CM 6
Méthodes de Simulation Informatique

Amaya Nogales Gómez
amaya.nogales-gomez@univ-cotedazur.fr

Licence 3 Informatique
Université Côte d'Azur

25 mars 2022

Plan du cours

1. Introduction
   - Préliminaires
   - Python: numpy, pandas
2. Base de données
   - Generation des données synthétiques
   - Base de données reels
3. Analyse descriptive
4. Techniques d'apprentissage supervisée
5. Techniques d'apprentissage non supervisée
6. Contrôle de connaissances
7. Techniques de validation
8. Elements de la méthodologie scientifique
9. LaTeX
   - Écriture de textes scientifiques
   - Beamer: présentations et posters scientifiques

Précision

Étant donnéun objet i, il est classé dans la classe positive ou négative selon la valeur de la fonction score, $sign(\omega^\top x_i + b)$, tandis que pour le cas $\omega^\top x_i + b = 0$, le l'objet est classé au hasard. La précision de la classification est définie comme le pourcentage d'objets correctement classés par le classifieur sur un base de données.

$$Précision = \frac{correct\,predictions}{total\,predictions} =$$

$$= P(\omega^\top x_i + b \geq 0 \wedge y_i = +1) + P(\omega^\top x_i + b < 0 \wedge y_i = -1)$$

Noyaux les plus utilisés

Noyau $k(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle = \phi(x_1)^\top \phi(x_2)$. Noyaux les plus populaires:

- Linear

$$k(x_1, x_2) = \langle x_1, x_2 \rangle$$

- Radial Basis Function (RBF)

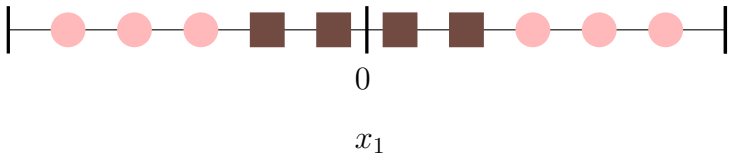
$$k(x_1, x_2) = e^{-\gamma \|x_1 - x_2\|^2}$$

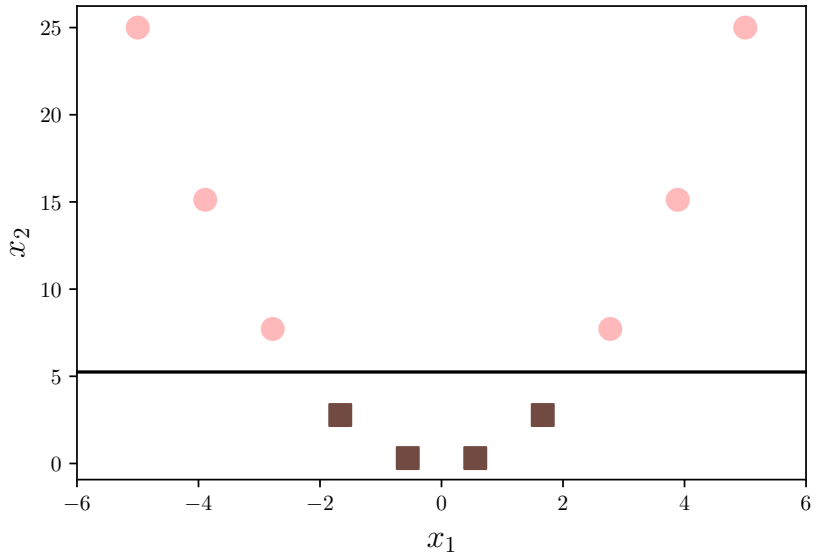
- Noyau polynomial (de dimension d):

$$k(x_1, x_2) = (x_1^\top x_2 + c)^d$$

Comment savons-nous que les noyaux aident à séparer les données?

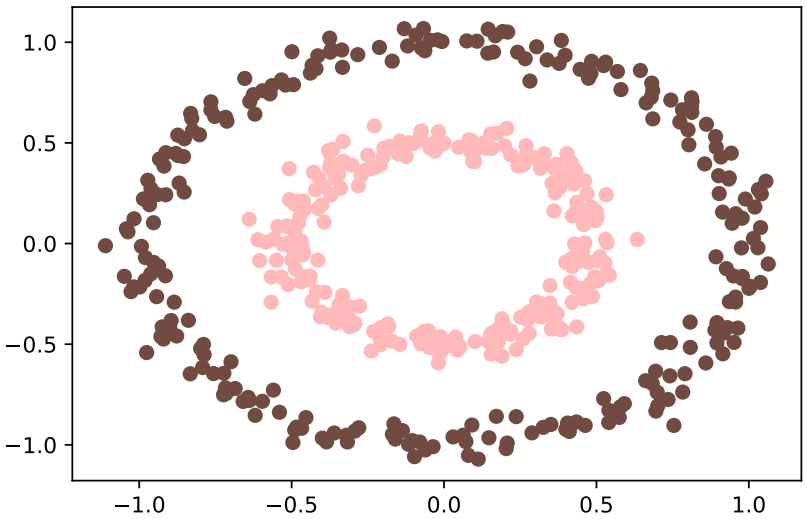
- Dans $\mathbb{R}^d$, tous les vecteurs indépendants d sont linéairement séparables.
- Si le noyau k est définie positif $\rightarrow$ données linéairement séparable.
- Exemple: $x_1 \in \mathbb{R}$, $\Phi(x_1) = (x_1, x_1^2) \in \mathbb{R}^2$

