3장 평가

Ch	3
날짜	@2025년 9월 15일
카테고리	개념 정리

3.1 정확도 (Accuracy)

개념

- 실제 데이터에서 예측 데이터가 얼마나 같은지 판단 지표
- **정확도(Accuracy)** = 예측 결과가 실제 레이블과 동일한 데이터 건수 ÷ 전체 데이터 건수

$$Accuracy = \frac{\text{예측이 맞은 건수}}{\text{전체 데이터 건수}}$$

특징

- 직관적이고 간단한 지표
- 하지만 **이진 분류에서 데이터 분포가 불균형(imbalanced)** 하면 성능을 왜곡할 수 있음
- 따라서 정확도 단독으로 성능을 평가하면 위험

Ex 1: Titanic 데이터

- 성별만으로 단순히 예(여성=생존, 남성=사망)해도 → 정확도가 **78.77%**
- 복잡한 모델과 유사한 수준 → **정확도만으로는 모델 성능을 제대로 반영하지 X**

Ex 2: MNIST 데이터 (불균형 변형)

- 숫자 7만 `1(True)`로, 나머지는 `0(False)`로 변환 → 90%가 0, 10%가 1
- 모든 데이터를 0으로 예측하는 **Dummy Classifier**도 정확도 = **90%**
- 의미 없는 예측임에도 불구하고 높은 정확도

⚠️정확도의 한계⚠️

- 불균형 데이터에서 정확도가 높게 나올 수 있음
- ex) 레이블이 90%가 0, 10%가 1일 때, 모든 예측을 0으로 하면 정확도는 90%
- 따라서 반드시 **오차 행렬(Confusion Matrix), 정밀도(Precision), 재현율(Recall), F1 score, ROC/AUC**와 함께 사용해야 함

3.2 오차 행렬 (confusion matrix, 혼동행렬)

개념

- 분류 모델이 **예측하면서 얼마나 헷갈리는지(confused)**를 보여주는 지표
- 단순 정확도(Accuracy)보다 **예측 오류의 유형**을 알 수 있음
- **실제값(Actual Class)** 과 **예측값(Predicted Class)** 을 비교하여 4가지 경우로 나뉨

실제 \ 예측 클래스	Negative (0)	Positive (1)
Negative (0)	**TN (True Negative)**예측=0, 실제=0	**FP (False Positive)**예측=1, 실제=0

실제 \ 예측 클래스	Negative (0)	Positive (1)
Positive (1)	**FN (False Negative)**예측=0, 실제=1	**TP (True Positive)**예측=1, 실제=1

TN / FP / FN / TP 의미

- **TN (True Negative)** : 0으로 예측했고 실제도 0
- **FP (False Positive)** : 1로 예측했지만 실제는 0 (→ Type I Error)
- **FN (False Negative)** : 0으로 예측했지만 실제는 1 (→ Type II Error)
- **TP (True Positive)** : 1로 예측했고 실제도 1

```
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, fakepred)
```

```
array([[405,   0],
       [ 45,   0]])
```

정확도와의 관계

- 정확도(Accuracy)는 오차 행렬로 다음처럼 정의 가능
- 정확도 = 예측 결과와 실제 값이 동일한 건수 / 전체 데이터 수

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- 문제점
  : 데이터가 **불균형(imbalanced)** 한 경우, 대부분 Negative로만 예측해도 정확도가 높게 나타날 수 있음

  →ex) 사기 탐지, 암 검진처럼 Positive가 매우 적은 데이터셋에서 **정밀도(Precision), 재현율(Recall)** 이 더 중요한 경우

3.3 정밀도와 재현율

개념 정리

지표	공식	의미	다른 이름
정밀도 (Precision)	TP / (TP + FP)	Positive로 예측한 것 중 실제 Positive 비율	양성 예측도
재현율 (Recall)	TP / (TP + FN)	실제 Positive 중 올바르게 예측한 비율	민감도(Sensitivity), TPR(True Positive Rate)

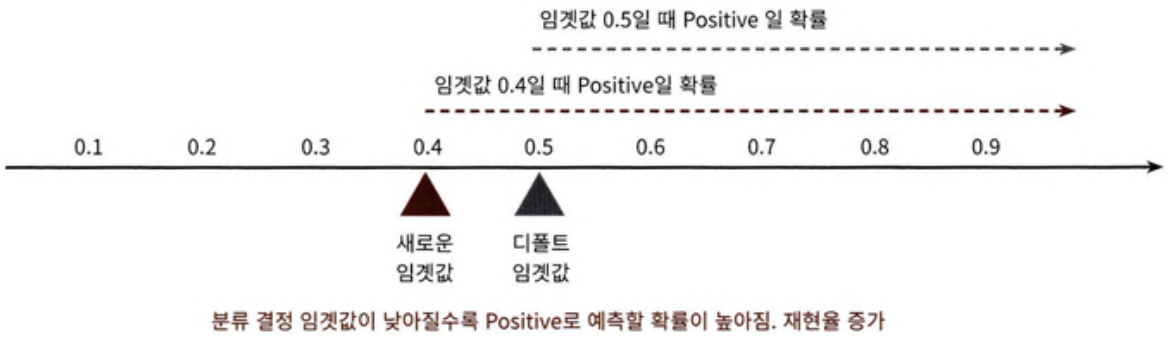
중요성

- **재현율**은 Positive(양성) 놓치면 안 되는 상황에서 중요
- **정밀도**는 Negative(음성)를 잘못 Positive로 분류하면 문제될 때 중요

