MANAGEMENT & DATASCIENCE

COMMENT LE MACHINE LEARNING PERMET DE DÉTECTER LA FRAUDE BANCAIRE ?

ALEXANDRE ALFOCEA

Publié dans Management & Datascience, le 7 avril 2021
Catégorie : Application
https://management-datascience.org/articles/16671/

RÉSUMÉ

Avec l'explosion des données massives, les banques font désormais appel à la Data Science et au Machine Learning pour détecter et prédire la fraude. Les difficultés du domaine se placent au sein même des données à exploiter, avec pour difficulté principale le déséquilibre des classes. La fraude ne représente qu'une infime partie des transactions bancaires et les modèles de prédiction classiques ne permettent pas de la prédire convenablement. Pour cela, il existe des solutions au problème de déséquilibre des classes. Dans ce sens, cet article a pour objectif d'exploiter les différentes méthodes de détection de fraude bancaire, en passant par le traitement du déséquilibre afin de permettre aux banques de mieux les combattre.

BANQUES | MACHINE LEARNING

Citation : ALFOCEA, A. (Avr 2021). Comment le machine learning permet de détecter la fraude bancaire ?. *Management et Datascience*, Article 0016671. https://management-datascience.org/articles/16671/.

L'auteur :

- **Alexandre ALFOCEA**
  - (Pas d'affiliation)

Copyright : © 2021 l'auteur. Publication sous licence Creative Commons CC BY-ND.

Liens d'intérêts : Le ou les auteurs déclarent ne pas avoir connaissance de conflit d'intérêts impliqués par l'écriture de cet article.

Financement : Le ou les auteurs déclarent ne pas avoir bénéficié de financement pour le travail mis en jeu par cet article.

TEXTE COMPLET

Introduction

Le secteur bancaire est au centre des flux de big data, qu'ils soient financiers ou non et l'application d'outils de machine learning par des datascientists expérimentés permet aux banques d'améliorer grandement leurs techniques de traitement des données, et indirectement, de multiplier les bénéfices. Dans cet article, nous nous focaliserons sur la recherche d'un algorithme de détection de fraude bancaire par un moyen de supervised learning.

La difficulté principale repose sur le déséquilibre des classes. Selon le rapport annuel 2019 de l'OSMP (2020), les transactions bancaires frauduleuses ne représentent que 0.001% des transactions, alors que le montant des transactions frauduleuses représente 1% des montants de transaction. Afin de résoudre ces problèmes de déséquilibre des classes, nous nous focaliserons dans un premier temps sur les données elles-mêmes à l'aide de méthodes de rééchantillonnage (resampling). Ensuite, nous étudierons des méthodes 'one-class', qui ont pour objectif d'étudier exclusivement une seule classe (généralement la fraude). Nous exploiterons le potentiel des méthodes dites 'cost-sensitive' ou sensibles aux coûts, qui ont pour objectif d'attribuer des coûts à une mauvaise classification et enfin nous étudierons des méthodes ensemblistes, méthodes robustes qui permettent d'assembler plusieurs algorithmes dans un seul et unique ensemble prédictif.

L'objectif de cette recherche est de répondre à la problématique « **Comment utiliser le machine learning pour résoudre le problème de déséquilibre des classes dans la détection de fraude bancaire ?** ». Afin d'y répondre, nous élaborerons des hypothèses et exploiterons différentes méthodes de traitement de déséquilibre des classes

afin de créer un modèle performant de détection de fraude bancaire.

