

3장 분류

주요 내용

- MNIST 데이터셋
- 이진 분류기 훈련
- 분류기 성능 측정
- 다중 클래스 분류
- 오류 분석
- 다중 라벨 분류

3.1. MNIST 데이터셋

- 미국의 고등학생과 인구조사국 직원들이 손으로 쓴 70,000개의 숫자 이미지로 구성된 데이터셋
- 사용된 0부터 9까지의 숫자는 각각 $28 \times 28 = 784$ 크기의 1차원 어레이로 이미지 데이터.
- 라벨: 총 70,000개의 사진 샘플이 표현하는 값

sklearn.datasets 모듈

`sklearn.datasets` 모듈은 데이터셋을 다운로드하거나 생성하는 세 종류를 함수를 제공하며 함수명에 사용된 접두사에 따라 용도가 다름.

- `fetch_*`: 다운로드 및 적재. `sklearn.utils.Bunch` 객체 반환.
- `load_*`: 미니 데이터셋 적재. 다운로드 없음.
- `make_*`: 데이터셋 임의 생성. 입력 데이터셋과 타깃 데이터셋으로 구분된 `(X, y)` 모양의 넘파이 어레이 생성.

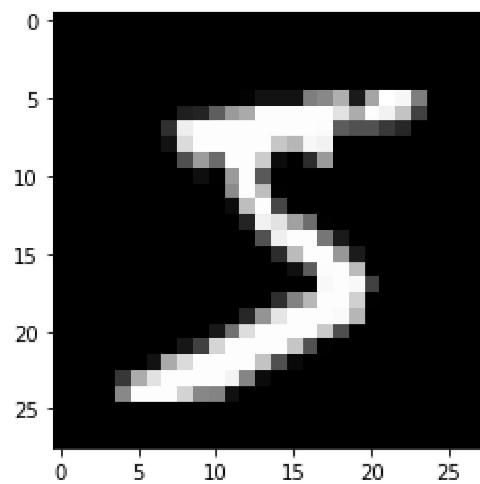
```
from sklearn.datasets import fetch_openml  
  
mnist = fetch_openml('mnist_784', as_frame=False)
```

입력 데이터셋과 타깃 데이터셋

- 입력 데이터: 0부터 9까지의 숫자는 모두 28x28 크기의 1차원 어레이로 제공됨
- 라벨: 정수가 아니라 '0', '1', ..., '9' 처럼 문자열로 지정됨.

```
X, y = mnist.data, mnist.target
```

첫째 손글씨 이미지



첫 100 개 손글씨 이미지

5 0 4 1 9 2 1 3 1 4
3 5 3 6 1 7 2 8 6 9
4 0 9 1 1 2 4 3 2 7
3 8 6 9 0 5 6 0 7 6
1 8 7 9 3 9 8 5 9 3
3 0 7 4 9 8 0 9 4 1
4 4 6 0 4 5 6 1 0 0
1 7 1 6 3 0 2 1 1 7
8 0 2 6 7 8 3 9 0 4
6 7 4 6 8 0 7 8 3 1

문제 정의

- 지도학습: 각 이미지가 담고 있는 숫자가 라벨로 지정됨.
- 모델: 이미지 데이터를 분석하여 0부터 9까지, 총 10개의 범주로 분류하는 다중 클래스 분류
- 실시간 훈련 여부: 배치 또는 온라인 학습 둘 다 가능하지만 여기서는 확률적 경사하강법과 랜덤 포레스트 분류기를 이용하여 배치 학습 실행

훈련셋과 테스트셋

- MNIST 데이터셋 이미 6:1 분류되어 있음
- 훈련 세트: 앞쪽 60,000개 이미지
- 테스트 세트: 나머지 10,000개의 이미지

```
X_train, X_test, y_train, y_test = X[:60000], X[60000:], y[:60000], y[60000:]
```

3.2. 이진 분류기: 숫자-5 감별기

- 이미지 샘플이 숫자 5를 표현하는지 여부를 판정하는 이진 분류기
- 모든 라벨을 0 또는 1로 수정해야 함
 - 0: 숫자 5 이외의 수를 가리키는 이미지 라벨
 - 1: 숫자 5를 가리키는 이미지 라벨

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

SGD 분류기 활용

- 확률적 경사 하강법 분류기
- 한 번에 하나씩 훈련 샘플 처리 후 파라미터 조정
- 매우 큰 데이터셋을 효율적으로 훈련시킬 수 있으며 온라인 학습에도 적합함
- 훈련: `fit()` 메서드 호출

```
from sklearn.linear_model import SGDClassifier  
  
sgd_clf = SGDClassifier(max_iter=1000, tol=1e-3, random_state=42)  
sgd_clf.fit(X_train, y_train_5)
```

