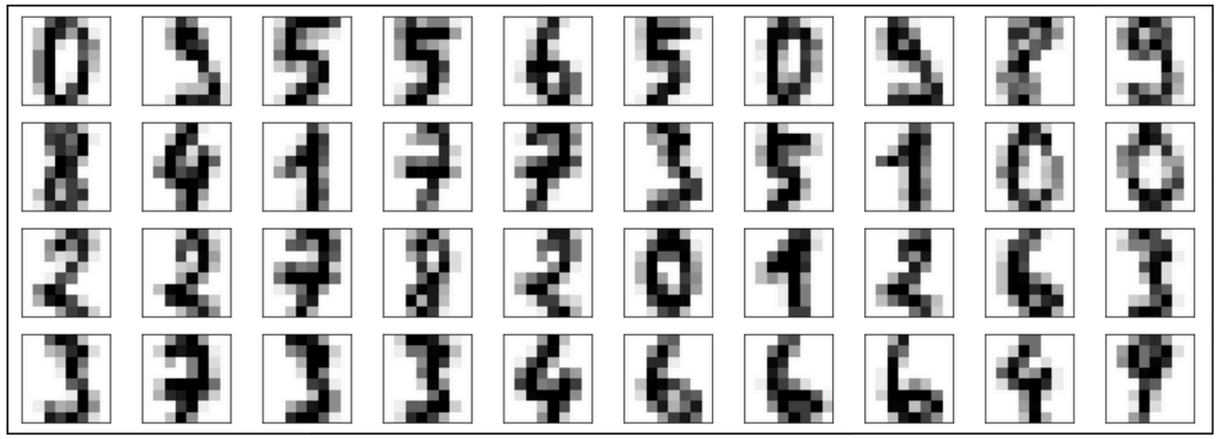
Validation of a model

train-test split and cross-validation

Full dataset

```
from sklearn.datasets import load_digits

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

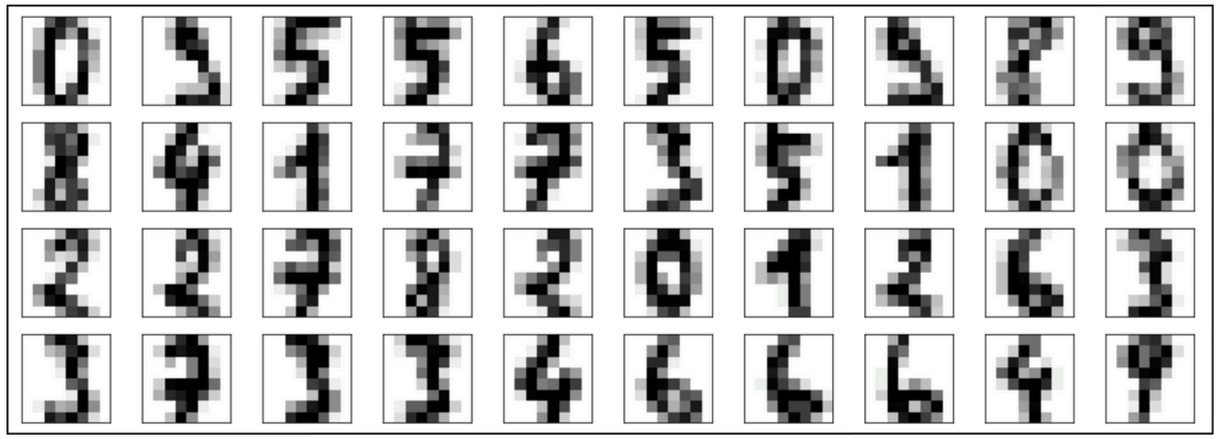


Goal: to evaluate the generalization performance of a model.

Full dataset

