# Ch6. 모델 평가와 하이퍼파라미터 튜닝

1기 김지원
2기 박영은
2기 양지선
2기 장현정

6.1 파이프라인을 사용한 효율적인 워크플로

6.2 k-겹 교차 검증을 사용한 모델 성능 평가

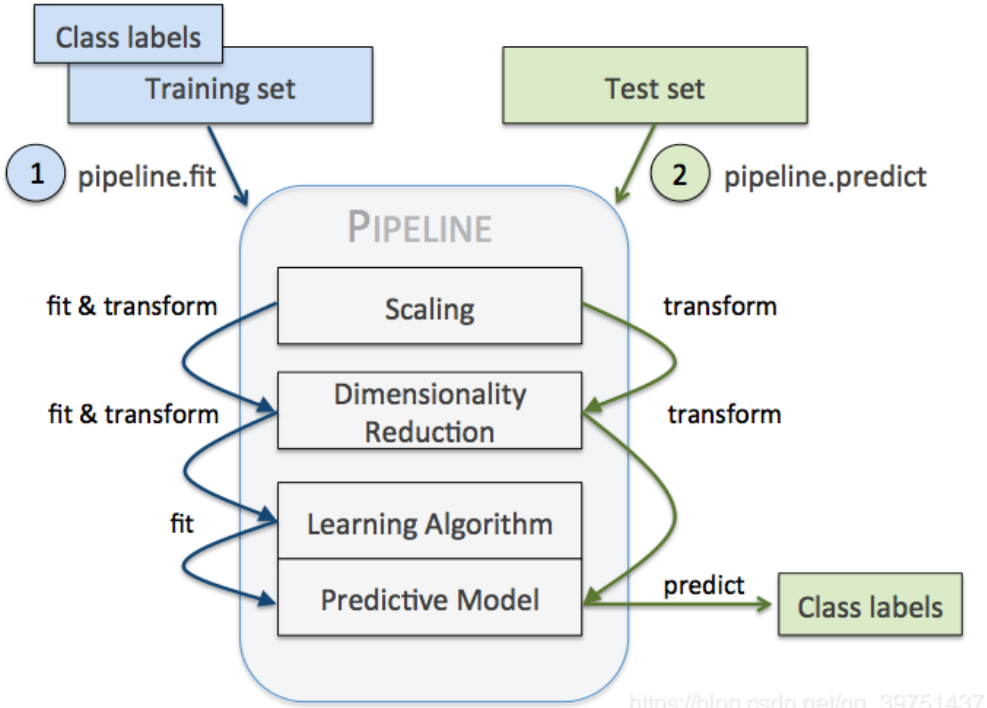
6.3 학습곡선과 검증 곡선을 사용한 알고리즘 디버깅

6.4 그리드 서치를 사용한 머신 러닝 모델 세부 튜닝

6.5 여러가지 성능 평가 지표

6.6 불균형한 클래스 다루기

# 사이킷런, Pipeline 클래스




훈련세트와 테스트 세트를 각각 학습하고 변환하는 단계를 구성하는 대신 객체들을 하나의 파이프라인으로 연결

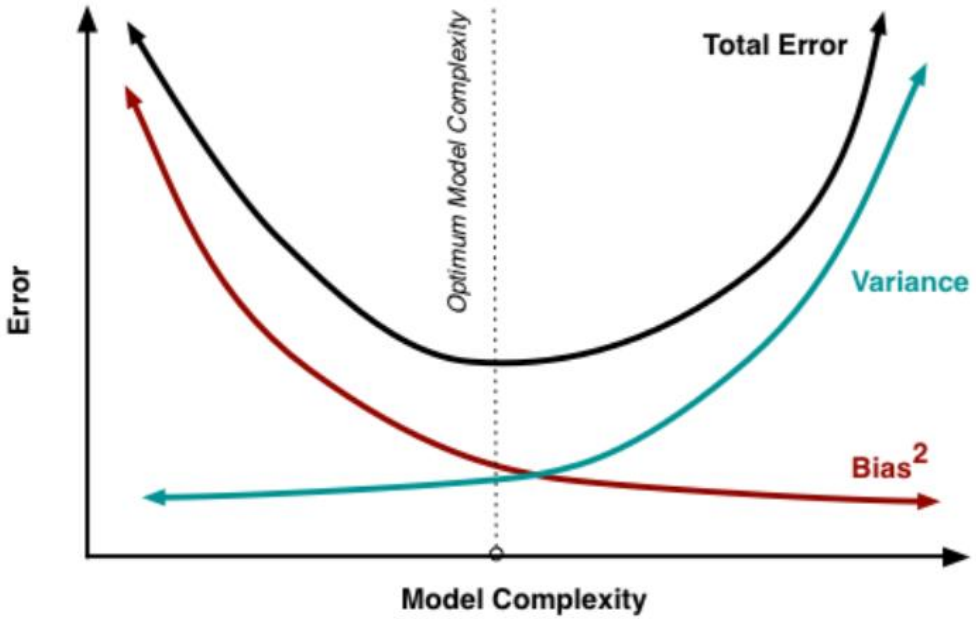
# 교차 검증으로 모델 성능 평가하기

너무 간단한 모델→ 과소적합(높은 편향)의 문제

너무 복잡한 모델→ 과대적합(높은 분산)의 문제

따라서 **적절한 편향-분산 트레이드 오프**가 필요.

