Rapport : Détection d'anomalies avec Python

Miryam Atamna – M2 DS

Novembre 2025

1 Introduction et Présentation du TP

La détection d'anomalies, également appelée détection d'outliers ou détection de nouveautés, est une tâche essentielle de l'apprentissage automatique visant à identifier les observations dont le comportement diffère significativement de la majorité des données. Ces anomalies peuvent correspondre à des erreurs de mesure, des défaillances techniques, ou encore à des comportements anormaux tels que des fraudes bancaires ou des intrusions dans un réseau informatique. L'intérêt de cette tâche réside dans le fait que, dans de nombreux contextes réels, les anomalies sont rares mais critiques. Leur identification rapide permet d'anticiper ou de corriger des situations problématiques. Par exemple, dans le domaine financier, une dépense atypique par carte bancaire peut révéler une fraude ; dans les réseaux informatiques, une activité inhabituelle peut indiquer une tentative d'intrusion ; et dans l'industrie, une mesure de capteur incohérente peut signaler une défaillance imminente d'un équipement.

Le TP a donc les objectifs suivants :

1. Expérimenter différentes méthodes de détection d'anomalies
2. Comparer leurs performances en fonction du contexte et du type de données
3. Proposer une méthodologie factorisée, modulaire et reproductible, permettant de généraliser la détection d'anomalies à divers domaines.

Ainsi, ce rapport présente successivement les expérimentations menées sur trois jeux de données de complexité croissante : le jeu Mouse pour l'exploration visuelle, le jeu Credit Card Fraud pour la détection de fraudes bancaires dans un contexte déséquilibré, et le jeu KDDCup99 pour la détection d'intrusions réseau. Chaque partie détaille la préparation des données, l'application des modèles, les visualisations et l'interprétation des résultats.

2 Partie I — Jeu de données Mouse

2.1 Présentation du dataset

Pour débuter ce travail, nous avons utilisé le jeu de données *Mouse*, composé de 500 observations bidimensionnelles décrites par deux variables continues : `x1` et `x2`. Ce jeu est couramment utilisé pour illustrer les algorithmes de détection d'anomalies car il présente une structure bien définie et quelques points aberrants.

Jeu de données chargé avec succès

Chemin : `../data/mouse.txt`

Dimensions : 500 lignes × 2 colonnes

Colonnes : x1, x2

	x1	x2
0	0.456601	0.432806
1	0.611378	0.528625
2	0.450299	0.711606
3	0.639015	0.460744
4	0.628957	0.323470

Figure 1 : Informations générales du dataset

2.2 Analyse exploratoire des données

Résumé rapide du dataset

	Nombre de lignes	Nombre de colonnes	Noms des colonnes	Valeurs manquantes	Types
0	500	2	x1, x2	0	float64, float64

Statistiques descriptives

	count	mean	std	min	25%	50%	75%	max
x1	500.000000	0.505892	0.191763	0.040555	0.328481	0.504702	0.681430	0.916030
x2	500.000000	0.599706	0.156863	0.138940	0.469010	0.629494	0.735364	0.897028

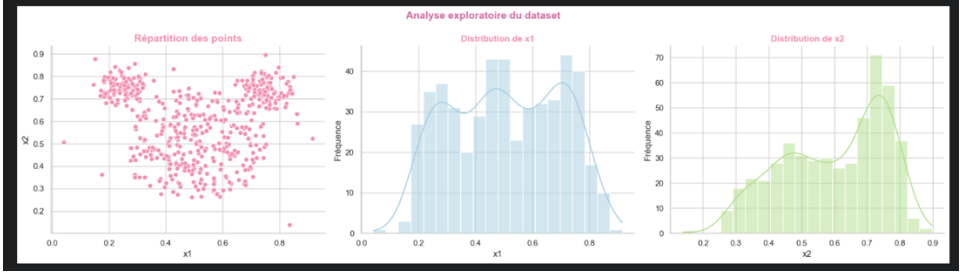


Figure 2 : Analyse exploratoire

Les statistiques descriptives obtenues confirment que les deux variables `x1` et `x2` sont numériques continues, normalisées dans l'intervalle [0, 1] et exemptes de valeurs manquantes. La répartition des points met en évidence une structure interne claire, composée de trois amas distincts formant la silhouette d'une tête de Mickey : deux zones supérieures (représentant les oreilles) et une zone centrale plus dense (représentant le visage).
Les dix dernières instances du fichier sont décrites dans l'énoncé comme des outliers, probablement situées en périphérie des amas principaux.

En résumé :

- Les données sont propres et bien normalisées
- La structure interne du jeu est clairement visible (trois clusters distincts)
- Un petit nombre d'observations atypiques est présent, constituant des anomalies à identifier.

Ainsi, le jeu de données *Mouse* constitue une base d'expérimentation particulièrement adaptée à l'évaluation d'approches non supervisées et supervisées de détection d'anomalies.

2.3 Application des méthodes non supervisées

Dans un premier temps, deux approches non supervisées ont été appliquées sur le jeu de données *Mouse* :

- Isolation Forest, fondée sur le principe d'isolation aléatoire des points atypiques ;
- Local Outlier Factor (LOF), basée sur la densité locale des observations.

Ces modèles ont été entraînés avec un paramètre de contamination fixé à 0.02, correspondant à environ dix observations anormales attendues. L'objectif de cette première étape est d'obtenir une détection visuelle des anomalies et d'analyser la distribution des scores produits par chaque méthode.

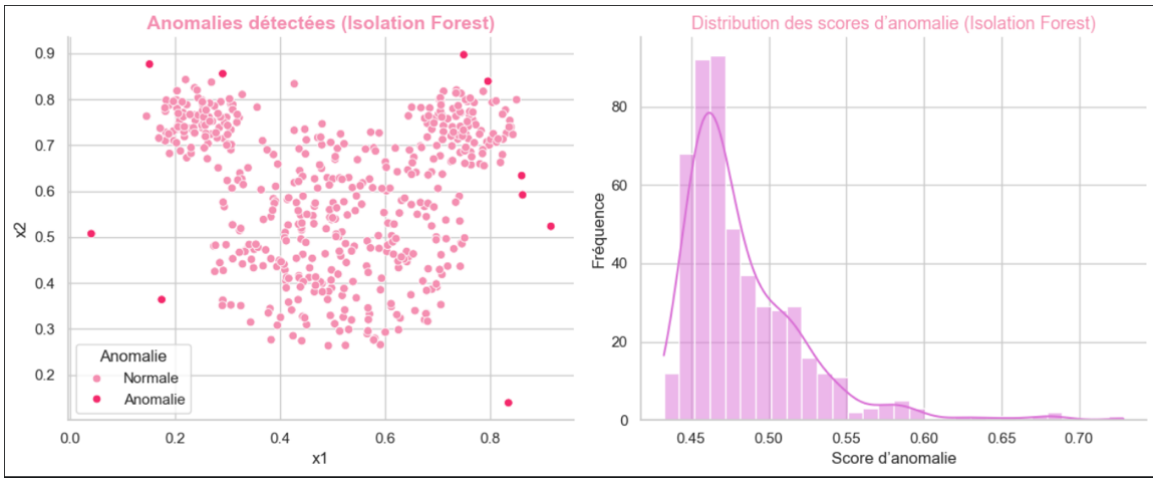


Figure 3 - Anomalies détectées par Isolation Forest et distribution des scores d'anomalie

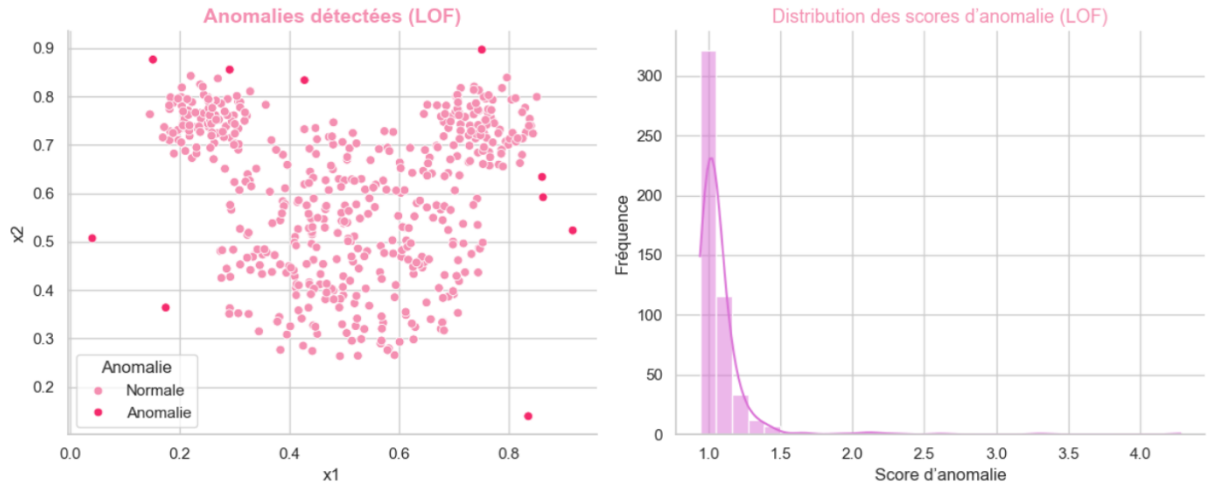


Figure 4 — Anomalies détectées par LOF et distribution des scores d'anomalie

Les figures ci-dessus montrent les résultats obtenus pour les deux modèles.

