

머신러닝 단순교차검증

scikit-learn 의 `train_test_split()` 함수를 사용하여

데이터를 훈련 세트와 테스트 세트로 한 번 나누는 것보다 더 성능이 좋은 평가방법은

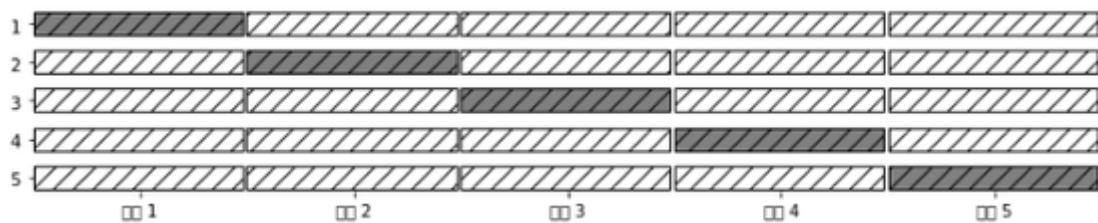
교차검증 Cross-Validation

이다.

k-겹 교차검증에서

k 에는 5 or 10 과 같은 숫자가 들어가며,

데이터를 비슷한 크기의 집합 'k 개'로 나눈다. 이를 fold 라고 한다.



예를 들어, 5-겹 교차검증일 경우에는

데이터를 5 개로 분할한 다음 첫 번째 폴드를 테스트 데이터로 사용하고

나머지 2~5 폴드는 훈련용으로 사용하여 정확도를 평가한다.

그 다음, 두 번째 폴드를 테스트용으로 사용하고

1, 3~5 폴드를 훈련용으로 사용한다.

이런 식으로 폴드 1,2,3,4,5 를 각각 테스트용으로 사용하는데,

각각의 분할마다 정확도를 측정하고 이를 평균 내어 값을 구한다.

교차 검증을 위해 `cross_val_score` 함수를 불러오고,
사용할 데이터셋은 scikit-learn 의 유방암 데이터셋이다.

선형회귀와

KNN,

SVM,

의사결정트리,

랜덤포레스트 모델로

데이터를 학습시키고, 교차검증으로 정확도를 평가해보겠다.

```

# 교차 검증
from sklearn.model_selection import cross_val_score

# 유방암
from sklearn.datasets import load_breast_cancer

# 선형회귀
from sklearn.linear_model import LinearRegression

# KNN
from sklearn.neighbors import KNeighborsClassifier

# SVM
from sklearn.svm import LinearSVC

# 의사결정트리
from sklearn.tree import DecisionTreeClassifier

# 랜덤포레스트
from sklearn.ensemble import RandomForestClassifier

```

```
cancer = load_breast_cancer()
```

Variable explorer			
Name	Type	Size	Value
cancer	utils.Bunch	6	Bunch object of sklearn.utils module

모델의 매개변수는 되도록이면 기본값인 상태로 진행한다.

각각의 모델들을 불러온 후, lr, knn, svm, tree, forest 에 저장한다.

매개변수는 되도록이면 기본값으로 설정.

선형회귀 학습모델

```
lr = LinearRegression()
```

```

IPython console
Console 1/A
In [81]: lr
Out[81]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)

```

KNN 학습모델

```
knn = KNeighborsClassifier(n_neighbors=4)
```

```
IPython console
Console 1/A ✕

In [82]: knn
Out[82]:
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=4, p=2,
                     weights='uniform')
```

SVM 학습모델

```
svm = LinearSVC(random_state=0)
```

```
IPython console
Console 1/A ✕

In [83]: svm
Out[83]:
LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
          intercept_scaling=1, loss='squared_hinge', max_iter=1000,
          multi_class='ovr', penalty='l2', random_state=0, tol=0.0001,
          verbose=0)
```

의사결정트리 학습모델

```
tree = DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
IPython console
Console 1/A ✕

In [85]: tree
Out[85]:
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=3,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

랜덤포레스트 학습모델

```
forest = RandomForestClassifier(n_estimators=6)
```

```
IPython console
Console 1/A ✕

In [84]: forest
Out[84]:
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto',
                       max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=6,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
# 선형회귀 학습 후, 교차검증
score1 = cross_val_score(lr, cancer.data, cancer.target)

# KNN 학습 후, 교차검증
score2 = cross_val_score(knn, cancer.data, cancer.target)

# SVM 학습 후, 교차검증
score3 = cross_val_score(svm, cancer.data, cancer.target)

# 의사결정트리 학습 후, 교차검증
score4 = cross_val_score(tree, cancer.data, cancer.target)

# 랜덤포레스트 학습 후, 교차검증
score5 = cross_val_score(forest, cancer.data, cancer.target)
```

Variable explorer			
Name	Type	Size	Value
cancer	utils.Bunch	6	Bunch object of sklearn.utils module
score1	float64	(3,)	[0.68274692 0.74112749 0.68552679]
score2	float64	(3,)	[0.90526316 0.95263158 0.9047619]
score3	float64	(3,)	[0.89473684 0.91578947 0.93121693]
score4	float64	(3,)	[0.9 0.95789474 0.89417989]
score5	float64	(3,)	[0.92105263 0.96842105 0.94179894]

교차 검증의 결과 확인

