Exposé

Thème:

Cross-Validation Techniques

Réalisé par
Fofana Mamadou Moussa

Encadrant
Prof Harchli Fidae

Plan de présentation:

I-Introduction à la validation croisée

II-Principales techniques de validation croisée

III-Avantages et inconvénients

IV-Implémentation pratique

V-Études de cas

VI-Conclusion

I- Introduction à la validation croisée

- La validation croisée est une technique fondamentale en machine learning qui permet d’évaluer la capacité d’un modèle à généraliser sur de nouvelles données.
- Elle repose sur une idée simple mais puissante : diviser les données disponibles en plusieurs sous-ensembles, souvent appelés "folds", puis utiliser ces sous-ensembles pour entraîner et tester le modèle de différentes manières.
- Ce processus garantit une évaluation plus rigoureuse et fiable des performances du modèle sur des données qu’il n’a pas vues auparavant.

OBJECTIF

- L’objectif principal de la validation croisée est d’obtenir une estimation précise de la performance du modèle tout en évitant le surajustement (overfitting).

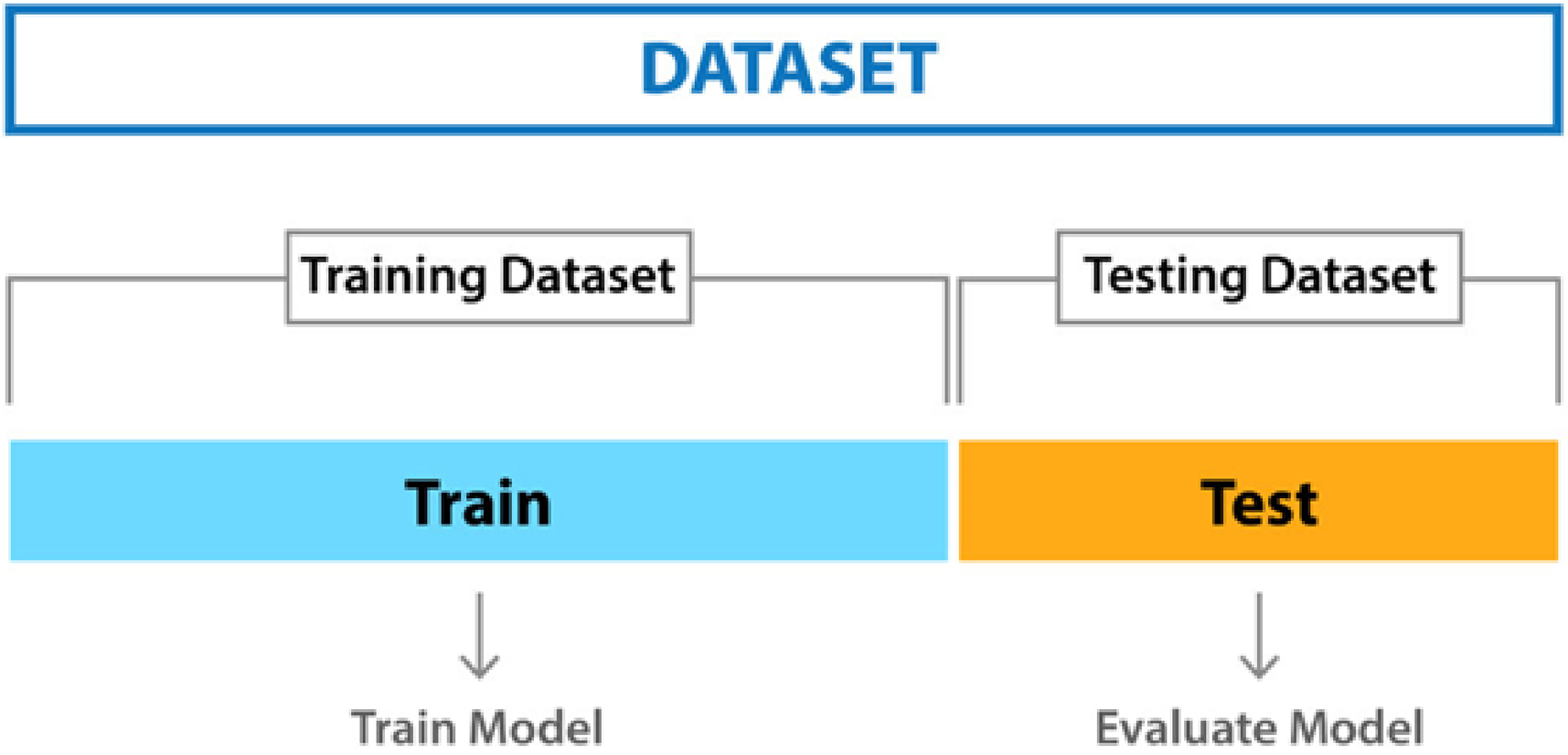
- Le surajustement survient lorsqu’un modèle apprend trop bien les particularités des données d’entraînement, au point de ne plus être capable de généraliser correctement sur de nouvelles données.