Dans le cas d'Isolation Forest, les points identifiés comme anormaux se situent principalement à la périphérie de la structure, notamment autour du contour inférieur du "visage". La distribution des scores d'anomalie présente une forme unimodale, avec une majorité de valeurs proches de la normale et une queue droite correspondant aux observations isolées. Le modèle LOF aboutit à un résultat visuellement similaire, mais avec une sensibilité locale plus élevée. Il repère également des points situés en bordure des amas principaux (zones de faible densité), ce qui correspond à la logique de l'algorithme.

En résumé les deux méthodes parviennent à identifier les mêmes zones anormales. Isolation Forest met en évidence les anomalies de manière globale, tandis que LOF offre une lecture plus fine à l'échelle locale. À ce stade, aucune méthode de calcul de seuil n'a encore été appliquée : la détection repose uniquement sur la proportion fixée (`contamination = 0.02`).

2.4 Détermination du seuil d'anomalie

Afin d'affiner la séparation entre observations normales et anomalies, les deux méthodes ont été utilisées pour estimer un seuil de décision à partir des scores d'anomalie générés par les modèles :

- La méthode de l'écart interquartile (IQR), fondée sur les quartiles de la distribution,
- Et le clustering K-Means à deux groupes, qui distingue automatiquement les points normaux des points anormaux selon leurs scores.

Les distributions et seuils obtenus pour chaque méthode sont comparés dans la figure suivante.

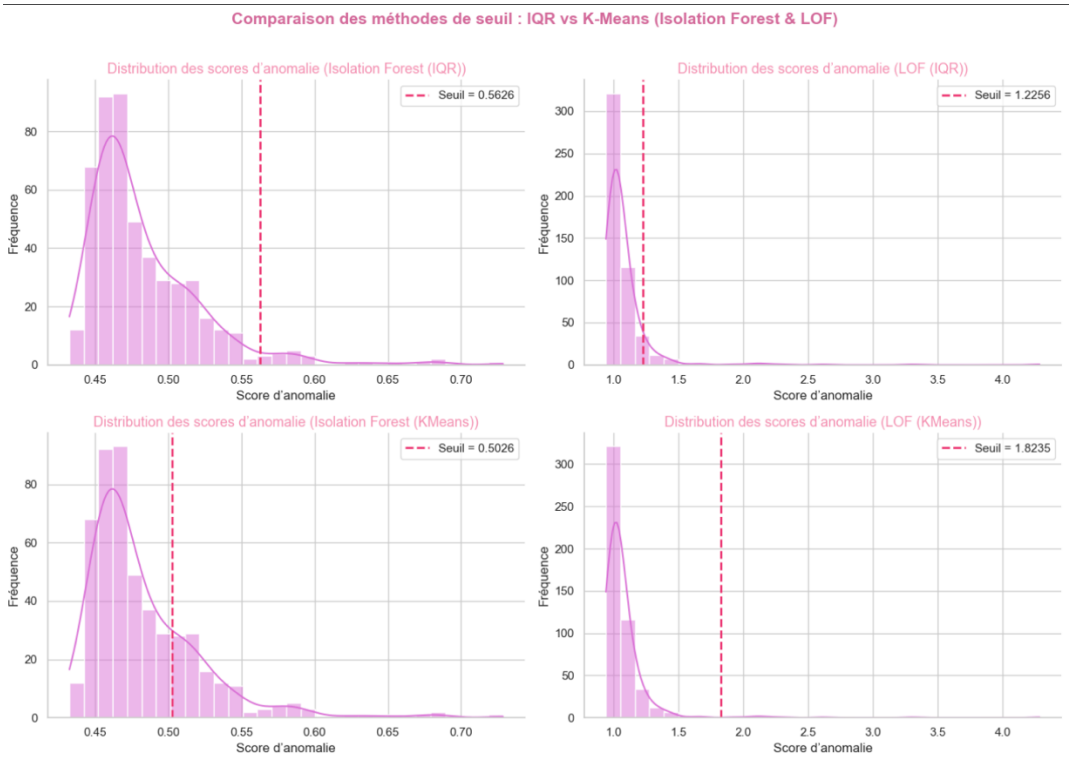


Figure 6 - Comparaison des méthodes de seuil : IQR vs K-Means (Isolation Forest & LOF)

Sur la figure ci-dessus, chaque histogramme illustre la fréquence des scores d'anomalie, tandis que les lignes pointillées rouges indiquent le seuil retenu par la méthode correspondante. Les seuils obtenus sont les suivants :

Modèle	Méthode IQR	Méthode K-Means
Isolation Forest	0.5626	0.5026
LOF	1.2256	1.8235

D'un point de vue numérique, un seuil d'anomalie est satisfaisant

- Lorsqu'il se situe juste au-delà de la majorité des scores normaux,
- Sans atteindre la queue extrême de la distribution,
- La différence entre IQR et K-Means reste cohérente avec la forme de la distribution.

Pour Isolation Forest, les seuils IQR et K-Means sont proches, traduisant une bonne cohérence numérique et une séparation nette entre valeurs normales et aberrantes.

À l'inverse, pour LOF, l'écart entre les deux méthodes est plus marqué, ce qui reflète une plus grande variabilité des scores et la complexité de la distribution liée à la densité locale. Ainsi, le seuil IQR correspond mieux au comportement attendu pour Isolation Forest, tandis que K-Means s'ajuste davantage à la dispersion observée chez LOF.

Sur le plan graphique, un bon seuil est celui

- qui coupe la courbe au point où la fréquence des scores chute brusquement,
- marquant la frontière naturelle entre les observations normales et les anomalies.

Visuellement, pour Isolation Forest, le seuil IQR (≈0.56) se place idéalement à la sortie du pic principal, séparant clairement les valeurs normales des extrêmes. Le seuil K-Means (≈0.50), légèrement plus bas, tend à couper trop tôt.

Pour LOF, la tendance est inversée : le seuil IQR (≈1.22) reste trop proche du pic et détecte trop d'anomalies, alors que le seuil K-Means (≈1.82) se situe plus loin dans la queue de la distribution, isolant plus efficacement les véritables outliers.

En combinant ces deux perspectives, on peut conclure que la méthode IQR est la plus adaptée à Isolation Forest, car elle fournit un seuil stable et cohérent, tandis que la méthode K-Means est mieux pour LOF, grâce à sa capacité à s'ajuster à la variabilité locale.

2.5 Détection de nouveautés avec LOF

Une dernière expérimentation a été menée à l'aide du modèle Local Outlier Factor (LOF) configuré en mode `novelty=True`. Contrairement au mode classique, cette approche n'identifie pas les anomalies présentes dans le jeu d'apprentissage, mais apprend à partir de données considérées comme normales afin de détecter ensuite les comportements nouveaux ou inattendus. Le modèle a été entraîné sur 98 % de données normales, soit 490 points, puis testé sur l'ensemble complet de 500 observations. Le paramètre `n_neighbors` a été fixé à 20.

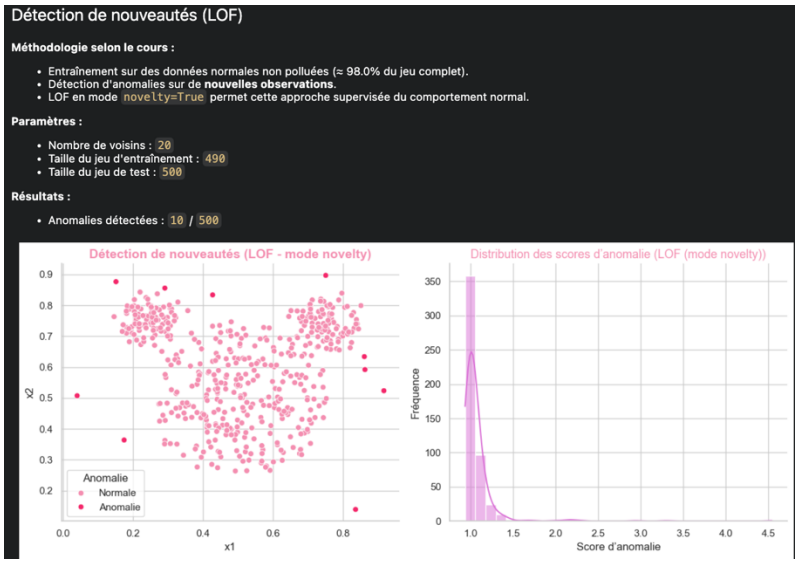


Figure 7 — Détection de nouveautés avec LOF (mode novelty)

Le graphique montre que la répartition des points détectés reste visuellement proche de celle observée avec le LOF classique, mais la philosophie du modèle diffère. Ici, le modèle apprend le comportement normal lors de l'entraînement, puis signale uniquement les nouvelles observations qui s'en écartent. Les résultats confirment la détection de dix anomalies, localisées principalement en périphérie des clusters au niveau des "oreilles" et du contour de la tête. La distribution des scores est plus concentrée, avec une petite queue à droite indiquant la présence de quelques points isolés correspondant aux nouveautés détectées.

2.6 Conclusion du dataset Mouse

En résumé, le dataset *Mouse* a permis de valider la logique des méthodes non supervisées avant de passer à un cas plus réaliste. Les comportements observés entre Isolation Forest et LOF ont montré des différences cohérentes, et la version *novelty* du LOF introduit une approche plus proche de la détection en continu, que nous approfondirons avec le jeu de données sur la fraude bancaire.

Remarque : Aucune métrique quantitative (telle que le F1-score, l'Average Precision ou la courbe Précision–Rappel) n'a été calculée sur ce premier jeu de données. En effet, le dataset *Mouse* a une vocation illustrative : il permet d'observer visuellement le comportement des modèles non supervisés, leur répartition des scores et la manière dont ils isolent les points aberrants. Les 10 outliers connus ne servent qu'à valider qualitativement la cohérence des résultats, sans constituer un ensemble de test étiqueté. L'évaluation quantitative à l'aide de métriques supervisées sera réalisée dans la suite du TP.