상황	더 중요한 지표	이유
암 판별 모델	재현율 ↑	암 환자를 놓치면 치명적
보험/금융 사기 탐지	재현율 ↑	사기를 놓치면 회사 손실 큼
스팸메일 필터링	정밀도 ↑	정상 메일이 스팸으로 분류되면 큰 불편 초래

정밀도/재현율 trade-off

- ***임곗값(Threshold)*을 조절**하면서 두 지표가 상반 관계를 가짐.
- 임곗값 ↓ → Positive 더 많이 예측 → 재현율 ↑, 정밀도 ↓
- 임곗값 ↑ → Positive 엄격히 예측 → 정밀도 ↑, 재현율 ↓



분류 결정 임곗값이 낮아질수록 Positive로 예측할 확률이 높아짐. 재현율 증가

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

정밀도/재현율의 맹점

구분	설명	문제점
정밀도 (Precision)	양성으로 예측한 것 중 실제 양성 비율	- 양성 판단 자체를 적게 하면 정밀도 ↑ (예: 극히 확실한 경우만 양성으로 예측)
재현율(Recall)	실제 양성 중에서 양성으로 맞춘 비율	- 양성 판단을 마구 하면 재현율 ↑ (예: 거의 전부 양성으로 예측)
공통 맹점	두 지표는 서로 trade-off 관계	- 단독으로 모델 성능 평가 시 한쪽으로 치우칠 수 있음 - 예: 정밀도만 높게 맞추면 실제 양성을 많이 놓칠 수 있음 (재현율↓), 반대로 재현율만 높이면 잘못된 양성 예측이 많아짐 (정밀도↓)

scikit-learn 함수	설명
`precision_score(y_test, pred)`	정밀도 계산
`recall_score(y_test, pred)`	재현율 계산
`predict_proba(X)`	각 클래스 예측 확률 반환
`Binarizer(threshold=값)`	확률 기반 임곗값 적용
`precision_recall_curve(y_true, probas_pred)`	임곗값별 정밀도·재현율 반환

3.4 F1 score

개념

- **F1 스코어**는 **정밀도(Precision)와 재현율(Recall)을 결합**한 지표
- 두 지표가 **균형 있게 유지될 때** 값이 높음

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

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

예시 비교

모델	Precision	Recall	F1 Score	특징
A 모델	0.9	0.1	0.18	한쪽으로 치우쳐서 낮음
B 모델	0.5	0.5	0.50	균형 → 더 좋은 성능

```
from sklearn.metrics import f1_score

f1 = f1_score(y_test, pred)
print('F1 스코어: {0:.4f}'.format(f1))

# 출력
# F1 스코어: 0.7966
```

3.5 ROC 곡선, AUC

ROC Curve 개념

- **이진 분류**의 예측 성능 측정 - 지표
- **ROC (Receiver Operating Characteristic) Curve 수산자 판단 곡선**
  - 원래 제2차 세계대전 당시 **통신 장비 성능 평가**를 위해 개발
  - 현재는 **의학·머신러닝 이진 분류 모델 성능 평가**에 널리 사용
- **축 정의**
  - X축: **FPR (False Positive Rate)**
  - Y축: **TPR (True Positive Rate, 재현율/민감도)**

- **민감도 (Sensitivity, TPR)**
  → 실제 Positive 중 올바르게 Positive로 예측한 비율

$$\text{TPR (Sensitivity, Recall)} = \frac{TP}{TP + FN}$$

- **특이성 (Specificity, TNR)**
  → 실제 Negative 중 올바르게 Negative로 예측한 비율

$$\text{TNR (Specificity)} = \frac{TN}{TN + FP}$$

- **FPR (False Positive Rate)**

$$\text{FPR} = \frac{FP}{FP + TN} = 1 - \text{TNR} = 1 - \text{Specificity}$$

$$0.5 \leq \text{AUC} \leq 1$$

ROC Curve 해석

- **임곗값 조정**에 따라 FPR, TPR 값이 변함
  - 임곗값 ↑ → Positive 예측이 줄어들어 FPR ↓, TPR ↓
  - 임곗값 ↓ → Positive 예측이 늘어나 FPR ↑, TPR ↑
- **가운데 대각선(랜덤 분류, AUC=0.5)**: 동전 던지기 수준
- **왼쪽 상단 모서리에 가까운 곡선(AUC→1)**: 좋은 분류 성능

AUC (Area Under Curve)

$$0.5 \leq \text{AUC} \leq 1$$

- ROC 곡선 밑의 면적
- 범위: **0.5 ~ 1**
  - **0.5** → 랜덤 수준
  - **1.0** → 완벽한 분류기
- 일반적으로 **0.8 이상**이면 좋은 성능
- **0.9 이상**이면 매우 우수

```
from sklearn.metrics import roc_curve, roc_auc_score

# ROC Curve
fprs, tprs, thresholds = roc_curve(y_test, pred_proba_class1)

# AUC Score
roc_score = roc_auc_score(y_test, pred_proba_class1)
print("ROC AUC 값: {:.4f}".format(roc_score))
```

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