Dacon_nuclear_study_1st_3조

3 조 2 주차 발표자료_교차검증

교차검증

교차검증의 필요성

일반적으로 알고리즘을 학습시키는 학습 데이터와 예측 성능을 평가하기 위한 별도의 테스트용 데이터 필요

만약 별도의 테스트용 데이터 (즉, 학습용 데이터로만 테스트를 진행하는 경우)가 없다면 정확도가 100%에 이르는 모델 탄생..!

하지만 단순히 테스트용 데이터를 생성한다면 과적합에 취약하게 됨

과적합(Overfitting) : 모델이 학습 데이터에만 과도하게 최적화되어 실제 예측을 다른 데이터로 수행할 경우 예측 성능이 과도하게 떨어지는 것

즉, 고정된 학습 데이터와 테스트 데이터로 평가를 하는 경우, 해당 테스트 데이터에만 최적의 성능을 발휘하도록 편향됨 -〉 다른 테스트용 데이터로 테스트하는 경우 성능이 급격히 저하

이를 개선하기 위해 교차검증 방식이 도입됨

교차검증의 정의

데이터 편중을 막기 위해 여러 세트로 구성된 학습 데이터 세트와 검증 데이터 세트에서 학습과 평가를 수행하는 것

➔ 각 세트 별 수행 결과에 따라 하이퍼파라미터 튜닝 등 모델 최적화를 더욱 손쉽게 할 수 있음

대부분 ML의 성능 평가는 교차검증 기반 1차 평가한 후 최종 테스트 데이터 세트에 적용하여 평가하는 방식

➔ 테스트 데이터 세트 외 별도의 검증 데이터 세트를 둬서 최종 평가 이전에 학습된 모델을 다양하게 평가하는 데 사용

Hyper parameter

튜닝 옵션, 연구자나 기술자들이 직접 세팅해야 하는 설정

교차 검증의 필요성

고정된 학습 데이터와 테스트 테이터로 평가하게 되면, 주어진 데이터에 대해서만 최적의 성능을 발휘하도록 편향된다!

Overfitting을 방지하고 정확도를 높이기 위해 사용됨

test set 외에 별도의 validation set을 두어 평가함

K-fold Cross-validation

Dacon_nuclear_study_1st_3조

K개의 데이터 폴드 세트를 구성해 K번 만큼 각 세트에 대해 학습과 평가를 수행

+) fix random seed for 'reproducibility'

우승자 코드에서 seed값을 임의로 부여해 의문이었는데, 찾아보니 재생산성을 위해 보통 시드값을 지정한다고 한다.

K-Fold 교차검증

가장 보편적으로 사용되는 교차 검증 기법

K개의 데이터 폴드 세트를 구성하여 K번 만큼 각 폴드 세트에 학습과 검증 평가를 반복적으로 수행

```
kfold = KFold(n_splits=4, random_state = rs, shuffle = True)
```

〈우승자 코드 중..〉

n_splits=4	K=4,
random_state = rs	random_state는 고정되지 않음(random seed)
shuffles = True	데이터를 섞어서 샘플의 숫자를 랜덤하게 만듦 (학습 데이터의 다양성 높아짐)

주요 단점은 연산 비용이 늘어난다는 것

StratifiedKFold

불균형한 분포도를 가진 레이블 데이터 집합을 위한 K폴드 방식

불균형한 분포도를 가진 레이블 데이터 집합은 특정 레이블 값이 특이하게 많거나 매우 적어서 값의 분포가 한쪽으로 치우치는 경우를 말함

K폴드가 레이블 데이터 집합이 원본 데이터 집합의 레이블 분포를 학습 및 테스트 세트에 제대로 분배하지 못하는 경우의 문제를 해결

➔ 원본 데이터의 레이블 분포를 고려 + 해당 분포와 동일하게 학습과 검증 데이터 세트 분배!

따라서, 일반적인 분류에서의 교차 검증은 KFold 보다 StratifiedKFold를 이용하는 것이 더욱 적합다고 판단됨!