3 Partie II – Détection de fraudes bancaires

3.1 Présentation du dataset

Le jeu de données *Credit Card Fraud Detection* contient des transactions bancaires réelles et anonymisées issues d'un échantillon de paiements par carte effectués en septembre 2013 en Europe. Il totalise 284 807 transactions décrites par 31 variables, dont 28 (*V1–V28*) proviennent d'une réduction de dimension par Analyse en Composantes Principales (PCA). Ces composantes sont donc déjà centrées et réduites. La variable `Amount` correspond au montant de la transaction, et la variable `Class` indique si l'observation correspond à une transaction normale (0) ou une fraude (1).

Jeu de données chargé avec succès

Chemin : `../data/creditcard.csv`

Dimensions : 284807 lignes × 31 colonnes

Colonnes : Time, V1, V2, V3, V4, V5, V6, V7, V8, V9, V10, V11, V12, V13, V14, V15, V16, V17, V18, V19, V20, V21, V22, V23, V24, V25, V26, V27, V28, Amount, Class

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412
3	1.0	-0.966272	-0.185226	1.792993	-0.863291	-0.010309	1.247203	0.237609	0.377436	-1.387024	...	-0.108300	0.005274	-0.190321
4	2.0	-1.158233	0.877737	1.548718	0.403034	-0.407193	0.095921	0.592941	-0.270533	0.817739	...	-0.009431	0.798278	-0.137458

5 rows × 31 columns

Figure 8 - Chargement du jeu de données

3.2 Préparation et nettoyage des données

Avant l'entraînement des modèles, la variable `Time` a été supprimée car elle n'apporte pas d'information utile à la détection d'anomalies. Les données ont ensuite été séparées en deux ensembles :

- X_credit : les variables explicatives (V1–V28, Amount)
- y_credit : la variable cible (`Class`).

Les variables issues de la PCA étant déjà normalisées, seule la variable `Amount` a été standardisée via un StandardScaler afin d'éviter les écarts d'échelle susceptibles de fausser les calculs de distance ou les décisions des modèles.

3.3 Application des méthodes non supervisées

Dans un premier temps, nous avons appliqué deux méthodes non supervisées de détection d'anomalies sur le jeu de données : Isolation Forest (IF) et Local Outlier Factor (LOF). Ces approches permettent d'identifier des transactions atypiques sans utiliser la variable cible, et servent de point de référence avant les modèles supervisés.

3.3.1 Isolation Forest

Le modèle Isolation Forest repose sur la construction aléatoire d'arbres de partition : les observations isolées tôt dans le processus sont considérées comme des anomalies. Une recherche d'hyperparamètres a été menée sur plusieurs combinaisons du nombre d'estimateurs (`n_estimators`), de la proportion d'anomalies attendue (`contamination`) et de la taille d'échantillon (`max_samples`). Le meilleur modèle obtenu correspond à `n_estimators = 400`, `max_samples = 0.8`, `contamination = 0.002`.

Optimisation du modèle **Isolation Forest** sur le dataset **credit card**

- Nombre de combinaisons testées : **5**
- Meilleur F1 : **0.318**
- Meilleur AP : **0.230**

Résultats de l'optimisation - Isolation Forest (credit card)

	n_estimators	max_samples	contamination	F1	AP
3	400	0.800000	0.002000	0.318267	0.230322
4	300	0.600000	0.002000	0.306968	0.224134
2	300	0.800000	0.005000	0.291080	0.226673
1	200	auto	0.002000	0.282486	0.164056
0	100	auto	0.001000	0.270270	0.170371

Figure 9 - Optimisation du modèle Isolation Forest

L'évaluation du modèle sur le jeu de test montre une valeur de **F1-score de 0.317** et une **Average Precision (AP) de 0.228**, illustrant une capacité limitée mais réelle à repérer des comportements rares. La matrice de confusion révèle que la majorité des fraudes sont correctement détectées, mais au prix d'un certain nombre de faux positifs. La courbe précision–rappel montre une décroissance progressive de la précision lorsque le rappel augmente, typique des modèles non supervisés.

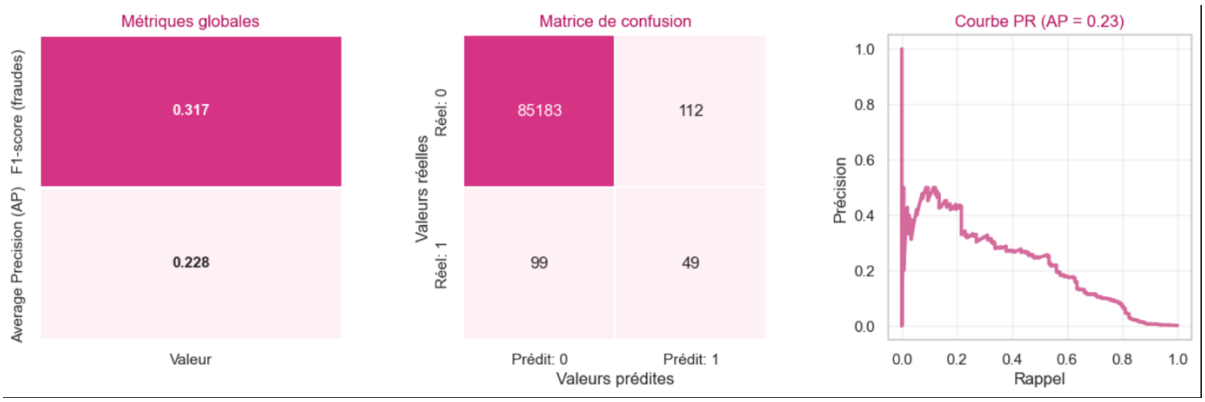


Figure 10 - Résultats de l'Isolation Forest

3.3.2 Local Outlier Factor

Le modèle Local Outlier Factor estime le degré d'isolement d'un point en comparant sa densité locale à celle de ses voisins. Il est particulièrement sensible aux zones de faible densité, ce qui permet d'identifier des fraudes "locales" dans des sous-régions du jeu de données. L'optimisation des paramètres a porté sur le nombre de voisins (`n_neighbors`) et la proportion d'anomalies (`contamination`). Le meilleur modèle a été obtenu pour : `n_neighbors = 500`, `contamination = 0.01`.

Optimisation du modèle **Local Outlier Factor** sur le dataset **credit card**

- Nombre de combinaisons testées : **3**
- Meilleur F1 : **0.240**
- Meilleur AP : **0.258**

Résultats de l'optimisation - Local Outlier Factor (credit card)

	n_neighbors	contamination	F1	AP
1	500	0.010000	0.240048	0.257974
2	700	0.020000	0.132170	0.436004
0	200	0.010000	0.029333	0.009500

Figure 11 - Optimisation du modèle Local Outlier Factor

L'évaluation finale donne un **F1-score de 0.240** et une **Average Precision (AP) de 0.258**. Bien que le score F1 soit inférieur à celui d'Isolation Forest, la courbe précision–rappel montre un léger gain en précision moyenne. En revanche, la matrice de confusion indique un nombre plus important de faux positifs : le modèle tend à confondre certaines transactions normales avec des fraudes à cause de sa forte sensibilité aux variations locales.

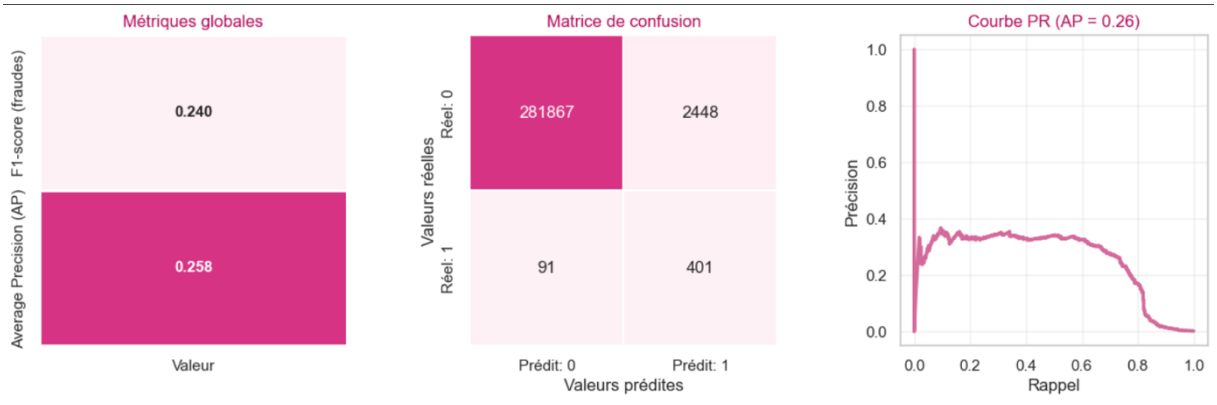


Figure 11 - Résultats du Local Outlier Factor

3.3.3 Interprétation globale

Les résultats obtenus montrent que les deux approches détectent effectivement une partie des fraudes, mais de manière incomplète. En résumé, ces deux méthodes non supervisées constituent une première étape utile pour la détection de comportements atypiques, mais leur efficacité reste limitée par l'absence de supervision et le fort déséquilibre du jeu de données. Elles serviront de référence de base avant l'application des méthodes supervisées sur données rééquilibrées.

