모델 평가

학습 내용

- 교차 검증에 대해 알아봅니다.

01 교차 검증에 대해 알아보기

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

In [1]:

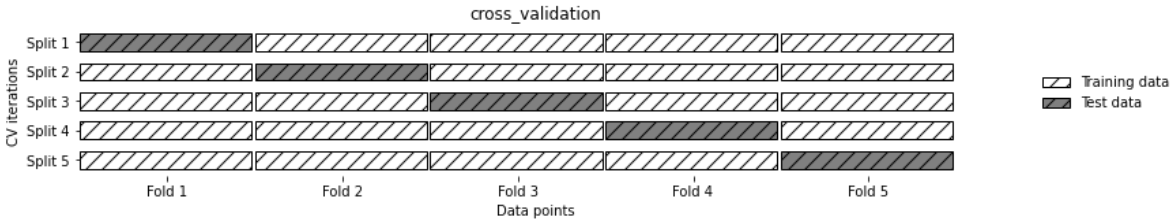
```
import os, warnings
import numpy as np
# 경고 메시지 무시하거나 숨길때(ignore), 다시보이게(default)
# warnings.filterwarnings(action='default')
warnings.filterwarnings(action='ignore')
```

In [2]:

```
import mglearn
```

In [3]:

```
mglearn.plots.plot_cross_validation()
```



교차 검증 실습

- sklearn의 21버전은 cv(k폴더의 수)가 3으로 기본 지정
- sklearn의 22버전부터는 cv가 기본이 5으로 지정

In [7]:

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
```

In [8]:

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

- cross_val_score(평가모델, X, y=None, scoring=None, cv=None, n_jobs=None ...

In [9]:

```
scores = cross_val_score(logreg, iris.data, iris.target)
print("교차 검증 점수 : {}".format(scores))
```

```
교차 검증 점수 : [0.96666667 1.         0.93333333 0.96666667 1.        ]
```

실습해 보기 : cv의 매개변수를 이용하여 폴더의 수를 3로 하여 실습해 보기

In [12]:

```
scores = cross_val_score(logreg, iris.data, iris.target, cv=3)
print("교차 검증 점수 : {}".format(scores))
```

```
교차 검증 점수 : [0.98 0.96 0.98]
```

In [13]:

```
print("교차 검증 점수 : {:.2f}".format(scores.mean()))
```

```
교차 검증 점수 : 0.97
```

교차 검증의 장점

- 데이터를 무작위로 나눌 때 운 좋게 학습용 세트에는 분류하기 어려운 샘플이 담길 수 있음.
  - 이경우 테스트 세트에 분류에 좋은 샘플이 담긴다면 좋은 정확도가 얻어질 것임.
- 반대로 훈련세트에 분류가 쉽고, 테스트의 세트에 분류가 어려운 샘플이 담긴다면,
  - 이경우, 테스트 세트의 정확도는 낮은 결과가 나올 것임

첫번째 장점 : 일반화된 모델을 생성할 수 있음.

두번째 장점 : 분할을 한번 했을 때보다 데이터를 더 효과적으로 사용이 가능함

교차 검증의 단점

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

02 계층별 k-겹 교차 검증에 대해 알아보기

- 데이터셋을 나열 순서대로 k개의 폴드로 나누는 것은 항상 좋지 않음.

In [14]:

```
from sklearn.datasets import load_iris
iris = load_iris()
print("iris 레이블 :₩n{}".format(iris.target))
```

```
iris 레이블 :
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

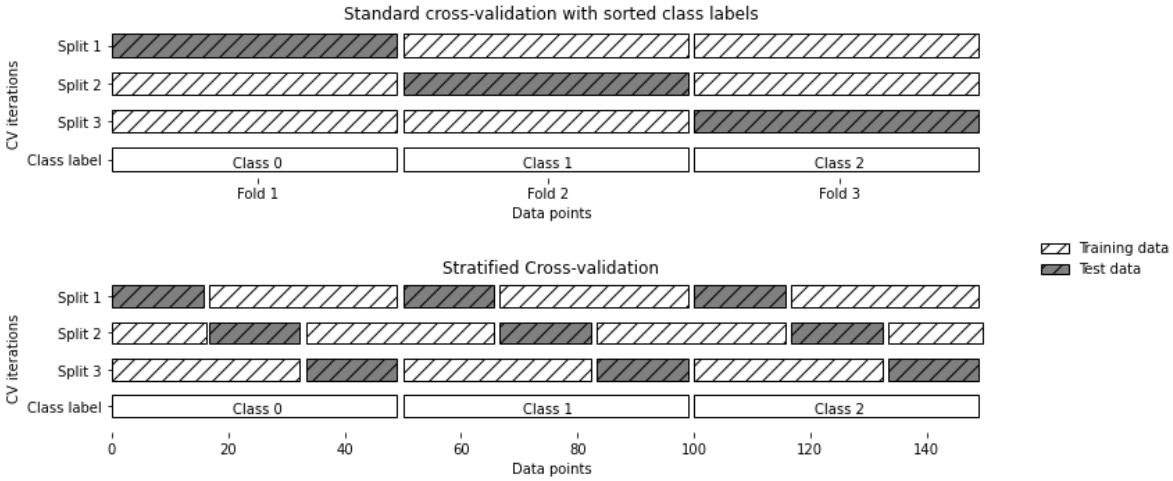
- 위의 데이터를 순서대로 나눌 경우,편향이 발생함.

단순한 k-겹 교차 검증은 문제가 발생. scikit-learn에서 계층별 교차 검증을 사용.

- 계층별 교차 검증 : stratified k-fold cross-validation

In [15]:

```
mglearn.plots.plot_stratified_cross_validation()
```



- 폴더 안에 클래스의 비율이 같도록 데이터를 나눈다.

- 대부분의 경우 회귀에서는 k-겹 교차 검증
- 분류에서는 계층별 K-겹 교차 검증의 기본값이 잘 동작

- model_selection 모듈에서 KFold 분할기를 임포트하고 원하는 폴더 수를 넣어 객체를 생성.

iris 데이터 셋에서 3겹 교차 검증을 사용할 때,

- 점수가 0이 됨.

In [18]:

```
kfold = KFold(n_splits=3)
print("교차 검증 점수 : ₩n{}".format(cross_val_score(logreg,
                                            iris.data,
                                            iris.target, cv=kfold)))
```

```
교차 검증 점수 :
[0. 0. 0.]
```

iris 데이터 셋에서 3겹 교차 검증을 사용할 때,

- 계층별 폴더를 만드는 대신에 샘플의 순서를 뒤죽박죽으로 섞기(shuffle=True 로 준다.)

In [19]:

```
kfold = KFold(n_splits=3, shuffle=True, random_state=0)
print("교차 검증 점수 : ₩n{}".format(cross_val_score(logreg,
                                            iris.data,
                                            iris.target, cv=kfold)))
```

```
교차 검증 점수 :
[0.98 0.96 0.96]
```

In [20]:

```
from sklearn.model_selection import KFold
kfold = KFold(n_splits=5)
```

In [21]:

```
print("교차 검증 점수 : ₩n{}".format(cross_val_score(logreg,
                                            iris.data,
                                            iris.target, cv=kfold)))
```

```
교차 검증 점수 :
[1.         1.         0.86666667 0.93333333 0.83333333]
```

과제

- kaggle 또는 공개 데이터 셋을 이용하여 모델을 만들고, CrossValidation을 이용하여 모델을 만들고 이를 소스코드로 제출해 주세요

In [ ]:

```

```