

# Classification(분류)

## - Part 1 -

## 주요내용

---

- MNIST 데이터셋
- 이진 분류기 훈련
- 분류기 성능 측정

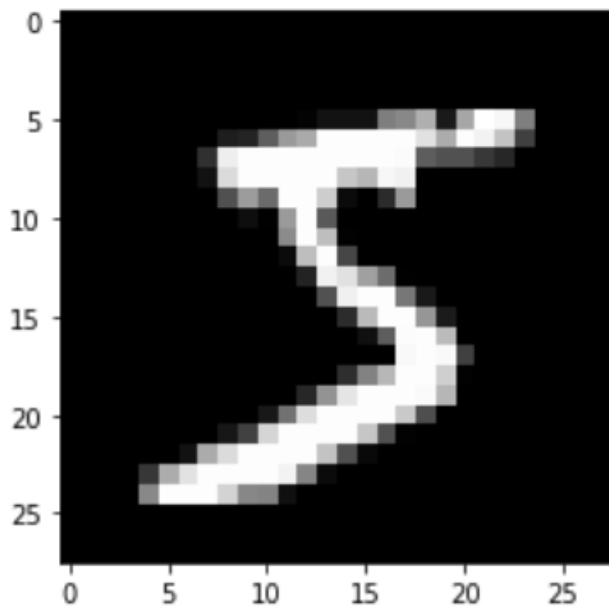
# MNIST 데이터셋

---

- 미국 고등학생과 인구조사국 직원들이 손으로 쓴 70,000개의 숫자 이미지로 구성된 데이터셋
  - 0부터 9까지 숫자들에 대한 손글씨 이미지
- 각 이미지는  $28 \times 28 = 784$ 개의 픽셀들로 구성된 이미지 데이터
- 각 이미지는 2차원 배열이 아닌 길이가 784인 1차원 배열로 제공
- (이미지 데이터셋의 shape) = (70000, 784)
  
- 레이블: 총 70,000개의 사진 샘플들 각각이 어떤 숫자를 나타내는지에 대한 레이블링이 되어 있음
- 레이블 데이터셋의 shape은 (70000,)

# MNIST 데이터셋

- 숫자 5에 대한 이미지 샘플 & MNIST 데이터셋에 처음 100개 이미지들



## 문제 정의

---

- 지도학습: 각 이미지가 어떤 숫자를 나타내는지에 대한 레이블 지정되어 있음
- 분류: 주어진 이미지 데이터가 0부터 9까지 중 어떤 숫자에 해당하는지를 예측
  - 여기서는 0~9까지의 숫자 각각이 하나의 클래스에 해당
  - 주어진 이미지 샘플이 총 10개 클래스 중 어느 클래스에 해당하는지를 분류하는 문제이며 이와 같은 문제를 다중 클래스 분류(multiclass classification) 또는 다항 분류(multinomial classification)라고 부름
- 배치 또는 온라인 학습: 둘 다 가능
  - 확률적 경사하강법(stochastic gradient descent, SGD) 분류기: 배치와 온라인 학습 모두 가능
  - 랜덤 포레스트 분류기: 배치 학습

## 훈련셋과 테스트셋 나누기

---

- MNIST 데이터셋은 이미 6:1의 비율로 훈련셋과 테스트셋으로 분류되어 있음
  - 훈련셋
    - 훈련용 이미지 데이터셋(X\_train): 첫 60000개 이미지
    - 훈련용 레이블(y\_train): 첫 60000개 이미지에 대한 레이블
  - 테스트셋
    - 테스트용 이미지 데이터셋(X\_test): 나머지 10000개 이미지
    - 테스트용 레이블(y\_test): 나머지 10000개 이미지에 대한 레이블

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

- 모든 샘플들이 랜덤하게 잘 섞여 있는 상태임

# 이진 분류기 훈련

## 숫자 5-감지기 (5-detector)

- 10개 클래스로 분류하는 다중 클래스 분류 모델을 훈련하기에 앞서 주어진 이미지 샘플이 숫자 5에 해당하는지 아닌지 여부를 판단하는 이진 분류기(binary classifier)를 훈련시키고자 함
- 이를 통해 분류기의 기본 훈련 과정과 성능 평가 방법을 알아보하고자 함
- 각 이미지에 대한 레이블은 0 또는 1로 수정되어야 함
  - 레이블 0: 해당 이미지가 5 이외의 숫자에 해당함을 의미하는 레이블
  - 레이블 1: 이미지가 숫자 5에 해당함을 의미하는 레이블
- 훈련셋 레이블(y\_train\_5), 테스트셋 레이블(y\_test\_5)

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



## 이진 분류기로 SGD 분류기 활용

- SGD Classifier
  - 확률적 경사 하강법(stochastic gradient descent) 분류기라고 불림
  - 한번에 하나씩 훈련 샘플을 이용하여 학습한 후 파라미터를 조정
  - 매우 큰 데이터셋 처리에 효율적이며 온라인 학습에도 적합
  - 사이킷런의 `SGDClassifier` 클래스 활용
- `SGDClassifier` 객체의 `fit()` 메서드 호출을 통해 분류기 훈련
  - `X_train`: 훈련용 이미지 데이터셋, `y_train_5`: 훈련용 이미지에 대한 레이블(0 또는 1로 이루어짐)

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(random_state=42)
sgd_clf.fit(X_train, y_train_5)
```
- `Predict()` 메서드 호출을 통해 주어진 이미지가 0(숫자 5에 해당하지 않음) 또는 1(숫자 5에 해당) 중 어느 클래스에 해당하는지 예측