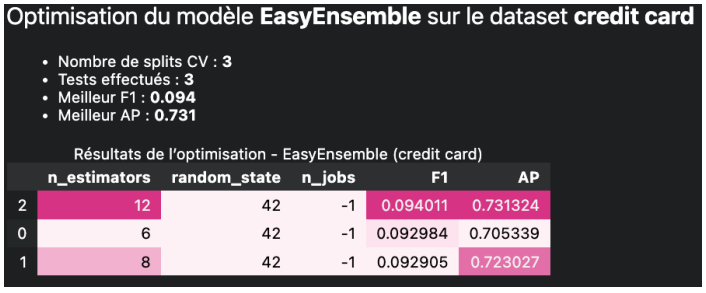
3.4 Introduction aux approches supervisées

Après avoir expérimenté des méthodes non supervisées, l'étape suivante consiste à exploiter les étiquettes réelles de la variable `Class` afin de construire des modèles supervisés. Ces approches apprennent à distinguer les transactions normales des transactions frauduleuses à partir d'exemples annotés, ce qui permet d'obtenir des performances généralement plus stables. Cependant, le jeu de données présente une forte inégalité de classes : 0,173 % des transactions sont frauduleuses. Ce déséquilibre complique l'entraînement et peut conduire les modèles à favoriser la classe majoritaire (transactions normales). Dans cette partie, nous verrons dans quelle mesure l'exploitation des labels et l'application de stratégies de rééquilibrage peuvent renforcer la capacité des modèles à identifier efficacement les transactions anormales.

3.5 Easy Ensemble Classifier

Le modèle EasyEnsembleClassifier repose sur un principe d'apprentissage par ensemble spécialement conçu pour les jeux de données fortement déséquilibrés. Au lieu de créer un seul modèle entraîné sur un jeu rééquilibré artificiellement, il construit plusieurs sous-classifieurs (par défaut des arbres de décision), chacun entraîné sur un sous-échantillon équilibré entre la classe majoritaire et la classe minoritaire. Les prédictions finales résultent de l'agrégation de ces modèles, ce qui améliore la robustesse et réduit la variance. Contrairement aux approches de rééchantillonnage explicite, EasyEnsemble intègre directement la gestion du déséquilibre dans son processus d'entraînement. Il s'agit donc d'une méthode supervisée capable de traiter efficacement le déséquilibre sans altérer la distribution d'origine.

Le modèle a donc été entraîné sur le jeu de données original, sans rééquilibrage préalable. Une recherche d'hyperparamètres a été effectuée sur le nombre d'ensembles (`n_estimators`) avec des valeurs comprises entre 6 et 12, à l'aide d'une validation croisée à 3 plis. L'objectif est d'évaluer la stabilité du modèle et d'identifier la configuration la plus performante en termes de F1-score et d'Average Precision (AP).


Optimisation du modèle **EasyEnsemble** sur le dataset **credit card**

- Nombre de splits CV : **3**
- Tests effectués : **3**
- Meilleur F1 : **0.094**
- Meilleur AP : **0.731**

Résultats de l'optimisation - EasyEnsemble (credit card)

	n_estimators	random_state	n_jobs	F1	AP
2	12	42	-1	0.094011	0.731324
0	6	42	-1	0.092984	0.705339
1	8	42	-1	0.092905	0.723027

Figure 11 - Résultats de l'optimisation du modèle EasyEnsemble

Le meilleur modèle obtenu correspond à `n_estimators = 8`, avec un **F1-score de 0.753** et une **Average Precision (AP) de 0.703** sur le jeu de test. Le seuil de décision optimal, déterminé automatiquement, est de 0.673, valeur à partir de laquelle le modèle sépare les transactions normales des fraudes détectées. La matrice de confusion montre que le modèle parvient à détecter la majorité des fraudes tout en maintenant un nombre limité de faux positifs, ce qui confirme sa bonne capacité de généralisation. La courbe précision–rappel illustre un équilibre stable entre sensibilité et précision, le modèle conservant une performance correcte même pour des seuils plus stricts.

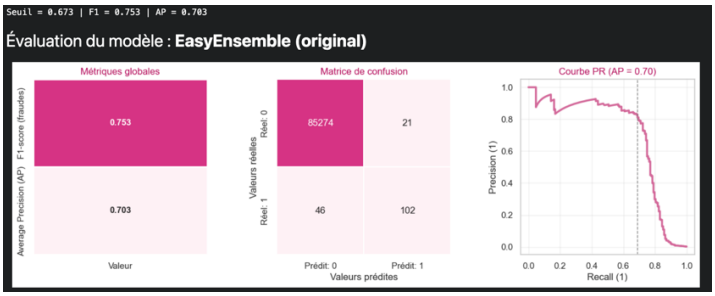


Figure 12 - Résultats du modèle EasyEnsemble

Les résultats obtenus démontrent qu'EasyEnsemble constitue une approche particulièrement adaptée aux contextes de déséquilibre extrême. En pratique, il atteint des performances solides sans rééchantillonnage externe, ce qui en fait une référence robuste pour la suite de l'étude, où seront évaluées d'autres approches supervisées combinées à différentes stratégies de rééquilibrage.

3.6 Modèles supervisées avec rééquilibrages des classes

Après l'évaluation du modèle Easy Ensemble, cette section explore d'autres approches supervisées afin d'évaluer l'impact du rééquilibrage des classes sur les performances de détection des fraudes. Trois modèles ont été sélectionnés pour leur complémentarité :

- XGBoost, modèle de boosting très performant sur les données tabulaires,
- Random Forest, approche ensembliste robuste aux valeurs extrêmes,
- Régression Logistique, modèle linéaire de référence, simple et interprétable.

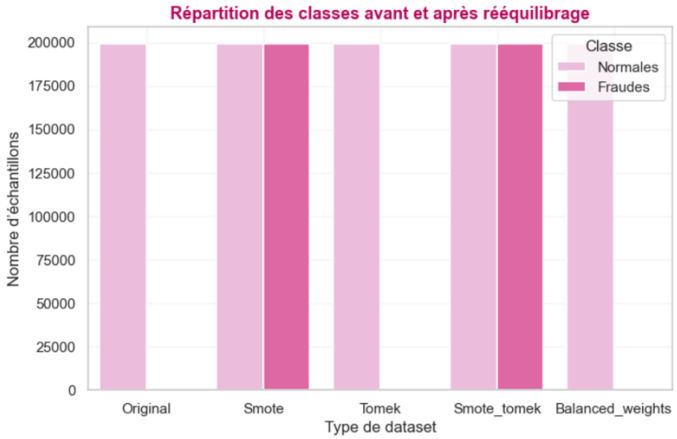
Ces modèles ont été appliqués sur plusieurs versions du dataset `creditcard` : l'original fortement déséquilibré, ainsi que ses versions rééquilibrées à l'aide de différentes stratégies d'échantillonnage.

3.6.1 Gestion du déséquilibre des classes

On rappelle que le jeu de données initial contient 0,173 % de transactions frauduleuses, contre 99,827 % de transactions normales, ce qui rend l'apprentissage difficile. Un modèle classique tend à prédire uniquement la classe majoritaire, entraînant une précision trompeusement élevée mais aucun rappel sur les fraudes. Pour remédier à ce problème, plusieurs techniques de rééquilibrage ont été testées :

Méthode	Type	Description	Effet attendu
Original	-	Jeu de données d'origine, très déséquilibré	Baseline de comparaison
SMOTE	Oversampling	Crée des fraudes synthétiques pour équilibrer les classes	Équilibre parfait 50/50
Tomek Links	Undersampling intelligent	Supprime les *normales ambiguës* proches des fraudes	Nettoyage des frontières
SMOTE + Tomek	Mixte	Combine SMOTE et Tomek pour équilibrer et nettoyer	Équilibre + stabilité
Balanced Weights	Pondération	Ajuste le poids des classes dans l'apprentissage (sans changer les données)	Influence l'entraînement, pas la distribution

Cette étape permet d'évaluer, dans la suite du TP, l'effet concret de chaque stratégie sur les modèles supervisés.



Ratios de classes avant/après rééquilibrage

	Dataset	Normales	Fraudes	Ratio (%)
0	Original	199,020	344	0.173
1	Smote	199,020	199,020	50.000
2	Tomek	198,997	344	0.173
3	Smote_tomek	199,020	199,020	50.000
4	Balanced_weights	199,020	344	0.173

Figure 13 – Répartition des classes avant et après rééquilibrage

3.6.2 Résultats des modèles supervisées

Chaque modèle a été optimisé par validation croisée à trois plis, en testant plusieurs configurations d'hyperparamètres afin de sélectionner celle offrant les meilleures performances. L'évaluation finale s'appuie toujours sur les deux métriques principales : le F1-score, mesurant le compromis entre précision et rappel, et l'Average Precision (AP), plus adaptée aux jeux de données fortement déséquilibrés.

Performances sur le dataset original :

- XGBoost obtient les meilleures performances globales (**F1 = 0.842**, **AP = 0.824**).
- Random Forest atteint **F1 = 0.803**, avec une bonne stabilité.
- Régression Logistique reste plus limitée (**F1 = 0.807**, **AP = 0.692**).

