Construction non-supervisée de variables pour la détection d'anomalies dans les séries temporelles

Marine Hamon[*,**], Vincent Lemaire[*],
Nour Eddine Yassine Nair-Benrekia[*], Samuel Berlemont[*], Julien Cumin[*]

[*] Orange Innovation Lannion, Châtillon et Meylan, France
[**] Université de Rennes, France

Résumé. Pour détecter avec précision et sans a priori des anomalies dans une série temporelle, vaut-il mieux apprendre les détecteurs à partir de la représentation temporelle initiale ou calculer une nouvelle représentation (tabulaire) à l'aide d'une bibliothèque existante de construction automatique de variables? Dans cet article, nous répondons à cette question en menant une étude expérimentale approfondie pour deux détecteurs très populaires (Isolation Forest et Local Outlier Factor). Les résultats obtenus, pour 5 jeux de données différents, montrent que la nouvelle représentation, calculée à l'aide de la bibliothèque *tsfresh*, permet à Isolation Forest d'améliorer significativement sa performance.

1 Introduction

La littérature sur les séries temporelles traite diverses tâches d'apprentissage telles que la prévision, le clustering, ou encore la classification. Nous abordons dans cet article le problème de la détection d'anomalies (Boniol et al., 2023) dans les séries temporelles (TSAD en anglais). On note $\tau_i = \langle (t_1, x_1), \ldots, (t_m, x_m) \rangle$ une série temporelle univariée [1], où x_k est la valeur de la série a l'instant t_k. Dans cet article, on se place dans le cas où l'objectif est de prédire si le couple de valeur (t_i, x_i) correspond à une anomalie ("détection point par point" ou "point-wise detection" en anglais).

Ces dernières années, un consensus s'est dégagé au sein de la communauté sur le fait que la transformation des séries temporelles du domaine temporel vers un espace de représentation alternatif est l'un des moyens efficaces pour améliorer la précision des modèles. Ceci a pu être observé en classification (Renault et al., 2023), en classification précoce Bondu et al. (2022) mais aussi partiellement en détection d'anomalie sur quelques publications très ciblées sur des cas d'usage particuliers (Teh et al., 2021; Zhang et al., 2020).

Ces travaux récents ont motivé l'étude qui est présentée dans cet article. Nous reprenons cette idée de changer d'espace de représentation puis d'appliquer des détecteurs d'anomalie "usuels" dédiés aux données tabulaires (Renault et al., 2023). La question posée ici est : peut-on obtenir de meilleures performances de détection d'anomalies dans l'espace des caractéristiques calculées ou vaut-il mieux rester dans la représentation temporelle initiale ?

1. Cet article ne traite que des séries temporelles univariées

Le reste de cet article est organisé comme suit : la section 2 présente le contexte et les concepts clés utilisés dans cet article, de sorte qu'il puisse être correctement positionné dans la très vaste littérature de la détection d'anomalies pour séries temporelles. La section 3 présente la chaîne de traitement proposée. La section 4 présente le protocole expérimental. Puis la présentation des résultats détaillés est réalisée au sein de la section 5 avant de conclure dans la section finale.

2 Contexte et Concepts utilisés

Pour que le lecteur puisse correctement positionner les travaux menés dans cet article, nous présentons ci-dessous les grands axes présents au sein de la littérature de la détection d'anomalies pour séries temporelles (Boniol et al., 2023) et explicitons notre positionnement. La thématique de la détection des anomalies dans les séries temporelles étant très largement couverte dans la littérature, nous ne prétendons pas être exhaustifs mais juste factuels pour les besoins de l'étude menée.

2.1 Types d'anomalies

La littérature distingue à minima trois grands types d'anomalies : (i) les anomalies ponctuelles : par exemple, une transaction financière inhabituellement élevée par rapport à l'historique des transactions d'un client, (ii) les anomalies collectives (au sens d'une succession d'anomalies ponctuelles corrélées) : par exemple, une baisse soudaine du trafic sur un site web, par exemple due à une panne de serveur ou à une attaque par déni de service, (iii) les anomalies contextuelles : par exemple, une augmentation anormale de la consommation d'électricité dans une région donnée, par exemple en raison d'une tempête de neige ou d'une vague de chaleur exceptionnelle.

Dans l'étude portée dans cet article, nous faisons aussi l'hypothèse d'être le plus "agnostique" possible : (i) nous considérons tout au long de l'étude que nous ne possédons pas de boucle de retour (rétroaction) d'un utilisateur expert permettant de contextualiser les données observées (ii) de régler les paramètres des méthodes (iii) de régler la taille de fenêtre glissante (voir ci-dessous), (iv) nous ne faisons pas d'hypothèse si les anomalies sont ponctuelles ou collectives. Nous nous plaçons dans un cas le plus généraliste possible étant donné l'objectif de notre étude.