```
>>> sgd_clf.predict([some_digit])
array([ True])
```

## 분류기 성능 측정

---

- 분류기 성능 평가는 회귀 모델 평가보다 고민할 점들이 많음
- 분류기 성능 측정 기준으로 다음 세가지가 많이 사용됨
  - 정확도(accuracy)
  - 정밀도(precision) / 재현율(recall)
  - ROC curve의 AUC

## 교차 검증을 사용한 정확도(accuracy) 측정

- k-fold 교차 검증(cross validation) 기법을 이용하여 SGD 분류기의 성능 평가
- 성능 평가 기준: 정확도(accuracy)
  - $\text{accuracy} = (\text{number of correctly classified samples}) / (\text{total number of samples}) \rightarrow$  간단하게 말하면 분류기에 의해 올바르게 분류된 샘플들의 비율을 의미

- 사이킷런의 `cross_val_score()` 함수 활용

```
>>> from sklearn.model_selection import cross_val_score
>>> cross_val_score(sgd_clf, X_train, y_train_5, cv=3, scoring="accuracy")
array([0.95035, 0.96035, 0.9604 ])
```

- 95%를 넘는 정확도를 보임

## SGD 분류기의 정확도에 대한 고찰

- 무조건 “5가 아님”으로 찍는 DummyClassifier 분류기를 생성하여 정확도 측정

```
from sklearn.dummy import DummyClassifier
```

```
dummy_clf = DummyClassifier()  
dummy_clf.fit(X_train, y_train_5)
```

```
>>> cross_val_score(dummy_clf, X_train, y_train_5, cv=3, scoring="accuracy")  
array([0.90965, 0.90965, 0.90965])
```

- 무조건 “5가 아님”으로 찍는 경우에도 90% 넘는 정확도를 보임
- 훈련셋의 10%만 숫자 5 이미지이고 나머지 90%는 5가 아닌 이미지들이기 때문
- 이처럼 훈련셋 샘플들의 클래스 불균형이 심한 경우에는 분류기의 성능 측정 지표로 정확도 (accuracy)는 부적합 함

6

10

5

가

.

# 오차 행렬(Confusion Matrix)

- 클래스 별 예측 결과를 정리한 행렬
- 행은 실제 클래스를(true label), 열은 분류기에 의해 예측된 클래스(predicted label)를 의미
  - (오른쪽 오차 행렬의 cat 행에 대한 설명)
- E.g., 오른쪽 오차 행렬에서 cat에 해당하는 이미지 샘플을 dog으로 잘못 분류한 횟수를 알고 싶다면 (cat 행, dog 열)에 위치한 값을 확인

234      cat      dog

Confusion Matrix

True label \ Predicted label	airplane	automobile	bird	cat	deer	dog	frog	horse	ship	truck
airplane(n=1000)	779	8	47	7	16	10	9	38	32	54
automobile(n=1000)	5	866	3	0	1	4	4	3	8	106
bird(n=1000)	32	1	676	16	67	67	85	33	6	17
cat(n=1000)	7	4	34	453	48	234	119	66	8	27
deer(n=1000)	7	2	27	21	723	32	82	88	6	12
dog(n=1000)	1	3	19	53	28	813	27	48	2	6
frog(n=1000)	5	1	18	16	15	19	903	12	4	7
horse(n=1000)	4	1	5	7	17	41	4	912	0	9
ship(n=1000)	53	14	3	7	0	5	7	5	856	50
truck(n=1000)	7	20	3	4	0	3	1	8	10	944

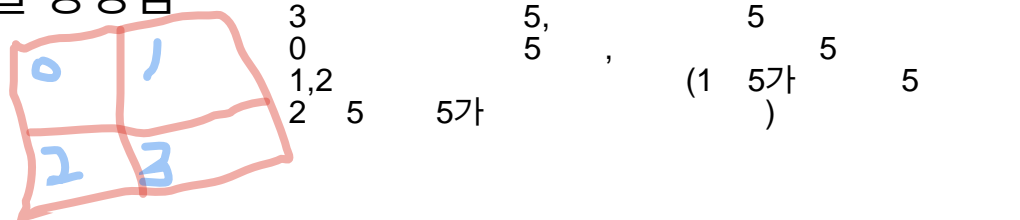
## 숫자5-감지기에 대한 오차 행렬

```
from sklearn.model_selection import cross_val_predict

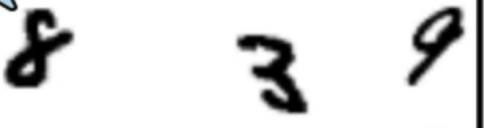
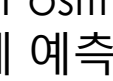
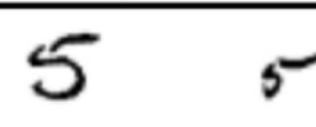
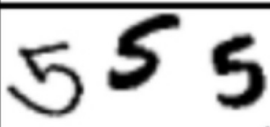
y_train_pred = cross_val_predict(sgd_clf, X_train, y_train_5, cv=3)

