〈데이터분석과기계학습 7주차〉

모델 평가와 하이퍼파라미터 튜닝

인공지능융합공학부 데이터사이언스전공

곽찬희

전반학기에는

- 퍼셉트론과 Adaline
  - ✓ 학습률, SGD/GD/MBGD
  - ✓ 비용함수와 최적화
- 분류모델
  - ✓ 로지스틱 회귀, SVM, DT, k-NN
- 데이터 전처리
  - ✓ 누락데이터 처리
  - ✓ 범주형데이터 코딩
  - ✓ 스케일 맞추기 (표준화, 정규화)
  - ✓ 특성선택
- 차원축소
  - ✓ PCA, LDA

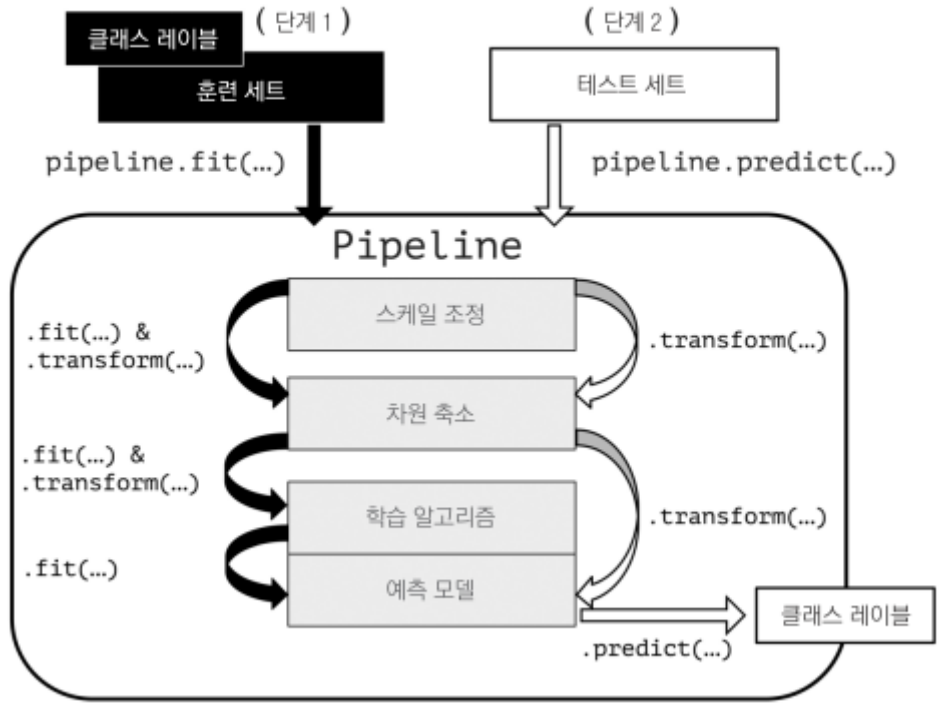
오늘 배울 내용

- 편향되지 않은 모델 성능 추정
- 머신 러닝 알고리즘에서 일반적으로 발생하는 문제 분석
- 머신 러닝 모델 세부 튜닝
- 여러 성능 지표를 사용해 모델의 예측 성능 평가

6.1. 파이프라인을 사용한 워크플로

- sklearn 의 pipeline 클래스
  - ✓ Transformation, feature extraction/selection, ML algorithm 등을 파이프형태로 연결하여 관리가 용이
  - ✓ make_pipeline 함수를 사용
  - ✓ 일종의 블록 조립?

