MASTER EN SCIENCES INFORMATIQUES À FINALITÉ SPÉCIALISÉE EN DATA SCIENCE

Détection d'anomalies dans des logs applicatifs à l'aide de méthodes ensemblistes non-supervisées

VAN AELST, Antoine

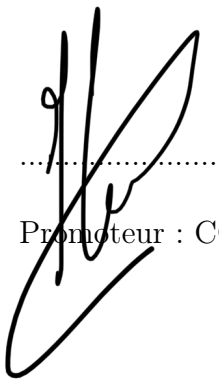
Award date:
2023

Awarding institution:
Universite de Namur

UNIVERSITÉ DE NAMUR
Faculté d'informatique
Année académique 2022-2023

Détection d'anomalies dans des logs applicatifs à l'aide de méthodes ensemblistes non-supervisées

VAN AELST Antoine



..... (Signature pour approbation du dépôt - REE art. 40)

Promoteur : COLIN Jean-Noël

Mémoire présenté en vue de l'obtention du grade de Master 120 en Sciences Informatiques à finalité Data Science

Faculté d'Informatique – Université de Namur

RUE GRANDGAGNAGE, 21 ● B-5000 NAMUR(BELGIUM)

Remerciements

Ce document est le fruit d'un travail de longue haleine, débutant bien avant les premières lectures autour de ce sujet de recherche. Je tiens par cette section à remercier les personnes qui ont contribué à cette rédaction.

Je remercie mon promoteur M. Jean-Noël Colin pour ses retours pertinents et ses critiques constructives sur le projet, mais aussi pour le soutien moral qu'il a pu apporter lorsque c'était nécessaire.

Je remercie l'entièreté du corps pédagogique de la Faculté d'Informatique de l'Unamur pour le contexte d'apprentissage qu'ils ont pu mettre en place et la qualité de la formation qu'ils ont pu apporter tout au long de ces 5 années.

Je tiens également à témoigner ma reconnaissance envers l'entièreté du groupe de recherche en cybersécurité de l'Université de Sherbrooke pour la qualité du stage que j'ai pu vivre à leurs côtés. Particulièrement, je tiens à remercier M. Pierre Martin Tardif et M. Thibaud Ecarot pour leur accueil chaleureux et leur disponibilité sans faille, mois après mois.

Enfin, je tiens à remercier plus personnellement quelques personnes pour l'aide directe ou indirecte qu'ils ont pu apporter dans mon parcours.

Mes parents pour l'éducation qu'ils m'ont apporté, et ma mère tout particulièrement pour sa relecture attentive.

Simon Lejoly, pour sa relecture, pour m'avoir soutenu à travers chaque épreuve de ces 5 dernières années mais aussi pour toutes les valeurs qu'il incarne au quotidien.

Blandine, Thomas, Antoine, pour les beaux moments qu'ils m'ont fait traverser et les rires qu'on a pu partager.

Enfin, mes pensées vont vers mon groupe d'amis les plus proches : Artiom, Emilien, Josua, Tristan D., Jürgen, Tristan L., Thomas, Gaël, Louis, Cédric, Hugo, Matteo, Nathan. Je suis fier de grandir avec vous.

Résumé

Les modèles ensemblistes, ou ensemble models sont une solution peu exploitée dans le paysage actuel de la détection d'anomalies non-supervisée. Cette approche est toutefois utilisée par l'article de Louis-Simon Letourneau afin de traiter des logs applicatifs de l'entreprise Sherweb dans un cadre de cybersécurité. La solution proposée offre une solution satisfaisante à une majorité des challenges imposés par l'absence de labels dans le dataset, cependant, un manque d'étude et de compréhension du dataset pousse à obtenir quelques résultats problématiques. Ce document propose une revue du travail effectué par Letourneau pour ensuite aborder plusieurs axes d'amélioration. Malgré la difficulté à valider les résultats en l'absence de labels, des métriques introduites dans le cadre de la recherche poussent à valider deux améliorations du modèle en place : l'intégration d'une feature supplémentaire basée sur le type d'évènement, ainsi que le remplacement du sous-modèle LOF pour préférer un modèle d'Isolation Forest. Les résultats obtenus s'avèrent encourageants et offrent de nombreuses autres perspectives d'amélioration.
Mots-clés *: détection d'anomalies, méthode ensembliste, machine learning, apprentissage non-supervisé, cybersécurité*

Abstract

Ensemble models are a little-exploited solution in the current landscape of unsupervised anomaly detection. However, this approach is used in Louis-Simon Letourneau's article to process Sherweb's application logs within a cybersecurity framework. The proposed solution offers a satisfactory solution to a majority of the challenges imposed by the absence of labels for its data, however, a lack of study and understanding of the dataset leads to some problematic results. This document provides a review of Letourneau's work, followed by a number of suggestions for improvement. Despite the difficulty of validating the results in the absence of labels, the metrics introduced during this research lead to the validation of two improvements to the existing model : the integration of an additional feature based on event type, and the replacement of the LOF submodel in favor of an Isolation Forest model. The results obtained are encouraging, and offer many other prospects for improvement.
Keywords *: anomaly detection, ensemble method, machine learning, unsupervised learning, cybersecurity*

Table des matières

1 Introduction

1.1 Contexte

Les données, sous toute taille et toute forme, sont un enjeu central dans à peu près toute discipline et champ professionnel moderne. Cette réalité est d'autant plus concrète depuis la récente émergence des modèles d'intelligence artificielle dite "avancée", redistribuant les cartes de nombreux secteurs, et amenant avec eux des enjeux économiques, culturels et éthiques qui n'ont pas à rougir à côté de ceux amenés par les plus grandes révolutions technologiques de notre histoire. En l'espace de quelques mois, ces nouveaux modèles et leur démocratisation ont amené nombre de bouleversements dans les sphères de l'information, mais surtout la désinformation, l'apprentissage mais aussi l'éducation, de l'art mais aussi du rapport au travail. ChatGPT, Midjourney, DALL-E, les noms de ces technologies sont nombreux et leurs conséquences probables sur le quotidien des citoyens du futur le sont tout autant.

Le point commun de toutes ces intelligences artificielles 2.0, c'est leur appétit sans limite pour les données de toute forme, qui renforcent leurs résultats au terme d'une phase d'entraînement demandant à chaque génération plus de ressources. Heureusement pour elles, les données se font elles aussi plus nombreuses et variées que jamais. Les progrès exponentiels que l'on observe depuis plusieurs dizaines d'années en terme de vitesse de calcul, de capacités de stockage, et la montée en puissance de réseaux sociaux publics amènent naturellement avec eux un goût tout nouveau pour la récolte de données, données dans lesquelles se cachent parfois la solution miracle pour comprendre au mieux les besoins et problématiques des clients, et faire décoller son entreprise, dans un marché plus concurrentiel que jamais.

Avec ces amas de données grandissant de jour en jour, d'heure en heure et de seconde en seconde, de nouvelles problématiques naissent. Le temps où chaque ligne client de la base de donnée pouvait être monitorée à la main est maintenant révolu et il faut désormais trouver de nouvelles solutions pour tirer l'essence de ces données qui s'écrivent plus vite qu'on arrive à les étudier. Parmi les domaines les plus touchés par cette problématique, on cite le plus souvent le secteur bancaire, la médecine ou bien de la sécurité informatique, dans lesquels chaque petite transaction, fiche patient ou commande système a une portée potentiellement critique, quand bien même ces lignes sont chaque minute plus nombreuses. Dans les domaines de ce genre qui ne laissent que peu droit à l'erreur, il est central de placer l'optimisation et l'amélioration de l'analyse de données au coeur des problématiques.

La détection d'anomalies est un bon exemple de domaine concerné par ces pro-

blématiques. L'identification d'un élément dénotant du reste parmi un ensemble de données considérées comme "normales" est aujourd'hui une nécessité. Là où il était possible à une époque pour un médecin de passer en revue les profils d'un hôpital pour vérifier que chaque patient est en bonne santé, la quantité actuelle de dossiers transitant d'un établissement à un autre rend nécessaire l'utilisation d'algorithmes faisant le tri pour réduire les profils à passer en revue. Un noyau de profils certes, mais au sein duquel les risques sont plus élevés qu'à l'accoutumée. De la même manière, les failles informatiques étant découvertes jour après jour, il est impossible pour un expert en sécurité de prévoir la signature des prochaines attaques. Les solutions telles que les Intrusion Detection Systems (IDS), entièrement basées sur la détection de schémas d'attaques déjà observés auparavant montrent désormais leurs limites. Définir un profil habituel des utilisateurs et sonner l'alarme quand leur comportement a l'air de s'en éloigner se présente dès lors comme une philosophie de défense fondamentalement meilleure.

À cela s'ajoute un autre problème. Les ensembles de données étant de plus en plus titanesques, donner à un expert les moyens et le temps de passer en revue un ensemble de données de manière exhaustive pour les labelliser de plus en plus couteux, voire inenvisageable pour une grande partie de l'industrie. Ainsi, la majorité des entreprises disposent aujourd'hui de grands sets de données vierges de toute annotation. Cela pose un problème pour les modèles de machine learning classiques dits "supervisés", pour qui les labels d'indication attachés à ces données sont au coeur de leurs techniques d'apprentissage.

Face à ces situations, le machine learning non supervisé se révèle être l'outil le plus efficace pour clusteriser, trier, ou même noter ces ensembles de données. De nombreuses techniques existent se basant sur des aspects différents des données : leur proximité, leur densité, leur répartition, ... Chacune de ces techniques révèle ainsi quelque chose de différent sur les données, et une transaction bancaire peut ainsi être considéré comme "anormale" de beaucoup de façons différentes.

Il existe cependant un moyen de combiner les défauts et avantages de plusieurs techniques entre elles par l'utilisation de méthodes dites "ensemblistes". Celles-ci reposent sur l'application de différents algorithmes de détection, décision, .. et permettent d'apporter une décision plus nuancée par la combinaison des résultats différentes méthodes entre eux, de la façon de notre choix. Ainsi, si les modèles renvoient des scores, il est possible de faire la moyenne des différents scores associés. Si les modèles sont chargés de labelliser des données, le modèle ensembliste peut décider de fonctionner à la majorité, ou encore en pondérant chaque algorithme à l'aide d'un niveau de confiance associé. Les modèles ensemblistes sont de ce point de vue un moyen de décupler les possibilités offertes par les modèles de machine learning isolés. C'est dans ce contexte que se déroule notre travail de recherche.

1.2 Description du projet

Le mémoire présenté au sein de ce document se place dans le cadre d'un Master en Data Science suivi au sein de la Faculté d'informatique de l'Unamur. Le travail confié se base sur un travail précédemment effectué au sein de l'Université de Sherbrooke, au Québec, par Louis-Simon Letourneau[15]. Le travail a donc été réalisé dans le cadre d'un stage de recherche d'une durée 3 mois au sein du laboratoire en cybersécurité de l'Université de Sherbrooke.

La mission proposée est ainsi de reprendre le travail réalisé où il est laissé afin de l'étendre de la manière qui semble la plus pertinente. Celui-ci est décrit dans un article scientifique utilisée comme base à cette production, en plus du code directement repris.

Le travail de Letourneau, décrit plus en détail dans une section suivante, propose un modèle de détection d'anomalies, adapté à un contexte de cybersécurité. Ce modèle propose comme spécificité d'être construit autour d'une méthode ensembliste(le terme ensemble model sera utilisé dans la suite du document afin d'être fidèle à la littérature), sur base de 4 méthodes de détection non supervisées. Au sein de la Section 3, les thématiques de cohésion d'ensemble method ainsi que de variance des résultats seront abordées afin d'en dériver des modifications de la solution en place.

1.3 Question de recherche

Sur base des fondations à disposition, la question de recherche suivante est donc formulée :

Comment améliorer la cohésion d'une méthode ensembliste non supervisée appliquée à de la détection d'anomalies, tout en réduisant la variance au sein des résultats ?

Organisation du document

Le document est organisé de la manière suivante. La Section 2 pose les bases théoriques nécessaires à la compréhension du lecteur tout en proposant un tour d'horizon des technologies employées dans le paysage actuel de la recherche. Elle conclut en présentant en détail le travail de Letourneau. La Section 3 passe en revue le travail en question sous un oeil critique, y identifie des axes d'amélioration et présente les solutions proposées en ce sens. La Section 4 décrit les résultats dérivés des différentes modifications. La Section 5 apporte un oeil critique sur ces résultats et en propose une analyse, tout en proposant un raffinement des modifications présentées précédemment. La Section 6, enfin, propose un résumé

du projet et des pistes pour de potentielles extensions, tout en apportant réponse à la question de recherche.

2 État de l'art

Le projet décrit dans ce document se présente comme la suite d'un travail effectué quelques mois plus tôt au sein de l'Université de Sherbrooke, par l'équipe de recherche composée de Louis-Simon Letourneau, Thibaud Ecarot, Marc Frappier et Pierre-Martin Tardif, au sein du laboratoire de cybersécurité. Ce travail est compilé au sein d'un article intitulé Data-Driven Comparison of Four Unsupervised Point-Anomaly Detection Methods [15]. Un état de l'art s'impose donc pour 2 raisons : premièrement, le travail déjà effectué doit être décrit afin de saisir la pertinence des modifications proposées. Deuxièmement, la littérature scientifique doit être explorée afin de saisir quelles solutions ont déjà été explorées ou non, au sein de ce domaine de recherche. Cette section commence donc par faire un tour d'horizon des différentes technologies et concept-clés nécessaires pour comprendre le travail réalisé par après. Enfin, la section se clotûre par une présentation en détail du travail de Letourneau.

2.1 Détection d'anomalies

La définition d'une anomalie varie en fonction du champ de recherche. Hawkins [13] propose une définition globale : 'Une anomalie est une observation qui dévie tellement des autres observations que l'on pourrait suspecter que celle-ci ait été générée par un autre mécanisme.' Cette définition suggère que derrière un ensemble de données observables se cache un modèle de génération classique de ces données, qui diffère d'un ou plusieurs autres modèles de génération mêlant leurs propres données observables 'anormales' au sein des données observables 'normales'. Ce sont ces derniers modèles que l'on cherche à identifier le plus précisément possible, sur base de l'ensemble des données observables, des connaissances préalables sur les modèles de comportement classiques, et de connaissances préalables sur les modèles de comportement anormaux.

Une première et grande partie du travail repose ainsi sur la distinction entre les données normales et anormales. L'étape suivante consiste en l'identification d'un modèle de génération de ces données normales, et ensuite de son application à des données non étudiées pour y détecter les déviations par rapport au comportement normal modélisé plus tôt.

Si cette tâche peut en apparence sembler simpliste ou binaire, elle amène toutefois avec elle de nombreux challenges, décrits par Chandola et al. [3] :

- La limite entre des éléments normaux et anormaux est parfois fine. Définir une région englobant parfaitement un comportement normal est dès lors difficile car les éléments anormaux se superposent parfois aux régions de

comportement normal.

- Lorsqu'une anomalie est le résultat d'un comportement malicieux, l'origine de l'anomalie a tout intérêt à adapter son comportement afin que l'événement paraisse normal.
- Dans de nombreux domaines, la définition de comportement normal n'est pas fixée dans le marbre et évolue continuellement. Un modèle de détection peut dès lors fonctionner un jour mais pas sur le long terme.
- Une déviation du comportement normal doit être interprétée différemment en fonction du domaine d'application. Par exemple, dans le milieu médical, une très faible variation de la température corporelle est un signal fort alors que le milieu de la gestion de stocks, de grosses variations peuvent ne rien signaler d'alarmant.
- La disponibilité de données labellisées, utiles pour l'entraînement des modèles, est loin d'être une garantie.
- Les données normales contiennent systématiquement une part de bruit, se mêlant aux anomalies et brouillant le travail de détection.

2.2 Machine learning

Le machine learning est la branche de l'informatique chargée de concevoir des modèles d'apprentissage automatisés spécialisés dans la réalisation d'une tâche. On différencie en général les tâches de classification (est-ce qu'une donnée appartient à une catégorie A ou B), de régression (quel serait vraisemblablement la valeur de cette donnée à un temps T+1) ou encore de clusterisation (quelles données semble former un groupe). Cette dernière catégorie implique naturellement une possibilité d'utilisation du machine learning dans le cadre de la détection d'anomalies. En effet, identifier quelles données semble former un groupe permet mathématiquement d'identifier quelles données semblent à l'inverse ne pas former de groupe, être isolées du comportement habituel des autres données. On identifie plusieurs types de machine learning menant à plusieurs approches de la détection d'anomalies. Goldstein [9] définit les 3 catégories suivantes :

— La détection d'anomalie supervisée fonctionne autour de datasets comprenant des données entièrement labellisées. Ce scénario fonctionne autour des mêmes règles que le machine learning supervisé classique à la différence que les classes s'avèrent le plus souvent extrêmement déséquilibrées, où les données étiquetées comme des anomalies sont en général en extrême minorité face aux données étiquetées 'normales'.

— La détection d'anomalies semi-supervisée, ou "One-class" classification [20] fonctionne autour de datasets 'nettoyés' de toutes données anormales. La

philosophie du modèle repose ainsi sur le fait d'apprendre la représentation d'un comportement classique pour identifier et signaler un comportement qui sort de ce schéma.

— La détection d'anomalies non-supervisée, enfin, fonctionne autour de datasets 'bruts', mêlant données normales et anomalies sans aucune labellisation. Le modèle, plutôt que proposer un classement net et binaire entre un événement normal et un événement anormal, fonctionne autour d'un système de scoring plus flexible, basé sur des modèles statistiques ou des métriques relatives à la densité. Ce système ne fonctionne donc pas autour de datasets de 'validation' ou de 'test' comme les modèles classiques. Dans la suite de ce document, par facilité, les termes de dataset 'test' ou de 'détection' seront parfois employés de manière équivalente.

Dans le cas du machine learning non-supervisé sous sa forme la plus pure, l'absence complète de labels pour identifier les données anormales en amont force à passer à la seconde étape en incluant les données anormales au sein de l'entrainement. Tous les modèles ne sont pas égaux face à cette inclusion. Certains modèles d'apprentissage disposent ainsi de mécanismes pour réduire l'impact des anomalies et du bruit sur leur entraînement, là où d'autres le réduisent par leurs simples règles de conception. Une autre approche est de supprimer le bruit au sein des données avant de fournir celles-ci au modèle [6].