2.2 Analyse à chaud ou à froid

Dans le cadre de la détection d'anomalies sur les séries temporelles, il est important de faire la distinction entre la détection à chaud et l'analyse à froid. La détection à chaud se réfère à la détection en temps réel des anomalies au fur et à mesure qu'elles se produisent. Elle est souvent utilisée dans des systèmes de surveillance en temps réel où une réponse rapide est essentielle pour prévenir des événements indésirables. En revanche, l'analyse à froid se concentre sur l'examen rétrospectif des données pour identifier les anomalies après coup. Elle est généralement utilisée pour l'analyse post-mortem, la compréhension des causes profondes des anomalies et l'amélioration des systèmes de détection à chaud. Cet article se place dans le deuxième cas, on considère que l'on est dans le cas de l'analyse exploratoire (les modèles

de détection se sont pas déployés ultérieurement). C'est pourquoi, dans la partie expérimentations, toutes les données seront utilisées en apprentissage et les résultats présentés aussi en apprentissage.

2.3 Connaissances croisées

Lorsqu'il s'agit d'analyser plusieurs séries temporelles, croiser les connaissances permet d'extraire une information plus riche que de traiter chaque série individuellement. En combinant les différentes séries, il est possible de détecter des tendances, des corrélations ou des schémas qui ne seraient pas visibles lors de l'analyse de chaque série séparément. Cette approche permet d'obtenir une compréhension plus approfondie des interactions et des dynamiques entre les différentes séries temporelles. En exploitant ces informations croisées, il devient possible d'améliorer la précision des prévisions, d'identifier des anomalies ou des événements inhabituels. Cependant comme évoqué dans (Wu et Keogh, 2021) : "Chaque série temporelle doit être considérée comme totalement indépendante des autres, sauf à connaître les interactions ou clairement le domaine applicatif de génération des séries temporelles. Or dans cet article nous nous plaçons le plus possible de manière agnostique. De ce fait nous étudierons les séries temporelles une à une, sans croiser la moindre information commune. Chacune d'elle devient donc "un jeu de données" à part entière sur lequel une méthode de détection peut être appliquée et des résultats de performance récoltés.

2.4 Typologie des approches

La littérature considère trois grandes familles d'approches de détection d'anomalies : (i) supervisée, où l'on considère que chaque série temporelle, ou une portion de série temporelle, possède une étiquette (normale / anormale) ; (ii) semi-supervisée, où l'on fait l'hypothèse que le début de la série temporelle ne contient pas d'anomalie ; dans ce cas, le modèle est entraîné sur les données normales uniquement puis déployé sur la suite de la série temporelle à des fins de détection ; (iii) enfin l'approche non supervisée, où aucune hypothèse n'est faite : les anomalies peuvent être à n'importe quel moment dans la série temporelle et il n'y a pas de notion d'étiquettes (normale / anormale) disponibles au moment de l'apprentissage. Dans cet article, toujours dans l'idée d'être agnostique et d'une analyse à froid, nous nous plaçons dans le troisième cas.

2.5 Méthodes de détection dédiées au domaine temporel ou généralistes

Il existe de très nombreuses méthodes de détection d'anomalies pour séries temporelles univariées (Braei et Wagner, 2020a; Gupta et al., 2014) et tout autant pour les données tabulaires (Chandola et al., 2009). Il est intéressant de noter que ces deux domaines partagent certaines méthodes allant des plus simples (méthodes statistiques par exemple) aux plus élaborées. En effet, des méthodes conçues pour des données tabulaires fonctionnent souvent très correctement sur des données temporelles. On citera dans cet article "Isolation Forest" (IF) (Liu et al., 2008) et "Local Outlier Factor" (LOF) (Breunig et al., 2000). Le lecteur intéressé trouvera dans la présentation orale de (Boniol et al., 2023) une comparaison d'un assez grand nombres de méthodes (notamment transparents 125 à 130) et qui montre le très bon positionnement de IF et LOF sur des données temporelles. Étant donné l'objectif de l'étude présentée

dans cet article et du fait que ces deux méthodes fonctionnent correctement tant dans le domaine temporel que dans le domaine tabulaire, ce seront les méthodes que nous utiliserons dans la suite de notre étude comparative.

2.6 Découpage de la série temporelle par fenêtrage

Le dernier concept que nous avons besoin de poser est celui des fenêtres glissantes (et les paramètres associés) qui permet aux méthodes dédiées initialement aux données tabulaires d'être réutilisées sur des données temporelles. Pour une série temporelle donnée (et dans laquelle on souhaite détecter les anomalies), la méthode consiste à la "découper" en une succession de F fenêtres (de taille W) puis de les incorporer dans un tableau qui contiendra W colonnes et F lignes, se ramenant ainsi au "monde tabulaire". Le but sera donc ensuite de détecter quelles lignes du tableau contiennent une anomalie. La fenêtre que l'on fait passer sur la série temporelle peut être soit "sautante" (ou encore nommée sans chevauchement), soit glissante (avec chevauchement) avec dans ce cas un paramètre supplémentaire qui est la valeur du pas de décalage α (valeur de chevauchement). Une lecture assez attentive de la littérature Schmidl et al. (2022); Braei et Wagner (2020b) montre que la plupart des méthodes, si elles ne possèdent pas de boucle de rétroaction d'un expert, utilisent une fenêtre glissante et une valeur de 1 pour α, ce qui sera aussi notre cas dans la suite de cet article.