그림 6-1 사이킷런의 파이프라인 작동 방식



6.2. 교차 검증 (Cross Validation)

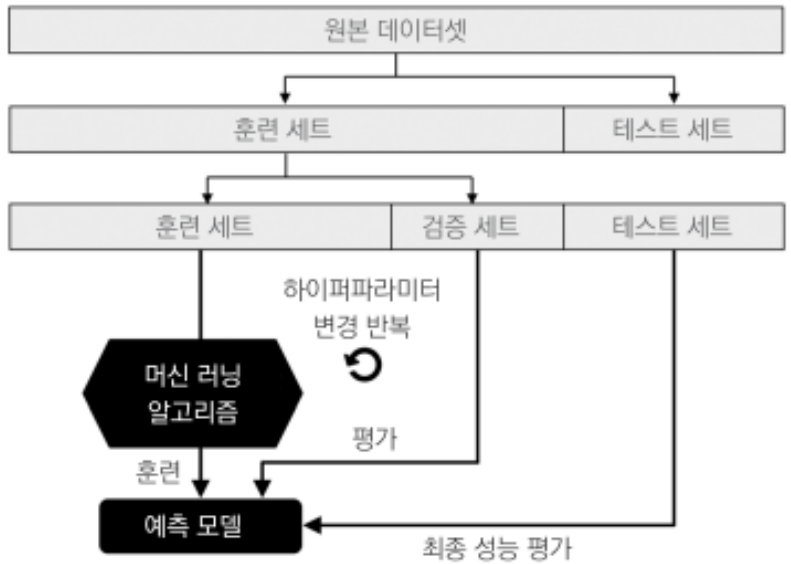
- 과대적합과 과소적합을 피하기 위해, 훈련데이터를 나눠서 학습에 사용
- 적절한 편향-분산 트레이드오프를 찾기 위해 교차 검증 기법을 활용함 (cross validation)
  - ✓ 홀드아웃 교차 검증 (holdout cross-validation)
  - ✓ K-fold 교차 검증(k-fold cross validation)

6.2.1. 홀드아웃 교차검증

- 데이터를 훈련, 검증, 테스트 세트로 나눠 사용
  - ✓ 훈련: 모델 훈련용
  - ✓ 검증: 모델 선택/평가용 (튜닝, 일종의 내부 테스트)
  - ✓ 테스트: 모델 성능 평가용 (일반화 성능 추정)

