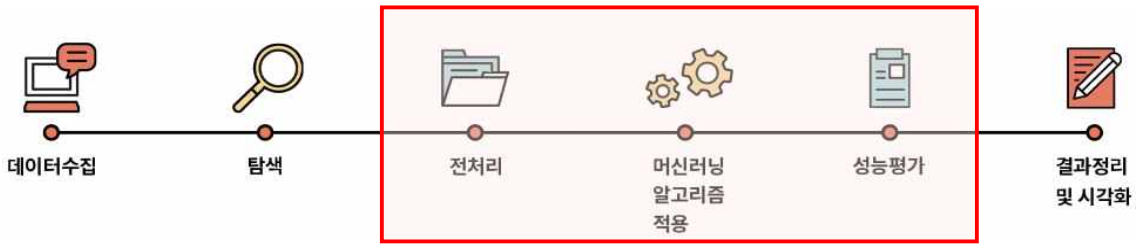
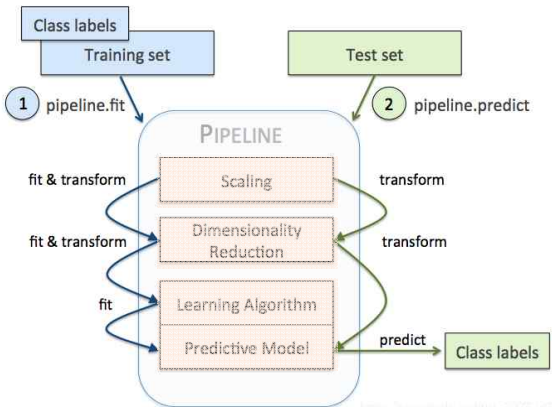


모델 최적화: 주어진 데이터 성능 평가 결과가 가장 좋은 모델을 찾는 과정

전처리 ~머신러닝 알고리즘 적용 ~ 성능평가 ⇒ 최적의 모델을 찾는 반복 구간

파이프 라인(Pipeline)



sklearn.pipeline.make_pipeline(*steps)

: 연속된 변환을 순차적으로 처리하는

기능을 제공하는 래퍼(Wrapper) 도구

모델 성능 평가

교차 검증(Cross Validation)

: 모델 성능을 검증하는 방법

1. 홀드아웃 교차 검증(Holdout Cross Validation)

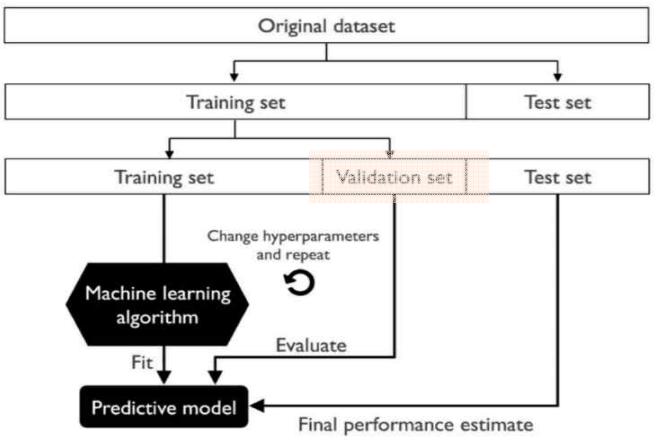
전체 데이터를 학습 데이터와 테스트 데이터로 나눴다

학습 데이터 → 모델 학습, 테스트 데이터 → 일반화 성능 추정

모델 선택을 통한 하이퍼 파라미터 튜닝

테스트 데이터 기반 튜닝을 시도하지만 올바르지는 않다

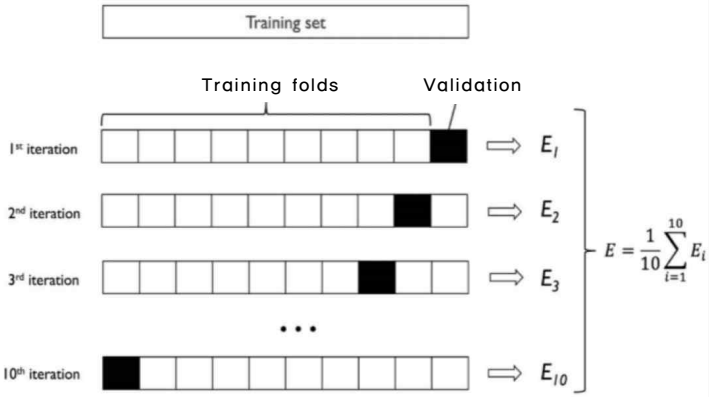
해결방법: 검증 데이터(Validation set)