2.3 Logs applicatifs

Un fichier de log se présente, de manière générique, comme une compilation d'événements générés automatiquement par un processus, généralement en background de ses actions normales. De manière plus intuitive, un fichier de log permet donc de retracer le plus fidèlement possible l'utilisation d'un logiciel, d'une fonction système ou plus généralement de tout système d'information. Un log système, par exemple, répertorie exhaustivement les ouvertures de session, la création d'un fichier ou encore l'exécution d'une commande, de manière à pouvoir se représenter le plus fidèlement, minute après minute, l'utilisation d'un programme ou d'un appareil donné. Les avantages de ce mécanisme sont nombreux, permettant par exemple de faciliter la correction de bugs, dans un monde où la taille des programmes augmente de façon continue et la logique qu'ils décrivent se complexifie sans cesse. Dans des cas plus extrêmes, lors d'une panne, un système de logs permet également de retracer le déroulé des événements précédant le problème, à la manière de la boîte noire d'un avion. Enfin, et c'est ce qui sera traité dans ce travail, les logs, en fonction du contexte, peuvent se révéler être une vraie mine d'or informative pour des entreprises qui prennent le temps de les accumuler. Grover[10] identifie cependant 3 challenges autour des logs applicatifs :

- Textes brut non structuré : Une partie des logs implémentés au sein des systèmes repose encore sur du texte brut pas ou peu structuré. Cela pose problème car cela empêche ou complexifie le traitement de ces données par d'autres programmes, en imposant un framework différent pour chaque application.
- Infomations redondantes sur l'exécution : il arrive fréquemment que des logs contiennent de nombreux éléments non pertinents pour une analyse. Des informations telles que les adresses ip des serveurs ou autres numéros de série peuvent alourdir les logs et compliquer leur traitement.
- Grands sets de données déséquilibrés : Par essence, un fichier de log se retrouve rapidement écrasé sous les données 'normales' et dispose d'une part infime de données d'intérêt pour les analyses. En plus d'alourdir le poids de ces fichiers sur les systèmes, cela allonge aussi le temps nécessaire pour les acteurs humains de traiter ces données, quand cela est nécessaire.

La standardisation de ces logs joue un rôle clé dans leur utilité et permet d'apporter une réponse à la première problématique. Si rien n'interdit concrètement une ligne de log d'indiquer les dernières actions d'un système d'information de manière non structurée, une ligne de log standardisée dès sa création facilite immédiatement son traitement, son analyse mais aussi sa potentielle migration. De nombreux formats de standardisation existent et sont décrits par Roudjane [25] :

- Fichier CSV
- Traces médicales HL7
- Fichier XML
- Fichier XES
- Fichier JSON
- Fichier CTF

2.4 Cybersécurité

Dans le domaine de la sécurité informatique, une anomalie n'a pas la même portée que dans un système de détection classique. Les enjeux tels que la protection de données confidentielles peuvent être vitaux pour une entreprise, voilà pourquoi la considération d'une anomalie dans les données doit être l'objet d'une politique assumée.

Patcha et Park [23] divisent les types d'intrusions en 4 catégories :

- Intrusif mais non-anormal : Ceux-ci sont des faux négatifs que le modèle échoue à détecter.

- Non-intrusif mais anormal : Ces événements sont des faux positifs que le modèle détecte à tort.
- Non-intrusif et non-anormal : Ces événements sont les vrais négatifs, ne cachant aucune activité intrusive et n'étant pas suspectés par le modèle de détection.
- Intrusif et anormal : Ceux-ci sont les vrais positifs, représentant une activité intrusive et détectés comme tels.

Un modèle de détection efficace est donc un modèle minimisant les deux premières catégories et maximisant les 2 dernières. Patcha et Park [23] indiquent que lorsque le nombre de faux négatifs doit être minimisé, le seuil doit être fixé à une valeur basse. Cela implique mathématiquement une augmentation du nombre de faux positifs, et demande donc plus de travail d'investigation à l'administrateur de sécurité. Deux politiques de traitement d'anomalies existent donc : un seuil haut réduisant le nombre d'événements à vérifier, ou bien un seuil bas réduisant le nombre de faux négatifs et donc la sécurité globale au prix d'une augmentation de la charge de travail. Ce choix appartient à l'entreprise en fonction des ressources dont elle dispose et du niveau de sécurité auquel elle aspire.

Bhuyan et al. [1] proposent un récapitulatif des différents types d'attaques au sein d'un système informatique. Ces catégories comprennent

- Virus
- Worm
- Trojan
- Dos
- Network Attack
- Physicial Attack
- Password Attack
- Information Gathering Attack
- User to Root Attack
- Remote to Local Attack

Parmi ces scénarios, nous nous intéressons particulièrement dans le contexte actuel aux Password Attack et aux Information Gathering Attack. C'est autour de ces 2 catégories que gravitent beaucoup des scénarios d'attaques entourant les audits de Sherweb. Premièrement, la création d'un accès illicite, par l'une des méthodes suivantes : Un 'man in the middle' mis en place au préalable peut tenter de récupérer les informations ou encore un token de connexion. La tactique

de la fatigue MFA permet de contourner les systèmes de double authentification en lançant simplement des demandes de connexion à répétition jusqu'à ce que l'utilisateur valide la connexion afin d'être débarrassé de la notification. Cette technique simple mais efficace, car utilisant le facteur humain, a provoqué l'émergence récente des systèmes de double authentification demandant de valider la connexion à l'aide d'un chiffre autre autre artefact à faire correspondre entre les 2 appareils. Enfin, une autre technique classique fonctionnant autour du facteur humain reste le phishing. Une fois les accès récupérés par l'attaquant, ce sont les scénarios d'Information Gathering Attack qui se mettent en place. Ainsi, l'attaquant a accès a un véritable éventail de techniques pour mettre à profit son accès à une session de l'organisation. Un premier réflexe peut être la création d'un accès supplémentaire, à l'aide par exemple d'une nouvelle session, pour sécuriser une porte d'entrée supplémentaire sur le système, si le mot de passe venait à être modifié. Une méthode moins discrète consiste à enlever à l'utilisateur légitime son accès à la session. Cependant, il est bon de noter que ces attaques ne sont souvent qu'une première étape d'un plan à plus long terme. Ainsi, le plus rentable pour un cybercriminel consiste le plus souvent non pas à accéder à une session et créer le plus de problèmes possibles, mais plutôt à se faire discret et cesser toute activité pendant une période plus ou moins longues. Cela rend son activité plus difficile à détecter et prévoir, et lui permet de mettre en place des mécanismes de récolte d'information. Ainsi, le hacker peut décider d'automatiser le forward de chaque mail reçu et envoyé par l'utilisateur, ou bien de consulter discrètement ses documents personnels. Ces informations sensibles et confidentielles peuvent ensuite être réutilisées pour un phishing de plus grande ampleur envers lequel l'entreprise en question pourrait s'avérer être bien moins méfiante.

2.5 Ensemble models

Les ensemble models, en français modèles ensemblistes, sont une philosophie de machine learning appliquant des principes collaboratifs à la prise de décision d'un modèle. Plus précisément, cette philosophie fait reposer une tâche de classification, de clusterisation, de régression, non pas sur un modèle de machine learning spécialisé le plus finement pour une tâche précise, mais sur plusieurs sous-modèles, appelés modèles de base, qui apportent chacun leur réponse propre au problème. Ces réponses individuelles sont ensuite combinées par la méthode ensembliste pour en dériver une réponse finale. On observe une relative sous-documentation à propos de ce sujet au sein de la littérature, comparativement aux modèles de machine learning classiques, dits "unitaires". Il semble donc pertinent de proposer un tour d'horizon de la technologie.

Oza[21] divise ces modèles selon 2 aspects clés différents :

Le premier aspect-clé est le mode d'entraînement des sous-modèles. Ceux-ci peuvent être entraînés à l'aide d'un ensemble de données uni, ou bénéficier chacun d'un training dataset propre. Ce deuxième mode d'entraînement promeut une plus grande diversité dans l'entraînement en donnant volontairement à chaque modèle un set de données d'entraînement différent pondéré de manière unique. Une philosophie inverse est de justement promouvoir la plus grande cohésion possible entre les modèles en leur fournissant des sets de données identiques.

Le second aspect clé est la méthode de collaboration entre les sous-modèles. A la manière d'une élection politique, un ensemble de modèles donnant chacun la même réponse peut aboutir à une décision finale différente en fonction des règles mises en place. Le vote à la majorité pure est un mode de fonctionnement souvent utilisé. Au dessus de tout ça, il est également possible de pondérer chaque sous-méthode en fonction d'une mesure de la fiabilité de leurs résultats. Ainsi, la décision d'un sous-modèle A peut moins peser dans la balance qu'un sous-modèle B. C'est un mécanisme qui est employé par la méthode des 'Mixtures of Experts' [14]. Cette méthode fonctionne autour d'un modèle supplémentaire entraîné préalablement qui pondère dynamiquement l'impact de chaque sous-modèle, en fonction des performances de chacun d'entre eux sur les données d'entraînement. Enfin, le système de Borda Count propose ses propres avantages, en ne demandant pas à chaque sous-méthode son avis sur une seule donnée, mais en leur donnant plutôt un set de données à classer [7]. La décision finale se fait ainsi en combinant le classement de chaque sous-modèle à l'aide d'un algorithme de décision au choix. Ces algorithmes sont illustrés au sein des figures 1, 2 et 3.

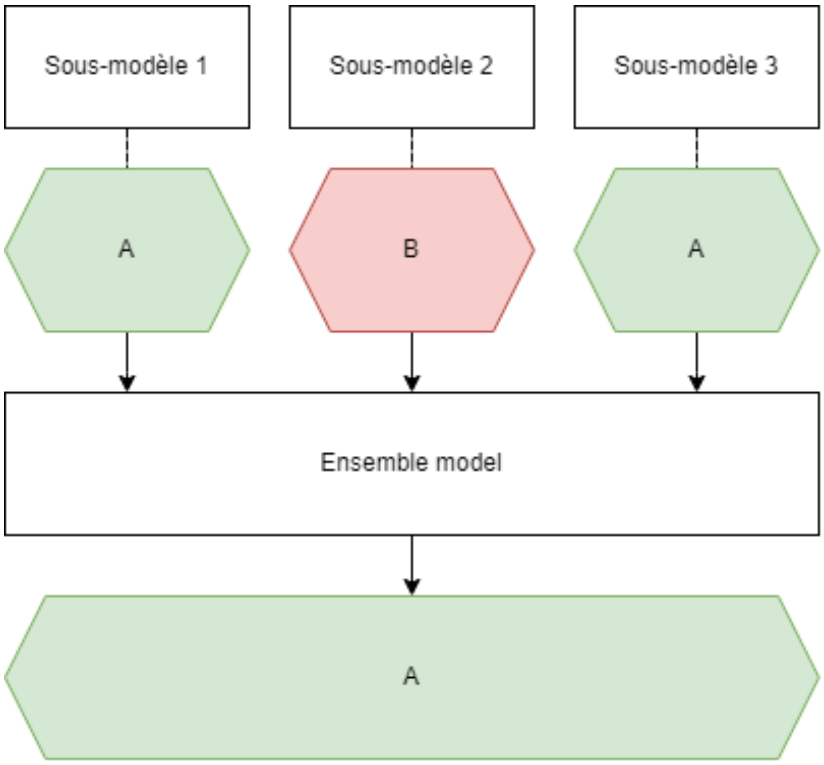


FIGURE 1 – Illustration d'une décision à la majorité au sein d'un ensemble model

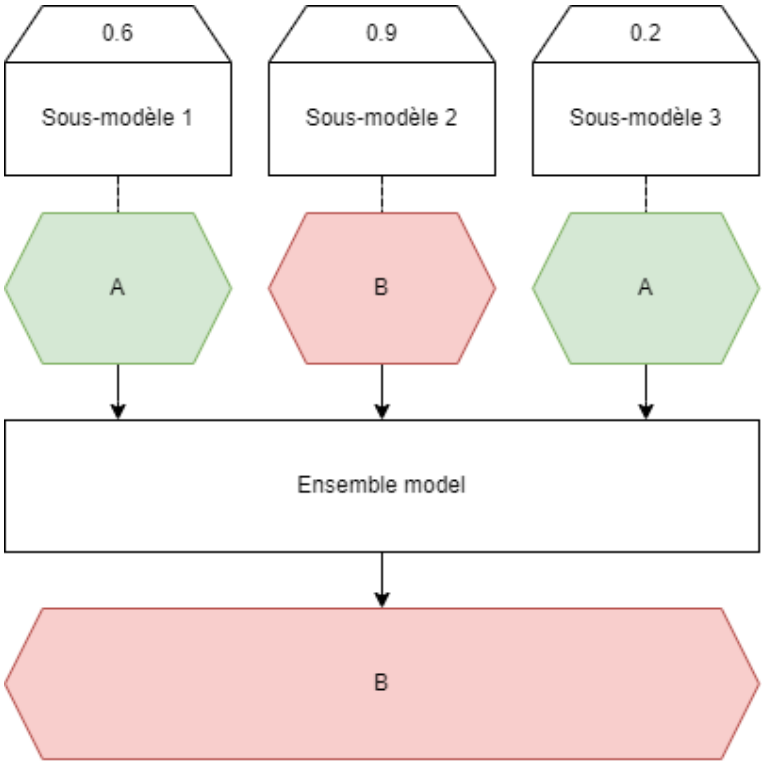


FIGURE 2 – Illustration d'une décision par pondération au sein d'un ensemble model

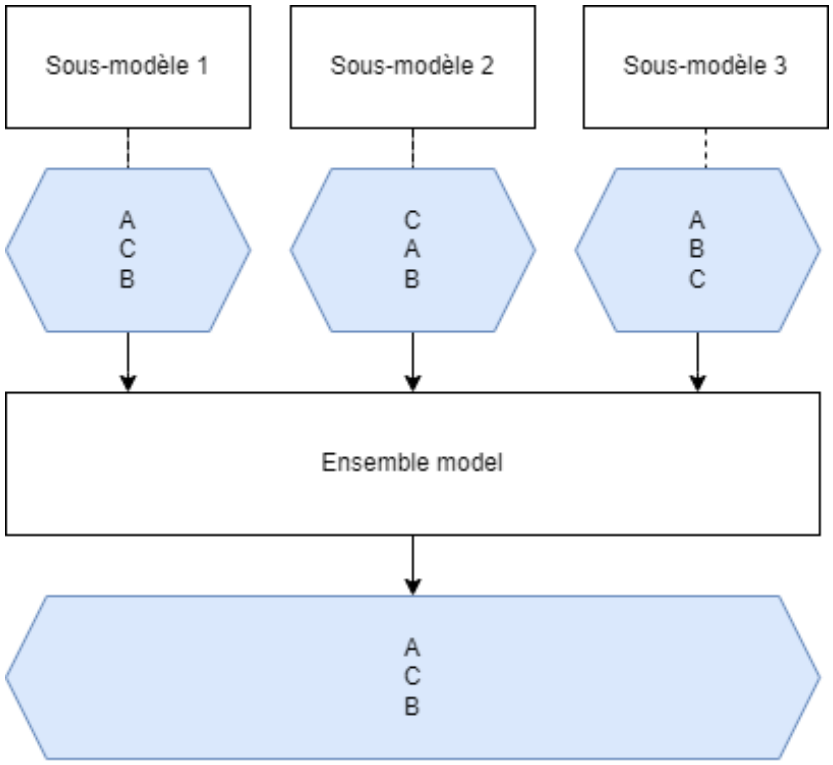


FIGURE 3 – Illustration d'une classement par borda count au sein d'un ensemble model, entre les données A, B et C

On peut ajouter à ces 2 aspects clés la sélection de sous-méthode. Là aussi, beaucoup de paramètres entrent en compte, du nombre de méthodes sélectionnées jusqu'au type d'algorithmes sélectionnés. Il est ainsi possible de sélectionner des algorithmes extrêmement éloignés tout comme sélectionner des algorithmes proches permet de capter des patterns plus précis. Il est également possible de réutiliser plusieurs fois le même algorithme avec des paramètres différents. C'est une philosophie qu'emprunte par exemple l'Isolation Forest[16].

Finalement, les avantages d'une ensemble method sont les suivants :

- Ce modèle de machine learning offre la possibilité de déceler des patterns plus fins, en traitant à l'aide d'algorithmes indépendants des patterns de génération indépendants au sein des données.
- Le caractère fondamentalement plus modulaire de cette technologie rend les implémentations de modèles plus flexibles. Il est ainsi possible de supprimer, ajouter ou changer les paramètres d'un modèle sans influencer les performances des autres. Cela rend les modèles ensemblistes plus fiables sur le long terme.

Inversément, cette philosophie apporte avec elle quelques contraintes à prendre en compte :

- La paramétrisation de la solution globalement plus complexe, car composée de plusieurs modèles devant chacun être paramétrés individuellement, avec idéalement autant d'intelligence qu'un modèle unitaire.
- Le modèle impose des sacrifices en terme de temps de calcul, comparativement à un modèle classique.

2.6 Évaluation

L'évaluation des résultats est une problématique récurrente dans le domaine du non supervisé. Goix explore dans cet article [8] différentes métriques d'évaluation pour valider les résultats d'un dataset contenant peu ou pas de labels. Les critères sont basées sur les courbes d'Excess-Mass et de Mass-Volume. Celles-ci perdent malheureusement en intérêt sur un nombre large de dimensions. Une méthode est cependant proposée pour réduire le scaling de ces critères sur les dimensions, par le biais du sub-sampling des features. En sélectionnant aléatoirement un nombre limité de features pour réduire l'analyse, et répétant l'opération à plusieurs reprises, et en effectuant une moyenne sur les résultats obtenus, il est ainsi possible de garder l'information de chaque feature tout en réduisant les calculs nécessaires et rendant la méthode utilisable sur des datasets à plus haute dimension.

2.7 Présentation de l'article

Le contexte scientifique étant posé, le point d'entrée de l'étude sera présenté au sein de cette section. La globalité du travail est décrite ici tandis que certains aspects sont explicités plus plus en profondeur au sein de la Section 3. Dans son document[15], Letourneau propose un modèle de détection d'anomalies destiné à l'usage industriel. Les données utilisées pour développer, entraîner et tester la solution proviennent de l'entreprise Sherweb et sont groupées par semaine. Ces données correspondent à des logs applicatifs de solutions Microsoft professionnelles, pour une entreprise donnée. Les données ne possèdent pas de labels et sont anonymisées au préalable, puis compilées sous forme de fichiers CSV. La solution développée propose ainsi de filtrer la totalité des événements relatifs à chaque utilisateur afin d'en isoler uniquement un nombre réduit d'événements suspects. Cette sélection peut ensuite être proposée à un expert chargé d'y identifier des événements problématiques, d'un point de vue sécurité. Les intrusions sont les principaux événements visés par la solution. Les données qui ne sont pas retenues par le modèle sont mises de côté en tant 'qu'événements normaux'. La détection des anomalies est donc réalisée à l'aide d'un modèle de machine learning non supervisé, sur un système de rolling window. Ainsi, un modèle est entraîné hebdomadairement, sur base du comportement de la semaine précédente, afin d'analyser

les données de la semaine suivante. Le fonctionnement schématisé de la solution est illustré dans la figure 4.