- 검증 세트를 이용해 반복적으로 다른 파라미터 값에 훈련을 진행한 후 성능을 평가

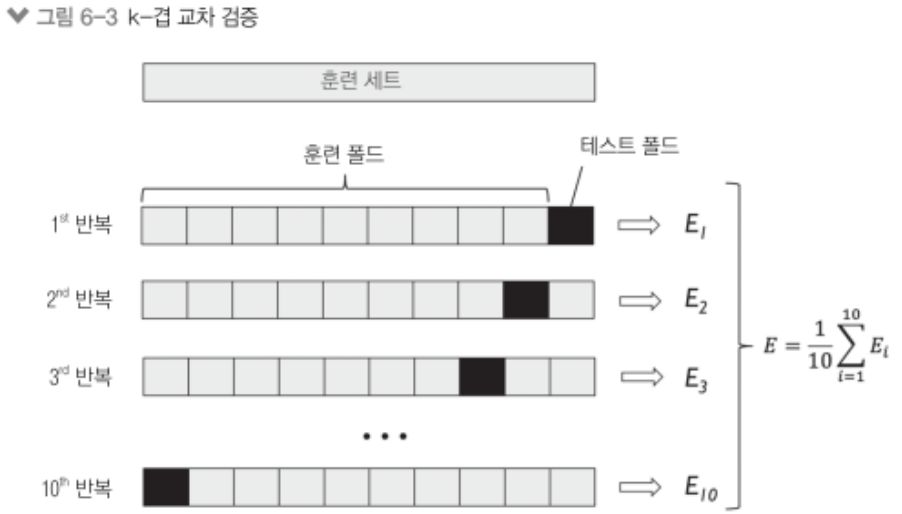
그림 6-2 홀드아웃 교차 검증



- 홀드아웃의 단점은, 훈련/검증 세트를 나누는 방법에 따라 성능 추정이 민감해질 수 있다는 것

6.2.2.k-fold Cross Validation

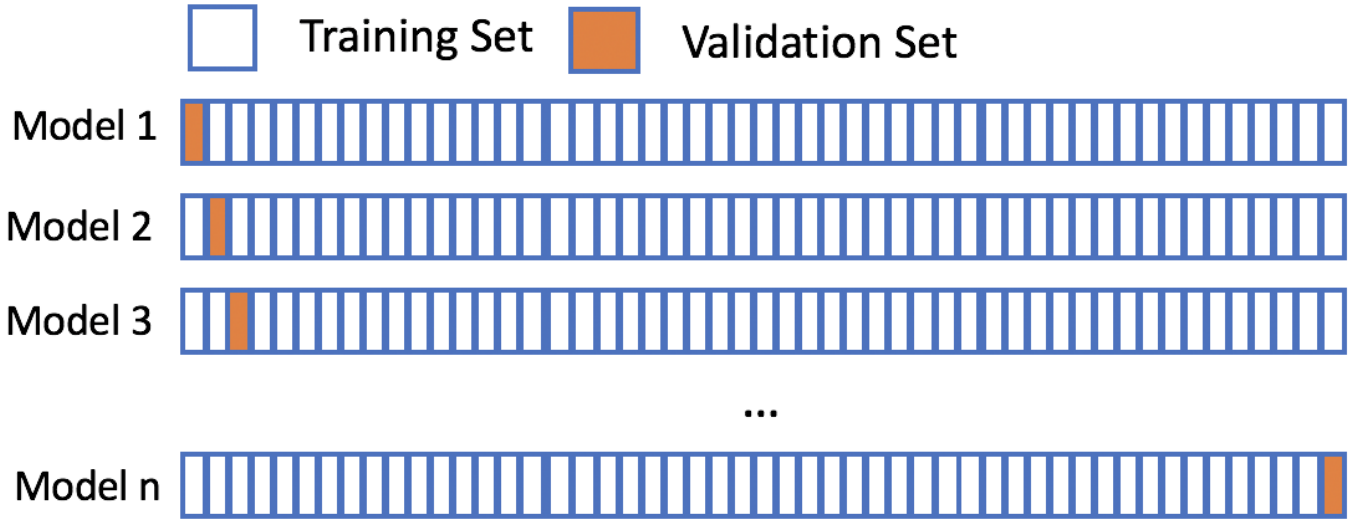
- K-fold CV
  - ✓ 데이터셋을 k개의 폴드로 랜덤하게 나눔
  - ✓ k-1개의 폴드로 모델 훈련, 1개의 폴드로 성능 평가
  - ✓ 위 과정을 k번 반복 후, 모델의 평균 성능 계산
  - ✓ 최적의 하이퍼파라미터를 찾은 뒤, 전체 훈련 세트를 활용해 모델 학습 시행



- K-fold CV는 중복을 허용하지 않는 리샘플링 기법이기에, 모든 샘플 포인트가 훈련하는 동안 테스트 폴드 검증에 단 1번 사용되는 장점이 있음
  - ✓ 이로 인해 홀드아웃보다 모델 성능 추정 분산이 낮음
- K는 정해진 것은 없으나 보통 10을 사용
  - ✓ 너무 큰 k는 실행시간을 늘리고, 분산이 높은 추정을 만듦/ 너무 작은 k는 ?

