k-fold cross validation for finding parameters and measuring accuracy

Deciding on an evaluation protocol

- Common evaluation protocols
  - Maintaining a hold-out validation set—The way to go when you have plenty of data
  - Doing K-fold cross-validation—The right choice when you have too few samples for hold-out validation to be reliable
  - Doing iterated K-fold validation—For performing highly accurate model evaluation when little data is available

Evaluating your hypothesis

dataset

	Size	Price
60%	2104	400
	1600	330
	2400	369
	1416	232
	3000	540
	1985	300
20%	1534	315
	1427	199
20%	1380	212
	1494	243

training set

$$(x^{(1)}, y^{(1)})$$
$$(x^{(2)}, y^{(2)})$$
$$\vdots$$
$$(x^{(m)}, y^{(m)})$$

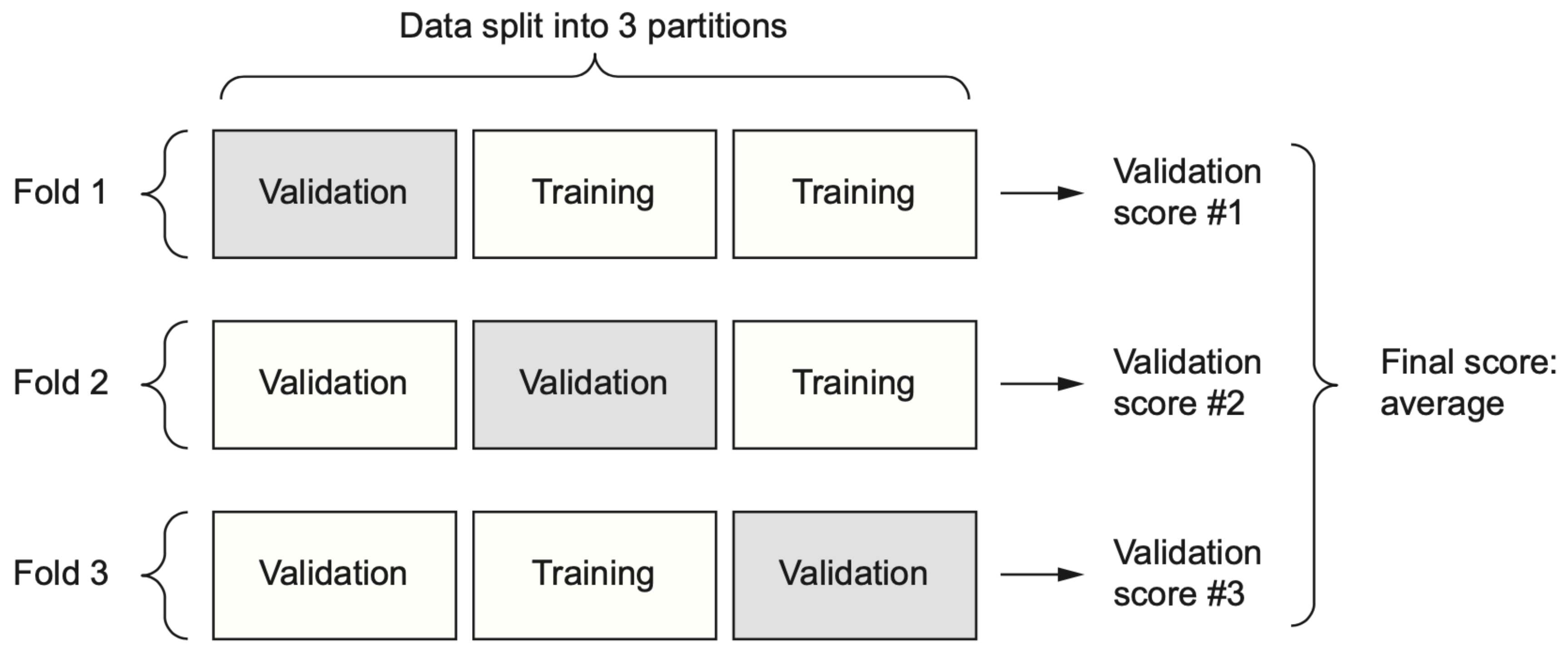
cross-validation set

$$(x_{cv}^{(1)}, y_{cv}^{(1)})$$
$$(x_{cv}^{(2)}, y_{cv}^{(2)})$$
$$\vdots$$
$$(x_{cv}^{(m_{cv})}, y_{cv}^{(m_{cv})})$$