```
from sklearn.datasets import load_digits

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

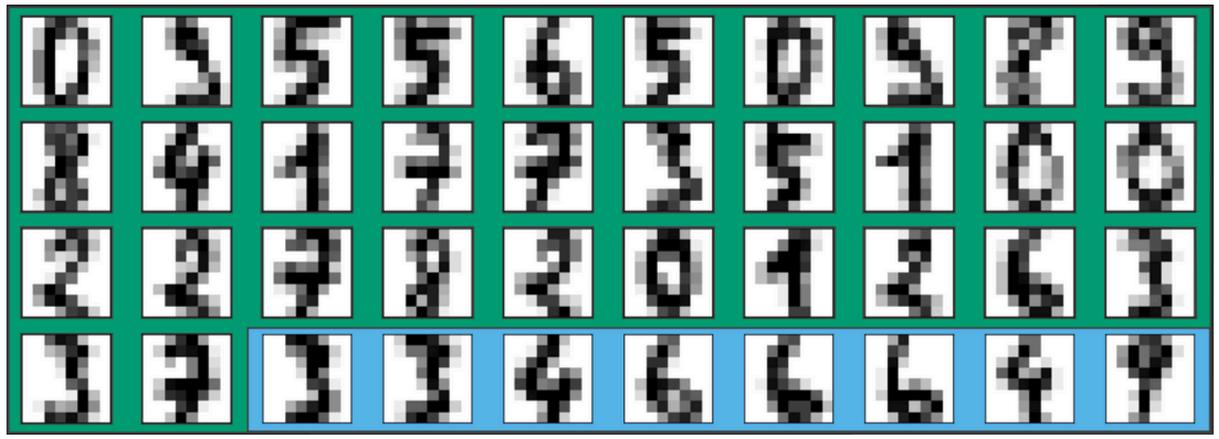


Goal: to evaluate the generalization performance of a model.

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

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 1.000

Train-test split

- In general, the score of a model depends on the split:

Train-test split

- In general, the score of a model depends on the split:
  - the train-test proportion

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

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

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

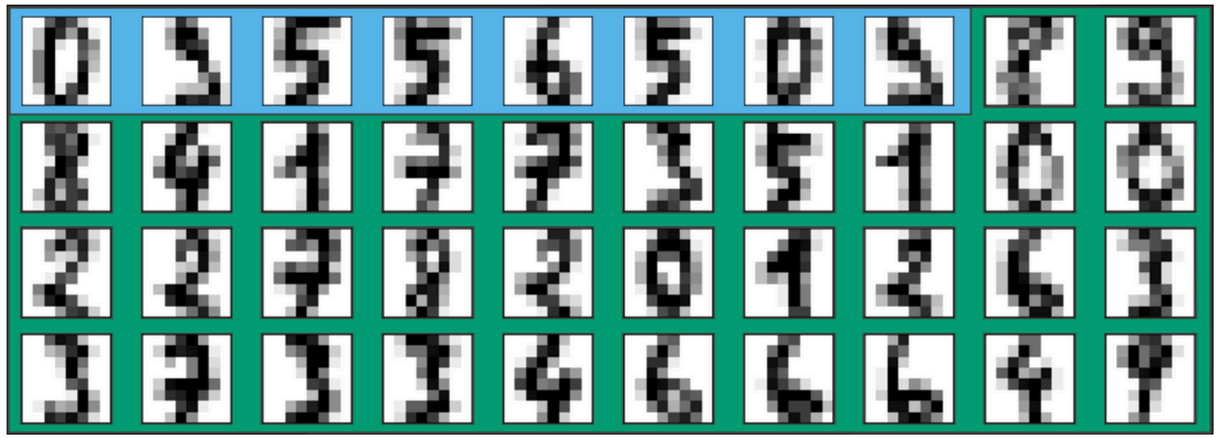
- Cross-validation consists of repeating the split such that the training and testing sets are different for each evaluation