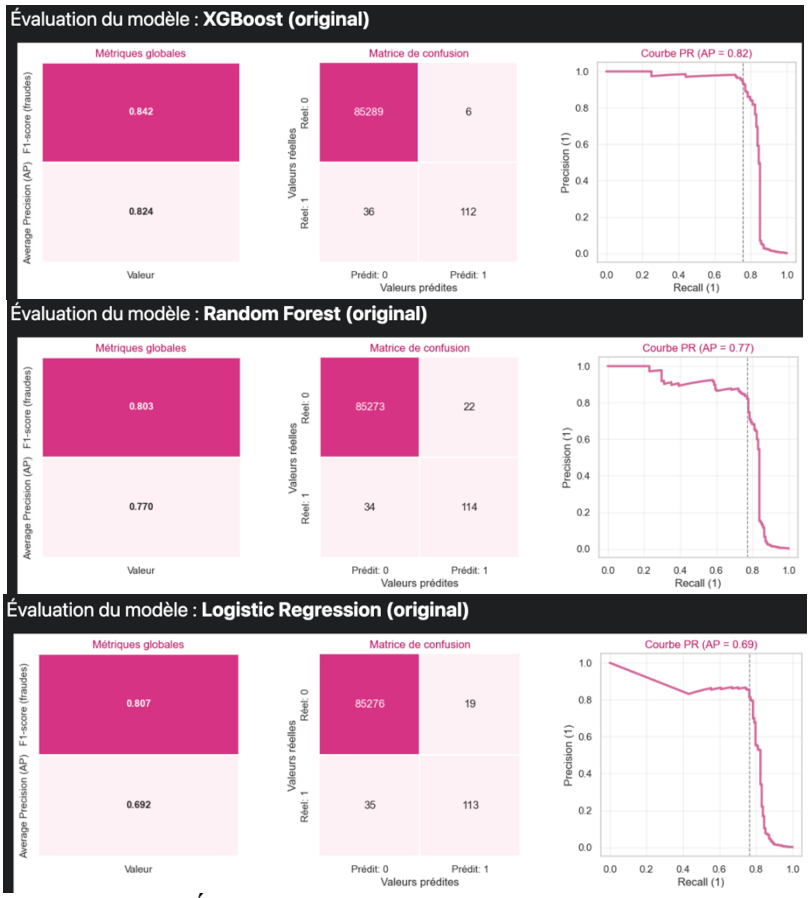
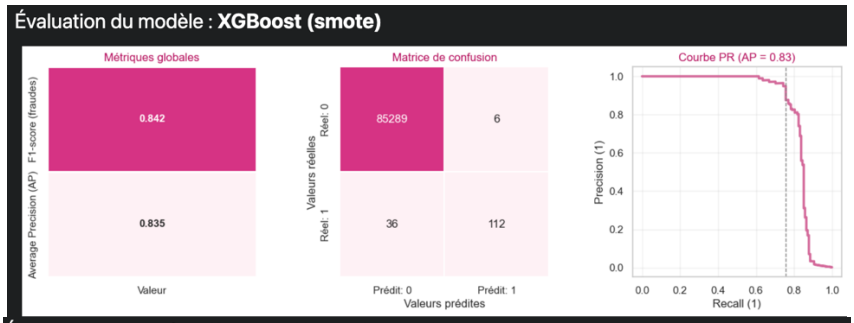
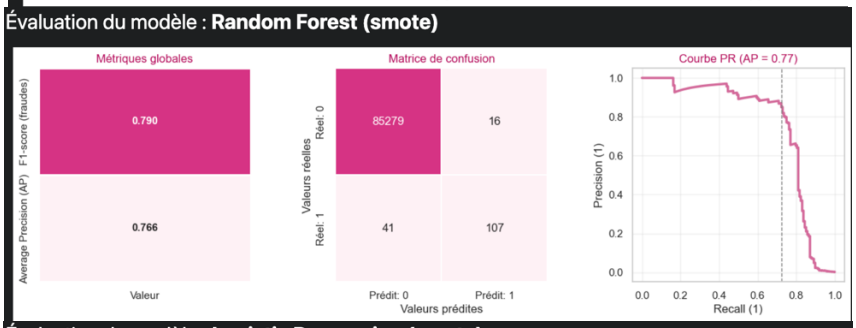


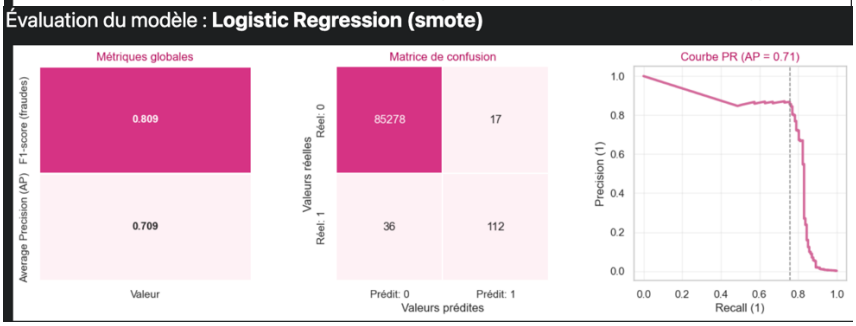
Figure 14 – Évaluation des modèles sur le dataset original

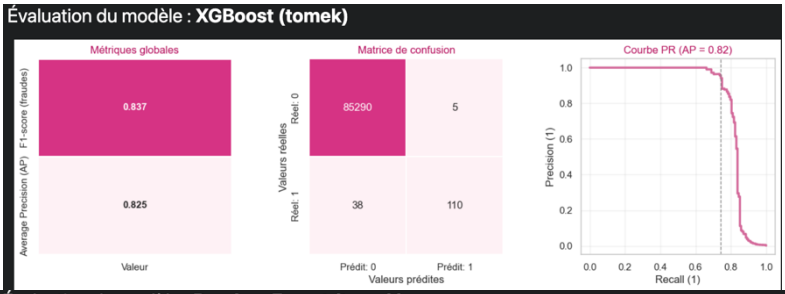
Les performances observées sur les jeux rééquilibrés mettent en évidence plusieurs tendances :

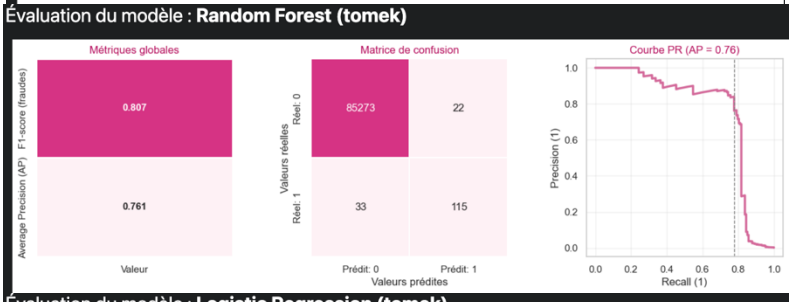
- SMOTE et SMOTE+Tomek offrent un gain significatif sur le rappel, améliorant la détection des fraudes sans perte majeure de précision.
- Tomek Links seul dégrade légèrement le F1-score (~0.70), en supprimant trop d'exemples proches de la frontière.
- Balanced Weights produit des résultats comparables à l'original, prouvant qu'un modèle robuste peut déjà gérer le déséquilibre via des pondérations internes.

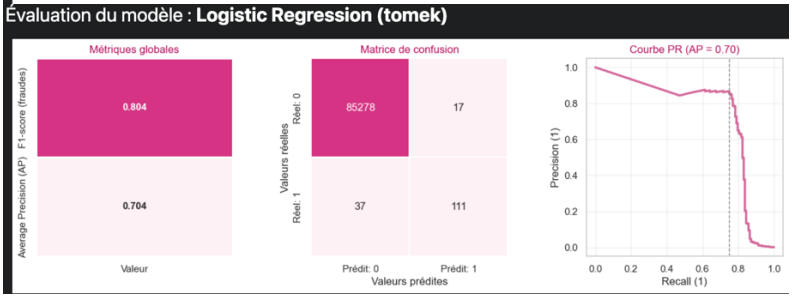

Évaluation du modèle : XGBoost (smote)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.83)
F1-score (fraudes)
0.842
Average Precision (AP)
0.835
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85289
6
36
112
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)


Évaluation du modèle : Random Forest (smote)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.77)
F1-score (fraudes)
0.790
Average Precision (AP)
0.766
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85279
16
41
107
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)


Évaluation du modèle : Logistic Regression (smote)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.71)
F1-score (fraudes)
0.809
Average Precision (AP)
0.709
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85278
17
36
112
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)


Évaluation du modèle : XGBoost (tomek)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.82)
F1-score (fraudes)
0.837
Average Precision (AP)
0.825
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85290
5
38
110
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)


Évaluation du modèle : Random Forest (tomek)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.76)
F1-score (fraudes)
0.807
Average Precision (AP)
0.761
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85273
22
33
115
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)


Évaluation du modèle : Logistic Regression (tomek)
Métriques globales
Matrice de confusion
Courbe PR (AP = 0.70)
F1-score (fraudes)
0.804
Average Precision (AP)
0.704
Valeur
Valeurs réelles
Réel: 0
Réel: 1
85278
17
37
111
Prédit: 0
Prédit: 1
Valeurs prédites
Precision (1)
Recall (1)