이미지 픽셀과 데이터 입력 특성

- 입력 데이터셋의 모든 샘플은 길이가 784인 1차원 어레이로 주어짐
- 어레이의 각 항목은 `28x28` 모양의 손글씨 이미지에 포함된 하나의 픽셀값에 해당함
- `sgc_clf` 모델은 784 개의 픽셀 정보를 이용하여 해당 이미지 샘플이 가리키는 숫자가 5인지 여부를 판정하도록 훈련됨.

3.3. 분류기 성능 측정

분류기의 성능 측정 기준으로 보통 다음 세 가지를 사용한다.

- 정확도
- 정밀도/재현율
- ROC 곡선의 AUC

3.3.1. 오차행렬

- 오차 행렬: 클래스별 예측 결과의 참/거짓을 정리한 행렬
- 숫자-5 감지기에 대한 오차 행렬

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import confusion_matrix

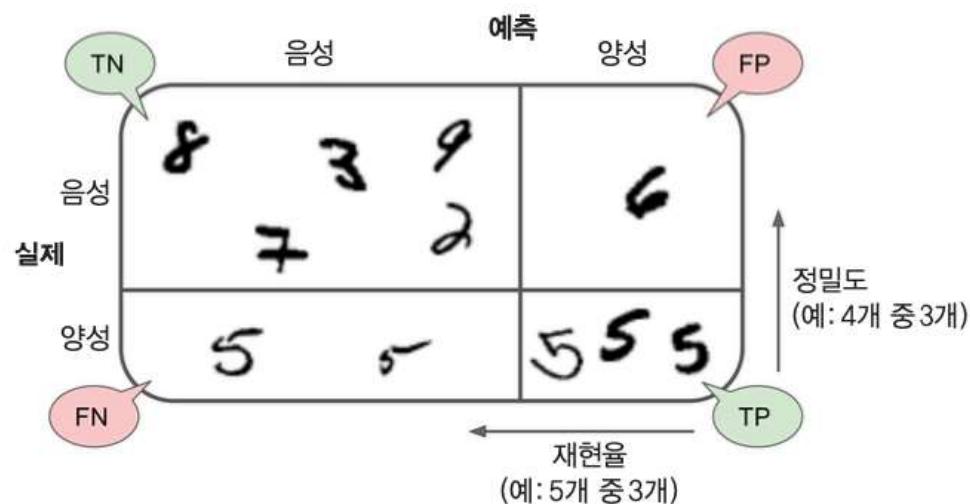
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)
confusion_matrix(y_train_5, y_train_pred)
```

```
array([[53892,    687],
       [ 1891,  3530]])
```

오차 행렬 예제

아래 그림은 숫자-5 감별기의 오차 행렬을 단순화하여 보여준다.