6.2.2.k-fold Cross Validation

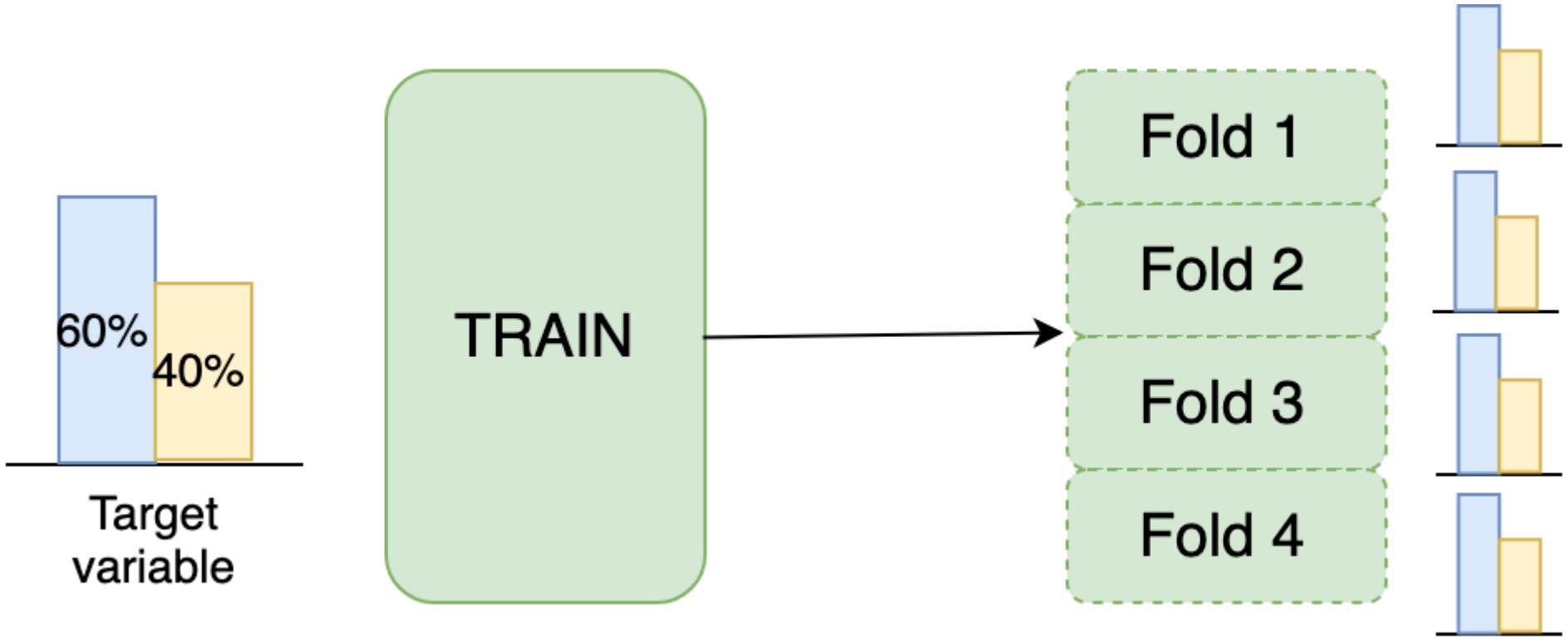
- LOOCV(Leave-One-Out Cross Validation)
  - ✓ 훈련 샘플의 개수와 k 가 같을 때.
  - ✓ 아주 작은 데이터셋을 사용할 때



https://blog.kakaocdn.net/dn/bpUvKx/btqv8ZEu3IC/r5FBV8ptTJWOWnlX6ShXzk/img.png

6.2.2.k-fold Cross Validation

- **Stratified k-fold cross validation (층화 k-fold CV)**
  - ✓ 클래스 비율을 고려하여 폴드를 구성하는 방법



https://assets.datacamp.com/production/repositories/4443/datasets/a09316fc1a59a42af7ad0893dc6ee83da2b908d7/my_stratified_kfold.png

