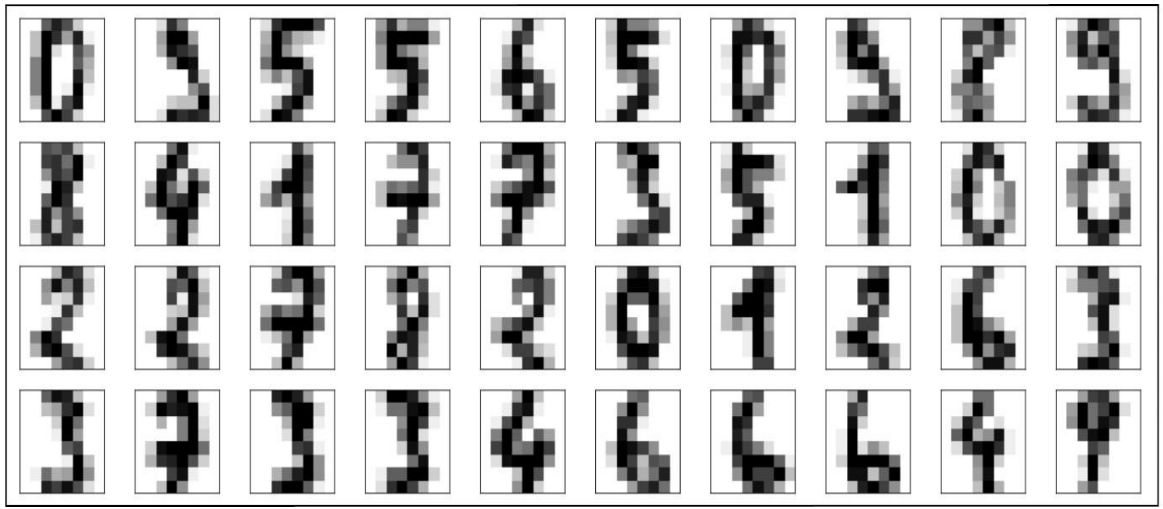
Validation of a model

Train-test split and Cross-validation

Validation of a model - full dataset

```
from sklearn.datasets import load_digits

digits = load_digits()
data = digits.images[30:70].reshape((4, 10, -1))
```

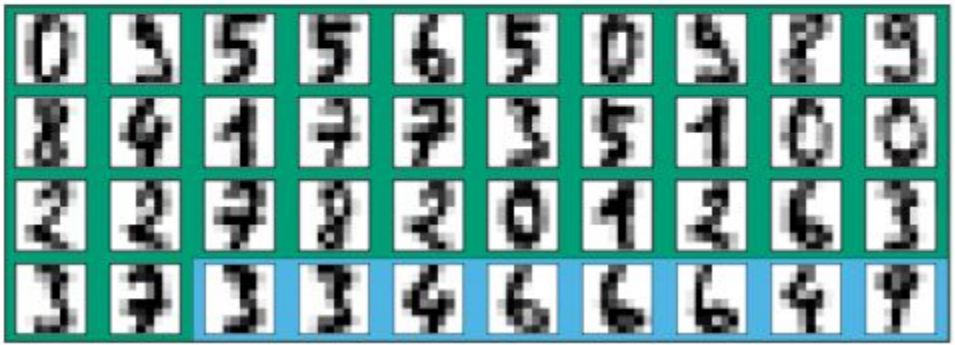


Goal: To evaluate the **generalization performance** of a model.