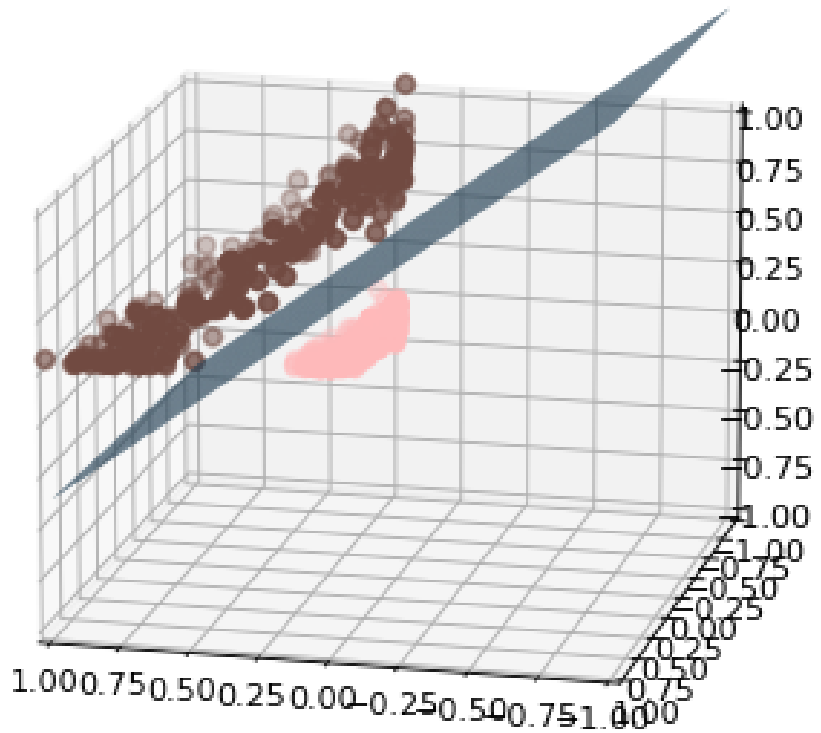




Exemple: $\mathbb{R}^2 \rightarrow \mathbb{R}^3$

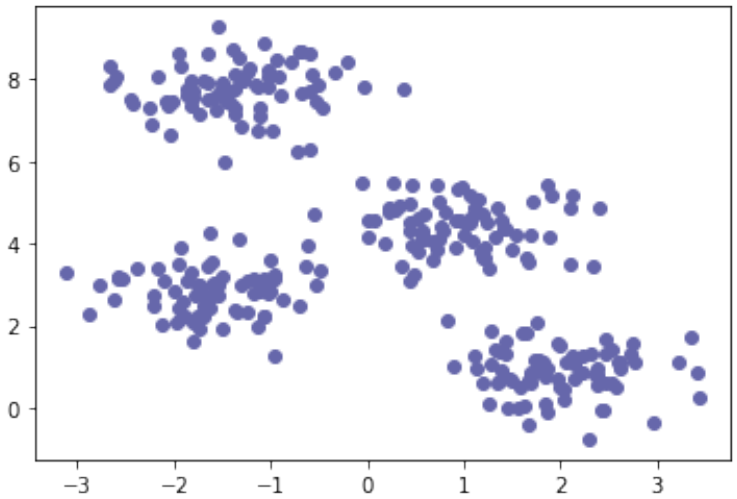
$$x \in \mathbb{R}^2, \phi(x) \in \mathbb{R}^3, \phi(x) = (x_1^2, \sqrt{2}x_1, x_2, x_2^2)$$

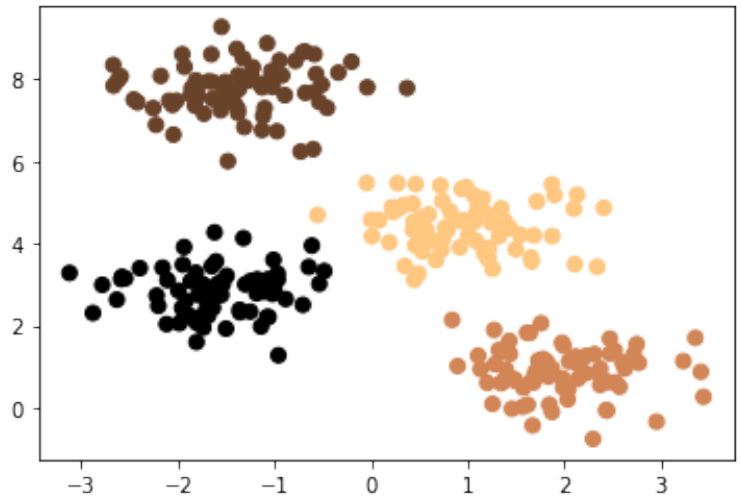




Apprentissage non supervisé

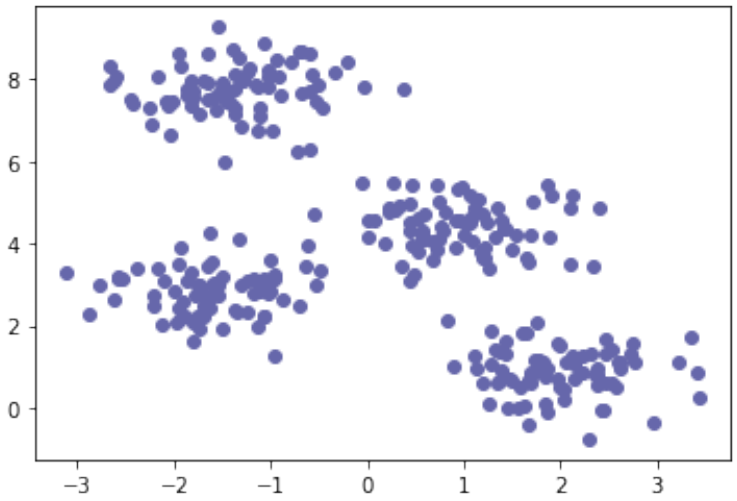
L'algorithme d'apprentissage non supervisé mets en évidence les groupes "naturels" c.-à -d. qui se démarquent significativement les uns des autres.





1. Combien de groupes?
2. Calcul de la délimitation des groupes

K-means: Une algorithme de regroupement itératif



- Initialiser: Choisissez K points au hasard comme centres de cluster
- Réitérer:
  1. Affectez des points de données au centre du cluster le plus proche
  2. Remplacer le centre du cluster pour la moyenne de ses points affectées
- Arrêtez quand aucun affectation de point change

Partitionnement K-moyennes

Étant donné un ensemble de points $(x_1, x_2, \ldots, x_n)$, on cherche à partitionner les n points en k ensembles (clusters) $C = \{C_1, C_2, \ldots, C_k\} (k \leq n)$ en minimisant la distance entre les points à l'intérieur de chaque partition :

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

oú μ_i est moyenne des points dans C_i, c'est-a-dire, $\mu_i = \frac{1}{|C_i|} \sum_{x \in C_i} x$.

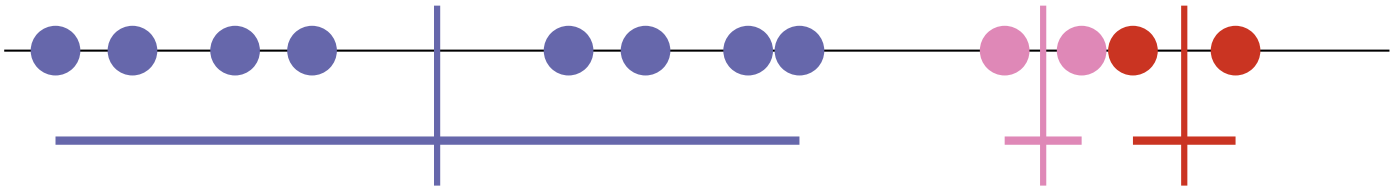
Inertia

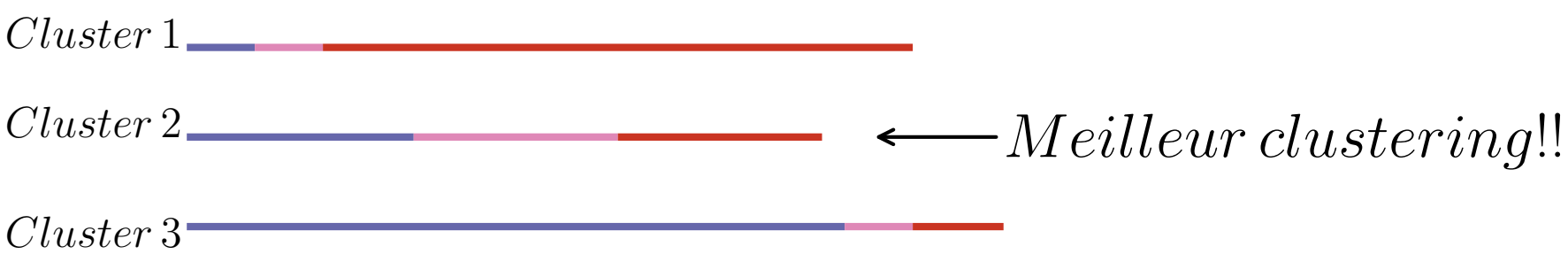
La inertia est la somme au carré des distances des données à leur centre de cluster le plus proche. C'est aussi appelé Within-Cluster Sum of Squares (WCSS).