Cross-validation

```
from sklearn.model_selection import KFold

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

Fold 1



Training data

Testing data

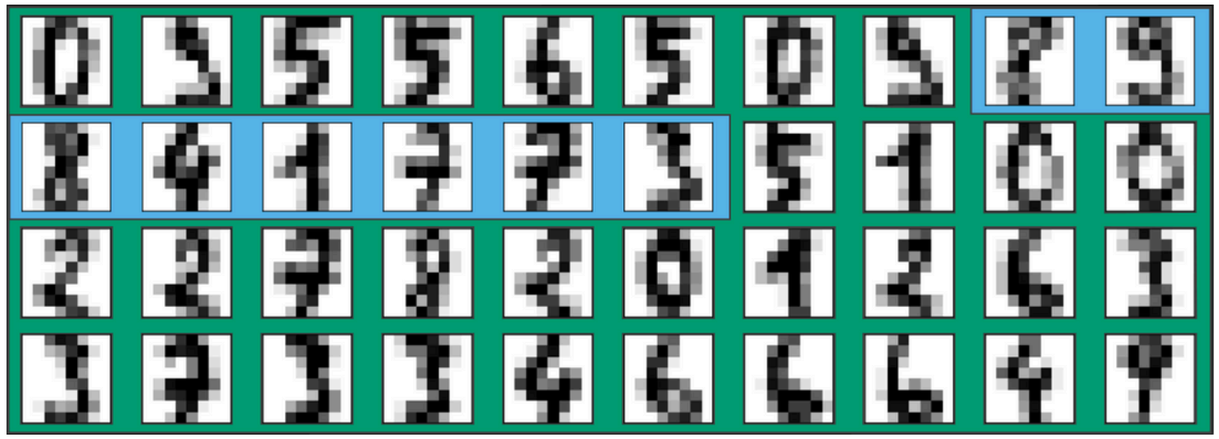
The test accuracy in this fold is 0.625

Cross-validation

```
from sklearn.model_selection import KFold

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

Fold 2



Training data

Testing data

The test accuracy in this fold is 0.750

Cross-validation

```
from sklearn.model_selection import KFold

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

Fold 3

Training data

Testing data

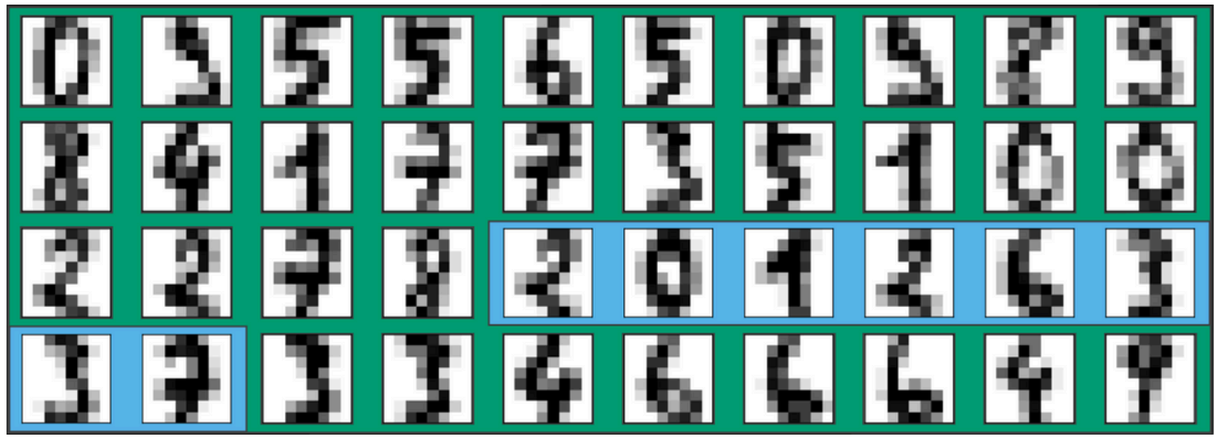
The test accuracy in this fold is 1.000

Cross-validation