Modèles utilisés

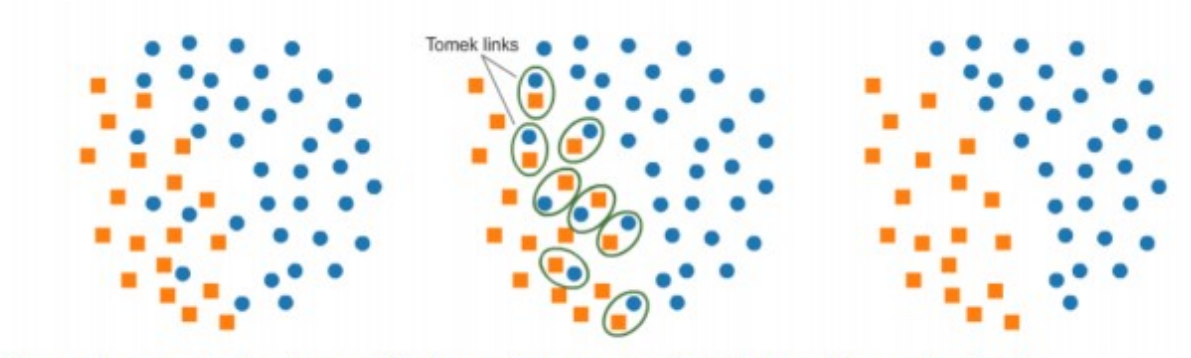
Over et under sampling

Les méthodes d'over et under sampling ont pour objectif de retraiter le volume de données pour obtenir un équilibre des classes. Les méthodes de random sampling vont rééquilibrer aléatoirement les classes en dupliquant la classe minoritaire pour l'oversampling et en réduisant la classe majoritaire pour le undersampling. Outre les méthodes aléatoires, il existe des méthodes contrôlées comme les liens Tomek pour le undersampling et la méthode SMOTE pour l'oversampling.

Tomek Links undersampling

Les liens Tomek sont des paires d'observations très proches, mais de classes opposées. La suppression des observations de la classe majoritaire de chaque paire augmente l'espace entre les deux classes, ce qui facilite le processus de classification. (Tomek I., 1976).

Graphique 1 : Processus de resampling Tomek links.

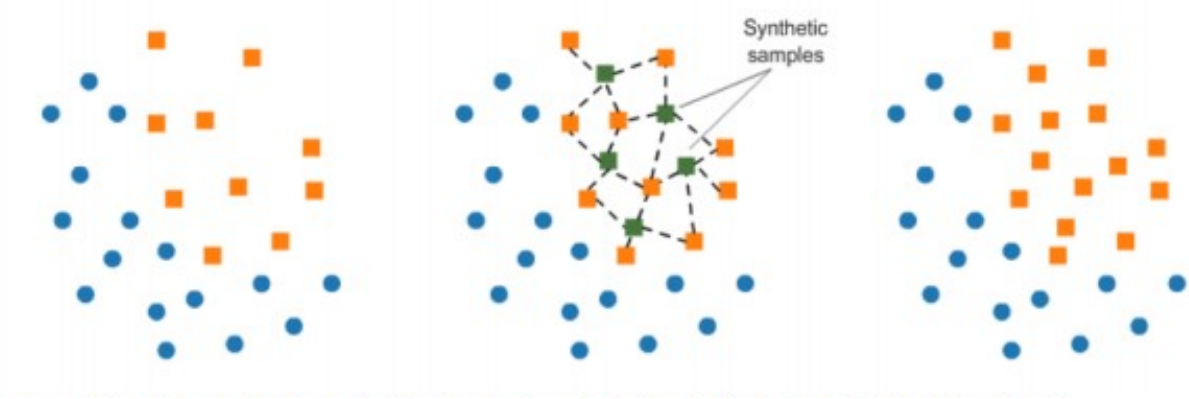


Source : https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook.

SMOTE oversampling

SMOTE (Synthetic Minority Oversampling Technique) consiste à synthétiser des éléments de la classe minoritaire, sur la base de ceux qui existent déjà. Il sélectionne aléatoirement un point de la classe minoritaire et calcul les k-voisins les plus proches pour ce point. Les points synthétiques sont ajoutés entre le point choisi et ses voisins. (Chawla 2002, Gu X., 2019).

Graphique 2 : Processus de resampling SMOTE.



