3장 평가

01 정확도

$$\text{정확도(Accuracy)} = \frac{\text{예측 결과가 동일한 데이터 건수}}{\text{전체 예측 데이터 건수}}$$

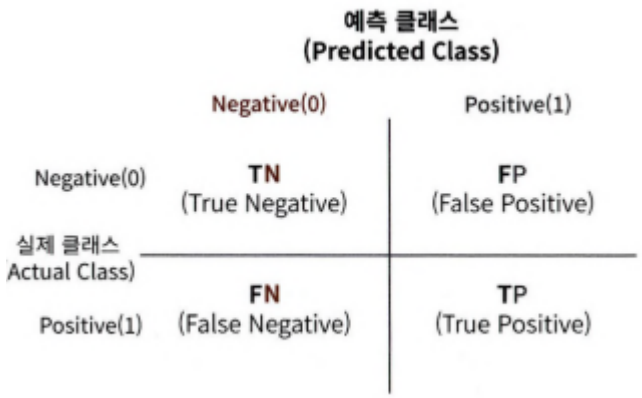
- 정확도 평가 지표는 불균형한 레이블 데이터 세트에서는 성능 수치로 사용하면 안됨
- 정확도 뿐만 아니라 여러 가지 분류 지표와 함께 적용하여 ML 모델 성능을 평가해야 함
- 예시) MNIST 데이터셋을 변환해 불균형한 데이터 세트로 만든 뒤에 정확도 지표 적용 시 생기는 문제점
    - MNIST 데이터셋: 0부터 9까지의 숫자 이미지의 픽셀 정보를 가지고 있고, Digit를 예측하는 데 사용됩니다.
    - 사이킷런은 load_digits() API를 통해 MNIST 데이터 세트를 제공함
    - 원래 MNIST 데이터셋: 레이블 값이 0부터 9까지 있는 멀티 레이블 분류를 위한 것입니다.
    - 변경된 MNIST 데이터셋: 레이블 값이 7인 것만 True, 나머지 값은 모두 False로 변환

      즉, 전체 데이터의 10%만 True, 나머지 90%는 False인 불균형한 데이터 세트로 변형..
    - 모든 데이터를 False로 예측하는 classifier를 이용해 정확도: 약 90% ⇒ 데이터 분포가 균일하지 않은 경우, 높은 수치가 나타날 수 있

02 오차행렬

오차행렬

- 학습된 분류 모델이 예측을 수행하며 얼마나 헷갈리고(confused)있는지 보여주는 지표
- 오차 행렬 사분면



  - TN: 예측값 - Negative 값 0, 실제 값 - Negative 값 0
  - FP: 예측값 -Positive 값 1, 실제 값 - Negative 값 0
  - FN: 예측값 - Negative 값 0, 실제 값 - Positive 값 1
  - TP, TN, FP, TN 값은 Classifier 성능의 여러 면모를 판단할 수 있는 기반 정보를 제공함
  - 이 값을 조합해 Classifier의 성능을 측정할 수 있는 주요 지표인 정확도(Accuracy), 정밀도(Precision), 재현율(Recall) 값을 알 수 있음
- 사이킷런에서 오차행렬 API: confusion_matrix()
- 정확도: 예측값과 실제 값이 얼마나 동일한가에 대한 비율만으로 결정됨

  ⇒ 오차 행렬에서 TTN과TP에 좌우됨

  > 정확도 = 예측 결과와 실제 값이 동일한 건수/전체 데이터 수 = (TN + TP)/(TN + FP + FN + TP)

- 중점적으로 찾아야 하는 매우 적은 수의 결괏값에 Positive를 설정해 1값을 부여함
- 그렇지 않으면 보통 Negative로 0 값을 부여함
- 불균형한 이진 분류 데이터 세트에서는 Positive 데이터 건수가 매우 작음

  ⇒ Positive보다는 Negative로 예측 정확도가 높아지는 경향이 발생
- Negative로 예측할 때 정확도가 높아서 FN(Negative로 예측할 때 틀린 데이터 수)이 매우 작고, Positive로 예측하는 경우가 작음 ⇒ FP도 매우 작아짐

03 정밀도와 재현율

정밀도 = TP / (FP + TP)
재현율 = TP / (FN + TP)

정밀도 (양성 예측도)

- 정의: 예측을 Positive로 한 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율
- FP 十 TP: 예측을 Positive로 한 모든 데이터 건수
- TP: 예측과 실제 값이 Positive로 일치한 데이터 건수
- TP 높이고 FP 낮추기
- 실제 Negative 음성인 데이터 예측을 Positive 양성으로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우 ⇒ 정밀도가 상대적으로 더 중요

재현율 (민감도 or TPR)

