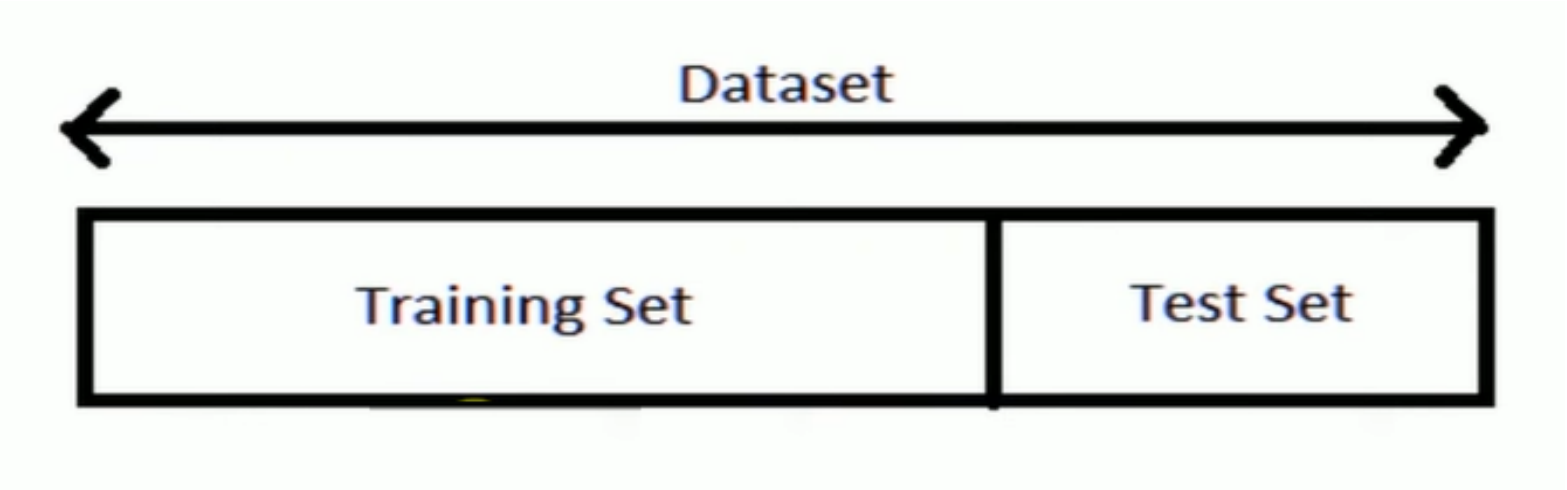
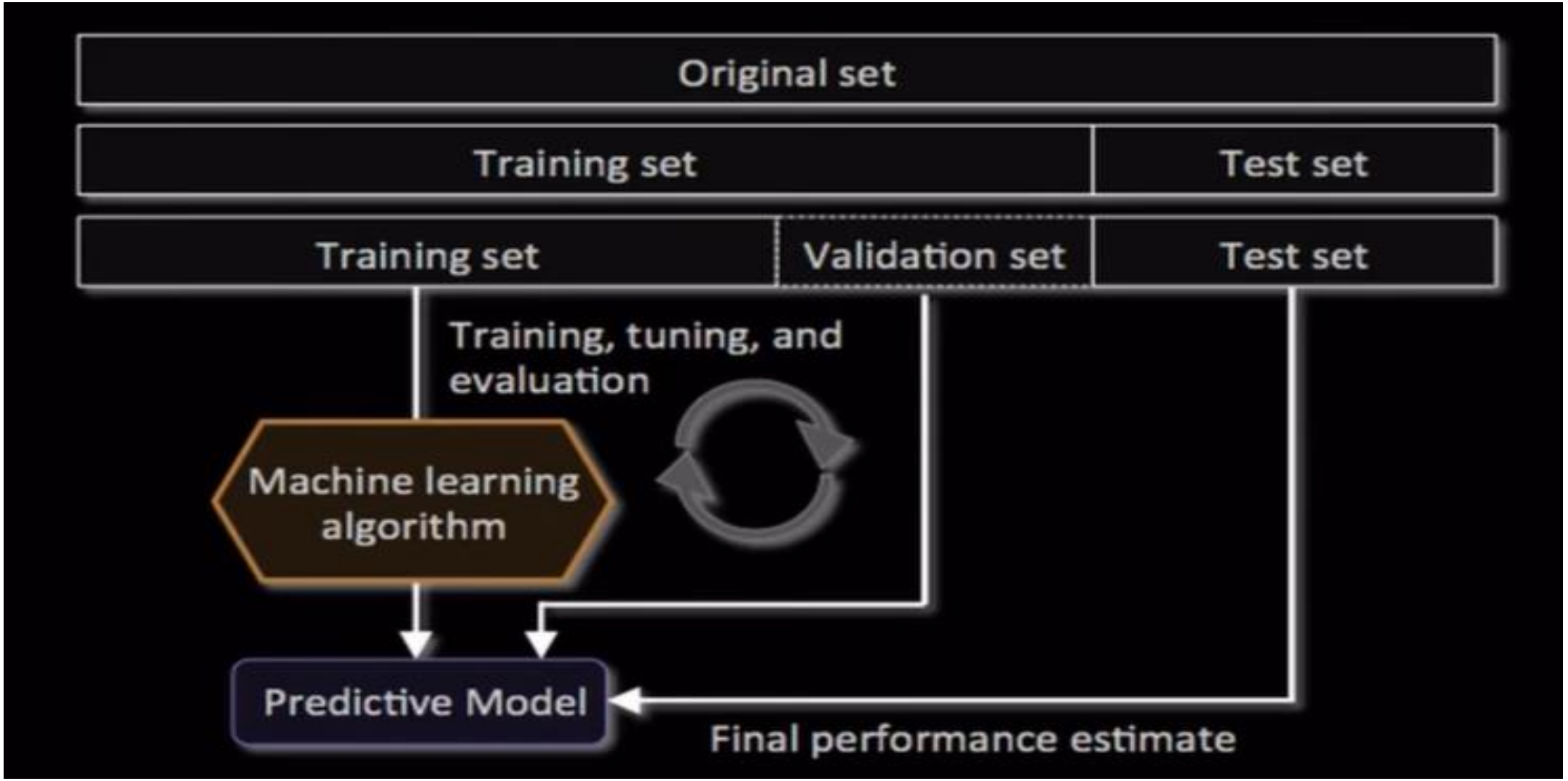
Dataset