test set

$$(x_{test}^{(1)}, y_{test}^{(1)})$$
$$(x_{test}^{(2)}, y_{test}^{(2)})$$
$$\vdots$$
$$(x_{test}^{(m_{test})}, y_{test}^{(m_{test})})$$

k-fold cross validation for measuring accuracy

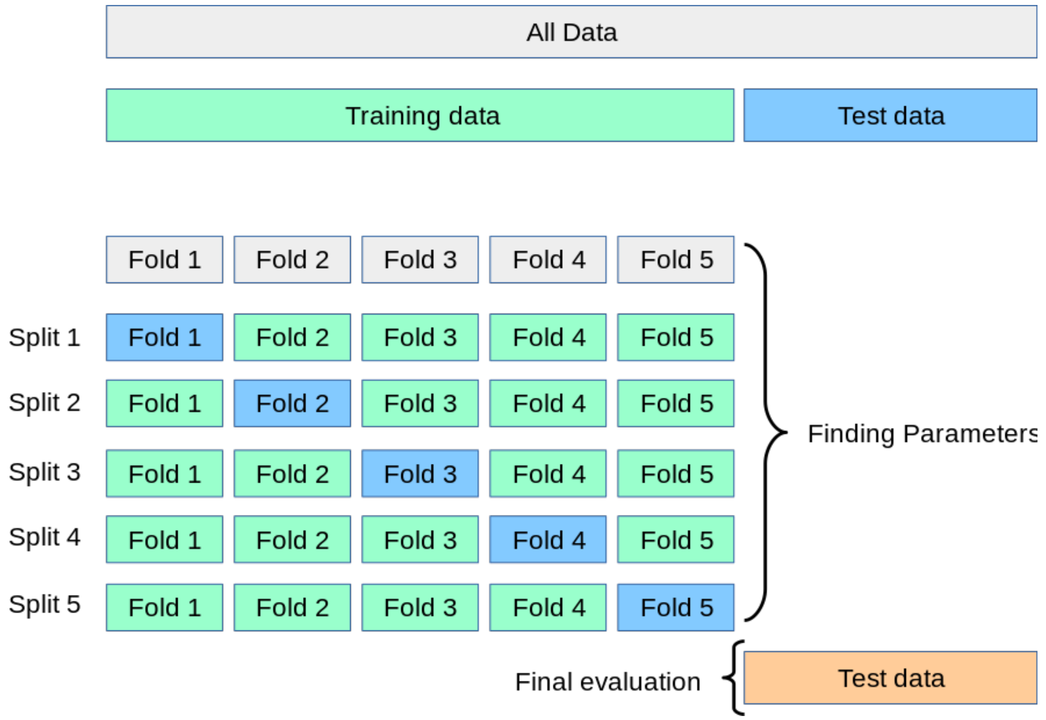


https://scikit-learn.org/stable/modules/cross_validation.html

k-fold cross validation, the training set is split into k smaller sets The following procedure is followed for each of the k “folds”:

- A model is trained using k–1 of the folds as training data;
- the resulting model is validated on the remaining part of the data (i.e., it is used as a test set to compute a performance measure such as accuracy).

The performance measure reported by k-fold cross-validation is then the average of the values computed in the loop. This approach can be computationally expensive, but does not waste too much data (as is the case when fixing an arbitrary validation set), which is a **major advantage in problems where the number of samples is very small**



k-fold cross validation in scikit-learn for measuring accuracy

```
import numpy as np
from sklearn.model_selection import train_test_split, cross_val_score

from sklearn import datasets
from sklearn import svm

X, y = datasets.load_iris(return_X_y=True)

# train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.4, random_state=0)

clf = svm.SVC(kernel='linear', C=1).fit(X_train, y_train)
clf.score(X_test, y_test)
>>> 0.96

clf = svm.SVC(kernel='linear', C=1)
scores = cross_val_score(clf, X, y, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))

>>> Accuracy: 0.98 (+/- 0.03)
```