UNIVERSITE BADJI MOKHTAR-ANNABA

DEPARTEMENT D'INFORMATIQUE
M1 IATI

Problèmes et défis de la classification

par

Dr. Samira LAGRINI

Année universitaire:2024/2025

Introduction

- Bien que la classification soit largement utilisée dans de nombreux domaines, tels que la reconnaissance d'images, le traitement du langage naturel, et la détection de fraude, elle présente de nombreux défis qui peuvent affecter la précision et la robustesse des modèles

- L'objectif de ce cours est d'explorer les différents problèmes rencontrés lors de la classification et d'examiner les solutions possibles.

Problèmes de la classification

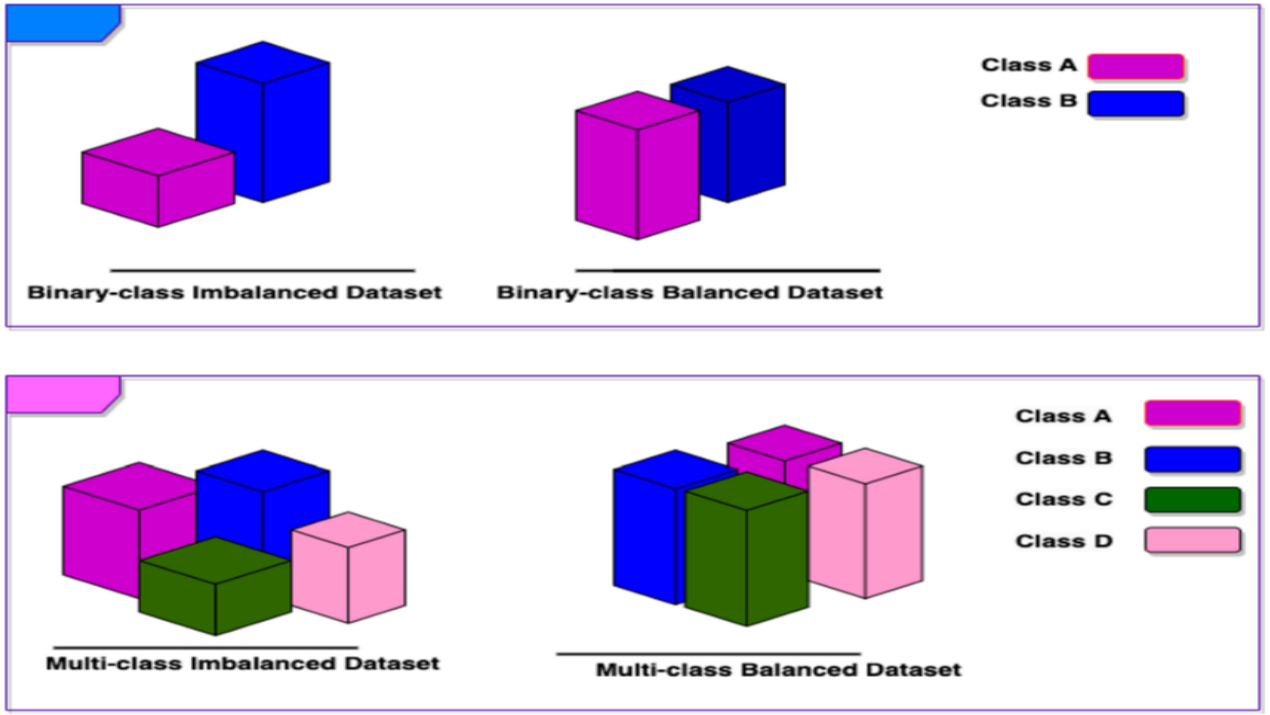
- ❑ Le déséquilibre des classes (Class Imbalance)
- ❑ Overfitting
- ❑Underfitting

Déséquilibre des classes (Class Imbalance)

- Un problème fréquent dans les tâches de classification où certaines classes sont beaucoup plus représentées que d'autres dans le dataset

Exemple:

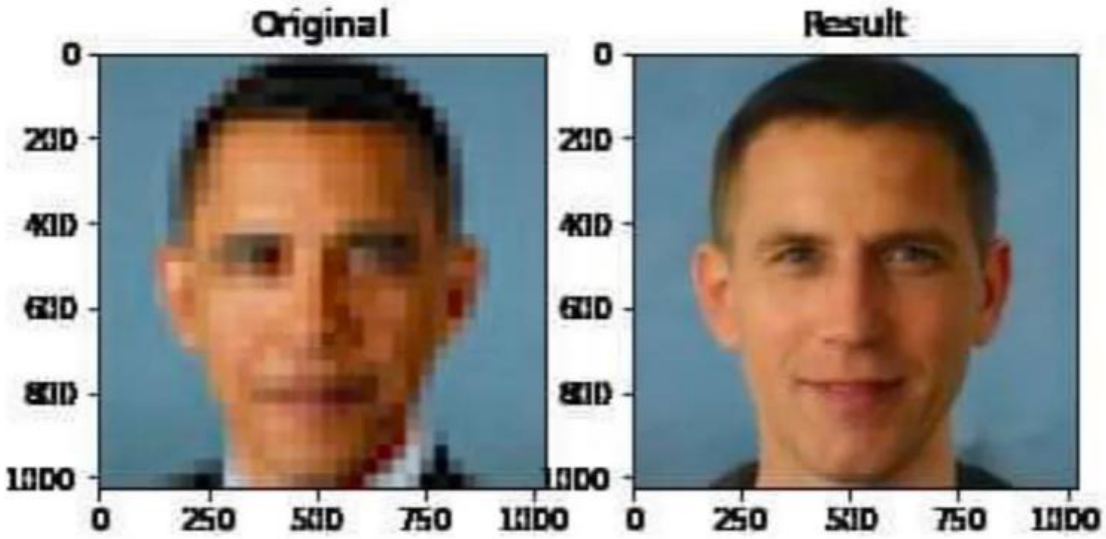
Dans un jeu de données sur les fraudes bancaires, où 98% des transactions sont légitimes et 2% sont frauduleuses, il y a un **déséquilibre des classes** important.



