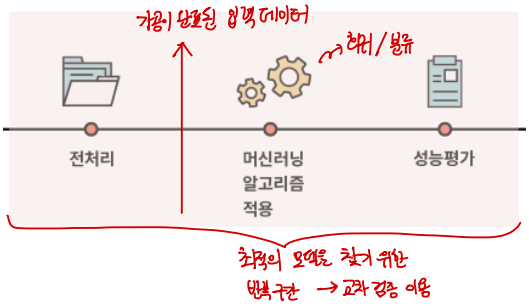
교차 검증

무인이동체공학과 17011882 김 우 혁

- **모델 최적화**: 주어진 데이터 성능 평가 결과 최적의 모델을 찾는 과정 ex) 교차 검증



- **파이프 라인**: 사이킷런의 Pipeline 클래스는 연속된 변환을 순차적으로 처리할 수 있는 기능을 제공하는 유용한 래퍼(Wrapper) 도구

```
pipe_lr = make_pipeline(StandardScaler(),
                        PCA(n_components=2),
                        LogisticRegression(solver='liblinear', random_state=1))

pipe_lr.fit(X_train, y_train)
y_pred = pipe_lr.predict(X_test)
print('테스트 정확도: %.3f' % pipe_lr.score(X_test, y_test))
```

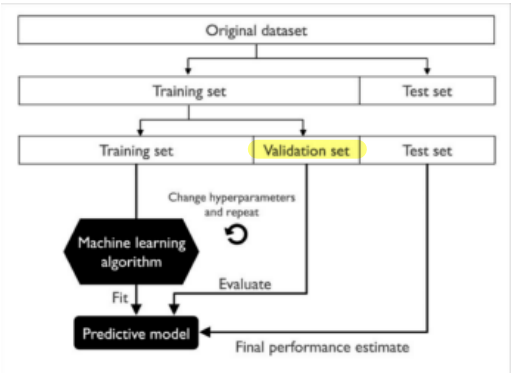
⇒ 이렇게 묶어서 사용하여 매우 효율적 !

- **교차 검증**(Cross Validation)

- 모델 성능 검증하기 위한 방법: 홀드아웃 교차 검증(우리가 지금까지 사용한 방법), K-fold 교차 검증(추천!)

  - **홀드아웃 교차 검증**: 전체 데이터를 ① 학습 데이터, ② 검증 데이터, ③ 테스트 데이터로 나누고, 학습 데이터는 모델 학습에 / 검증 데이터는 하이퍼파라미터 튜닝에 / 테스트 데이터는 성능 평가에 사용

최종적으로 한번만 적용
테스트 데이터 기반 튜닝을 수행하는 것은 올바른 방법이 아님.



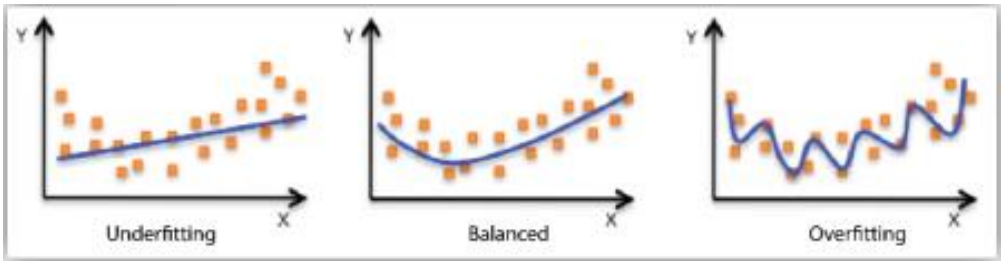
■ K-fold 교차 검증: K-fold 교차 검증에서는 "중복없이" 훈련 데이터를 K겹으로 랜덤하게 나눔 → K-1 겹으로 모델을 훈련하고, 나머지 하나로 성능을 평가 → K번 반복하므로 K개의 서로 다른 모델을 얻을 수 있음

➔ K겹 교차 검증은 각각의 폴드에서 얻은 성능을 기반 → 평균 성능을 계산

➔ 홀드아웃 방법보다 데이터 분할에 덜 예민한 성능 평가 가능

➔ 모든 샘플이 검증에 딱 한번 사용됨(중복X) + 추천 K값은 10

● 과적합 문제



■ 과대적합(overfitting): 모델이 학습 데이터에 너무 잘 맞지만 일반화가 떨어짐

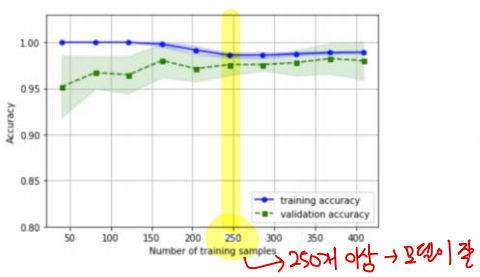
- 여기서, 일반화(generalization)란? 테스트 데이터에 대한 높은 성능을 갖추는 것