Source : https://www.kaggle.com/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook.

One-Class classification

La classification One-Class est une méthode de classification se focalisant uniquement sur une seule classe du dataset. Cette méthode permet d'apprendre en profondeur la classe majoritaire (appelées valeurs normales, inliers en anglais) afin d'optimiser les prédictions de cette classe. Le classifieur met de côté la classe minoritaire (appelées valeurs aberrantes ou anomalies, outliers en anglais).

OneClassSVM

La Machine Vectorielle de Support SVM est un algorithme qui permet de créer un hyperplan séparant les inliers (non-fraude) et les outliers (fraude). Le principe de SVM one-class repose sur le principe de l'algorithme SVM mais se focalise uniquement sur les observations de la classe majoritaire. Un hyperplan sera créé là où la classe majoritaire est la plus dense (son support), séparant ainsi tous les autres points qui seront considérés comme anomalies ou fraudes. (Kamaruddin et Ravi, 2016).

Cost-sensitive models

A la différence des modèles de prédictions classiques qui ont pour objectifs de minimiser la fonction de perte afin de réduire les erreurs de classification, les méthodes cost-sensitives ont pour objectif de réduire le coût attribué à l'ensemble du modèle. Ces méthodes attribuent un coût à chaque mauvaise classification et un coût global est calculé.

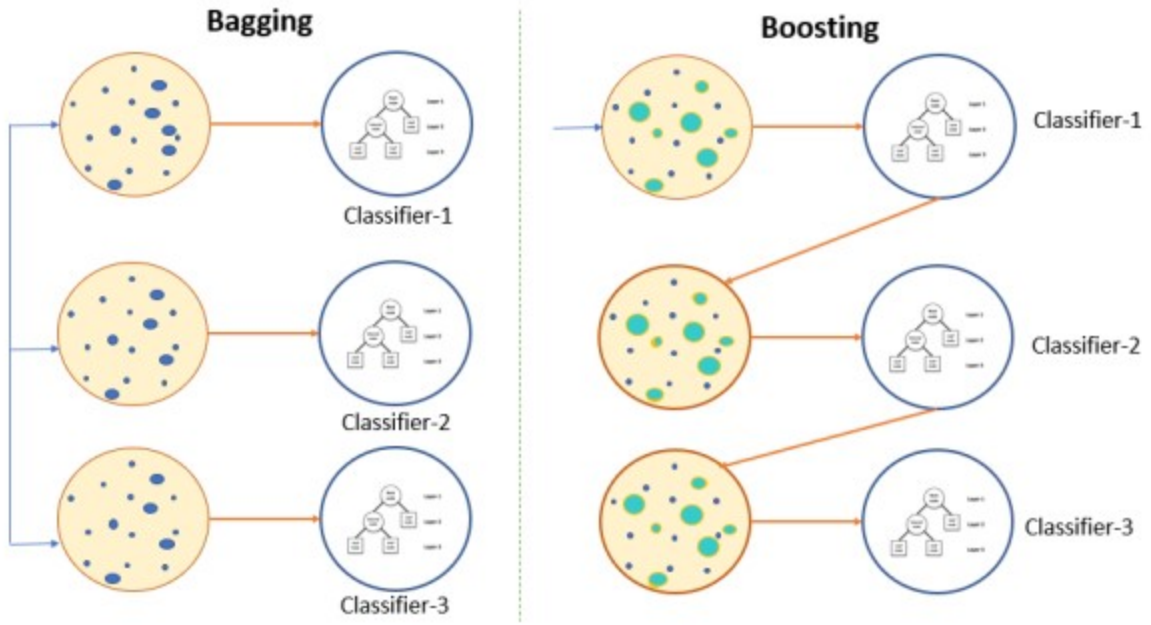
Cost-sensitive Decision Tree

Un arbre de décision est un enchainement de règles appliquées sur un dataset afin de séparer les données en deux ou plusieurs groupes. Nous avons choisi un coût de 3 pour chaque mauvaise classification de fraude et de 1 pour chaque mauvaise classification de non-fraude : {0:1, 1:3).