$$\text{Inertia} = \sum_{i=1}^{k} \sum_{x_j \in C_i} \|x_j - \mu_i\|^2$$

Exemple en une dimension

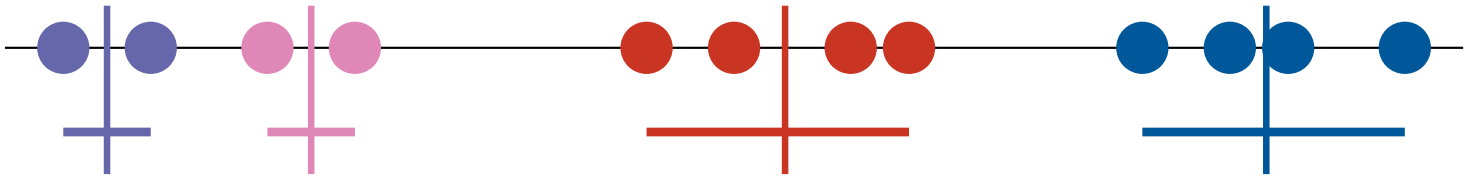
Á ce stade, K-means sait que le deuxième regroupement est le meilleur à ce jour. Mais il ne sait pas si c'est le meilleur dans l'ensemble, donc il fera quelques clusters supplémentaires (il en fera autant que vous lui dites de faire), puis reviendra et renverra le meilleur

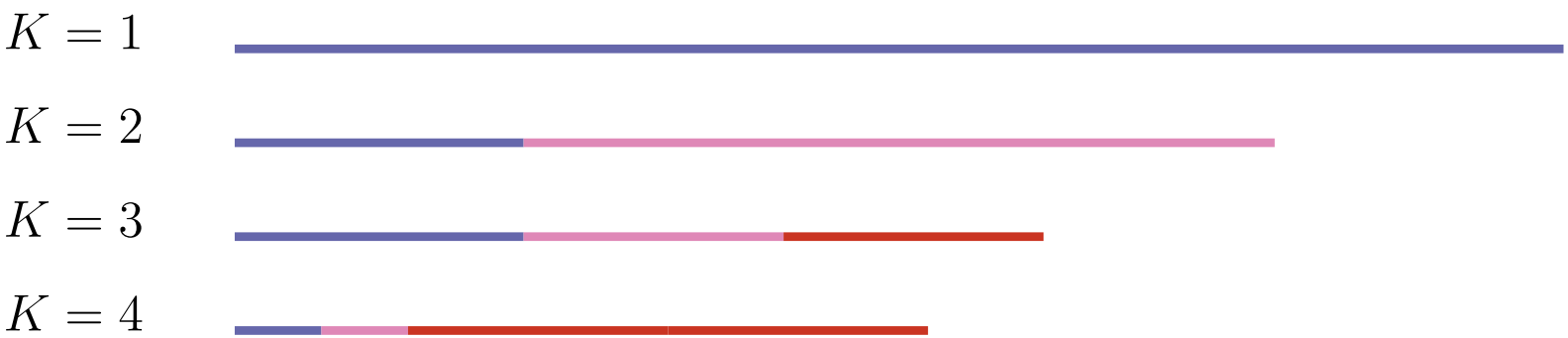




Exemple en une dimension: choix de k

Maintenant on essaie avec $K = 4$

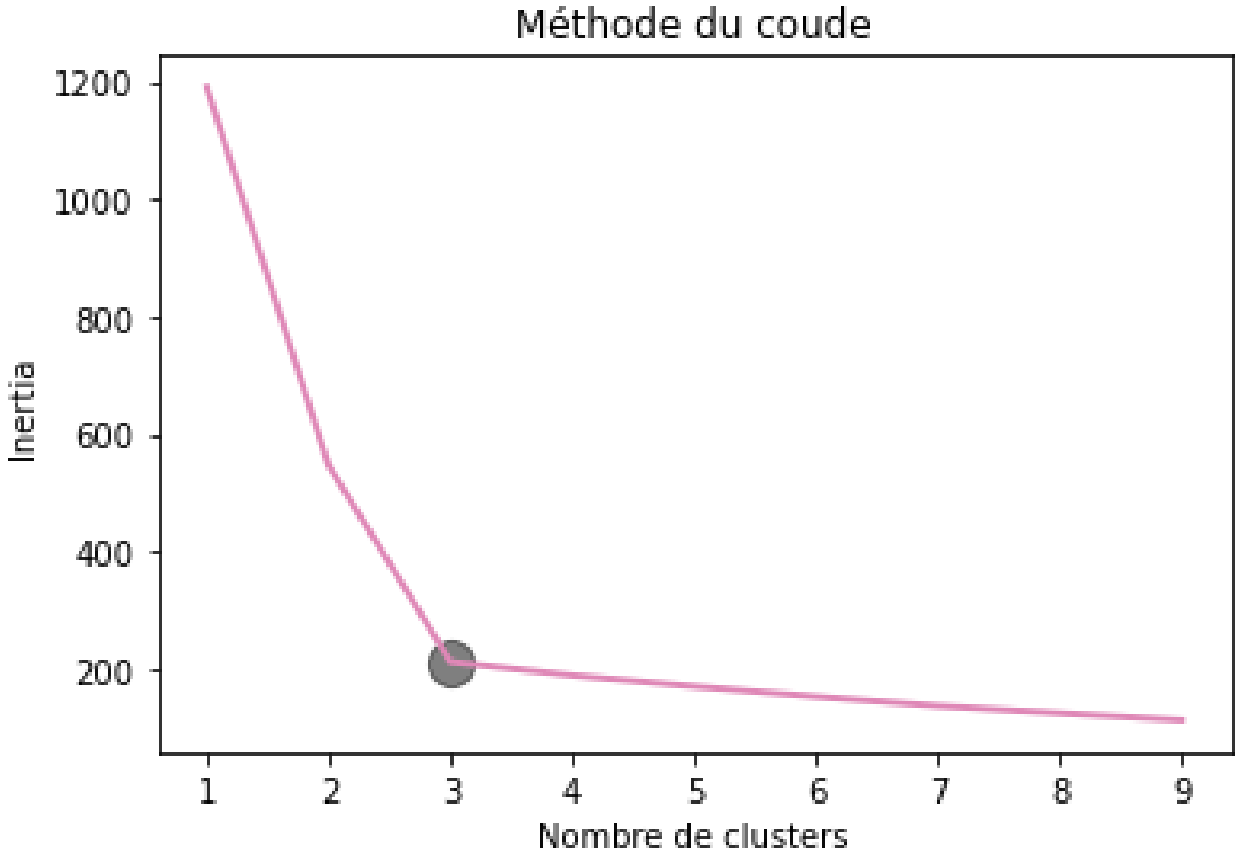




Le valeur d'inertia pour $K = 4$ est encore mieux que pour $K = 3$.

Chaque fois que nous ajoutons un nouveau cluster, la valeur d'inertia diminue. Et lorsqu'il n'y a qu'un seul point par cluster, l'inertia est égale à zéro.

C'est ce qu'on appelle le tracé du coude, et vous pouvez choisir le k optimal en trouvant le coude dans le tracé.



Il y a une énorme réduction de l'inertia lorsque $K = 3$, mais après cela, l'inertia ne diminue pas aussi rapidement

Methodes de validation: Cross-validation

La validation croisée (*cross-validation*) est, en apprentissage automatique, une méthode d'estimation de fiabilité d'un modèle fondé sur une technique d'échantillonnage.