>>> from sklearn.metrics import confusion_matrix
>>> cm = confusion_matrix(y_train_5, y_train_pred)
>>> cm
array([[53892,   687],
       [ 1891, 3530]])
```

- **cross\_val\_predict()** 함수: cross\_val\_score() 함수와 마찬가지로 k-fold 교차 검증 수행하지만 성능 평가 점수 대신에 각 validation fold의 샘플들에 대해 분류기가 예측한 결과(y\_train\_pred)를 반환
- **confusion\_matrix()** 함수: 타깃 클래스(y\_train\_5)와 예측된 클래스(y\_train\_pred) 정보를 이용하여 숫자5-감지기에 대한 오차 행렬 생성
- 결과적으로 2x2 크기의 오차 행렬 생성됨
  - (위 오차 행렬에 대한 세부 설명)



## 숫자5-감지기에 대한 오차 행렬

		Predicted	
		Negative	Positive
Actual	non-5 Negative	<div>TN</div> 	<div>FP</div> 
	Positive 5	<div>FN</div> 	<div>TP</div> 

Precision (e.g., 3 out of 4)

Recall (e.g., 3 out of 5)

true가  
false가  
positive

가  
, negative

- True Positive(TP): 실제 5인 이미지를 5로 맞게 예측한 경우
  - False Positive(FP): 실제 non-5 이미지를 5로 틀리게 예측한 경우
  - True Negative(TN): 실제 non-5인 이미지를 non-5로 맞게 예측한 경우
  - False Negative(FN): 실제 5 이미지를 non-5로 틀리게 예측한 경우
- 오차 행렬이 분류 결과에 대한 많은 정보를 제공하지만 더 요약된 지표가 필요함
    - 정밀도(precision), 재현율(recall)

## 정밀도(Precision), 재현율(Recall)

\* 앞서 살펴본 accuracy는 “정확도”로 번역됨



## 정밀도(Precision)

$$\text{precision} = \frac{TP}{TP + FP}$$

가

- **Positive 예측의 정확도**를 나타내는 지표:

- 정밀도는 classifier가 **positive(e.g., 숫자5 클래스)**라고 예측한 인스턴스(샘플)들 중에서 실제 **positive** 인 인스턴스의 **비율**

- 분류기의 정밀도(precision)가 높다는 것은 주어진 instance를 classifier가 positive라고 예측하면 그 예측이 맞을(진짜 positive일) 확률이 높다는 의미

- E.g., 정밀도가 높은 classifier가 숫자 5라고 예측한 이미지는 진짜 숫자 5 이미지일 확률이 높다는 의미

- 앞서 살펴본 SGD classifier의 정밀도:  $\frac{3530}{3530+687} = 0.83708..$

```
>>> from sklearn.metrics import precision_score, recall_score
>>> precision_score(y_train_5, y_train_pred) # == 3530 / (68
0.8370879772350012
```

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

# 정밀도

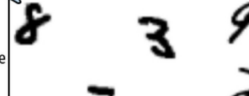

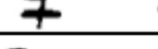
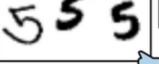
- 정밀도 100%를 달성할 수 있는 간단한 방법
  - E.g., 기본적으로 항상 non-5로 예측하다가 정말로 숫자 5가 확실하다고 판단되는 이미지 샘플 하나만 5로 예측 →  $FP=0$  &  $TP=1$  →  $Precision = \frac{1}{1/(0+1)} = 100\%$
  - 하지만 이러한 분류기는 숫자 5인 이미지들 중 오직 하나만 숫자 5로 맞게 예측한 것이기 때문에 전혀 유용하지 않음
  - 결과적으로 정밀도만으로는 좋은 분류기를 가려낼 수 없으며, 따라서 정밀도를 보완할 수 있는 성능 지표가 추가로 필요함

precision( )가

??? -> NO

## 재현율(Recall)

$$\text{recall} = \frac{TP}{TP + FN} \quad (1)$$

		Predicted	
		Negative	Positive
Actual	Negative	<div>TN</div> 	<div>FP</div> 
	Positive		<div>TP</div> 
		<div>FN</div>	<div>Precision (e.g., 3 out of 4)</div>
		<div>Recall (e.g., 3 out of 5)</div>	

- Positive(e.g., 숫자5 클래스에 해당하는) 샘플에 대한 예측 정확도
  - 데이터셋에 포함된 모든 positive instance들 중에서 classifier가 positive라고 분류한 비율
  - E.g., 실제 숫자5에 해당하는 이미지 샘플들 중 분류기가 숫자5라고 맞게 예측한 비율
- 재현율(recall)이 높다는 것은 classifier가 positive instance를 놓치지 않고 잘 감지해낸다는 의미
- SGD classifier의 재현율:  $\frac{3530}{3530+1891} = 0.6511..$

```
>>> recall_score(y_train_5, y_train_pred)
0.6511713705958311
```

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

## F<sub>1</sub> Score

?

- 정밀도와 재현율의 harmonic mean
  - 서로 다른 종류의 두 분류기 모델의 성능을 하나의 metric으로 비교하고자 할 때 유용

- F<sub>1</sub> Score 계산식 예:

$$F_1 = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = \frac{TP}{TP + \frac{FN+FP}{2}}$$

- 정밀도와 재현율이 모두 높아야 F<sub>1</sub> Score도 높게 나타나는 경향

- SGD classifier의 F<sub>1</sub> Score :