# 편향-분산 트레이드 오프




편향, 분산이 최소화한 모델이 가장 좋다. 하지만 편향과 분산을 동시에 최소화할 수 없다.

Error를 최소화 하기 위해 편향과 분산의 합이 최소가 되는 적당한 지점을 찾는다.

# 모델의 일반화 성능 추정을 위한 교차 검증 기법

1. 홀드아웃 교차 검증
2. k 겹 교차 검증

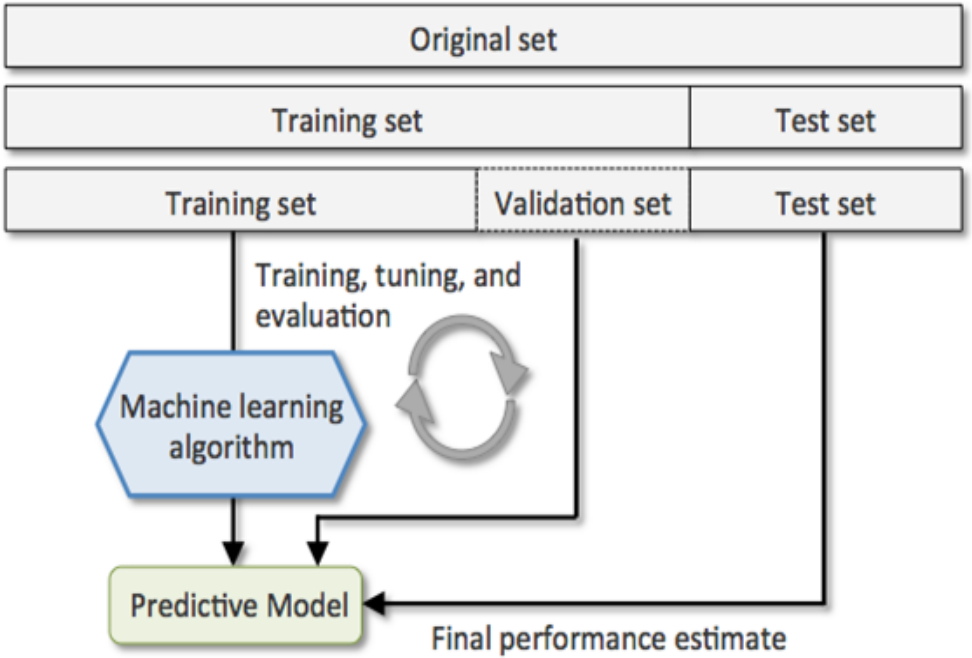
교차 검증 기법으로 처음 본 데이터에 모델이 얼마나 잘 동작하는지 추정한다.

# 홀드아웃 교차 검증

전통적이고 널리 사용되는 머신러닝 모델, 일반화 성능 추정방법

장점: 새로운 데이터에 대한 일반화 능력을 덜 편향되게 추정할 수 있다.

단점:  훈련 데이터의 훈련 세트와 검증 세트 나누는 방법에 따라 성능 추정 민감할 수도 있다.



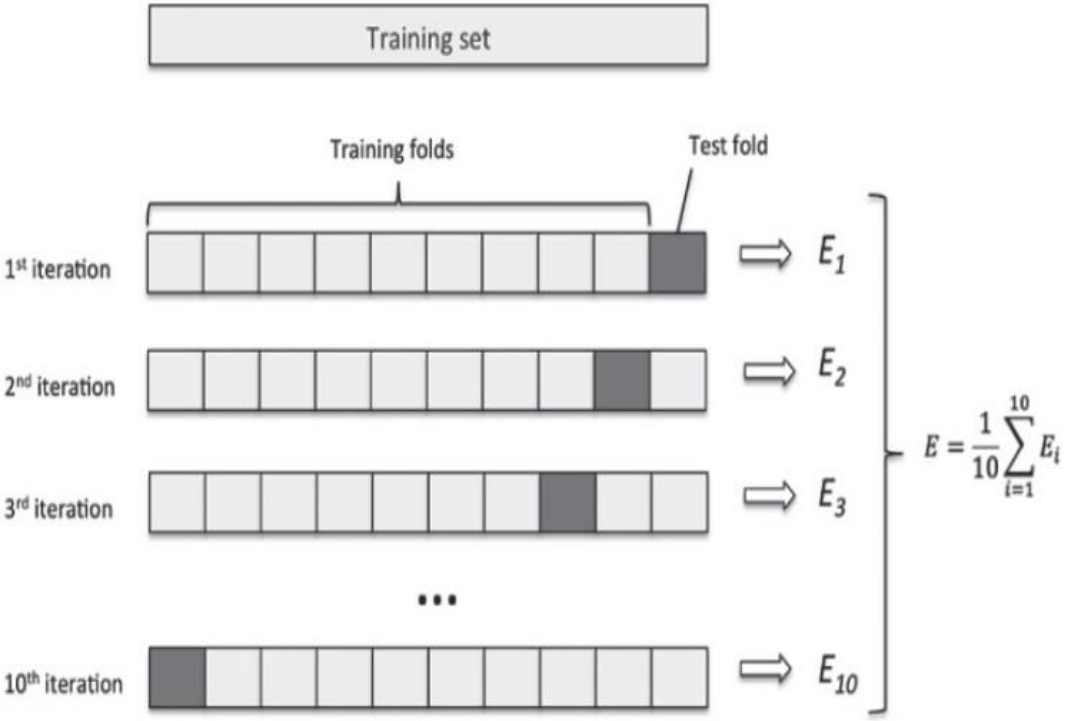

1. 데이터를 훈련세트, 검증세트, 테스트 세트로 나눈다.
2. 훈련세트로 여러 가지 모델 훈련
3. 검증세트로 예측모델 평가
4. 테스트세트에서 모델의 일반화 성능 추정(새로운 데이터에 모델이 잘 작동하는지)