Validation of a model - Train-test split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    data, target, test_size=0.2, shuffle=False)
```



Training data

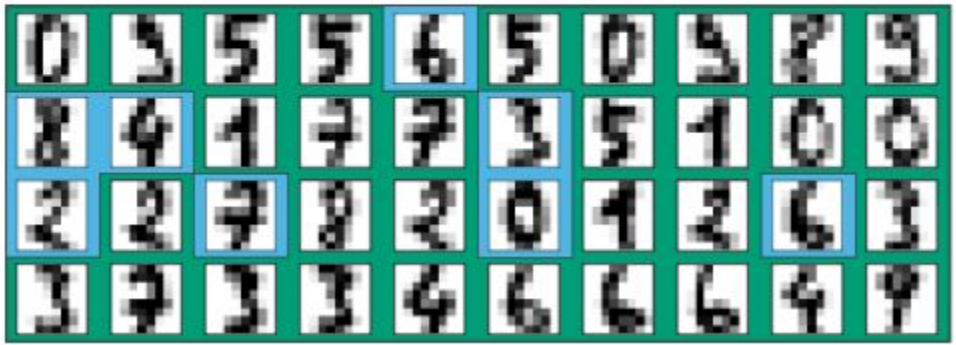
Testing data

The test accuracy using a “**LogisticRegression()**” is 0.875

Validation of a model - Train-test split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    data, target, test_size=0.2, random_state=0, shuffle=True)
```



Training data

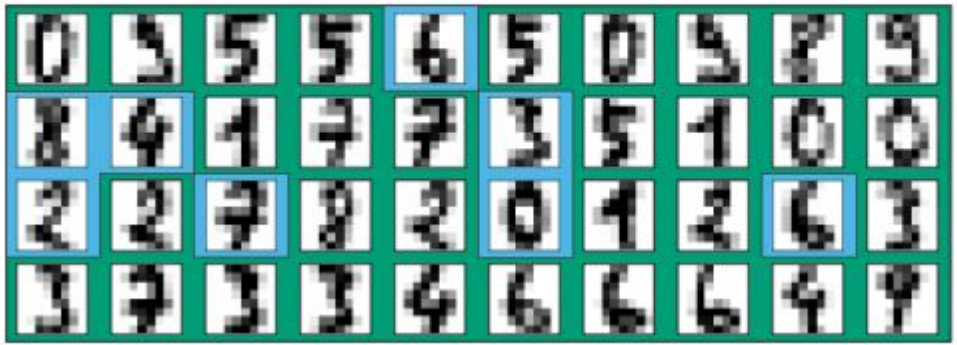
Testing data

The test accuracy using a “**LogisticRegression()”** is 1.00

Validation of a model - Train-test split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    data, target, test_size=0.2, random_state=0, shuffle=True)
```



Training data

Testing data

The test accuracy using a “**LogisticRegression()”** is 1.00

Validation of a model - Train-test split

- In general, the score of a model depends on the split:
  - the train-test proportion
  - the representativeness of the elements in each set

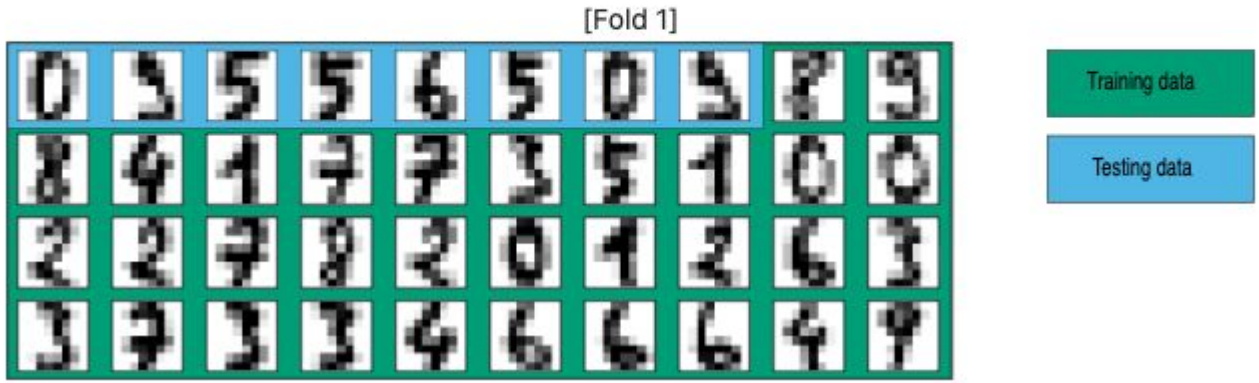
- A more systematic way of evaluating the generalization performance of a model is through cross-validation

- Cross-validation consists of repeating the split such that the training and testing sets are different for each evaluation.

Validation of a model - Cross-validation - 1

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, shuffle=False)
```