Reste à déterminer la taille de la fenêtre (W). Ce point est assez crucial mais assez étrangement peu débattu dans la littérature. La plupart des études règlent la valeur de W par validation croisée. A notre connaissance, un seul article s'est raisonnablement penché sur la question d'un réglage automatique de la valeur de W dans le cas de séries temporelles "périodiques" (Ermshaus et al., 2023), article dans lequel les auteurs testent plusieurs méthodes de détection de "saisonnalité / périodicité", et dans le cas de l'approche semi-supervisée. Dans notre cas, nous ne ferons pas l'hypothèse que les séries temporelles étudiées sont périodiques. La section expérimentale de cet article testera donc plusieurs valeurs de W et discutera de l'impact de cette valeur dans le cadre de notre étude.

2.7 Calcul de caractéristiques sur des séries temporelles

L'approche consiste à transformer les données du monde temporel en données tabulaires en utilisant des méthodes qui calculent des caractéristiques sur les séries temporelles. Les caractéristiques peuvent être diverses afin de capturer différentes propriétés de la série temporelle, telles que la saisonnalité, les tendances ou l'auto-corrélation, et peuvent donc être adaptées à différents domaines d'application. Cette transformation permet de capturer l'essence des données temporelles tout en les rendant compatibles avec les techniques d'analyse de données tabulaires plus traditionnelles.

Après une première proposition au travers de librairie HCTSA [2], plusieurs outils d'ingénierie de caractéristiques non supervisées ont été développés indépendamment dans différents langages de programmation : par exemple, CATCH22, FEATURETOOLS, TSFRESH, TSFEL, TSFEATURES, FEASTS, etc. L'étude assez approfondie présentée dans (Renault et al., 2023) (pour la tâche de classification) mais aussi celles présentées dans (Zhang et al., 2020; Teh et al., 2021) (pour la tâche de détection d'anomalies) montrent que TSFRESH (Christ

2. Pour des raisons de place nous ne pouvons pas mettre toutes les citations bibliographiques des librairies.

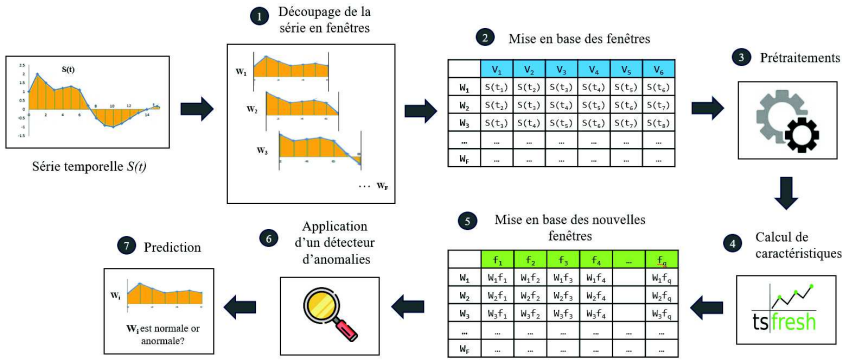

FIG. 1 – *Chaîne de traitement proposée.*

et al., 2018) est parmi les plus performantes. C'est celle que nous utiliserons dans la suite de cet article mais l'approche proposée pourrait en utiliser d'autres, voire les combiner.

3 Chaîne de traitement proposée

Muni de l'ensemble des concepts présentés ci-dessus, la chaîne de traitement proposée pour mener notre étude comparative sera celle présentée dans la figure 1. Cette chaîne de traitement est constituée de 7 étapes et appliquée à chaque série temporelle :

— Étape 1 : découpage de la série en fenêtres ($W = 6$ à titre illustratif dans la figure 1) ;
— Étape 2 : mise en base des F fenêtres, de taille W, obtenues (W_1 à W_F) ;
— Étape 3 : application potentielle de prétraitements ;
— Étape 4 : application de la librairie TSFRESH ;
— Étape 5 : mise en base des F fenêtres obtenues (W_1 à W_F) décrites par les q caractéristiques calculées à l'aide de TSFRESH ;
— Étape 6 : application du détecteur d'anomalie (IF ou LOF) ;
— Étape 7 : obtention de la prédiction pour chaque fenêtre pour calcul ultérieur de la qualité des résultats.

L'étude expérimentale décrite dans la section 4 suivante comparera les résultats obtenus avec ou sans les étapes 4 et 5, pour répondre à la question de recherche posée dans l'introduction de cet article.

4 Protocole expérimental

Cette section décrit les choix réalisés lors des expériences menées : choix des paramètres utilisateurs, taille des fenêtres glissantes (W), etc. L'ensemble des expériences menées peut être rejoué grâce à la mise à disposition du code produit (Sup, 2024). Certains test préliminaires qui seront évoqués ci-dessous ne seront pas présentés intégralement dans cet article pour des raisons de place. Le lecteur intéressé pourra néanmoins les retrouver dans le fichier "Matériel supplémentaire" disponible dans le GitHub juste ci-dessus mentionné (Sup, 2024).