Déséquilibre des classes – **Conséquences**-

- **Biais du modèle**

L'algorithme peut être biaisé vers la classe majoritaire et ignorer les classes minoritaires, ce qui nuit à la précision des prédictions pour ces dernières.



- **Haute précision mais faible performance sur les classes minoritaires** qui sont souvent présentant le plus grand intérêt.

- **Généralisation limitée** pour la classe minoritaire

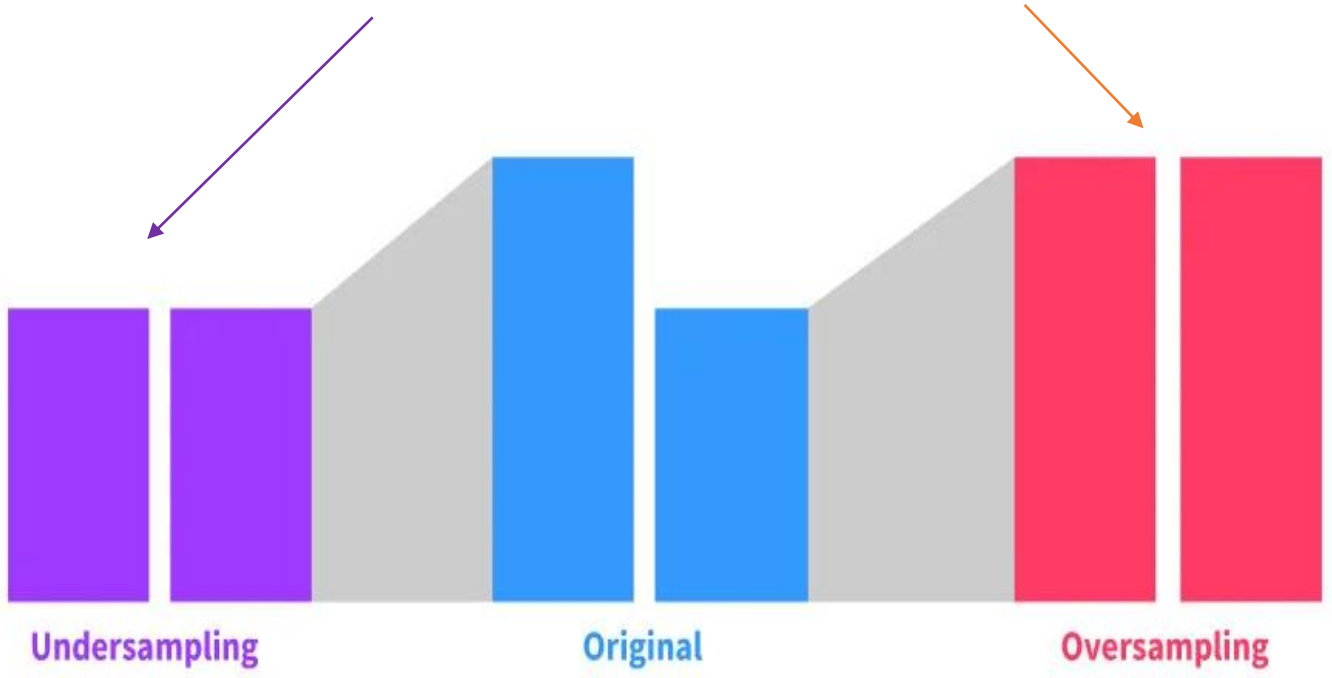
Stratégies pour Gérer le Déséquilibre des Classes

Rééchantillonnage (Resampling)

- Le rééchantillonnage consiste à **rééquilibrer** la distribution des classes, afin d'améliorer l'apprentissage des modèles, en particulier pour les classes minoritaires

- Ses méthodes sont classées en deux catégories :

le **sous-échantillonnage** et le **sur-échantillonnage**



Sous-échantillonnage

- Réduire le nombre d'exemples dans la classe majoritaire pour la rendre égale à la classe minoritaire
- Cette méthode peut entraîner une perte d'informations importantes ce qui peut nuire à la performance du modèle.

Exemple de sous-échantillonage:

Le Random Under-sampling de la bibliothèque imbalanced-learn en Python effectue une suppression aléatoire d'exemples de la classe majoritaire afin d'égaliser le nombre d'exemples entre les différentes classes.

