[ML/DL] python 을 통한 교차검증 (k -Fold , stratifiedkFold) — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북
만든 날짜: 2021-01-01 오후 6:41
URL: https://continuous-development.tistory.com/166?category=736685

ML,DL

[ML/DL] python 을 통한 교차검증 (k -Fold , stratifiedkFold)

2020. 10. 28. 17:29 수정 삭제 공개

교차검증이란?

머신러닝을 돌리기 전에 train test로 나눠 머신을 훈련한다. 하지만 이 훈련 때 학습 데이터에 과도하게 초점을 맞춰 머신이 훈련될 수가 있다.

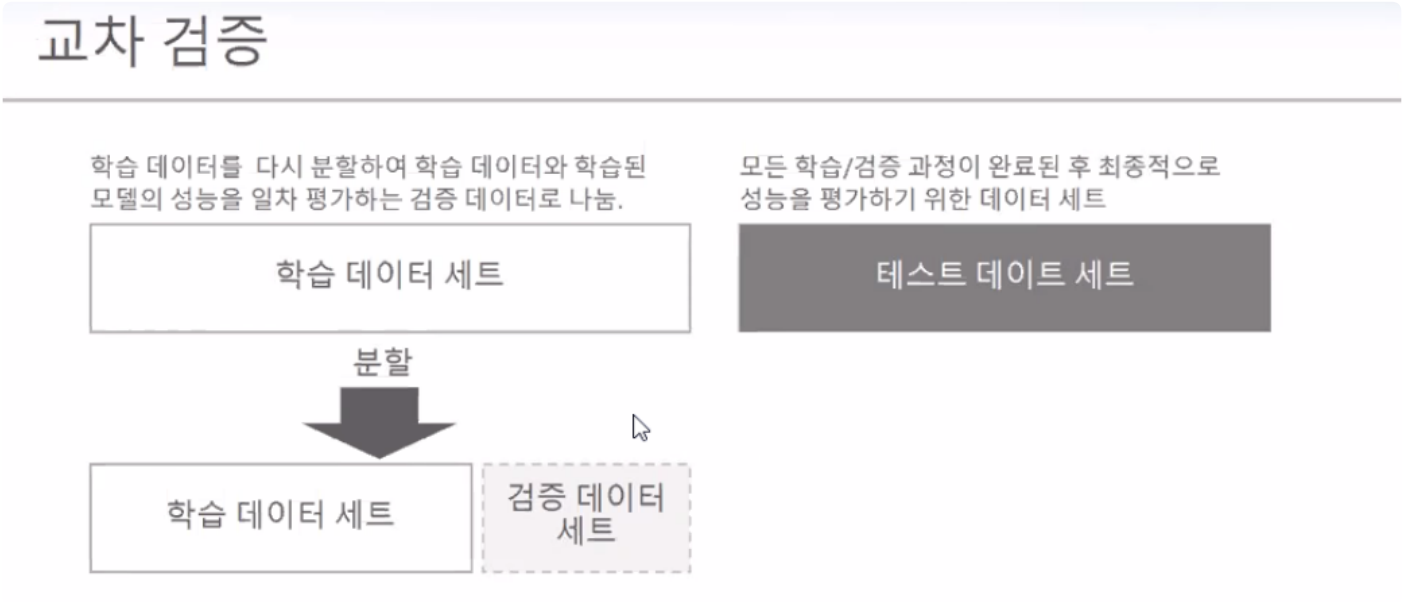
이 같은 경우에는 훈련시에는 점수가 잘 나오지만 실제 테스트를 할 때는 점수가 잘 나오지 않는다.

이걸 과적합(overfitting)이라고 한다.

우리는 훈련시에 이 같은 과적합을 막아야 한다. 이걸 위해 교차검증 이란 것 을 사용한다.