단, 회귀 모델의 경우 지원하지 않음

회귀의 결정값은 이산값 형태의 레이블(분류)이 아니라 연속된 숫자값이므로 결정값별로 분포를 정하는 것은 의미 없음

Cross_val_score()

교차 검증을 간단하게 할 수 있는 사이킷런의 API

일반적인 KFold의 코드 구성

1) 폴드 세트 설정 2) 루프(for문)에서 반복으로 학습 및 테이스 데이터의 인덱스 추출

3) 반복적으로 학습 및 예측 수행 4) 예측 성능 반환

위 일련의 과정을 Cross_val_score()는 한꺼번에 수행!

```
cross-val-score(estimator, X, y=None, scoring=None, cv=None, n_jobs=1,
                verbose=0, fit_params=None, pre_dispatch='2*n_jobs')
```

〈cross_val_score() 선언형태〉

estimator : 사이킷런의 분류 알고리즘 클래스인 Classifier 또는 회귀 알고리즘 클래스인 Regressor을 의미

Classifier 가 입력되면 stratified K폴드 방식으로 레이블 값의 분포에 따라 학습/테스트 세트를 분할

X : 피처 데이터 세트

y : 레이블 데이터 세트

scoring : 예측 성능 평가 지표를 기술

cv : 교차 검증 폴드 수를 의미

반환값은 scoring파라미터로 지정된 성능 지표 측정값을 배열 형태로 반환

cf. Cross_val_score()는 하나의 평가지표만 반환하지만, cross_validate()는 여러 개의 평가지표를 반환할 수 있음

GridSearchCV

사이킷런에서 제공하는 API로 교차 검증과 최적 하이퍼파라미터 튜닝을 한번에 할 수 있음

하이퍼 파라미터 : ML 알고리즘을 구성하는 주요 구성 요소 -> 이 값을 조정하여 알고리즘의 예측 성능 개선 가능

Classifier나 Regressor 알고리즘에 사용되는 하이퍼 파라미터를 순차적으로 입력하여 최적의 파라미터 도출

1) 데이터 세트를 교차 검증을 위한 학습-테스트 세트로 자동 분할

2) 하이퍼 파라미터 그리드에 기술된 모든 파라미터를 순차적으로 적용하여 최적의 파라미터를 찾음

예. 총 6번 학습(for루프)

```
grid_parameters = {'max_depth' : [1, 2, 3], 'min_samples_split' : [2,3]}
```

〈예시〉

즉, for루프로 모든 파라미터를 번갈아 입력하면서 학습시키는 방법을 api레벨에서 제공한 것

➔ 교차 검증을 기반으로 하이퍼 파라미터의 최적 값을 찾게 해줌

```
GridSearchCV(estimator, prarm_grid, scoring, cv, refit)
```

〈선언 형태〉

estimator : classifier, regressor, pipeline이 사용될 수 있음

param_grid : parameters값 지정(미리 딕셔너리 형태로 설정하여 불러오기)

scoring : 예측 성능을 측정할 평가 방법을 지정 *보통은 사이킷런의 성능 평가 지표를 지정하는 문자열로(accuracy이런 식으로!)*

cv: 교차 검증을 위해 분할되는 학습/데이터 세트의 개수 지정

refit : 디폴트값 true. 가장 최적의 하이퍼 파라미터를 찾은 뒤 입력된 estimator객체를 해당 하이퍼파라미터로 재학습

연산 비용 상당히 큰 편

토의

1. 질문

여기서 lucky_seed는 어떻게 생성되었는지.. random seed 값을 고정해 놓은 건지?!

In []:

```
# 4FOLD, 3SEED ENSEMBLE
# 총 12개의 모델을 평균내어 예측한다

lucky_seed=[4885,1992,1022]

for num,rs in enumerate(lucky_seed):

    kfold = KFold(n_splits=4, random_state = rs, shuffle = True)
```

〈우승자 코드 중..〉

2. 원자력 대회 데이터에 GridSearchCV를 적용해보면 좋을 것 같아요!

참고자료

Hands-On Machine Learning with Scikit-Learn & Tensor flow

파이썬 ML 완벽가이드

이것이 데이터 분석이다