4.1 Jeux de données utilisés

Actuellement, la détection d'anomalies fait l'objet de nombreuses recherches, et de nombreux ensembles de données de référence ont été établis pour permettre la comparaison des performances des nouveaux algorithmes. Parmi ces derniers, nous avons retenus ceux décrits ci-dessous. Nous n'avons parfois pas conservé toutes les séries temporelles de ces jeux de données. Nous en indiquons à chaque fois les raisons et la liste exacte (identifiants) des séries conservées, pour chaque jeu de données, est disponible dans (Sup, 2024).

SVDB (MIT-BIH Supraventricular Arrhythmia Database) : disponible sur le github de TimeEval. Ce jeu de données contient initialement 78 séries temporelles. Cependant 2 séries ont un taux de contamination (TC) supérieur à 50%. Nous ne gardons pas ces enregistrements dont le TC nous parait extrêmement élevé pour un taux d'anomalie. Au final, nous avons pour ce jeu de données 76 séries temporelles.

NAB (Numenta Anomaly Benchmark) : disponible sur le la page TimeEval. Créé par la société Numenta en 2015, il est aujourd'hui le deuxième benchmark le plus utilisé dans la littérature concernée. Il est composé de 58 séries temporelles réparties en 7 groupes dont 2 sont artificiels. Nous avons fait le choix de ne pas conserver ces données artificielles qui contiennent 11 séries. Les 5 autres regroupent des sujets divers et variés (dont les informations sont disponibles sur github). Parmi ces 5 groupes nous retirons une série qui ne contient aucune anomalie. Au final nous avons pour ce jeu de données 46 séries temporelles.

AIOPS 2018 : cet ensemble de données a été créé en 2018 pour le challenge AIOps. 29 séries ont été collectées auprès de diverses sociétés comme Sogou, Tencent ou eBay. Elles correspondent à des indicateurs de performance qui reflètent l'échelle, la qualité des services Web et l'état de santé d'une machine comme expliqué dans Ren et al. (2019). Les données utilisées proviennent de l'ensemble de référence Paparrizos et al. (2022) : TSB-UAD. De plus, il est à noter que nous avons joint la partie apprentissage et de test pour chaque série. Après une analyse des corrélations entre les 29 séries, nous notons que certaines d'entre elles sont corrélées, ce qui introduirait potentiellement un biais dans les tests statistiques menés lors de l'analyse des résultats. Aussi, nous ne gardons que les séries non corrélées ($r < 0.3$) ; ce qui au final nous amène à sélectionner 13 séries pour les expérimentations.

NormA : cet ensemble de données est également issu du github de TimeEval. Il est composé de 21 séries temporelles, dont 14 synthétiques que nous écartons. Les 7 séries réelles peuvent être regroupées en 4 catégories comme indiqué sur la page de NormA. Sur ces 7 séries réelles, les 3 premières sont très corrélées ; nous ne conservons au final que 5 séries temporelles pour ce jeu de données.

UCR Anomaly Archive : L'ensemble de données UCR Anomaly Archive a été publié en 2020 comme alternative à plusieurs benchmarks jugés défaillants par (Wu et Keogh, 2021). Il est disponible sur UCR. Pour ce dernier nous avons fait le choix de n'écarter que les trois séries qui sont vraiment plus longues que les autres (pour des raisons de temps de calcul) et donc d'utiliser 247 séries temporelles au total.

Au final l'ensemble de nos 5 jeux de données représente une large diversité de problèmes, de taux d'anomalie et de problématiques (voir Table 1). Lors de la présentation des résultats, les jeux de données AIOPS et NormA contenant peu de série temporelles seront fusionnés de manière à pouvoir calculer des tests statistiques, tel que celui de Wilcoxon (Wilcoxon, 1992), qui requiert un minimum de valeurs pour pouvoir être utilisé.

TAB. 1 – *Résumé des informations des bases de données utilisées.*

Nom	#séries	Longueur		Fréq (Hz)	% d'anomalies		Domaines
		Min	Max		Min	Max	
SVDB	76	230 400	230 400	0.008	0.34	45.54	ECG
NAB	46	1 127	22 695	variée	8.30	10.29	serveurs, tweets, trafic, publicités
AIOPS	13	16 441	295 414	[0.003-1.7]	0.06	7.50	indicateurs de performance
NormA	5	2 000	35 040	inconnue	3.08	9.13	aérospatiale, santé, gestuelle, électricité
UCR	247	6 674	300 262	variée	0.0005	4.9	médecine, météorologie, biologie, industrie

4.2 Taille des fenêtres glissantes

Comme évoqué plus haut, ce point est assez crucial mais assez étrangement peu débattu dans la littérature. À notre connaissance, un seul article s'est raisonnablement penché sur la question d'un réglage automatique de la valeur de W dans le cas de séries temporelles "périodiques" (Ermshaus et al., 2023). Dans notre cas où nous souhaitons rester agnostique et sans boucle de rétroaction, nous avons décidé de tester simplement 4 tailles de fenêtres correspondant à des valeurs observées dans la littérature. Nous sommes partis d'une petite taille $W = 32$ puis nous avons doublé sa taille, en espérant augmenter la quantité d'information capturée, plusieurs fois. Au final, les valeurs testées pour W sont 32, 64, 128, et 256. Le but des expérimentations ne sera pas de déterminer la taille de fenêtre optimale mais de confirmer que les conclusions ne dépendent pas d'une taille très spécifique.