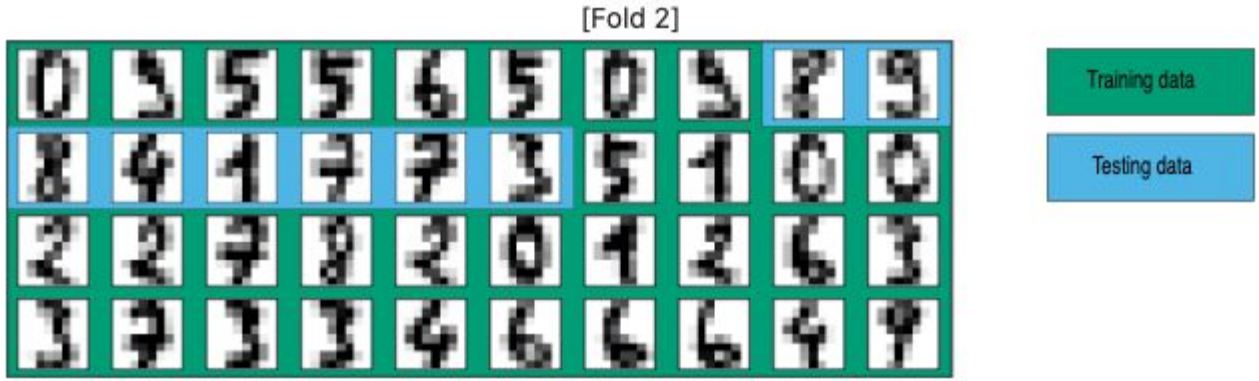


The test accuracy in this fold is 0.625

Validation of a model - Cross-validation - 2

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, shuffle=False)
```

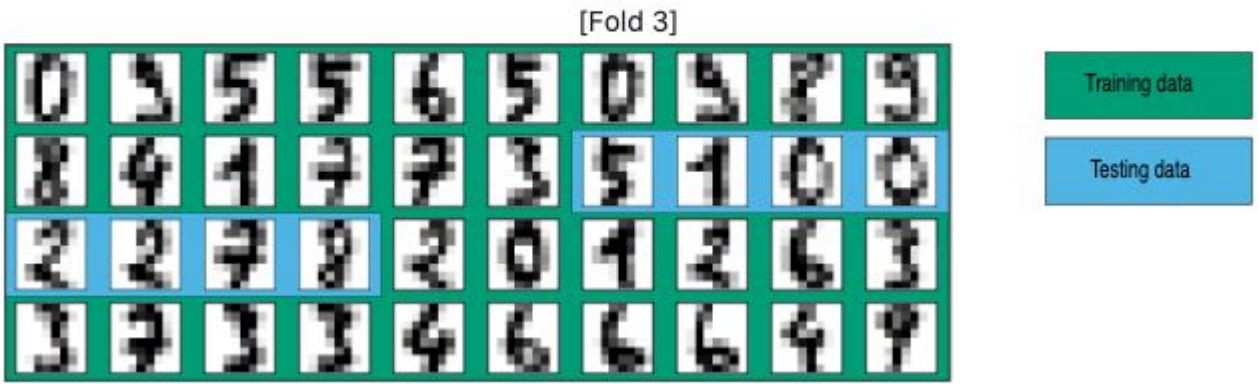


The test accuracy in this fold is 0.75

Validation of a model - Cross-validation - 3

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, shuffle=False)
```