- Supposons posséder un modèle statistique avec un ou plusieurs paramètres inconnus, et un ensemble de données d'apprentissage sur lequel on peut apprendre (ou « entraîner ») le modèle.
- Le processus d'apprentissage optimise les paramètres du modèle afin que celui-ci corresponde le mieux possible aux données d'apprentissage.

Methodes de validation: Cross-validation

- Si on prend ensuite un échantillon de validation indépendant, supposément issu de la même population que l'échantillon d'apprentissage, il s'avérera en général que le modèle ne modélise pas aussi bien les données de validation que les données d'apprentissage : on parle de surapprentissage (*underfitting*).
- De plus, d'un échantillon de validation à un autre, la performance de validation du modèle peut varier.

La validation croisée permet de tirer plusieurs ensembles de validation d'une même base de données et ainsi d'obtenir une estimation plus robuste de la performance de validation du modèle.

Techniques de validation

- **Sélection de modèles**: estimation des performances de différents modèles afin de choisir le meilleur.
- **Évaluation du modèle** : après avoir choisi un modèle final, estimer son erreur de prédiction (performance) sur les nouvelles données.

Dans un base de données, on va differencier entre:

1. Échantillon d'apprentissage (ou d'entraînement)
2. Échantillon de test
3. Échantillon de validation

Parfois:
Échantillon de test=Échantillon de validation

Évaluation du modèle

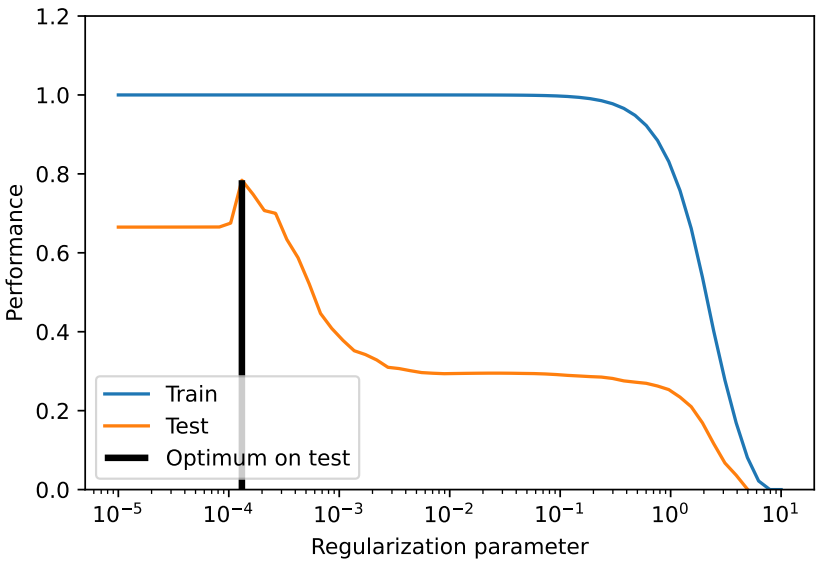
- Les performances de généralisation d'un méthode ML concerne sa capacité de prédiction sur des échantillon de test.
- L'évaluation de cette performance est extrêmement importante dans la pratique, puisqu'elle guide le choix du méthode ou modèle ML.
- De plus, cela nous donne une mesure de la qualité du modèle choisi.

Évaluation du modèle

- **Erreur de test**
  - L'erreur moyenne qui résulte de l'utilisation d'un méthode pour prédire la réponse sur une **nouvelle** observation.
  - L'erreur de prédiction sur un échantillon de test indépendant.
- **Erreur d'entraînement**
  - La perte moyenne sur l'échantillon d'apprentissage (*Mean Squared Error*):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Remarque : Le taux d'erreur d'apprentissage peut considérablement sous-estimer le taux d'erreur de tests.



- Le modèle devient de plus en plus complexe: utilise les données d'entraînement et est capable de s'adapter à structures plus compliquées.
- Il y a donc une diminution du biais (erreur) mais une augmentation de la variance.
- Cependant, l'erreur de formation n'est pas une bonne estimation de l'erreur de test.

- L'erreur d'apprentissage diminue systématiquement avec la complexité du modèle.
- Un modèle sans erreur d'apprentissage est surajusté aux données d'apprentissage et généralisera généralement mal.

Évaluation du modèle

- Si nous sommes dans une situation riche en données, la meilleure approche pour la sélection et l'évaluation du modèle consiste à diviser au hasard l'ensemble de données en trois parties : échantillon d'apprentissage, échantillon de validation et échantillon de test.
- L'échantillon d'apprentissage est utilisé pour ajuster les modèles. L'échantillon de test est utilisé pour estimer l'erreur de prédiction pour la sélection du modèle. L'échantillon de validation est utilisé pour l'évaluation de l'erreur de prédiction du modèle final choisi.
- Une répartition typique pourrait être de 50% pour l'échantillon d'apprentissage et de 25% chacun pour validation et test.

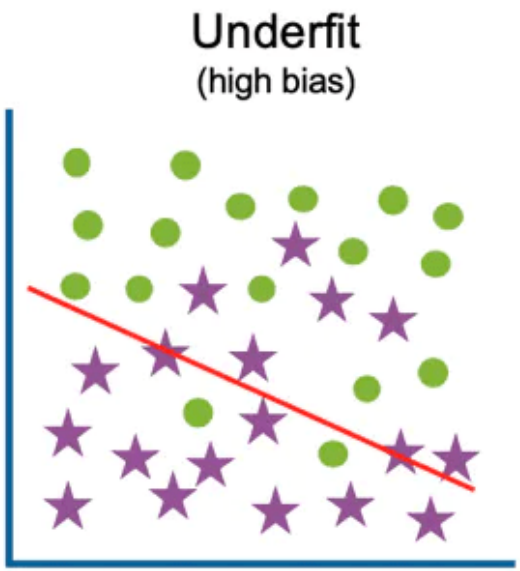
Sélection du modèle

- L'utilisation la plus importante de la validation concerne la sélection de modèles.
- Cela pourrait être le choix entre un modèle linéaire et un modèle non linéaire modèle, le choix de l'ordre du polynôme dans un modèle, le choix d'un paramètre de régularisation, ou tout autre choix affectant le processus d'apprentissage.
- Dans presque toutes les situations d'apprentissage, il y a des choix à faire et nous avons besoin d'une méthode fondée sur des principes pour faire ces choix.

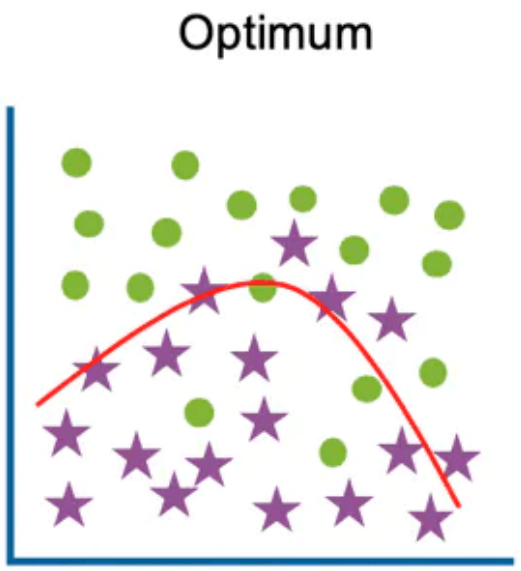
Sélection du modèle

- Supposons que nous ayons M modèles; les techniques de validation peuvent être utilisés pour sélectionner l'un de ces des modèles.
- Nous utilisons les données d'apprentissage pour ajuster le modèle, et nous évaluons chaque modèle sur l'échantillon de test pour obtenir le erreurs de test.
- Il est maintenant simple de sélectionner le modèle avec l'erreur de test le plus petit.
- Si le biais est élevé, nous devons permettre à notre modèle de être plus complexe.
- Si la variance est élevée, nous devons réduire la complexité du modèle.

