모델 평가

학습 내용

- 여러가지 검증 방법에 대해 알아본다.
- 교차 검증에 대해 알아본다.

목차

In [1]:

```
import os, warnings
import numpy as np
# 경고 메시지 무시하거나 숨길때(ignore), 다시보이게(default)
# warnings.filterwarnings(action='default')
warnings.filterwarnings(action='ignore')
```

01. 교차 검증에 대해 알아보기

목차로 이동하기

- 훈련 세트와 테스트 세트로 한번 나누는 것보다 더 안정적이고 뛰어난 통계적 평가 방법
- 데이터를 여러번 반복해서 나누고 여러 모델을 학습
- 가장 널리 쓰이는 교차 검증 방법은 k-겹 교차 검증(k-fold cross-validation)
- 보통 5또는 10을 사용한다.

02. 교차 검증 실습

목차로 이동하기

- 기본값은 5겹 교차 검증이다. (sklearn 0.22부터 3->5로 변경)

In [2]:

```
from sklearn.model_selection import cross_val_score
from sklearn.datasets import load_boston
from sklearn.linear_model import LinearRegression
import sklearn
import pandas as pd
import mglearn

print(sklearn.__version__)
```

0.23.2

In [3]:

```
boston = load_boston()
df = pd.DataFrame(boston.data, columns=boston.feature_names)
df['price'] = boston.target
print(df.shape)
```

(506, 14)

In [4]:

```
df.head()
```

Out[4]:

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LS
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	

In [5]:

```
X = df.drop(['price'], axis=1)
y = df['price']
```

- scoring='neg_mean_squared_error' : 반환되는 값이 음수이다.
  - 평가 지표 : NMSE(음수로 표현됨)

In [6]:

```
lr_model = LinearRegression()
neg_mse_scores = cross_val_score(lr_model, X, y,
                                 scoring='neg_mean_squared_error', cv=5 )

rmse = np.sqrt(-1 * neg_mse_scores)

print(rmse)
print("평균 RMSE : {0:.3f}".format( np.mean(rmse) ) )
```

```
[3.52991509 5.10378498 5.75101191 8.9867887  5.77179405]
평균 RMSE : 5.829
```

03. 교차 검증의 장단점

목차로 이동하기

교차 검증의 장점

- 첫째, 데이터를 전반적으로 검증하는 **여러개의 모델을 사용**하기에 일반화된 모델을 생성할 수 있다.
- 둘째, 분할을 한번하는 것보다 **데이터를 더 효율적으로 사용 가능.**

교차 검증의 단점

- 주요 단점은 연산 비용이 늘어남. 모델을 k개를 만들어야 하므로 데이터를 한번 나눴을 때보다 k배가 더 느림.

일반적인 방법

- 분류에는 **StratifiedKFold를 사용**하며, **계층별 K-겹 교차 검증의 기본값**이 잘 동작
- 회귀에는 단순한 KFold를 적용(k-겹 교차 검증)
  - KFold에서 shuffle 매개변수를 **기본값 False 대신 True를 지정**하면 폴더를 나누기 전에 무작위 섞는 것이 가능.
- cross_val_score함수를 사용시에는 KFold의 매개변수를 제어가 안되기에, KFold 객체를 만들어 cross_val_score함수의 cv의 매개변수로 전달해야 한다.
- model_selection 모듈에서 KFold 분할기를 임포트하고 원하는 폴더 수를 넣어 객체를 생성.

04. LOOCV (Leave-one-out cross-validation)

목차로 이동하기

- **폴드 하나에 샘플 하나**만 들어있는 k-겹 교차 검증
- 각 반복에 **하나의 데이터 포인트**를 선택해 **테스트 데이터 세트**로 사용.
- 데이터 셋이 클 경우 시간이 매우 오래 걸리지만, **작은 데이터 셋에서 이따금 좋은 결과**를 만든다.

In [7]:

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

In [8]:

```
loo = LeaveOneOut()
iris = load_iris()
tree = DecisionTreeClassifier()
scores = cross_val_score(tree, iris.data, iris.target, cv=loo)

print("교차 검증 분할 횟수 : ", len(scores))
print("평균 정확도 : {:.2f}".format(scores.mean()))
```

```
교차 검증 분할 횟수 :  150
평균 정확도 : 0.95
```