```
array([[5, 1],  
       [2, 3]])
```



3.3.2. 정확도 accuracy

- 라벨을 정확하기 맞힌 비율

$$\text{정확도} = \frac{TP + TN}{TP + FP + TN + FN} = \frac{3530 + 53892}{3530 + 687 + 53892 + 1891} = 0.957$$

정확도의 한계

- 정확도가 96% 정도로 매우 좋은 결과로 보임.
- "무조건 5가 아니다" 라고 예측하는 모델도 90%의 정확도를 보임.
- 특정 범주에 속하는 데이터가 상대적으로 너무 많을 경우 정확도는 신뢰하기 어려운 평가 기준임
- 이런 경우엔 정확도 보다는 정밀도와 재현율을 이용하여 평가함.

3.3.3. 정밀도와 재현율

정밀도 precision

- 양성 예측의 정확도
- 예제: 숫자 5라고 예측된 값들 중에서 진짜로 5인 숫자들의 비율

$$\text{정밀도} = \frac{TP}{TP + FP} = \frac{3530}{3530 + 687} = 0.837$$

재현율

recall

- 양성 샘플에 대한 정확도, 즉, 분류기가 정확하게 감지한 양성 샘플의 비율
- 재현율을 **민감도**(sensitivity) 또는 **참 양성 비율**(true positive rate)로도 부름

$$\text{재현율} = \frac{TP}{TP + FN} = \frac{3530}{3530 + 1891} = 0.651$$

정밀도와 재현율의 상대적 중요도

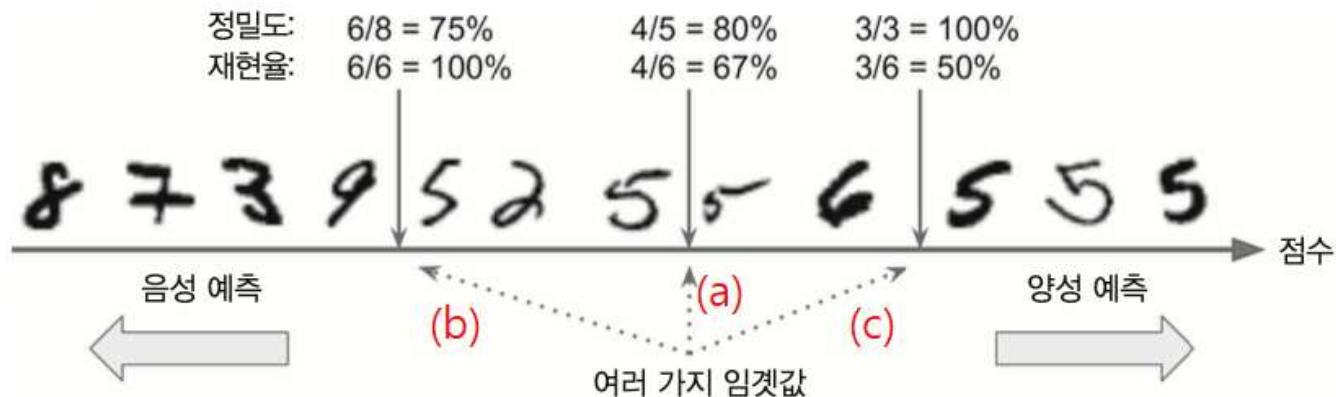
- 목적에 따라 정밀도와 재현율의 중요도가 다름
- 재현율이 보다 중요한 경우: 암 진단 기준
 - 정밀도: 암이라고 진단했는데 진짜 암인 경우의 비율
 - 재현율: 암이 실제로 있는데 암이라고 진단한 경우의 비율
- 정밀도가 보다 중요한 경우: 아동용 동영상 선택 기준
 - 정밀도: 아동용으로 판단된 동영상 중에서 실제로 아동용인 동영상의 비율
 - 재현율: 아동용 동영상 중에서 아동용 동영상이라고 판단된 동영상의 비율

3.3.4. 정밀도/재현율 트레이드오프

- 정밀도와 재현율은 상호 반비례 관계임.
- 정밀도와 재현율 사이의 적절한 비율을 유지하는 분류기를 찾아야 함.
- 적절한 결정 임곗값을 지정해야 함.

결정 함수와 결정 임곗값

- **결정 함수** decision function: 각 훈련 샘플에 대한 점수를 계산하는 함수
- **결정 임계값** decision threshold: 결정 함수가 양성 클래스 또는 음성 클래스로 분류하는 데에 사용하는 기준값
- 결정 임곗값이 클 수록 정밀도는 올라가지만 재현율은 떨어짐.



SGD 분류기의 결정 함수

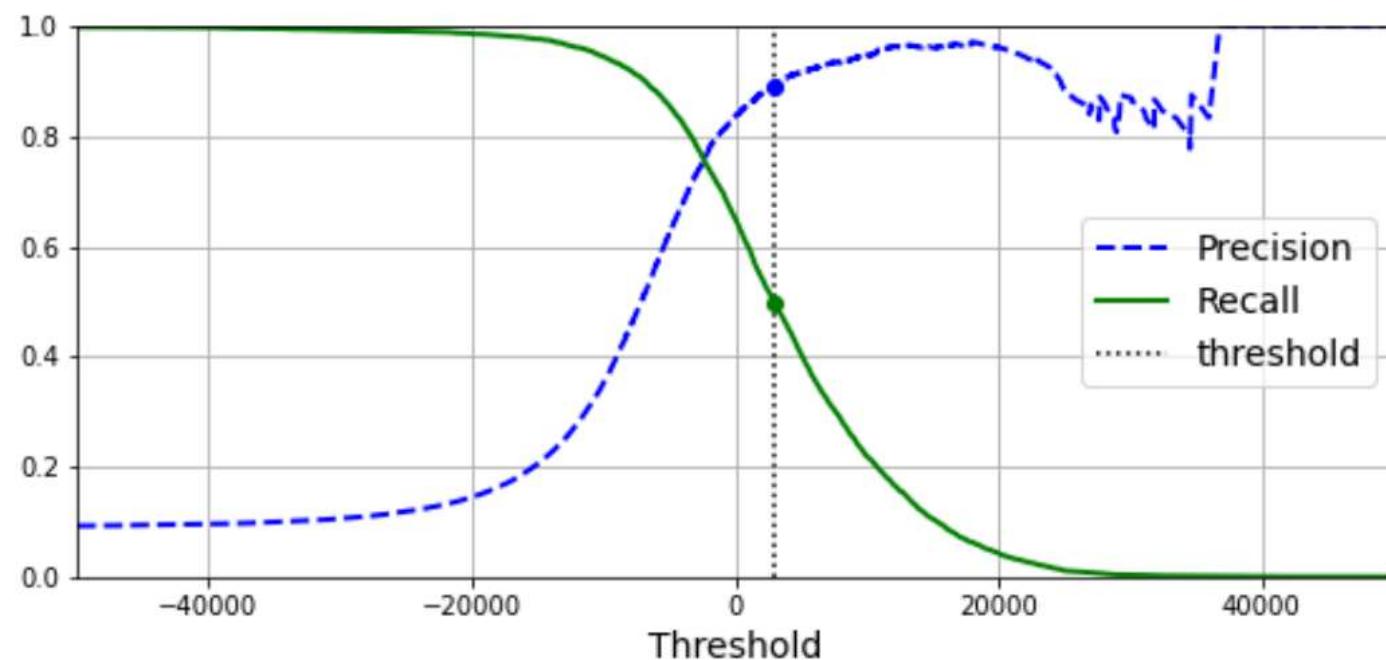
- 결정 함수를 이용해서 교차검증을 실행하면 각 샘플에 대한 결정 함수의 값으로 구성된 어레이가 생성됨.
- 앞서 오차 행렬 생성에 사용된 `cross_val_predict()` 함수를
`method="decision_function"` 키워드 인자와 함께 호출