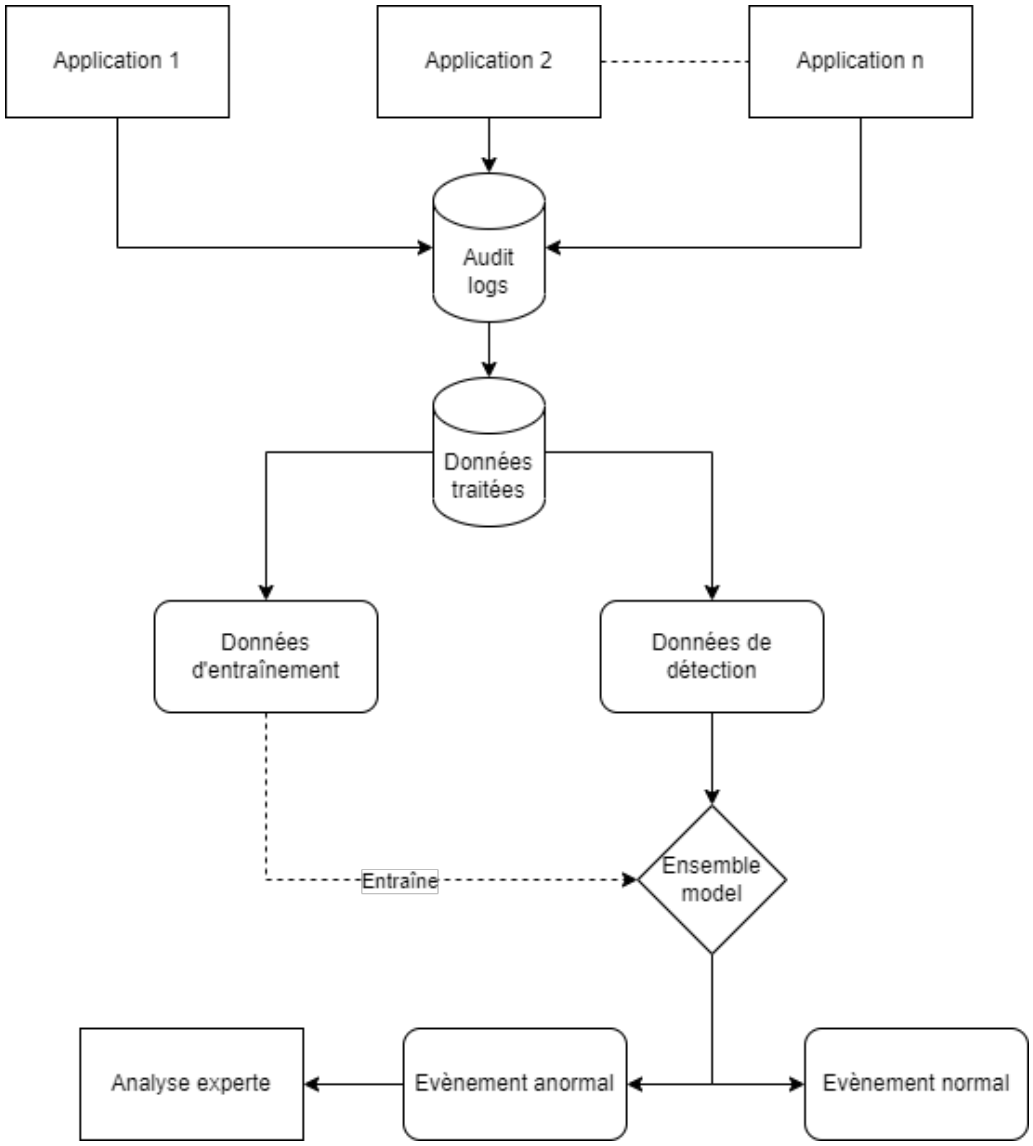


FIGURE 4 – Fonctionnement schématisé du modèle à haute échelle tel que proposé par Letourneau

En guise de modèle de détection, l'article fait donc le choix de partir sur un ensemble model. Celui-ci fonctionne autour de 4 méthodes non supervisées : k-Neighbors, k-Means, Local Outlier Factor et Kernel Density Estimation. Ces méthodes et leurs atouts propres sont explicités dans une section dédiée. Ainsi, l'algorithme crée une copie de chacun de ces modèles pour chaque utilisateur unique. L'avantage de cette méthode est qu'elle permet de représenter le plus fidèlement possible le comportement de chaque utilisateur, ce qui est nécessaire vu le type d'utilisateurs très différents présents au sein des donneés. Chaque modèle créé, après avoir été entraîné sur un set de données d'entraînement, devient ainsi capable d'associer un score dit "d'anomalie" à chaque événement qui lui est soumis.

Les 20 événements au plus haut score d'anomalies sont identifiés en tant qu'événement "flaggé". Enfin, il est possible d'ordonner ces événements flaggés entre eux sur base du nombre de méthodes l'ayant identifié. Un événement avec un score d'anomalie bas détecté par 3 méthodes peut ainsi être considéré comme plus suspect qu'un événement détecté par une seule des 4 méthodes, quand bien même son score d'anomalie est haut. Cette méthode de vote peut ainsi être assimilée à une version dérivée du système de borda count. Le fonctionnement de cette ensemble method est illustré dans la figure 5.

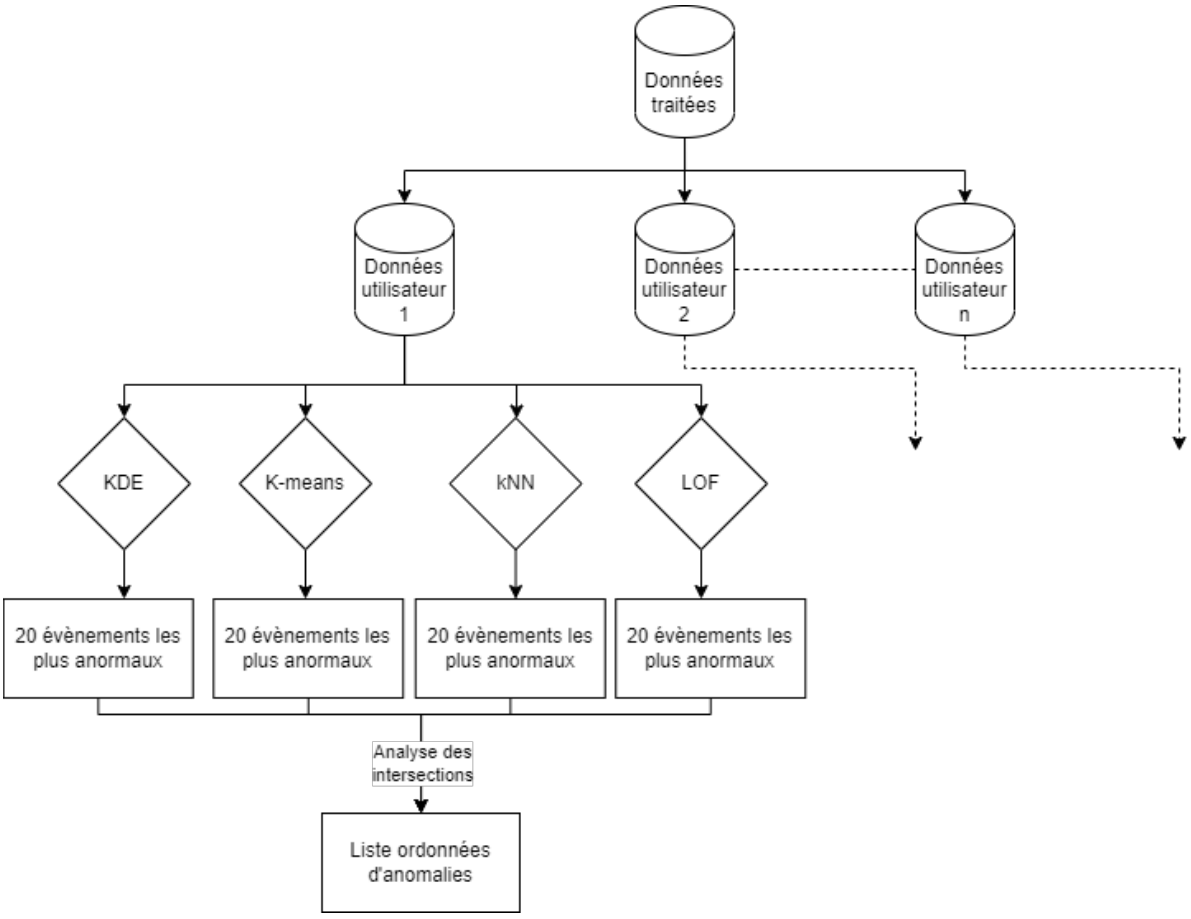


FIGURE 5 – Fonctionnement schématisé de l'ensemble model tel que proposé par Letourneau

2.8 Seuil

Le seuil, ou threshold est défini classiquement comme une valeur seuil fixant une limite entre deux catégories d'éléments. Cette notion est d'autant plus importante en apprentissage non-supervisé qu'elle est au coeur de la plupart des mécaniques de clusterisation. Le seuil défini ici concerne la part d'événements traités comme des anomalies au sein des événements "normaux". Celui-ci peut être exprimé sous forme d'un nombre fixe d'événements, ou bien d'un pourcentage d'événements par rapport aux événements complets. L'article original fait le choix du nombre fixe

de 20 événements par méthode, qui sera maintenu dans ce travail. La justification derrière ce choix est la réalité industrielle, qui limite le nombre d'événements pouvant être étudiés par un seul expert. Ainsi, peu importe la taille du dataset, une unique entreprise n'aura le temps et les ressources que pour traiter un nombre fixe d'anomalies par mois, et non un pourcentage. Il reste bon de noter que dans les deux cas, le seuil reste fondamentalement mobile et la valeur exacte à laquelle il doit être fixé dépend entièrement des ressources qu'une entreprise est prête à allouer ou non au traitement des anomalies de sécurité.

Passage d'un modèle linéaire à un modèle cyclique

L'article original propose une analyse univariable basée sur les timestamps des événements, converti en nombre de minutes depuis le début de la journée. Le défaut de cette valeur sous sa forme de base est qu'elle échoue à créer une proximité entre la première et dernière minute de la journée. Une transformation de cette variable vers un modèle cyclique et ainsi proposée et implémentée afin de résoudre ce problème. Ainsi, plutôt que de situer les événements de manière linéaire entre un point de départ 00h00 (0 minutes) et un point d'arrivée de 23h59 (1440 minutes), celui-ci place les événements sur une "roue" virtuelle trigonométrique, à l'aide d'un couple de valeurs comprises entre -1 et 1. Ces différentes représentations sont illustrées au sein des figures 6 et 7. Il est important de comprendre que cette modification n'est appliquée qu'au cours de l'exécution du programme et ne modifie pas le contenu même des datasets.

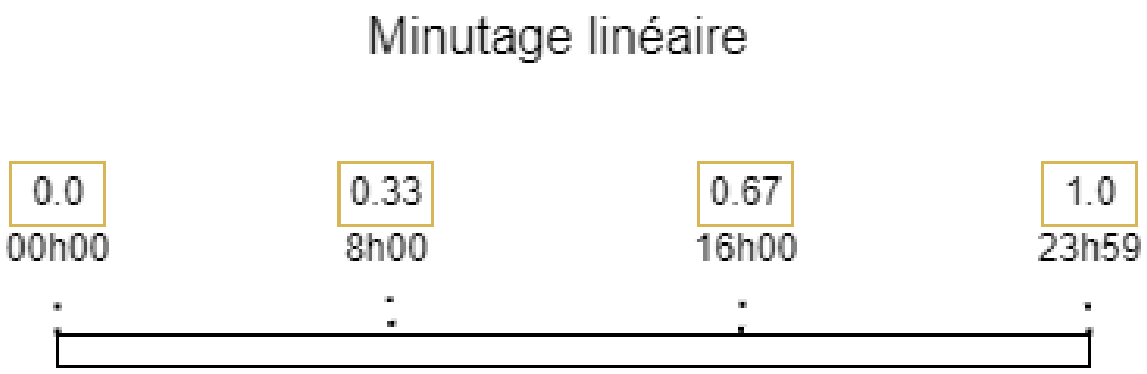


FIGURE 6 – Représentation linéaire des données sur une échelle de 0 à 1, cadrées en jaune à côté de leur heure associée

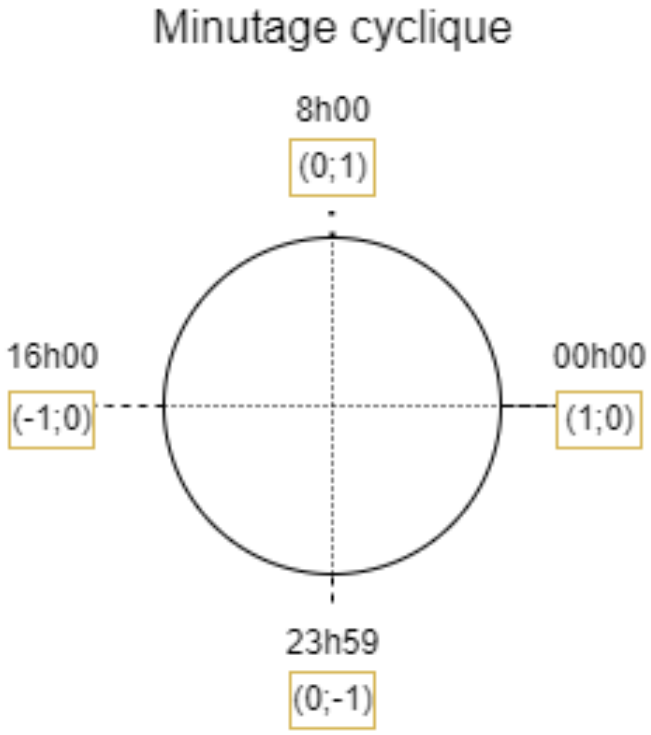


FIGURE 7 – Représentation cyclique des données, cadrées en jaune à côté de leur heure associée

3 Développement de la recherche

L'objet de cette section est de décrire les différentes étapes et tâches structurant le travail de recherche durant tout son déroulé. La structure suit le cheminement haut niveau des problèmes et solutions et n'intègre pas la majeure partie des écarts et fausses pistes empruntées lors du processus de recherche.

3.1 Méthodologie

Poursuivre un projet déjà entamé et prenant racine dans la réalité industrielle offre un contexte de recherche hors du commun, balisé de manière différente d'un projet de recherche classique. Voici dès lors les étapes jugées nécessaires pour le bon déroulé du projet.

Dans un premier temps, une évaluation la plus exhaustive du projet déjà en place s'impose pour identifier des points d'amélioration possibles. Cette évaluation s'articule principalement autour de l'article scientifique Data-Driven Comparison of Four Unsupervised Point-Anomaly Detection Methods. Sa méthodologie, ses interprétations et autres choix sont revus sous un oeil critique. Conjointement, le code du projet est passé en revue pour être modifié et optimisé quand cela s'avère nécessaire.

Après cette étape d'évaluation, plusieurs axes d'amélioration du projet plus ou moins indépendants sont identifiés et explorés les uns après les autres de manière concurrente. L'objectif de chacun de ces axes est d'identifier une modification permettant d'atteindre des résultats considérés comme satisfaisants, selon un critère rigoureux lorsque le contexte l'autorise.

3.2 Évaluation de l'article scientifique

3.2.1 Haute variance au sein des résultats

On observe premièrement des variances anormales au sein de résultats renvoyés par les algorithmes. Les scores associés aux événements par LOF, particulièrement, proposent des écarts très élevés, au delà des échelles de score normales associées par l'algorithme, illustrées sur la figure 8, là où les autres algorithmes renvoient des résultats habituels.

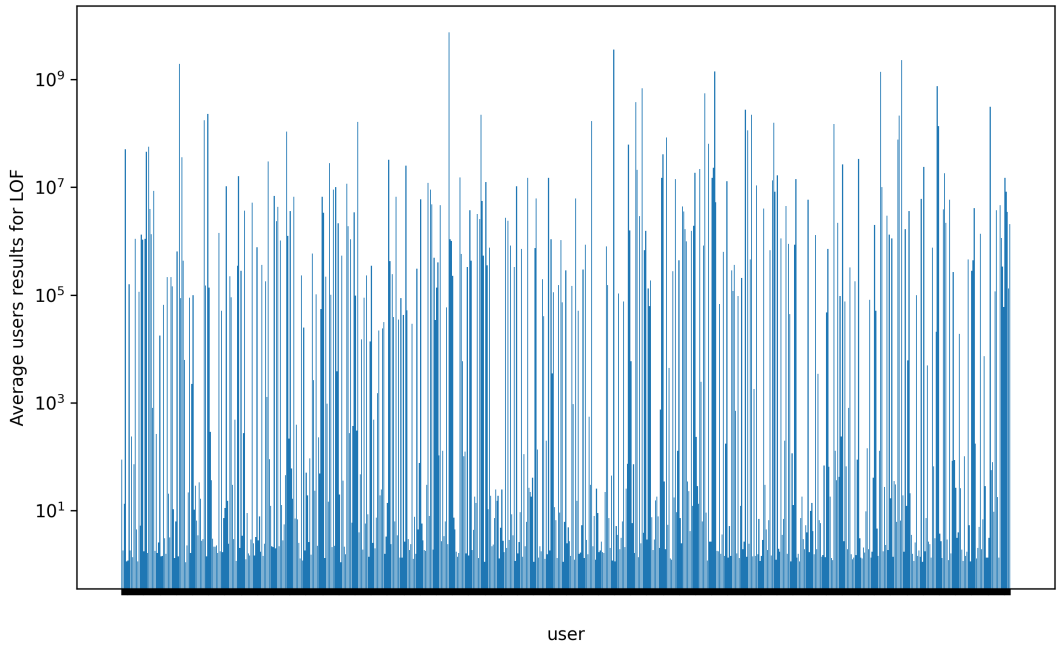


FIGURE 8 – Visualisation du score moyen attribué aux 20 anomalies détectées par l'algorithme LOF, par utilisateur

On observe ainsi régulièrement des valeurs atteignant le milliard, anormalement hautes pour le modèle LOF. Breunig[2] indique dans sa présentation du modèle des scores compris entre 0 et 1.0 pour des événements normaux, et dépassant 1.0 pour des événements anormaux. Cette problématique se présente comme le premier point de départ de l'amélioration du modèle.

3.2.2 Cohésion limitée des sous-modèles

Le second élément ressortant d'une analyse du document repose sur la cohésion des modèles proposés. Un des résultats les plus mis en évidence dans l'article de Letourneau est l'intersection des anomalies détectées par chaque algorithme de détection.

Une mesure moyenne du nombre d'anomalies détectées au sein des différentes intersections est par exemple proposée, reprise dans la figure 9. Ces métriques quantifient à quel point les méthodes suspectent les mêmes événements, pour un utilisateur donné, et sont utilisées comme critère de validation. Le calcul de chaque colonne est effectué en additionnant le nombre d'évènements flaggés détectés par respectivement 1 méthode, 2 méthodes, 3 méthodes et 4 méthodes. Ainsi, si aucune des méthodes ne détecte la moindre anomalie en commun avec les autres sous-méthodes, le tableau correspond aux valeurs 80, 0, 0, 0. De la même manière, si les 4 méthodes détectaient le même exact set de 20 anomalies, le tableau correspond

à 0, 0, 0, 20. Dans ces 2 cas, la plus value du modèle s'avérerait très limitée. Malgré tout, les chiffres proposés dans la figure 9 semblant laisser une fenêtre d'amélioration raisonnable. Ainsi, le choix est fait de chercher à optimiser ces métriques. L'idée est ainsi de maximiser les trois dernières valeurs du tableaux, et de minimiser la première.