- iris의 경우는 150개의 데이터이므로 150번의 교차 검증

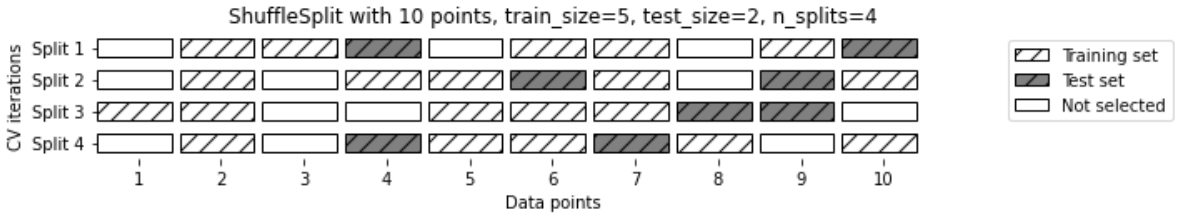
05. 임의 분할 교차 검증

목차로 이동하기

- Shuffle-split cross-validation
  - train_size 만큼 샘플로 학습용 세트를 만들고,
  - test_size만큼(중복되지 않음)의 테스트 세트를 만든다.
  - n_splits 횟수만큼 반복

In [9]:

```
mglearn.plots.plot_shuffle_split()
```



- 샘플이 10개, 5개는 학습용, 2개는 테스트용으로 4번 반복

실습

- 샘플이 150개
  - 10개를 훈련 셋으로 5개(중복안됨)을 테스트 데이터 셋으로 하여 10번 반복 분할

In [10]:

```
from sklearn.model_selection import ShuffleSplit
```

In [11]:

```
shuffle_split = ShuffleSplit(train_size=10, test_size=5,  n_splits=10)

scores = cross_val_score(tree, iris.data, iris.target,
                         cv=shuffle_split)

print("교차 검증 점수 : ₩n{}".format(scores))
avg = scores.mean()
print("평균 : {:.2f}".format( avg )  )
```

```
교차 검증 점수 :
[1.  0.6 1.  0.8 0.4 1.  1.  1.  1.  1. ]
평균 : 0.88
```

샘플이 150개의 50%를 훈련 셋으로 50%를 테스트 데이터 셋으로 하여 10번 반복 분할

In [12]:

```
from sklearn.model_selection import ShuffleSplit
shuffle_split = ShuffleSplit(test_size=0.5, train_size=0.5, n_splits=10)

scores = cross_val_score(tree, iris.data, iris.target, cv=shuffle_split)
print("교차 검증 점수 : ₩n{}".format(scores))
scores.mean()
```

```
교차 검증 점수 :
[0.93333333 0.96       0.93333333 0.94666667 0.97333333 0.94666667
 0.93333333 0.86666667 0.94666667 0.90666667]
```

Out[12]:

```
0.9346666666666665
```

- 임의 분할 교차 검증은 반복 횟수를 훈련 세트나 테스트 세트의 크기와 독립적으로 조절해야 할 때 유용.
- 또한 train_size와 test_size의 합을 전체와 다르게 함으로써 전체 데이터의 일부만 사용할 수 있습니다.

ShuffleSplit 계층별 버전으로 StratifiedShuffleSplit 도 있음.

06. 반복 교차 검증

목차로 이동하기

- 교차 검증을 반복하여 여러번 수행하는 경우가 많다.
  - 이를 위해 scikit-learn 0.19버전에서 RepeatedKFold와 RepeatedStratifiedKFold 분할기가 추가.

- 회귀 문제 : RepeatedKFold의 KFold 클래스 사용
- 분류 문제 : RepeatedStratifiedKFold의 StratifiedKFlod 클래스 사용
  - 분할 폴드 수는 n_splits 매개변수로 설정하며 기본값은 5이다.
  - 반복 횟수는 n_repeats 매개변수로 설정. 기본값은 10이다.

In [13]:

```
from sklearn.model_selection import cross_val_score, KFold, StratifiedKFold
from sklearn.datasets import load_iris
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import RepeatedStratifiedKFold
```

In [14]:

```
iris = load_iris()
logreg = LogisticRegression()
```

In [15]:

```
# n_split : 5    분할 폴드 수(기본값 : 5)
# n_repeat = 10  반복횟수(기본값 : 10)
# Point : 반복할때마다 데이터를 다시 섞습니다.
rskfold = RepeatedStratifiedKFold(random_state=42, n_splits=5, n_repeats=10)
scores = cross_val_score(logreg, iris.data, iris.target, cv=rskfold)

print("횟수 : ", len(scores))
print("교차 검증 점수 : ₩n",  scores)
print("교차 검증 평균 점수 : {:.3f}".format(scores.mean() ) )
```

```
횟수 :  50
교차 검증 점수 :
 [1.         0.96666667 0.93333333 1.         0.93333333 0.96666667
 0.96666667 0.93333333 1.         0.96666667 0.93333333 1.
 1.         0.96666667 0.96666667 0.9        1.         1.
 0.93333333 0.96666667 0.93333333 0.96666667 0.96666667 1.
 0.96666667 1.         0.96666667 0.96666667 0.9        1.
 0.96666667 0.96666667 0.96666667 0.96666667 0.93333333 0.96666667
 0.96666667 1.         1.         0.9        0.96666667 1.
 0.9        0.96666667 0.96666667 0.9        0.96666667 0.96666667
 1.         0.96666667]
교차 검증 평균 점수 : 0.965
```

- 전체 검증 점수는 n_splits X n_repeats개수만큼 만들어진다. 기본값을 사용하여 만들경우 5 x 10 = 50개의 모델이 만들어진다.