- 실제 값이 Positive인 대상 중에 예측과 실제 값이 Positive로 일치한 데이터의 비율
- FN + TP: 실제 값이 Positive인 모든 데이터 건수
- TP: 예측과 실제 값이 Positive로 일치한 데이터 건수
- TP 높이고 FN 낮추기
- 실제 Positive 양성인 데이터 예측을 Negative로 잘못 판단하게 되면 업무상 큰 영향이 발생하는 경우 ⇒ 재현율이 상대적으로 더 중요

정밀도/재현율 트레이드오프

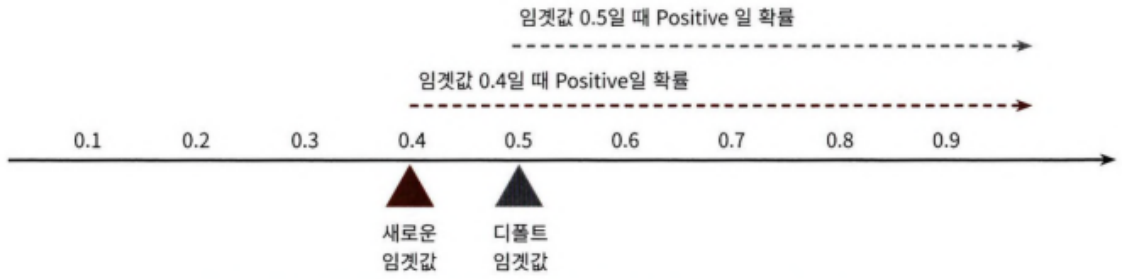
- 분류의 결정 임계값 조정 → 정밀도 or 재현율 수치 높이기
- 정밀도와 재현율은 상호 보완적인 평가 지표

  ⇒ 정밀도/재현율의 트레이드오프: 어느 한 쪽을 강제로 높이면 다른 하나의 수치는 떨어지기 쉬움

- predict_proba()

입력 파라미터	predict( ) 메서드와 동일하게 보통 테스트 피처 데이터 세트를 입력
반환 값	개별 클래스의 예측 확률을 ndarray m x n (m: 입력값의 레코드 수, n: 클래스 값 유형) 형태로 반환. 입력 테스트 데이터 세트의 표본 개수가 100개이고 예측 클래스 값 유형이 2개(이진 분류)라면 반환 값은 100 x 2 ndarray임. 각 열은 개별 클래스의 예측 확률입니다. 이진 분류에서 첫 번째 칼럼은 0 Negative의 확률, 두 번째 칼럼은 1 Positive의 확률입니다.

- Postive 예측값이 많아지면 재현율 높아짐



분류 결정 임곗값이 낮아질수록 Positive로 예측할 확률이 높아짐. 재현율 증가

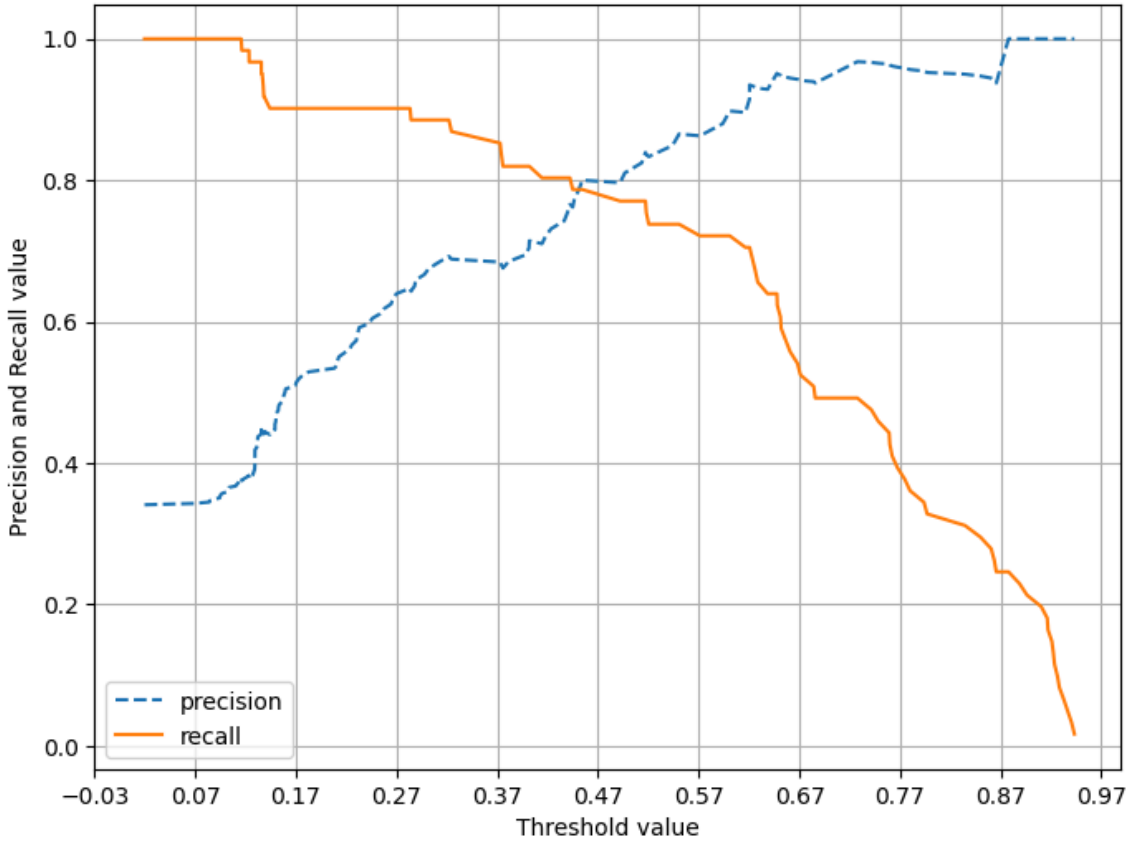
- 재현율을 향상시키면서 다른 수치들을 어느정도 감소시켜야 함 ⇒ 임곗값 0.45가 적당함