	$\sum S_1$	$\sum S_2$	$\sum S_3$	S_4
Sum	35	12	4.2	1.7

FIGURE 9 – Somme des anomalies par groupe de technique, dans l'article original

3.3 Diminution de la variance

3.3.1 Exploration des données

Un axe logiquement envisagé pour identifier l'origine de la haute variance au sein des données est celui de l'exploration des données. Il semble intuitivement primordial de comprendre le comportement des données pour anticiper au mieux les réactions des différents algorithmes. La définition d'une anomalie changeant d'un algorithme à l'autre, il est nécessaire de comprendre à quoi ressemble une anomalie sous un oeil humain. Pour étudier ces données, plusieurs outils d'analyse sont présents au sein du code. Nous les divisons en 2 catégories : outils d'analyse statistique et outils d'analyse graphique.

Les outils d'analyse statistiques sont chargés de résumer la qualité des données ou des résultats sous une forme numérique facile à comparer. Parmi ces mesures on retrouve un utilitaire de moyenne de scores attribués par une méthode, sur l'entièreté des événements d'une semaine. Un autre utilitaire se charge lui, pour chaque méthode, de définir le nombre d'événements dont le critère de cohésion est égal à 1. Ce critère de cohésion sera explicité dans une section suivante. Les fonctions statistiques créées dans le cadre du projet sont adaptées pour une analyse globale, ou bien une analyse par utilisateur.

Ensuite, les outils d'analyse graphique, plus largement utilisés au sein du projet car efficaces pour l'identification de tendances et patterns cycliques, incluent diagrammes de VENN (ou diagrammes logiques), histogrammes et pie charts. Ceux-ci sont présentés plus en détail au fur et à mesure de cette section.

3.3.2 Profils d'utilisateurs

Un résultat qui ressort rapidement de cette étude des données est la présence de différents types de profils au sein des données, ce qui n'est pas abordé par Letourneau. Ainsi, en modélisant individuellement le comportement des utilisateurs sur une semaine, on identifie rapidement plusieurs patterns au sein des visualisations.

- Les utilisateurs dits "En plateaux" illustré sur la figure 10. Ces utilisateurs s'illustrent par de larges blocs d'événements très réguliers, ainsi qu'à des périodes dénuées d'événements, généralement associées aux heures nocturnes, mais pas forcément. On associe ces comportements à du matériel purement automatisé, fonctionnant potentiellement en background, input matériel. Une grande partie de ces utilisateurs est par exemple associé à des événements de type "MailboxLogin".
- Les utilisateurs dits "En pic" illustrés sur la figure 11. Ces utilisateurs s'illus-

trent par un comportement globalement calme, parsemé de pics d'événements très concentrés, en l'espace de quelques minutes, une ou deux fois par jour. Ces pics sont en général causés par des événements FileSyncUploadedFull, correspondant visiblement à des synchronisations de service cloud.

- Les utilisateurs dits "Grésillants" illustrés sur la figure 12. Ces utilisateurs se présentent comme une version alternative des utilisateurs "en plateaux", avec un nombre d'événements par minute moins stable, alternant en général entre 2 ou 3 valeurs proches. Ce comportement, tout comme les utilisateurs en plateaux, s'explique probablement par l'usage de scripts, qui créent des événements, non pas chaque minute mais sur un rythme proche, créant ainsi un pattern cyclique malgré une variation de minute en minute.
- Les utilisateurs dits "Gaussiens" illustrés sur la figure 13. Ces utilisateurs sont représentés par un nombre d'événements variant de minute en minute, suivant une courbe gaussienne et composée d'une grande partie de bruits. On observe également une très grande baisse d'activité de ces utilisateurs aux heures les plus nocturnes. Tous ces éléments portent à croire que ce type d'utilisateur correspond aux machines employées par les véritables utilisateurs humains de l'entreprise.

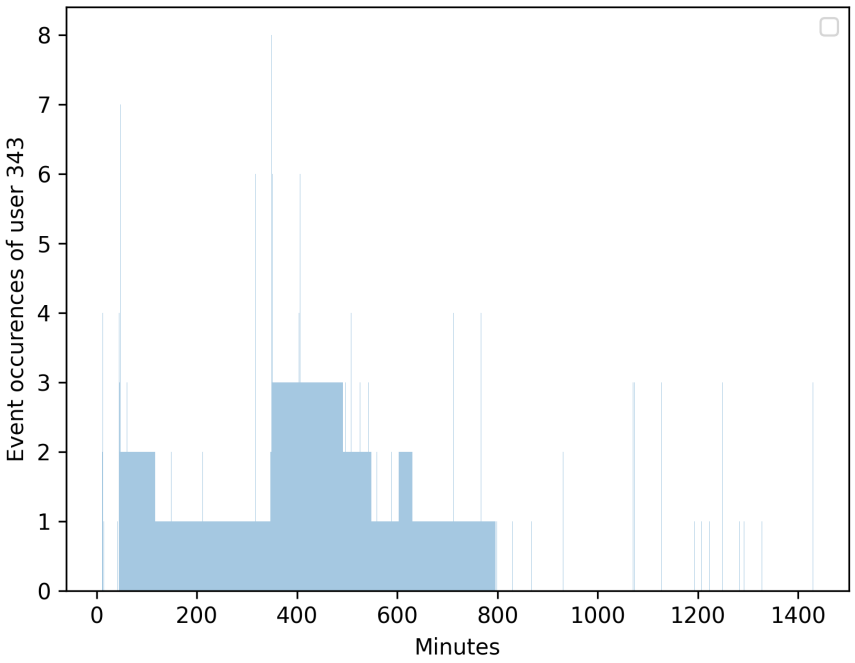


FIGURE 10 – Visualisation du nombre d'activités d'un utilisateur dit 'En plateau' sur une semaine, par minute de la journée

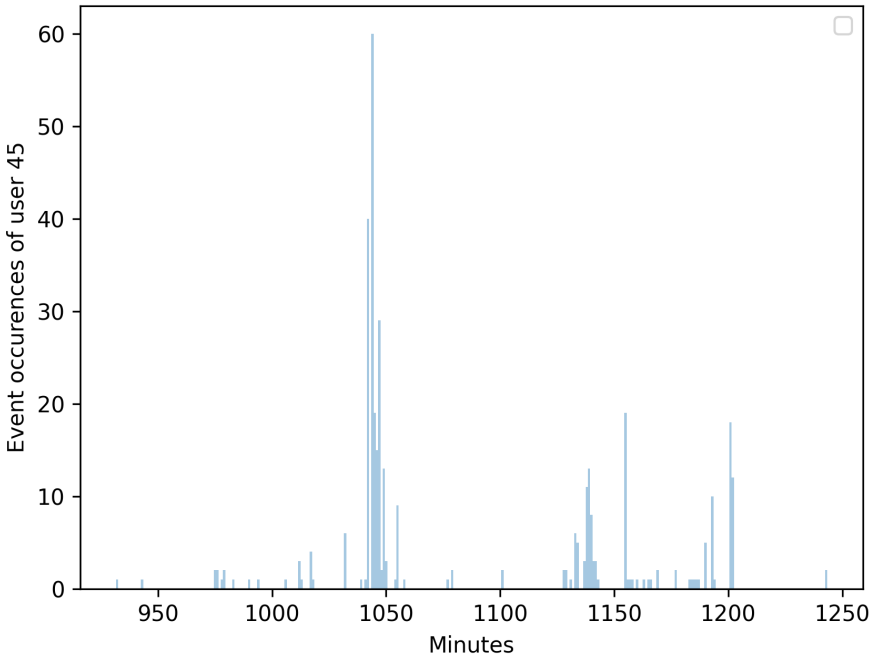


FIGURE 11 – Visualisation du nombre d'activités d'un utilisateur dit 'En pic' sur une semaine, par minute de la journée

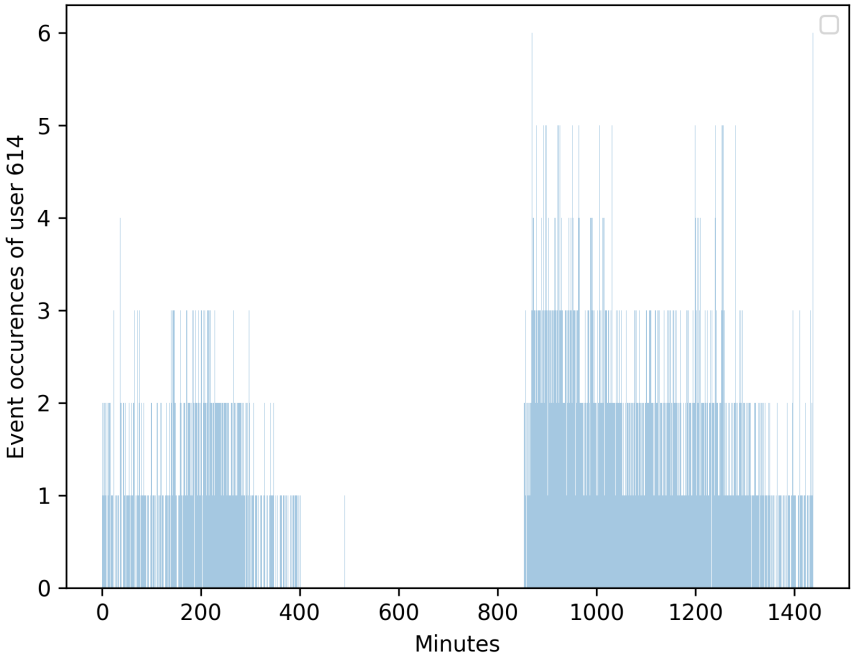


FIGURE 12 – Visualisation du nombre d'activités d'un utilisateur dit 'Grésillant' sur une semaine, par minute de la journée

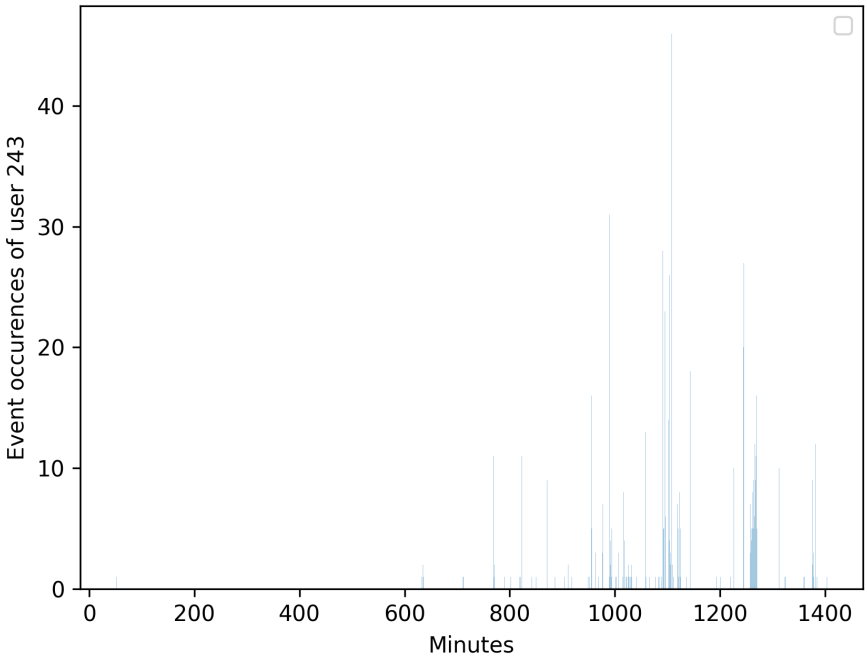


FIGURE 13 – Visualisation du nombre d'activités d'un utilisateur dit 'Gaussien' sur une semaine, par minute de la journée

On note malgré tout une part d'utilisateurs combinant les caractéristiques de plusieurs de ces catégories.

L'identification de ces différents patterns d'utilisateurs, s'il n'est pas réutilisé par la suite, permet cependant de se rendre compte que des types d'utilisateurs radicalement différents sont présents au sein du système. La question se posant donc naturellement est la suivante : les utilisateurs se distinguent-ils donc d'une autre manière, au sein des données, ou gardent-t-il un comportement très proche, au delà de leur simple rythme de création d'événement ?

3.3.3 Intégration de features

Une ligne d'audit Microsoft telle qu'étudiée dans le cadre de l'ensemble model est structurée sous la forme suivante :

Attribut	Description
id	Représente l'identifiant de la ligne
ContentType	Identifie à quel outil est relié la commande (Sharepoint, Exchange, AzurectiveDirectory,...).
Event.CreationTime	Date précisément l'événement, à la seconde près.
Event.Operation	Indique l'action enregistrée par le log. (Un fichier consulté, un dossier renommé, une synchronisation avec le serveur, ...).
Event.UserType	Indique le type de l'utilisateur responsable de l'événement.
Event.ResultStatus	Indique si l'événement enregistré a réussi ou échoué.
TenantId.Seq	Identifie l'abonnement auquel est associé l'utilisateur.
Event.UserKey.Seq	Indique à quel groupe d'utilisateurs appartient l'utilisateur responsable de l'événement.
Event.ObjectId.Seq	Identifie l'objet concerné par l'événement (un dossier, un fichier, un identifiant,...)
Event.UserId.Seq	Identifie l'utilisateur responsable de l'événement.
Event.ClientId.Seq	Identifie le groupe réseau auquel est associé l'utilisateur.

L'article de Letourneau fait le choix de limiter l'entièreté de l'analyse à la colonne 'Event.CreationTime' de ces données, plus précisément à la portion HH :MM de ces données. Ainsi, un élément n'est traité uniquement qu'en regard de sa minute d'occurence, sur les 1440 minutes qui composent chaque journée. Cet aspect des données est très intéressant pour modéliser au mieux le comportement quotidien d'un utilisateur, ses heures d'utilisation, mais surtout d'absence habituelles. L'idée est ainsi de considérer comme un événement suspect un événement qui aurait lieu en dehors de ces patterns habituels d'utilisation. Une connexion à 2h du matin sur un ordinateur de travail utilisé habituellement de 9h30 à 17h pourrait ainsi être assimilée à un intrus tentant d'utiliser l'appareil pour accéder à des données sensibles. Il est bon de noter que cette contrainte est avant tout posée par une demande explicite du partenaire industriel. Si cette simplification vers un modèle univariable ne pose pas de problème d'un point de vue de mise en place des modèles et de l'analyse des résultats, elle empêche malgré tout au modèle de comprendre pleinement les différences concrètes entre l'un et l'autre utilisateur. Comme nous l'avons vu, un utilisateur automatisé et un utilisateur humain peuvent très bien agir aux mêmes heures pour réaliser des actions complètement

différentes.

Les features suivantes sont ainsi envisagées pour une potentielle intégration au sein du modèle :

3.3.4 Jour de la semaine

Le jour de la semaine, de lundi jusqu'à dimanche peut être facilement extrait de la colonne 'Event.CreationTime' des données et être ajouté aux vecteurs de données considérés par les différents modèles. Ces audits appartenant primairement à des utilisateurs professionnels et particuliers des outils Microsoft, il semble intuitif de considérer que leur comportement vis à vis de ces outils varie d'un jour à l'autre de la semaine. L'exemple de l'utilisateur n'employant son ordinateur uniquement que du lundi au vendredi et passant son week-end loin de son appareil vient rapidement en tête. Ainsi, une action enregistrée par le système, même au coeur des heures de connexion habituelles de l'utilisateur, pourrait d'un seul coup devenir sensiblement suspecte si réalisée lors d'un de ses jours habituels d'absence.

3.3.5 Type d'événement

Le type d'événement, représenté par l'attribut 'Event.Operation' représente le type d'action auquel se rattache l'événement, au sens de l'audit Microsoft. Le nombre de labels différents est élevé et se compte en dizaines, allant d'actions intuitives comme 'FolderCreated', informant de la création d'un dossier, jusqu'à des actions plus complexes telles que 'FileSyncUploadedFull'. On devine vite l'intérêt que cette information pourrait avoir pour détecter les comportements les plus suspects ou dangereux. Ainsi, si renommer un dossier une fois la nuit tombée semble légèrement suspect mais avec des enjeux limités, le même événement mais marqué de 'UserLoginFailed' peut sonner comme une vraie alerte de sécurité. Enfin, comme nous l'avons vu, cet attribut définit presque à part entière le comportement de certains utilisateurs. Ceci se vérifie facilement à l'aide des figures 14 et 15. On y observe ainsi qu'un utilisateur ne navigue en général que sur un nombre très limité d'actions différentes. Certains utilisateurs, probablement automatisés, tels que celui illustré sur la figure 15 ne réalisent quant à eux qu'un seul type d'action sur l'espace d'une semaine. Dans ce contexte, la valeur d'un attribut 'Event.Operation' semble indiscutable, car elle permettrait d'identifier immédiatement un événement comme suspect s'il venait à avoir un type différent du pattern habituel de l'utilisateur. Il reste intéressant de noter que la colonne 'Event.Operation' contient des données d'importance très variable et reste écrasée sous des actions sans grand impact, ce qui doit être pris en compte lors de l'implémentation. Le choix est malgré tout fait d'ajouter cette feature à l'implémentation des modèles compte tenu de la valeur qu'elle représente.

