3. 평가

goblurry · 방금 전

데이터분석 머신러닝 파완머

파이썬 머신러닝 완벽가이드

목록 보기 3/3

📊 파이썬 머신러닝 완벽 가이드 (위키북스, 개정 2판) 교재를 바탕으로 학습한 내용을 정리한 포스트입니다.
https://github.com/goblurry/ML-Study 에서 예시 코드를 확인할 수 있습니다.

모델 평가

머신러닝은 데이터를 **전처리 → 모델 학습 → 예측 → 평가**하는 과정을 거친다. 이 중 평가(Evaluation) 단계는 학습된 모델이 얼마나 잘 작동하는지를 확인하는 핵심 과정이다.
단순히 정확도 하나만으로는 모델의 성능을 충분히 설명할 수 없다. 특히 데이터가 불균형할 경우, 정확도만 높게 나오고 실제로는 제대로 분류하지 못하는 문제가 발생할 수 있다. 따라서 다양한 평가 지표(Evaluation Metric) 를 종합적으로 살펴야 한다.

분류의 성능 평가 지표로는

- 정확도 (Accuracy)
- 오차행렬 (Confusion Matrix)
- 정밀도 (Precision)
- 재현율 (Reacall)
- F1 스코어
- ROC AUC
  가 있다.

분류는 결정 클래스값 종류 유형에 따라 2개의 결과만을 가지는 **이진 분류**와 여러 개의 결정 클래스값을 가지는 **멀티 분류**로 나뉜다. 위의 성능 지표는 이 둘 모두에 적용되지만, 이진 분류에서 더욱 중요하게 강조된다.

정확도

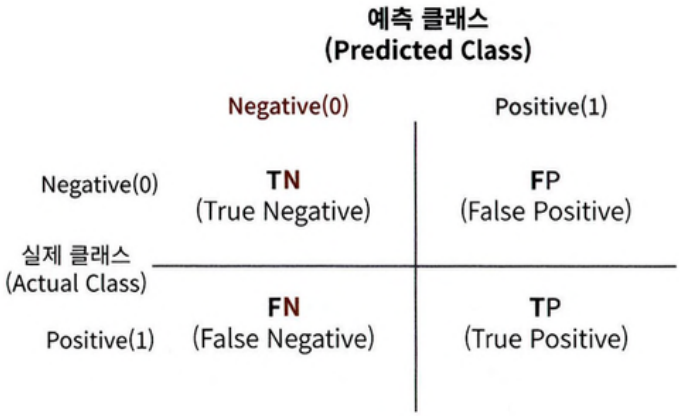
- 정의: 전체 데이터 중에서 모델이 맞춘 비율
- **Accuracy = 예측 결과가 동일한 데이터 건수 / 전체 예측 데이터 건수**
- 이해하기 쉽고 직관적인 지표지만, 불균형 데이터셋에서는 왜곡될 가능성이 있음.

```
from sklearn.metrics import accuracy_score

# accuracy_score(y_true, y_pred)
# y_true : 실제 레이블
# y_pred : 예측 레이블
accuracy = accuracy_score(y_test, predictions)
```

오차 행렬

- 정의:분류 결과를 실제 클래스와 비교하여 정답/오답을 유형별로 정리한 표
- 네 가지 값으로 구성된다:
  1. TP (True Positive): 실제 1, 예측도 1
  2. TN (True Negative): 실제 0, 예측도 0
  3. FP (False Positive): 실제 0인데 1로 잘못 예측 → Type I Error
  4. FN (False Negative): 실제 1인데 0으로 잘못 예측 → Type II Error



- 오차행렬을 통해 단순 정확도보다 **어떤 오류가 발생했는지**까지 파악할 수 있다.
- 특히 FP와 FN의 차이는 상황에 따라 중요도가 크게 달라진다.
  - 예: 질병 진단 → FN(환자를 건강으로 잘못 분류) 위험이 더 크다.
    - 예: 스팸 필터링 → FP(정상 메일을 스팸으로 분류) 불편함이 크다.

```
from sklearn.metrics import confusion_matrix

# confusion_matrix(y_true, y_pred)
cm = confusion_matrix(y_test, predictions)
print(cm)
```

정밀도와 재현율

정밀도와 재현율은 Pos. 데이터 세트의 예측 성능에 조금 더 초점을 맞춘 평가 지표이다.