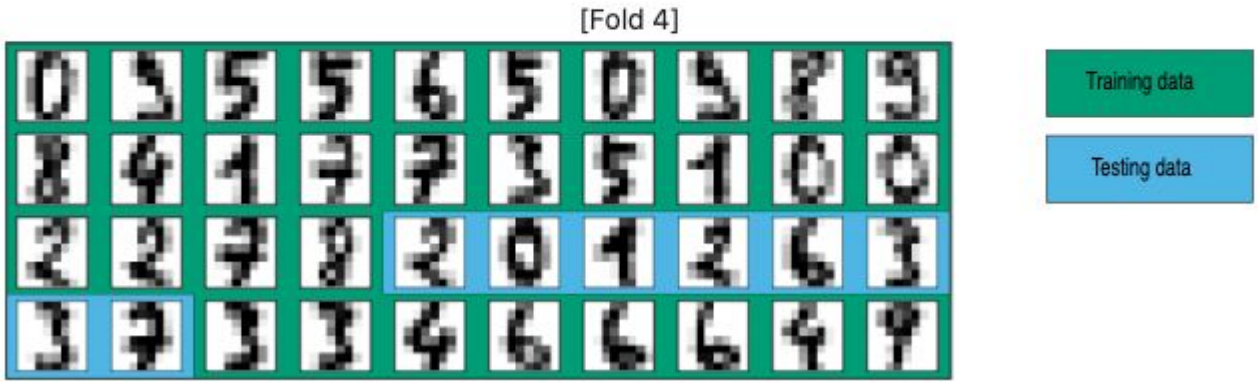


The test accuracy in this fold is 1.000

Validation of a model - Cross-validation - 4

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, shuffle=False)
```

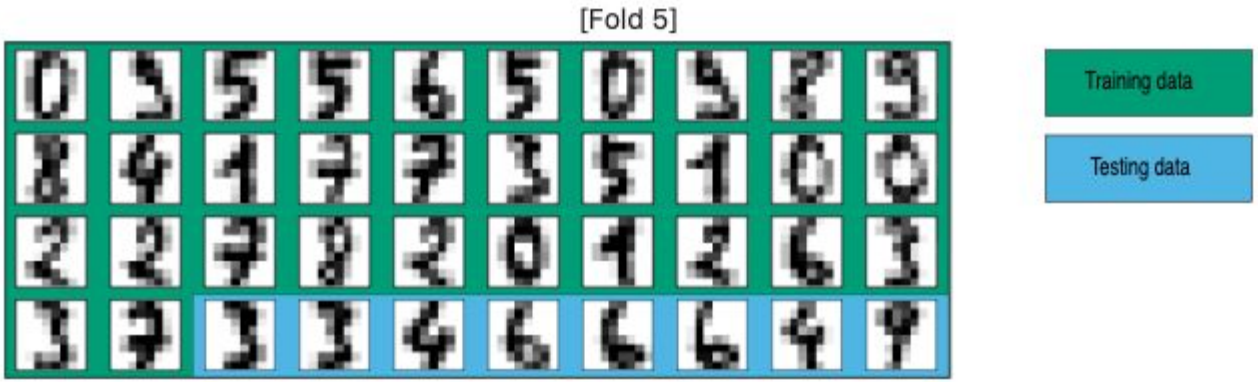


The test accuracy in this fold is 1.000

Validation of a model - Cross-validation - 5

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, shuffle=False)
```