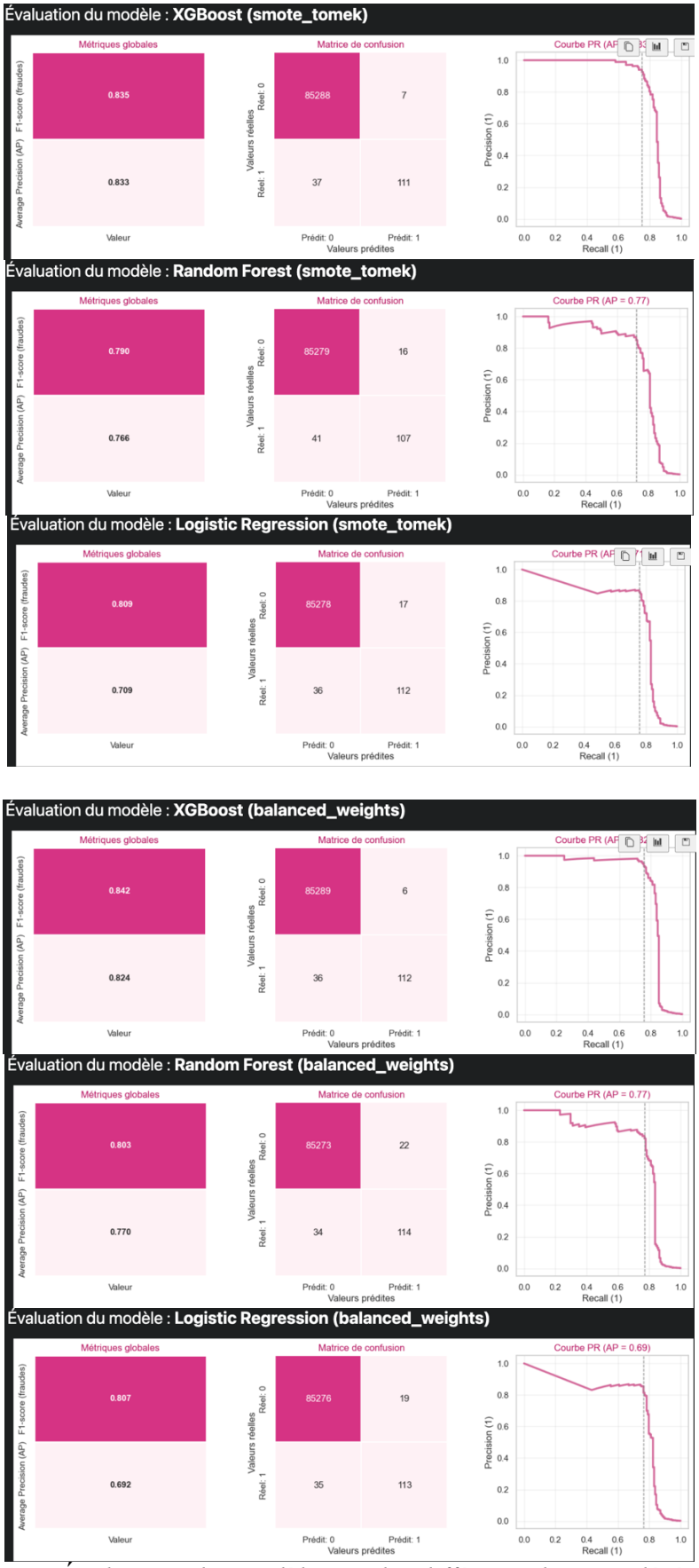


Figure 15 – Évaluation des modèles sur les différents datasets déséquilibrés

Les résultats synthétisés sur les quinze combinaisons (modèle × dataset) montrent une hiérarchie claire :

Modèle	F1 moyen	AP moyen	Interprétation
XGBoost	0.84	0.82	Excellent compromis précision/rappel, courbes PR dominantes
Random Forest	0.80	0.77	Solide et stable, mais légèrement en retrait
Régression logistique	0.78	0.69	Simple et interprétable, mais moins adaptée aux données non linéaires

	Dataset	Model	F1	AP	Threshold
0	original	XGBoost	0.842	0.824	0.455
1	original	Random Forest	0.803	0.770	0.462
2	original	Logistic Regression	0.807	0.692	1.000
3	smote	XGBoost	0.842	0.835	0.981
4	smote	Random Forest	0.790	0.766	0.914
5	smote	Logistic Regression	0.809	0.709	1.000
6	tomek	XGBoost	0.837	0.825	0.550
7	tomek	Random Forest	0.807	0.761	0.504
8	tomek	Logistic Regression	0.804	0.704	1.000
9	smote_tomek	XGBoost	0.835	0.833	0.988
10	smote_tomek	Random Forest	0.790	0.766	0.914
11	smote_tomek	Logistic Regression	0.809	0.709	1.000
12	balanced_weights	XGBoost	0.842	0.824	0.455
13	balanced_weights	Random Forest	0.803	0.770	0.462
14	balanced_weights	Logistic Regression	0.807	0.692	1.000

Figure 16 - Comparaison globale des modèles supervisés sur tous les rééquilibrages

Le modèle XGBoost se démarque par sa capacité à pondérer dynamiquement la classe minoritaire (`scale_pos_weight`) et à modéliser des interactions complexes. Les courbes Précision–Rappel confirment cette domination : ses courbes sont les plus hautes et les plus régulières, traduisant une excellente stabilité même pour des seuils stricts.

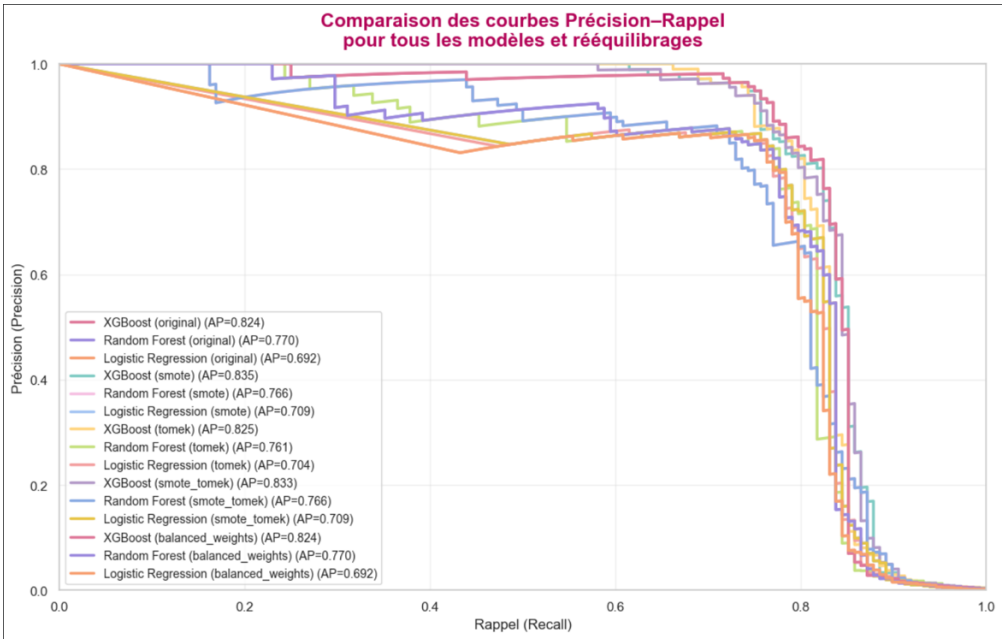


Figure 17 - Courbes Précision–Rappel pour tous les modèles et rééquilibrages

Les résultats montrent que le rééquilibrage n'améliore pas systématiquement les performances, surtout pour les modèles déjà robustes. Le modèle XGBoost reste le plus performant, grâce à sa gestion interne des classes minoritaires tandis que le modèle Random Forest constitue une alternative fiable et stable. La Régression Logistique elle, offre une baseline interprétable, utile pour valider la cohérence des tendances.

3.6 Conclusion

L'étude du jeu de données *Credit Card Fraud Detection* a permis de comparer différentes approches de détection d'anomalies dans un contexte de déséquilibre extrême.
Les modèles non supervisés (Isolation Forest et LOF) ont mis en évidence leurs limites dans la détection des fraudes : bien qu'ils identifient visuellement certaines observations aberrantes, leurs performances restent limitées en l'absence d'étiquettes d'apprentissage.
L'introduction des approches supervisées a nettement amélioré les résultats. Le modèle EasyEnsemble s'est montré efficace sans rééchantillonnage externe, confirmant la pertinence des méthodes intégrant une gestion interne du déséquilibre. Les tests menés sur les modèles classiques (XGBoost, Random Forest, Régression Logistique) et leurs versions rééquilibrées ont montré que le choix du modèle a un impact plus fort que la méthode de rééquilibrage elle-même. En particulier, XGBoost a offert les meilleures performances globales avec un F1-score supérieur à 0.84 et une Average Precision avoisinant 0.83, tout en conservant une excellente stabilité sur l'ensemble des variantes.
Globalement, les stratégies de rééquilibrage telles que SMOTE ou SMOTE + Tomek améliorent légèrement le rappel, mais leur effet reste secondaire face à la puissance des modèles ensemblistes. Ces résultats confirment que, dans les contextes à très faible taux d'anomalie, la performance repose avant tout sur des modèles capables d'intégrer la pondération des classes et d'exploiter les signaux faibles.

4 Partie III – Détection d'intrusions réseaux

4.1. Présentation du dataset

Le jeu de données *KDDCup99* est un dataset historique pour la détection d'intrusions. Il contient 494 020 connexions réseau décrites par 42 variables (numériques et binaires) et réparties entre trafic normal et types d'attaques.
Ces attaques ont été regroupées sous une seule étiquette binaire :

- `0` → trafic normal
- `1` → intrusion

Après encodage et normalisation (`StandardScaler`), le dataset final compte 115 variables normalisées. La distribution montre une majorité d'intrusions (≈ 80 %) contre 20 % de trafic normal.

Jeu de données chargé avec succès

Chemin : `../data/KDDCup99.csv`

Dimensions : 494020 lignes × 42 colonnes

Colonnes : duration, protocol_type, service, flag, src_bytes, dst_bytes, land, wrong_fragment, urgent, hot, num_failed_logins, logged_in, lnum_compromised, lroot_shell, lsu_attempted, lnum_root, lnum_file_creations, lnum_shells, lnum_access_files, lnum_outbound_cmds, is_host_login, is_guest_login, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate, label

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_
0	0	tcp	http	SF	181	5450	0	0	0	0	...	
1	0	tcp	http	SF	239	486	0	0	0	0	...	
2	0	tcp	http	SF	235	1337	0	0	0	0	...	
3	0	tcp	http	SF	219	1337	0	0	0	0	...	
4	0	tcp	http	SF	217	2032	0	0	0	0	...	