Sous-échantillonnage - Tomek Links-

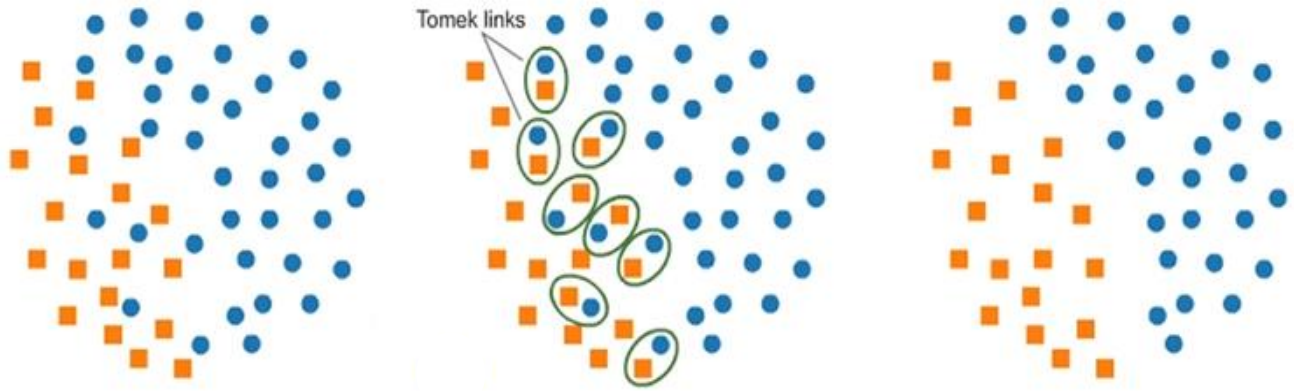
- Tomek Links, est une méthode de sous-échantillonnage (undersampling) utilisée pour éliminer les **exemples frontaliers** (*qui se situent à la frontière entre les classes*), ce qui permet de mieux séparer les classes et d'améliorer la performance du modèle.

Comment ça fonctionne ?

1. Pour chaque exemple dans le dataset, on cherche son plus proche voisin dans une autre classe.

2. Si un exemple de la classe majoritaire a son voisin le plus proche dans la classe minoritaire, cette paire est un Tomek Link.

3.L'exemple de la classe majoritaire faisant partie de cette paire est alors supprimé du dataset.



Sur-échantillonnage (Oversampling)

- Augmenter la taille de la classe minoritaire en dupliquant des instances ou en générant de nouvelles instances synthétiques.

- La **duplication** des instances de la classe minoritaire peut entraîner un **surapprentissage**, car elle ne génère pas de nouvelles informations

Alors, comment générer des nouvelles instances synthétiques ??!!!

Méthodes de Génération de données synthétiques

SMOTE

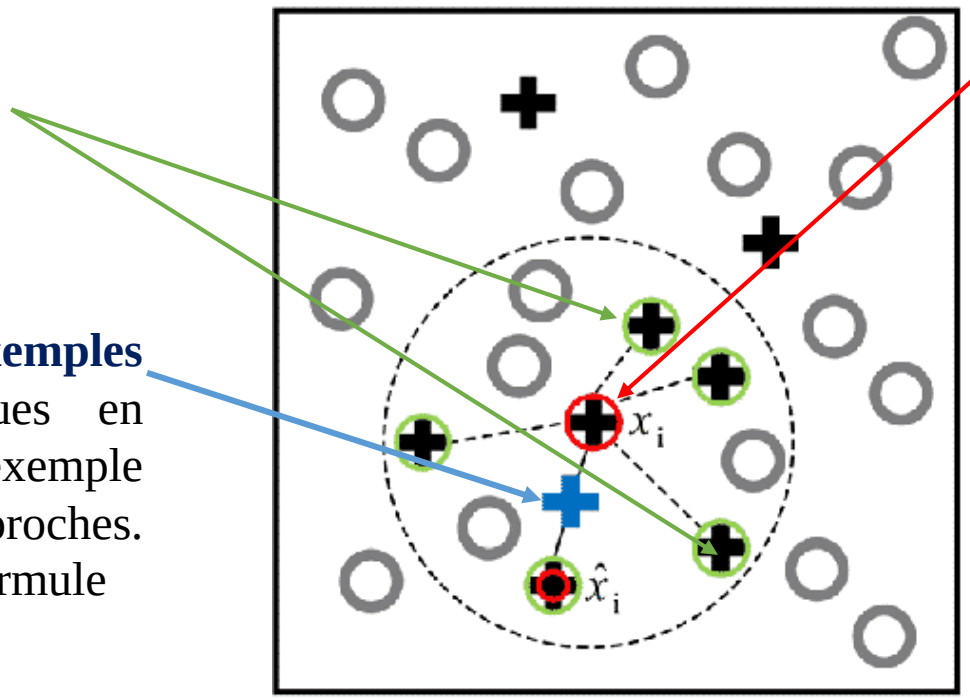
❑ **SMOTE(S**ynthetic **M**inority **O**ver-sampling **T**echnique).

Génère de nouvelles instances en créant des interpolations entre des points voisins de la classe minoritaire.

Comment ça fonctionne?