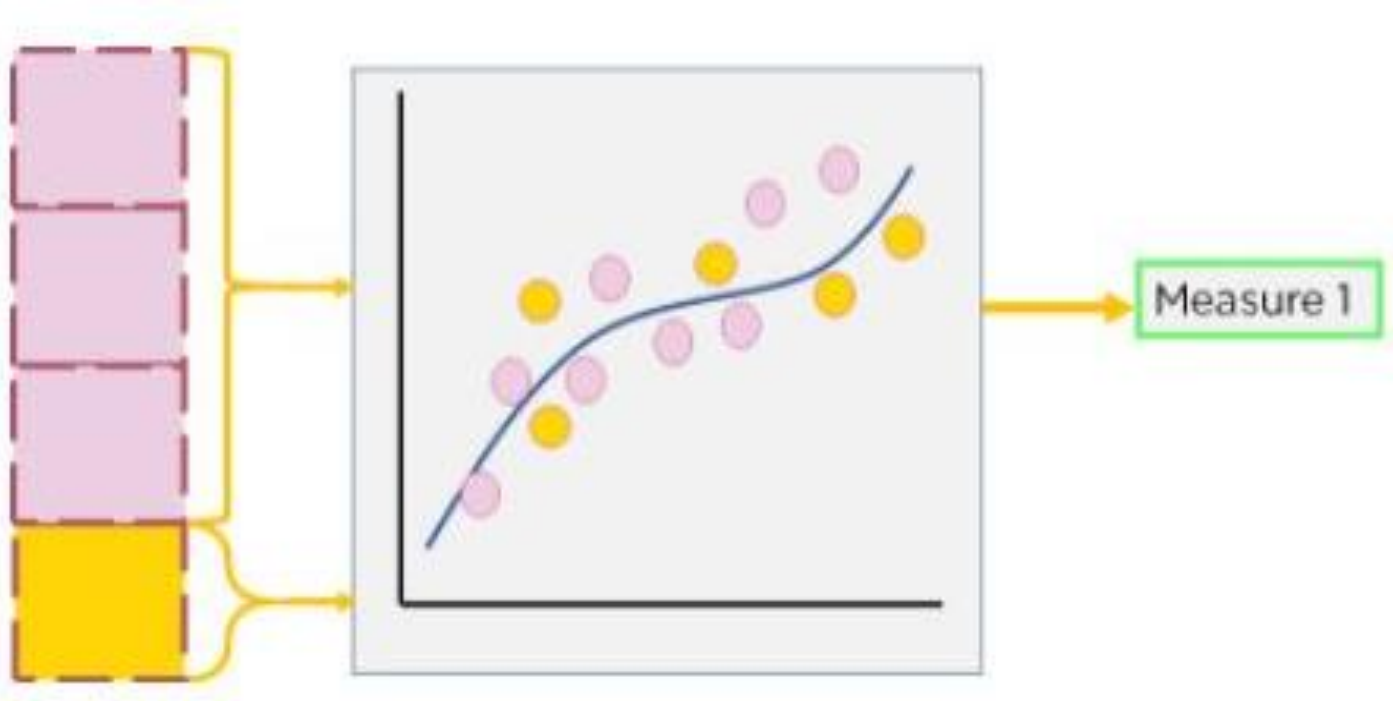

Original set
Training set
Test set
Training set
Validation set
Test set
Training, tuning, and evaluation
Machine learning algorithm
Predictive Model
Final performance estimate

What is Cross-Validation?

- Instead of splitting our dataset into two parts, one to train on and another to test on, in