1) 전체 데이터를 학습 데이터와
검증 데이터, 테스트 데이터로 나눈다

2) 학습데이터 → 모델학습
검증데이터 → 하이퍼파라미터 튜닝
테스트데이터 → 성능 추정

2. K겹 교차 검증(K-fold Cross Validation)



1) 중복없이 학습 데이터를
K겹으로 랜덤하게 나눈다
2) K-1겹으로 모델을 훈련하고
나머지 하나로 성능을 평가한다
3) 각각의 폴드에서 얻은 성능을
기반으로 평균 성능을 계산한다

추천하는 K값은 10이다
but K가 크면 실행시간이 길어지므로 큰 데이터는 작은 K값 선택해도 ok

장점
1. K번 반복하므로 K개의 서로 다른 모델을 얻을 수 있다
2. 홀드아웃 방법보다 데이터 분할에 덜 예민한 성능평가가 가능하다
3. 중복을 허락하지 않기 때문에 모든 샘플이 검증에 딱 한 번 사용된다

과적합 문제
1. 과대적합(overfitting)
: 모델이 학습 데이터에 너무 잘 맞지만 일반화가 떨어지는 상황
일반화(generalization): 테스트 데이터에 대한 높은 성능을 갖추는 것

해결방법
1. 학습 데이터 추가하기
2. 모델 제약 늘리기: 규제(regularization) 값 늘리기(C값↓ → 규제↑)
3. 학습 데이터 잡음 줄이기(오류 수정 및 이상치 제거)

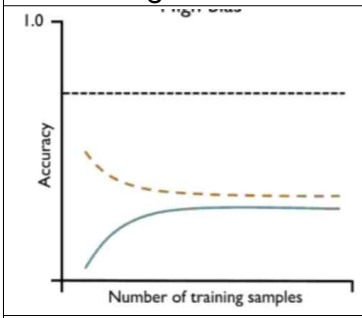
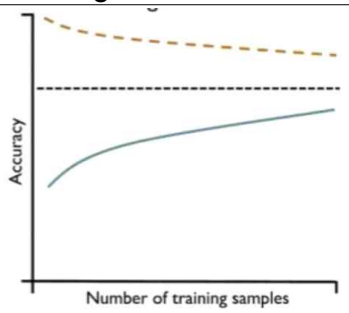
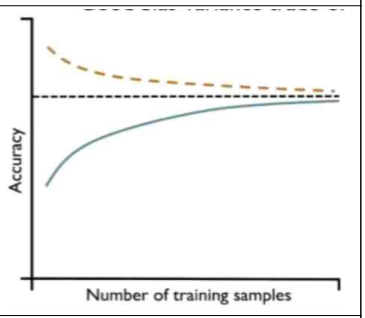
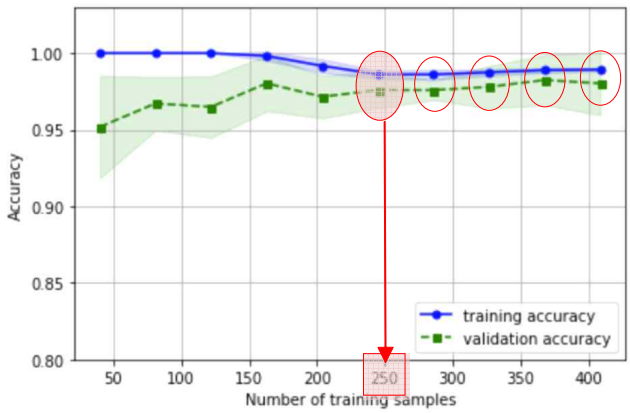
2. 과소적합(underfitting)
: 모델이 너무 단순하여 데이터에 내재된 구조를 학습하지 못하는 현상
해결방법
1. 파라미터가 더 많은(복잡도가 더 높은) 모델을 선택하거나 변경하기
2. 모델의 제약 줄이기: 규제 값 줄이기(C값↑ → 규제↓)
3. 과적합 이전까지 충분히 학습하기

모델 최적화

과적합 판단하기

1. 학습 곡선(Learning Curve)의 편향과 분산 분석

샘플 데이터의 수에 따른 정확도 변화

<table>
<tr><th>High Bias</th><th>High Variance</th><th>Good Bias-Variance trade-off</th></tr>
<tr><td> 
</td><td> 
</td><td> 
</td></tr>
<tr><td>Underfitting
– 모델 파라미터 수 늘리기
– 규제 강도 줄이기</td><td>Overfitting
– 학습 데이터 보강하기
– 규제 강도 높이기</td><td>Training accuracy
Validation accuracy
Desired accuracy</td></tr>
<tr><td colspan="3"> 
</td></tr>
</table>

2. 검증 곡선(Validation Curve)

매개변수에 따른 정확도 변화: 로지스틱 회귀의 매개변수 C($\propto \frac{1}{\text{규제 강도}}$)