```
y_scores = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3,  
                             method="decision_function")
```

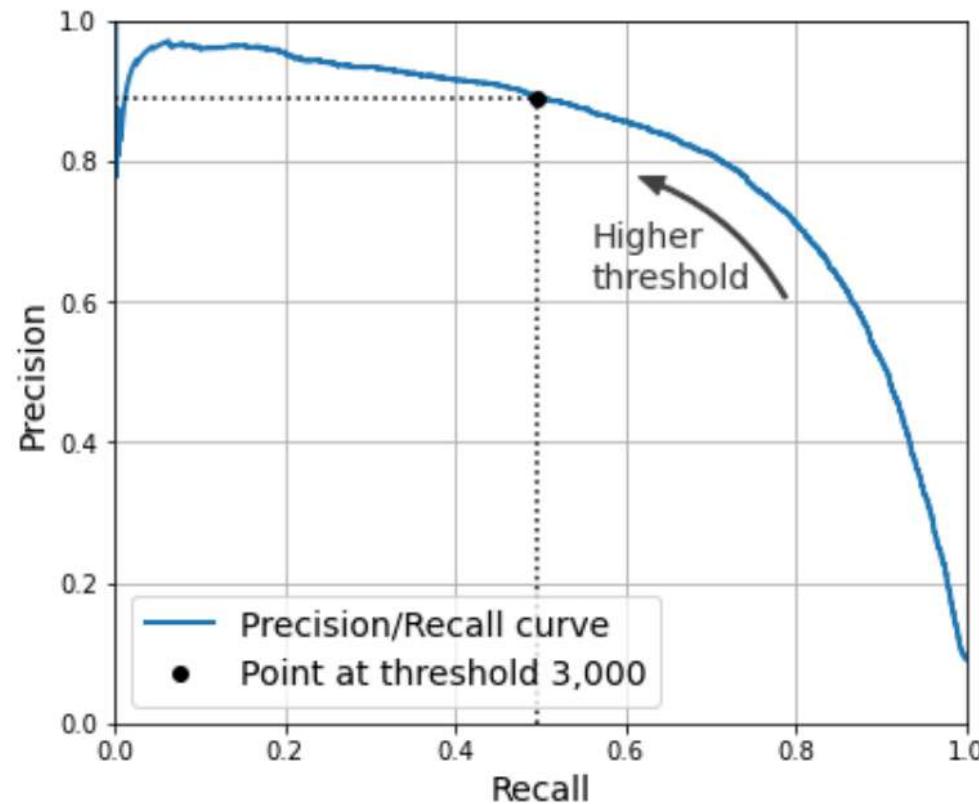
```
y_scores[:10]
```