```
>>> from sklearn.metrics import f1_score
>>> f1_score(y_train_5, y_train_pred)
0.7325171197343846
```

- F<sub>1</sub> Score 계산 시 정밀도와 재현율에 대한 가중치 설정
  - 위 계산식은 정밀도와 재현율을 동일한 중요도로 반영하여 F<sub>1</sub> Score 계산하는 경우
  - 경우에 따라서는 정밀도와 재현율 중 하나에 더 높은 가중치를 두어야 할 때도 있음

## 정밀도(Precision) vs 재현율(Recall)

- 모델 사용의 목적에 따라 정밀도와 재현율의 중요도가 상이할 수 있음
- 재현율이 보다 중요한 경우: e.g., 암 진단 분류기
  - 정밀도: 암(즉, positive)으로 진단된 경우 중에 실제로 암인 경우의 비율. 암으로 진단한 경우에 대한 신뢰도/정확도를 의미.
  - 재현율: 실제로 암인 케이스들 중에서 놓치지 않고 암으로 진단한 경우의 비율
  - 일반적으로 이 경우에는 암인 케이스를 놓치지 않고 암으로 예측해내는 것이 중요함
- 정밀도가 보다 중요한 경우: e.g., 아이가 시청해도 되는 안전한 동영상 분류기
  - 정밀도: 분류기에 의해 안전하다고(positive 클래스에 해당) 판단된 동영상 중에서 실제로도 안전한 동영상의 비율
  - 재현율: 실제로 안전한 동영상 중에서 분류기에 의해 안전하다고 예측된 영상 샘플들 비율
  - 이 경우에는 안전한 동영상으로 예측된 샘플 중에 실제로는 안전하지 않은 동영상이 포함되어 있을 경우 문제가 됨

19

가

## 정밀도와 재현율 간 Trade-off

- 정밀도와 재현율은 trade-off 관계임(상호 반비례 관계)
  - E.g., 정밀도가 높아질 수록 재현율은 저하됨
- 예1: 아이가 시청해도 되는 안전한 동영상 분류기
  - 정밀도를 높이기 위해서는 확실하게 안전하다고 판단되는 영상만을 안전한 것으로 예측해야함.  
하지만 이렇게 되면 실제 안전한 동영상임에도 (작은 이유로) 안전하지 않다고 잘못 예측되는 비율이 증가하게 되며, 이는 재현율(recall) 저하를 의미한다.
- 예2: 암 진단 분류기
  - 재현율을 높이기 위해서는 조금이라도 암일 가능성이 있는 케이스를 보수적으로 암으로 진단해야함. 하지만 이런 접근을 취할 경우 암으로 진단한 케이스 중에서 실제로는 암이 아닌 케이스의 비율이 높아질 수 있으며, 이는 정밀도(precision)가 떨어진다는 의미.