```
print('선형회귀 교차검증 점수 : {:.2f}'.format(score1.mean()))
print('KNN 교차검증 점수 : {:.2f}'.format(score2.mean()))
print('SVM 교차검증 점수 : {:.2f}'.format(score3.mean()))
print('의사결정트리 교차검증 점수 : {:.2f}'.format(score4.mean()))
print('랜덤포레스트 교차검증 점수 : {:.2f}'.format(score5.mean()))
```

```
IPython console
Console 1/A

In [91]: print('선형회귀 교차검증 점수 : {:.2f}'.format(score1.mean()))
선형회귀 교차검증 점수 : 0.70

In [92]: print('KNN 교차검증 점수 : {:.2f}'.format(score2.mean()))
KNN 교차검증 점수 : 0.92

In [93]: print('SVM 교차검증 점수 : {:.2f}'.format(score3.mean()))
SVM 교차검증 점수 : 0.91

In [94]: print('의사결정트리 교차검증 점수 : {:.2f}'.format(score4.mean()))
의사결정트리 교차검증 점수 : 0.92

In [95]: print('랜덤포레스트 교차검증 점수 : {:.2f}'.format(score5.mean()))
랜덤포레스트 교차검증 점수 : 0.94
```

Linear Regression 의 정확도는 70%,
KNN 은 92%,
SVM 은 91%,
DecisionTree 는 92%,
RandomForest 는 96%가 나왔다.

그렇다면, 다른 모든 조건을 그대로 유지하되
교차검증이 아닌 train_test_split()함수로 훈련셋과 테스트셋을 한 번만 나눠보겠다.

```
from sklearn.model_selection import train_test_split
```

```
X_train, X_test, Y_train, Y_test = train_test_split(cancer.data,  
                                                    cancer.target,  
                                                    random_state=0)
```

Variable explorer			
Name	Type	Size	Value
X_test	float64	(143, 30)	[[1.340e+01 2.052e+01 8.864e+0... [1 ...
X_train	float64	(426, 30)	[[1.185e+01 1.746e+01 7.554e+0... [1 ...
Y_test	int32	(143,)	[0 1 1 ... 1 1 0]
Y_train	int32	(426,)	[1 1 0 ... 1 1 1]

```
lr = LinearRegression().fit(X_train, Y_train)  
knn = KNeighborsClassifier(n_neighbors=4).fit(X_train, Y_train)  
svm = LinearSVC(random_state=0).fit(X_train, Y_train)  
tree = DecisionTreeClassifier(max_depth=3,  
                              random_state=0).fit(X_train, Y_train)  
forest = RandomForestClassifier(n_estimators=6).fit(X_train, Y_train)
```

결과 확인

```
print('선형회귀 정확도 : {:.2f}'.format(lr.score(X_test, Y_test)))  
print('KNN 정확도 : {:.2f}'.format(knn.score(X_test, Y_test)))  
print('SVM 정확도 : {:.2f}'.format(svm.score(X_test, Y_test)))  
print('의사결정트리 정확도 : {:.2f}'.format(tree.score(X_test, Y_test)))  
print('랜덤포레스트 정확도 : {:.2f}'.format(forest.score(X_test, Y_test)))
```

```
IPython console
Console 1/A ✕

In [106]: print('선형회귀 정확도 : {:.2f}'.format(lr.score(X_test, Y_test)))
선형회귀 정확도 : 0.73

In [107]: print('KNN 정확도 : {:.2f}'.format(knn.score(X_test, Y_test)))
KNN 정확도 : 0.92

In [108]: print('SVM 정확도 : {:.2f}'.format(svm.score(X_test, Y_test)))
SVM 정확도 : 0.87

In [109]: print('의사결정트리 정확도 : {:.2f}'.format(tree.score(X_test, Y_test)))
의사결정트리 정확도 : 0.94

In [110]: print('랜덤포레스트 정확도 : {:.2f}'.format(forest.score(X_test, Y_test)))
랜덤포레스트 정확도 : 0.95
```

성능이 좋아진 것도 있고 나빠진 것도 있다.

train_test_split 은 데이터를 무작위로 나누는데
이럴 경우 무작위로 나뉘어진 셋에
어떤 데이터들이 담기느냐에 따라서 정확도가 높고 낮아질 수 있다.

그러나, 교차검증은
각 폴드가 한 번씩 테스트 세트가 되므로
train_test_split 보다 데이터가 편향될 확률은 낮다