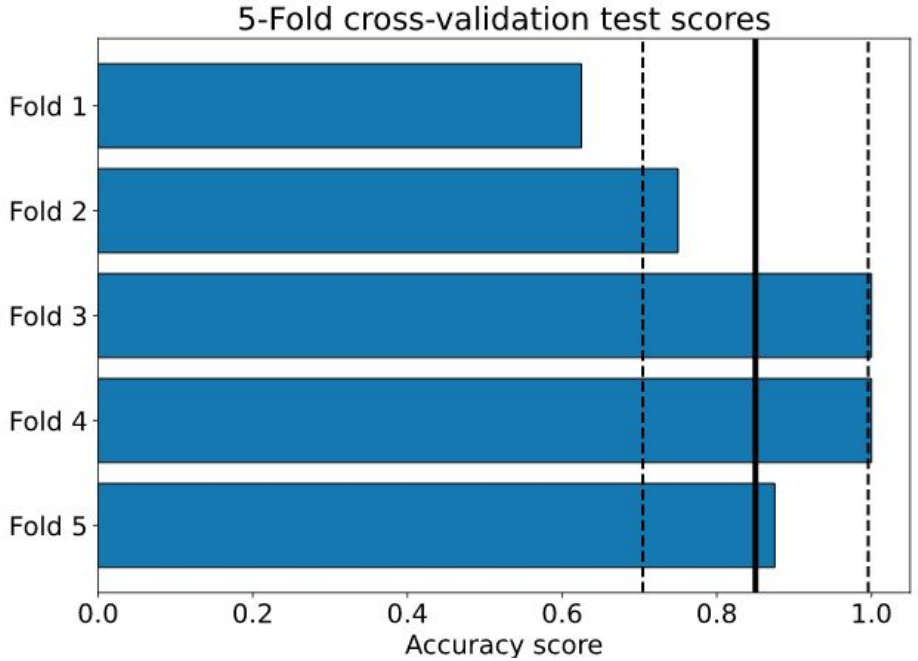


The test accuracy in this fold is 0.870

Validation of a model - Score variability

```
from sklearn.model_selection import cross_val_score

cv = KFold(n_splits=5, shuffle=False)
test_scores = cross_val_score(model, data, target, cv=cv)
```

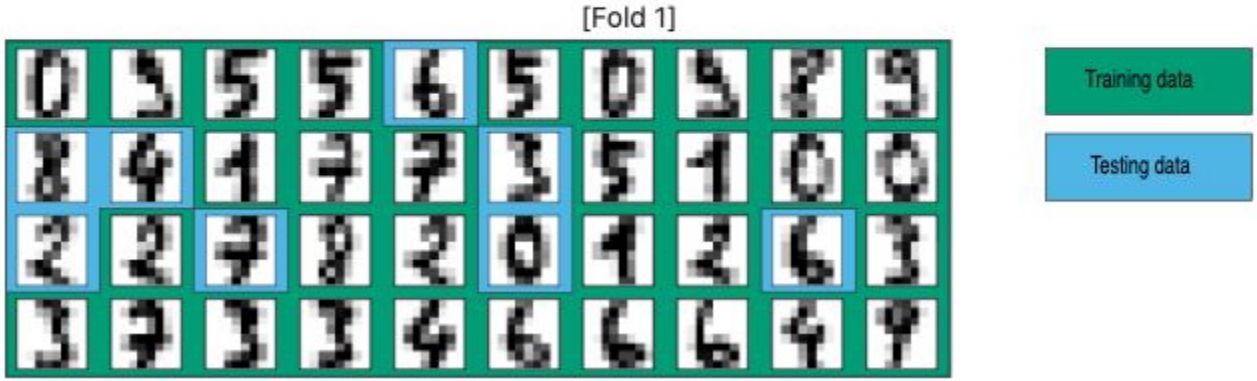


The average accuracy is 0.85 ± 0.15

Validation of a model - Cross-validation - shuffle=True - 1

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, random_state=0, shuffle=True)
```

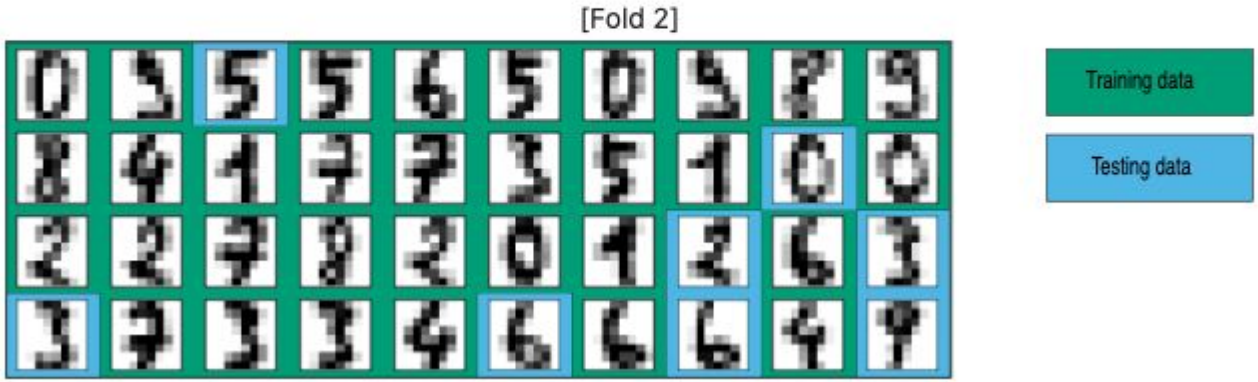


The test accuracy in this fold is 1.000

Validation of a model - Cross-validation - shuffle=True - 2

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, random_state=0, shuffle=True)
```