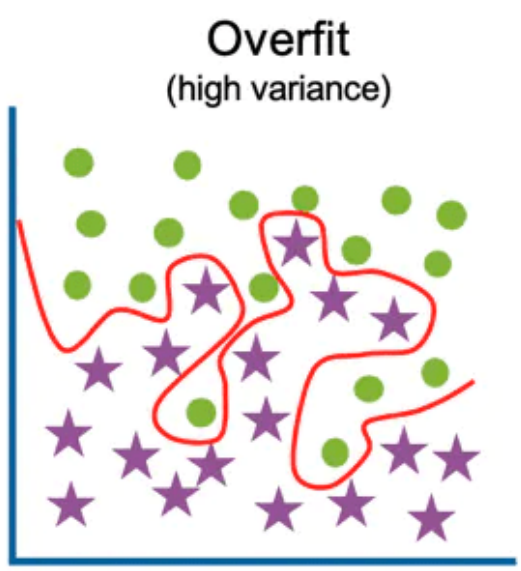
Underfitting et overfitting



High training error
High test error



Low training error
Low test error



Low training error
High test error

Source d'image: `https://www.ibm.com/cloud/learn/underfitting`

Exemple: Choix du parametre C dans le SVM.

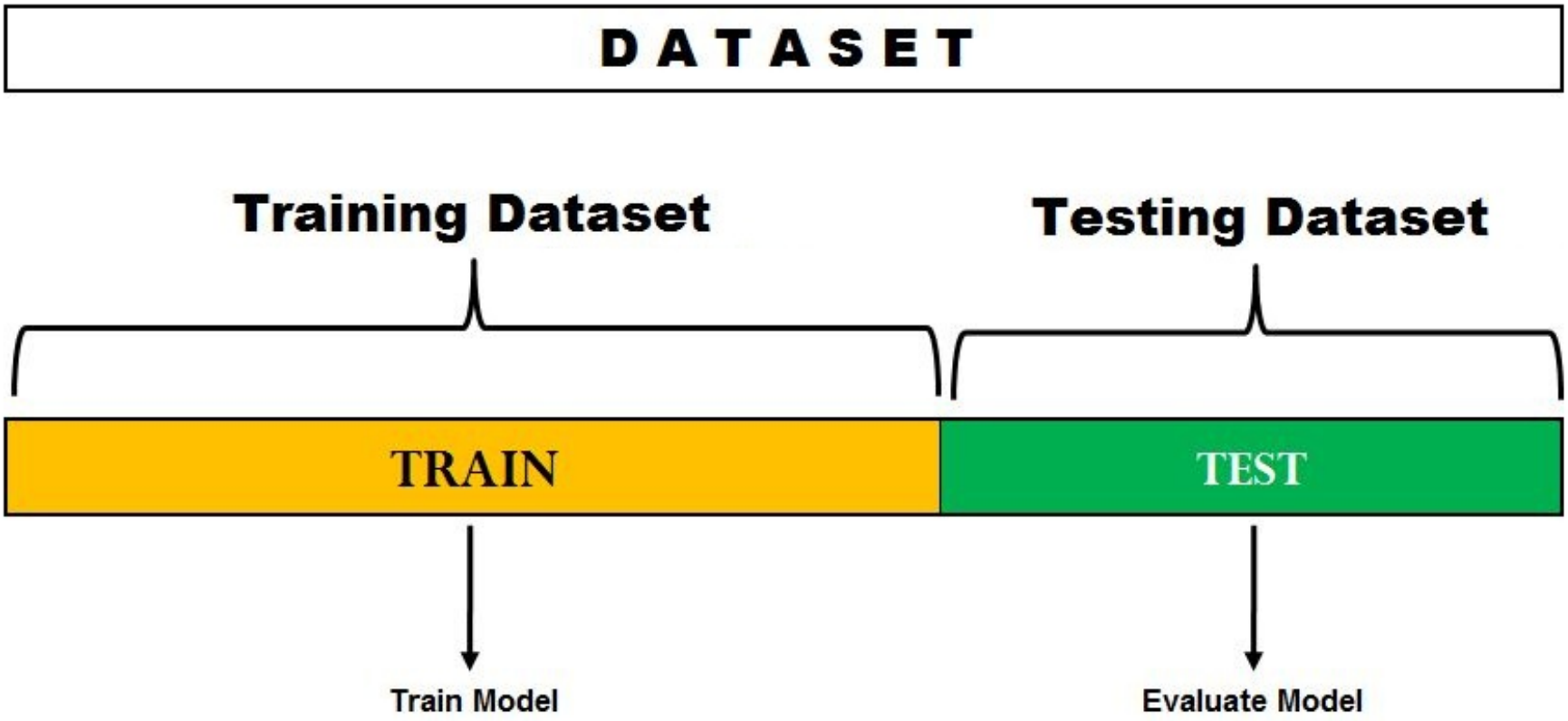
Pour donner la precision finale d'un ensemble de données, nous devons:

1. Entraînez le SVM dans l'échantillon d'apprentissage pour tous les valeurs de C.
2. Choisissez C^* (optimal) comme celui donnant la précision maximale dans l'échantillon de test.
3. Rapportez la précision dans l'échantillon de validation.

Méthode Hold-out: validation non-croisée

- La méthode d'évaluation la plus simple et largement utilisée dans les projets d'apprentissage automatique.
- L'ensemble de données (population) est divisé en 2 ensembles - échantillon d'apprentissage et échantillon de test.
- Les données peuvent être divisées en 70-30 ou 60-40, 75-25 ou 80-20, voire 50-50 selon le cas d'utilisation.
- En règle générale, la proportion de données d'apprentissage doit être supérieure aux données de test.

Hold-out



Source d'image: DataVedas

Hold-out

Le fractionnement des données se produit de manière aléatoire, et nous ne pouvons pas être sûrs des données qui se retrouvent dans le échantillon d'entraînement et de test pendant le fractionnement, sauf si nous spécifions *random_state*.
Cela peut entraîner une variance extrêmement élevée et chaque fois que la répartition change, la précision change également.

Hold-out: inconvénients

1. Taux d'erreur de test sont très variables (variance élevée) et cela dépend totalement des observations qui se retrouvent dans l'échantillon d'apprentissage et l'échantillon de test.
2. Seule une partie des données est utilisée pour former le modèle (biais élevé), ce qui n'est pas une très bonne idée lorsque la taille de la base est petite.
3. Surestimation de l'erreur de test.
   Avantage: peu coûteuse en termes de calcul par rapport aux autres techniques de validation croisée.

