



[파머완] 03 평가

[통계](#) [수정](#) [삭제](#)

진규빈 · 9분 전

0

파이썬 머신러닝 완벽 가이드

[▼ 목록 보기](#)

3/3



- 머신러닝 프로세스
 - 데이터 가공/변환
 - 모델 학습/예측
 - 평가(Evaluation)
- 성능 평가 지표는 일반적으로 모델이 분류냐 회귀냐에 따라 여러 종류로 나뉨
- 분류의 성능 평가 지표
 - 정확도(Accuracy)
 - 오차행렬(Confusion Matrix)
 - 정밀도(Precision)
 - 재현율(Recall)
 - F1 스코어
 - ROC AUC
- 분류는 이진 분류(결정 클래스 값 종류의 유형에 따라 긍정/부정과 같은 2개의 결과값만 가짐)와 멀티 분류(여러 개의 결정 클래스 값 가짐)로 나뉨
- 위의 성능 지표는 특히 이진 분류에서 더욱 중요!

(1) 정확도

실제 데이터에서 예측 데이터가 얼마나 같은지 판단하는 지표

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

- 불균형한 레이블 값 분포에서 ML 모델의 성능을 판단할 경우 적합하지 X
- 아무것도 하지 않고 무조건 특정한 결과로만 예측해도 데이터 분포도 균일하지 않은 경우 높은 수치 나타날 수 있음

(2) 오차 행렬

학습된 분류 모델이 예측 수행하면서 얼마나 헛갈리고 있는지도 함께 보여주는 지표
(이진 분류의 예측 오류가 얼마인지 + 어떠한 유형의 예측 오류가 발생하고 있는지 함께 나타내는 지표)

- 정확도의 분류 평가 지표로서의 한계점 극복 위해 함께 적용
- True/False, Positive/Negative의 4분면으로 구성
- TN, FP, FN, TP => 예측 클래스의 실제 클래스의 Positive 결정 값(값 1)과 Negative 결정 값(값 0)의 결합에 따라 결정

		예측 클래스 (Predicted Class)	
		Negative(0)	Positive(1)
실제 클래스 (Actual Class)	Negative(0)	TN (True Negative)	FP (False Positive)
	Positive(1)	FN (False Negative)	TP (True Positive)

- TN는 예측값을 Negative 값 0으로 예측했고 실제 값 역시 Negative 값 0
 - FP는 예측값을 Positive 값 1로 예측했는데 실제 값은 Negative 값 0
 - FN은 예측값을 Negative값 0으로 예측했는데 실제 값은 Positive 값 1
 - TP는 예측값을 Positive값 1로 예측했는데 실제 값 역시 Positive 값 1
- `confusion_matrix()` : 사이킷런이 오차 행렬 구하기 위해 제공하는 API
 - 인자: 예측 결과, 실제 결과
 - 출력된 오차 행렬은 ndarray 형태
 - 정확도는 오차 행렬에서 True에 해당하는 값인 TN과 TP에 좌우됨
 - 정확도
 - = (예측 결과와 실제 값이 동일한 건수) / (전체 데이터 수)
 - = $(TN + TP) / (TN + FP + FN + TP)$
 - 불균형한 이진 분류 데이터 세트에서는 Positive 데이터 건수가 매우 작기 때문에 데이터에 기반한 ML 알고리즘은 Positive보다는 Negative로 예측 정확도 높아지는 경향 발생
 - Negative로 예측할 때 정확도 높기 때문에 FN이 매우 작고, Positive로 예측하는 경우 작기 때문에 FP 역시 매우 작아짐

∴ 정확도 지표 => 비대칭한 데이터 세트에서 Positive에 대한 예측 정확도 판단 못한 채 Negative에 대한 예측 정확도만으로도 분류 정확도 매우 높게 나타나는 수치적 판단 오류 발생

(3) 정밀도와 재현율

- Positive 데이터 세트의 예측 성능에 좀 더 초점 맞춘 평가 지표
- 불균형한 데이터 세트에서 정확도보다 더 선호됨

<정밀도>

- $TP / (FP + TP)$
- 예측을 Positive로 한 대상 중 예측과 실제 값이 Positive로 일치한 데이터의 비율
- Positive 예측 성능을 더 정밀하게 측정하기 위한 평가 지표

- 양성 예측도라고도 불림
- '실제 Negative 음성 데이터 예측을 Positive로 잘못 판단하면 업무상 큰 영향이 발생하는 경우'에 중요 지표

<재현율>

- $TP / (FN + TP)$
- 실제 값이 Positive인 대상 중 예측과 실제 값이 Positive로 일치한 데이터의 비율
- 민감도(Sensitivity) 또는 TPR(True Positive Rate)라고도 불림
- '실제 Positive 양성 데이터 예측을 Negative로 잘못 판단하면 업무상 큰 영향이 발생하는 경우'에 중요 지표