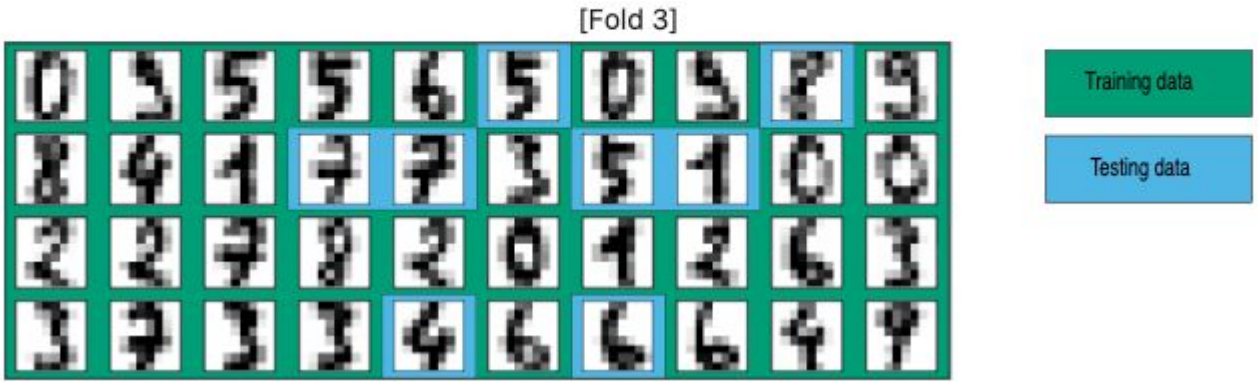


The test accuracy in this fold is 0.875

Validation of a model - Cross-validation - shuffle=True - 3

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, random_state=0, shuffle=True)
```

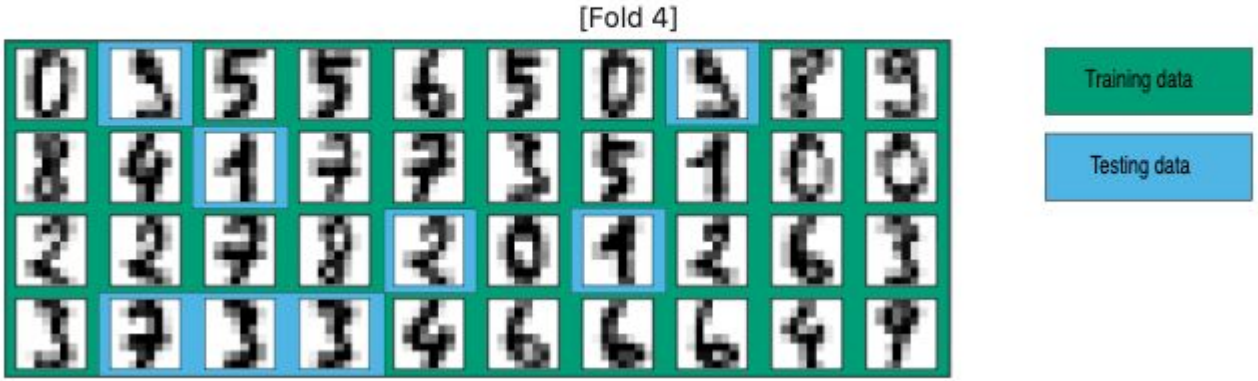


The test accuracy in this fold is 1.000

Validation of a model - Cross-validation - shuffle=True - 4

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, random_state=0, shuffle=True)
```