정밀도 = TP / (FP+TP)

- 모델이 Positive라고 예측한 것 중 실제로 Positive인 비율
- 적용 예시: 스팸 메일 필터링 → 스팸(Positive)이라고 분류된 메일이 실제로 스팸인지 확인하는 것이 중요

재현율 = TP / (FN+TP)

- 실제 Positive 중에서 모델이 Positive로 제대로 맞춘 비율
- 적용 예시: 암 진단 검사 → 환자(Positive)를 최대한 놓치지 않는 것이 중요

두 지표의 관계: Trade-off

정밀도와 재현율은 Threshold 조정에 따라 균형이 달라진다.

- 임곗값을 낮추면 Positive로 분류하는 경우가 늘어나 Recall ↑, Precision ↓
- 임곗값을 높이면 Positive 판정이 줄어들어 Precision ↑, Recall ↓

즉, 두 지표 사이에는 **상충관계**가 존재한다.
Recall을 중시하는 경우: 의료 진단, 범죄 탐지, 부정 거래 탐지처럼 놓치면 안 되는 Positive가 중요한 상황.
다소 오탐(FP)이 있어도 FN을 최소화하는 것이 목표.
Precision을 중시하는 경우: 스팸 메일 분류, 광고 추천처럼 Positive 판정이 틀리면 사용자 불편이 큰 상황.
FP를 줄여 모델의 신뢰도를 높이는 것이 목표.

정밀도와 재현율은 보통 동시에 높이기 어렵다. 핵심은 분류기의 임곗값 조정에 있다. 대부분의 사이킷런 분류 모델은 predict() 메서드 외에도 `predict_proba()` 메서드를 제공한다. predict_proba() 를 활용하면 임곗값을 직접 조정해 Precision과 Recall의 균형을 원하는 방향으로 맞출 수 있다!

정밀도와 재현율의 맹점

하지만 정밀도와 재현율에도 한계가 있다.
1. 편향적 해석 위험

- Precision만 강조하면 Recall이 떨어지고, Recall만 강조하면 Precision이 떨어지는 구조적 한계가 있음.

2. 단일 지표로는 균형을 판단하기 어려움

- Precision과 Recall은 항상 함께 보아야 의미가 있음.
- 하나만 높게 나오더라도 다른 하나가 극도로 낮으면 모델은 실질적으로 무의미해질 수 있다.

3. 불균형 데이터셋에서 오해 가능

- Positive 비율이 극도로 낮을 경우, Recall은 높게 나오지만 Precision은 거의 0에 수렴할 수 있음. 반대로, Precision은 높지만 Recall은 극도로 낮을 수 있음.

👉 그래서 두 지표의 균형을 보기 위해 F1-score 같은 보완 지표가 필요하다!

F1 Score

앞서 말했듯, 정밀도(Precision)와 재현율(Recall)은 서로 상충하는 관계다. 따라서 한쪽만 높이고 다른 쪽이 낮아지면 모델 성능을 제대로 평가하기 어렵다. 이때 두 지표를 결합한 지표가 F1 스코어다. 이는 정밀도와 재현율이 균형을 이룰 때 상대적으로 높은 값을 가진다.