# K-겹 교차 검증

1. 중복을 허락하지 않고 훈련 데이터셋을 k개의 폴드로 랜덤하게 나눈다.
2. k-1개의 폴드로 모델을 훈련하고 나머지 하나의 폴드로 성능 평가
3. k번을 반복하여 k개의 모델과 성능 추정을 얻기
4. 3번에서 얻은 각 폴드의 성능 추정 기반으로 모델의 평균 성능계산
5. 전체 훈련세트로 모델 다시 훈련
6. 독립적 테스트 세트로 최종 성능 추정

모든 샘플 포인트가 훈련하는 동안 검증에 딱 한 번 사용되어
홀드아웃 방법보다 모델 성능 추정에 분산이 낮다.

# 10-겹 교차 검증




훈련데이터를 10개의 폴드로 나누고 10번의 반복 동안 9개의 폴드는 훈련에, 1개의 폴드는 모델평가에 사용.

각 폴드의 추정성능을 사용하여 평균 성능 E 계산.

# 계층적 k-겹 교차 검증

기본 k-겹 교차 검증 방법보다 좀 더 향상된 방법

각 폴드에서 클래스 비율이 전체 훈련 세트에 있는 클래스 비율을 대표하도록 유지

→ 클래스 비율이 동등하지 않을 때 좀 더 나은 편향과 분산 추정을 만든다.

# 학습곡선과 검증곡선

학습 곡선, 검증 곡선 → 학습 알고리즘 성능 향상

모델 복잡(자유도, 파라미터 많으면) → 과대적합 → 일반화 x

Sol) 훈련 샘플을 더 모으기 but 실전에서는 불가능할 때 많음

⇒ 학습곡선으로 편향과 분산 문제 분석

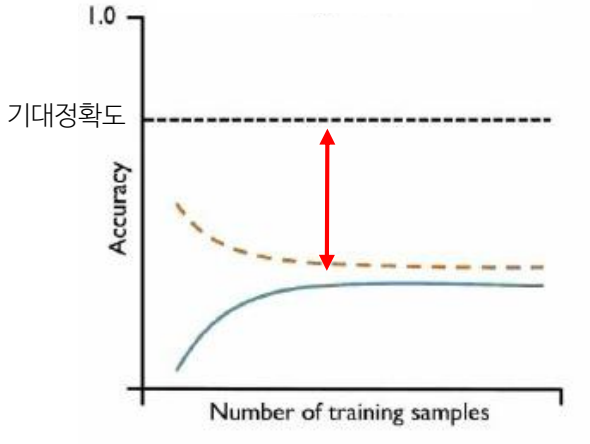
★ 학습 곡선: 모델의 훈련 정확도와 검증 정확도를 훈련 세트의 크기 함수로 그래프 그리기