Hold-out

```
from sklearn.model_selection import train_test_split
X = np.array([1,2,3,4,5,6,7,8,9,10])
X_train, X_test= train_test_split(X,test_size=0.3, random_state=1)
print('Train:',X_train,'Test:' ,X_test)
```

```
Train: [5 1 4 2 8 9 6] Test: [ 3 10  7]
```

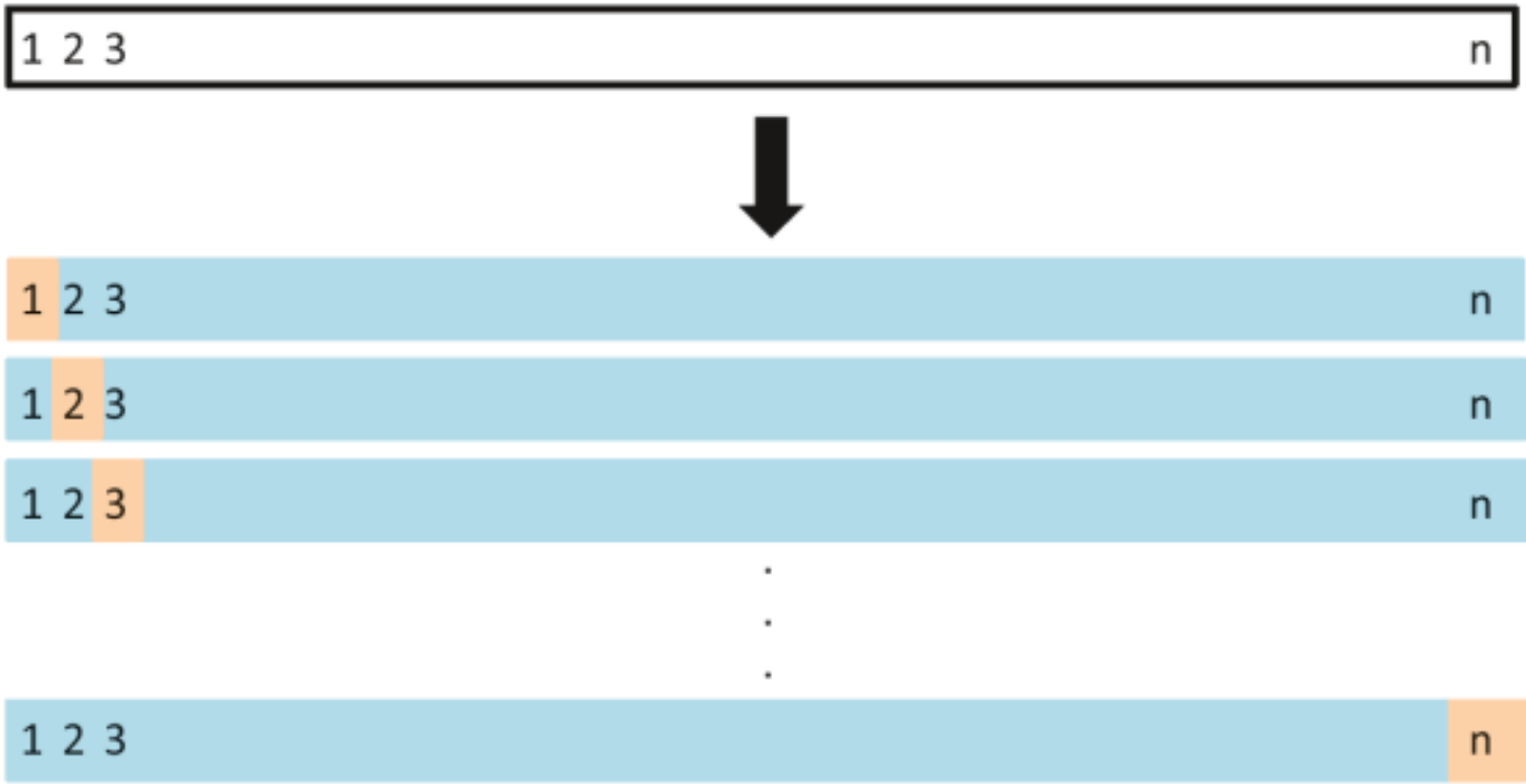
```
X = np.array([1,2,3,4,5,6,7,8,9,10])
X_train, X_test= train_test_split(X,test_size=0.3, shuffle=False)
print('Train:',X_train,'Test:' ,X_test)
```

```
Train: [1 2 3 4 5 6 7] Test: [ 8  9 10]
```

Leave One Out Cross-Validation (LOOCV)

- Dans cette méthode, nous divisons les données en échantillon d'entraînement et de test - mais avec une difference.
- Au lieu de diviser les données en 2 sous-ensembles, nous sélectionnons une seule observation comme échantillon de test, et tout le reste est étiqueté comme échantillon d'apprentissage.
- Maintenant, la 2ème observation est sélectionnée comme échantillon de test et le modèle est formé sur les données restantes.

Leave One Out Cross-Validation



Leave One Out Cross-Validation

Ce processus se poursuit « n » fois et la moyenne de toutes ces itérations est calculée et estimée comme l'erreur de l'échantillon de test.

$$error_{(n)} = \frac{1}{n}\sum_{i=1}^{n} MSE_i \tag{1}$$

- En ce qui concerne les estimations d'erreur de test, LOOCV donne des estimations non biaisées (faible biais).
- Mais le biais n'est pas le seul sujet de préoccupation dans les problèmes d'estimation.
- Nous devrions également considérer la variance.

Leave One Out Cross-Validation

- LOOCV a une variance extrêmement élevée car nous faisons la moyenne de la sortie de n-modèles qui sont ajustés sur un ensemble d'observations presque identique, et leurs sorties sont fortement corrélées positivement les unes avec les autres.
- Cela coûte cher en termes de calcul, car le modèle est exécuté "n" fois pour tester chaque observation dans les données.
- Notre prochaine méthode s'attaquera à ce problème et nous donnera un bon équilibre entre biais et variance.

Leave One Out Cross-Validation

```
from sklearn.model_selection import LeaveOneOut
X = np.array([10,20,30,40,50,60,70,80,90,100])

l = LeaveOneOut()

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

    print("%s %s"% (train,test))
```

```
[1 2 3 4 5 6 7 8 9] [0]
[0 2 3 4 5 6 7 8 9] [1]
[0 1 3 4 5 6 7 8 9] [2]
[0 1 2 4 5 6 7 8 9] [3]
[0 1 2 3 5 6 7 8 9] [4]
[0 1 2 3 4 6 7 8 9] [5]
[0 1 2 3 4 5 7 8 9] [6]
[0 1 2 3 4 5 6 8 9] [7]
[0 1 2 3 4 5 6 7 9] [8]
[0 1 2 3 4 5 6 7 8] [9]
```