```
array([-1200.93051237, -26883.79202424, -33072.03475406, -15919.5480689 ,  
       -20003.53970191, -16652.87731528, -14276.86944263, -23328.13728948,  
       -5172.79611432, -13873.5025381 ])
```

결정 임곗값/정밀도/재현율 그래프



정밀도/재현율 그래프

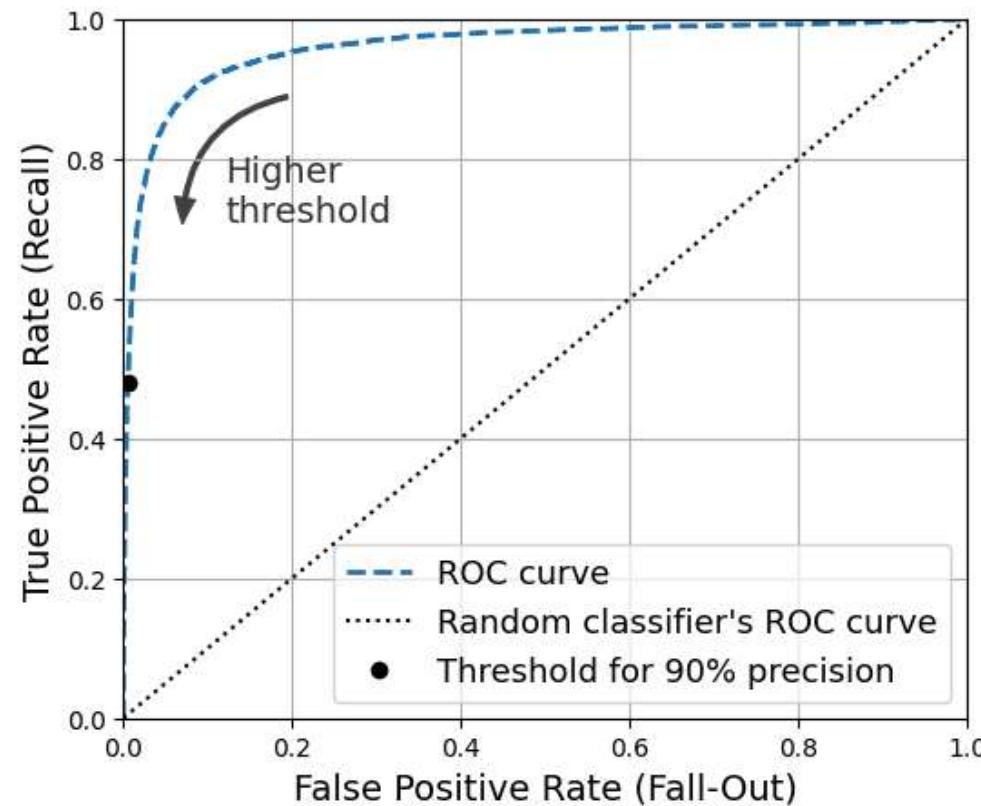


3.3.5. ROC 곡선의 AUC

- **수신기 조작 특성**(receiver operating characteristic, ROC) 곡선을 활용하여 이진 분류기의 성능 측정 가능
- **ROC 곡선: 거짓 양성 비율**false positive rate(FPR)에 대한 **참 양성 비율**true positive rate(TPR)의 관계를 나타내는 곡선
- 참 양성 비율: 재현율
- 거짓 양성 비율: 원래 음성인 샘플 중에서 양성이라고 잘못 분류된 샘플들의 비율. 예를 들어, 5가 아닌 숫자중에서 5로 잘못 예측된 숫자의 비율

$$\text{FPR} = \frac{FP}{FP + TN}$$

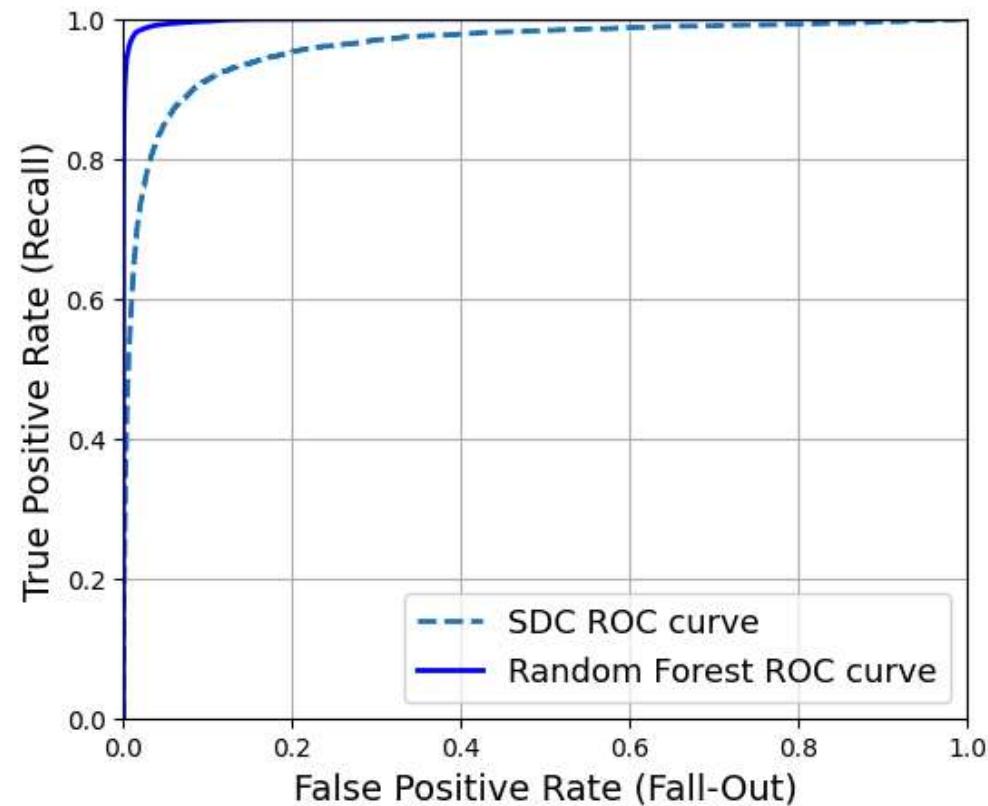
참 양성 비율(TPR) vs. 거짓 양성 비율(FPR)



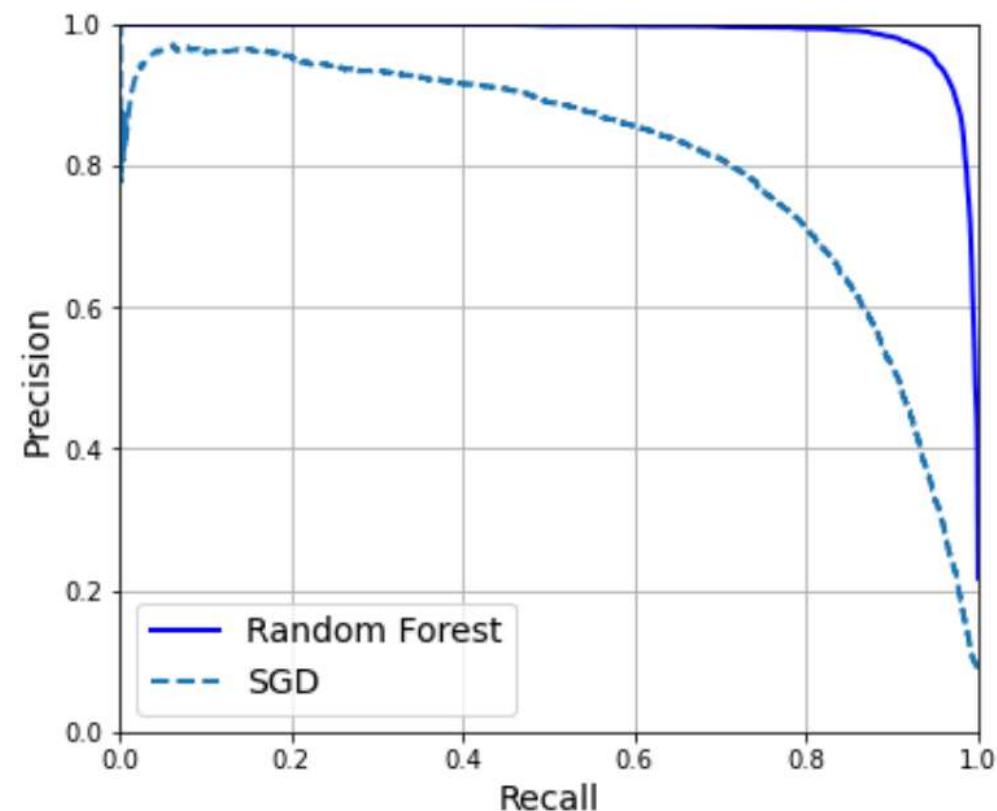
AUC와 분류기 성능

- 재현율(TPR)과 거짓 양성 비율(FPR) 사이에도 서로 상쇄하는 기능이 있다는 것을 확인 가능