- En alternant les rôles des sous-ensembles entre entraînement et test, la validation croisée permet de détecter ce problème et d’assurer que le modèle reste robuste

II- Principales techniques de validation croisée

1.Holdout Method :

- **Description:** Les données sont divisées en deux ensembles : un pour l’entraînement (par exemple, 70 %) et un pour le test (par exemple, 30 %). Le modèle est entraîné sur l’ensemble d’entraînement et évalué sur l’ensemble de test.
- **Quand utiliser:** Utile pour une évaluation rapide lorsque qu’on disposes d’un grand dataset.


DATASET
Training Dataset
Testing Dataset
Train
Test
Train Model
Evaluate Model

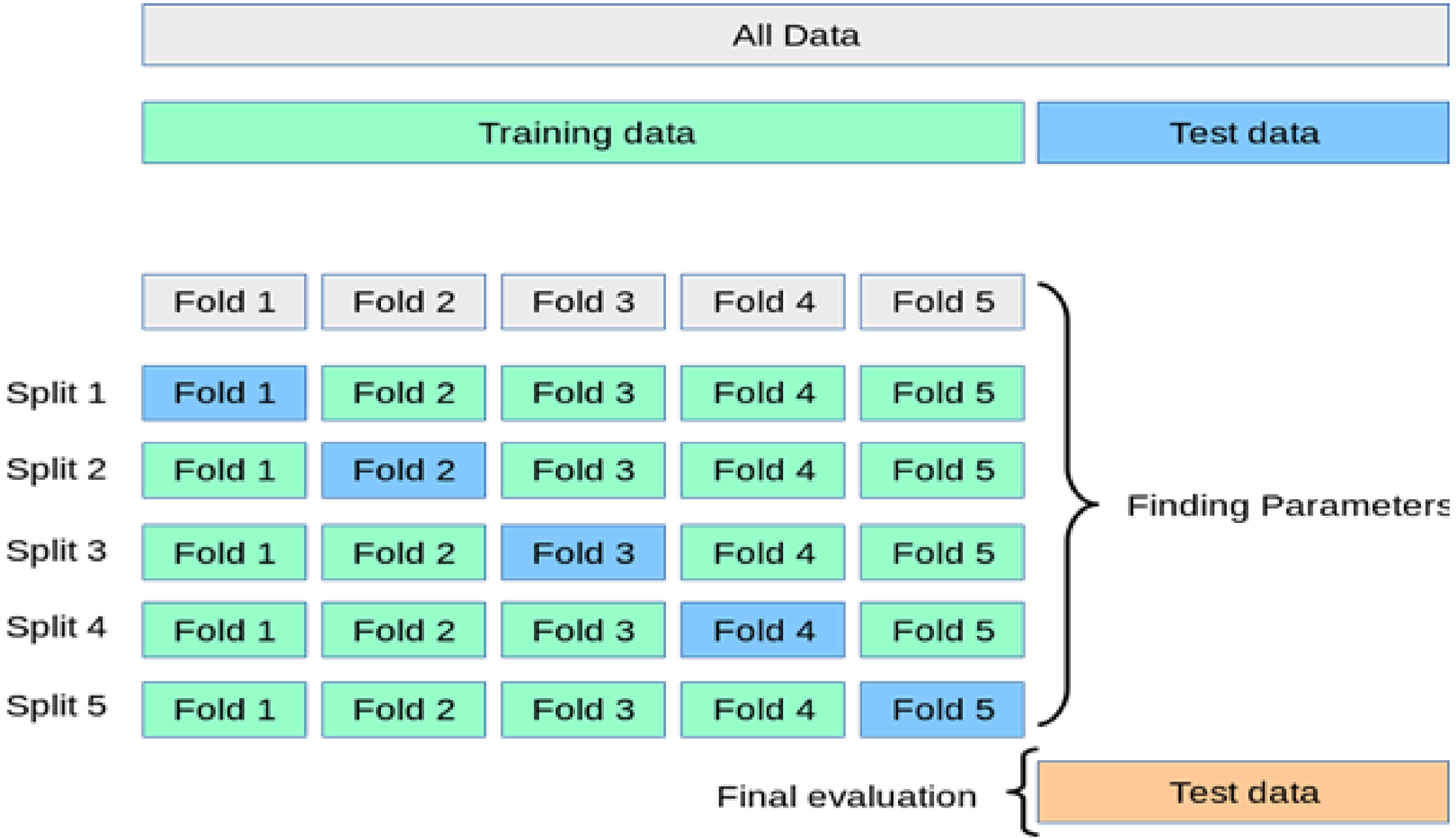
2.K-Fold Cross Validation :