4.3 Paramétrage et utilisation de TSFRESH et des détecteurs

La librairie TSFRESH (Christ et al., 2018) (acronyme de *Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests*) peut être utilisée en créant plus ou moins de caractéristiques (la valeur de q mentionnée section 3). Tous les tests que nous avons effectués montrent de meilleurs résultats en poussant au plus le curseur et donc avec un grand nombre de caractéristiques calculées par la librairie (soit donc sa version "Efficient" et $q = 777$ caractéristiques). On indique néanmoins que parfois, par exemple avec $W = 32$, il n'est pas possible pour la librairie de calculer toutes les caractéristiques demandées. Dans ce cas, nous avons retiré les colonnes concernées (à valeur non renseignée) de la table produite à l'étape 5 de la chaîne de traitement présentée section 3. Deux détecteurs d'anomalies pour données tabulaires ayant fait leurs preuves sur les séries temporelles sont utilisés : Isolation Forest (IF) et Local Outlier Factor (LOF). Dans toutes les expérimentations menées ci-après, ils ont été utilisés dans leur version de la librairie PyOD (version 1.1.3) avec leur paramétrage par défaut.

4.4 Prétraitements

Des tests préliminaires ont été effectués dans le but d'analyser l'intérêt qu'il y aurait à "normaliser" les valeurs temporelles contenues dans les fenêtres glissantes avant d'appliquer les deux approches : 1) détection d'anomalies à partir de la représentation initiale (appelée "TS" dans Table 2) et 2) détection d'anomalies à partir de la nouvelle représentation de caractéristiques calculées par TSFRESH (appelée "FE" dans Table 2). Pour cela, une étude comparative a été menée entre "ne rien faire" (Sans Normalisation) et 3 méthodes de normalisation usuelles à savoir (i) Min-Max, (ii) Médiane-IQR et (iii) Moyenne-Ecart type.

TAB. 2 – *Comparaison des rangs moyens des normalisations selon la taille de fenêtre utilisée (en combinant les résultats de IF et LOF).*

			Sans Normalisation	MinMax	MedianeIQR	MeanStd
SVDB	32	TS	2.250	2.036	3.063	2.651
		FE	1.474	2.250	3.135	3.141
	64	TS	2.211	2.227	2.908	2.655
		FE	1.750	2.260	2.934	3.056
	128	TS	2.174	2.243	2.609	2.974
		FE	2.049	2.418	2.474	3.059
	256	TS	2.079	2.638	2.424	2.859
		FE	2.260	2.563	2.230	2.947
AIOPS + NormA	32	TS	1.806	2.583	2.472	3.139
		FE	2.236	2.708	2.750	2.306
	64	TS	1.903	2.431	2.722	2.944
		FE	2.458	2.722	2.625	2.194
	128	TS	1.722	2.583	2.667	3.028
		FE	2.500	2.722	2.417	2.361
	256	TS	1.736	2.944	2.319	3.000
		FE	2.250	2.944	2.250	2.556

Cette étude est réalisée en se basant sur (i) trois des jeux de données décrits ci-dessus : SVDB, NormA et AIOPS parmi les cinq sélectionnés et (ii) le paramètre de la librairie TS-FRESH fixé à "minimal" ($q = 10$). [3]

Les résultats présentés dans la Table 2, agrégés quelle que soit la méthode de détection (IF ou LOF) et versus la taille de fenêtre [4], ne montrent aucun apport des normalisation testées. Au contraire, l'absence de normalisation donne les meilleurs résultats. De ce fait dans la section suivante présentant les résultats (Section 5), seuls ceux sans normalisation sont présentés. Une analyse plus approfondie des résultats peut être trouvée dans le matériel supplémentaire (Sup, 2024) selon l'axe "méthode de détection" (IF ou LOF) sans en changer les conclusions.

4.5 Critère d'évaluation des résultats

La littérature propose de nombreux critères d'évaluation tels que la précision, la mesure du critère F1, le critère "PA%K" (Kim et al., 2021), ou l'AUC (*Area Under the receiver operating characteristics Curve*) (Fawcett, 2006). Dans notre cas, toujours dans l'idée de ne pas faire appel dans cette étude à un expert, nous avons décidé de ne pas utiliser un critère qui requiert une valeur de seuil. En effet, ce seuil peut être un paramètre difficile à sélectionner et/ou qui demande de partir d'une heuristique sur la distribution des scores d'anomalie. De ce fait, nous avons sélectionné l'AUC.