5 rows × 42 columns

Répartition des classes

- Trafic normal : **19.691%**
- Intrusions : **80.309%**

Figure 18 – Aperçu général du dataset

4.2. Méthodes non supervisées

Les deux mêmes algorithmes ont été appliqués sans utiliser les étiquettes : Isolation Forest et Local Outlier Factor (LOF).

4.2.1 Isolation Forest

Une grille d'hyperparamètres a été testée et le meilleur modèle est obtenu pour : `n_estimators=400`, `max_samples=0.8`, `contamination=0.25`.

Optimisation du modèle **Isolation Forest** sur le dataset **KDDCup99**

- Nombre de combinaisons testées : **6**
- Meilleur F1 : **0.225**
- Meilleur AP : **0.676**

Résultats de l'optimisation - Isolation Forest (KDDCup99)

	n_estimators	max_samples	contamination	F1	AP
4	500	0.800000	0.250000	0.225463	0.676356
3	400	0.800000	0.250000	0.222290	0.675986
5	700	0.800000	0.250000	0.220191	0.675565
1	200	0.600000	0.200000	0.183437	0.681723
2	300	0.800000	0.200000	0.172399	0.674189
0	100	0.600000	0.150000	0.138122	0.684538

Figure 19 - Optimisation du modèle Isolation Forest

Les résultats montrent une capacité modérée de détection avec : **F1 = 0.223** et **Average Precision (AP) = 0.676**.

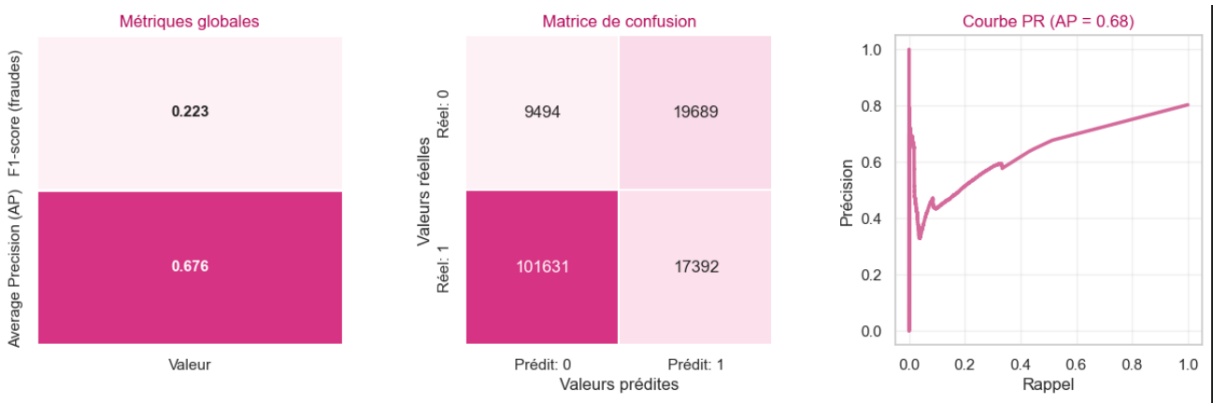


Figure 20 - Résultats de l'Isolation Forest

4.2.2 Local Outlier Factor

Une grille d'hyperparamètres a été testée et le meilleur modèle est obtenu pour : `n_neighbors = 50` et `contamination = 0.20` Afin de réduire le temps de calcul, le modèle a été entraîné sur un échantillon stratifié représentant 30 % du dataset d'origine (environ 150 000 lignes) tout en conservant la même proportion entre classes normales et intrusions.

Optimisation du modèle **Local Outlier Factor** sur le dataset **KDDCup99**

- Nombre de combinaisons testées : **3**
- Meilleur F1 : **0.266**
- Meilleur AP : **0.784**

Résultats de l'optimisation - Local Outlier Factor (KDDCup99)

	n_neighbors	contamination	F1	AP
2	100	0.250000	0.265731	0.784219
1	50	0.200000	0.219905	0.781977
0	20	0.150000	0.162671	0.767762

Figure 21 - Optimisation du modèle Local Outlier Factor

Les résultats montrent que LOF est légèrement plus sensible que l'Isolation Forest, **F1 = 0.252** et **AP = 0.770**. Ceci s'explique grâce à sa capacité à détecter les anomalies locales dans des zones de faible densité.

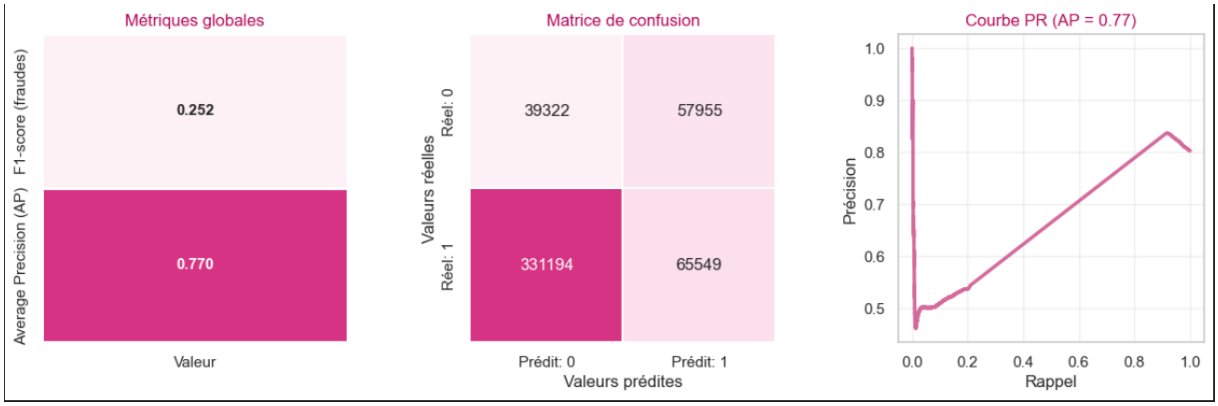


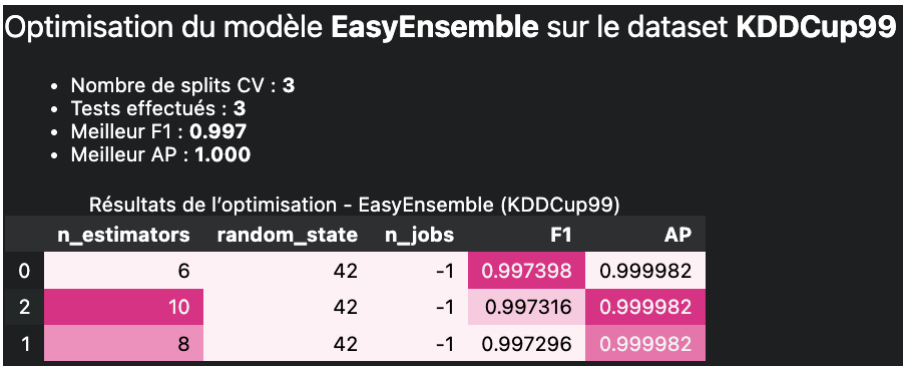
Figure 22 - Résultats de Local Outlier Factor

Les deux méthodes repèrent une partie des intrusions mais restent limitées sans supervision. LOF offre un meilleur rappel, tandis qu'Isolation Forest produit des scores plus stables. Ces approches constituent une première base utile pour la détection d'anomalies non étiquetées.

4.3 Easy Ensemble Classifier

L'approche EasyEnsembleClassifier a été appliquée afin de traiter le déséquilibre des classes dans le dataset KDDCup99 en interne. Le modèle combine plusieurs sous-échantillons équilibrés du jeu d'entraînement et entraîne un classifieur AdaBoost sur chacun d'eux, avant d'agréger les prédictions. Une optimisation a été menée sur le nombre d'estimateurs (`n_estimators ∈ {6, 8, 10}`) à l'aide d'une validation croisée à trois plis.

Les résultats montrent des performances très élevées et stables pour toutes les valeurs testées, avec un léger avantage pour `n_estimators = 6`.



	n_estimators	random_state	n_jobs	F1	AP
0	6	42	-1	0.997398	0.999982
2	10	42	-1	0.997316	0.999982
1	8	42	-1	0.997296	0.999982

Figure 22 – Optimisation du modèle EasyEnsemble

Le modèle final atteint un **F1-score de 0.998** et une **Average Precision (AP) de 1.000**, avec un seuil de décision optimal à 0.476.