Description :

Les données sont divisées en K sous-ensembles (ou "folds") de taille égale. Le modèle est entraîné K fois : à chaque fois, un fold est utilisé comme ensemble de test, et les K-1 autres comme ensemble d’entraînement. La performance finale est la moyenne des K évaluations.

Quand l’utiliser :

Bon compromis entre précision et temps de calcul.


All Data
Training data
Test data
Fold 1
Fold 2
Fold 3
Fold 4
Fold 5
Split 1
Split 2
Split 3
Split 4
Split 5
Finding Parameters
Final evaluation
Test data

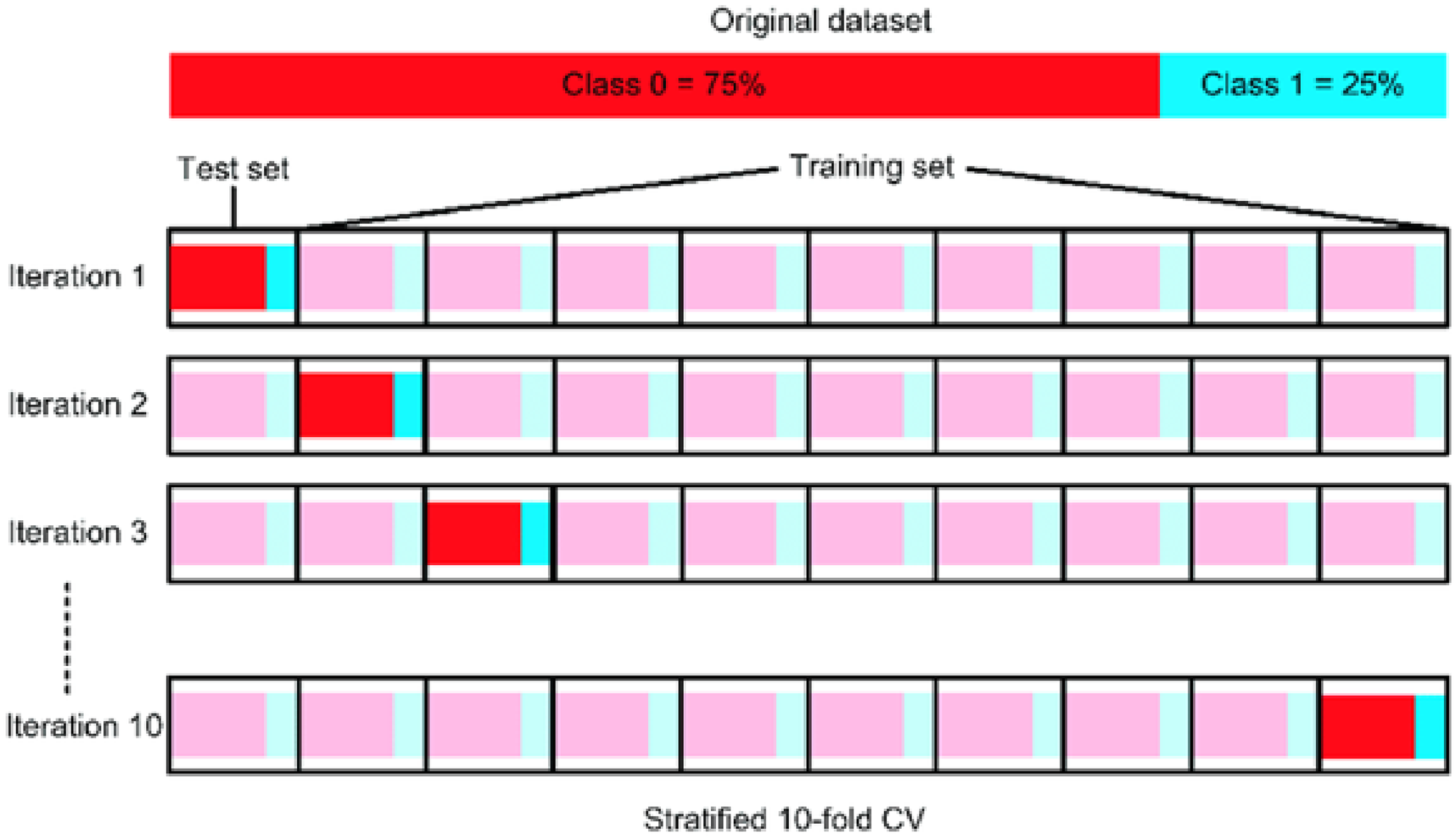
3.Stratified K-Fold Cross Validation :