→ 높은 분산 문제인지 높은 편향 문제인지 감지

→ 샘플 모으는게 문제 해결을 할지 판단 가능

# 학습곡선

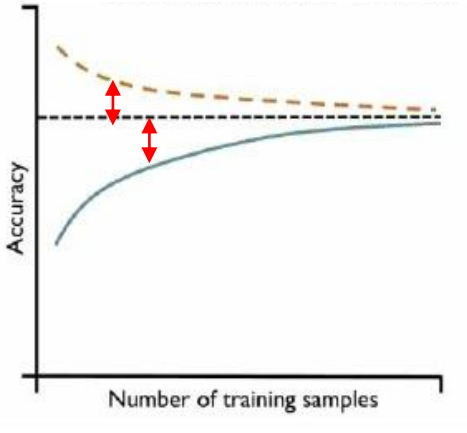
높은 편향



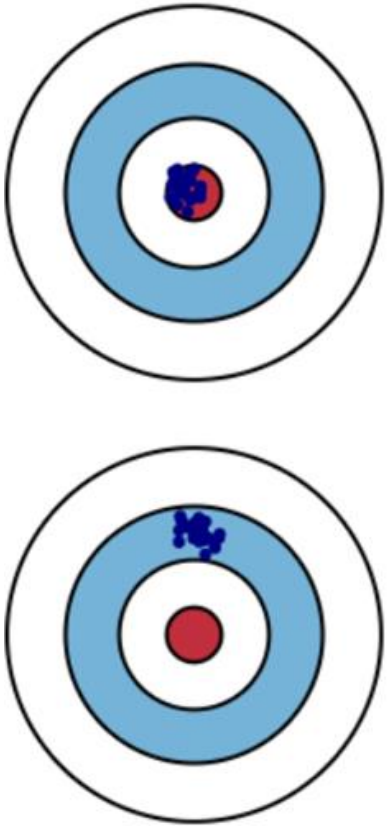

→ 과소적합

Sol) 파라미터 개수 증가,
규제 강도 감소

좋은 편향-분산 trade-off

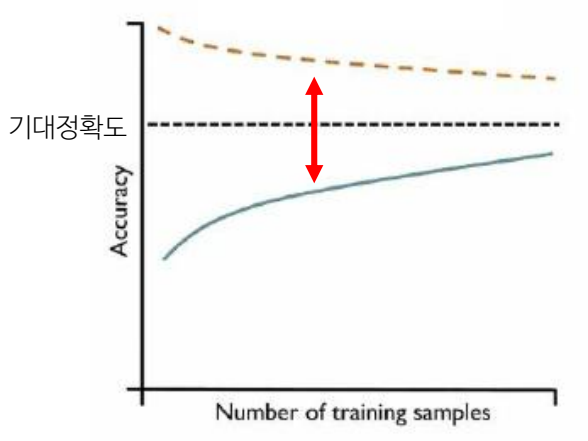




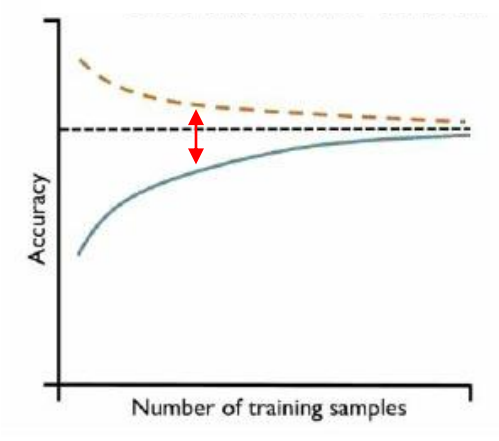


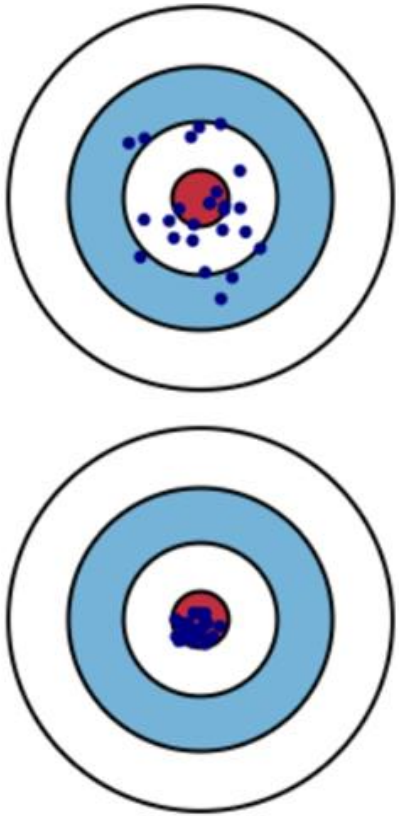
# 학습곡선

높은 분산




좋은 편향-분산 trade-off



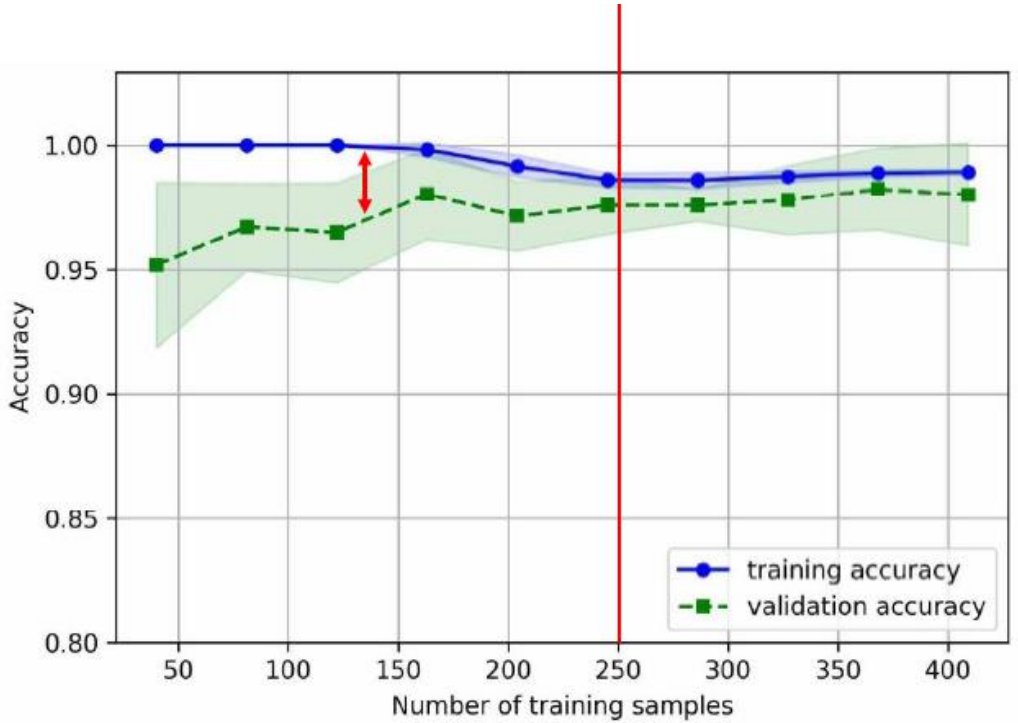



→ 과대적합

Sol1) 많은 훈련 데이터, 복잡도 감소, 규제 증가

Sol2) 규제 없는 모델의 경우, 특성 선택, 특성 추출을 통한 특성 개수 감소

# 학습곡선




샘플 250개 이상: 잘 작동

샘플 250개 이하:

훈련 정확도, 검증 정확도 간의 차이 심해짐

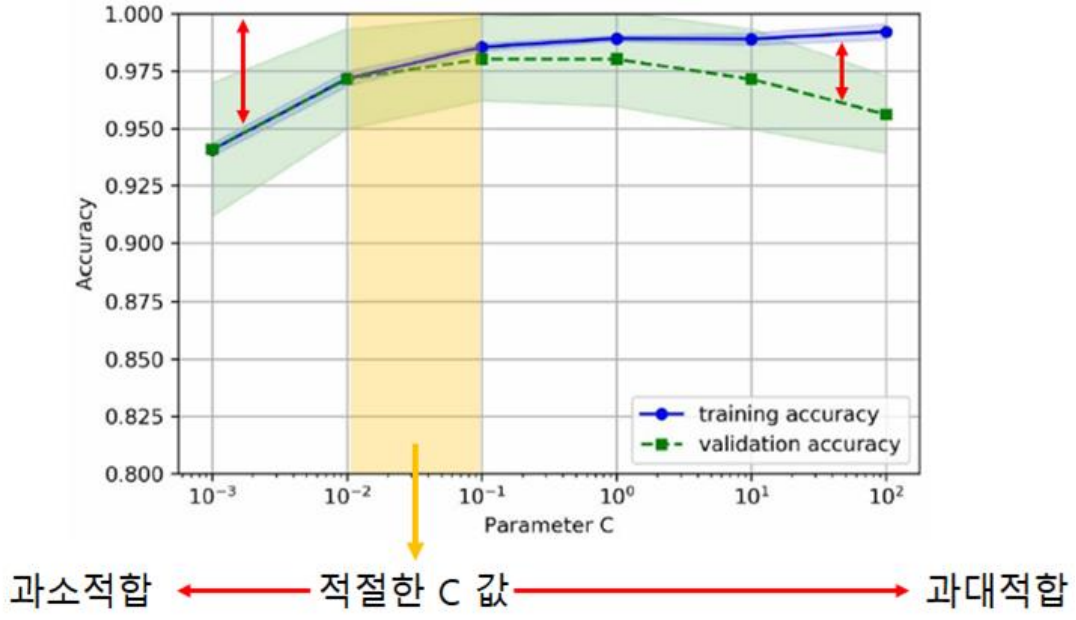
→ 훈련 데이터에 과대적합

# 검증곡선

★ 검증 곡선: 모델의 훈련 정확도와 검증 정확도를 **모델 파라미터 값**의 함수로 그래프 그리기

- 훈련 데이터에 잡음 많거나, 모델이 이미 최적화된 경우에도 다룰 수 있음

Ex. 로지스틱 회귀의 규제 매개변수 C



# 하이퍼파라미터

**파라미터의 종류**

| 훈련 데이터에서 학습되는 파라미터 | **하이퍼파라미터** (모델의 튜닝 파라미터) |
|---|---|
| 훈련 데이터에 따라 자동으로 정해지는 매개변수 | 별도로 최적화되는 학습알고리즘의 파라미터 |
| 사용자가 변경 불가 | 사용자가 직접 설정 가능 |
| | 정해진 최적의 값이 없어<br>적합한 하이퍼파라미터를 튜닝을 통해 찾아야함 |
| 로지스틱 회귀의 가중치 | 로지스틱 회귀의 규제 매개변수(C값),<br>결정트리의 깊이 매개변수, KNN의 K값 |

# 그리드 서치

리스트로 저장된 여러가지 하이퍼파라미터값 전체를 모두 조사(**사용자가 파라미터값 지정**)

→ 모든 조합에 대해 모델 성능 평가

→ 최적의 조합 찾기

Sklearn.model_selection모듈의 GridSearchCV 클래스 사용

튜닝하려는 하이퍼파라미터를 딕셔너리의 리스트로 지정.

Ex. 선형 SVM의 경우 svc_C / RBF 커널 SVM의 경우 svc_C, svc_gamma

최상의 모델점수와 이 모델의 매개변수 확인. 테스트 세트를 사용하여 최고 모델의 성능 추정.