cross-validation, we split our dataset into multiple portions, train on some of these and use the rest to test on.
- We then use a different portion to train and test our model on.
- This ensures that our model is training and testing on new data at every new step.

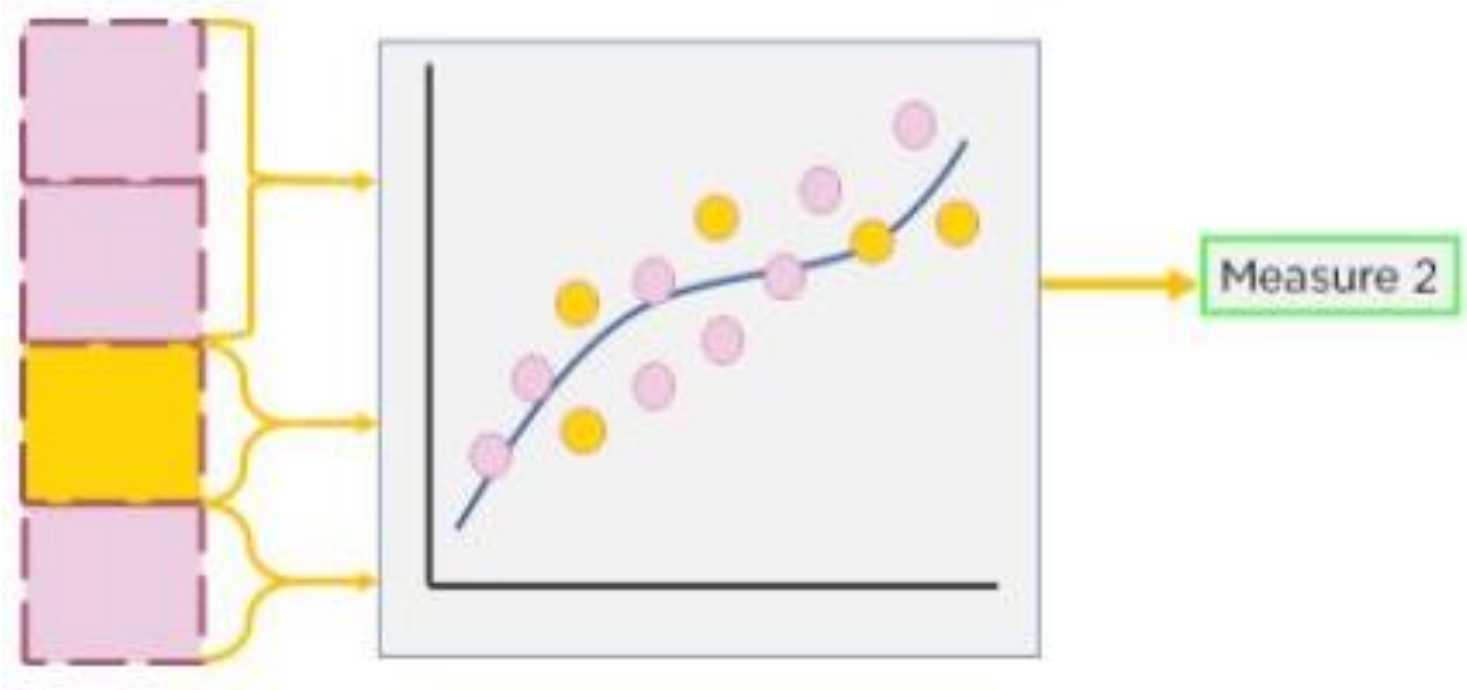
Steps in Cross-validation (1/5)