1. **Sélection d'un exemple de la classe minoritaire**

2. Sélectionner ses **k plus proches voisins** dans la même classe.



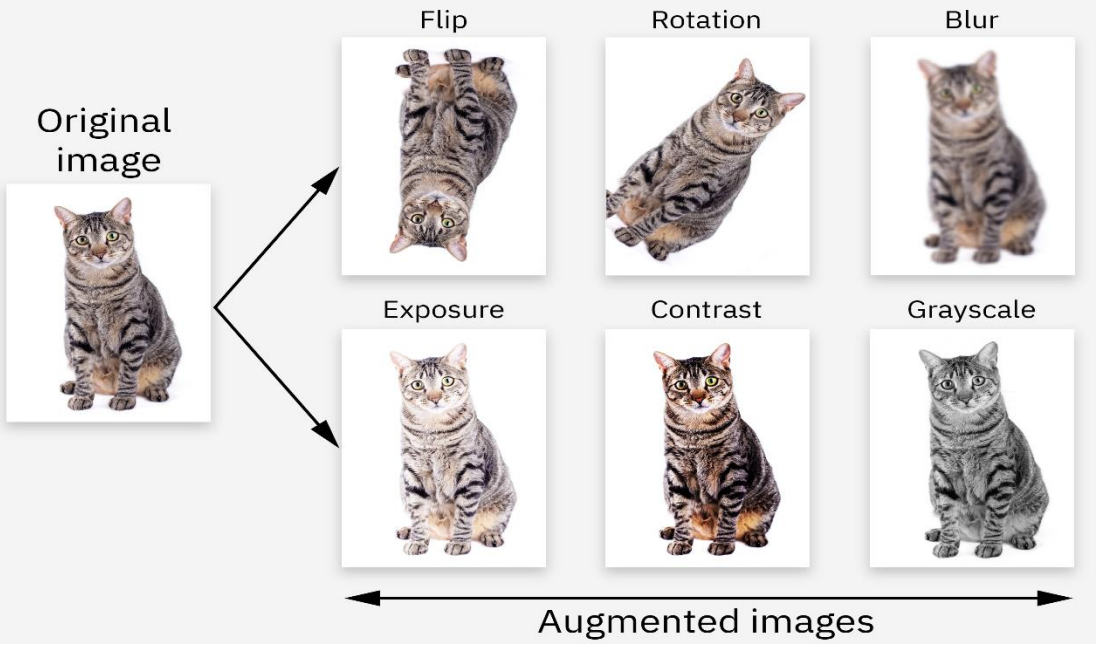
3. Interpolation des nouveaux exemples Génération des exemples synthétiques en interpolant de manière aléatoire entre l'exemple sélectionné et ses voisins les plus proches. L'interpolation est effectuée selon une formule

4. Génération de nouveaux exemples en répétant le processus avec les différents voisins

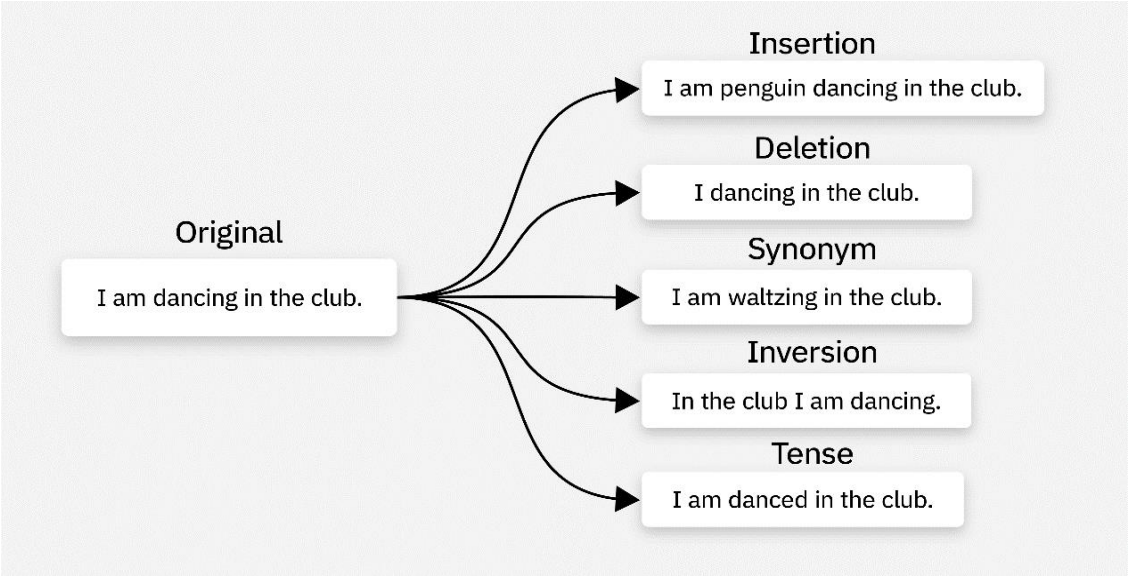
Augmentation de données

❑ Consiste à appliquer des transformations aux données existantes pour créer de nouvelles variations.

❑ Pour les images, ces transformations peuvent inclure des rotations, des redimensionnements, des changements de luminosité ou l'ajout de bruit.