Le lecteur notera que dans ce cas, mais aussi pour les autres critère potentiels, il faudra pouvoir comparer une vérité terrain à la prédiction effectuée lors de l'étape 7 de notre chaîne de traitement. Pour la classe prédite, cette information est donnée par la méthode de détection utilisée, ici IF ou LOF. Pour la classe à prédire, nous avons opéré comme suit (ce qui correspond à l'usage observé) : lors de l'étape 1 et du découpage de la série temporelle en fenêtre, si une anomalie existe dans la fenêtre alors toute la fenêtre est étiquetée anormale et inversement dans le cas opposé. De ce fait, il est possible de comparer pour chaque fenêtre la classe prédite

3. L'ensemble NormA n'étant composé que de cinq séries temporelles, il est ici combiné dans les résultats avec celui d'AIOPS.

4. Le lecteur pourra trouver des résultats complémentaires dans (Sup, 2024) versus la méthode de détection corroborant ceux ici présentés.

TAB. 3 – *Comparaison des rangs moyens des méthodes, par jeu de données, selon la taille de fenêtre utilisée.*

		Isolation Forest			Local Outlier Factor		
		TS	FE	p-value	TS	FE	p-value
SVDB	32	1.882	1.118	$\mathbf{1.522 \times 10^{-12}}$	1.329	1.671	$\mathbf{2.066 \times 10^{-3}}$
	64	1.855	1.145	$\mathbf{5.375 \times 10^{-9}}$	1.480	1.520	4.454×10^{-1}
	128	1.842	1.158	$\mathbf{5.553 \times 10^{-10}}$	1.750	1.250	$\mathbf{2.490 \times 10^{-6}}$
	256	1.592	1.408	1.161×10^{-1}	1.684	1.316	$\mathbf{5.541 \times 10^{-6}}$
AIOPS + NormA	32	1.889	1.111	$\mathbf{1.930 \times 10^{-3}}$	1.556	1.444	8.317×10^{-1}
	64	1.778	1.222	$\mathbf{1.930 \times 10^{-3}}$	1.556	1.444	7.660×10^{-1}
	128	1.778	1.222	$\mathbf{1.930 \times 10^{-3}}$	1.417	1.583	4.925×10^{-1}
	256	1.833	1.167	$\mathbf{4.745 \times 10^{-3}}$	1.389	1.611	1.674×10^{-1}
NAB	32	1.652	1.348	5.059×10^{-2}	1.261	1.739	$\mathbf{8.504 \times 10^{-4}}$
	64	1.630	1.370	$\mathbf{1.727 \times 10^{-2}}$	1.261	1.739	$\mathbf{7.430 \times 10^{-5}}$
	128	1.674	1.326	$\mathbf{5.525 \times 10^{-3}}$	1.217	1.783	$\mathbf{1.261 \times 10^{-4}}$
	256	1.641	1.359	$\mathbf{1.723 \times 10^{-2}}$	1.283	1.717	$\mathbf{3.163 \times 10^{-3}}$
UCR	32	1.826	1.174	$\mathbf{2.581 \times 10^{-29}}$	1.156	1.844	$\mathbf{5.159 \times 10^{-22}}$
	64	1.844	1.156	$\mathbf{9.264 \times 10^{-32}}$	1.170	1.830	$\mathbf{7.227 \times 10^{-26}}$
	128	1.848	1.152	$\mathbf{1.792 \times 10^{-32}}$	1.223	1.777	$\mathbf{2.247 \times 10^{-20}}$
	256	1.796	1.204	$\mathbf{8.983 \times 10^{-27}}$	1.243	1.757	$\mathbf{1.055 \times 10^{-19}}$

par un score de confiance délivré par IF ou LOF et la vérité terrain. L'ensemble des comparaisons nous permet de calculer l'AUC pour chaque série temporelle (de manière individuelle) des jeux de données utilisés. L'ensemble de ces AUC nous permettra aussi de mener des tests statistiques et de présenter des résultats en terme de rang ou de diagramme critique.

5 Résultats

5.1 Résultats détaillés par paires de méthodes

On présente dans la Table 3 les résultats obtenus : avec (FE) ou sans (TS) l'utilisation de TSFRESH dans la chaîne de traitement proposée ; avec les détecteurs Isolation Forest (IF) ou Local Outlier Factor (LOF). Ces résultats sont détaillés par jeu de données et par taille de fenêtre. Les valeurs présentées sont celles des rangs moyens obtenus sur l'ensemble des séries temporelles de chaque jeu de données. Les valeurs de p-value sont issues du test de Wilcoxon et une valeur en gras indique si la différence de rang est significative.

Les résultats sont relativement clairs : (i) pour Isolation Forest sur 16 résultats (4 jeux de données x 4 tailles de fenêtre), la méthode proposée FE avec l'utilisation de TSFRESH a 16 fois un meilleur rang moyen, dont 14 sont significativement meilleurs (ii) pour Local Outlier Detection, la méthode proposée FE n'a que 4 fois un meilleur rang moyen dont seulement 2 sont significativement meilleurs. La chaîne de traitement proposée profite à IF, basé sur des arbres (donc des rangs de valeurs), mais pas à LOF, basé sur des estimations de densités (et donc des calculs de distances).