Leave One Out Cross-Validation

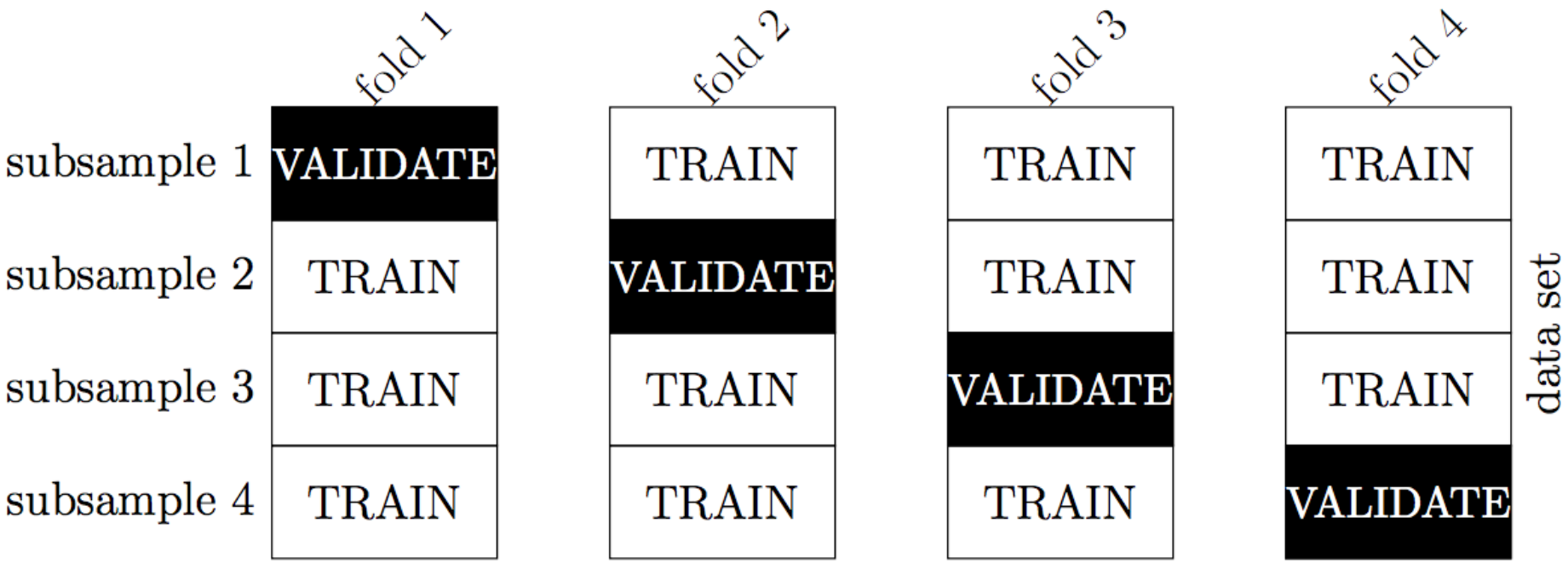
```
for train_index, test_index in l.split(X):
    X_train, X_test = X[train_index], X[test_index]
    print(X_train, X_test)

    #print("%s %s"% (X[train_index],X[test_index]))
```

```
[ 20  30  40  50  60  70  80  90 100] [10]
[ 10  30  40  50  60  70  80  90 100] [20]
[ 10  20  40  50  60  70  80  90 100] [30]
[ 10  20  30  50  60  70  80  90 100] [40]
[ 10  20  30  40  60  70  80  90 100] [50]
[ 10  20  30  40  50  70  80  90 100] [60]
[ 10  20  30  40  50  60  80  90 100] [70]
[ 10  20  30  40  50  60  70  90 100] [80]
[ 10  20  30  40  50  60  70  80 100] [90]
[10 20 30 40 50 60 70 80 90] [100]
```

K-fold Cross-Validation

- Dans cette technique de rééchantillonnage, l'ensemble des données est divisé en k ensembles de tailles presque égales.
- Le premier ensemble est sélectionné comme échantillon de test et le modèle est entraîné sur les $k-1$ ensembles restants.
- Le taux d'erreur de test est ensuite calculé après ajustement du modèle aux données de test.
- Dans la deuxième itération, le 2ème ensemble est sélectionné comme échantillon de test et les $k-1$ ensembles restants sont utilisés comme échantillon d'apprentissage et l'erreur est calculée.
- Ce processus continue pour tous les k ensembles.

K-fold Cross-Validation



K-fold Cross-Validation

$$error_{(k)} = \frac{1}{k}\sum_{i=1}^{k} MSE_i \tag{2}$$

- Dans K-Fold CV, le nombre d'ensembles (*folds*) k est inférieur au nombre d'observations dans les données ($k < n$).
- Nous faisons la moyenne des sorties de k modèles ajustés qui sont un peu moins corrélés entre eux puisque le chevauchement entre les échantillons d'apprentissage est plus petit.
- Cela conduit à une faible variance en comparaison avec LOOCV.
- La meilleure partie de cette méthode est que chaque point de données se trouve exactement une fois dans l'échantillon de test et fait partie de l'échantillon d'apprentissage $k - 1$ fois.

K-fold Cross-Validation

- En règle générale, la K-fold CV est effectuée en utilisant $k = 5$ ou $k = 10$ car il a été démontré empiriquement que ces valeurs donnent des estimations d'erreur de test qui n'ont ni biais ni variance élevés.
- Le principal inconvénient de cette méthode est que le modèle doit être exécuté à partir de zéro k fois.
- C'est plus coûteux en calcul que la méthode Hold Out mais meilleur que la méthode Leave One Out.

K-fold Cross-Validation

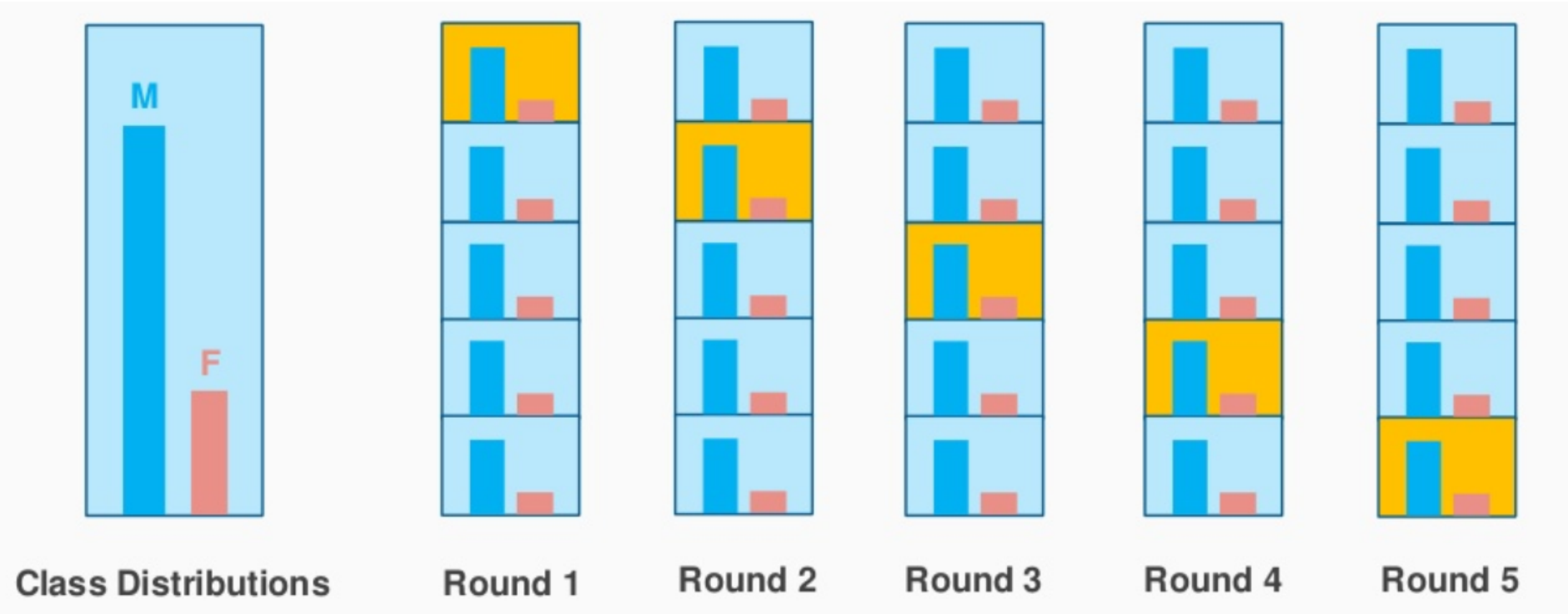
```
from sklearn.model_selection import KFold
X = np.array([10,20,30,40,50,60,70,80,90,100])

kf = KFold(n_splits=5, shuffle=False, random_state=None)

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

    print("Train data",X[train],"Test data",X[test])
```

```
Train data [ 30  40  50  60  70  80  90 100] Test data [10 20]
Train data [ 10  20  50  60  70  80  90 100] Test data [30 40]
Train data [ 10  20  30  40  70  80  90 100] Test data [50 60]
Train data [ 10  20  30  40  50  60  90 100] Test data [70 80]
Train data [10 20 30 40 50 60 70 80] Test data [ 90 100]
```