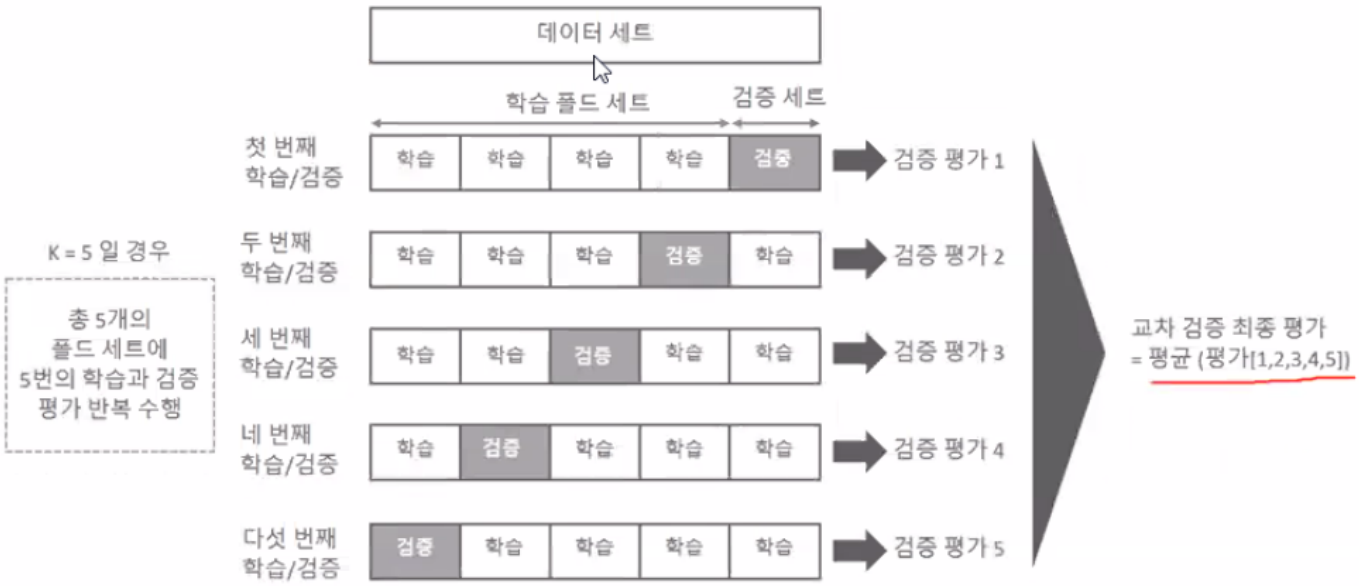
교차검증이란 훈련 데이터 세트를 바꿔가면 훈련하면서 나온 평균을 정확도로 보는 방법을 뜻한다.

이렇게 훈련 데이터 세트를 교차하면서 검증을 하기에 교차 검증이라고 한다.



K-fold 교차검증

K 폴드 교차 검증



학습세트와 검증 세트를 나눠 반복해서 검증한다. 이걸 k값만큼의 폴드 세트에 k번의 학습과 검증을 한다. 이러한 방법으로 k번 평가를 하게 된다.

한 번의 학습을 통해 평가를 할 경우 과적합이 일어날 가능성이 크다. 이럴 때를 대비해 교차검증을 통해 과적합을 막아준다.

K-Fold 사용법

```
from sklearn.datasets import load_iris
from sklearn.tree     import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

from sklearn.model_selection import KFold

import pandas as pd
import numpy  as np
```

sklearn을 통해 KFold를 import 해준다.

```
fold_iris.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names', 'filename'])
```

간단하게 sklearn을 통해 데이터 셋을 가져오고

```
fold_iris = load_iris()
features = fold_iris.data
# print(features)
label    = fold_iris.target
# print(label)
fold_df_clf = DecisionTreeClassifier()
```

그 부분에서 data(독립변수)와 target(종속변수) 를 가져온다.

그리고 여기서는 DecisionTree를 사용한다.

```
# 5개의 폴드 세트를 분리하여 각 폴드 세트별 정확도를 담을 리스트를 생성
kfold = KFold(n_splits=5)
cv_accuracy = []
```

우리는 KFold를 5개로 split할 예정이다. 이 값은 사용자가 임의로 정할 수 있다.

```
n_iter = 0
for train_idx , test_idx in kfold.split(features) :
    X_train , X_test = features[train_idx] , features[test_idx]

    y_train , y_test = label[train_idx] , label[test_idx]

    # 학습을 진행하겠다면?
```

```
    fold_df_clf.fit(X_train , y_train)
    # 예측
    fold_pred = fold_df_clf.predict(X_test)

    # 정확도 측정
    n_iter += 1
    accuracy = np.round( accuracy_score(y_test , fold_pred) , 4)
    print('\n{} 교차검증 정확도 : {} , 학습 데이터 크기 : {} , 검증 데이터 크기 : {}'.format(n_iter, accuracy

    cv_accuracy.append(accuracy)
print('\n')

print('\n 평균검증 정확도 :  ' , np.mean(cv_accuracy))
```

위에서 분할했던 kfold를 넣고 split을 하는데 우리가 가지고 있던 fold_iris.data를 분할한다. 이것을 train_idx와 test_idx로 분할한다.