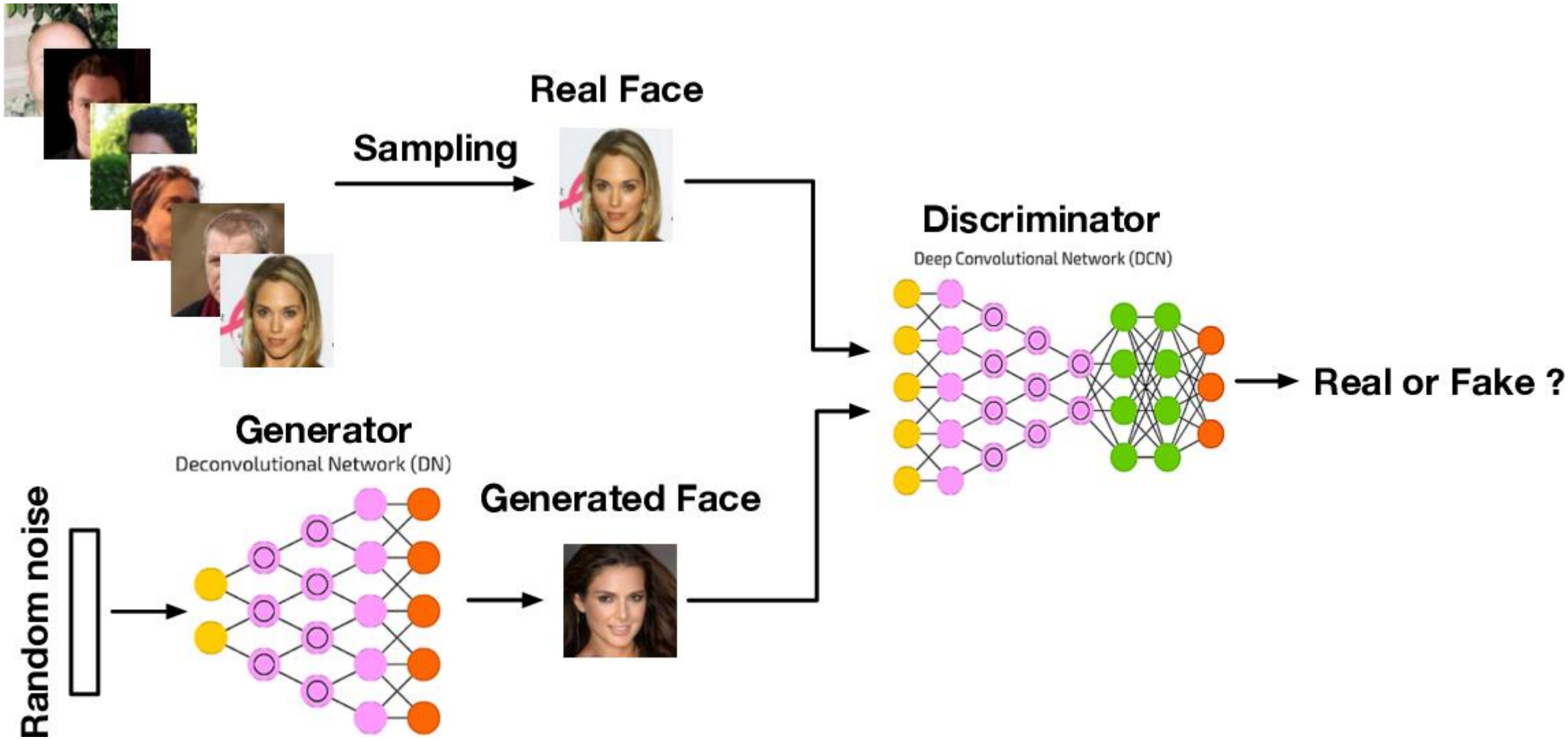
❑ Pour les données textuelles, des techniques comme la **paraphrase**, la substitution par synonymes, ou l'utilisation de modèles de génération de texte (comme GPT) peuvent être utilisées pour augmenter la classe minoritaire.



Générateurs adverses (*GANs*)

- Les réseaux antagonistes génératifs (GANs) représentent une approche plus avancée pour la génération de données synthétiques.
- Les GANs sont composés de deux réseaux neuronaux : un générateur et un discriminateur, qui sont entraînés simultanément dans un cadre compétitif.
- Le générateur est responsable de la création de données synthétiques à partir de bruit aléatoire. Son but est de produire des données qui semblent aussi réelles que possible.
- Le discriminateur a pour but de distinguer les données réelles des données générées.
- Le générateur et le discriminateur sont entraînés simultanément de manière à ce que, à chaque itération, le générateur devienne meilleur pour produire des données réalistes, et le discriminateur devienne meilleur pour identifier les fausses données.

Générateurs adverses (*GANs*)



Techniques hybrides

- Ces techniques combinent le sur-échantillonnage et le sous-échantillonnage pour obtenir de meilleurs résultats.

Par exemple

- Dans la méthode **SMOTE-Tomek**, SMOTE génère des exemples synthétiques pour la classe minoritaire, puis **Tomek Links** est appliqué pour supprimer les exemples de la classe majoritaire qui se trouvent à proximité des exemples de la classe minoritaire.

Overfitting (Surapprentissage)

- L'overfitting survient lorsqu'un modèle apprend **tres bien** les données d'entraînement, au point de **mémoriser** leur bruit et leurs particularités au lieu d'en extraire des motifs généraux.

Conséquences :

- Le modèle présente **de très bonnes performances sur les données d’entraînement**, mais **échoue à généraliser** à de nouvelles données (test ou données réelles).