- 해결책: 학습데이터 추가 수집/ 규제(regularization)값 늘리기/ 학습 데이터 잡음을 줄임(오류 수정 및 이상치 제거)

↳ 여기서, C가 작아지면 규제 ↑

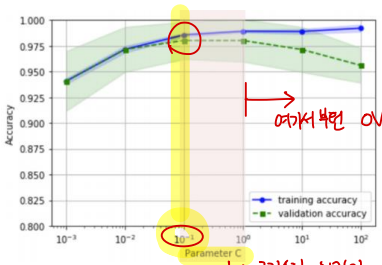
■ 과소적합(underfitting)이란? 모델이 너무 단순하여 데이터에 내재된 구조를 학습하지 못하는 현상

- 해결책: 파라미터가 더 많은 모델 선택/ 규제(regularization)값 줄이기/ 과적합 이전까지 충분히 학습



↳ 250개 이상 → 모델이 잘 작동함.

↳ 샘플 데이터 순서 따라 정확도 변화



여기서부턴 overfitting !

↳ 교차검증 과정의 매개변수 C (규제 강도의 반비례)

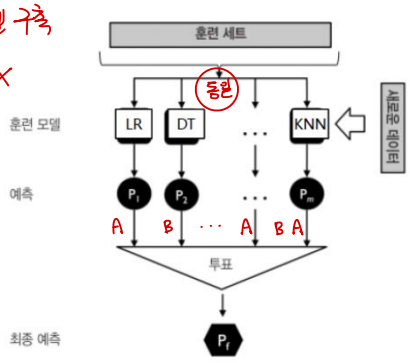
↳ 매개변수에 따른 정확도 변화

- **앙상블** → 집단 지성

- 목적: 여러 분류기를 하나로 연결하여 개별 분류기 보다 더 좋은 일반화 성능을 달성
- 방법
  - 여러 분류 알고리즘 사용: 다수결 투표(Voting) → LR, DT, KNN, LDA
  - 하나의 분류 알고리즘을 여러 번 사용: 배깅(Bagging), 부스팅(Boosting) → DT ⇝ RF
- 종류
  - 다수결 투표(Majority Voting): 동일한 학습 데이터 사용
  - 배깅(Bagging): 알고리즘 수행마다 서로 다른 학습 데이터 추출 사용 ex) Random Forest
  - 부스팅(Boosting): 샘플 뽑을 때 잘못 분류된 데이터 50% 재학습에 사용 또는 가중치 사용

- **다수결 투표**

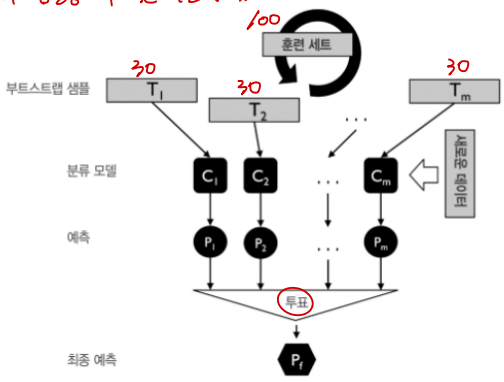
– 동일한 학습데이터로 모델 구축
– 샘플을 뽑을 때 중복 X



- **배깅**

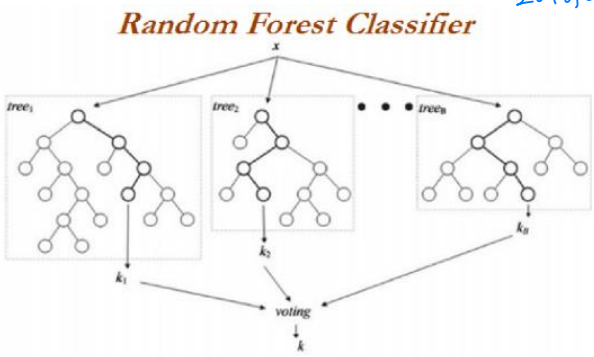
알고리즘마다 변수의 학습 데이터를 추출 (샘플링) 하여 모델 구축에 사용

  - ★ 부트스트랩(Bootstrap) 사용 ⇝ 복원 추출!
    ↳ 학습 데이터 샘플링 시 복원 추출 (중복)



■ 랜덤 포레스트 → 배깅의 일종

- 단일 분류 알고리즘(Decision Tree) 사용
- Forest 구축: 무작위로 예측변수 선택하여 모델 구축
- 결합 방식: 투표(분류) / 평균화(예측)



● 부스팅

- 샘플 뽑을 때 잘못 분류된 데이터의 50%를 재학습에 사용
- AdaBoost: 전체 학습데이터를 사용하고 잘못 분류된 데이터에 가중치 적용