이것을 통해 훈련하는 부분과 검증하는 부분을 나눈다. 그렇게 인덱스를 나누고 해당 하는 인덱스를 features와 label에 넣는다.

지금 까지 했던 것을 토대로 예측을 한다. 이 예측은 for문에 의해서 5번 돌게끔 된다.

```
1 교차검증 정확도 : 1.0 , 학습 데이터 크기 : 120 , 검증 데이터 크기 : 30
2 교차검증 정확도 : 1.0 , 학습 데이터 크기 : 120 , 검증 데이터 크기 : 30
3 교차검증 정확도 : 0.8333 , 학습 데이터 크기 : 120 , 검증 데이터 크기 : 30
4 교차검증 정확도 : 0.9333 , 학습 데이터 크기 : 120 , 검증 데이터 크기 : 30
5 교차검증 정확도 : 0.7333 , 학습 데이터 크기 : 120 , 검증 데이터 크기 : 30


 평균검증 정확도 :  0.89998
```

kfold 의 문제점

kfold의 경우 일정한 간격으로 잘라서 사용한다.

여기서보면 답이 0 , 1, 2 이 세가지로 나눠지는데 만약에 이상황에서 잘라서 학습을하는데

0,1만 답으로 가지고 있는 학습데이터를 가지고 학습을 했을때는 2라는 답을 도출 할 수 없다.

0,2, 만 가지고 학습을 했을 경우에 1이라는 답을 도출 할 수 없다. 이것이 Kfold의 문제점이다.

```
In [90]: # 기존 KFold 의 문제점 다시 한번 확인
kfold_iris_data = load_iris()
kfold_iris_data_df = pd.DataFrame(data=kfold_iris_data.data , columns = kfold_iris_data.feature_names)
kfold_iris_data_df['target'] = kfold_iris_data.target
print("value_counts : ₩n", kfold_iris_data_df['target'].value_counts())
```

```
value_counts :
 2    50
1    50
0    50
Name: target, dtype: int64
```

```
In [92]: kfold_iris_data_df.head()
```

Out [92]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

In [100]:
```
kfold_iris = KFold(n_splits=3)
cnt_iter=0
for train_idx, test_idx in kfold_iris.split(kfold_iris_data_df): # 내가 나눌 값을 넣는다.
#     print(train_idx, test_idx)

    #정확도 측정
    cnt_iter +=1
    label_train = kfold_iris_data_df['target'].iloc[train_idx]
    label_test = kfold_iris_data_df['target'].iloc[test_idx]
#     print('label_train₩n', label_train)
    print('교차검증: {}'.format(cnt_iter))
    print('학습 레이블데이터 분포 : ₩n',label_train.value_counts())
    print('검증 레이블데이터 분포 : ₩n',label_test.value_counts())
# print('₩n 평균검증정확도: ',np.mean(cv_accuracy))
```

```
교차검증: 1
학습 레이블데이터 분포 :
 2    50
1    50
Name: target, dtype: int64
검증 레이블데이터 분포 :
 0    50
Name: target, dtype: int64
교차검증: 2
학습 레이블데이터 분포 :
 2    50
0    50
Name: target, dtype: int64
검증 레이블데이터 분포 :
 1    50
Name: target, dtype: int64
교차검증: 3
학습 레이블데이터 분포 :
 1    50
0    50
Name: target, dtype: int64
검증 레이블데이터 분포 :
 2    50
Name: target, dtype: int64
```

이러한 문제를 해결하기 위해 나온것이 stratifiedkFold이다.

stratifiedkFold

stratifiedkFold 는 target에 속성값의 개수를 동일하게 하게 가져감으로써 kfold 같이 데이터가 한곳으로 몰리는것을 방지한다.

```python
# 레이블 값의 분포를 반영해주지 못하는 문제를 해결하기위해서 stratifieldKFold 를 이용
from sklearn.model_selection import StratifiedKFold

skf_iris = StratifiedKFold(n_splits=3)
cnt_iter = 0

for train_idx , test_idx in skf_iris.split(kfold_iris_data_df , kfold_iris_data_df['target']) :
    # print(train_idx , test_idx)
    cnt_iter += 1
    label_train = kfold_iris_data_df['target'].iloc[train_idx]
    label_test  = kfold_iris_data_df['target'].iloc[test_idx]
    # print('label_train\n' , label_train)
    print('교차 검증 : {}'.format(cnt_iter))
    print('학습 레이블 데이터 분포 : \n' , label_train.value_counts())
    print('검증 레이블 데이터 분포 : \n' , label_test.value_counts())
```

```
교차 검증 : 1
학습 레이블 데이터 분포 :
 2    33
1    33
0    33
Name: target, dtype: int64
검증 레이블 데이터 분포 :
 2    17
1    17
0    17
Name: target, dtype: int64
교차 검증 : 2
학습 레이블 데이터 분포 :
 2    33
1    33
0    33
Name: target, dtype: int64
검증 레이블 데이터 분포 :
 2    17
1    17
0    17
Name: target, dtype: int64
교차 검증 : 3
학습 레이블 데이터 분포 :
 2    34
1    34
0    34
Name: target, dtype: int64
검증 레이블 데이터 분포 :
 2    16
1    16
0    16
Name: target, dtype: int64
```

