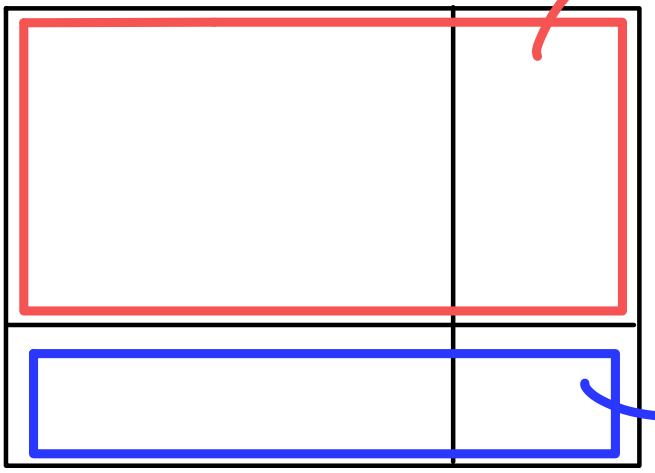
220124

< Review >



< K-Fold Cross Validation >

여러등분을 내고

학습용데이터를 미리 성능분석하여

일정 정확도 이상시 test 진행

test data (실전)

cross_val_score 이용하여

train을 진행 → 검증까지

DecisionTreeClassifier. → max_depth에 따라 성능의 변화.

: 최적의 max_depth를 구하기 , for 구문을 돌려서 확인하기.

cf (min_samples_leaf 내려가고 samples 개수
min_samples_split 내려가기전 개수.
⇒ 분할의 조건.

< Hyperparameter >

- 알고리즘을 이용하여 모델링 할때 , 최적화를 위한 옵션
- 지식. 경험. 다양한 시도를 통해 성능을 향상시킬 수 있다.

⇒ 파라미터 확인. get_params

ex. KNN에서 k값(n_neighbors)

DecisionTree에서 (min_samples_leaf
max_depth

⇒ 어떤식으로 최적의 parameter를 찾을수있는가?

< Random Search. Grid Search >

→ a = [1, 2, 3, ⋯, 10]

b = [1, 2, 3, ⋯, 20]

→ 최적의 parameter 쌍을 구하기위해 200번시도.

+ CV=10 → 2000번.

따라서, 여기서 10개만 Random하게 골라라. (Random)

모든경우의수 - parameter 검증 (grid)

Sklearn → RandomizedSearchCV : RandomSearchCV(model, param, cv, n_iter, scoring)

→ MAE의 최고 값을 선택

따라서 neg_mean_average_error로 선언, 그러나 보통 r2_score를 선택.

☆ n_jobs = -1.

외부의 core를 이용하여 병렬처리를 활용하겠다 → 성능향상.

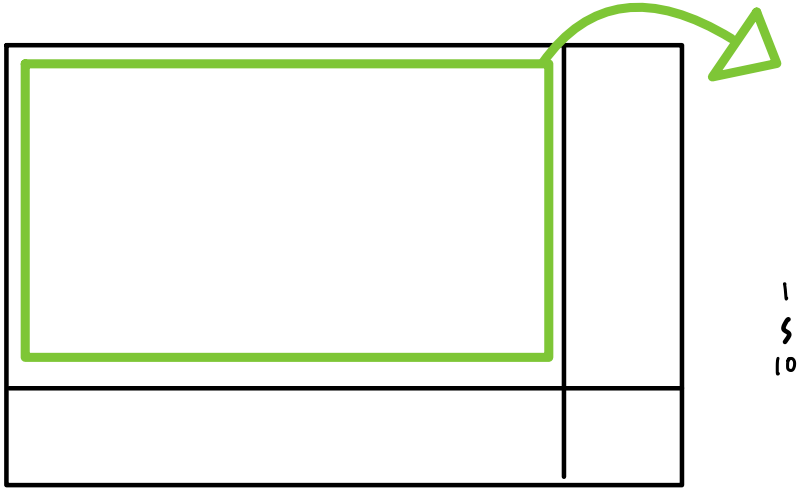
: -1 → 모두활용.

✓ GridSearchCV = 모든경우에 대해서 Search를 진행.

☆ CV : Cross Validation.

기존의 K-Fold CV의 기능이 포함됨. → CV의 형태, parameter.

mean_test_score



params = { 'max_depth' : range(1,21) }.

⇒ 깊이를 1부터 20까지 설정.

1. RandomSearchCV (… , cv=10 , n_iter=30, …)

cv = data의 범위를 몇등분 시킬것인가?

n_iter = 몇개의 samples를 고려것인가?

1
5
10

1, 3, 5, 10, ···, 61, 73, 59 (30개)

⇓

→ max_depth의 1~20까지의 결과가중기.

model.best_score_ = 해당 모델에서 최고점수기.

model.best_params_ = 해당 모델에서 가장 좋은 params 선택.

model.best_estimator_

⇒ get_params()를 이용하면 선언한 parameter중 최고의 모델이 존재.