- 즉, 재현율(TPR)을 높이고자 하면 거짓 양성 비율(FPR)도 함께 증가
- 좋은 분류기는 재현율은 높으면서 거짓 양성 비율은 최대한 낮게 유지해야함
- ROC 곡선이 y축에 최대한 근접하는 결과가 나오도록 해야함.
- **AUC**(ROC 곡선 아래의 면적)가 1에 가까울 수록 성능이 좋은 분류기로 평가됨.

SGD와 랜덤 포레스트의 AUC 비교



SGD와 랜덤 포레스트의 정밀도/재현율 비교



3.4. 다중 클래스 분류

- **다중 클래스 분류** multiclass classification: 세 개 이상의 범주(클래스)로 데이터 분류.
- MNIST 손글씨 숫자 분류기: 손글씨 이미지 데이터가 주어지면 0부터 9까지 10개의 범주로 분류하는 다중 클래스 모델

다중 클래스 분류 지원 모델

- LogisticRegression 모델
- RandomForestClassifier 모델
- SGDClassifier 모델
- SVC 모델

LogisticRegression 모델

- 딥러닝 기법을 활용한 분류 모델에서도 많이 활용됨.
- 이진 분류: 로지스틱 회귀 활용
- 소프트맥스 회귀: 다중 클래스 분류
- [4장 모델 훈련](#)에서 자세히 다룸.

다중 클래스 분류 모델 교차 검증

- 정확도 기준 교차 검증: 0부터 9까지 각 숫자에 해당하는 데이터가 고르게 분포되어 있어서 정확도가 모델 성능 평가의 기준으로 적합
- 아래 코드를 이용하여 SGDClassifier` 모델에 대해 교차 검증으로 정확도를 계산하면 86.7% 정도로 확인됨.
- 정밀도와 재현율 교차 검증: scoring 키워드의 인자로 "precision" 또는 "recall" 지정