Il est vraisemblable que pour LOF la taille du vecteur produit par TSFRESH conduise à un problème de dimension lors des calculs de distance utilisées par cette méthode (Ortner et al., 2017; Zimek et al., 2012) alors qu'IF lui ne subit pas ce problème. La performance de LOF soulève néanmoins des interrogations mais nous pensons qu'une normalisation des caractéristiques (de manière verticale) pourrait améliorer ses performances dans de futurs travaux. Il

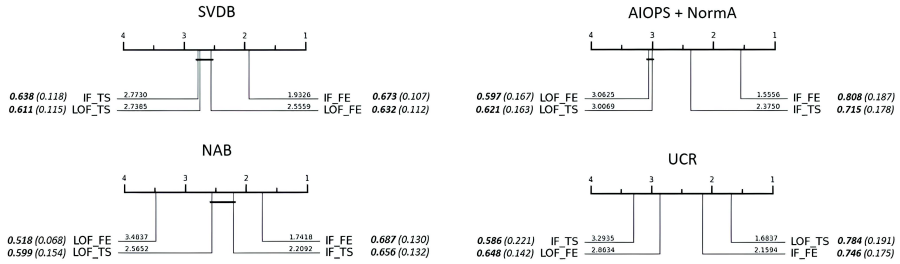

FIG. 2 – *Diagrammes critiques pour chaque jeu de données, moyenné sur les 4 tailles de fenêtres.*

s'agit d'un modèle qui requiert plus d'attention par rapport à Isolation Forest (basé sur des statistiques de rang) qui n'a eu besoin ni de normalisation, ni d'optimisation de ses hyperparamètres. On peut aussi imaginer que le réglage de la valeur de k dans LOF pourrait apporter une amélioration, mais, dans le cas sans boucle de rétroaction avec un expert du domaine, nous n'avons pas exploré cette possibilité. Ces points restent à explorer dans de futurs travaux pour le confirmer.

5.2 Résultats par valeur d'AUC et entre détecteurs

La figure 2 présente le diagramme critique (DC) des méthodes (i) par jeu de données puis (ii) avec ou sans l'utilisation des caractéristiques calculées et (iii) quelle que soit la taille de la fenêtre glissante. Dans chacune des sous-figures et pour chaque barre du diagramme critique, on trouve dans cet ordre : le rang moyen de la méthode, le nom de la méthode (IF ou LOF) préfixé de TS ou FE (respectivement pour "Time Series" initiale ou FE pour "Feature Engineering" via TSFRESH), puis la valeur d'AUC moyenne correspondante avec enfin l'écart type de l'AUC entre parenthèses. Le lecteur intéressé trouvera dans (Sup, 2024) d'autres résultats avec d'autres axes d'analyse.

On note à nouveau que, de manière générale, c'est Isolation Forest qui profite le plus des caractéristiques calculées à l'aide de TSFRESH prenant de manière significative la première place sur 3 des 4 jeux de données. Pour LOF, les résultats sont bien plus contrastés, la création de caractéristiques n'apportant pas ou peu d'amélioration vis-à-vis de la représentation temporelle initiale [5]. Ces résultats confirment ceux présentés dans la Table 3. Enfin, LOF appliqué à des données temporelles obtient le meilleur résultat sur le jeu de données UCR. Ce jeu a la particularité de contenir très peu d'anomalies (voir Section 4.1), avec des anomalies qui n'apparaissent qu'en deuxième partie de chaque série temporelle. Cela a un impact remarquable sur IF associé à la chaîne de traitement proposée dans cet article : bien que se positionnant en seconde place, IF s'améliore très fortement, avec un AUC moyen qui passe de 0.586 à celui de 0.746, soit +36%.

5. Dans la chaîne de traitement, nous n'avons pas cherché à normaliser les données issues de TSFRESH. Ce point serait à tester pour déterminer si c'est l'une des raisons de ces résultats pour LOF.

6 Conclusion

Dans le contexte de la détection d'anomalies au sein d'une série temporelle, cet article a proposé l'idée de la transformation des séries temporelles du domaine temporel vers un espace de représentation alternatif tabulaire, puis d'appliquer des détecteurs d'anomalies dédiés aux données tabulaires. L'idée est de considérer le processus d'extraction de caractéristiques comme étant une source de connaissances et d'information utile à la détection. La question posée était : peut-on obtenir de meilleurs résultats dans l'espace des caractéristiques calculées ou vaut-il mieux rester dans la représentation temporelle initiale ? Au cours d'expériences approfondies sur 5 jeux de données et deux détecteurs (IF et LOF), nous avons comparé les deux approches. Les résultats expérimentaux montrent une amélioration des résultats de manière significative pour IF mais pas dans le cas de LOF. Dans de futurs travaux il sera intéressant (i) d'étendre le nombre de détecteur dans le comparatif (ii) de tester d'autres librairies de création de caractéristiques (voir de les combiner) (iii) de réfléchir au réglage de la valeur de la taille de la fenêtre, même si cette dernière n'est pas inhérente à la chaîne de traitement proposée.