= 가 =

- 
- 정밀도와 재현율 모두 높은 분류기는 이론적으로 불가능함
  - 주어진 문제의 성격에 따라 요구되는 최소한의 정밀도/재현율 수준이 있을 것이고 그것을 충족시키는 분류기를 도출해야함
  - 이를 위해서는 분류기가 주어진 인스턴스(샘플)의 클래스를 예측하는 기본 원리에 대한 이해가 필요

## 결정 함수(Decision Function)와 결정 임계값(Decision Threshold)

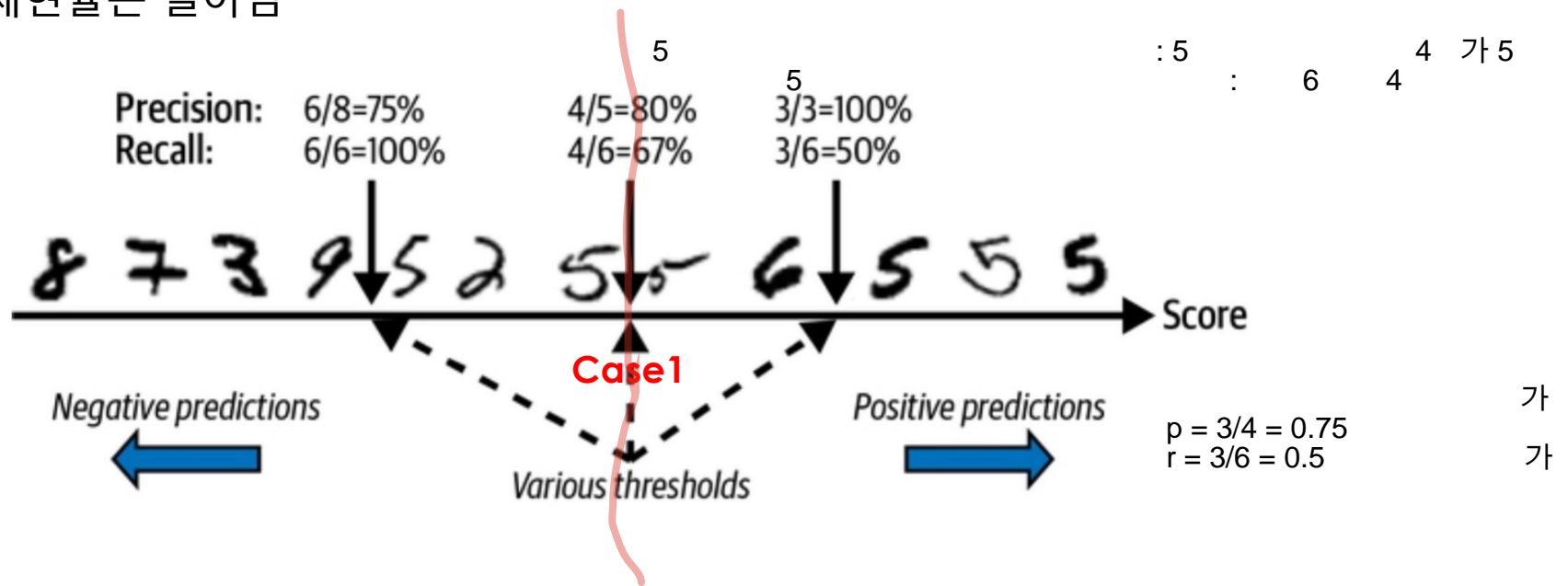
---

- SGD 분류기는 결정 함수(decision function)를 사용하여 주어진 인스턴스의 클래스를 결정하기 위한 점수를 계산함
- 계산된 점수가 결정 임계값(decision threshold) 보다 크면 positive 클래스(e.g., 숫자 5 클래스)로 예측하고, 반대의 경우 negative 클래스(e.g., non-5)로 예측함



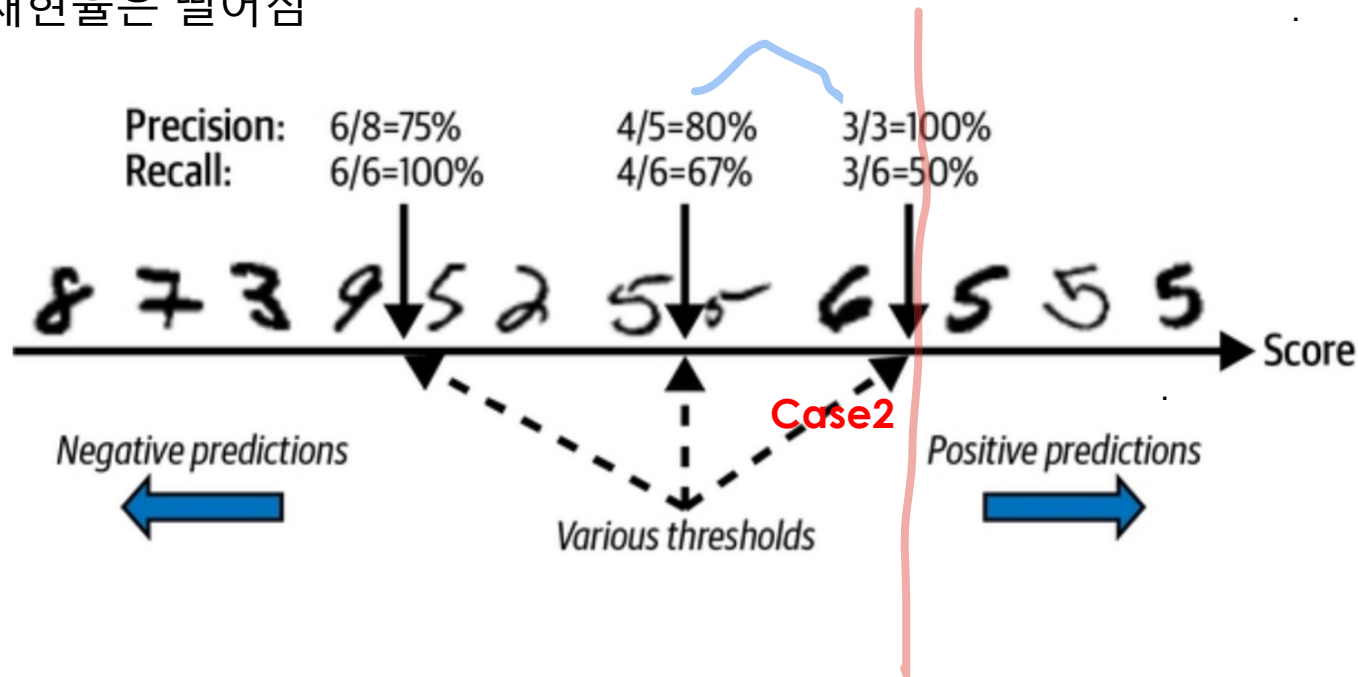
## 결정 함수(Decision Function)와 결정 임계값(Decision Threshold)

- 결정 임계값에 따른 분류기의 정밀도 및 재현율 변화 추이
  - E.g., 임계값을 높이면(5로 판단하는 기준을 더 까다롭게 한다는 의미로 이해) 정밀도는 높아지지만 재현율은 떨어짐



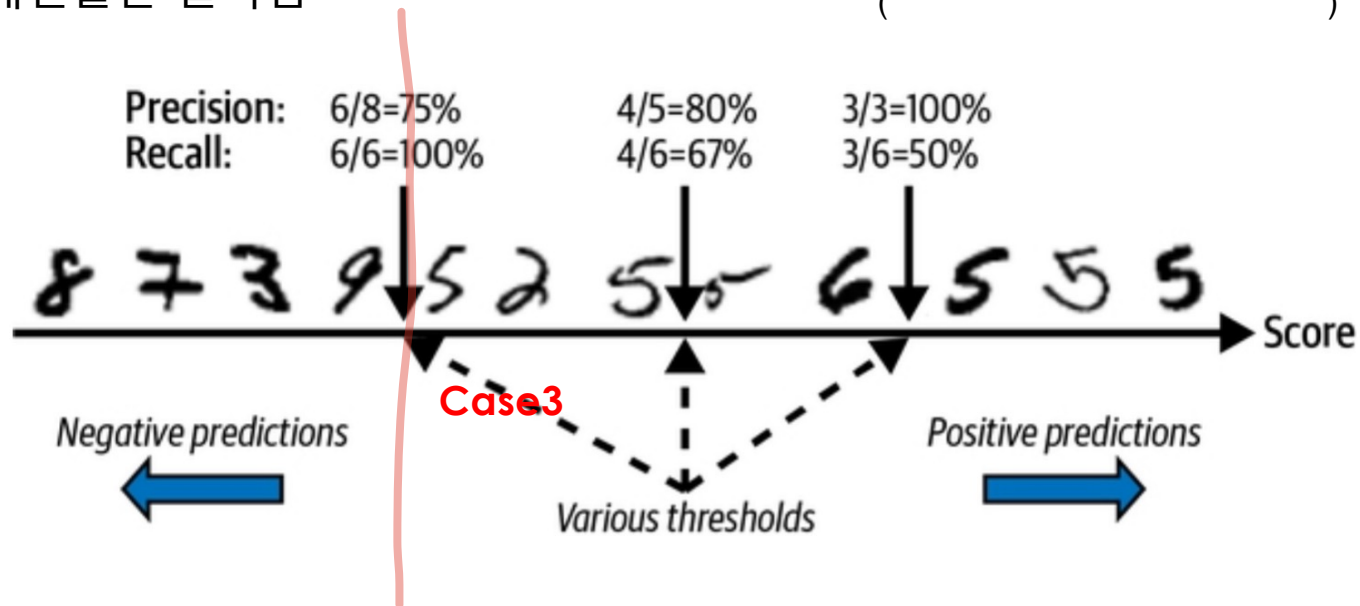
## 결정 함수(Decision Function)와 결정 임계값(Decision Threshold)

- 결정 임계값에 따른 분류기의 정밀도 및 재현율 변화 추이
  - E.g., 임계값을 높이면 (5로 판단하는 기준을 더 까다롭게 한다는 의미로 이해) 정밀도는 높아지지만 재현율은 떨어짐



## 결정 함수(Decision Function)와 결정 임계값(Decision Threshold)

- 결정 임계값에 따른 분류기의 정밀도 및 재현율 변화 추이
  - E.g., 임계값을 높이면(5로 판단하는 기준을 더 까다롭게 한다는 의미로 이해) 정밀도는 높아지지만 재현율은 떨어짐 ( )



---

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

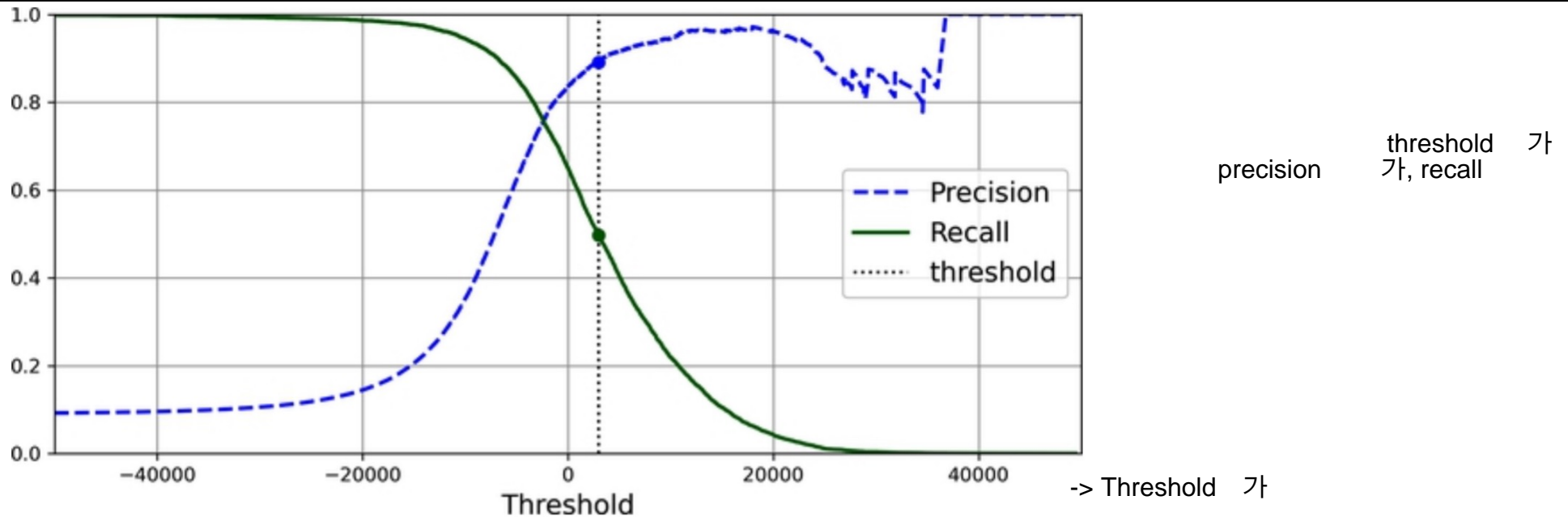
- k-fold 교차 검증 수행하며 결과로는 훈련셋의 각 인스턴스에 대해 decision function을 통해 계산된 점수를 반환