평가 지표	분류 결정 임곗값				
	0.4	0.45	0.5	0.55	0.6
정확도	0.8212	0.8547	0.8659	0.8715	0.8771
정밀도	0.7042	0.7869	0.8246	0.8654	0.8980
재현율	0.8197	0.7869	0.7705	0.7377	0.7213

- precision_recall_curve()

입력 파라미터	y_true: 실제 클래스값 배열 (배열 크기= [데이터 건수]) probas_pred: Positive 칼럼의 예측 확률 배열 (배열 크기= [데이터 건수])
반환 값	**정밀도**: 임곗값별 정밀도 값을 배열로 반환 **재현율**: 임곗값별 재현율 값을 배열로 반환

- 임곗값이 낮을수록 많은 수의 양성 예측으로 인해 재현율 값 극도로 상승, 정밀도 값 극도로 감



정밀도 재현율의 맹점

- Positive 예측의 임곗값을 변경함에 따라 정밀도와 재현율의 수치가 변경됨

  ⇒ 정밀도와 재현율을 상호 보완할 수 있는 수준에서 적용돼야 함
- 정밀도 = 100% 방법: 확실한 기준 O ⇒ Positive, 나머지 ⇒ Negative
- 재현율 = 100% 방법: 모든 인자를 Positive로 예측

04 F1 Score

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

- 정의: 정밀도 + 재현율

- 정밀도와 재현율이 어느 한쪽으로 치우치지 않을때 상대적으로 높은 값을 가짐
- F1 score 구하는 API: f1_score()
- 임곗값이 0.6일 때 가장 좋은 값, 하지만 재현율이 크게 감소중..

평가 지표	분류 결정 임곗값				
	0.4	0.45	0.5	0.55	0.6
정확도	0.8212	0.8547	0.8659	0.8715	0.8771
정밀도	0.7042	0.7869	0.8246	0.8654	0.8980
재현율	0.8197	0.7869	0.7705	0.7377	0.7213
F1	0.7576	0.7869	0.7966	0.7965	0.800

05 ROC 곡선과 AUC

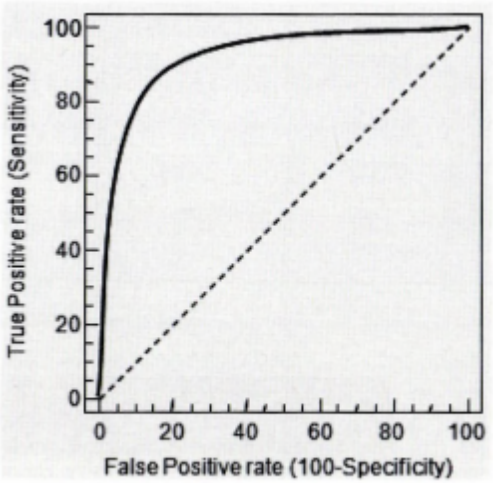
TPR

- True Positive Rate의 약자
- 재현율/민감도를 나타냄; 실제값 positve가 정확이 예측돼야 하는 수준
- TPR = TP / (FN + TP)
- TNR(True Negative Rate)이라고 불리는 특이성(Specificity)이 있음

TNR

- True Negative Rate의 약자
- TNR = TN / (FP + TN)
- FPR = FP / (FP + TN) = 1 - TNR = 1- 특이성

