3.
MODEL_SELECTION?

POINT 1

학습/테스트 데이터 세트 분리 :
train_test_split()

POINT 2

교차 검증

POINT 3

GridSearchCV, RandomSearchCV

3. model_selection

1. model_selection 이란?

- 학습 데이터와 테스트 데이터 세트를 분리
  : train_test_split()

- 교차 검증 분할 및 평가 (Cross Validation)
  : KFold / Stratified KFold

- Estimator의 하이퍼파라미터 튜닝을 위한 다양한 함수와 클래스 제공
  : GridSearchCV / RandomizedSearchCV

위의 기능을 수행하는 함수와 클래스를 제공하는 모듈

3-1. train_test_split()

1. data split의 목적
   - 모델을 학습 시킨 후 그 성능을 검증하기 위해
   원본 데이터에서 train / test 용 데이터로 나눠줌
   - 모델의 일반화 능력을 평가함

X_train, X_test

X_1	X_2	X_p

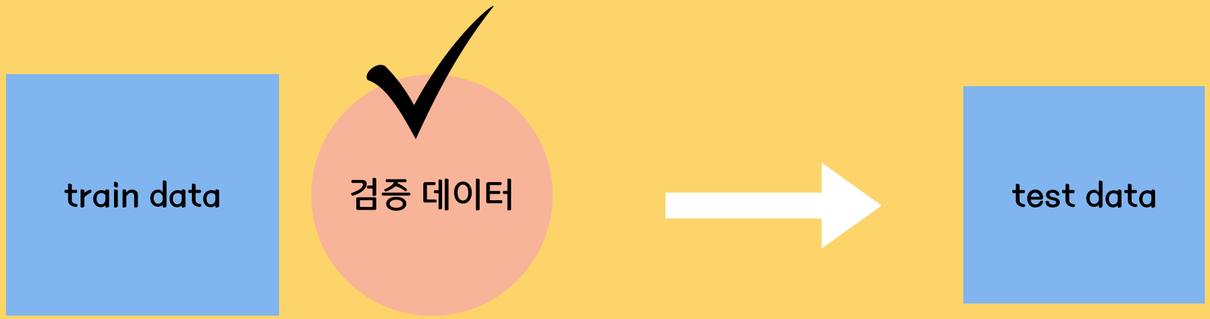
Y

y_train, y_test

2. train_test_split(X, y, test_size, random_state, shuffle)
   - X, y : feature, label 데이터
   - test_size : 테스트 데이터셋 크기 결정 (default = 0.25)
   - random_state : 동일한 학습/ 테스트 데이터셋 결정
   - shuffle : 데이터를 섞어서 학습 진행

3-2. 교차 검증

1. 교차 검증 이란?



만약, 검증 데이터 1개로만 테스트를 한다면, 전체 데이터에 대해 적합한지 판단 불가
즉, 모델의 과적합을 방지하고 일반화된 데이터에 대한 성능을 측정할 방법이 필요!

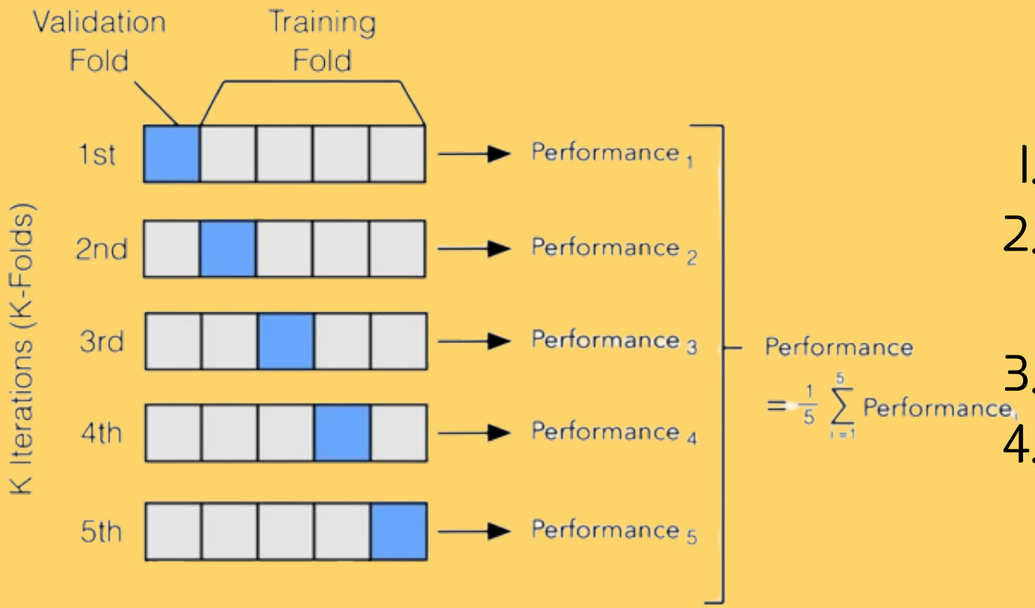
*** 교차 검증의 등장 > 여러 세트로 구성된 데이터셋을 나눠서 학습과 평가를 진행함