```
from sklearn.metrics import precision_recall_curve
```

```
precisions, recalls, thresholds = precision_recall_curve(y_train_5, y_scores)
```

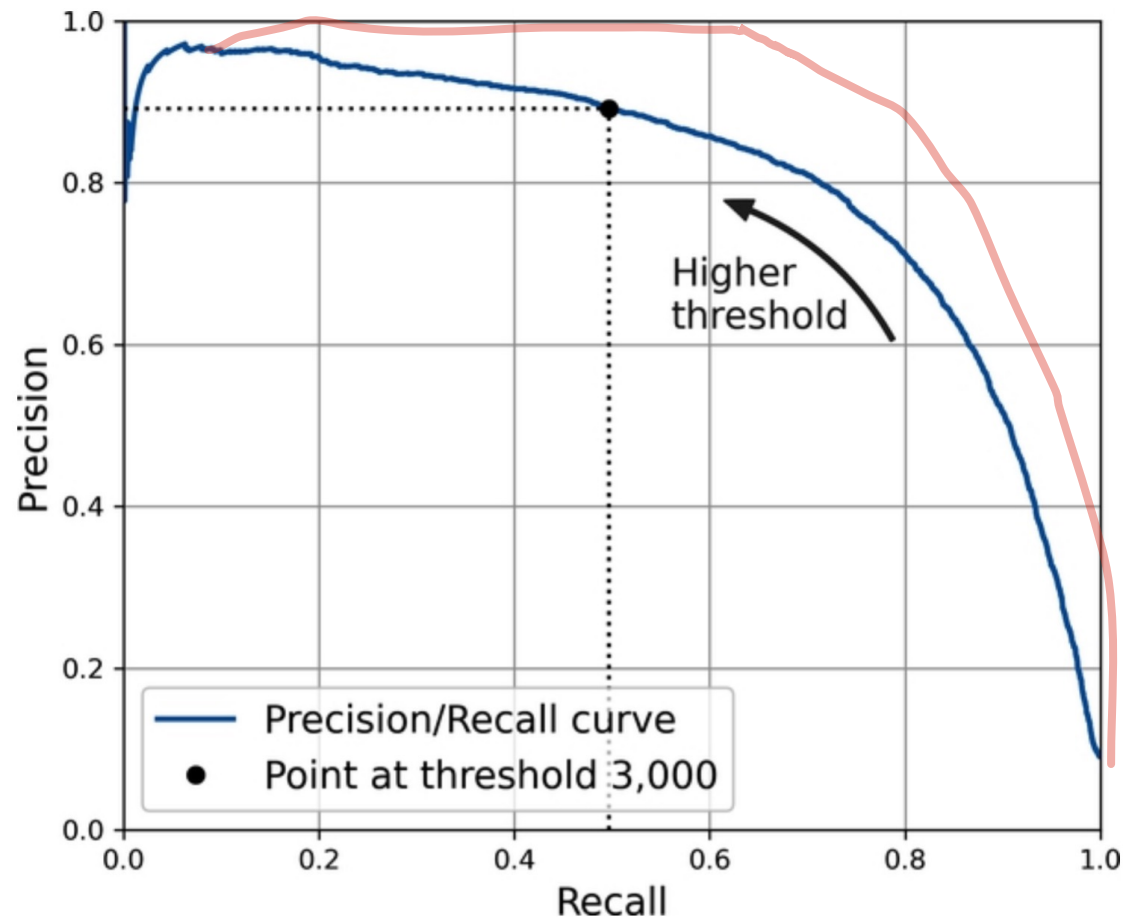
- 모든 가능한 결정 임계값에 대해 SGD 분류기의 precision과 recall을 계산해서 반환
- `precision_recall_curve()` 함수 결과로 3개 배열 리턴됨
  - precisions, thresholds 배열은 인덱스가 커짐에 따라 값이 증가하는 배열                   가                   가 .
  - 반대로 recalls 배열은 인덱스가 커짐에 따라 값이 감소하는 배열

## 결정 임계값에 따른 정밀도와 재현율 그래프



- 세로 방향 점선으로 표시된 결정 임계값을 사용할 경우 대략 90% 정밀도와 50% 재현율을 달성함
- 결정 임계값을 높였을 때 일시적으로 정밀도가 떨어지는 패턴이 관찰될 수 있음. 하지만 결정 임계값을 계속 높이면 결국엔 정밀도가 상승하게 됨

## 재현율 대 정밀도 그래프



## 정밀도/재현율 요구조건을 충족시키기 위한 결정 임계값 찾기

- 예 1: `precision_recall_curve()` 함수를 통해 계산된 `precisions`, `recalls`, `thresholds` 배열을 이용하여 정밀도 0.9 이상을 만족하기 위한 최소한의 결정 임계값 찾기