- 재현율과 정밀도 모두 TP 높이는 데 초점 맞추는 건 동일
- 재현율은 FN을, 정밀도는 FP를 낮추는 데 초점 맞춘다는 차이 존재
- 서로 보완적인 지표로 분류의 성능 평가하는 데 적용되며, 둘 다 높은 수치 얻는 것이 가장 좋은 성능 평가

- `precision_score()` : 사이킷런이 정밀도 계산 위해 제공하는 API
- `recall_score()` : 사이킷런이 재현율 계산 위해 제공하는 API

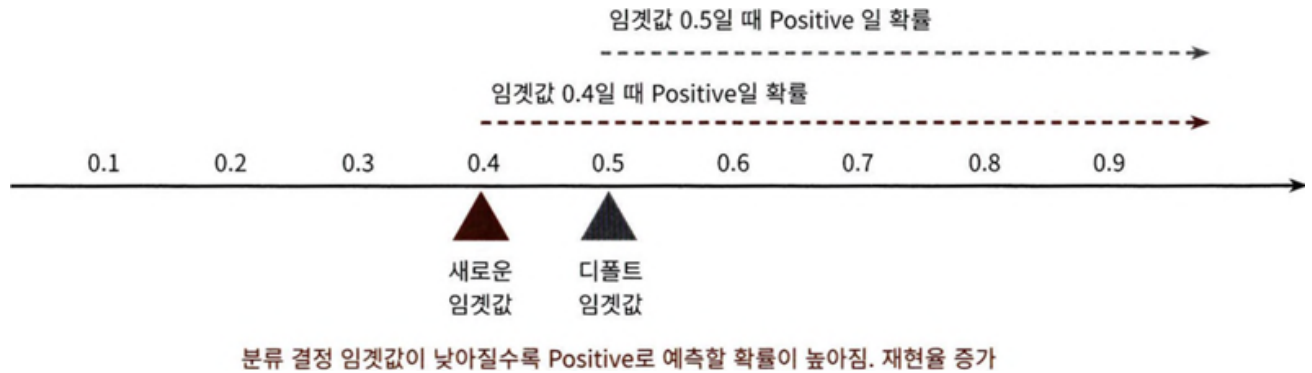
정밀도/재현율의 트레이드오프(Trade-off)

- 정밀도와 재현율은 상호 보완적인 평가 지표이기 때문에 어느 한 쪽 강제로 높이면 다른 하나의 수치 떨어지기 쉬움
- 사이킷런의 분류 알고리즘 => 예측 데이터가 특정 레이블에 속하는지 계산하기 위해 먼저 개별 레이블별 결정 확률 구함
- `predict_proba()` : 개별 데이터별로 예측 확률 반환하는 사이킷런 메서드

입력 파라미터	<code>predict()</code> 메서드와 동일하게 보통 테스트 피쳐 데이터 세트를 입력
반환 값	<p>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.</p> <p>각 열은 개별 클래스의 예측 확률입니다. 이진 분류에서 첫 번째 칼럼은 0 Negative의 확률, 두 번째 칼럼은 1 Positive의 확률입니다.</p>

- `predict()` : `predict_proba()` 메서드에 기반해 생성된 API
- `predict_proba()` 호출 결과로 반환된 배열에서 분류 결정 임계값보다 큰 값이 들어 있는 칼럼의 위치를 받아 최종적으로 예측 클래스 결정

- 사이킷런의 Binarizer 클래스 이용
- 생성된 Binarizer 객체의 fit_transform() 메서드 이용해 넘파이 ndarray 입력
- 입력된 ndarray의 값이 threshold 값보다 같거나 작으면 0을, 크면 1을 반환
- 분류 결정 임계값 => Positive 예측값 결정하는 확률의 기준 (임계값 낮출수록 True 값 많아짐)
- Positive 예측값 많아지면 상대적으로 재현율 값 높아짐 => 양성 예측 많이 하다 보니 실제 양성을 음성으로 예측하는 횟수가 상대적으로 줄어들기 때문



- precision_recall_curve(): 임계값 변화에 따른 평가 지표 값 알아보는 과정과 유사한 사이킷런 API
- 임계값 증가할수록 정밀도 값 동시에 높아지나 재현율 값은 낮아짐
- 정밀도와 재현율의 임계값에 따른 값 변화를 곡선 형태의 그래프로 시각화하는 데 이용 가능

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: 임계값별 정밀도 값을 배열로 반환
	재현율: 임계값별 재현율 값을 배열로 반환

정밀도와 재현율의 맹점

- Positive 예측의 임계값 변경은 업무 환경에 맞게 정밀도 & 재현율 수치를 상호 보완할 수 있는 수준에서 적용돼야 함!