3-2. KFold

2. KFold 교차 검증

전체 데이터를 K개의 데이터 폴드 세트로 만들어서 K번만큼 각 폴드 세트에 학습과 검증 평가를 반복적으로 수행하는 방법

** 폴드 : 나눠진 각각의 데이터셋



대략적 순서

1. 학습 데이터를 K개의 폴드로 나눈다.
2. 첫번째 폴드를 검증 데이터로 두고 나머지 데이터로 모델을 학습시킨다. 그 후 모델의 평가를 진행한다.
3. 검증 데이터를 바꿔 가면서 K=5 까지 반복한다.
4. K번 진행된 평가를 각각 평균 내어 모델의 최종 평가 지표(정확도)를 구한다.

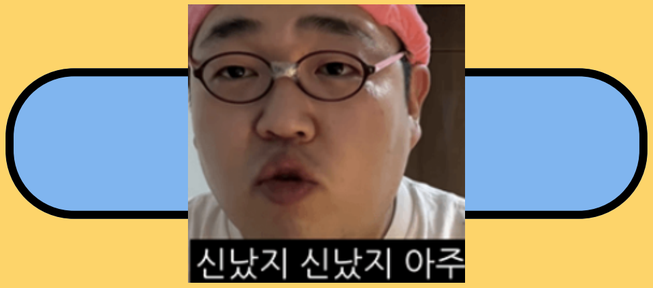
3-2. KFold의 문제점

0, Setosa	1, Versicolor	2, Virginica
50	50	50

<iris datasets>

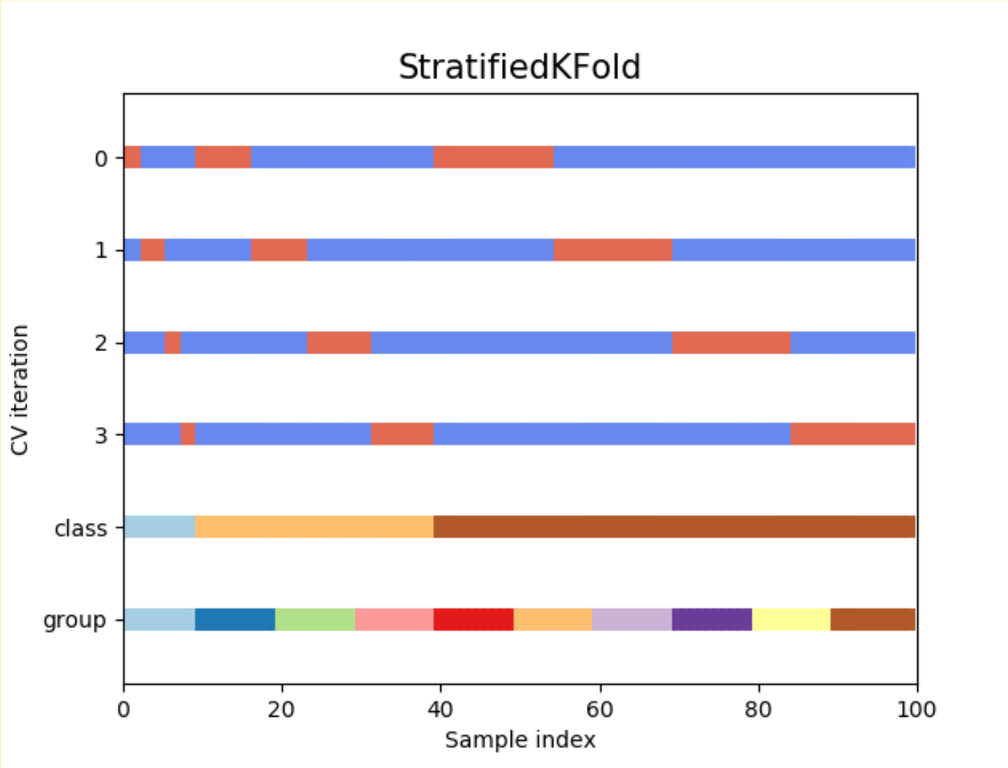
K = 3이고, 레이블 0, 2의 데이터로만 학습을 진행하고 레이블 1의 데이터로 검증한다면, 모델의 정확도는 0