```
>>> idx_for_90_precision = (precisions >= 0.90).argmax()  
>>> threshold_for_90_precision = thresholds[idx_for_90_precision]  
>>> threshold_for_90_precision  
3370.0194991439557
```

- `precisions` 배열에서 0.9 이상의 값이 저장된 lowest index를 확인  
    why                      ?    가                      recall                      가
- 확인된 index 위치에 해당하는 결정 임계값을 `thresholds` 배열에서 확인하면 그 임계값이 정밀도 0.9 이상을 충족시키는 임계값 최소치에 해당함

`thresholds`, `precisions`  
`recall`

가    가

가(                      )

## 정밀도/재현율 요구조건을 충족시키기 위한 결정 임계값 찾기

Colab  
실습

- 예2: `precision_recall_curve()` 함수를 통해 계산된 `precisions`, `recalls`, `thresholds` 배열을 이용하여 재현율 0.8 이상을 충족시키기 위한 결정 임계값 찾기

recall

recall



## 정밀도/재현율 요구조건을 충족시키기 위한 결정 임계값 찾기

- 예2: `precision_recall_curve()` 함수를 통해 계산된 `precisions`, `recalls`, `thresholds` 배열을 이용하여 재현율 0.8 이상을 충족시키기 위한 결정 임계값 최대치 찾기

```
idx_for_80_recall = (recalls >= 0.80).argmin()
```

- recalls 배열은 값이 줄어드는 패턴  
- recalls 배열에서 값이 0.8 미만인 lowest index 확인

```
thresholds[idx_for_80_recall-1]
```

(idx\_for\_80\_recall-1) index 위치의 임계값이 재현율 80% 이상을 충족시키기 위한 임계값 최대치에 해당함

```
y_train_pred_80_recall = (y_scores >= thresholds[idx_for_80_recall-1])
```

선택된 임계값으로 훈련셋 샘플들에 대한 예측 수행

```
precision_score(y_train_5, y_train_pred_80_recall)
```

0.7120341487440486

타겟 클래스(y\_train\_5)와 예측 클래스(y\_train\_pred\_80\_recall)을 이용하여 precision score 계산

```
recall_score(y_train_5, y_train_pred_80_recall)
```

0.8000368935620734

타겟과 예측을 이용하여 recall score 계산

## ROC 곡선과 AUC

- ROC(receiver operating characteristic) 곡선은 이진 분류기 성능 평가에 널리 사용됨
- ROC 곡선은 결정 임계값이 달라짐에 따라 FPR과 TPR의 변화 추이를 보여주는 곡선
  - TPR(true positive rate, 재현율의 또 다른 명칭)
  - FPR(false positive rate): 실제 negative 클래스에 해당하는 모든 샘플들 중 positive로 잘못 예측된 비율
    - E.g., non-5인 모든 이미지 샘플들 중 숫자 5로 잘못 분류된 샘플들의 비율
    - $FPR = FP / (FP + TN)$

TPR

FPR

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

## ROC 곡선과 AUC

- ROC 곡선을 그리기 위해서는 모든 가능한 결정임계값에 대해 FPR과 TPR을 계산해야하며 `roc_curve()` 함수를 이용하여 계산 가능

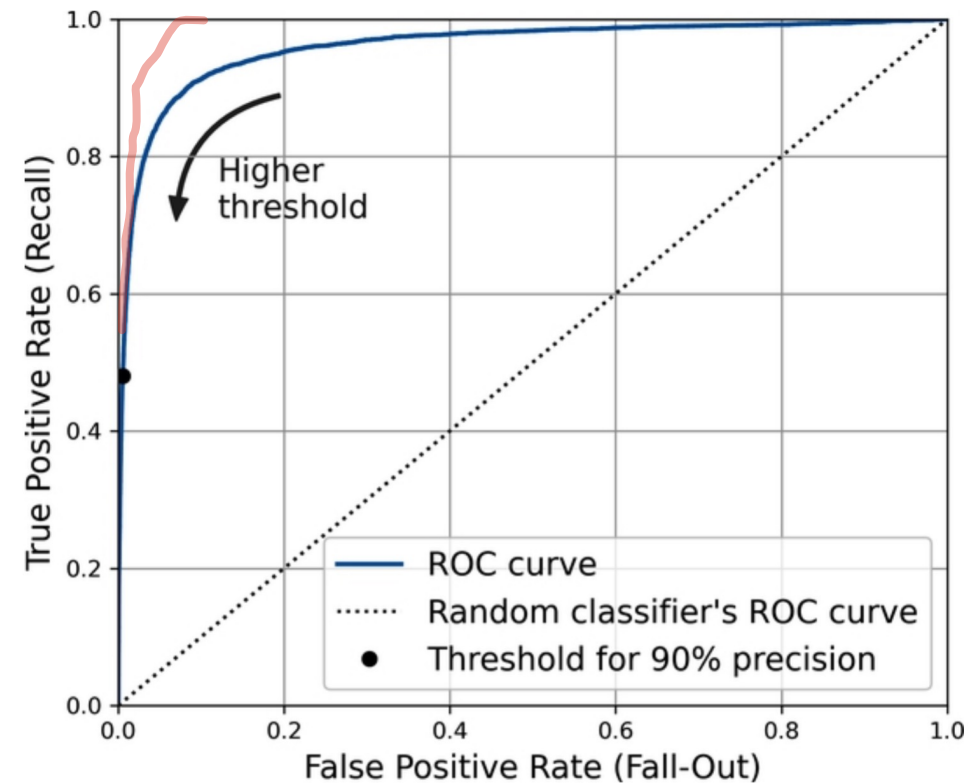
```
from sklearn.metrics import roc_curve  
  