```
from sklearn.model_selection import KFold

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

Fold 4



Training data

Testing data

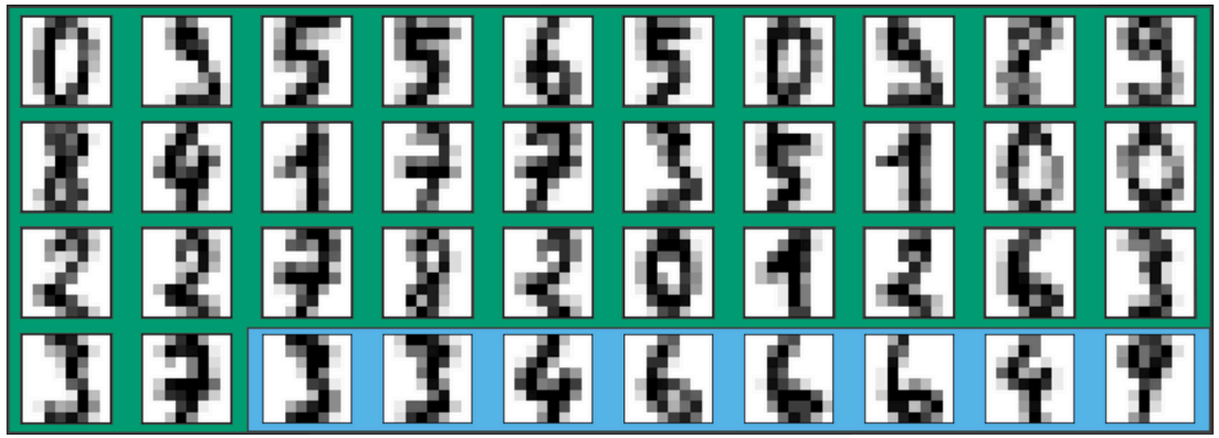
The test accuracy in this fold is 1.000

Cross-validation

```
from sklearn.model_selection import KFold

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

Fold 5



Training data

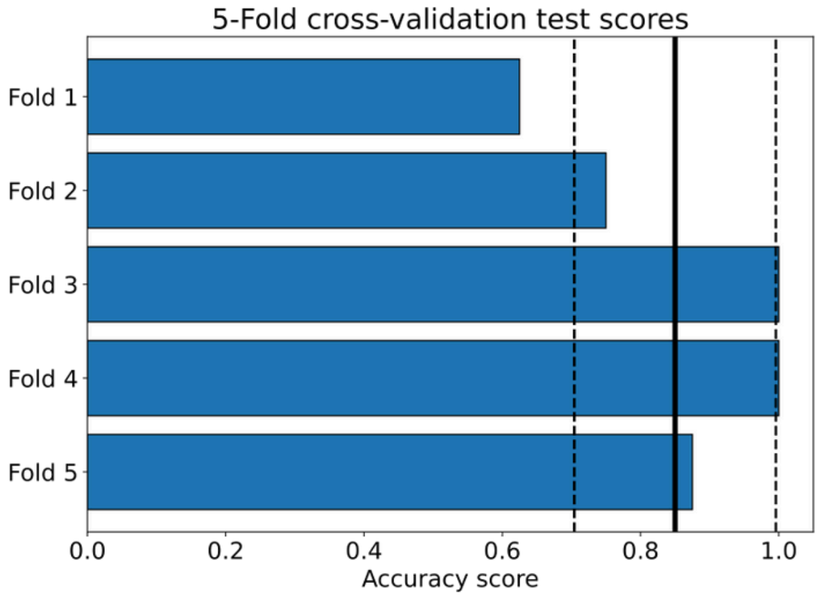
Testing data

The test accuracy in this fold is 0.875

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)
```



