Université de Montpellier

HAX907X - Apprentissage statistique

Apprentissage de classes déséquilibrées

Étudiants:
EL SAWADOGO Kader
GERMAIN Marine
LABOURAIL Célia
MARIAC Damien

Encadrant :
BENSAID Bilel

8 octobre 2025

Table des matières

1 Contexte et Objectifs

Dans le cadre de ce travail, nous nous intéressons à l'apprentissage supervisé dans des contextes où les classes sont déséquilibrées. Ce problème est fréquent dans de nombreux domaines, tels que la détection de fraudes, la médecine ou le diagnostic industriel. Un déséquilibre important des classes peut conduire à des modèles biaisés : ils tendent à prédire correctement les classes majoritaires en négligeant les classes minoritaires.

La problématique scientifique que nous étudions est donc la suivante : "Comment atténuer le déséquilibre de classes pour améliorer la performance des modèles ?"

L'objectif de ce travail est d'étudier des méthodes permettant de résoudre ce déséquilibre. Plus précisément, nous nous concentrons sur trois approches issues d'articles scientifiques :

1. Le **Random Over-Sampling** (ROS) cité dans l'article *Survey on Deep Learning with Class Imbalance* [3], une méthode de rééchantillonage qui consiste à augmenter artificiellement la proportion de la classe minoritaire.
2. Le **Random Under-Sampling** (RUS), une autre méthode de rééchantillonage, cité également dans l'article [3] qui consiste à réduire la proportion de la classe majoritaire
3. La **Synthetic Minority Over-sampling Technique** (SMOTE) présenté dans l'article [1] qui génère de nouvelles instances synthétiques pour la classe minoritaire à partir des observations existantes

2 Les différentes méthodes

Les méthodes data-level agissent directement sur les données d'entraînement pour atténuer le déséquilibre entre classes. Le principe est soit d'augmenter le poids de la classe minoritaire en lui fournissant plus d'exemples (réels ou synthétiques), soit au contraire de réduire le poids de la classe majoritaire en éliminant certains de ses exemples. L'objectif est d'obtenir une distribution de classes plus équilibrée, ce qui force l'algorithme d'apprentissage à prêter autant d'attention à la minorité qu'à la majorité. Ces techniques peuvent toutefois introduire de la variance (sur-apprentissage) ou du biais supplémentaire, il faut donc les appliquer judicieusement.

2.1 Sur-échantillonnage (Over-sampling)

Le sur-échantillonnage (oversampling) consiste à ajouter des copies ou des variantes des exemples de la classe minoritaire jusqu'à augmenter sa fréquence dans le jeu de données. Dans sa forme la plus simple, le sur-échantillonnage aléatoire (Random Over-Sampling, ROS) duplique aléatoirement des instances minoritaires existantes jusqu'à atteindre un équilibre désiré. Par exemple, si l'on dispose de 100 exemples minoritaires et 1000 majoritaires, le ROS peut répliquer les minoritaires (éventuellement plusieurs fois chacun) jusqu'à en obtenir 1000, rétablissant ainsi un ratio équilibré. On échantillonne avec remise parmi les indices de la classe minoritaire pour générer de nouvelles instances d'entraînement.

La méthode présente des inconvenients. En effet, la duplication augmente le risque de surapprentissage (overfitting), car le modèle apprend alors trop précisément les exemples déjà présents dans la

base de données, ce qui nuit à sa capacité à généraliser sur de nouvelles données. De plus, la duplication de données minoritaire augmenterait significativement la taille des données ce qui implique un coût de calcule plus important.

2.2 Sous-échantillonnage (Under-sampling)

À l'inverse, le sous-échantillonnage (undersampling) vise à réduire la proportion de la classe majoritaire en retirant certains de ses exemples du jeu de données. Le sous-échantillonnage aléatoire (Random Under-Sampling, RUS) élimine au hasard des instances de la classe majoritaire jusqu'à atteindre un ratio plus équilibré avec la minorité.le RUS est utile lorsque l'on dispose d'une grande abondance de données majoritaires, potentiellement redondantes. Plutôt que de tout utiliser, ce qui peut être coûteux et inutile, on peut se permettre d'en élaguer une partie. En réduisant drastiquement le nombre d'exemples majoritaires, on élimine le biais numérique et on accélère l'entraînement (moins de données à parcourir).

Selon différents articles comme Bee Wah Yap et al. (2013) [2], il est souvent préférable d'utiliser le Random Undersampling plutôt que le Random Oversampling, à condition que la taille du jeu de données le permette.

Les deux méthodes de rééchantillonnage (ROS et RUS) présentent aussi certaines limites. Elles peuvent notam- ment conduire le modèle à croire que certaines combinaisons de variables expliquent à elles seules la majorité des observations minoritaires, ce qui peut fausser la compréhension réelle des relations entre variables.