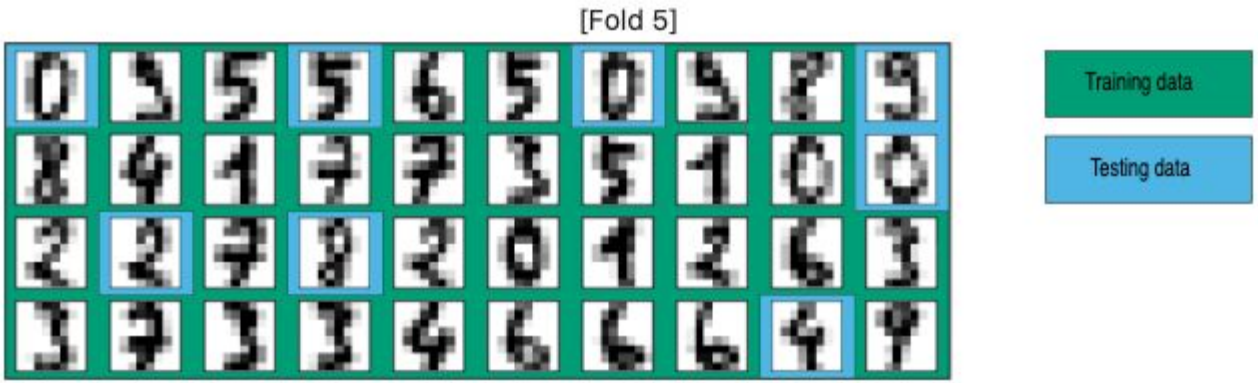


The test accuracy in this fold is 0.75

Validation of a model - Cross-validation - shuffle=True - 5

```
from sklearn.model_selection import KFold

cv = KFold(n_splits=5, random_state=0, shuffle=True)
```



The test accuracy in this fold is 1.000

Validation of a model - Cross-validation in scikit-learn

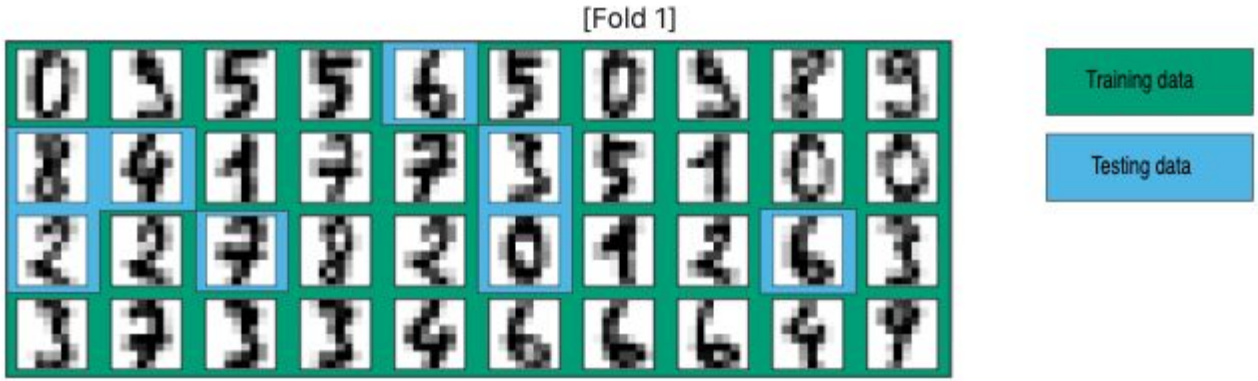
- Other than `**KFold**`, scikit-learn provides several techniques for cross-validation

- One example is `**ShuffleSplit**`, where the number of splits no longer determines the size of the train and test sets.