Overfitting : les causes

- Modèle trop complexe (comportant trop de paramètres par rapport aux données d'entraînement)
- Les données d'entraînement ne sont pas représentatives de la population cible ou manquent de diversité.
- Un faible volume de données d’entraînement
- Nombre excessif d’époques d’apprentissage (training trop long)
- Un trop grand nombre de caractéristiques

Les signes d’un problème d’overfitting

- Un **écart important** entre les performances du modèle sur les données d’entraînement et de validation.

- Une courbe d’erreur (perte) d’entraînement qui **continue de diminuer** tandis que la courbe d’erreur de validation stagne ou augment.

Cela signifie quoi????

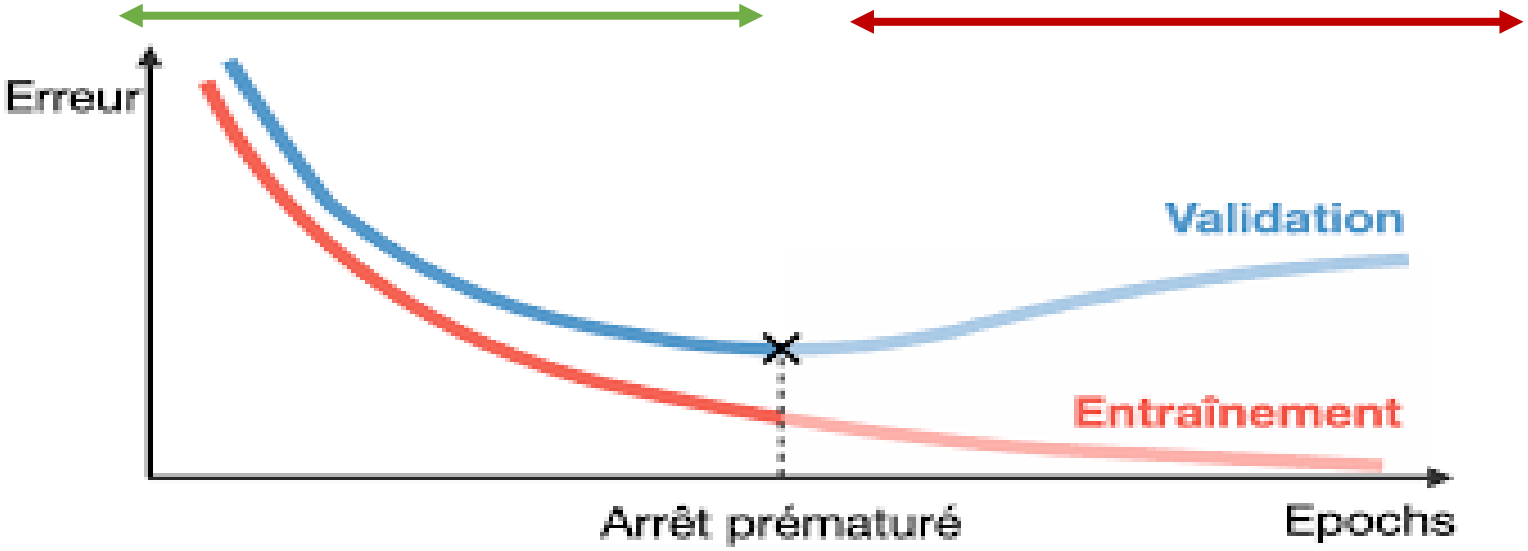
- La **perte** (**loss**) est une fonction qui mesure **l'écart** entre les prédictions du modèle et les valeurs réelles
- Lorsque le modèle s’entraîne, il ajuste ses paramètres pour minimiser la perte sur l'ensemble **d'entraînement**.

❖ **Avant le surapprentissage**

La perte sur les données d'entraînement et de validation diminuent de manière similaire

❖ **Pendant et après le surapprentissage**

La perte d'entraînement continue de diminuer, mais la perte de validation **commence à augmenter** après un certain nombre d'époques



Pour détecter le surapprentissage, il est important de comparer la perte entre les données d'entraînement et les données de validation

Comment éviter le surapprentissage?

1) Arrêter l'entraînement dès que la perte de validation commence à augmenter

2) Augmenter la taille et de la diversité de données d’entraînement afin que le modèle peut capturer des schémas plus généraux