Cost-sensitive XGboost

Contrairement au bagging, le boosting est un enchainement du même algorithme qui produit un nouvel échantillon après chaque apprentissage. Chaque nouvel échantillon est généré à partir des observations difficiles à classifier, ce qui permet d'être de plus en plus précis dans les prédictions. XGboost repose sur le principe de gradient (Metzler G., 2020). Pour la sensibilité au coût, nous avons choisi un coût de {0:1, 1:3).

Graphique 3 : Processus de bagging vs boosting.



Parallel ***Sequential***

Source : https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting.

Cost-sensitive SVC

Le SVC (Support Vector Classifier) est le terme utilisé du SVM pour la classification. Nous avons choisi des coûts de : {0:2, 1:4).

Cost-sensitive ensembles

Les méthodes ensemblistes utilisent plusieurs algorithmes ou plusieurs fois le même algorithme et combinent les résultats afin d'obtenir les meilleurs. L'utilisation d'un apprentissage ensembliste permet de rendre les prédictions plus stables sur de nouveaux échantillons de test. (Yong Zhang and Dapeng Wang, 2013).

BalancedBaggingClassifier et EasyEnsembleClassifier (équivalent)

Le Balanced bagging classifier est un algorithme d'apprentissage ensembliste qui utilise des méthodes de bagging (ou bootstrap aggregating) en ajoutant à chaque boostrap (échantillon) une méthode de resampling. Ici nous utiliserons la méthode de resampling par défaut Random Under Sampling.

RUSBoostClassifier

Ce classifieur est un ensemble d'algorithme d'arborescence et de resampling itératif (pas de bagging).

Hypothèses

- Les méthodes 'cost-sensitive' sont plus efficaces que les méthodes d'échantillonnage pour détecter la fraude.
- Les algorithmes de boosting sont plus performants que les algorithmes classiques pour détecter la fraude.
- Les algorithmes ensemblistes sont plus efficaces que les algorithmes non-ensemblistes.

Méthodologie

Afin de simplifier notre démarche empirique, nous avons décidé de nous focaliser sur le supervised learning et de travailler avec des données dont la variable à prédire est labellisée en fraude ou non-fraude. Le travail s'effectue en langage de programmation python via la plateforme Jupyter Notebook et accessoirement la plateforme Google Colab pour la puissance de calcul.

Nous avons utilisé les packages Scikit-learn, imblearn et des packages de base comme numpy, pandas (retraitement des données), matplotlib, seaborn (visualisation).

Dataset

Extrait des 5 premières lignes :

```
df.head()
```

	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1	0
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1	0
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0	0

Description des variables

- ‘type’ : type de transaction : payment, transfer, cash out, cash in, debit.
- ‘amount’ : montant de la transaction.
- ‘nameOrig’ : identifiant de l’émetteur.
- ‘oldbalanceOrig’ : solde de l’émetteur avant la transaction.
- ‘newbalanceOrig’ : solde de l’émetteur après la transaction.
- ‘nameDest’ : identifiant du destinataire.
- ‘oldbalanceDest’ : solde du destinataire avant la transaction.
- ‘newbalanceDest’ : solde du destinataire après la transaction.
- ‘isFraud’ : 1 si la transaction est frauduleuse, sinon 0.
- ‘isFraggedFraud’ : 1 si la transaction a été signalée frauduleuse, sinon 0.

Preprocessing : Retraitement des données

Avant d’analyser les données et d’entraîner les modèles, il convient de retraiter les données pour qu’elles soient compréhensibles par l’API de scikit-learn.

Il est important de noter que scikit-learn ne peut travailler que sur des variables numériques. Il convient donc de transformer les variables catégorielles en numériques.