Step 1: Split the data into train and test sets and evaluate the model's performance



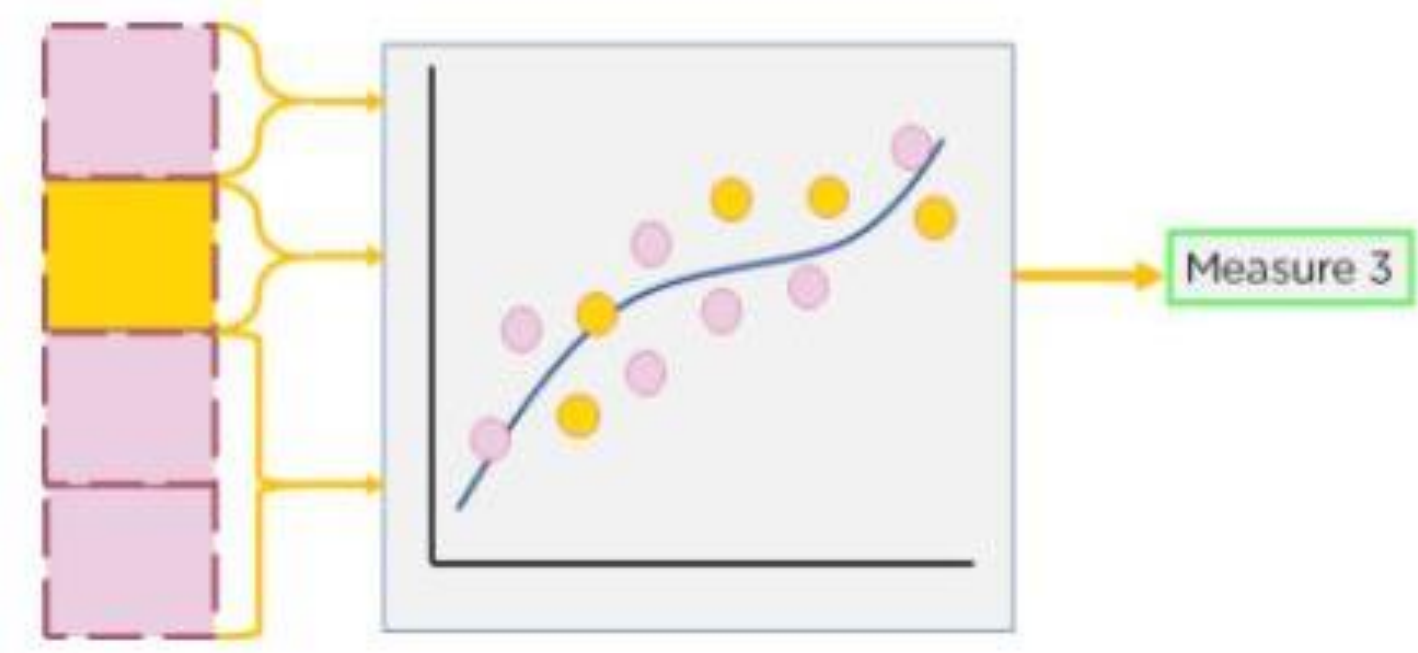
Steps in Cross-validation (2/5)