Stratified K-Fold Cross-Validation

- Il s'agit d'une légère variation par rapport à la validation croisée K-Fold, qui utilise un "échantillonnage stratifié" au lieu d'un "échantillonnage aléatoire".
- Qu'est-ce qu'est l'échantillonnage **stratifié** et en quoi il diffère de l'échantillonnage aléatoire?
- Supposons que vos données contiennent des avis sur un produit cosmétique utilisé à la fois par la population masculine et féminine. Lorsque nous effectuons un échantillonnage aléatoire pour diviser les données en échantillons d'apprentissage et de test, il est possible que la plupart des données représentant les hommes ne soient pas représentées dans les données d'apprentissage mais se retrouvent dans les données de test. Lorsque nous formons le modèle sur des échantillons d'apprentissage qui ne sont pas une représentation correcte de la population réelle, le modèle ne prédit pas les données de test avec une bonne précision.

Stratified K-Fold Cross-Validation



Exemple: Stratified K-Fold Cross-Validation

- Considérons l'exemple ci-dessus qui a une revue de produits cosmétiques de 1000 clients dont 60% sont des femmes et 40% sont des hommes.
- On voudrait diviser les données en échantillons d'apprentissage et test en proportion (80:20).
- 80% sur 1000 clients seront 800 qui seront choisis de telle façon qu'il y ait 480 avis associés à la population féminine et 320 représentant la population masculine.
- De la même manière, 20% de 1000 clients seront choisis pour l'échantillon de test (120 femmes, 80 hommes).

Stratified K-Fold Cross-Validation

- Le K-fold CV stratifié cree d'échantillons en préservant le pourcentage de representation pour chaque classe.
- Cela résout le problème d'échantillonnage aléatoire associé aux méthodes Hold out et K-Fold.

Stratified K-Fold Cross-Validation

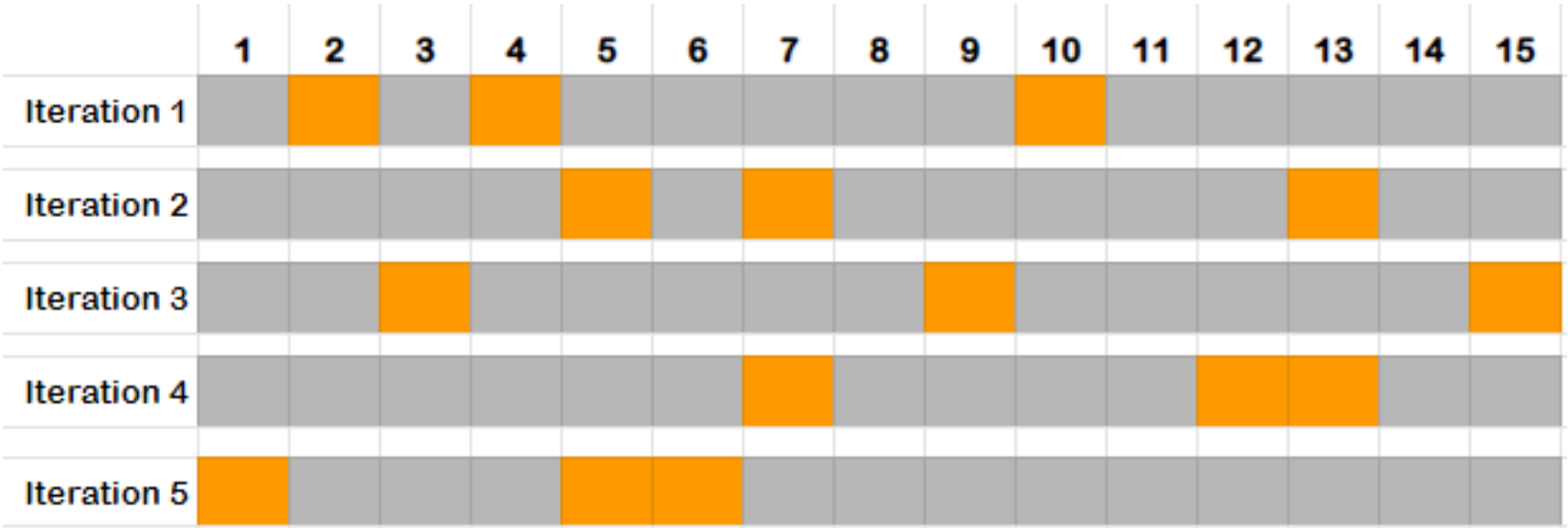
```
from sklearn.model_selection import StratifiedKFold
X = np.array(['a','b','c','d','e','f'])
y= np.array([0,0,0,1,1,1])

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

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

```
Train: ['b' 'c' 'e' 'f'] Test: ['a' 'd']
Train: ['a' 'c' 'd' 'f'] Test: ['b' 'e']
Train: ['a' 'b' 'd' 'e'] Test: ['c' 'f']
```

Repeated random subsampling

- La validation **Repeated random subsampling**, également appelée validation Monte Carlo , divise l'ensemble de données de manière aléatoire en échantillons d'apprentissage et test de façon repetée (nombre d'itérations).

Repeated random subsampling



Repeated random subsampling

Le nombre d'itérations n'est pas fixé et est décidé par nous même.
Les resultats sont ensuite moyennés sur les itérations.

$$error_{(k)} = \frac{1}{k} \sum_{i=1}^{k} MSE_i \tag{3}$$

Repeated random subsampling

Avantages: La taille d'échantillons de train et de validation ne dépend pas du nombre d'itérations ou de partitions.

Inconvénients: Certains échantillons peuvent ne pas être sélectionnés pour l'apprentissage ou la validation.

Pas conveniente pour un base de données déséquilibré.

Exemple: K-fold vs repeated random subsampling

Nous avons un base de données de taille $n = 100$. Cela implique que si nous choisissons la validation croisée 10 fois (K-fold pour $k = 10$), les tailles résultantes pour nos différents échantillon seraient :
apprentissage:90
test: 10
Alors que pour repeated random subsampling cross-validation pour 10 échantillonnages, selon le TP (apprentissage 50%, test 25%, validation 25%), les tailles étaient:
apprentissage=50
test=25
validation=25
Ici, la probabilité d'être ou non dans l'ensemble d'apprentissage pour une observation est de 0.5, mais nous répétons l'expérience 10 fois, il est donc très peu probable que le fait qu'une observation ne soit jamais choisie dans l'ensemble d'apprentissage posera un problème. D'un autre côté, avoir un ensemble de test de taille 10 pourrait causer des problèmes.