Nous avons effectué différents retraitements comme un OneHotEncoder pour la colonne ‘type’ et pour les identifiants ‘nameOrig’ et ‘nameDest’, nous avons séparé le premier caractère de la chaîne (binaire : ‘C’ ou ‘M’) du reste pour le transformer en binaire numérique (0 ou 1).

Dataset retraité

Extrait des 5 premières lignes :

```
df.head()
```

	step	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud	isFlaggedFraud	typeDestBin	type_CASH_OUT	type_DEBIT	type_PAYMENT
0	1	9839.64	1231006815	170136.0	160296.36	1979787155	0.0	0.0	0	0	0	0	0	1
1	1	1864.28	1666544295	21249.0	19384.72	2044282225	0.0	0.0	0	0	0	0	0	1
2	1	181.00	1305486145	181.0	0.00	553264065	0.0	0.0	1	0	1	0	0	0
3	1	181.00	840083671	181.0	0.00	38997010	21182.0	0.0	1	0	1	1	0	0
4	1	11668.14	2048537720	41554.0	29885.86	1230701703	0.0	0.0	0	0	0	0	0	1

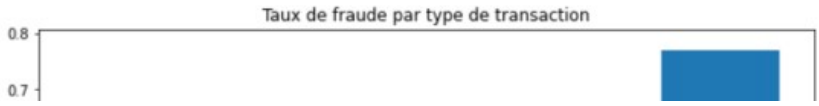
De plus, nous avons décidé d’inclure un StandardScaler et un PCA (Principal Component Analysis) directement dans les pipelines des modèles prédictifs.

- Le Standard Scaler a pour objectif de standardiser les variables en soustrayant la moyenne et en divisant par l’écart type afin d’avoir des variables sur la même échelle.
- Le PCA a pour objectif de décorréler les variables autour du point [0,0] et de sélectionner uniquement les composantes principales (variables à plus haute variance).

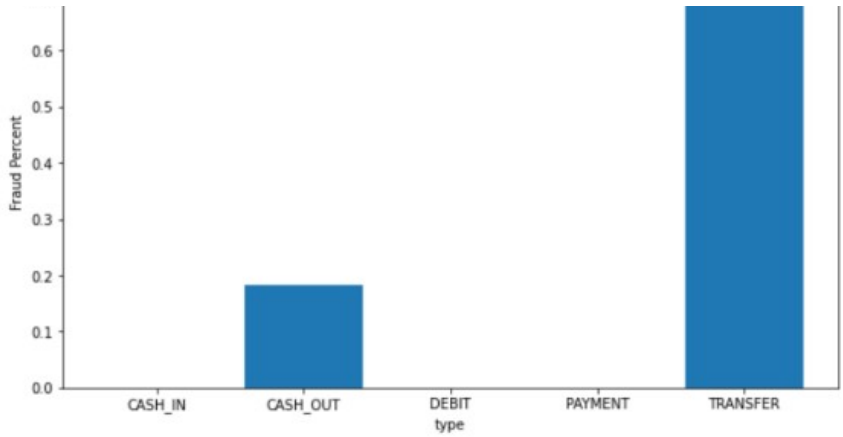
EDA (Exploratory Data Analysis) : Analyse des données

Cette étape permet d’analyser les données pour mieux les comprendre.

Graphique 4 : Taux de fraude par type de transaction.