- Step 2: Split the data into new train and test sets and re-evaluate the model's performance



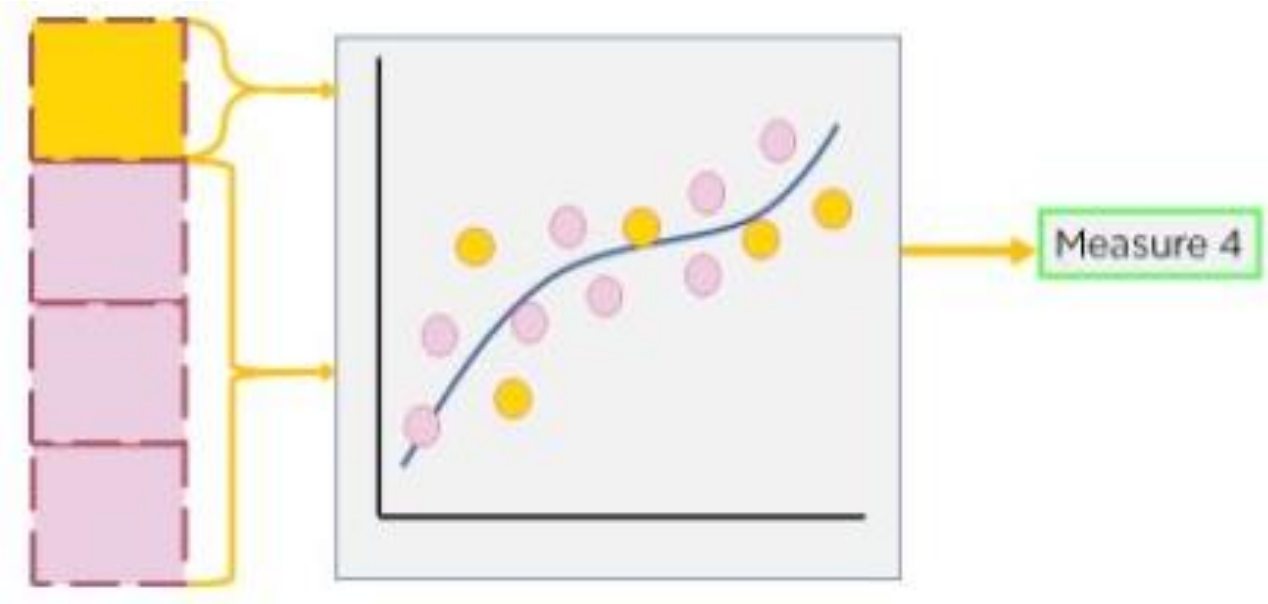
Steps in Cross-validation (3/5)

- Repeat Step 2: Split the data into new train and test sets and re-evaluate the model's performance