Q2. 한 레이블이 검증 데이터에만 편향되는 문제점을 극복한 클래스는?



3-2. Stratified KFold

레이블 분포를 먼저 고려한 뒤 이 분포와 동일하게 학습과 검증 데이터 세트를 분배함



KFold vs Stratified KFold

KFold

estimator : 회귀(regressor)

scoring: MSE, MAE, RMSE

Stratified KFold

estimator: 분류(classifier)

scoring: 정확도, 정밀도, 재현율

3-2. cross_val_score

```
# Kfold
SPLITS = 5
kf = KFold(n_splits = SPLITS)
n_iter = 0

features = data.iloc[:,:-1]
label = pd.DataFrame(data['label'])

score_list = []
for train_idx, test_idx in kf.split(features, label):
  n_iter += 1
  print(f'--------------------{n_iter}번째 KFold-------------------')
  print(f'train_idx_len : {len(train_idx)} / test_idx_len : {len(test_idx)}')

  label_train = label.iloc[train_idx]
  label_test = label.iloc[test_idx]

  X_train, X_test = features.iloc[train_idx, :], features.iloc[test_idx, :]
  y_train, y_test = label.iloc[train_idx,:], label.iloc[test_idx,:]

  model.fit(X_train, y_train)
  preds = model.predict(X_test)
  score = accuracy_score(y_test, preds)

  print(f'{n_iter}번째 단일 accuracy_score:{score}')
  score_list.append(score)


print('=====================================================')
print(f'최종 평균 accuracy_socre : {sum(score_list)/len(score_list)}')
```

score = cross_val_score( )

위 과정을 한꺼번에 수행해주는 API

- 폴드 세트 설정
- 학습/ 데스트 데이터 추출
- 학습/평가 반복

내부적으로 Stratified KFold 이용

3-2. K(폴드 수)는 어떻게 설정할까?

3-3. Hyperparameter Optimization

01. GridSearchCV

02. Random Search

3-3. GridSearchCV -> Optimizaion + CV

GridSearchCV(estimator, param_grid, scoring, cv, refit)

1. estimator : 모델

2.param_grid
: 사용할 파라미터가 정의된 dictionary
가능한 모든 조합 이용

3.scoring : 성능 평가 지표

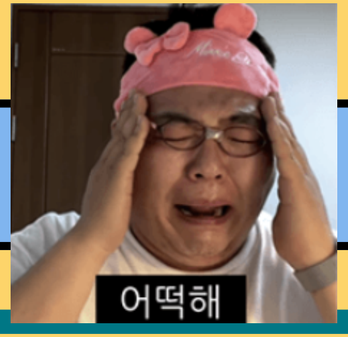
4. cv : cv 시 폴드 수

5. refit : 최적의 파라미터로 자동 재학습 처리

결정트리

분류모델

Q3. 결정트리에서 최대 깊이와 노드를 분할하기 위한 최소 샘플 수를 의미하는 2가지 파라미터는?



