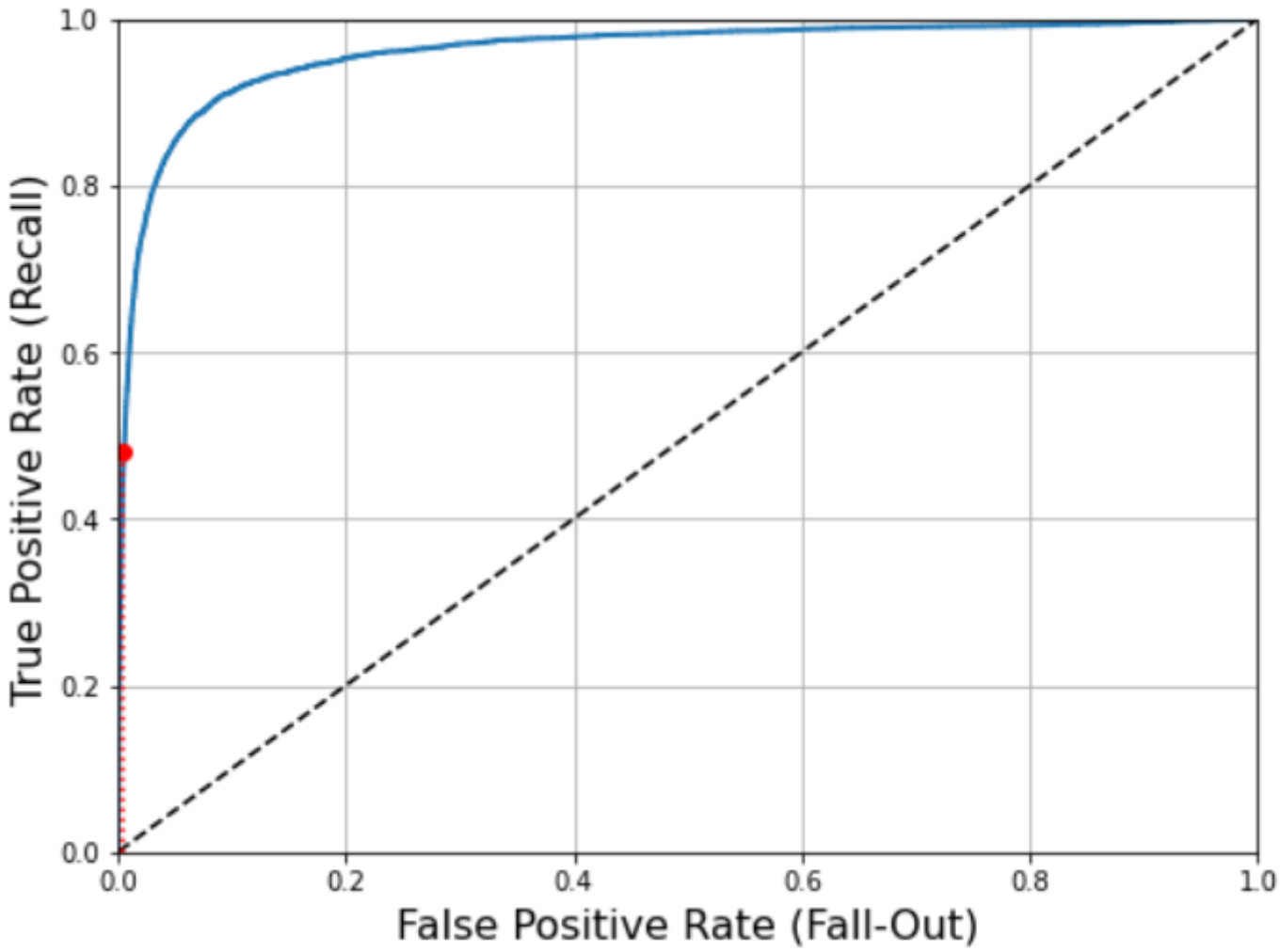


→ 재현율 80% 근처에서 정밀도가 급격히 줄어듦. 이 하강점 직전을 정밀도/재현율 트레이드 오프로 선택하는 것이 좋다. 예를들어 재현율이 60% 정도인 지점이다. 물론 이런 선택은 프로젝트에 따라 달라진다.

▼ 3.3.5 ROC 곡선

ROC 곡선 (수신기 조작 특성 곡선)

- 이진분류에서 널리 사용하는 도구로, 거짓양성비율 (FPR) 에 대한 진짜 양성 비율 (TPR= 민감도 = 재현율) 의 곡선이다.
- FPR : 양성으로 잘못 분류된 음성 샘플의 비율 = 1 - (진짜 음성 비율 : 음성으로 정확하게 분류한 음성샘플의 비율 = TNR, 특이도라고도 한다)
- **재현율**에 대한 **1-특이도** 그래프
- ROC 곡선 그리기
  - 1) `roc_curve()` 함수를 사용해 여러 임곗값에서 TPR과 FPR을 계산해야 한다.
  - 2) 맷플롯립을 사용해 TPR에 대한 FPR 곡선 나타내기



→ 재현율(TPR)이 높을수록 분류기가 만드는 거짓양성이(FPR) 늘어난다.