Références

Bondu, A., Y. Achenchabe, A. Bifet, F. Clerot, A. Cornuejols, J. Gama, G. Hebrail, V. Lemaire, et P.-F. Marteau (2022). Open challenges for machine learning based early decision-making research. *SIGKDD Explor. Newsl. 24*(2), 12–31.

Boniol, P., J. Paparrizos, et T. Palpanas (2023). Tutorial : New trends in time series anomaly detection. In *International Conference on Extending Database Technology*. [Transparents ...] [Vidéo ...

Braei, M. et S. Wagner (2020a). Anomaly detection in univariate time-series : A survey on the state-of-the-art. *ArXiv abs/2004.00433*.

Braei, M. et S. Wagner (2020b). Anomaly detection in univariate time-series : A survey on the state-of-the-art. *arXiv preprint arXiv :2004.00433*.

Breunig, M. M., H.-P. Kriegel, R. T. Ng, et J. Sander (2000). Lof : identifying density-based local outliers. *SIGMOD Rec. 29*(2), 93–104.

Chandola, V., A. Banerjee, et V. Kumar (2009). Anomaly detection : A survey. *ACM Comput. Surv. 41*(3).

Christ, M., N. Braun, J. Neuffer, et A. W. Kempa-Liehr (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh - A python package). *Neurocomputing 307*, 72–77.

Ermshaus, A., P. Schäfer, et U. Leser (2023). Window size selection in unsupervised time series analytics : A review and benchmark. In *ECML PKDD Workshop, AALTD*, pp. 83–101. Springer-Verlag.

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters 27*(8).

Gupta, M., J. Gao, C. C. Aggarwal, et J. Han (2014). Outlier detection for temporal data : A survey. *IEEE Transactions on Knowledge and Data Engineering 26*(9), 2250–2267.

Kim, S., K. Choi, H.-S. Choi, B. Lee, et S. Yoon (2021). Towards a rigorous evaluation of time-series anomaly detection. In *AAAI Conference on Artificial Intelligence*.

Liu, F. T., K. M. Ting, et Z.-H. Zhou (2008). Isolation forest. In *2008 IEEE International Conference on Data Mining (ICDM)*, pp. 413–422.

Ortner, T., P. Filzmoser, M. Rohm, S. Brodinova, et C. Breiteneder (2017). Local projections for high-dimensional outlier detection. *METRON 79*, 189 – 206.

Paparrizos, J., Y. Kang, P. Boniol, R. S. Tsay, T. Palpanas, et M. J. Franklin (2022). Tsb-uad : an end-to-end benchmark suite for univariate time-series anomaly detection. *Proceedings of the VLDB Endowment 15*(8), 1697–1711.

Ren, H., B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, et Q. Zhang (2019). Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 3009–3017.

Renault, A., A. Bondu, V. Lemaire, et D. Gay (2023). Automatic feature engineering for time series classification : Evaluation and discussion. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE.

Schmidl, S., P. Wenig, et T. Papenbrock (2022). Anomaly detection in time series : A comprehensive evaluation. *Proceedings of the VLDB Endowment 15*(9), 1779–1797.

Sup (2024). Matériel supplémentaire et github de l'article "Construction non-supervisée de variables pour la détection d'anomalies dans les séries temporelles". [Github ...] [Résultats supplémentaires...].

Teh, H. Y., I. Kevin, K. Wang, et A. W. Kempa-Liehr (2021). Expect the unexpected : unsupervised feature selection for automated sensor anomaly detection. *IEEE Sensors Journal 21*(16), 18033–18046.

Wilcoxon, F. (1992). *Individual Comparisons by Ranking Methods*, pp. 196–202.

Wu, R. et E. J. Keogh (2021). Current time series anomaly detection benchmarks are flawed and are creating the illusion of progress. *IEEE transactions on knowledge and data engineering 35*(3), 2421–2429.

Zhang, W., X. Dong, H. Li, J. Xu, et D. Wang (2020). Unsupervised detection of abnormal electricity consumption behavior based on feature engineering. *Ieee Access 8*, 55483–55500.

Zimek, A., E. Schubert, et H.-P. Kriegel (2012). A survey on unsupervised outlier detection in high-dimensional numerical data. *Statistical Analysis and Data Mining : The ASA Data Science Journal 5*.

Summary

To accurately detect anomalies and without prior knowledge in a time series, is it better to build the detectors from the initial temporal representation, or to compute a new (tabular) representation using an existing automatic variable construction library? In this article, we answer this question by conducting an in-depth experimental study for two popular detectors (Isolation Forest and Local Outlier Factor). The results obtained, for 5 different datasets, show that the new representation, calculated using the *tsfresh* library, allows Isolation Forest to significantly improve its performance.