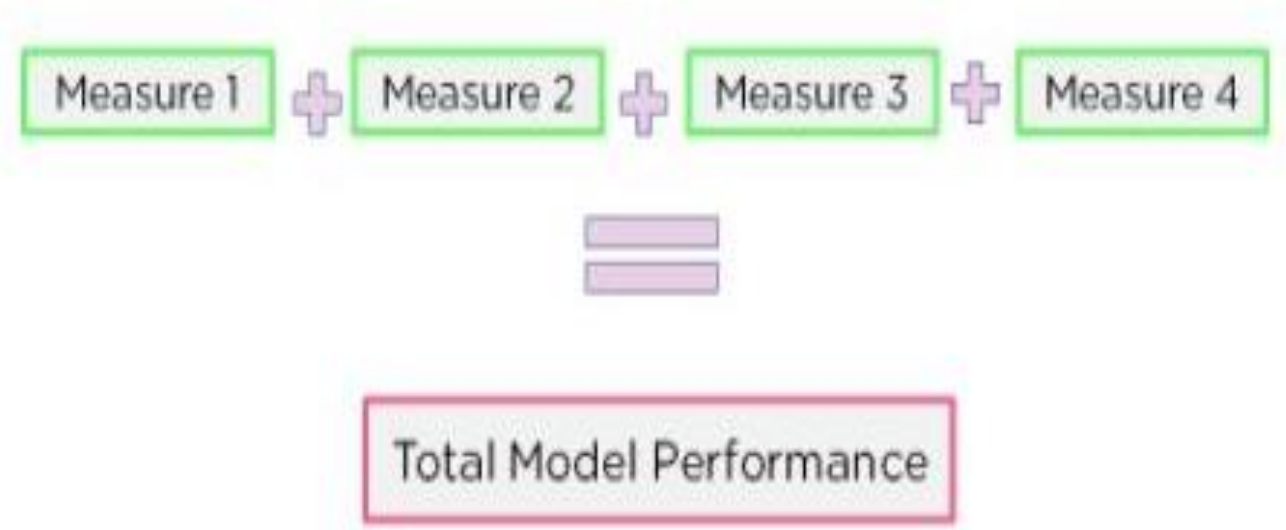
Steps in Cross-validation (4/5)

- Repeat Step 2: Split the data into new train and test sets and re-evaluate the model's performance



Steps in Cross-validation (5/5)

- Step 3: To get the actual performance metric the average of all measures is taken