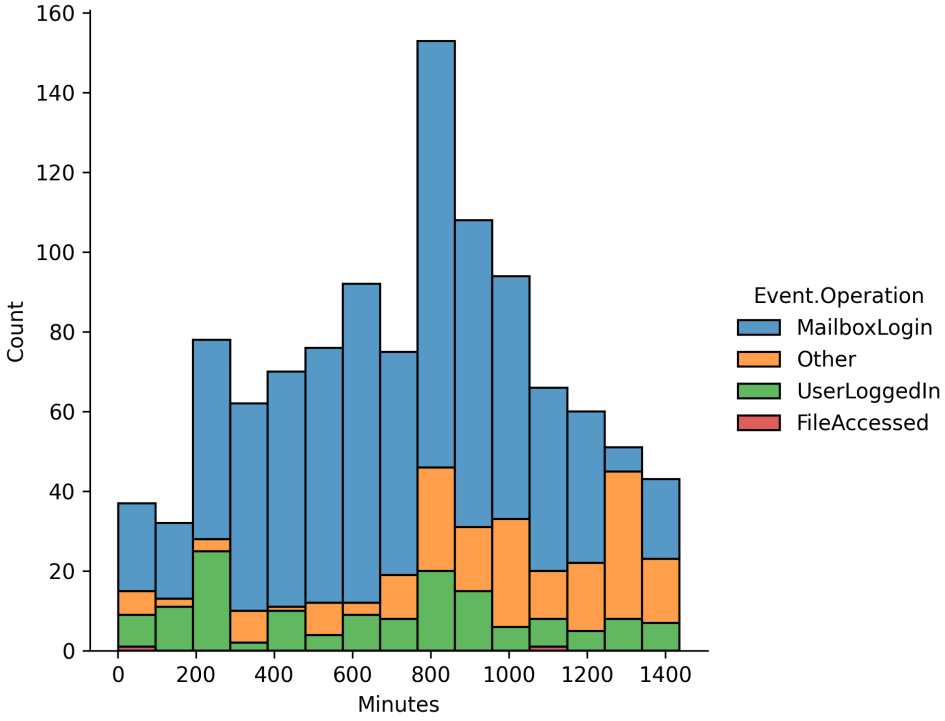


FIGURE 14 – Répartition des événements d'un utilisateur le long d'une semaine, en fonction des minutes de la journée

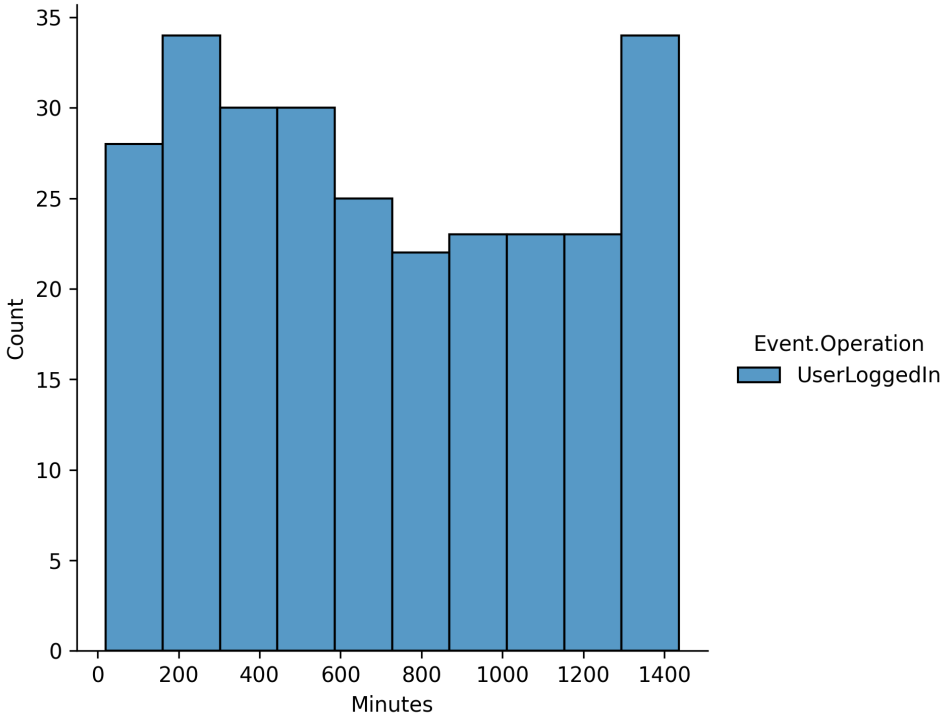


FIGURE 15 – Répartition des événements d'un utilisateur le long d'une semaine, en fonction des minutes de la journée

3.3.6 Encodage

L'attribut 'Event.Operations' étant représenté à l'aide de labels nominaux, il est nécessaire de procéder à un encodage afin de l'intégrer au modèle. La méthode d'encodage sélectionnée pour la feature est le One-Hot encoding. Les labels ne disposant pas de notion d'ordonnancement entre eux, ce mode d'encodage empêche ainsi tout rapprochement d'un attribut à l'autre de la part du modèle. Comme présenté précédemment, le One-hot encoding perd grandement en efficacité face à un nombre élevé de labels. Ici, l'attribut présente plus d'une centaine de labels, ce qui nous pousse à sélectionner un nombre plus réduit de labels. Un pie chart est généré afin d'aider cette sélection, présenté au sein de la figure 16.

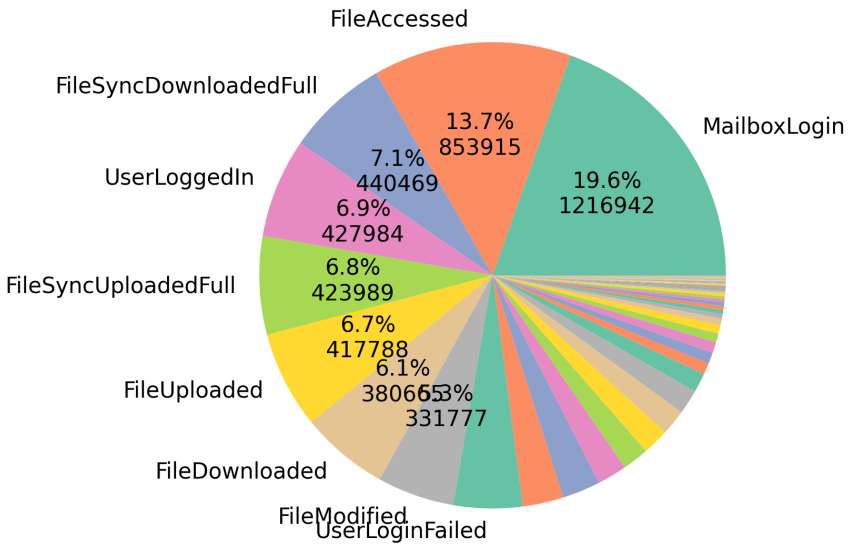


FIGURE 16 – Distribution des labels de 'Event.Operations' sur l'entièreté des événements

Il en ressort assez vite que la distribution des labels est loin d'être égale au sein de tous les événements. Le label 'MailboxLogin' décrit par exemple à lui seul près de 20% des événements présents dans le dataset. D'autres labels possèdent des répartitions comparables, tels que 'FileAccessed' ou 'FileSyncDownloadedFull'. Ainsi, le choix est fait de réduire la sélection totale des labels à un échantillon composé des 6 labels les plus présents, ainsi qu'à un label 'Other', englobant l'entièreté des autres événements. Les labels extraits sont donc les suivants :

- MailboxLogin
- FileAccessed

- FileSyncDownloadedFull
- UserLoggedIin
- FileSyncUploadedFull
- FileDownloaded
- Other

Il est bon de noter que cette simple sélection des 6 premiers labels suffit à prendre en compte plus de 60% des événements du dataset. L'application de ce filtre permet donc la génération de la figure 17, ainsi qu'à de nombreux graphes individuels tels que la figure 14 qui permettent de valider intuitivement la pertinence de cette sélection, en confirmant qu'une très grande partie des événements sont couverts par cette sélection. Finalement, les labels sélectionnés ici sont adjoints à la feature 'Event.Creation.Time' présente initialement afin de passer d'un modèle univariable à un modèle multivariable.

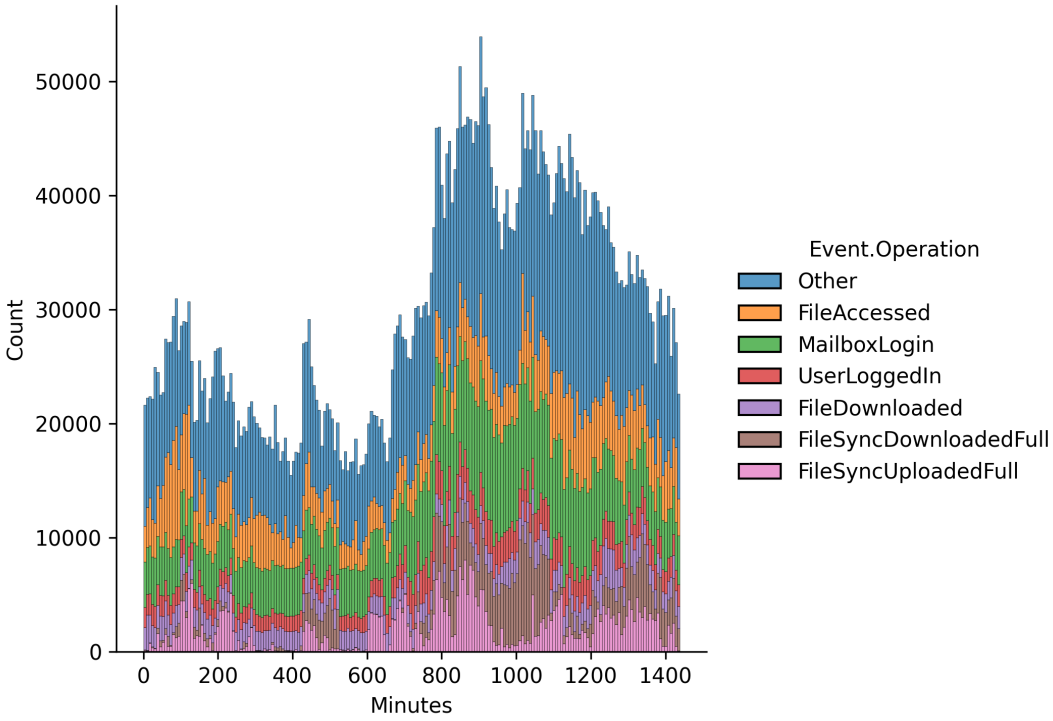


FIGURE 17 – Distribution des labels de 'Event.Operations' sur l'entièreté des événements, pour tous les utilisateurs, après sélection.

Il reste important de noter que cette sélection est basée sur des métriques purement statistiques et ne prend pas activement en compte la sémantique derrière chaque label. Cette prise en compte demanderait cependant une étude plus approfondie des labels et une expertise avancée, non seulement en cybersécurité mais

aussi en environnements Microsoft, ce qui n'est pas le sujet du travail présenté ici. Cette perspective est ainsi laissée ouverte pour des travaux ultérieurs.

3.3.7 Application

Les figures 18 et 19 illustrent un exemple de résultats de l'application de cette feature sur les modèles. Les événements détectés par chaque algorithme sont indiqués par des points de leur couleur. On observe rapidement une nouvelle position des points plus axée autour des "pics" au sein de la représentation, qui sortent des plateaux habituels. Ces résultats semblent intuitivement valider la plus value de l'attribut 'Event.Operation', dont l'impact positif sera étudié de manière plus globale au sein des Sections 4 et 5 du document. La variance au sein des résultats semble ne pas être impactée de manière notable par cette modification, malgré qu'elle ait été le point de départ de celle-ci.

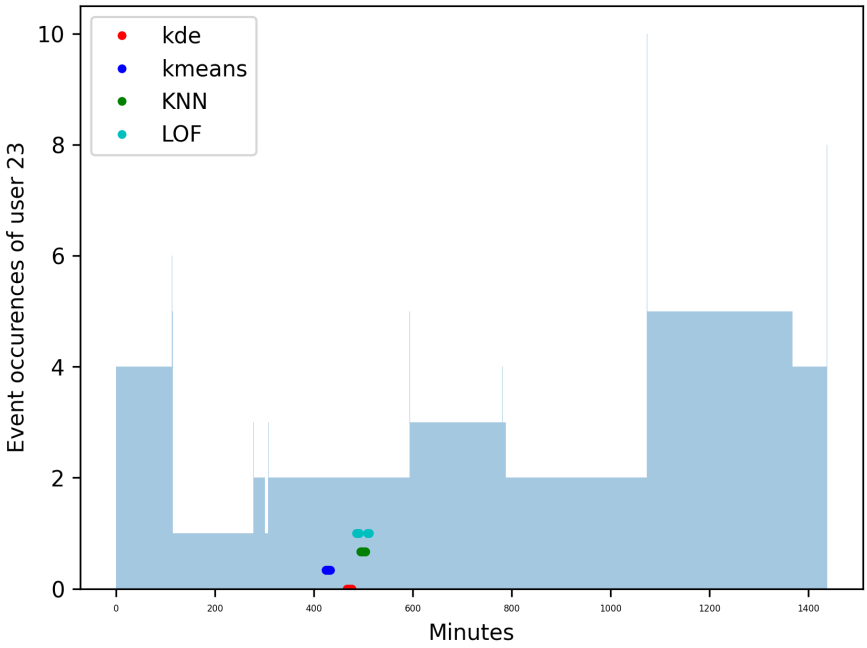


FIGURE 18 – Visualisation des anomalies détectées chez un utilisateur, avant intégration de la feature supplémentaire

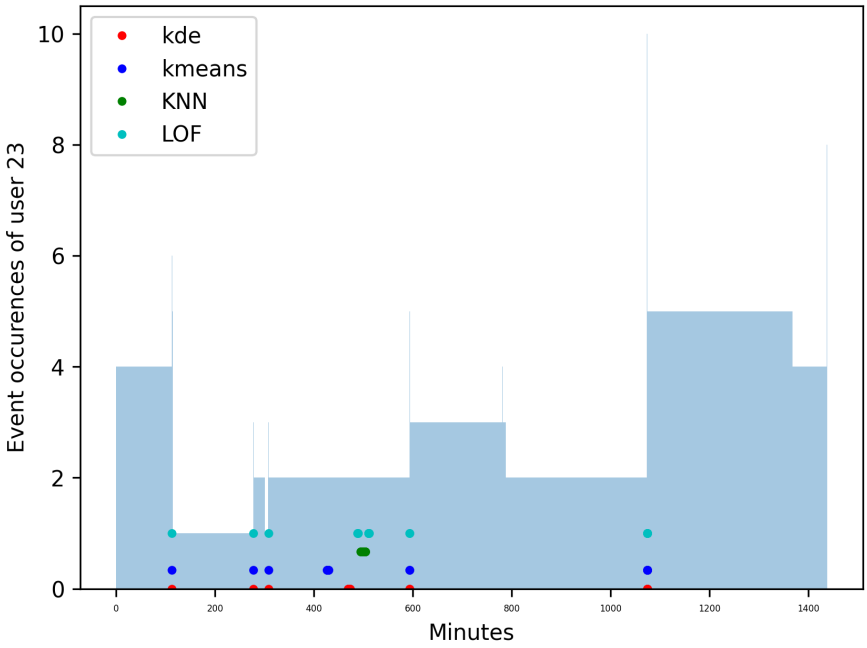


FIGURE 19 – Visualisation des anomalies détectées chez un utilisateur, après intégration de la feature supplémentaire

3.4 Résumé

Cette section présente les axes d'amélioration explorés afin de réduire la variance anormale au sein des résultats renvoyés par le sous-modèle LOF, suite à une revue du travail de Letourneau. Particulièrement, l'exploration des données semble indiquer l'existence de types d'utilisateurs différents au sein des données, mêlant utilisateurs humains, scripts, mais également des utilisateurs mixtes. En creusant un peu plus, ces utilisateurs semblent également se démarquer par le type d'événement présents au sein de leurs logs. D'autres explorations semblent confirmer que certains utilisateurs ne proposent qu'nu faible nombre de types d'événements différents (moins de 4) sur des centaines d'événements répartis tout le long de la semaine. Le type d'événement se présente alors naturellement comme une feature décisive pour l'ensemble model, qui pourrait l'aider à représenter le comportement de certains utilisateurs de manière beaucoup plus claire, en lui attribuant un haut score d'anomalie si un événement d'un type inhabituel venait à se produire. Seuls 7 labels au sein de cette feature sont sélectionnés, de manière à couvrir un maximum de profils différents. Cet ajout échoue à réduire les problèmes de variance observés plus haut mais semble donner des résultats concluants au niveau de la détection d'événements (explorés dans une section suivante), ce qui pousse à laisser la problématique de la variance de côté pour étudier les autres impacts de cette modification.

3.5 Amélioration de la cohésion

3.5.1 Addition de modèles

La solution envisagée pour améliorer la cohésion globale des modèles est l'ajout d'une ou plusieurs algorithmes de détection supplémentaires au sein de l'ensemble method. Ainsi, il est possible pour l'algorithme supplémentaire de potentiellement venir valider les flags émis par les autres sous-modèles, ce qui renforcerait les méthodes déjà mises en place. Cette modification doit être faite avec attention car si une méthode supplémentaire peut effectivement permettre de valider des résultats renvoyés par d'autres algorithmes, le risque que celle-ci soit en désaccord avec les autres méthodes tout en puisant du temps de calcul existe. Cette section débute par une description des sous-méthodes déjà en place afin d'expliciter le contexte, et décrit ensuite les méthodes sélectionnées sur base de ce contexte.

3.5.2 Mise en application de l'ensemble method

L'ensemble method présentée dans le document de Letourneau est donc composée de 4 méthodes non supervisées. Chacune d'entre elle possède une philosophie différente et est adaptée pour assigner à chaque événement un score. Il devient ainsi possible de trier les événements de chaque utilisateur par leur score pour ensuite identifier les événements les plus anormaux. Leur sélection a été faite pour couvrir un spectre large des méthodes non supervisées les plus couramment utilisées, mais également sur base de 3 points, indiqués dans l'article original :

- Ces modèles sont capables de modéliser le comportement normal d'un utilisateur et signaler les événements anormaux.
- Ces modèles produisent des règles plus personnalisées et flexibles pour chaque utilisateur, en les modélisant individuellement.
- Enfin, ces modèles s'adaptent sans problème à de l'entraînement en temps réel sur des fenêtres de temps précises.

Les méthodes en place sont décrites au sein des sections suivantes.