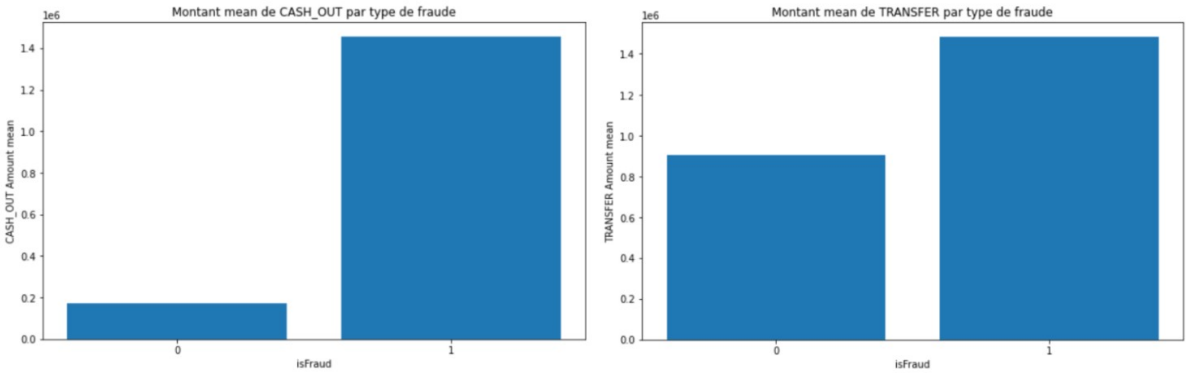


	type	isFraud
0	CASH_IN	0.000000
1	CASH_OUT	0.001840
2	DEBIT	0.000000
3	PAYMENT	0.000000
4	TRANSFER	0.007688



Nous pouvons observer que les transactions frauduleuses sont uniquement des retraits (cash_out) et des virements (transfer) avec des taux de fraude de 0.8% pour les virements et de 0.2% pour les retraits (graphique 4).

Graphique 5 : Montants moyens de CASH_OUT et de TRANSFER par type de transaction.



Observons que les montants des fraudes sont très supérieurs aux montants des transactions normales (graphique 5). En moyenne, les fraudes représentent un montant de 1,4 millions pour les retraits et 1.5 millions pour les virements, contre 170 000 et 900 000 pour les transactions normales respectivement.

Model exploration : Mesures de performance, comparaison des modèles.

Cette étape est la mise en place des modèles, les mesures de performances et la comparaison des modèles. Etudions cela dans les résultats.