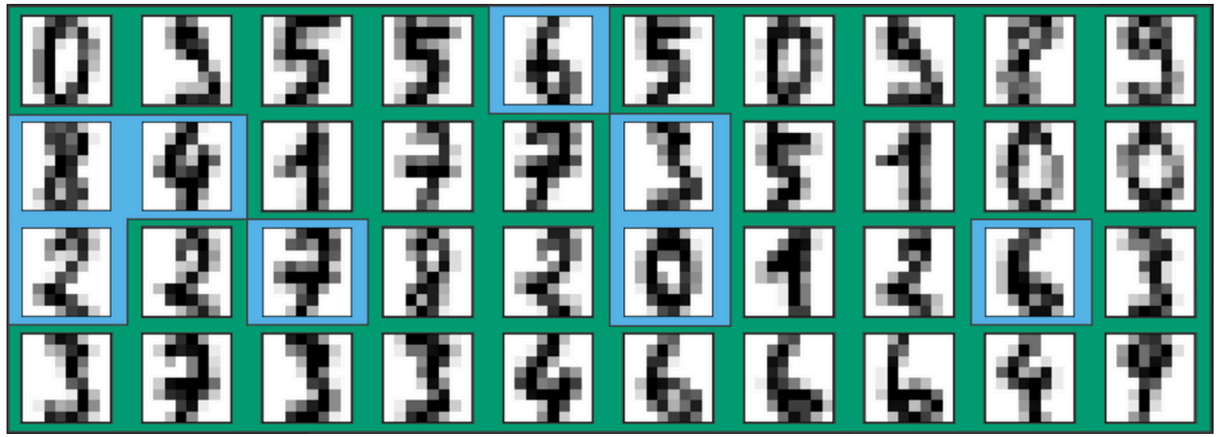
The average accuracy is 0.85 ± 0.15

Cross-validation with shuffle=True

```
from sklearn.model_selection import KFold

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

Fold 1



Training data

Testing data

The test accuracy in this fold is 1.000

.skolar

Cross-validation with shuffle=True

```
from sklearn.model_selection import KFold

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

Fold 2

Training data

Testing data

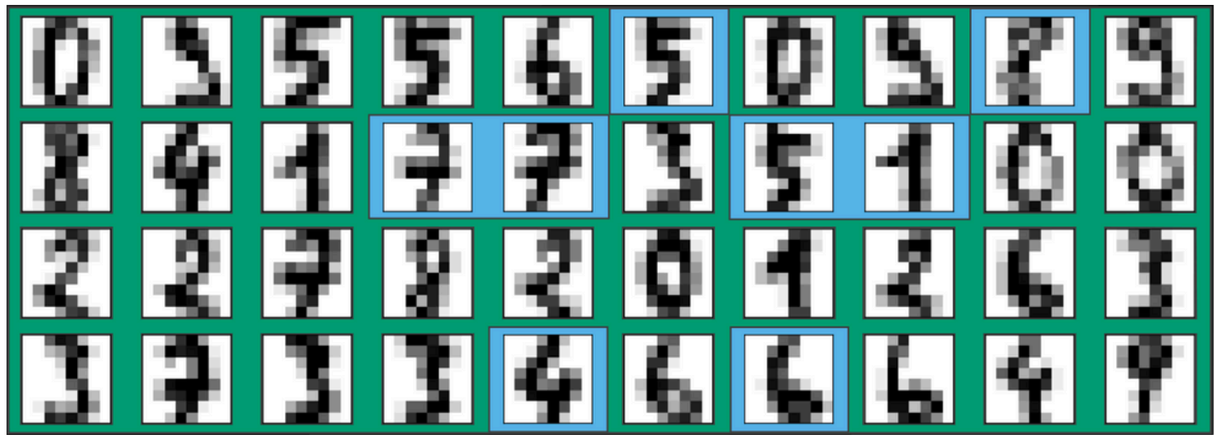
The test accuracy in this fold is 0.875

Cross-validation with shuffle=True

```
from sklearn.model_selection import KFold

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

Fold 3



Training data

Testing data

The test accuracy in this fold is 1.000

Cross-validation with shuffle=True