6.3.학습 곡선과 검증 곡선을 사용한 알고리즘 디버깅

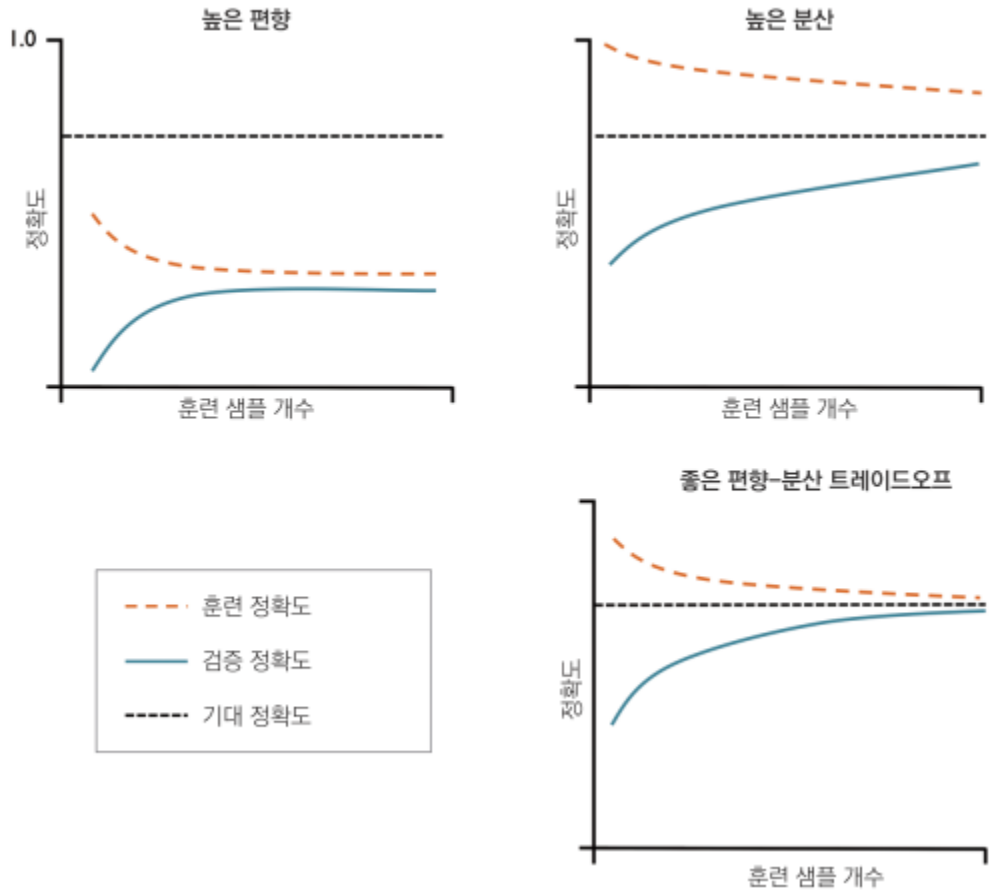
- 과대적합
  - ✓ 데이터셋에 비해 모델이 너무 복잡
  - ✓ 모델의 자유도가 큼
  - ✓ 모델 파라미터가 너무 많음

- 과대적합을 줄이려면
  - ✓ 데이터를 더 모으면 좋지만… (비싸고, 불가능하고, …)

- 그래서!
  - ✓ 훈련 세트의 크기 함수로 훈련 정확도와 검증 정확도를 그려보면서, 데이터를 어떻게 활용하는 것이 좋을지 판단

6.3.1. 학습 곡선으로 편향과 분산 문제 분석

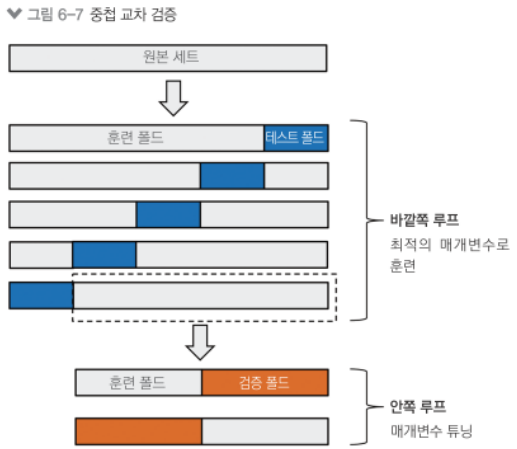
그림 6-4 편향-분산 트레이드오프



6.4. 그리드 서치 튜닝

- 머신러닝의 두 파라미터
  - ✓ 훈련 데이터에서 학습되는 파라미터 (ex. 회귀식의 가중치)
  - ✓ 최적화되는 학습의 파라미터 (하이퍼파라미터)