```
from sklearn.model_selection import cross_val_score  
cross_val_score(sgd_clf, X_train, y_train, cv=3, scoring="accuracy")
```

스케일링의 중요성

- 표준화 스케일링만 해도 모델의 예측 정확도가 89.7% 까지 향상됨

```
from sklearn.preprocessing import StandardScaler  
  
scaler = StandardScaler()  
X_train_scaled = scaler.fit_transform(X_train.astype("float64"))  
cross_val_score(sgd_clf, X_train_scaled, y_train, cv=3, scoring="accuracy")
```

3.5. 오류 분석

- 오류 분석을 통해 모델의 성능을 평가하고 개선시키는 방안을 모색해야 함.
- 모델의 성능 평가: 오차 행렬 활용

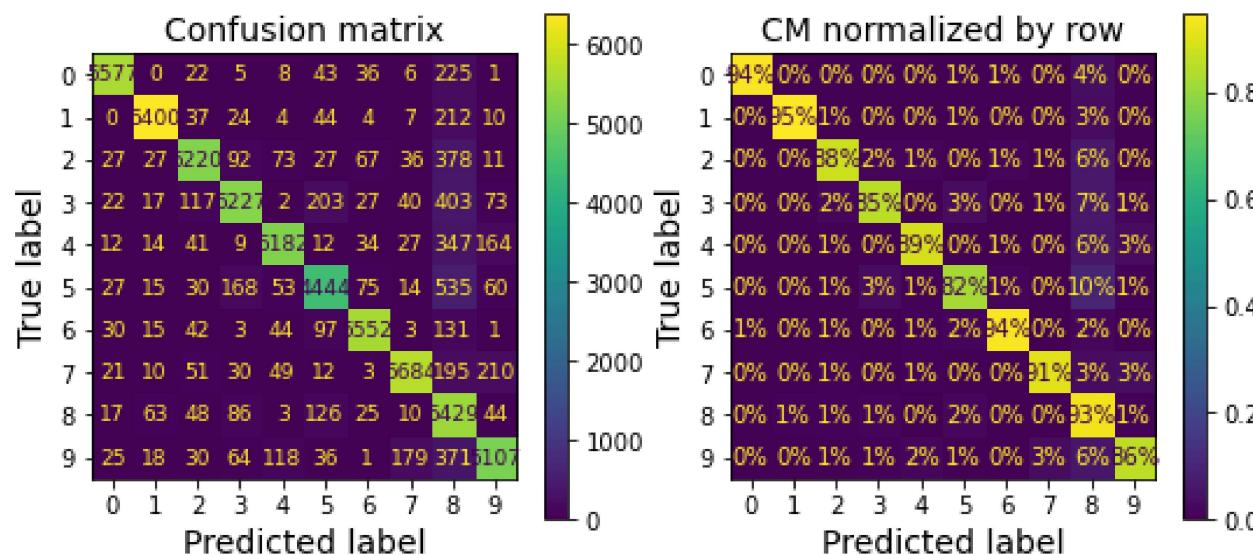
다중 클래스 분류 모델의 오차 행렬