지금 하고있는건 분류에서만 가능하다.

예제)

```
# [실습] random_state = 100
# 붓꽃 데이터 세트에서 Stratified KFold 를 이용하여 교차검증(3, 5)을 진행하고 평균정확도를 확인
# Stratified KFold 는 분류 , 회귀 X
# 회귀는 연속된 숫자 값이기 때문에 지원하지 않는다.


from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold

# 회귀에서는 지원하지 않는다.
from sklearn.model_selection import StratifiedKFold

import pandas as pd
import numpy  as np

result_iris = load_iris()
result_features = result_iris.data
result_label    = result_iris.target

# 학습기 생성
result_clf = DecisionTreeClassifier(random_state=100)
```

```
result_skfold = StratifiedKFold(n_splits=3)
idx_iter=0
cv_accuracy=[]

# StratifiedKFold의 split( ) 호출시 반드시 레이블 데이터 셋도 추가 입력 필요
for train_index, test_index  in result_skfold.split(features, label):
    # split( )으로 반환된 인덱스를 이용하여 학습용, 검증용 테스트 데이터 추출
    X_train, X_test = features[train_index], features[test_index]
    y_train, y_test = label[train_index], label[test_index]
```

```
    #학습 및 예측
    result_clf.fit(X_train , y_train)
    pred = result_clf.predict(X_test)

    # 반복 시 마다 정확도 측정
    idx_iter += 1
    accuracy = np.round(accuracy_score(y_test,pred), 4)
    train_size = X_train.shape[0]
    test_size = X_test.shape[0]

    print('\n#{0} 교차 검증 정확도 :{1}, 학습 데이터 크기: {2}, 검증 데이터 크기: {3}'
          .format(idx_iter, accuracy, train_size, test_size))
    print('#{0} 검증 세트 인덱스:{1}'.format(idx_iter,test_index))
    cv_accuracy.append(accuracy)
```

```
#1 교차 검증 정확도 :0.9804, 학습 데이터 크기: 99, 검증 데이터 크기: 51
#1 검증 세트 인덱스:[  0   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15
16  50
  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66 100 101
 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116]

#2 교차 검증 정확도 :0.9216, 학습 데이터 크기: 99, 검증 데이터 크기: 51
#2 검증 세트 인덱스:[ 17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32
33  67
  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83 117 118
 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133]

#3 교차 검증 정확도 :0.9792, 학습 데이터 크기: 102, 검증 데이터 크기: 48
#3 검증 세트 인덱스:[ 34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49
84  85
  86  87  88  89  90  91  92  93  94  95  96  97  98  99 134 135 136 137
 138 139 140 141 142 143 144 145 146 147 148 149]
```

```
# 교차 검증별 정확도 및 평균 정확도 계산
print('\n## 교차 검증별 정확도:', np.round(cv_accuracy, 4))
print('## 평균 검증 정확도:', np.mean(cv_accuracy))
```

```
## 교차 검증별 정확도: [0.9804 0.9216 0.9792]
## 평균 검증 정확도: 0.9604
```

'ML,DL' 카테고리의 다른 글□

[ML/DL] 정밀도와 재현율의 트레이드 오프 정의와 구현□

[ML/DL] python 을 통한 분류(classification) 성능평가지표 사용법(Accuracy,Pre...

[ML/DL] python 을 통한 교차검증 (k -Fold , stratifiedkFold)□

[ML/DL] python 을 통한 결측값 확인 및 결측치 처리 방법□

[ML/DL] 데이터 인코딩 - Label Encoding / One-hot Encoding/ dummies□

[ML/DL] 파이썬(python)을 이용한 분류(Classification)하기□

k fold k-fold StratifiedKFold 교차검증 교차검증 k-fold 교차검증 kfold

나아무늘보

혼자 끄적끄적하는 블로그 입니다.