Validation of a model - Cross-validation in scikit-learn

```
from sklearn.model_selection import ShuffleSplit

cv = ShuffleSplit(n_splits=2, test_size=0.2, random_state=0)
```

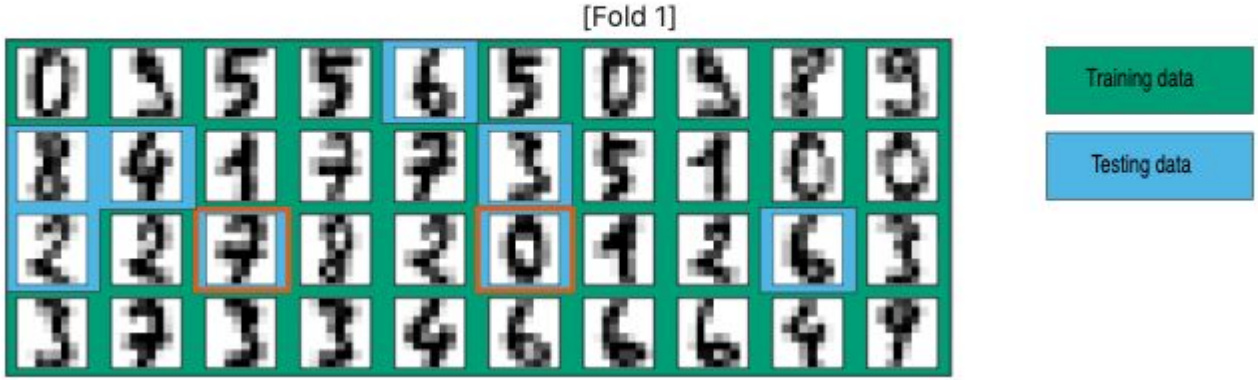


- The `ShuffleSplit` strategy is equivalent to manually calling `train_test_split` many times with different random states.

Validation of a model - Cross-validation in scikit-learn

```
from sklearn.model_selection import ShuffleSplit

cv = ShuffleSplit(n_splits=2, test_size=0.2, random_state=0)
```

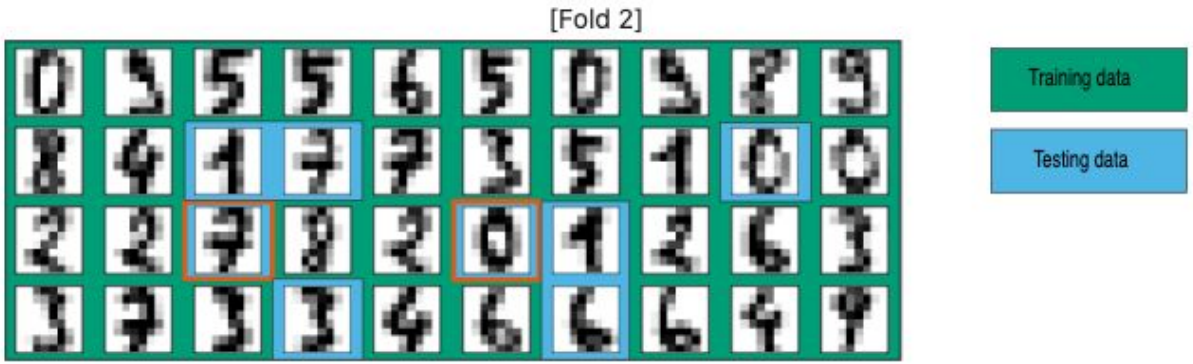


- The `ShuffleSplit` strategy is equivalent to manually calling `train_test_split` many times with different random states.
- In small dataset can happen that values are repeated and affect the representativity of the testing set

Validation of a model - Cross-validation in scikit-learn

```
from sklearn.model_selection import ShuffleSplit

cv = ShuffleSplit(n_splits=2, test_size=0.2, random_state=0)
```



- The `ShuffleSplit` strategy is equivalent to manually calling `train_test_split` many times with different random states.
- In small dataset can happen that values are repeated and affect the representativity of the testing set

Validation of a model - Take home messages

Full data

- should not be used for scoring a model

Train-test split

- evaluate the generalization performance on unseen data

Cross-validation

- evaluate the variability of our estimation of the generalization performance