```
from sklearn.model_selection import cross_val_predict
from sklearn.metrics import ConfusionMatrixDisplay

y_train_pred = cross_val_predict(sgd_clf, X_train_scaled, y_train, cv=3)

ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred)

ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
... normalize="true", values_format=".0%")
```

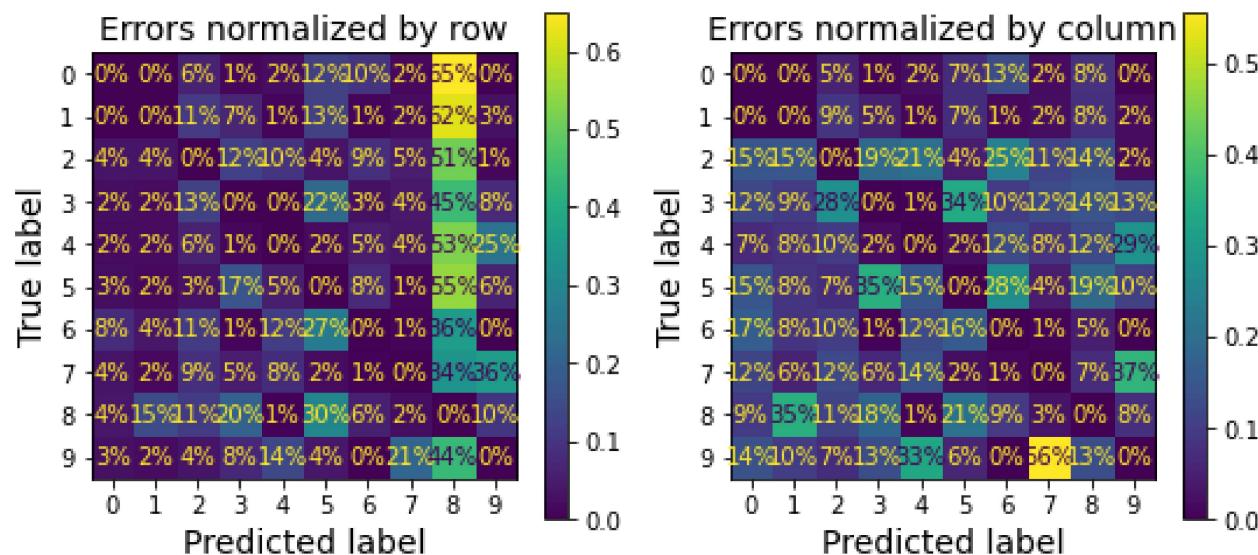


오차율 활용

```
sample_weight = (y_train_pred != y_train)

ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
... sample_weight=sample_weight, normalize="true", values_format=".0%")

ConfusionMatrixDisplay.from_predictions(y_train, y_train_pred,
... sample_weight=sample_weight, normalize="pred", values_format=".0%")
```



개별 오류 확인

		Predicted label				
		3	3	3	3	3
True label	3	3	3	3	3	3
	3	3	3	3	3	3
3	3	3	3	3	3	3
	3	3	3	3	3	3
3	3	3	3	3	3	3
	3	3	3	3	3	3
3	3	3	3	3	3	3
	3	3	3	3	3	3
5	5	5	5	5	5	5
	5	5	5	5	5	5
5	5	5	5	5	5	5
	5	5	5	5	5	5
5	5	5	5	5	5	5
	5	5	5	5	5	5
5	5	5	5	5	5	5
	5	5	5	5	5	5

데이터 증식

- SGD 분류 모델: 선형 회귀를 사용하기에 특히나 성능이 좋지 않음.
- 보다 좋은 성능의 모델을 사용할 필요 있음.
- 하지만 기본적으로 보다 많은 훈련 이미지가 필요함.
- 새로운 이미지를 구할 수 있으면 좋겠지만 일반적으로 매우 어려움.
- 데이터 증식 활용: 기존의 이미지를 조금씩 회전하거나, 뒤집거나, 이동하는 방식 등으로 보다 많은 이미지를 훈련셋에 포함시킬 수 있음.