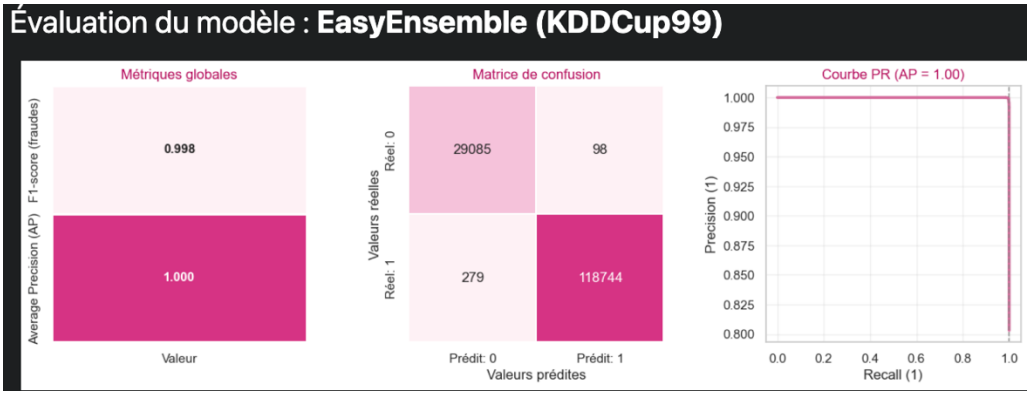


Figure 22 - Résultats du modèle EasyEnsemble

4.4 Approches supervisées avec modèle de rééquilibrage

Après les approches non supervisées (Isolation Forest et LOF), nous avons appliqué plusieurs modèles supervisés sur le dataset, en testant différentes stratégies de rééquilibrage pour gérer le déséquilibre important entre classes normales et intrusions. Les modèles évalués sont :

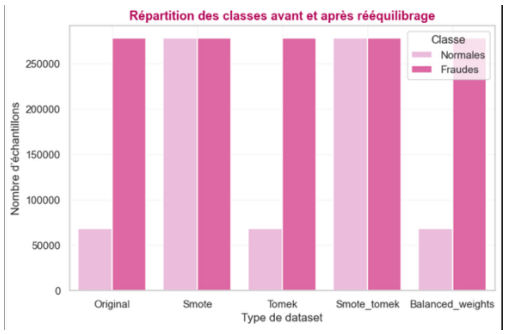
- XGBoost
- Random Forest
- Régression Logistique

Les jeux de données utilisés sont :

- Original (déséquilibré),
- SMOTE (sur-échantillonnage),
- Tomek Links (sous-échantillonnage),
- SMOTE + Tomek,
- Balanced Weights (pondération automatique des classes).

De la même manière que le dataset précédent, chaque modèle a été optimisé par validation croisée à trois plis, puis évalué selon deux métriques principales :

- F1-score, pour le compromis entre précision et rappel,
- Average Precision (AP), plus adaptée aux données déséquilibrées.



Ratios de classes avant/après rééquilibrage

	Dataset	Normales	Fraudes	Ratio (%)
0	Original	68,094	277,720	80.309
1	Smote	277,720	277,720	50.000
2	Tomek	68,094	277,703	80.308
3	Smote_tomek	277,713	277,713	50.000
4	Balanced_weights	68,094	277,720	80.309

Figure 23 – Répartition des classes avant et après rééquilibrages

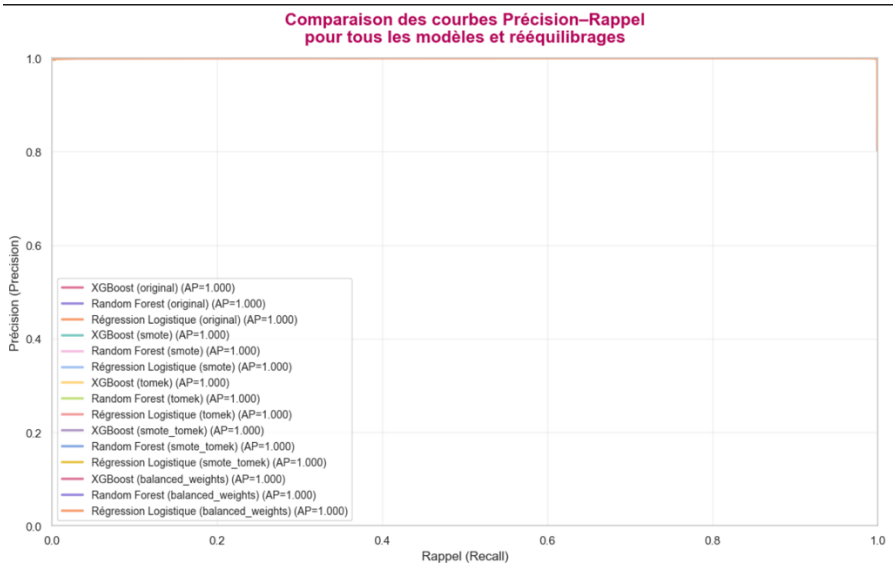
Les performances sont résumées dans le tableau ci-dessous. On observe des scores quasiment parfaits sur toutes les configurations : tous les modèles atteignent un **F1 proche de 1.0** et un **AP égal à 1.0**, indépendamment de la méthode de rééquilibrage utilisée.

	Dataset	Model	F1	AP	Threshold
0	original	XGBoost	1.000	1.000	0.418
1	original	Random Forest	1.000	1.000	0.271
2	original	Régression Logistique	0.999	1.000	0.251
3	smote	XGBoost	1.000	1.000	0.561
4	smote	Random Forest	1.000	1.000	0.334
5	smote	Régression Logistique	0.999	1.000	0.312
6	tomek	XGBoost	1.000	1.000	0.496
7	tomek	Random Forest	1.000	1.000	0.290
8	tomek	Régression Logistique	0.999	1.000	0.304
9	smote_tomek	XGBoost	1.000	1.000	0.521
10	smote_tomek	Random Forest	1.000	1.000	0.329
11	smote_tomek	Régression Logistique	0.999	1.000	0.290
12	balanced_weights	XGBoost	1.000	1.000	0.418
13	balanced_weights	Random Forest	1.000	1.000	0.271
14	balanced_weights	Régression Logistique	0.999	1.000	0.251

Figure 24 – Comparaison globale des modèles supervisés sur tous les rééquilibrages

Ces résultats montrent une très forte stabilité des modèles supervisés, quels que soient : la stratégie de rééquilibrage appliquée, ou le choix du modèle. Les trois approches atteignent quasiment la perfection sur le dataset KDDCup99, avec :

- Aucune perte de rappel sur les attaques,
- Précision maximale,
- Et courbes Précision–Rappel confondues à 1.0.



L'ajout de rééquilibrages (SMOTE, Tomek, pondération) n'a pas eu d'impact notable, ce qui s'explique par la taille très importante du dataset et la clarté des séparations entre classes dans KDDCup99. En revanche, cela démontre encore une fois que ces modèles sont très robustes face aux déséquilibres, ce qui serait un avantage considérable sur des données plus bruitées.

4.5 Conclusion sur le dataset KDDCup99

L'expérimentation sur le jeu KDDCup99 a permis d'évaluer l'efficacité des différentes approches de détection d'anomalies dans un contexte de cybersécurité. Les modèles non supervisés (Isolation Forest et LOF) ont démontré une capacité limitée à détecter les intrusions, avec des F1-scores compris entre 0.22 et 0.25 et une Average Precision autour de 0.70. Leur performance reste correcte pour une détection exploratoire sans étiquettes, mais leur sensibilité dépend fortement du paramétrage et de la structure locale des données.

Les approches supervisées ont, quant à elles, atteint des résultats quasi parfaits. Le modèle EasyEnsemble obtient F1 = 0.998 et AP = 1.000, tandis que XGBoost, Random Forest et Régression Logistique dépassent tous 0.999 de F1-score, indépendamment de la méthode de rééquilibrage appliquée. Ces scores reflètent la forte séparabilité des classes dans le dataset, où les comportements normaux et anormaux présentent des caractéristiques distinctes.

Ainsi, les méthodes supervisées surpassent largement les approches non supervisées, confirmant que l'apprentissage supervisé constitue la solution la plus adaptée lorsque des étiquettes fiables sont disponibles. Cependant, ces performances doivent être relativisées : la simplicité du dataset KDDCup99 rend la tâche de classification plus facile que sur des jeux de données modernes, où les anomalies sont plus subtiles et les distributions moins distinctes.

5 Conclusion générale

Ce travail a permis d'expérimenter et de comparer plusieurs méthodes de détection d'anomalies sur des jeux de données de nature différente :

- Mouse, illustratif et bidimensionnel,
- Credit Card Fraud, hautement déséquilibré,
- et KDDCup99, centré sur la détection d'intrusions réseaux.

Sur le plan méthodologique, le TP a montré la complémentarité entre approches non supervisées (Isolation Forest, LOF) et supervisées (EasyEnsemble, XGBoost, Random Forest, Régression Logistique). Les premières sont utiles dans un contexte exploratoire ou en absence de labels, tandis que les secondes offrent une détection beaucoup plus précise lorsqu'un apprentissage supervisé est possible.

Les expériences menées montrent que :

- Les modèles non supervisés présentent une efficacité modérée, limitée par le choix du paramètre de contamination et la distribution des données.
- Les modèles supervisés, en revanche, atteignent des performances très élevées sur des données bien étiquetées, même en présence d'un fort déséquilibre.
- Le rééquilibrage des classes (SMOTE, Tomek, pondération) améliore légèrement le rappel, mais son impact reste secondaire face à la puissance des modèles ensemblistes.

Globalement, la performance observée dépend moins du modèle utilisé que de la qualité de la préparation des données : encodage, normalisation et gestion du déséquilibre restent des étapes déterminantes. Les méthodes ensemblistes comme XGBoost et EasyEnsemble se démarquent par leur capacité à gérer les signaux faibles tout en maintenant une très forte précision. De plus, la combinaison d'une approche modulaire (fonctions factorisées) et d'une validation rigoureuse permet de reproduire facilement les expériences sur d'autres domaines (fraude, cybersécurité...).

En conclusion, ce travail met en évidence que la détection d'anomalies n'a pas de solution unique : elle repose sur un équilibre entre exploration non supervisée et apprentissage supervisé. Les premiers modèles permettent de découvrir, les seconds d'exploiter. Ensemble, ils offrent une méthodologie complète, robuste et réutilisable pour la détection d'événements rares dans des volumes massifs de données.