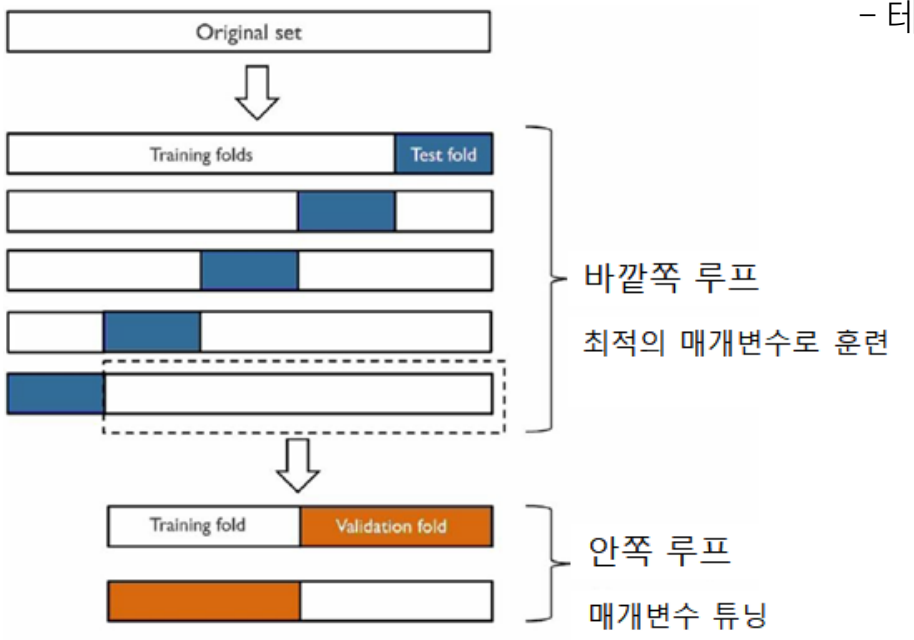
모든 조합 평가하여 계산 비용 매우 큼

→ **랜덤서치(randomized search)** : 제한된 횟수 안에서 샘플링 분포로부터 랜덤한 매개변수 조합을 뽑음

# 중첩 교차 검증

5x2 교차 검증

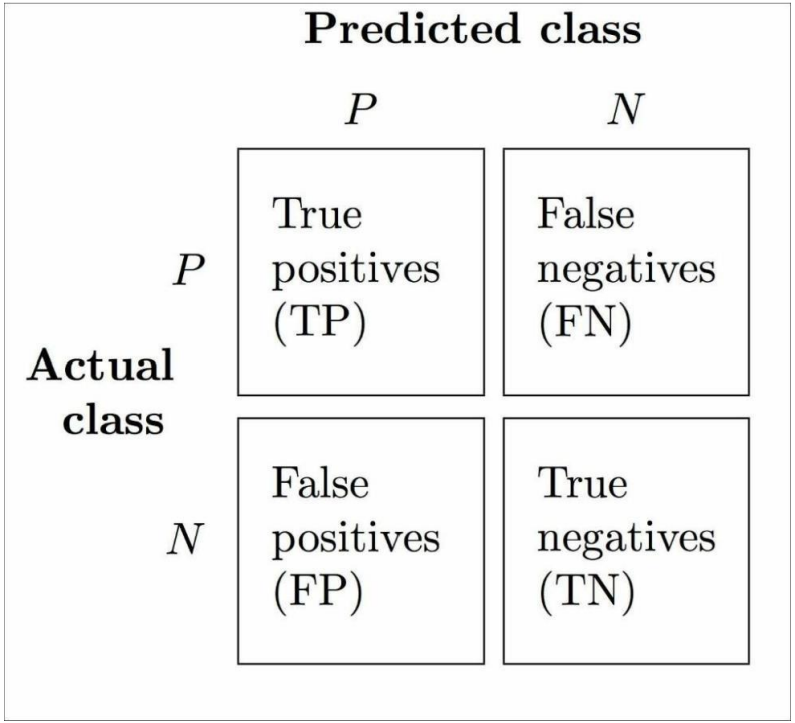
(바깥쪽 루프 폴드 개수 x 안쪽 루프 폴드 개수)




장점

- 여러 종류의 머신러닝 알고리즘을 비교하는 데 좋음
- 테스트 세트에 대한 추정오차 거의 편향 x

# 오차행렬




- **오차행렬**

  : TP, TN, FP, FN의 개수를 적은 정방행렬

- **FP(False Positive)**: 실제 Negative인 정답을 Positive라고 예측 (False) ➔ *Type I error (α)*
- **FN(False Negative)**: 실제 Positive인 정답을 Negative라고 예측 (False) ➔ *Type II error*

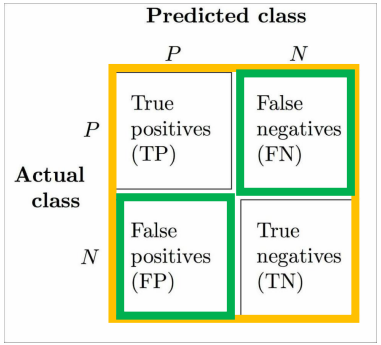
# 분류 모델의 정밀도와 재현율 최적화

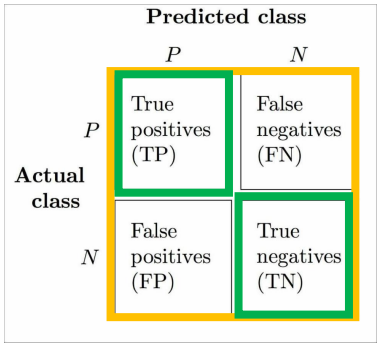
**예측오차(ERR)**: 전체 예측 중 잘못 예측한 경우의 비율

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

**정확도(ACC)**: 전체 예측 중 정확히 예측한 경우의 비율

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




## 문제점

서울에 시간당 1m 이상의 눈이 내리는 것과 같이 거의 불가능한 경우를 예측하는 경우,
무조건 negative라고 예측하면 100%에 가까운 정확도를 얻게 됨.
하지만 이 분류기는 정답을 True Negative로만 맞추고 True Positive는 하나도 발견하지 못하게 됨. ⇒ **정확도 역설**