3) Sélectionner que les caractéristiques les plus pertinentes

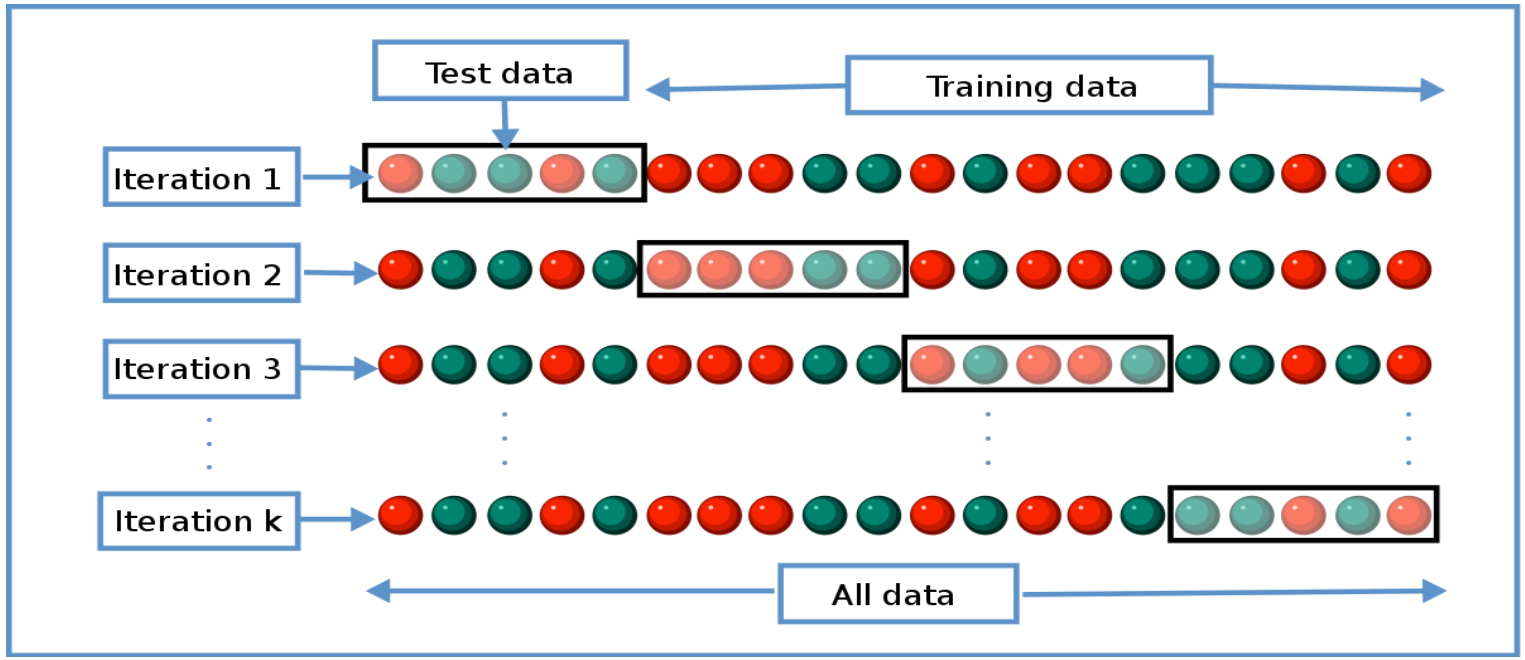
4) Réduire la complexité du modèle

(modèle plus simple, moins de paramètres, Réduire le nombre de couches dans les RNA...)

Comment éviter le surapprentissage

5) La validation croisée(Cross-Validation)

Diviser le dataset en **k** échantillon("folds"), puis entraîner et tester le modèle plusieurs fois, chaque fois sur un échantillon différent. Cela permet d'éviter que le modèle mémorise des détails spécifiques (bruit) des données d'entraînement, et de mieux estimer la performance du modèle



Comment éviter le surapprentissage

6) La régularisation

- Consiste à jouter un terme supplémentaire (une pénalité) à la fonction de perte pour pénalisent les coefficients trop élevés ce qui réduit la complexité du modèle et améliorer sa généralisation.

- Plutôt que de minimiser uniquement l'erreur entre les prédictions du modèle et les vraies valeurs (comme c'est le cas avec une simple fonction de perte), la régularisation ajoute un terme supplémentaire à cette fonction de perte.
- Ce terme pénalise les paramètres du modèle (les poids) s'ils deviennent trop grands ou trop complexes.

La régularisation

Les deux types de régularisation sont :

Régularisation L1 (Lasso)
pénalise la **somme des valeurs absolues** des coefficients du modèle, ce qui peut réduire certains coefficients à zéro, permettant ainsi de sélectionner les caractéristiques importantes.

$$\mathcal{L}_{\text{Lasso}} = \mathcal{L}_{\text{original}} + \lambda \sum_{j=1}^{m} |w_j|$$

Fonction de perte

pénalité

Régularisation L2 (Ridge)
pénalise la somme des carrés des coefficients, incitant le modèle à réduire leur taille.

$$\mathcal{L}_{\text{Ridge}} = \mathcal{L}_{\text{original}} + \lambda \sum_{j=1}^{m} w_j^2$$

Donc les deux types L1 et L2 pénalisent les coefficients trop élevés du modèle.

Sous-apprentissage (Underfitting)

- Le sous-apprentissage se produit lorsque le modèle est trop simple pour capter la structure sous-jacente des données (la relation entre les caractéristiques et les labels).
- Cela signifie que le modèle ne parvient pas à apprendre suffisamment à partir des données, ce qui entraîne des erreurs élevées aussi bien sur les données d'entraînement que sur les données de test.

Comment détecter le sous-apprentissage ?

- Erreur élevée sur les données d'entraînement et de test
- Performance similaire sur les données d'entraînement et de test
- Graphiques de l'erreur d'entraînement et de test en fonction des itérations ou de la complexité du modèle :
  - Si **les deux erreurs** (d'entraînement et de test) sont élevées et ne diminuent pas de manière significative avec l'augmentation de la complexité du modèle
  - une courbe de **perte plate** sur les deux ensembles, suggérant qu'il n'a pas appris les patterns dans les données.

Causes du sous-apprentissage

✓ Modèle trop simple

Par exemple, utiliser un modèle linéaire pour des données complexes ou non linéaires.