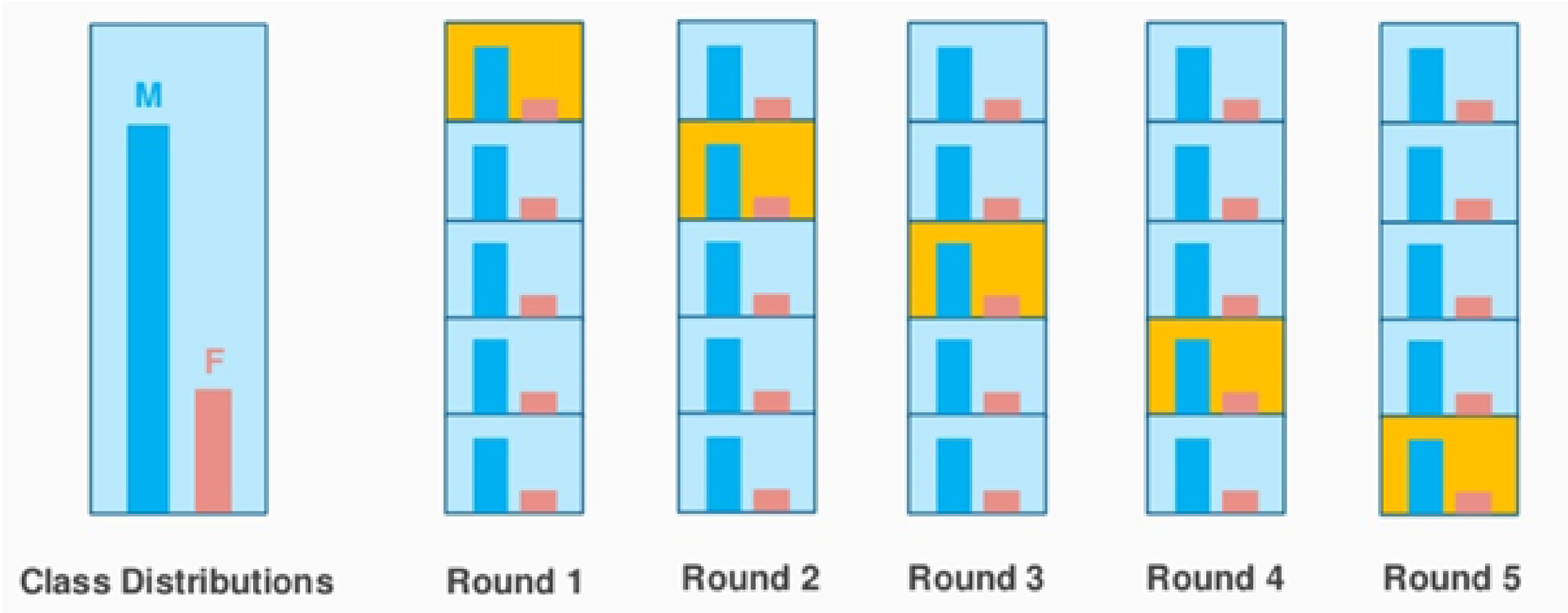
Description :

Similaire à la K-Fold, mais elle garantit que chaque fold conserve la même répartition des classes que l'ensemble global des données (par exemple, si une classe représente 20 % des données, elle sera à environ 20 % dans chaque fold).

Quand l'utiliser :

Parfait pour les données avec des classes déséquilibrées.