- 최적의 하이퍼파라미터를 찾는 법은?
  - ✓ 다 해보면 돼요 -> gridsearchCV
  - ✓ 다 해보기엔 너무 많아요 -> RandomizedSearchCV
  - ✓ Grid search + k-fold 를 섞은 더 강력한 방법 -> nested cross validation



6.5.여러 가지 성능 평가 지표

- **오차행렬**
  - ✓ **예측과 실제 클래스의 비교를 위한 표**
    - **진짜 양성 (True Positive, TP): 양성 예측, 실제 양성**
    - **진짜 음성 (True Negative, TN): 음성 예측, 실제 음성**
    - **거짓 양성 (False Positive, FP): 양성 예측, 실제 음성**
    - **거짓 음성 (False Negative, FN): 음성 예측, 실제 양성**

  - ✓ **오차행렬로 계산하는 다양한 지표**
    - **정확도: 전체 예측 중 맞은 예측의 비율**

❯ 그림 6-8 오차 행렬

실제 클래스 \ 예측 클래스	N	P
N	진짜 음성 (TN)	거짓 양성 (FP)
P	거짓 음성 (FN)	진짜 양성 (TP)

$$ACC = \frac{TP + TN}{FP + FN + TP + TN} = 1 - ERR = 1 - \frac{FP + FN}{FP + FN + TP + TN}$$

6.5.여러 가지 성능 평가 지표

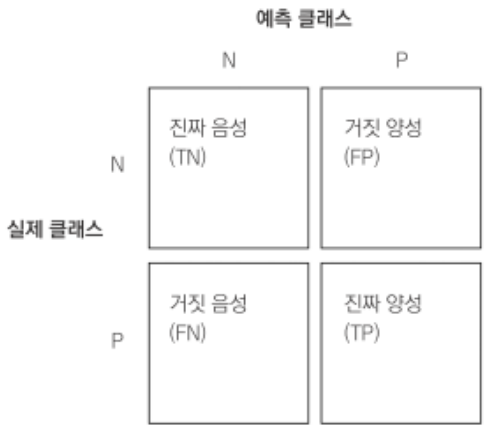
✓ 진양성비율 (True Positive Rate, TRP): 실제 양성 중 양성으로 예측한 비율

$$TPR = \frac{TP}{P} = \frac{TP}{TP + FN}$$

✓ 위양성비율 (False Positive Rate, FPR): 실제 음성인 것 중 양성으로 예측한 비율

$$FPR = \frac{FP}{N} = \frac{FP}{FP + TN}$$

❤ 그림 6-8 오차 행렬



6.5.여러 가지 성능 평가 지표

✓ 정확도 (precision): 양성 예측 중 실제 양성 비율

$$PRE = \frac{TP}{TP + FP}$$

✓ 재현율(recall): 실제 양성 중 양성으로 예측한 비율

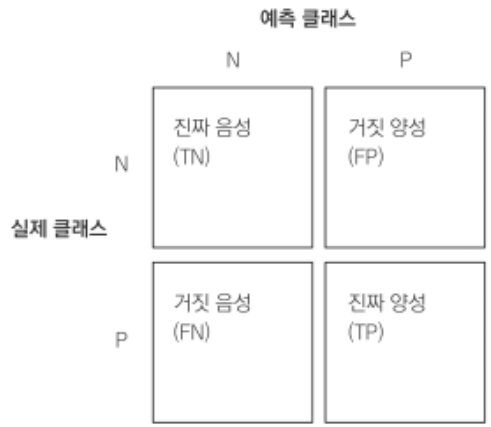
$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN + TP}$$