Résultats

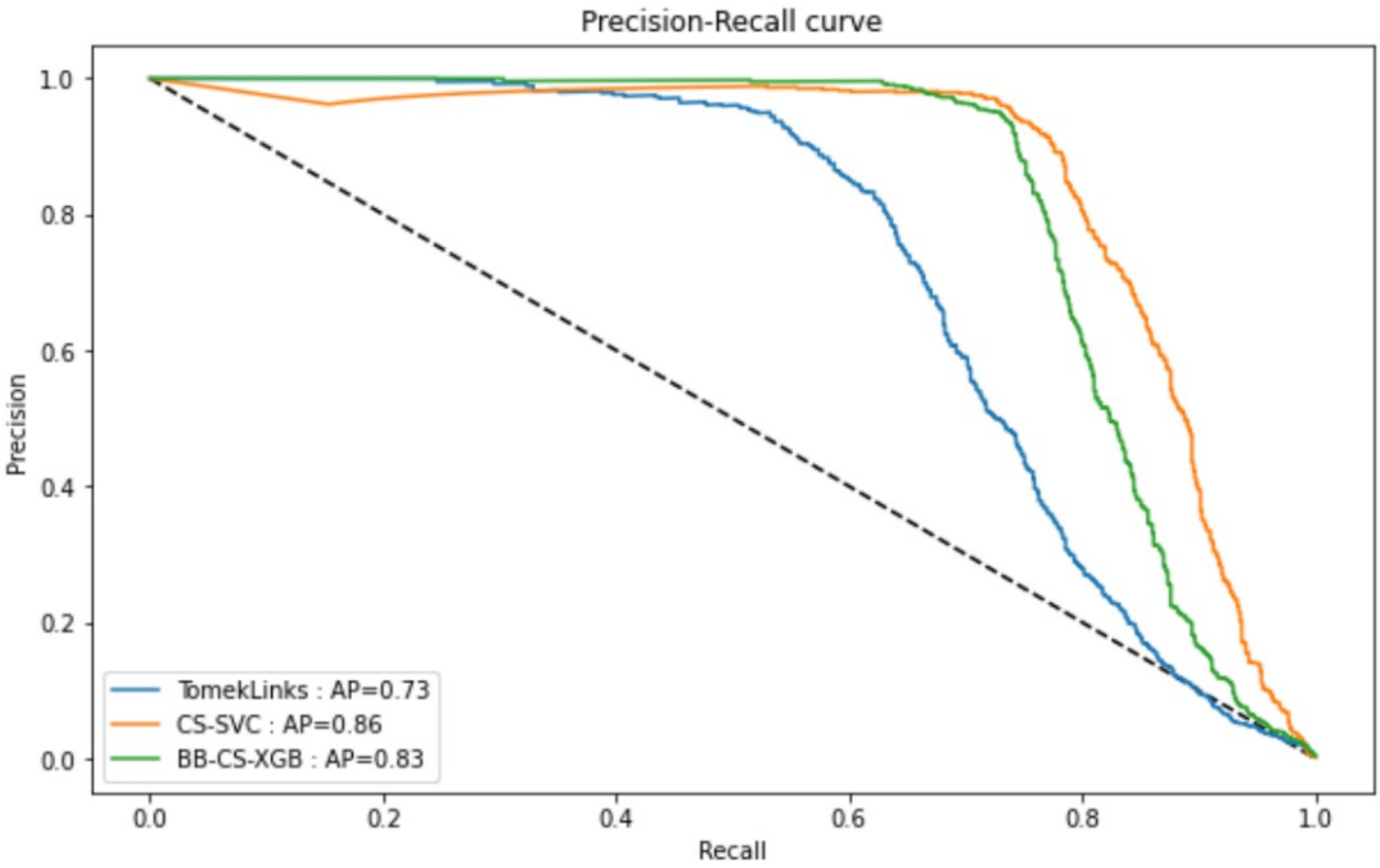
Les analyses des différentes méthodes de traitement de déséquilibre des classes et de détection de fraudes nous permettent d'identifier, parmi chaque méthode, l'algorithme le plus performant relativement à notre étude.

Tableau 2 : Scores des métriques des modèles les plus performants.

Modèle	F1-score	PR AUC	Recall	Precision
Tomek Links	0.70	0.73	0.57	0.90
OC SVM	0.17	-	0.22	0.15
CS-SVC	**0.83**	**0.86**	**0.72**	**0.97**

BB XGboost	0.81	0.83	0.70	0.96

Graphique 6 : Représentation graphique des PR AUC scores du tableau 2



Les méthodes de resampling sont inadaptées à notre dataset en raison du déséquilibre très important des classes (~0.001% de fraude). La méthode des liens Tomek n'a que très peu changé les scores du modèle (Régression Logistique) lorsqu'il est utilisé sans resampling.

Parmi les méthodes One-class, le SVM ressort avec les meilleures performances. Cependant, les résultats obtenus restent loin des performances attendues et se montrent bien inférieurs aux autres méthodes de classe déséquilibrées.

Observons que parmi les méthodes cost-sensitives, le CS-SVC ressort plus performant que des méthodes de boosting comme XGboost avec un PR AUC de 0.86 et un f1-score de 0.83. Attribuer des coûts à chaque mauvaise classification est donc une méthode très performante pour résoudre les problèmes de classes déséquilibrées.

Parmi les méthodes ensemblistes, le Balanced Bagging XGboost se présente comme étant le plus performant avec un PR AUC de 0.83 et un f1-score de 0.81.

Discussion

Krawczyk B. (2016) explique dans son étude que lorsque les données sont extrêmement déséquilibrées, comme c'est le cas pour la détection de fraude, les méthodes de resampling détériorent fortement les performances des algorithmes, ce que nous avons pu démontrer lors de notre étude. Nous pouvons donc répondre positivement à l'hypothèse 1 : « Les méthodes 'cost-sensitive' sont plus efficaces que les méthodes d'échantillonnage pour détecter la fraude. ».