<극단적인 수치 조작>

- 정밀도 100% 되는 방법
 - 확실한 기준 되는 경우만 Positive로, 나머지는 모두 Negative로 예측
 - FP는 0, TP는 1 되므로 정밀도 = $1 / (1 + 0) = 100\%$

- 재현율 100% 되는 방법
 - 모든 환자를 Positive로 예측

∴ 정밀도 & 재현율 수치 적절하게 조합되어 분류의 종합적 성능 평가에 사용될 수 있는 평가 지표 필요

(4) F1 스코어

정밀도 & 재현율을 결합한 지표

- 정밀도 & 재현율이 어느 한 쪽으로 치우치치 않는 수치 나타낼 때 상대적으로 높은 값 가짐

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

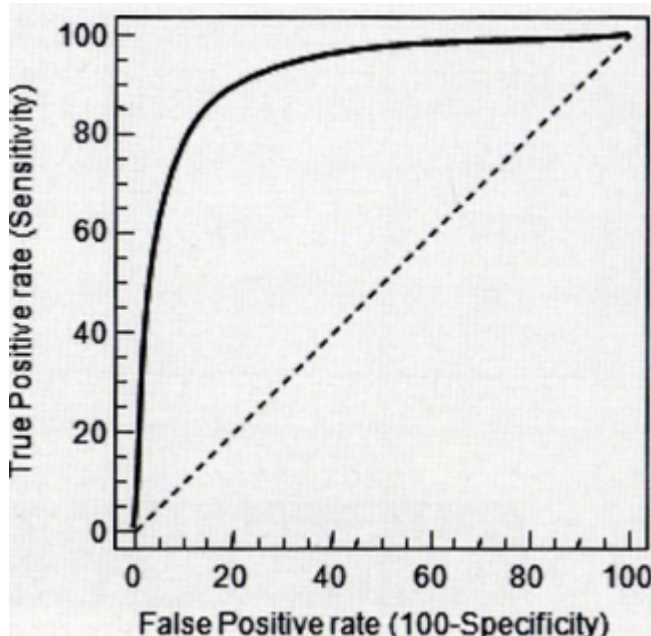
- `f1_score()` : F1 스코어 구하는 사이킷런 API

(5) ROC 곡선과 AUC

이진 분류의 예측 성능 측정에서 중요하게 사용되는 지표

<ROC 곡선>

- Receiver Operation Characteristic Curve = 수신자 판단 곡선
- **FFR(False Positive Rate)** 변할 때 **TPR(True Positive Rate)**이 어떻게 변하는지 나타내는 곡선
- FPR을 X축으로, TPR을 Y축으로 잡으면 FPR의 변화에 따른 TPR의 변화가 곡선 형태로 나타남



- 가운데 직선 => ROC 곡선의 최저 값
- 왼쪽 하단과 오른쪽 상단을 대각선으로 이은 직선 => 랜덤 수준의 이진 분류 ROC 직선 (AUC는 0.5)
- 가운데 직선에 가까울수록 성능 떨어지고 멀어질수록 성능 뛰어난 것
- FPR을 0부터 1까지 변경하면서 TPR의 변화 값 구함
 - FPR을 0으로: 분류 결정 임계값 1로 지정하면 됨
=> Positive 예측 기준 매우 높아 분류기가 임계값보다 높은 확률 가진 데이터를 Positive로 예측할 수 없기 때문
 - FPR을 1로: 분류 결정 임계값을 0으로 지정해 TN을 0으로 만들면 됨
=> 분류기의 Positive 확률 기준이 너무 낮아 다 Positive로 예측하고 아예 Negative 예측이 없기 때문

- **TPR(True Positive Rate)**

= 재현율 = $TP / (FN + TP)$ = 민감도

= 실제값 Positive(양성)가 정확히 예측돼야 하는 수준

- **TNR(True Negative Rate)**

= 특이성(Specificity) = $TN / (FP + TN)$

= 실제값 Negative(음성)가 정확히 예측돼야 하는 수준

= 민감도에 대응하는 지표

- **FPR(False Positive Rate)**

= $FP / (FP + TN)$

= $(1 - TNR)$ 또는 $(1 - \text{특이성})$ 으로 표현

- `roc_curve()` : ROC 곡선 구하는 사이킷런 API
 - `precision_recall_curve()` 와 사용법 유사

- 단지 반환값이 FPR, TPR, 임계값으로 구성돼 있을 뿐

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

일반적으로 ROC 곡선 자체는 FPR & TPR의 변화 값 보는 데 이용, 분류의 성능 지표로 사용되는 것은 **ROC 곡선 면적에 기반한 AUC 값**

<AUC(Area Under Curve)>

- ROC 곡선 밑의 면적 구한 것
- 일반적으로 1에 가까울수록 좋음
- AUC 수치 커지려면 FPR이 작은 상태에서 얼마나 큰 TPR 얻을 수 있느냐가 관건
- 가운데 직선에서 멀어지고 왼쪽 상단 모서리 쪽으로 가파르게 곡선이 이동할수록 직사각형에 가까운 곡선이 되어 면적이 1에 가까워짐
- 보통의 분류는 0.5 이상의 AUC 값 가짐



진규빈



이전 포스트

[파머완] 02 사이킷런으로 시작하는 머신러닝

0개의 댓글