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

정의는 위와 같다. (정밀도와 재현율의 조화평균)

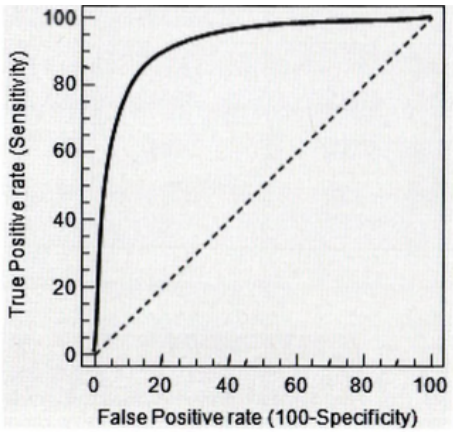
사이킷런의 metrics 모듈에서 제공하는 `f1_score()` 메서드는 정밀도와 재현율의 조화 평균인 F1 스코어를 계산한다.

```
from sklearn.metrics import f1_score
f1 = f1_score(y_test, pred)
...
```

ROC 곡선과 AUC

ROC 곡선 (Receiver Operating Characteristic Curve)

- 수신자 판단 곡선: 머신러닝의 **이진 분류 모델**의 예측 성능을 판단하는 중요한 평가 지표
- FPR(False Positive Rate)이 변할 때, TPR(True Positive Rate, 재현율)이 어떻게 변하는지 나타내는 곡선. 전자를 X축, 후자를 Y축으로 잡으면, 상대적인 변화가 곡선 형태로 나타난다.



〈 ROC 곡선 예시 〉

- 가운데 직선: 랜덤 수준의 이진 분류의 ROC 직선.
- ROC 곡선이 가운데 직선에 가까울수록 성능이 떨어지는 것이고, 멀어질수록 뛰어난 것이다.

FPR(False Positive Rate: FP/(FP+TN))을 0부터 1까지 변경하면서 TPR의 변화 값을 구한다. 변경은 **분류 결정 임곗값**을 변경시킴으로써 가능하다.

분류 결정 임곗값: Positive 예측값을 결정하는 확률의 기준.

- 1로 설정하면 FPR을 0으로 만들 수 있음. (임곗값을 1로 정하면 pos 예측 기준이 높아서 분류기가 임곗값보다 확률 높은 데이터를 pos로 예측하지 못함)
- 반대로 0으로 설정하면 FPR을 1로 만들 수 있음. (TN을 0으로 만든다. 분류기의 pos 확률 기준이 너무 낮아서 다 pos로 예측함 - negative 에측 없어서 tn이 0이 됨)

이렇게 임곗값을 0~1 사이에서 변화시키면서 FPR를 구하고, 이 값의 변화에 따른 TPR 값을 구하는 게 ROC 곡선!

사이킷런은 이 곡선을 구하기 위해 `roc_curve()` API를 제공함.

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 (Area Under Curve)

- ROC 곡선 아래 면적
- 값의 범위: 0.5 ~ 1
  - 0.5: 랜덤 추측과 동일
  - 1.0: 완벽한 분류
- FPR이 작은 상태에서 얼마나 큰 TPR을 얻냐에 따라 AUC 수치가 결정됨.
- `roc_auc_score` 메서드를 이용해 성능을 수치화할 수 있다.

정리

1. ROC 곡선: TPR(FPR에 따른 변화)을 시각적으로 나타낸 곡선
2. AUC: ROC 곡선 아래 면적, 수치적으로 모델 성능을 요약
3. API: roc_curve() (곡선 데이터 반환), roc_auc_score() (단일 성능 수치 반환)

📍 3장 총정리

분류 모델의 성능 평가는 단순 정확도(Accuracy)만으로는 부족하다. 특히 데이터가 불균형할 경우, 정확도는 모델 성능을 왜곡할 수 있다.

> **오차 행렬**(Confusion Matrix): 예측값과 실제값을 조합해 TN, FP, FN, TP로 구성된 행렬. 다양한 성능 지표의 기초가 된다.
> **정밀도**(Precision)와 **재현율**(Recall): Positive 예측 성능을 평가하는 핵심 지표.
> **F1-score**: 정밀도와 재현율의 조화 평균. 불균형 데이터에서 성능을 공정하게 평가.
> **ROC-AUC**: 분류 성능을 시각화하고 비교하는 대표 지표. AUC 값은 1에 가까울수록 좋은 모델을 의미한다.

goblurry

이전 포스트
2. 사이킷런으로 시작하는 머신러닝

0개의 댓글

댓글을 작성하세요

댓글 작성