3.5.3 Kernel Density Estimation

La Kernel Density Estimation[22][24] (en français Estimation par Noyau), abrégée KDE, est un modèle statistique non-paramétrique courant appliqué ici à l'apprentissage non-supervisé. Son fonctionnement repose sur l'estimation de densité d'une variable aléatoire, sur base d'un échantillon de population. Intuitivement, dans le cas d'une variable présentée sous forme d'un histogramme, le modèle KDE fonctionne comme un lissage de cet histogramme. Il devient ainsi possible d'associer à toute valeur de cette variable une mesure probabiliste. Le modèle dispose

de 2 paramètres : Le noyau associé ainsi que la fenêtre, qui définit le degré de lissage du modèle. Le noyau sélectionné ici est une distribution de Von Mises sur [0, 1440], représentant chaque minute d'une journée. Le paramètre de fenêtre est fixé à 1 et propose un bon équilibre entre le biais et la variance de la densité estimée[15]. Le fonctionnement de cette méthode est illustré dans la figure 20

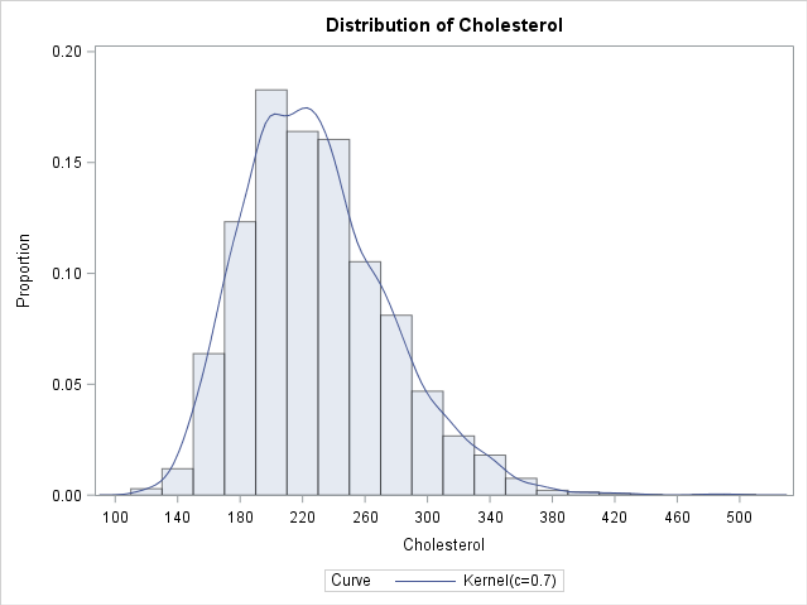


FIGURE 20 – Application du modèle KDE sur un histogramme [27]

3.5.4 K-nearest neighbors

Les K-nearest neighbors[5][4] (en français K plus proches voisins), abrégés kNN, est un modèle d'apprentissage supervisé courant adapté ici à la détection d'anomalies. De manière classique, l'algorithme fonctionne en identifiant les k membres les plus proches d'un élément, pour par exemple, dans une tâche de classification, déterminer la classe la plus représentée au sein de ces éléments. Dans un contexte probabiliste, il est également possible de simplement mesurer le taux de présence de chaque classe parmi les k membres les plus proches. Dans tous les cas, c'est le choix du paramètre k qui détermine les résultats de l'algorithme. La mesure de distance peut également influencer l'algorithme de la même manière. La version proposée ici de kNN est une version adaptée à l'apprentissage non supervisé fonctionnant de manière simple. Chaque élément est comparé à son k-ème membre le plus proche, suggérant ainsi un relatif isolement si cette distance s'avère être élevée. L'article de Letourneau fait le choix de la valeur 20 comme paramètre k, proposant à nouveau un bon équilibre entre biais et variance au sein de données simulées[15]. Le fonctionnement de cette méthode est illustré dans la figure 21

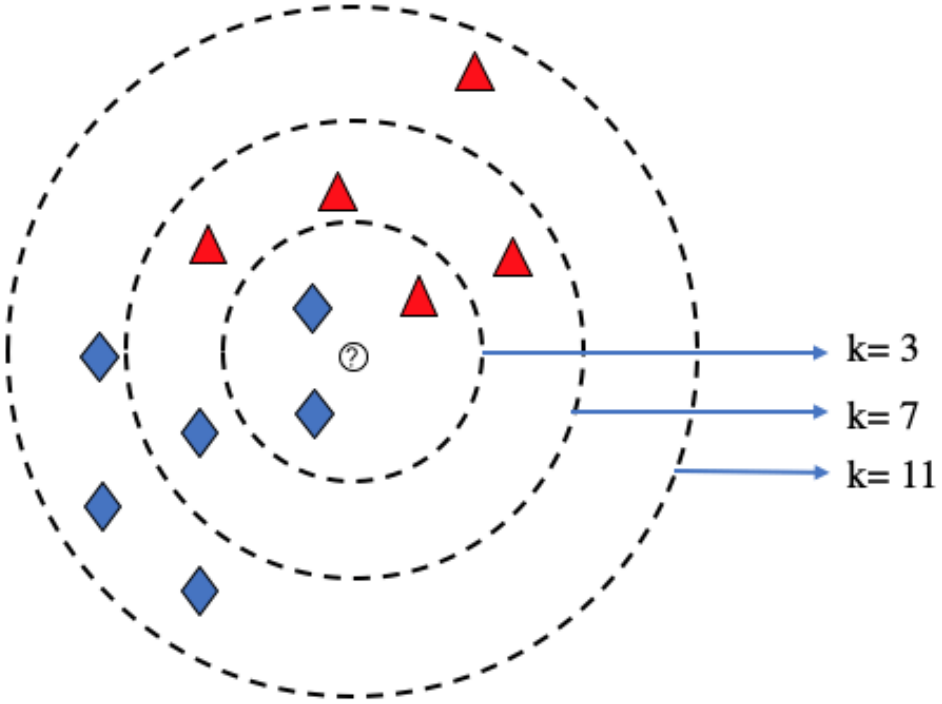


FIGURE 21 – Représentation du fonctionnement classique du modèle kNN, pourk = 3, k = 7 et k = 11 [29]

3.5.5 Local Outlier Factor

Le Local outlier factor[2] (en français Facteur de valeur aberrante locale), abrégé LOF, est un algorithme non supervisé de détection d'anomalies. Celui fonctionne sur la même base que l'algorithme kNN en poussant son fonctionnement. De manière simplifiée, LOF calcule pour chaque élément la densité de l'ensemble formé par cet élément et ses k voisins les plus proches. La formule exacte demande un certain nombres de calculs et ne sera pas explicitée ici. Il se sert ensuite de cette mesure pour la comparer à celle de chacun de ses k voisins les plus proches. Un élément peut ainsi se retrouver isolé de la majorité des autres éléments, mais présent dans un cluster limité, ce qui sera pris en compte par l'algorithme dans son calcul de score associé, contrairement à un algorithme kNN simple. Le fonctionnement de cette méthode est illustré dans la figure 22

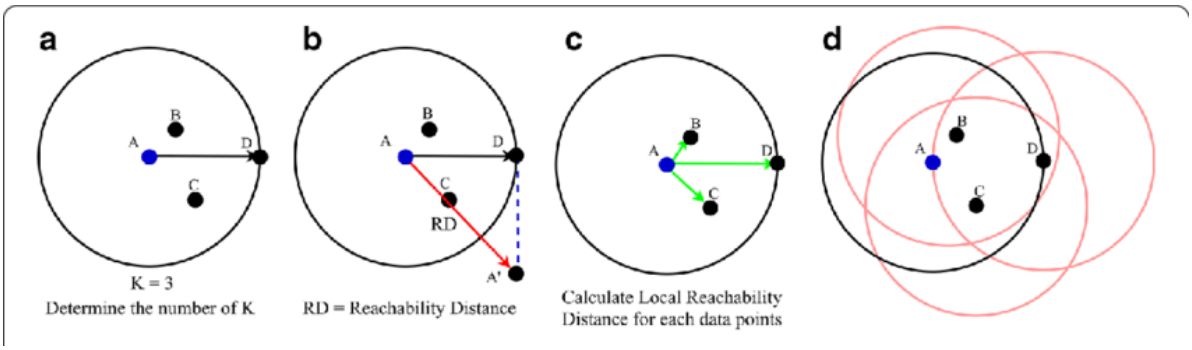


FIGURE 22 – Représentation du fonctionnement du modèle LOF [19]

3.5.6 K-means

K-means [18] [17] (en français les K moyennes), désigne un algorithme non supervisé utilisé en clusterisation. Son fonctionnement repose sur un nombre k de points imaginaires appelés les centroïdes placés au sein de l'espace des données. Ceux-ci sont dans un premier temps placés à des positions complètement aléatoires dans l'espace des variables. L'algorithme répète ensuite les étapes suivantes, jusqu'à l'atteinte d'un critère d'arrêt : L'algorithme détermine, pour chaque point du dataset, de quel centroïde il est le plus proche. Pour chaque centroïde, une moyenne de tous les points dont il est le plus proche est calculée. Ce point définit une nouvelle position dans l'espace des variables. Chaque centroïde est déplacé vers cette nouvelle position, et l'algorithme retourne à la première étape. Cet algorithme garantit pour chaque centroïde d'atteindre après un certain nombre d'itérations un optimum local représentant le centre d'un cluster donné du graphe. Les entités sont ensuite assignées au centroïde le plus proche. Le score associé à chaque élément devient ainsi sa distance à son centroïde le plus proche. La paramètre central de ce modèle est donc la valeur k qui définit le nombre de centroïdes à être définis. Celui-ci est défini dynamiquement sur base du coefficient de silhouette, entre les valeurs de 2 et 7. Le fonctionnement de cette méthode est illustré dans la figure 23

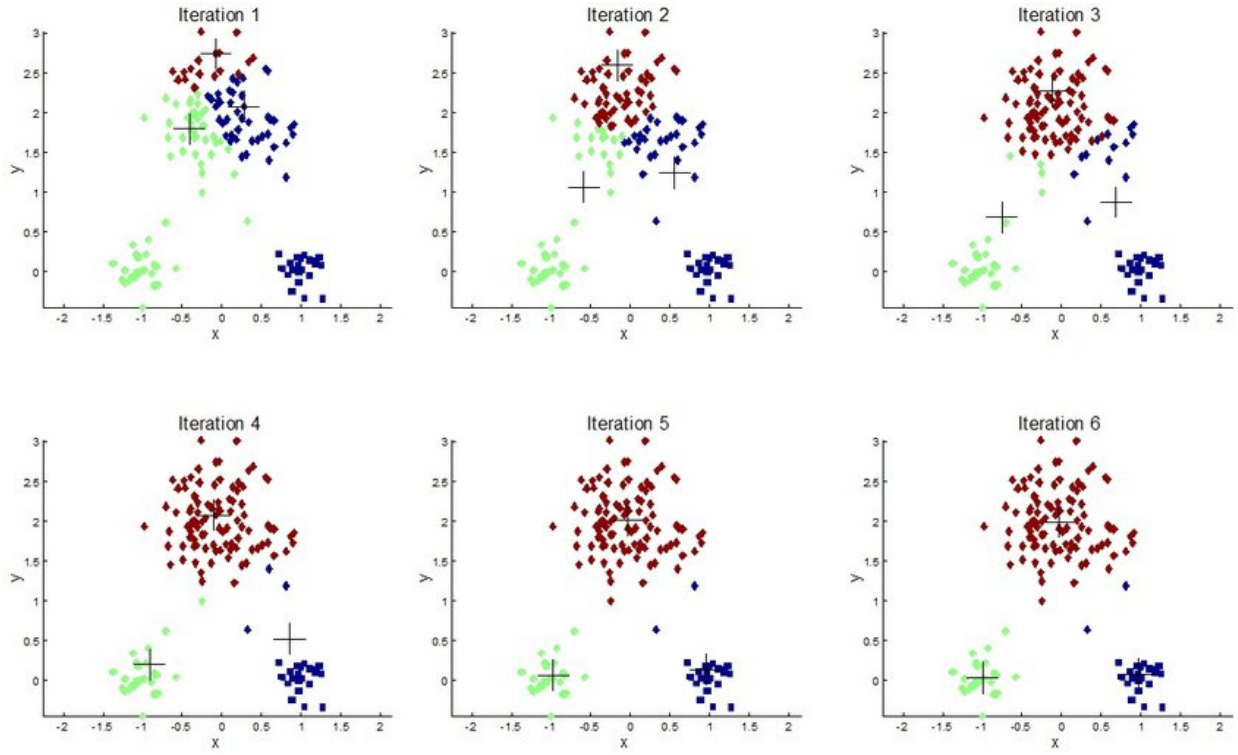


FIGURE 23 – Fonctionnement étape par étape de l'algorithme k-means [11]

3.5.7 Addition de modèles

Une solution rapidement envisagée pour améliorer la cohésion globale au sein de l'ensemble model est d'ajouter des modèles supplémentaires en son sein. Cette méthode présente l'avantage principal d'offrir la possibilité de détecter des patterns supplémentaires passant sous les radars des méthodes en place. En outre, de nouvelles méthodes peuvent valider des détections soutenues précédemment par une seule méthode. Ainsi, deux modèles de détection ont été sélectionnés sous plusieurs contraintes. Premièrement, le modèle doit garder une un aspect "référence" pour être dans la continuité des modèles présentés dans l'article original. Deuxièmement, le modèle doit proposer un type de détection différent des modèles déjà présents. Finalement, le modèle doit proposer son apprentissage et sa détection dans des échelles de temps et de mémoire raisonnables et comparables aux méthodes proposées précédemment. Sur base de ces critères, 2 modèles ont été sélectionnés.

3.5.8 Isolation Forest

L'Isolation forest[16] (en français Forêt d'Isolement) est un algorithme de détection d'anomalies utilisé couramment dans le monde de l'apprentissage non supervisé. Son fonctionnement est dérivé de celui des arbres de décisions classiques. Celui-ci consiste à, itérativement, diviser l'espace des variables en 2 parties, grâce

à une condition complètement aléatoire passée sur l'une des variables des données. Graphiquement, dans un espace à deux dimensions, l'opération consiste donc à tracer une ligne verticale ou horizontale au sein de l'espace des données. L'opération est répétée jusqu'à ce que l'un des membres du dataset se retrouve isolé dans une section. Plus le nombre de divisions nécessaires avant l'isolement est bas, plus les chances que l'élément en question soit isolé du reste des éléments sont élevées. Pour augmenter la fiabilité de l'algorithme, le processus est répété un certain nombre de fois, pour finalement combiner les résultats entre eux, créant ainsi une "forêt" de decision trees.

Le choix de cet algorithme est poussé par sa complexité temporelle linéaire et ses faibles besoins en mémoire, mais également par la base sur laquelle il repose, le decision tree, qui n'est explorée dans aucune des méthodes déjà présentes. Son paramètre principal est n, le nombre d'estimateurs successivement construits avant de combiner les résultats. La valeur de n est fixée à 100, valeur proposée par l'article original. L'Isolation forest étant l'une des méthodes les moins coûteuses en temps de ce projet, réduire ses performances en détection en échange de temps de vitesse d'exécution ne semble pas pertinent. A l'inverse, augmenter la valeur de n ne semblait pas offrir de résultats radicalement différents, en plus de rallonger le temps d'exécution. Le fonctionnement de cette méthode est illustré au sein de la figure 24.

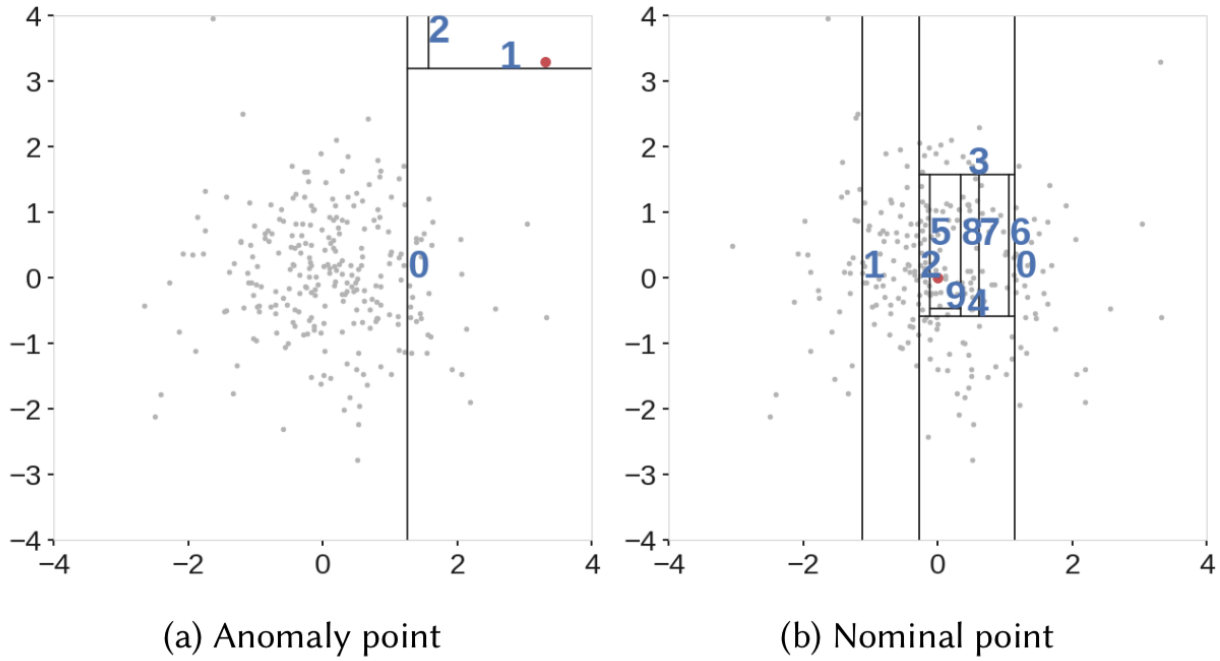


FIGURE 24 – Itérations successives de l'algorithme Isolation Forest, pour la détection d'une anomalie et d'un élément normal.[12]

3.5.9 One Class SVM

La One Class Support Vector Machine[26] (en français Machine de Vecteur à supports à une classe), abrégé OCSVM, est un modèle de machine learning largement utilisé pour des tâches de classification, mais aussi de régression. Celle-ci se base sur le principe plus général de la SVM. Le fonctionnement des SVM repose sur le fait d'élargir un ensemble de données sur une dimension supérieure afin d'identifier une fonction capable de séparer les données de la manière la plus optimale possible. Cette fonction est nommée la fonction noyau tandis que la distance entre ces éléments et cette séparation est ainsi nommée la marge, que l'on cherche à maximiser. La OCSVM, modifie ce principe en remplaçant la fonction noyau par une hypersphère. La tâche devient ainsi de passer à la dimension supérieure pour identifier l'hypersphère englobant au mieux les données. Le paramètre ν indique une fraction d'éléments à ignorer, ce qui sera utilisé par l'algorithme pour construire une hypersphère englobant le pourcentage restant des éléments, sur une dimension supérieure. Enfin, les scores des différents éléments sont définis en fonction de leur distance par rapport à cette hypesphère, les éléments les plus éloignés étant donc naturellement désignés comme des anomalies.

Tout comme l'Isolation Forest, cet algorithme est sélectionné pour son fonctionnement basé sur les SVM, fondamentalement différent des modèles kNN, LOF (neighbor-based), Kmeans (centroïd-based) et KDE (statistic-based). En terme de temps, la OCSVM se révèle coûteuse en temps d'exécution avec une complexité en temps de $O(n^2)$ ou plus. L'impact de cette complexité est étudié au sein d'une section suivante. Le fonctionnement de cette méthode est illustré dans la figure 25

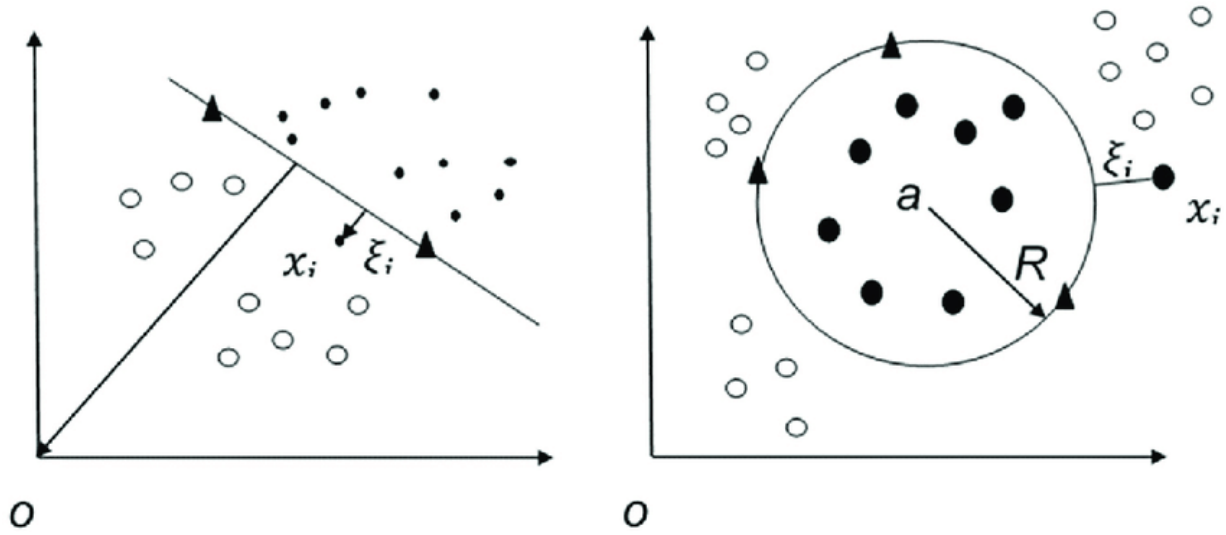


FIGURE 25 – Représentation d'une One Class SVM [28]

3.5.10 Révision du code

Dans un souci de simplification et de compatibilité du code avec les modifications proposées ici, certaines implémentations proposées par l'article original ont du être abandonnées. Particulièrement, celles de K-Means, itérant en fonction du coefficient de silhouette afin de sélectionner le paramètre k optimal[15], a été abandonnée au profit d'une implémentation plus flexible. De la même manière, l'implémentation de KDE posait de nombreux problèmes de compatibilité et a dû être revue également. Les nouvelles implémentations proposées sont issues du package Scikit-learn. Les paramètres ont été gardés identiques à l'implémentation originale proposée par Letourneau.