실습 : RandomForestClassifier를 사용해 보자.

07. 기타 검증 (GroupKFold)

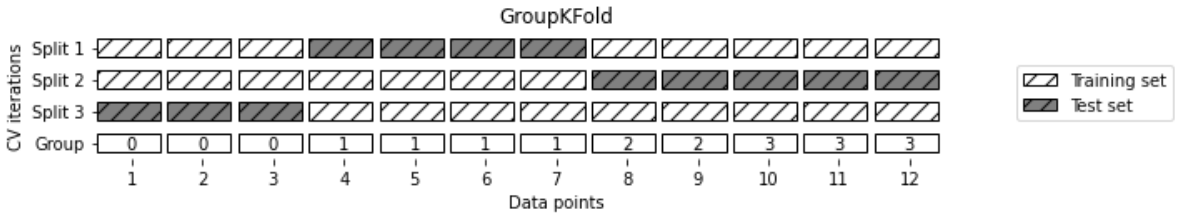
목차로 이동하기

- GroupKFold : 그룹별 교차 검증

- 인위적으로 만든 데이터 셋에 groups 배열로 그룹을 지정한다.
- 데이터의 그룹이 있는 예로는 의료 애플리케이션이 일반적이다.
  - 같은 환자로부터 여러 샘플을 가지고 새로운 환자에게 일반화시킨다.

In [16]:

```
mglearn.plots.plot_group_kfold()
```



- 사진을 찍어, 각 사람별로 그룹지어 이를 테스트, 훈련 세트에 활용

In [17]:

```
from sklearn.model_selection import GroupKFold
from sklearn.datasets import make_blobs

X, y = make_blobs(n_samples=12, random_state=0)
print(X.shape, y.shape)

# 처음 세 개의 샘플은 같은 그룹에 속하고,
# 다음은 네 개의 샘플이 같습니다.
groups = [0,0,0, 1,1,1,1, 2,2, 3,3,3] # 4개 그룹

scores = cross_val_score(logreg, X, y, groups, cv=GroupKFold(n_splits=3))
print("교차 검증 점수 : ₩n", scores)
```

```
(12, 2) (12,)
교차 검증 점수 :
 [0.75       0.6        0.66666667]
```

그룹 분할 확인

In [18]:

```
gkf = GroupKFold(n_splits=3)
groups = [0,0,0, 1,1,1,1, 2,2, 3,3,3]
X, y = make_blobs(n_samples=12, random_state=0)

for train, test in gkf.split(X, y, groups=groups):
    print("%s %s" % (train, test))
```

```
[ 0  1  2  7  8  9 10 11] [3 4 5 6]
[0 1 2 3 4 5 6] [ 7  8  9 10 11]
[ 3  4  5  6  7  8  9 10 11] [0 1 2]
```

In [19]:

```
gkf = GroupKFold(n_splits=4)
groups = [0,0,0, 1,1,1,1, 2,2, 3,3,3]
X, y = make_blobs(n_samples=12, random_state=0)

for train, test in gkf.split(X, y, groups=groups):
    print("%s %s" % (train, test))
```

```
[ 0  1  2  7  8  9 10 11] [3 4 5 6]
[0 1 2 3 4 5 6 7 8] [ 9 10 11]
[ 3  4  5  6  7  8  9 10 11] [0 1 2]
[ 0  1  2  3  4  5  6  9 10 11] [7 8]
```

In [20]:

```
from sklearn.model_selection import GroupKFold

X = [0.1, 0.2, 2.2, 2.4, 2.3, 4.55, 5.8, 8.8, 9, 10]
y = ["a", "b", "b", "b", "c", "c", "c", "d", "d", "d"]
groups = [1, 1, 1, 2, 2, 2, 3, 3, 3, 3]

gkf = GroupKFold(n_splits=3)
for train, test in gkf.split(X, y, groups=groups):
    print("%s %s" % (train, test))
```

```
[0 1 2 3 4 5] [6 7 8 9]
[0 1 2 6 7 8 9] [3 4 5]
[3 4 5 6 7 8 9] [0 1 2]
```

변경

- 2021/10/19 GroupKFold 수정 추가 및 업데이트
- 2022/08 최종 업데이트