2.3 SMOTE (Synthetic Minority Over-sampling Technique)

Plutôt que de copier des instances existantes, SMOTE crée de nouvelles instances minoritaires artificielles en interpolant entre des exemples réels. Concrètement, pour chaque exemple minoritaire original, SMOTE sélectionne aléatoirement l'un de ses k plus proches voisins (minoritaire également), puis génère un nouvel exemple situé aléatoirement le long du segment joignant les deux points dans l'espace des *features*. En répétant ce procédé, on peut synthétiser autant d'exemples minoritaires que souhaité.

Malgré ses avantages, SMOTE présente plusieurs inconvénients importants. Tout d'abord, son temps de calcul est plus long que celui des méthodes plus simples. Cela s'explique par le fait que l'algorithme doit calculer la distance entre chaque observation minoritaire et tous les autres points pour identifier ses k plus proches voisins. Ce calcul devient rapidement coûteux lorsque la taille du jeu de données augmente, car le nombre de comparaisons croît de manière quadratique (de l'ordre de $O(n^2)$). En pratique, cela signifie que sur des bases volumineuses ou avec beaucoup de variables, SMOTE peut nécessiter des temps de traitement bien plus élevés, surtout lorsqu'il est utilisé à chaque étape d'une validation croisée. Un autre problème souvent mentionné dans la littérature concerne la sensibilité de SMOTE aux points aberrants. Si une observation minoritaire isolée se trouve très proche d'un point appartenant à la classe majoritaire, l'algorithme risque d'utiliser cette proximité pour créer de nouvelles données synthétiques incorrectes. Cela peut introduire du bruit et perturber l'apprentissage du modèle. Pour limiter ce problème, il est possible d'ignorer les observations minoritaires dont les k plus proches voisins appartiennent principalement à la classe

majoritaire, afin d'éviter de générer des points non représentatifs.

Plusieurs variantes de SMOTE ont été développées pour corriger ces défauts :

- **SMOTE-NC (Nominal and Continuous)** : s'adapte aux jeux de données contenant à lafois des variables continues et qualitatives ;
- **Borderline-SMOTE** (Hui Han, Wen-Yuan Wang et Bing-Huan Mao, 2005) : se concentre sur les points situés à la frontière entre classes, considérés comme plus informatifs pour le modèle.

3 Conclusion

4 Annexe

4.1 Illustration de SMOTE

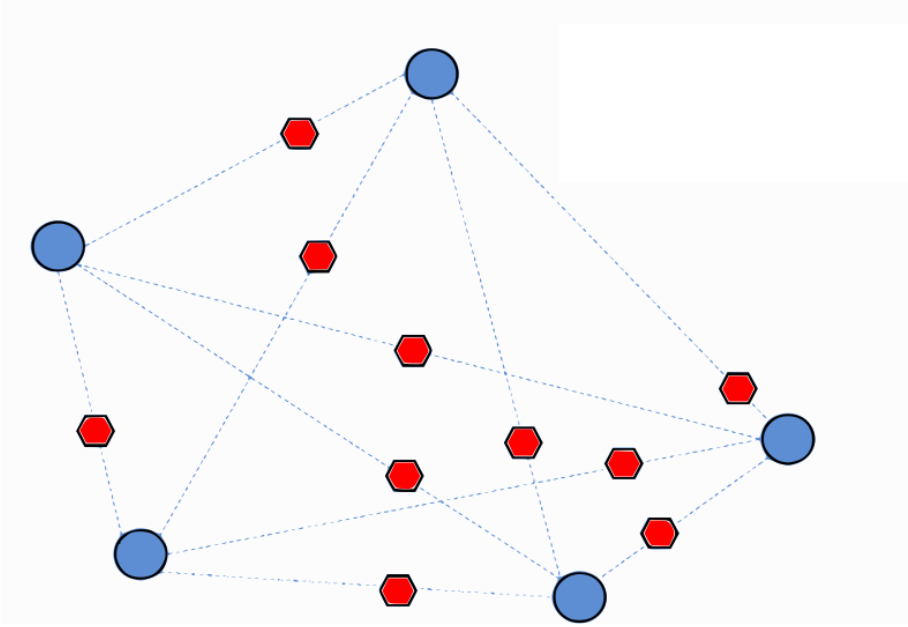

FIGURE 1 – Smote illustration

Les points bleu correspondent à la classe minoritaire dont on souhaite générer de nouveaux exemples, les points rouge sont les points générés par SMOTE.

4.2 Implémentation et résultats

Exemple d'application : Détection de fraudes par carte bancaire

L'objectif de cette application est de mettre en place un modèle de détection de fraudes à partir du jeu de données *Credit Card Fraud Detection* disponible sur *Kaggle*. Ce jeu contient 284 807 transactions, dont seulement 492 sont frauduleuses (soit environ 0,17 %). Le fort déséquilibre entre les classes rend la tâche difficile, car un modèle naïf qui prédirait toujours "non fraude" atteindrait déjà un score proche de 99,8 %. Pour dépasser cette limite, plusieurs stratégies de gestion du déséquilibre ont été appliquées et comparées en utilisant une régression logistique.