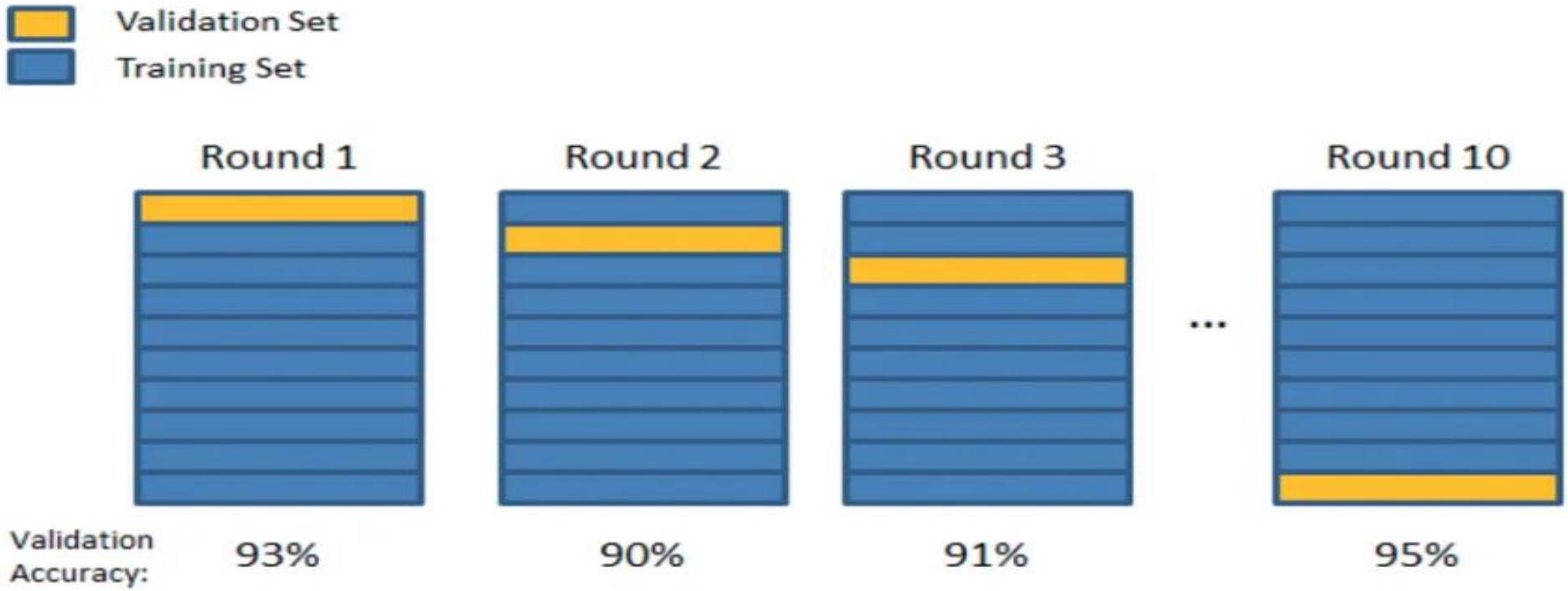


Major Type of Cross Validation

- K-Fold
- Stratified K-Fold

K-fold Cross-validation

- In K-Folds Cross Validation we split our data into k different subsets (or folds).
- We use k-1 subsets to train our data and leave the last subset (or the last fold) as test data.
- We then average the model against each of the folds and then finalize our model.
- After that we test it against the test set.



Visual representation of K-Folds. Again, H/t to Joseph Nelson!

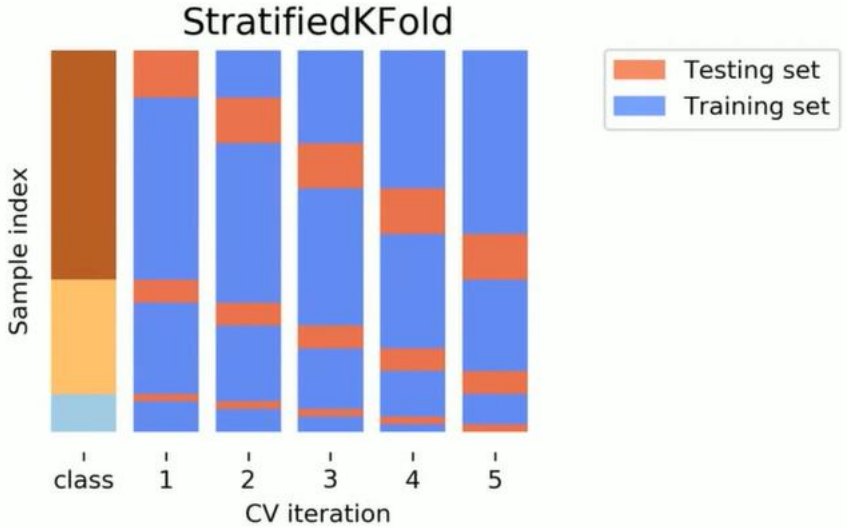
Simple implementation of K-Fold Cross-Validation in Python

```
from sklearn.model_selection import KFold
X = ["a",'b','c','d','e','f']

kf = KFold(n_splits=3, shuffle=False, random_state=None)

for train, test in kf.split(X):

    print("Train data",train,"Test data",test)
```

- Output
- Train: [2 3 4 5] Test: [0 1]
- Train: [0 1 4 5] Test: [2 3]
- Train: [0 1 2 3] Test: [4 5]

Stratified K-Fold Cross-Validation

- This is a slight variation from K-Fold Cross Validation, which uses **‘stratified sampling’** instead of ‘random sampling.’
- Here the data is split in such a way that it represents all the classes from the population.



Quick implementation of Stratified K-Fold Cross-Validation in Python

```
import numpy as np
from sklearn.model_selection import StratifiedKFold
X = np.array([[1,2],[3,4],[5,6],[7,8],[9,10],[11,12]])
y= np.array([0,0,1,0,1,1])

skf = StratifiedKFold(n_splits=3,random_state=None,shuffle=False)

for train_index,test_index in skf.split(X,y):
    print("Train:",train_index,'Test:',test_index)
    X_train,X_test = X[train_index], X[test_index]
    y_train,y_test = y[train_index], y[test_index]
```

Output

- Train: [1 3 4 5] Test: [0 2]
- Train: [0 2 3 5] Test: [1 4]
- Train: [0 1 2 4] Test: [3 5]
- The output clearly shows the stratified split done based on the classes ‘0’ and ‘1’ in ‘y’.