4 Résultats

4.1 Critères d'évaluation

En l'absence de ground truth, Letourneau propose l'évaluation des résultats via les critères suivants :

- Temps d'exécution requis pour entraîner les modèles et détecter les anomalies ;
- Interesection des anomalies détectées entre les techniques.

L'intersection des anomalies détectées entre les différentes techniques est calculée autour de 4 métriques que nous définissons plus formellement ici : L'indice d'intersection de degré 1, ou *ii1*, représente le nombre moyen d'anomalies détectées par une et une seule méthode. L'indice d'intersection de degré 2, ou *ii2* caractérise le nombre moyen d'anomalies détectées par 2 méthodes et aucune autre. Le même raisonnement suit pour les indices d'intersection 3 et 4+. Il est ainsi possible de déterminer à quel point les anomalies détectées sont corrélées par plusieurs méthodes ou non.

4.2 Modèles évalués

Plusieurs modèles sont évalués, combinant de manière différente les adaptations proposées lors de la section différente.

La première version de ce modèle est la version originale du modèle proposé dans l'article de Letourneau, proposant une analyse univariable sur base de 4 modèles. Nous appellerons cette version la V1.

Une deuxième version de ce modèle est une version adaptée de la V1 auquel sont adjoints 2 modèles supplémentaires, la OCSVM et l'Isolation forest. Nous appellerons cette version la V2.

Également dérivée du modèle V1, un troisième modèle intègre quant à lui une feature supplémentaire à l'analyse univariable, le type d'événement de la ligne de log. Nous appellerons cette version la V3.

4.3 Datasets utilisés

Cette évaluation est réalisée sur les datasets fournis par l'entreprise Sherweb afin de se rapprocher au plus près du contexte d'utilisation industriel. Le dataset d'entraînement et celui de test sont sélectionnés en fonction de leur proximité temporelle, pour maximiser le nombre d'utilisateurs simultanément présents sur les deux datasets, mais également sur base de leur taille. Sur base de ces critères, ce

sont les datasets extract-20200701_f et extract-20200709_f qui sont sélectionnés, respectivement pour l'entraînement et le testing.

Les mesures enregistrées dans cette section sont le fruit d'une itération unique et complète de chaque version du programme.

4.4 Spécifications techniques

L'entièreté des mesures proposées au sein de ce document sont calculées sur un ordinateur personnel dont les caractéristiques sont décrites ci-dessous :

- CPU : Intel(R) Core(TM) i5-6400 CPU @ 2.70GHz 2.70 GHz
- RAM : 16,0 Go
- GPU : NVIDIA GeForce GTX 960
- OS : Windows 10 Famille version 22H2
- Python version : 3.11

4.5 Résultats

4.5.1 Temps d'exécution

Les mesures en temps effectif du processus. Le travail présent étant réalisé sur du matériel différent, les mesures présentées par le papier original sont ignorées au sein de cette section.

Temps d'exécution total			
Version	V1	V2	V3
Entraînement	7 min 49.3 s	22 min 29.18 s	8 min 1.59 s
Détection	33 min 40.7	44 min 58.66 s	26 min 23.79 s

Temps d'exécution par méthode - Entraînement			
Version	V1	V2	V3
Kmeans	1 min 27.06 s	1 min 32.41 s	1 min 32.79 s
KDE	1 min 12.17 s	1 min 12.32 s	1 min 15.71 s
kNN	1 min 11.63	1 min 33.72 s	1 min 11.36 s
LOF	3 min 57.57	6 min 32.14 s	4 min 1.73 s
IsoForest	/	3 min 56.75 s	/
OCSVM	/	7 min 41.85 s	/

Temps d'exécution par méthode - Détection			
Version	V1	V2	V3
Kmeans	7 min 9.37 s	5 min 45.06 s	3 min 38.92 s
KDE	9 min 20.80 s	11 min 38.63 s	10 min 4.45 s
kNN	9 min 39.39 s	8 min 55.50 s	6 min 14.51 s
LOF	7 min 30.39 s	8 min 56.20 s	6 min 25.90 s
IsoForest	/	3 min 39.12 s	/
OCSVM	/	9 min 4.14 s	/

4.5.2 Intersection des différentes méthodes

La figure 26 présente le diagramme associé à la V1, la figure 27 présente le diagramme, plus complexe, associé à la V2, et la figure 28 présente le diagramme de la V3. Les indices d'intersection sont basés sur les diagrammes de Venn définis pour chaque version. Ceux-ci répertorient, pour chaque méthode en place, les 20 événements auquel il attribue le plus haut score, rassemblées au sein d'un disque. Le diagramme rassemble ainsi dans des sections communes (les superpositions de plusieurs disques) les événement signalés de cette manière par plusieurs méthodes de détection. Une moyenne est réalisée sur l'ensemble des utilisateurs du dataset de détection pour modéliser le comportement global des données. Les indices d'intersection *ii2*, *ii3* et *ii4+* ne sont pas présentés pour la V2 car la mesure n'est pas équivalente à celle des autres version, du fait du nombre d'intersections plus élevé. La mesure de *ii1* est maintenue car pondérée de la même manière sur les 3 et offre une interprétation intéressante des résultats à elle seule. Une illustration des emplacements couverts par chaque indice d'intersection est proposée par la figure 29.

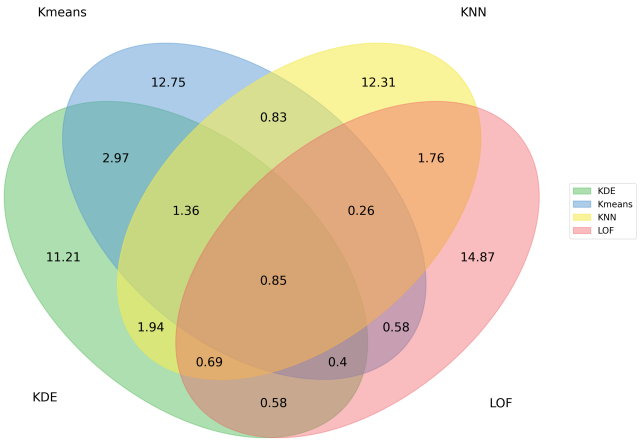


FIGURE 26 – Diagramme de Venn de la V1

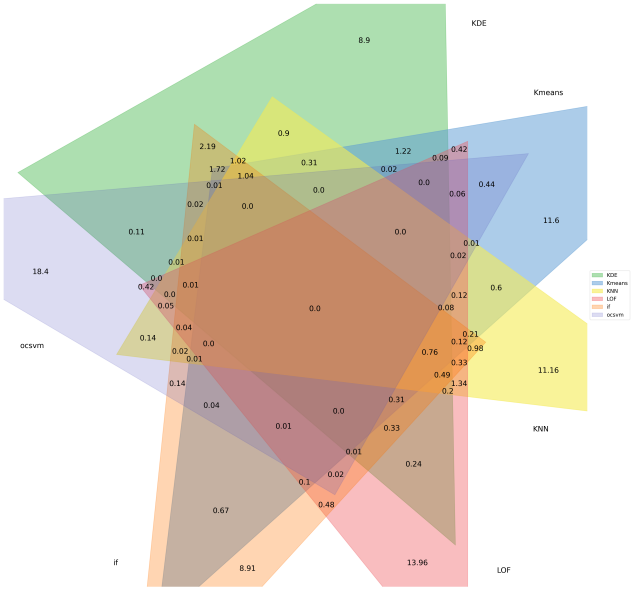


FIGURE 27 – Diagramme de Venn de la V2

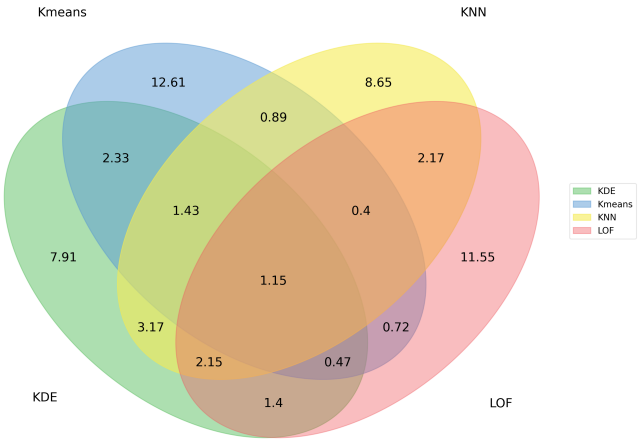


FIGURE 28 – Diagramme de Venn de la V3

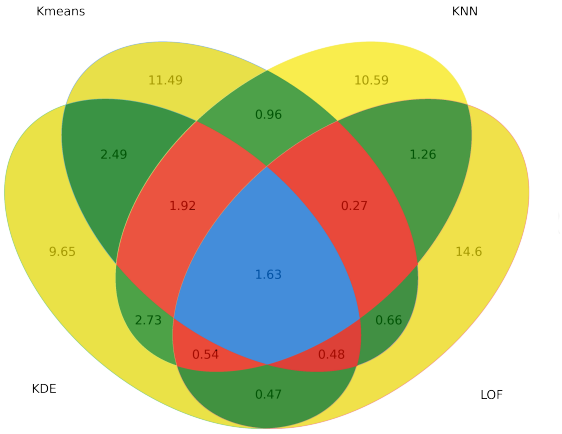


FIGURE 29 – Illustration de la répartition des intersections de degré 1, en jaune, de degré 2, en vert, de degré 3 en rouge, de degré 4+ en bleu, sur un diagramme quelconque

Le calcul des indices d'intersection est illustré sur la figure 29. Le calcul se

réalise par une moyenne des valeurs associés à toutes les intersections d'un degré donné. Les calculs amènent au tableau suivant.

Indices d'intersection			
Version	V1	V2	V3
ii1	12.79	12.23	10.18
ii2	1.44	/	1.78
ii3	0.68	/	1.11
ii4+	0.85	/	1.40

5 Discussion

5.1 Temps d'exécutions

Plusieurs éléments intéressants ressortent des temps d'exécutions des méthodes.

Lors de la phase d'entraînement, parmi les 4 méthodes de base proposées par Letourneau, seul l'algorithme LOF se distingue des autres, placés eux sur la même échelle de temps. Au sein de la V2, les 2 algorithmes proposés, en respect de leur complexité temporelle, proposent aussi un temps d'exécution au delà des autres méthodes.

Au vu de ces résultats, il apparaît rapidement que les algorithme OCSVM et LOF proposent les temps d'exécutions d'une échelle visiblement différente du reste des algorithmes. L'Isolation Forest elle se place dans une sorte d'entre-deux.

Lors de la phase de détection, on note tout d'abord des temps plus élevés à la détection qu'à l'apprentissage, dûs à la taille du set de détection, utilisé en conditions réelles, plus grand qu'un set de données de validation classique. L'algorithme KDE particulièrement, s'avère peu efficace sur des sets de données très larges. Kmeans, quant à lui, propose globalement les meilleures performances à la détection. Enfin, l'Isolation forest propose les temps de détection les plus courts de tous les algorithmes.

En temps d'exécution global, on se rend rapidement compte que l'addition de modèles supplémentaires, symbolisée dans la version V2 pèse beaucoup sur l'efficacité temporelle du programme, autant en apprentissage qu'en détection. A l'inverse, l'addition de features supplémentaires au sein du modèle semble ne pas avoir d'influence notable sur les performances.

En terme de temps d'exécution, ce sont donc les algorithmes Kmeans, KNN qui proposent les performances les plus satisfaisantes en restant en dessous de 1 min 30 au niveau de l'apprentissage, et en dessous de 10 min pour la détection. En dehors de ça, l'Isolation forest propose des résultats définitivement intéressants, moyens en terme d'apprentissage avec près de 4 minutes, et très bons en terme de détection avec un temps quasiment égal. A l'inverse, LOF et surtout l'OCSVM pèsent fort dans la balance du temps. L'algorithme KDE quant à lui, perd en intérêt à la détection au fur et à mesure de l'augmentation des données.

5.2 Intersections des différentes méthodes

De la même manière, de nombreux résultats intéressants apparaissent au sein des diagrammes de VENN et des différents indices d'intersection. Tout d'abord, on observe certains patterns dans les diagrammes associés à la V1. On note par exemple une cohésion plus forte entre Kmeans et KDE qu'entre les autres méthodes. L'algorithme LOF semble quant à lui proposer une cohésion plus limi-

tée que les autres méthodes, particulièrement visible avec le chiffre de 14.87. La méthode avec lequel il détecte en moyenne le plus d'anomalies en commun est KNN, ce qui s'explique potentiellement grâce à leur conception fondamentalement proche.

La V2 propose quant à elle des résultats globalement limités en terme de cohésion, portés par un *ii1* de 12.23, seulement légèrement plus bas que le *ii1* précédent. En prêtant attention aux intersections, c'est l'algorithme OCSVM qui ressort pour son manque de cohésion critique avec les autres méthodes, mis en valeur par son nombre d'anomalies moyen de 18.4. On en conclut que l'algorithme a une tendance extrêmement faible à valider les résultats d'autres méthodes. A l'inverse, les méthodes de KDE et de l'Isolation Forest proposent une cohésion notable avec les autres algorithmes, identifiant en moyenne moins de 9 erreurs uniques chacune.

Enfin, la V3 propose des patterns d'intersections globalement semblables à ceux de la V1, tout en témoignant d'un augmentation de la cohésion notable sur toutes les méthodes. Le *ii1* passe ainsi de 12.79 à 10.18, le *ii2* de 1.50 à 1.78, le *ii3* de 1.00 à 1.11 et enfin le *ii4+* de 0.85 à 1.40. La baisse du score *ii1* indique une baisse globale du nombre d'éléments détectés par un algorithme unique tandis que l'augmentation de *ii2*, *ii3* et *ii4+* prouve une montée du nombre de détections d'anomalies communes à plusieurs algorithmes. L'évolution du *ii4+* particulièrement démontre une montée du nombre d'éléments détectés par 4 anomalies (pouvant donc être considérés comme des événements critiques) de près de 60%.

5.3 Interprétations

Chacune des deux versions (V2, V3) implémentées dans le cadre de ce travail offre des informations pertinentes pour notre question de recherche. Il ressort rapidement que l'OCSVM propose des performances décevantes face aux métriques d'évaluation, ne validant que très peu les résultats des autres méthodes présentes tout en demandant de grands sacrifices au niveau du temps d'exécution. A l'inverse, l'Isolation Forest réalise des bonnes performances en validant beaucoup d'autres méthodes tout en étant moyennement voire peu gourmand en ressources. Enfin, l'algorithme LOF, déjà présent dans le document original, se révèle être la méthode la moins pertinente parmi les 4 proposées. Celle-ci, en plus de proposer un mode de fonctionnement fondamentalement proche de l'algorithme KNN, demande largement plus de ressources en temps que ses 3 méthodes partenaires.

5.4 Version supplémentaire

Il devient dès lors possible d'envisager un ensemble model composé de KDE, KNN, Kmeans et de l'Isolation Forest, proposant une détection basée sur le temps, mais également sur une autre feature pertinente, le type d'événement, combinant les points forts des versions du programme présentées ici. Une telle version est implémentée en tant que version V4 du programme, dont les résultats sont adjoints aux résultats précédents dans les tableaux suivants. Son diagramme de VENN est quant à lui présenté au sein de la figure 30.