ROC 곡선



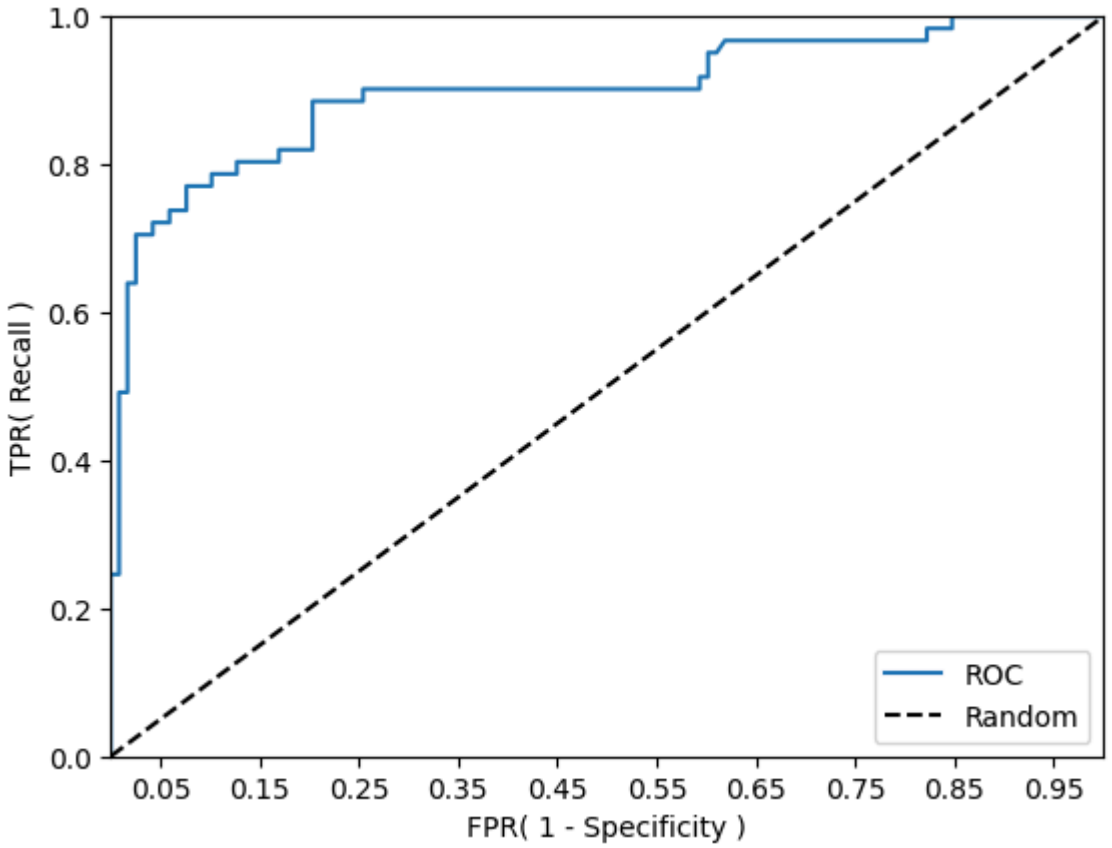
〈ROC 곡선 예시〉

왼쪽 하단과 오른쪽 상단을 대각선으로 이은 직선은 동전을 무작위로 던져 앞/뒤를 맞추는 랜덤 수준의 이진 분류의 ROC 직선

- FPR(False Positive Rate) 이 변할 때 TPR(True Positive Rate)이 어떻게 변하는지를 나타내는 곡선
- FOR를 0부터 1까지 변경하여 TPR의 변화값을 구함
  - 임곗값 = 1 → FP = 0 → FPR = 0
  - 임곗값 = 0 → TN = 0 → FPR = 1
- FPR을 X 축으로, TPR을 Y 축으로 잡으면 FPR의 변화에 따른 TPR의 변화가 곡선 형태로 나타남
- 가운데 직선은 ROC 곡선의 최저 값
- ROC 곡선이 직선에 가까워질수록 성능 저하
- ROC 곡선 구하는 API: roc_curve()

입력 파라미터	**y_true**: 실제 클래스 값 array (array shape = [데이터 건수]) **y_score**: predict_proba( )의 반환 값 array에서 Positive 칼럼의 예측 확률이 보통 사용됨. array, shape = [n_samples]
반환 값	**fpr**: fpr 값을 array로 반환 **tpr**: tpr 값을 array로 반환 **thresholds**: threshold 값 array

AUC



- ROC 곡선 아래의 면적을 구한 것
- 면적이 1에 가까울 수록 좋은 수치
- ROC AUC: 예측 확률값을 기반으로 계산, get_clfLeval(y_test, pred=None, pred_proba=None)로 함수형 변경