✓ PRE 와 REC을 조화평균을 구한 F1 Score 도 자주 사용

$$F1 = 2\frac{PRE \times REC}{PRE + REC}$$

✓ sklearn 에는 위의 대부분이 구현되어 있어서 따로 계산할 필요는 없음

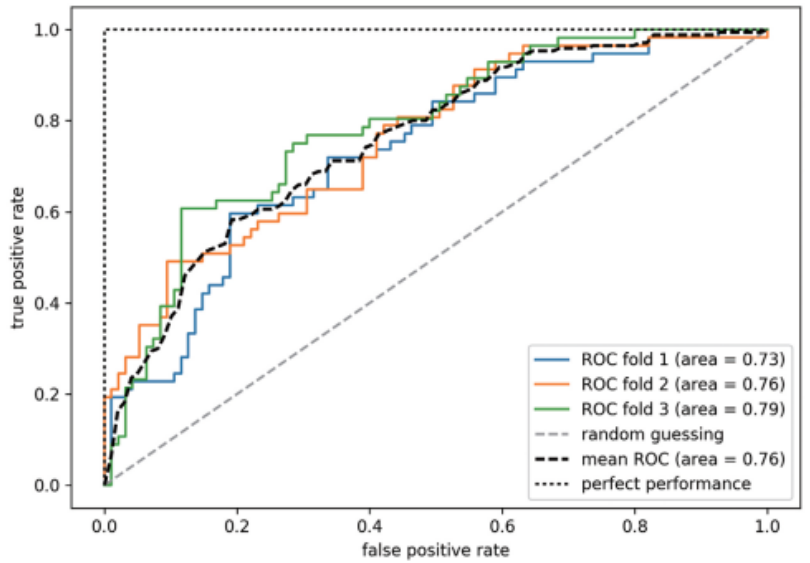
그림 6-8 오차 행렬



6.5.3. ROC 곡선 그리기

- RoC (Receiver Operating Characteristics)
  - ✓ FPR 과 TPR 점수를 기반으로 그래프를 그림
  - ✓ 왼쪽 위로 갈 수록 TPR이 커짐
  - ✓ RoC 아래의 면적인 AUC(Area Under the Curve) 로 모델의 성능을 종합할 수 있음

그림 6-10 ROC 곡선



6.5.4. 다중 분류의 성능 지표

- 클래스가 여러 개일 때 OvA(One-versus-All)방법으로 지표를 계산
- k개의 클래스가 있는 경우 정확도의 마이크로 평균은 다음과 같이 계산

$$PRE_{micro} = \frac{TP_1 + \cdots + TP_i}{(TP_1 + \cdots + TP_i) + (FP_1 + \cdots + FP_i)}$$

- 매크로 평균은 단순한 클래스별 정확도의 평균

$$PRE_{macro} = \frac{PRE_1 + \cdots + PRE_k}{k}$$

6.6. 불균형 클래스

- 클래스의 비율이 극단적으로 불균형한 경우가 생각보다 자주 발생
  - ✓ 스팸 필터링, 오류 감지, 질병 탐지 등등
  - ✓ 만약 0/1 의 비율이 1:9 라면, 어떤 알고리즘을 쓰더라도 평균적으로 90%의 성능을 보일 것

- 불균형 클래스를 해결하는 몇 가지 방법
  - ✓ weight 를 이용: 적은 쪽의 클래스를 맞추는 것에 더 점수를 주는 방식
  - ✓ Under-sampling: 많은 쪽의 클래스 샘플을 적게 뽑는 방식
  - ✓ Over-sampling: 적은 쪽의 샘플을 중복으로 추출하는 방식
  - ✓ Synthetic Sampling: 적은 쪽의 데이터와 비슷한, 가상의 샘플을 만드는 방식