Original dataset
Class 0 = 75%
Class 1 = 25%
Test set
Training set
Iteration 1
Iteration 2
Iteration 3
Iteration 10
Stratified 10-fold CV


M
F
Class Distributions
Round 1
Round 2
Round 3
Round 4
Round 5

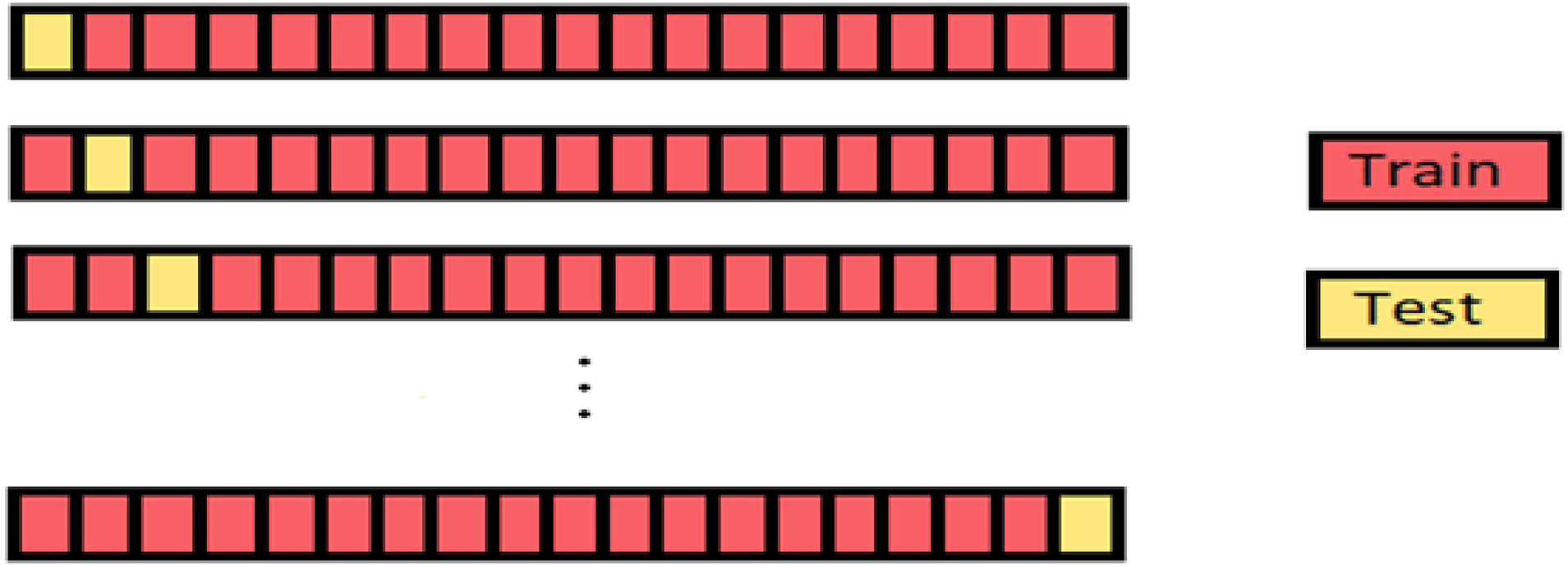
4.Leave-One-Out Cross Validation (LOOCV) :

Description :

Une variante de la K-Fold où K égale le nombre total d’exemples. À chaque itération, un seul exemple est utilisé pour le test, et tous les autres pour l’entraînement. Ce processus est répété pour chaque exemple.

Quand l’utiliser :

Recommandée pour les petits ensembles de données (moins de 100 exemples).


Train
Test

5.Time Series Cross Validation

Description :

Conçue pour les données séquentielles comme les séries temporelles, cette méthode respecte l’ordre chronologique. Les données sont divisées en fenêtres glissantes : par exemple, entraîner sur les données passées et tester sur les données futures.

Quand l’utiliser :

Indispensable pour les données où l’ordre est crucial, comme les prévisions météo ou les cours boursiers.