✓ Insuffisance de données d'entraînement

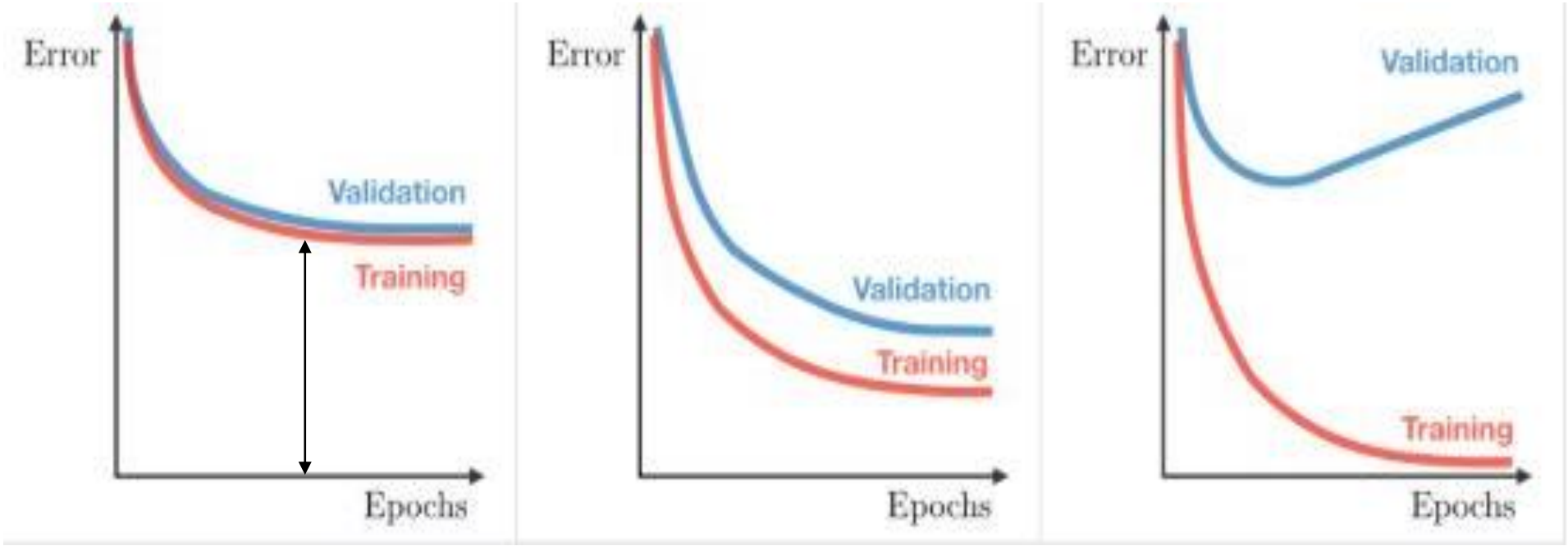
✓ Hyperparamètres mal réglés

Par exemple, un paramètre de régularisation trop élevé qui pénalise excessivement les coefficients et empêche le modèle d'apprendre.

✓Insuffisance d'itérations d'apprentissage

Si le modèle n'a pas suffisamment de temps pour apprendre les patterns des données.

Underfiting vs. Good fit vs. Overfitting



Comment résoudre le sous-apprentissage ?

- Utiliser un modèle plus complexe

Choisir un modèle plus complexe qui peut capturer des relations non linéaires ou des interactions plus complexes (par exemple, passer d'un modèle linéaire à un réseau de neurones ou un arbre de décision plus profond).

- Augmentation des données

Ajouter plus de données d'entraînement si possible, afin de permettre au modèle de mieux apprendre.

- Ajuster les hyperparamètres

Réduire la régularisation ou ajuster d'autres hyperparamètres pour permettre au modèle de mieux s'adapter aux données.

- Améliorer la qualité des données

Nettoyer les données ou effectuer des transformations pour mieux capturer les relations entre les variables.

Travaux pratiques

TP: Gestion des Problèmes Courants en Classification : Déséquilibre des Classes, Surapprentissage et Sous-apprentissage

- Étapes du TP:

1. Téléchargez un jeu de données **déséquilibré**, par exemple PIDD disponible sur : *https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database*

2. Examinez le problème du déséquilibre des classes : Analysez la distribution des classes.

3. Implémentez des stratégies de rééchantillonnage (sous-échantillonnage et sur-échantillonnage) pour équilibrer les classes

4. Analysez les performances du modèle de classification avant et après rééchantillonnage, en utilisant des métriques appropriées déjà vu au cours

5. détection de l'overfitting

- Entraînez un modèle simple de classification (par exemple, régression logistique ou un arbre de décision).

6. Analysez l'overfitting en traçant les courbes de **perte d'entraînement** et **perte de validation** au cours des itérations d'entraînement.

TP: Gestion des Problèmes Courants en Classification : Déséquilibre des Classes, Surapprentissage et Sous-apprentissage

7. Réduisez le surapprentissage

8. Sous-apprentissage et ajustement du modèle

- Vérifiez si le modèle souffre de sous-apprentissage et Résolvez le si ‘il existe

9. Évaluez les résultats obtenus avec les différents modèles et techniques (rééchantillonnage, régularisation, etc.).

THANK YOU