→ 점선 : 완전한 랜덤분류기의 ROC 곡선

→ 좋은 분류기는 이 점선에서 최대한 멀리 떨어져 있어야 한다. (왼쪽 위 모서리)

- 곡선 아래의 면적 (AUC)를 측정하면 분류기들을 비교할 수 있다. 완벽한 분류기는 AUC가 1이고 완전한 랜덤분류기 (훈련데이터의 클래스 비율을 무작위로 예측하는 것) 는 0.5이다.
  - AUC를 구하는 함수를 사이킷런에서 제공 : `roc_auc_score(y_train_5,y_score)`
- 일반적으로 양성 클래스가 드물거나 거짓 음성보다 거짓 양성이 더 중요할 때, PR 곡선을 (정밀도/재현율 곡선) 사용하고 그렇지 않으면 ROC 곡선을 사용한다.

📖 3.4 다중분류

이진 분류기는 두 개의 클래스를 구별하는 반면 다중분류기는 둘 이상의 클래스를 구별할 수 있다.

SGD 분류기, 랜덤 포레스트 분류기, 나이브 베이즈 분류기 같은 일부 알고리즘은 여러 개의 클래스를 직접 처리할 수 있는 반면, 로지스틱 회귀나 서포트 벡터 머신 분류기 같은 다른 알고리즘은 이진 분류만 가능하다. 하지만 이진 분류기를 여러 개 사용해 다중 클래스를 분류하는 기법도 많다. 예를 들어 특정 숫자 하나만 구분하는 숫자별 이진 분류기 10개를 훈련시켜 클래스가 10개인 숫자 이미지 분류 시스템을 만들 수 있다. 이미지를 분류할 때 각 분류기의 결정 점수 중에서 가장 높은 것을 클래스로 선택하면 된다. 이를 OvR 전략이라 한다.

또 다른 전략은 0과 1구별, 0과 2구별, 1과 2구별 등과 같이 각 숫자의 조합마다 이진 분류기를 훈련 시키는 것이다. 이를 OvO 전략이라고 한다. 클래스가 N개라면 분류기는 N(N-1)/2 개가 필요하다. 즉, MNIST 문제에서는 45개 분류기를 훈련시켜야 하는 것! 이미지 하나를 분류하려면 45개 분류기를 모두 통과시켜서 가장 많이 양성으로 분류된 클래스를 선택한다. OvO 전략의 주요 장점은 각 분류기의 훈련에 전체 훈련세트 중 구별할 두 클래스에 해당하는 샘플만 필요하다는 것이다.

다중 클래스 분류작업에 이진 분류 알고리즘을 선택하면 사이킷런이 알고리즘에 따라 자동으로 OvR 또는 OvO를 실행한다. 사이킷런에서 둘 중 하나를 강제로 사용하도록 한다면 **OneVsRestClassifier** 나 **OneVsOneClassifier** 를 사용한다.

→ 자세한 과정은 주피터 노트북 참고!

📖 3.5 에러분석

가능성이 높은 모델을 하나 찾았다고 가정하고 이 모델의 성능을 향상 시킬 방법으로 에러의 종류를 분석해보자.

먼저 오차 행렬을 살펴볼 수 있다. 오차행렬을 이미지로 표현해 보았을 때, 이미지가 올바르게 분류되었으면 주 대각선이 밝게 보인다. 숫자가 어두우면, 데이터셋에 이미지가 적거나 분류기가 다른 것만큼 잘 분류하지 못한다는 뜻이다.

행 = 실제 클래스 , 열 = 예측한 클래스

- 5의 이미지를 3으로 잘못 분류한 횟수를 알고 싶다면 오차행렬의 5행 3열을 보면 된다.
- 오차행렬을 분석하면 분류기의 성능향상 방안에 대한 통찰을 얻을 수 있다. 훈련 데이터를 더 많이 모아 분류기를 학습시키거나, 분류기에 도움이 될만한 특성을 더 찾아볼 수 있다. 개개의 에러를 분석해보면 분류기가 무슨일을 하고 왜 잘못되었나 통찰을 얻을 수 있지만 더 어렵고 시간이 오래 걸릴 것이다.
- 분류기는 이미지의 위치나 회전 방향에 매우 민감하다.

→ 자세한 과정은 주피터 노트북 참고!

📖 3.6 다중 레이블 분류

여러 개의 이진 꼬리표를 출력하는 분류 시스템을 **다중 레이블 분류 시스템**이라고 한다. ex) 얼굴 인식 시스템 : 같은 사진에 여러 사람이 등장하면 인식된 사람마다 하나씩 꼬리표를 붙여야 한다. 즉 분류기가 샘플마다 여러 개의 클래스를 출력해야 하는 경우

다중레이블 분류기를 평가하는 방법은 많다. 적절한 지표는 프로젝트에 따라 다르다.

→ 자세한 과정은 주피터 노트북 참고!

📖 다중 출력 분류

다중 출력 분류란 다중 레이블 분류에서 한 레이블이 다중 클래스가 될 수 있도록 일반화 한 것이다.

예를 들어 이미지에서 잡음을 제거하는 시스템은 다중 출력 분류의 예시가 될 수 있다. 잡음이 많은 숫자 이미지를 입력으로 받고, 깨끗한 숫자 이미지를 픽셀의 강도를 담은 배열로 출력한다고 하면, 분류기의 출력이 다중 레이블 (픽셀당 한 레이블) 이고 각 레이블 값은 여러개를 가진다. (0부터 255까지의 픽셀강도)