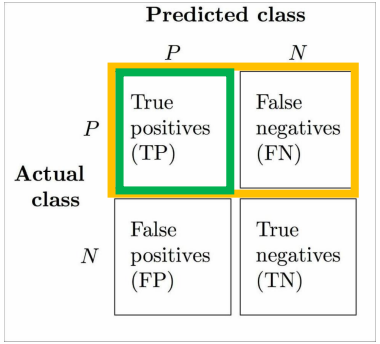
# 분류 모델의 정밀도와 재현율 최적화

**진짜 양성 비율(True Positive Rate, TPR): (=재현율, Recall)**
실제로 참인 경우 중 참이라고 제대로 예측한 경우의 비율

실제 데이터에 negative 비율이 너무 높아서 희박한 가능성으로 발생할 상황을 고려한 지표
강설량 예시에서 True Positive는 없으므로 재현율은 0이됨

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

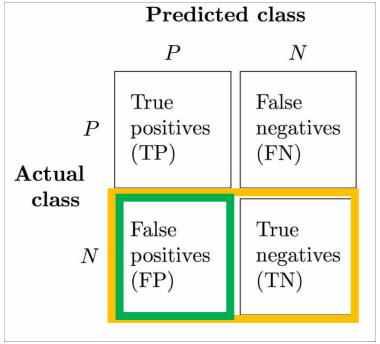
ex) 암환자를 진찰해서 암이라고 진단




**거짓 양성 비율(False Positive Rate, FPR):**
실제로 거짓인 경우 중 참이라고 잘못 예측한 경우의 비율

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

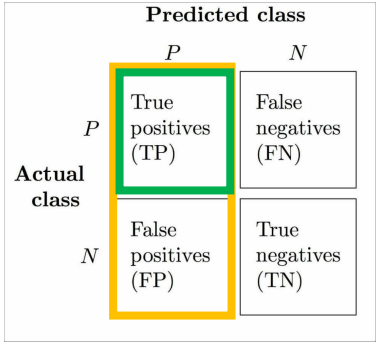
ex) 암환자가 아닌데 암이라고 진단




## 문제점

강설량이 1m 이상이 될지 분류하는 과제에서 언제나 True만 답하는 모델이 있다면,
정확도는 낮지만 눈이 실제로 많이 온 날은 정확히 맞출 수 있기 때문에 재현율은 1이 됨.

# 분류 모델의 정밀도와 재현율 최적화




**정밀도** (Precision)
참이라고 예측한 경우 중 실제로 참인 경우의 비율

$$\frac{True\ Positives}{True\ Positives + False\ Positives}$$

이전의 언제나 True만 답하는 모델에서 재현율은 1로 나오겠지만, 정밀도는 0에 가까움.
시간당 1m 이상 눈 내리는 날은 거의 없기 때문

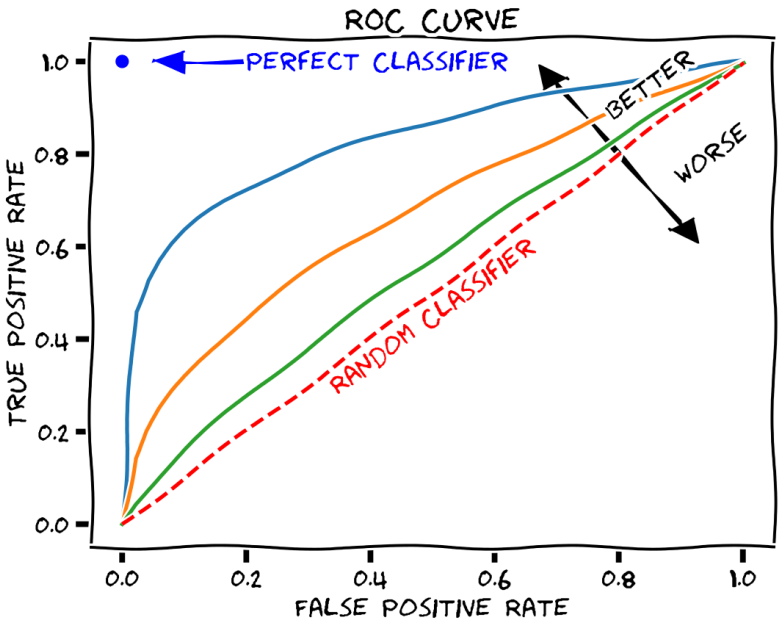
**문제점** **정밀도와 재현율은 서로 반대 개념의 지표**
재현율을 사용할 때의 장점은 정밀도을 사용할 때의 단점. ➔ F1 score

**F1 score**
재현율(recall)과 정밀도(precision)의 조화평균

$$2 * \frac{Precision * Recall}{Precision + Recall}$$

산술평균을 쓰지 않고 조화평균을 구하는 이유
: 재현율과 정밀도 둘 중 하나가 0에 가깝게 낮을 때 지표에 그것이 잘 반영되도록. 즉, 두 지표를 모두 균형있게 반영하여 모델의 성능이 좋지 않다는 것을 잘 확인하기 위함임