```
from sklearn.model_selection import KFold

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

Fold 4

Training data

Testing data

The test accuracy in this fold is 0.750

Cross-validation with shuffle=True

```
from sklearn.model_selection import KFold

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

Fold 5

Training data

Testing data

The test accuracy in this fold is 1.000

Cross-validation in scikit-learn

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

Cross-validation in scikit-learn

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

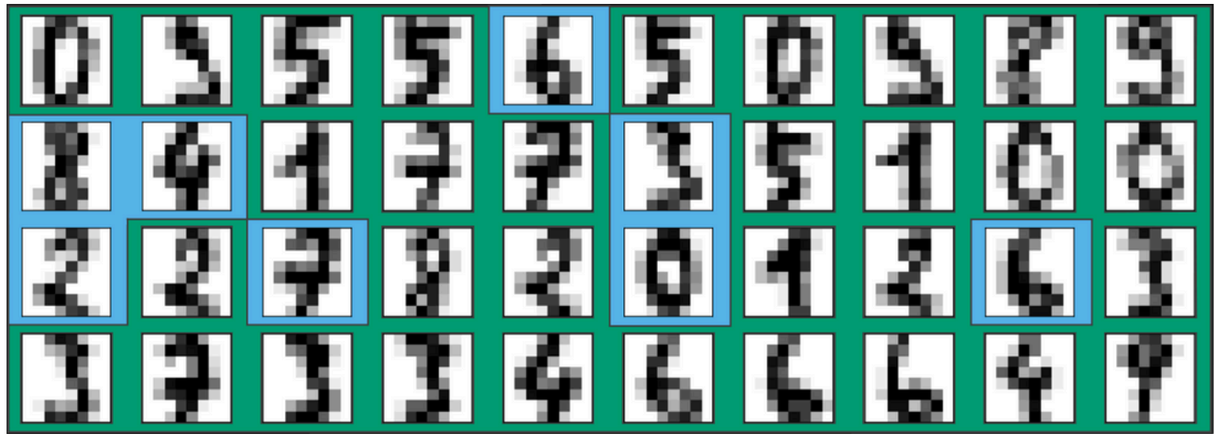
- One example is ShuffleSplit, where the number of splits no longer determines the size of the train and test sets.

Cross-validation with ShuffleSplit

```
from sklearn.model_selection import ShuffleSplit

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

Split 1



Training data

Testing data

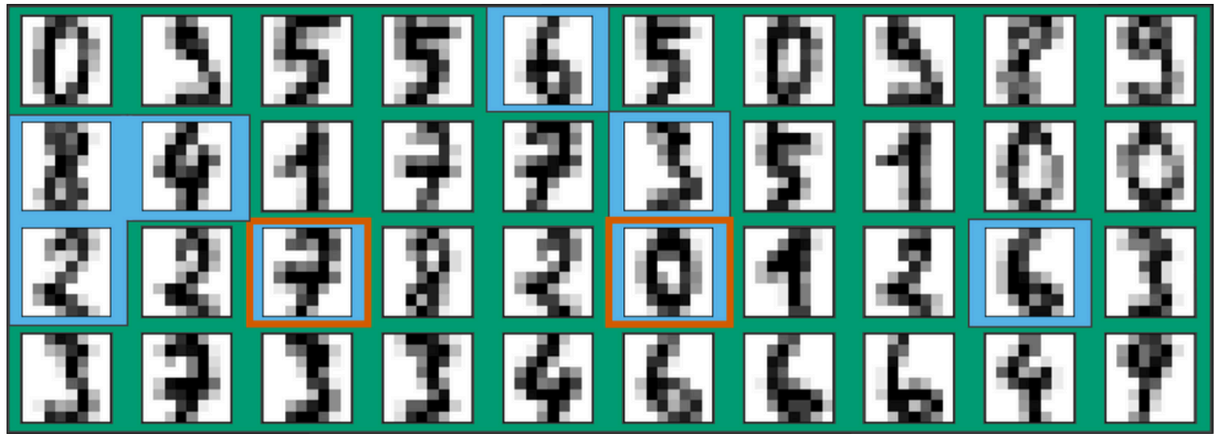
The test accuracy in this split is 1.000

Cross-validation with ShuffleSplit

```
from sklearn.model_selection import ShuffleSplit

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

Split 1



Training data

Testing data

The test accuracy in this split is 1.000

Cross-validation with ShuffleSplit

```
from sklearn.model_selection import ShuffleSplit

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

Split 2

Training data

Testing data

The test accuracy in this split is 1.000

Main takeaways

Full data

- should not be used for scoring a model

Train-test split

- evaluates the generalization performance on unseen data

Cross-validation

- evaluates the variability of our estimation of the generalization performance