T1 T2 T3 T4
T1 T2 T3 T4 T5
T1 T2 T3 T4 T5 T6
T1 T2 T3 T4 T5 T6 T7
T1 T2 T3 T4 T5 T6 T7 T8

Remarque: On rencontre à part ceux-ci plein d’autres techniques de Cross Validation tels que: Leave P-Out Cross validation, Repeated K-folds, Nested K-folds, Repeated Random Test-Train Splits, etc…

III- Avantages et inconvénients

- **Holdout : Simple et rapide, mais sensible à la répartition des données.**
- **K-Fold : Plus fiable, mais demande plus de calculs.**
- **Stratified K-Fold : Idéal pour les déséquilibres, un peu plus complexe à mettre en place.**
- **LOOCV : Très précis, mais trop lent pour de grands ensembles.**
- **Time Series CV : Essentiel pour les séries temporelles, mais limité par la structure des données.**

IV- Implémentation pratique

Holdout Method

In [ ]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# Test and Train will be split as 30% & 70% Respectively
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=4)

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
result = model.score(X_test, y_test)
print(result)
```

```
0.9005847953216374
```

K-Fold Method

In [ ]:

```
model=DecisionTreeClassifier()
kfold_validation=KFold(10)

from sklearn.model_selection import cross_val_score
results=cross_val_score(model,X,y,cv=kfold_validation)
print(results)
print("Best Mean Accuracy:- ",np.mean(results))
```

```
[0.96491228 0.9122807  0.89473684 0.94736842 0.92982456 0.96491228
 0.9122807  0.96491228 0.96491228 0.89285714]
Best Mean Accuracy:-  0.9348997493734336
```

Stratified KFold

In [ ]:

```
from sklearn.model_selection import StratifiedKFold
skfold=StratifiedKFold(n_splits=10)
model=DecisionTreeClassifier()
scores=cross_val_score(model,X,y,cv=skfold)
print(np.mean(scores))
```

```
0.9156954887218044
```

Leave One Out method

In [ ]:

```
from sklearn.model_selection import LeaveOneOut
from numpy import std
from numpy import absolute
from numpy import mean
model=DecisionTreeClassifier()
leave_validation=LeaveOneOut()
results=cross_val_score(model,X,y,cv=leave_validation)
#results.mean()
scores = absolute(results)

# report performance
print('Accuracy: %.3f (%.3f)' % (mean(scores), std(scores)))
```

```
Accuracy: 0.926 (0.261)
```

Time Series Split

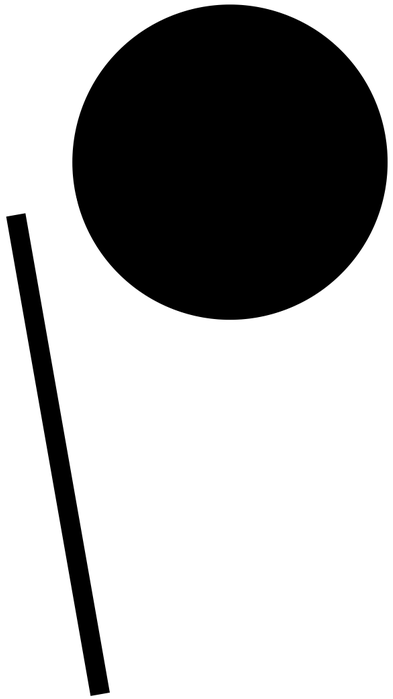
In [ ]:

```
import numpy as np
from sklearn.model_selection import TimeSeriesSplit
X = np.array([[1, 2], [3, 4], [1, 2], [3, 4], [1, 2], [3, 4]])
y = np.array([1, 2, 3, 4, 5, 6])
tscv = TimeSeriesSplit()
print(tscv)
for train_index, test_index in tscv.split(X):
  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]
```

```
TimeSeriesSplit(max_train_size=None, n_splits=5)
TRAIN: [0] TEST: [1]
TRAIN: [0 1] TEST: [2]
TRAIN: [0 1 2] TEST: [3]
TRAIN: [0 1 2 3] TEST: [4]
TRAIN: [0 1 2 3 4] TEST: [5]
```

VI- Conclusion:

En résumé, chaque technique de validation croisée possède des forces spécifiques qui la rendent adaptée à certains contextes. Que ce soit la simplicité de la Holdout Method, l’équilibre de la K-Fold, la précision de la Stratified K-Fold pour les données déséquilibrées, l’exactitude du LOOCV pour les petits ensembles, ou la pertinence de la Time Series CV pour les données séquentielles, le choix de la technique dépend du type de données et du problème à résoudre.



- Mais au-delà du choix, il faut insister sur un point essentiel : la validation croisée est indispensable pour garantir des modèles de machine learning fiables.
- Elle permet d’éviter le surajustement, d’évaluer rigoureusement les performances sur des données inédites et d’assurer que notre modèle sera robuste face aux défis du monde réel.
- Sans cette étape clé, même les modèles les plus sophistiqués risquent de décevoir une fois en production.
- Les Techniques de Validation Croisées sont utilisé dans l’Apprentissage Supervisé.