3-3. GridSearchCV(결정트리)

1. max_depth : 트리의 최대 깊이
   - 깊어질수록 과적합 가능성 증가

2. min_samples_split : 자식노드 분할을 위한 최소한의 샘플 데이터수

순번	max_depth	min_samples_split
1	1	2
2	1	3
3	2	2
4	2	3
5	3	2
6	3	3

Q4. cv = 3, 다음의 파라미터로 모델을 학습시킬 때,
총 학습 / 평가 횟수는?

param_grid = {'max_depth' = [1,2,3], 'min_samples_split' = [2,3]}

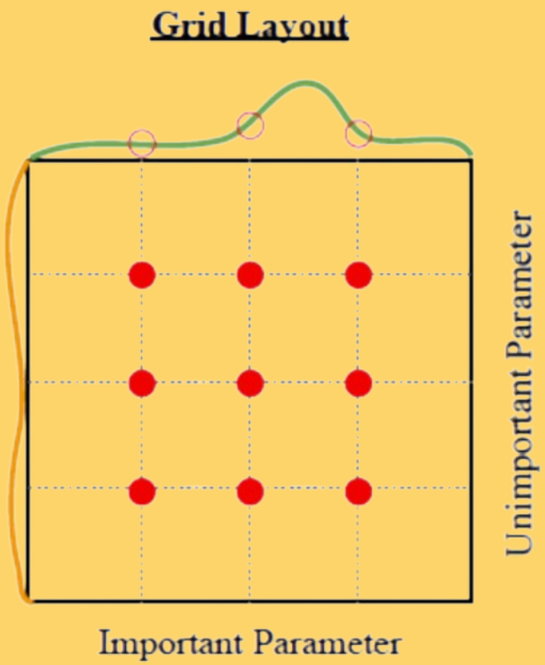
3-3. GridSearchCV(결정트리)

	params	mean_test_score	rank_test_score	split0_test_score	split1_test_score	split2_test_score
0	{'max_depth': 1, 'min_samples_split': 2}	0.700000	5	0.700	0.7	0.70
1	{'max_depth': 1, 'min_samples_split': 3}	0.700000	5	0.700	0.7	0.70
2	{'max_depth': 2, 'min_samples_split': 2}	0.958333	3	0.925	1.0	0.95
3	{'max_depth': 2, 'min_samples_split': 3}	0.958333	3	0.925	1.0	0.95
4	{'max_depth': 3, 'min_samples_split': 2}	0.975000	1	0.975	1.0	0.95
5	{'max_depth': 3, 'min_samples_split': 3}	0.975000	1	0.975	1.0	0.95

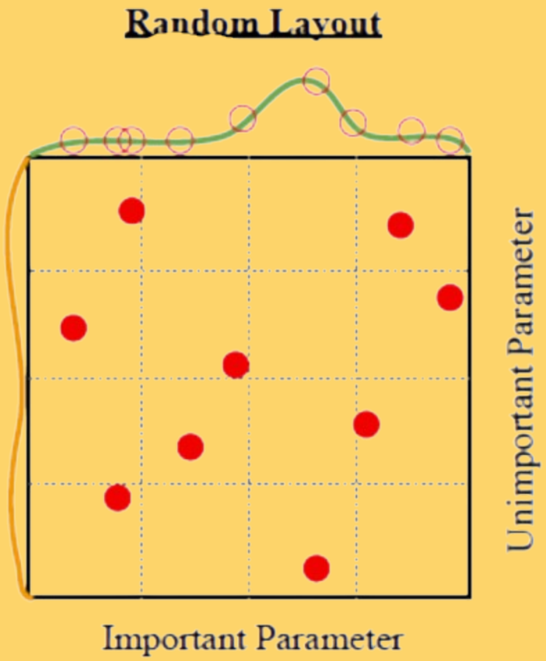
3 X 2 X 3(폴드 수) = 18번

3-3. Random Search

Lasso 의 예시



일정한 간격의 모든
하이퍼파라미터의 조합을 시도
파라미터 수, 범위가 작을 때 효과적



0-1 사이의 랜덤한 (ex.0.178) 수를
파라미터로 설정한 후 성능 검증
데이터가 많은 경우 학습시간 절약