Temps d'exécution total				
Version	V1	V2	V3	V4
Entraîn.	7 m 49.3 s	22 m 29.18 s	8 m 1.59 s	7 m 36.4 s
Détection	33 m 40.7	44 m 58.66 s	26 m 23.79 s	20 m 25.49 s

Temps d'exécution par méthode - Entraînement				
Version	V1	V2	V3	V4
Kmeans	1 m 27.06 s	1 m 32.41 s	1 m 32.79 s	1 m 38.49 s
KDE	1 m 12.17 s	1 m 12.32 s	1 m 15.71 s	1 m 21.52 s
kNN	1 m 11.63	1 m 33.72 s	1 m 11.36 s	1 m 20.05 s
LOF	3 m 57.57	6 m 32.14 s	4 m 1.73 s	/
IsoForest	/	3 m 56.75 s	/	3 m 15.58 s
OCSVM	/	7 m 41.85 s	/	/

Temps d'exécution par méthode - Détection				
Version	V1	V2	V3	V4
Kmeans	7 m 9.37 s	5 m 45.06 s	3 m 38.92 s	3 m 33.00 s
KDE	9 m 20.80 s	11 m 38.63 s	10 m 4.45 s	8 m 10.85 s
kNN	9 m 39.39 s	8 m 55.50 s	6 m 14.51 s	5 m 25.31 s
LOF	7 m 30.39 s	8 m 56.20 s	6 m 25.90 s	/
IsoForest	/	3 m 39.12 s	/	3 m 16.33 s
OCSVM	/	9 m 4.14 s	/	/

Indices d'intersection				
Version	V1	V2	V3	V4
ii1	12.79	12.23	10.18	11.28
ii2	1.44	/	1.78	1.34
ii3	0.68	/	1.11	0.99
ii4+	0.85	/	1.40	1.72

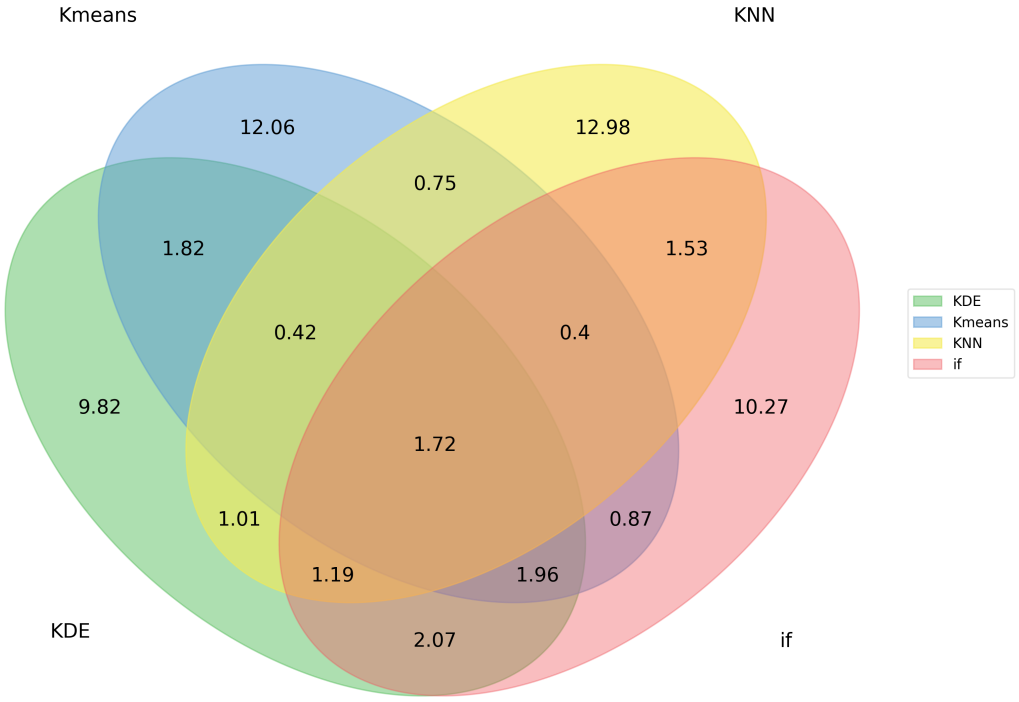


FIGURE 30 – Diagramme de Venn de la V4

5.5 Interprétation des résultats de la V4

La V4 offre des résultats satisfaisants autour de chacun de nos critères d'évaluation. Comparativement à la figure 28, on observe une montée de l'indice *ii1* de KNN, ce qui s'explique par la disparition de LOF, modèle dont il est fondamentalement proche. KDE aussi accuse une augmentation de son indice *ii1* tandis que Kmeans et l'Isolation Forest proposent des *ii1* légèrement plus bas que dans la V3. On observe qu'en dehors des zones partagées avec KNN, l'Isolation Forest propose une meilleure cohésion avec chacune des autres méthodes en place.

Ces observations se vérifient au niveau du tableau des indices d'intersection, où l'on observe des résultats globalement comparables à ceux de la V3. On observe une diminution de l'*ii2* et de l'*ii3*, tandis qu'on enregistre une montée des *ii1*

et *ii4+*. Ces résultats équilibrés poussent à penser que la cohésion globale reste inchangée entre les modèles V3 et V4. On observe cependant de des plus écarts entre les indices plus accentués qu'au sein de la V3, dans le sens où l'on trouve plus d'anomalies détectées par 1 seule ou 4 méthodes, et moins d'anomalies détectées par 2 ou 3 méthodes. Si elles semblent équivalentes d'un point de vue recherche, d'un point de vue métier, une méthode ou l'autre peut être préférée en fonction de la politique de détection que l'on désire mettre en place.

En terme de temps d'exécution, cette version se présente comme la plus efficace parmi toutes les versions en place, autant au niveau de l'entraînement que de la détection. On remarque que dans chacune de ces deux situations, l'absence de LOF a encore une fois un effet positif sur les performances, l'Isolation Forest s'avérant légèrement plus rapide lors de l'entraînement (entre 40 secondes et 3 minutes de différence avec LOF) et définitivement plus rapide lors de la détection (entre 3 minutes et 5 minutes 30 de différence avec LOF). Les temps des autres méthodes en place restent quant à eux cohérents avec les résultats calculés au sein des versions précédentes (de quelques secondes d'écart à 3 minutes).

5.6 Conclusions sur la V4

La V4 du programme confirme l'idée émise précédemment et combine effectivement les points forts de la V2 et la V3. Celle-ci maintient la cohésion globale des méthodes améliorée par la V3 à l'aide de l'intégration d'une feature supplémentaire, le type d'événement, tandis que le remplacement du sous-modèle LOF par le modèle de l'Isolation Forest permet d'offrir une vraie accélération du programme, autant à l'apprentissage qu'à la détection.

6 Conclusion

6.1 Résumé de la recherche

Le document présent reprend un travail effectué précedemment par Louis-Simon Letourneau, soutenu par un article scientifique. L'article présente un modèle ensembliste non-supervisé étudiant des logs applicatifs proposés par l'entreprise Sherweb. Le document présent aborde le code et l'article d'un oeil critique afin d'en dégager 2 axes d'amélioration. Un axe d'amélioration identifié est celui de la cohésion entre les sous-méthodes qui semble laisser entrevoir une fenêtre d'amélioration. L'ajout d'un modèle d'Isolation forest et d'OCSVM est proposé en tant que solution à ce problème, ce qui donne naissance à une version baptisée V2. Un autre axe d'amélioration identifié est la présence d'une haute variance au sein des résultats de LOF, qui mène à une étude approfondie des données. Cette exploration amène à la conclusion qu'intégrer comme feature le type d'événement permettrait au modèle de mieux interpréter les données. Une version V3 du programme est créée sur base de cette intégration de feature. La V3 semble échouer à diminuer la variance au sein des scores associés par LOF. Cette version donne cependant des résultats encourageants en terme de cohésion des différentes méthodes, sans trop sacrifier en temps d'exécution. La V2 donne quant à elle des résultats moins prometteurs, autant en temps d'exécution qu'en cohésion, principalement handicapés par la présence de l'OCSVM, mais fait la lumière sur la valeur ajoutée de l'Isolation forest. Une dernière version est donc proposée, la V4, combinant la valeur ajoutée des modèles V2, et V3. Celle-ci reprend le modèle initial et intègre la feature supplémentaire proposée par la V3 tout en remplaçant le sous-modèle LOF par un sous-modèle d'Isolation Forest. Les résultats enregistrés par cette version combinent bel et bien les forces des modèles V2 et V3 en maintenant l'amélioration de la cohésion amenée par la V3 et en réussissant à diminuer le temps d'exécution global du programme, autant à l'apprentissage qu'à la détection. Cette version semble donc bel et bien respecter lec critères d'évaluations établis et apporte une réponse à la question de recherche.

6.2 Réponse à la question de recherche

Nous pouvons dès lors formuler une réponse à la question de recherche présentée précédemment :

Comment améliorer la cohésion d'une méthode ensembliste non supervisée appliquée à de la détection d'anomalies, tout en réduisant la variance au sein des résultats ?

La cohésion d'une méthode ensembliste non supervisée appliquées à la détec-

tion d'anomalie peut être améliorée au travers de 2 axes : premièrement, l'intégration de features pertinentes, déterminées à partir d'une étude des données, deuxièmement, par l'ajout de modèles d'apprentissages supplémentaires tels que l'Isolation Forest, en remplacement de méthodes moins efficaces telles que le Local Outlier Factor. Ces 2 modifications proposent des améliorations notables en terme de cohésion des méthodes de détection, en impactant de manière négligeable le temps d'exécution du modèle. La variance au sein des résultats semble quant à elle ne pas être affectée par ces modifications.

6.3 Validité de la recherche

La validité de la recherche menée dans ce document est vérifiée par un ensemble d'éléments, autant intrinsèques qu'extrinsèques.

- La revue du travail précédent est réalisée sur base de l'article de recherche mais également sur l'implémentation même.
- Le travail suit une méthodologie de tests indépendants et de validation autour de critères repris de l'article scientifique précédent.
- Le travail propose plusieurs modifications, présentées individuellement, puis de manière combinée et compare ces versions sous des critères équivalents.
- Les modifications proposées dans ce travail sont entraînées et testées sur des datasets différents (plus petits) de ceux employés dans la section Résultats et ne sont donc pas spécialisées pour celui-ci.
- L'ensemble des modifications apportées au code déjà en place sont documentées.
- Les données employées proviennent d'un contexte professionnel reconnu et représentent de vrais comportements d'utilisateurs anonymisés.
- Les logs sont structurés selon des frameworks valables et reconnus.
- Les algorithmes employés au sein du modèles font preuve d'une reconnaissance scientifique et les implémentations employées proviennent de frameworks reconnus (scikit-learn).

6.4 Comment aller plus loin

De nombreuses extensions sont envisageables autour de ce projet.

Le système de vote de la méthode ensembliste est un aspect pertinent à étudier plus en profondeur. Différentes méthodes de combinaison des scores existent avec chacune leur impact propre. Des votes plus simples à la majorité, avec pondération ou non, présentent leurs propres atouts dont le système actuel ne propose pas,

et qui méritent d'être étudiées. Le système de borda count simplifié, dont est indirectement dérivé le système actuel, propose également des avantages uniques.

L'extraction des features réalisées ici est le fruit d'une exploration informelle des données qui n'a pas la même pertinence que des méthodes d'extraction des données reconnues. Ces méthodes ayant volontairement été mises de côté pour se concentrer sur le reste du projet, il est tout à fait envisageable d'intégrer ce genre de méthodes au projet.

La variance anormale au sein des résultats de l'algorithme LOF, mise en exergue par la phase de revue du travail de Letourneau, ne trouve suite à ce travail pas de réponse satisfaisante. De nombreux axes tels qu'une revue des paramètres des méthodes sont possiblement à l'origine de ce phénomène et méritent une exploration dans un travail ultérieur.

Enfin, comme vu dans les sections précédentes du document, de nombreux algorithmes non-supervisés de détection d'anomalies existent et seul un échantillon d'entre eux est traité dans le document présent. Ainsi, on peut imaginer la même étude être menée avec une autre sélection d'algorithmes.

Références

[1] Monowar H. Bhuyan, D. K. Bhattacharyya et J. K. Kalita. « Network Anomaly Detection: Methods, Systems and Tools ». In : *IEEE Communications Surveys & Tutorials* 16.1 (2014). Conference Name: IEEE Communications Surveys & Tutorials, p. 303-336. ISSN : 1553-877X. DOI : 10.1109/SURV.2013.052213.00046.

[2] Markus M Breunig et al. « LOF: Identifying Density-Based Local Outliers ». In : ().

[3] Varun Chandola, Arindam Banerjee et Vipin Kumar. « Anomaly detection: A survey ». In : *ACM Computing Surveys* 41.3 (30 juill. 2009), 15:1-15:58. ISSN : 0360-0300. DOI : 10.1145/1541880.1541882. URL : https://doi.org/10.1145/1541880.1541882 (visité le 23/02/2023).

[4] T. Cover et P. Hart. « Nearest neighbor pattern classification ». In : *IEEE Transactions on Information Theory* 13.1 (jan. 1967), p. 21-27. ISSN : 0018-9448, 1557-9654. DOI : 10.1109/TIT.1967.1053964. URL : http://ieeexplore.ieee.org/document/1053964/ (visité le 14/08/2023).

[5] Evelyn Fix et Joseph Lawson Hodges. « Nonparametric discrimination: consistency properties ». In : *Randolph Field, Texas, Project* (1951), p. 21-49.

[6] Masakiyo Fujimoto, Yotaro Kubo et Tomohiro Nakatani. « Unsupervised non-parametric Bayesian modeling of non-stationary noise for model-based noise suppression ». In : *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). ISSN: 2379-190X. Mai 2014, p. 5562-5566. DOI : 10.1109/ICASSP.2014.6854667.

[7] José Luis García-Lapresta et Miguel Martínez-Panero. « A Fuzzy Borda Count in Multi-person Decision Making ». In : *Multiple Objective and Goal Programming*. Sous la dir. de Tadeusz Trzaskalik et Jerzy Michnik. Advances in Soft Computing. Heidelberg : Physica-Verlag HD, 2002, p. 46-60. ISBN : 978-3-7908-1812-3. DOI : 10.1007/978-3-7908-1812-3_5.

[8] Nicolas Goix. *How to Evaluate the Quality of Unsupervised Anomaly Detection Algorithms?* 5 juill. 2016. DOI : 10.48550/arXiv.1607.01152. arXiv : 1607.01152[cs,stat]. URL : http://arxiv.org/abs/1607.01152 (visité le 30/04/2023).

[9] Markus Goldstein et Seiichi Uchida. « A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data ». In : *PLOS ONE* 11.4 (19 avr. 2016). Publisher: Public Library of Science, e0152173. ISSN : 1932-6203. DOI : 10.1371/journal.pone.0152173. URL : https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0152173 (visité le 23/02/2023).

[10] Aarish Grover. « Anomaly Detection for Application Log Data ». In : *Master's Projects* (1er avr. 2018). DOI : https://doi.org/10.31979/etd.znsb-bw4d. URL : https://scholarworks.sjsu.edu/etd_projects/635.

[11] Hani Guenoune et al. « Résolution du problème de tournées de véhicules avec collecte et livraison simultanées avec une approche coopérative de métaheuristiques. » Thèse de doct. 8 juin 2014. DOI : 10.13140/RG.2.2.33527.78245.

[12] Sahand Hariri, Matias Carrasco Kind et Robert J. Brunner. « Extended Isolation Forest ». In : *IEEE Transactions on Knowledge and Data Engineering* 33.4 (avr. 2021). Conference Name: IEEE Transactions on Knowledge and Data Engineering, p. 1479-1489. ISSN : 1558-2191. DOI : 10.1109/TKDE.2019.2947676.

[13] D. M. Hawkins. *Identification of Outliers*. OCLC: 851385856. Dordrecht : Springer Netherlands, 1980. isbn : 978-94-015-3994-4.

[14] Michael I. Jordan et Robert A. Jacobs. « Hierarchical Mixtures of Experts and the EM Algorithm ». In : *Neural Computation* 6.2 (mars 1994). Conference Name: Neural Computation, p. 181-214. issn : 0899-7667. doi : 10.1162/neco.1994.6.2.181.

[15] Louis-Simon Letourneau et al. « Data-Driven Comparison of Four Unsupervised Point-Anomaly Detection Methods ». In : ().

[16] Fei Tony Liu, Kai Ming Ting et Zhi-Hua Zhou. « Isolation Forest ». In : *2008 Eighth IEEE International Conference on Data Mining*. 2008 Eighth IEEE International Conference on Data Mining. ISSN: 2374-8486. Déc. 2008, p. 413-422. doi : 10.1109/ICDM.2008.17.

[17] S. Lloyd. « Least squares quantization in PCM ». In : *IEEE Transactions on Information Theory* 28.2 (mars 1982), p. 129-137. issn : 0018-9448. doi : 10.1109/TIT.1982.1056489. url : http://ieeexplore.ieee.org/document/1056489/ (visité le 14/08/2023).

[18] J. MacQueen. « Some methods for classification and analysis of multivariate observations ». In : *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. T. 5.1. University of California Press, 1er jan. 1967, p. 281-298. url : https://projecteuclid.org/ebooks/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fifth-Berkeley-Symposium-on-Mathematical-Statistics-and/chapter/Some-methods-for-classification-and-analysis-of-multivariate-observations/bsmsp/1200512992 (visité le 14/08/2023).

[19] Achmad Megantara et Tohari Ahmad. « A hybrid machine learning method for increasing the performance of network intrusion detection systems ». In : *Journal of Big Data* 8 (2 nov. 2021). doi : 10.1186/s40537-021-00531-w.

[20] Mary M. Moya et Don R. Hush. « Network constraints and multi-objective optimization for one-class classification ». In : *Neural Networks* 9.3 (1er avr. 1996), p. 463-474. issn : 0893-6080. doi : 10.1016/0893-6080(95)00120-4. url : https://www.sciencedirect.com/science/article/pii/0893608095001204 (visité le 13/08/2023).

[21] Nikunj C Oza. « Ensemble Data Mining Methods ». In : ().

[22] Emanuel Parzen. « On Estimation of a Probability Density Function and Mode ». In : *The Annals of Mathematical Statistics* 33.3 (sept. 1962). Publisher: Institute of Mathematical Statistics, p. 1065-1076. issn : 0003-4851, 2168-8990. doi : 10.1214/aoms/1177704472. url : https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-33/issue-3/On-Estimation-of-a-Probability-Density-Function-and-Mode/10.1214/aoms/1177704472.full (visité le 14/08/2023).

[23] Animesh Patcha et Jung-Min Park. « An overview of anomaly detection techniques: Existing solutions and latest technological trends ». In : *Computer Networks* 51.12 (22 août 2007), p. 3448-3470. issn : 1389-1286. doi : 10.1016/j.comnet.2007.02.001. url : https://www.sciencedirect.com/science/article/pii/S138912860700062X (visité le 02/03/2023).

[24] Murray ROSENBLATT. « Remarks on Some Nonparametric Estimates of a Density Function ». In : *The Annals of Mathematical Statistics* 27.3 (sept. 1956). Publisher: Institute of Mathematical Statistics, p. 832-837. ISSN : 0003-4851, 2168-8990. DOI : 10.1214/aoms/1177728190. URL : https://projecteuclid.org/journals/annals-of-mathematical-statistics/volume-27/issue-3/Remarks-on-Some-Nonparametric-Estimates-of-a-Density-Function/10.1214/aoms/1177728190.full (visité le 14/08/2023).

[25] Massiva ROUDJANE. « Détection des écarts de tendance et analyse prédictive pour le traitement des flux d'événements en temps réel ». Thèse de doct. Chicoutimi : Université du Québec à Chicoutimi, 2023. 237 p. URL : https://constellation.uqac.ca/id/eprint/9191/ (visité le 12/08/2023).

[26] David M.J. TAX et Robert P.W. DUIN. « Support Vector Data Description ». In : *Machine Learning* 54.1 (1er jan. 2004), p. 45-66. ISSN : 1573-0565. DOI : 10.1023/B:MACH.0000008084.60811.49. URL : https://doi.org/10.1023/B:MACH.0000008084.60811.49 (visité le 14/08/2023).

[27] *The area under a density estimate curve: Nonparametric estimates*. The DO Loop. 8 juill. 2011. URL : https://blogs.sas.com/content/iml/2011/07/08/the-area-under-a-density-estimate-curve-nonparametric-estimates.html (visité le 17/08/2023).

[28] Akram VASIGHIZAKER, Alok SHARMA et Iman (Abdollah) DEHZANGI. « A novel one-class classification approach to accurately predict disease-gene association in acute myeloid leukemia cancer ». In : *PloS one* 14 (11 déc. 2019), e0226115. DOI : 10.1371/journal.pone.0226115.

[29] Wei ZHANG et al. « A Distributed Storage and Computation k-Nearest Neighbor Algorithm Based Cloud-Edge Computing for Cyber-Physical-Social Systems ». In : *IEEE Access* PP (18 fév. 2020), p. 1-1. DOI : 10.1109/ACCESS.2020.2974764.