# ROC 곡선

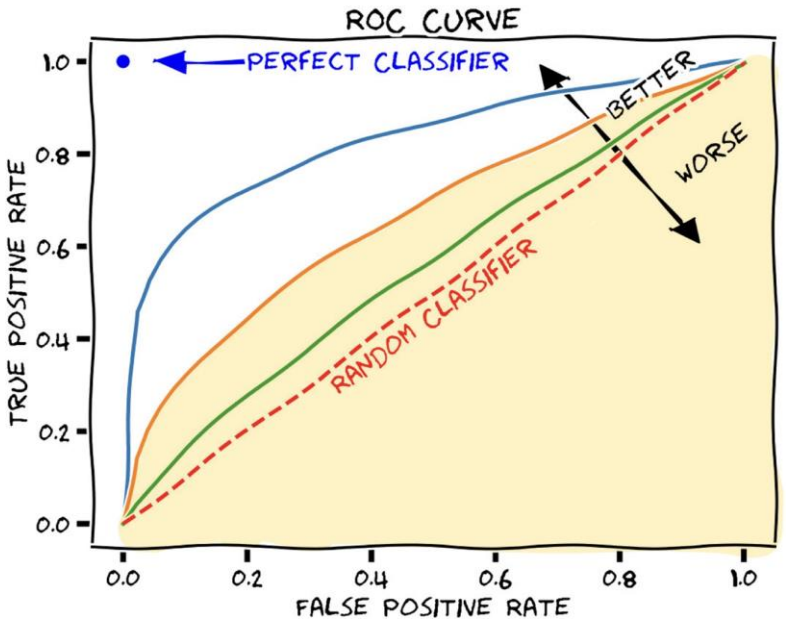



## ROC 곡선 (Receiver Operating Characteristic Curve)

분류기의 임계값을 바꿔가며 계산된 FPR과 TPR 점수를 기반으로 분류 모델을 선택하는 유용한 도구

- FPR과 TPR을 각각 x,y축으로 놓은 그래프
- X,Y가 둘다 [0,1]의 범위이고, (0,0) 에서 (1,1)을 잇는 곡선

# ROC 곡선




- ROC 그래프의 대각선은 랜덤 추측에서의 성능
- 대각선 아래에 위치한 분류 모델은 랜덤 추측보다 나쁘다는 것을 의미.
- 완벽한 분류기 ➔ TPR이 1이고 FPR이 -1인 왼쪽 위 구석에 위치
- ROC AUC(Area Under the Curve) : ROC 곡선 아래 면적.
  - 면적이 1에 가까울수록 성능 좋음
- x축의 FPR과 y축의 TPR은 서로 비례하는 관계
- ex) 암환자를 진단할 때, 성급한 의사는 아주 조금의 징후만 보여도 암인 것 같다고 할 것임. 이 경우 TPR은 1에 가까워지는 동시에 FPR도 1에 가까워짐 (정상인 사람도 전부 암이라고 하니까)
- 따라서 어떤 의사의 실력을 판단하기 위해서는 특정 기준(= 어느 정도의 징후일때 암이라고 예측 할 지)을 연속적으로 바꾸어 가면서 TPR과 FPR을 측정을 해야함. ROC 곡선을 통해 한 눈에 볼 수 있음.

# 다중 분류의 성능 지표

이진 분류 (2개의 클래스, 하나의 문제에 대해 참 혹은 거짓)

⇒ 다중 분류 (k개의 클래스가 있는 경우)

- **마이크로 평균:** 클래스별로 TP,TN,FP,FN을 계산

$$PRE_{micro} = \frac{TP_1 + \cdots + TP_k}{TP_1 + \cdots + TP_k + FP_1 + \cdots + FP_k}$$

각 샘플이나 예측에 동일한 가중치 부여

- **마크로 평균:** 클래스별 정밀도의 평균

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

모든 클래스에 동일한 가중치 부여
각 클래스 레이블의 샘플 개수를 가중하여 평균 계산
불균형한 클래스 다룰 때 유용

# 클래스 불균형

## 클래스 불균형

- 데이터에서 각 클래스 (주로 범주형 반응 변수) 가 갖고 있는 데이터의 양에 차이가 큰 경우
- ex) 질병이 있는 사람과 없는 사람의 데이터를 수집했을 때, 질병이 없는 사람이 더 많음

## 클래스 균형이 필요한 이유

- 소수의 클래스에 특별히 더 큰 관심이 있는 경우에 필요함
- ex) 집을 살지 말지 결정하는 예측 모델을 만드는 경우, 집을 사라고 예측하는 것이 더 큰 리스크를 수반하기 때문에 더 큰 정확도를 가져야 함. 하지만 데이터가 '집을 사지마라' 클래스에 몰려있으면 '집을 사라'라고 예측하는 것에는 성능이 안좋아짐. 따라서 '집을 사라' 클래스에 더 큰 비중을 두고 예측하도록 모델을 설계해야함

# 클래스 불균형 해결 방안

1. 모델 평가 지표로 정확도가 아닌 다른 지표를 활용
   : 정확도, 재현율, ROC 곡선 등 데이터와 목적에 맞는 지표를 사용
2. 데이터의 양이 적은 클래스에서 발생한 예측 오차에 큰 벌칙 부여
   : ex) 집을 사라는 클래스에 관해서는 더 큰 정확도가 필요하므로,
   이 클래스의 데이터에 관해서는 loss가 더 크도록 weight를 줌.
3. Under Sampling & Over Sampling