fpr, tpr, thresholds = roc_curve(y_train_5, y_scores)
```

roc\_curve

# SGD 분류기에 대한 ROC 곡선

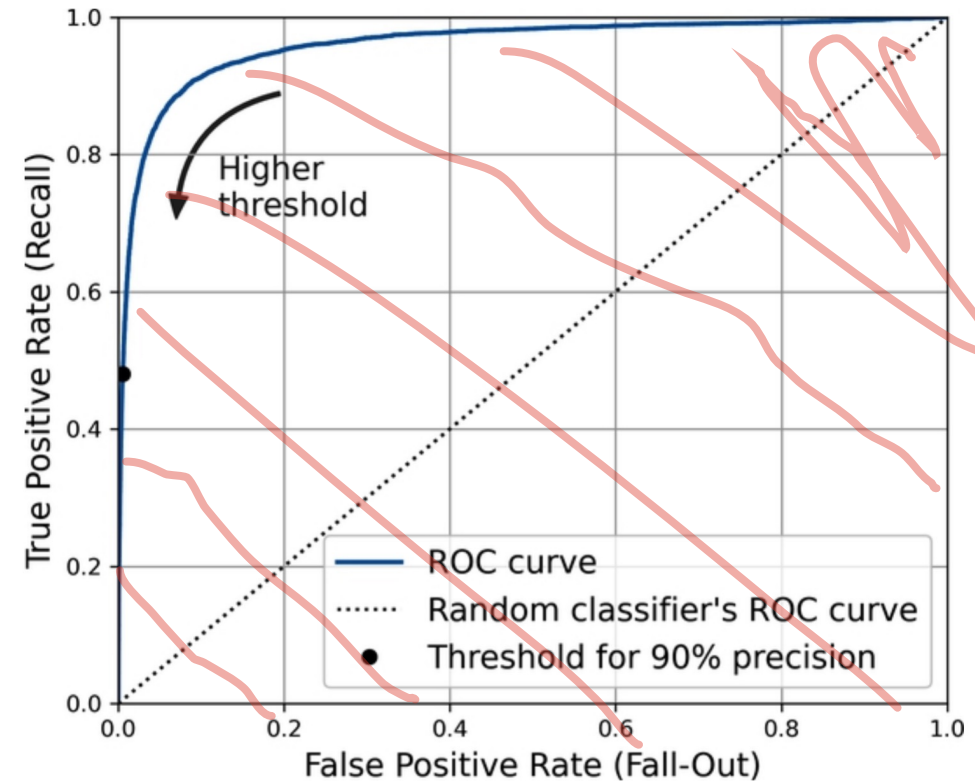
1 가

- FPR과 TPR(재현율) 간에도 trade-off 관계 존재
  - 결정 임계값 조정을 통해 TPR을 높이면 FPR 또한 증가하게 됨
- 좋은 분류기이기 위해서는 FPR은 최대한 낮게 유지하면서도 높은 수준의 TPR에 도달할 수 있어야 함



# AUC와 분류기 성능

- AUC(area under the curve)
  - ROC 곡선의 AUC는 ROC 곡선 아래의 면적을 의미한다.
- 좋은 분류기이기 위해서는 FPR은 최대한 낮게 유지하면서도 높은 수준의 TPR에 도달할 수 있어야 함
- 따라서 좋은 분류기의 ROC 곡선은 top-left 코너에 근접하게 됨
- AUC가 1에 가까울 수록 좋은 성능의 분류기로 평가된다.



## ROC 곡선 vs 정밀도(precision)/재현율(recall) 곡선

---

- 다음 경우에는 정밀도/재현율 곡선이 분류기 성능 평가 방법으로 선호됨
  - 전체 샘플들 중 positive 클래스에 해당하는 샘플들이 드문 경우
  - 또는 false negative 보다 false positive가 더 문제가 되는 경우
    - E.g., 아이가 시청해도 되는 안전한 동영상 분류기의 경우 실제로는 아이가 보면 안되는 위험한 동영상(negative 클래스에 해당)인데 안전한 것(positive 클래스)으로 분류되는 것이 심각한 문제임
- 나머지 경우에는 ROC 곡선이 선호됨

## 정밀도/재현율 곡선을 이용한 분류기 성능 비교

- 좋은 분류기이기 위해서는 높은 수준의 recall에서도 높은 수준의 precision을 유지할 수 있어야 함
- 분류기의 정밀도/재현율 곡선이 top-right 코너에 근접할 수록 좋은 성능의 분류기로 평가됨
  - 정밀도/재현율 곡선의 AUC 또한 1에 가까울 수록 좋은 분류기라는 의미
- SGD 분류기 vs Random Forest 분류기