De plus, dans sa thèse en 2019, Metzler G. obtient des résultats très performant avec des algorithmes basés sur des arbres de décision, et parmi ses résultats, il obtient des résultats plus performants avec la méthode de cost-sensitive Gradient Boosting. Ce que nous avons pu démontrer dans notre étude et permet de répondre positivement à l'hypothèse 2 « Les algorithmes de boosting sont plus performants que les algorithmes classiques pour détecter la fraude ».

Les résultats obtenus avec les méthodes cost-sensitive et les méthodes ensemblistes sont très similaires et par ailleurs, le CS-SVC présente des scores plus élevés que les autres algorithmes. Cependant, si nous considérons que les méthodes ensemblistes réduisent fortement les risques d'overfitting et permettent d'apprendre la totalité des données entrainées, nous pouvons conclure que ces méthodes sont très robustes et stables, et donc que l'hypothèse 3 est démontrée.

Limites de l'étude

Cependant, le déséquilibre des classes est un problème difficile à traiter en raison de la diversité des données à exploiter. La fraude n'est qu'un exemple. Il n'existe pas de méthode universelle pour traiter le déséquilibre des classes et il convient donc d'utiliser de nombreuses méthodes pour trouver la meilleure. De plus, certains algorithmes peuvent prendre beaucoup de temps d'exécution, il est donc parfois nécessaire de faire des concessions, et comme nous, de se séparer d'une partie des données.

Enfin, nous nous somme focalisés sur le Supervised Learning mais il peut être intéressant de travailler avec d'autres domaines comme le Unsupervised Learning ou encore le Deep Learning pour traiter ces problèmes et détecter la fraude.

Conclusion

A travers cet article, nous avons cherché à comprendre comment utiliser le machine learning pour résoudre le problème principal du déséquilibre des classes dans la détection de fraude bancaire.

Après avoir étudié différentes méthodes pour résoudre ce problème, nous avons pu répondre positivement à nos trois hypothèses. En adéquation avec les recherches de plusieurs scientifiques, nous avons pu conclure que les méthodes 'cost-sensitive' et ensemblistes sont les plus performantes, et par ailleurs, les méthodes (i) ensembliste de Balanced Bagging Cost Sensitive XGboost et (ii) CS-SVC ont montré des résultats très performants.

BIBLIOGRAPHIE

Alfocea A., 2021 « *L'utilisation du Machine Learning pour détecter la fraude bancaire* ». Thèse professionnelle.

Gu X., Angelov P. and Almeida Soares E., "*A Self-Adaptive Synthetic OverSamplingTechnique for Imbalanced Classification*", 2019.

Kamaruddin S. and Ravi V., Aug. 2016, ''*Credit card fraud detection using big data analytics: Use of PSOAANN based one-class classification*,'' in Proc. Int. Conf. Inform. Anal., pp. 1–8.

Krawczyk B., 2016, ''*Learning from imbalanced data: Open challenges and future directions*,'' Prog. Artif. Intell., vol. 5, no. 4, pp. 221–232.

Makki S., Haque R., Taher Y., Assaghir Z., Hacid M.S., Zeineddine H., 2019, "*A Cost-Sensitive Cosine Similarity K-Nearest Neighbor for Credit Card Fraud Detection*".

Metzler G., 2020, "*Learning from imbalanced data: an application to bank fraud detection*".

OSMP, 2020, « *Rapport annuel de l'Observatoire de la sécurité des moyens de paiement 2019.* ».

Tomek I., 1976, "*Two modifications of cnn. IEEE Trans. Systems, Man and Cybernetics*", 6: 769–772.

Zhang Y., Wang D., 2013, "*A Cost-Sensitive Ensemble Method for Class-Imbalanced Datasets*".

© 2021 MANAGEMENT & DATASCIENCE