Régression logistique sans traitement

Dans un premier temps, nous avons entraîné le modèle sur l'ensemble complet, sans traitement particulier du déséquilibre. Les résultats montrent une accuracy très élevée (0,9991), mais trompeuse, puisque le rappel (sensitivity) des fraudes n'est que de 0,61. Autrement dit, la majorité des fraudes ne sont pas détectées, ce qui rend le modèle inutilisable en pratique malgré une performance apparente.

Random undersampling

Afin de corriger ce problème, nous avons testé le *random undersampling*, qui consiste à équilibrer les classes en réduisant volontairement le nombre de transactions normales pour égaler celui des fraudes. Dans ce cas, le modèle obtient une *balanced accuracy* d'environ 0,956 avec un rappel de 0,959. Le modèle apprend donc à bien reconnaître les fraudes, mais au prix d'une perte importante d'information, puisque la majorité des transactions légitimes est écartée.

Random oversampling

La seconde approche est le *random oversampling*, qui duplique aléatoirement les fraudes afin de créer un jeu équilibré. Cette méthode conserve toutes les données disponibles et améliore le rappel (0,977), ce qui signifie que presque toutes les fraudes sont détectées. La *balanced accuracy* atteint 0,949. Toutefois, l'inconvénient de cette méthode est le risque de surapprentissage, car les exemples de fraude sont répétés artificiellement sans ajouter de diversité.

SMOTE

Enfin, nous avons appliqué la méthode *SMOTE (Synthetic Minority Oversampling Technique)*, qui génère artificiellement de nouveaux exemples de fraude en interpolant entre des observations existantes. Cette méthode dépasse les performances des précédentes, avec une accuracy globale de 0,981, un rappel de 0,992 et une spécificité de 0,970. Elle offre donc le meilleur compromis entre la détection des fraudes et la limitation des faux positifs.

4.3 Algorithmes

Algorithm 1: Random Undersampling

Input: Dataset D avec classes C_{maj} et C_{min}
Output: Dataset équilibré D'
$n_{min} \leftarrow |C_{min}|$;
Sélectionner aléatoirement n_{min} échantillons de C_{maj};
Construire $D' = C_{min} \cup$ sous-échantillon(C_{maj});
return D';

Algorithm 2: Random Oversampling

Input: Dataset D avec classes C_{maj} et C_{min}
Output: Dataset équilibré D'
$n_{maj} \leftarrow |C_{maj}|$;
while $|C_{min}| < n_{maj}$ **do**
    Dupliquer aléatoirement un échantillon de C_{min};
Construire $D' = C_{maj} \cup C_{min}$ (après duplication);
return D';

Algorithm 3: Borderline-SMOTE (Synthetic Minority Oversampling Technique)

Input: Dataset D avec classes C_{maj} et C_{min}, nombre de voisins k
Output: Dataset équilibré D'
foreach $x_i \in C_{min}$ **do**
    Trouver les k plus proches voisins de x_i dans D;
    if *tous les voisins appartiennent à* C_{maj} **then**
        Marquer x_i comme **bruit** (non utilisé);
    else if *la majorité des voisins appartiennent à* C_{maj} **then**
        Marquer x_i comme **danger**;
    else
        Marquer x_i comme **safe** (non utilisé);
foreach x_i *marqué comme* ***danger*** **do**
    Sélectionner aléatoirement un voisin minoritaire x_{voisin} parmi ses k plus proches voisins;
    Générer un nombre aléatoire $\lambda \sim U(0,1)$;
    Créer un point synthétique :

$$x_{\text{new}} = x_i + \lambda \cdot (x_{voisin} - x_i)$$

    Ajouter x_{new} à C_{min};
Construire $D' = C_{maj} \cup C_{min}$ (avec points synthétiques générés);
return D';

Étape finale commune

Algorithm 4: Application de la régression logistique et évaluation

Input: Dataset équilibré D' obtenu par Full Data, Undersampling, Oversampling ou SMOTE

Output: Performance du modèle

Diviser D' en données d'apprentissage (train) et de test (test);

Standardiser les variables explicatives;

Entraîner une **régression logistique** sur l'échantillon d'apprentissage;

Prédire les probabilités sur l'échantillon test;

Tracer la **courbe ROC** et calculer l'**AUC**;

Construire la **matrice de confusion** pour évaluer les performances;

return *ROC, AUC, matrice de confusion*;

Références

[1] Nitesh V Chawla et al. “SMOTE : synthetic minority over-sampling technique”. In : *Journal of artificial intelligence research* 16 (2002), p. 321-357.

[2] Merwan Chelouah. “Méthodes de rééquilibrage des classes en classification supervisée”. In : ().

[3] Justin M Johnson et Taghi M Khoshgoftaar. “Survey on deep learning with class imbalance”. In : *Journal of big data* 6.1 (2019), p. 1-54.