De l'importance de la validation croisée

Célestin Eve[1,2] Thomas Moreau[1] Gaël Varoquaux[2]

[1]Inria MIND, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France

[2]Inria SODA, 1 rue Honoré d'Estienne d'Orves, 91120 Palaiseau, France

Résumé – L'analyse comparative joue un rôle essentiel dans la recherche en apprentissage automatique. La validation croisée, c'est-à-dire une évaluation répétée des algorithmes sur des sous-ensembles de données afin de comparer leurs performances moyennes, est couramment utilisée à cette fin. Dans cette étude, nous analysons l'impact de différentes procédures de validation (absence de validation croisée, validation croisée à k blocs répétée et validation croisée par permutations aléatoires) sur la fiabilité du classement des algorithmes et le coût computationnel, en utilisant un jeu de données simulé. Nos résultats suggèrent que les procédures de validation présentant le même nombre de partitions produisent des performances comparables et que la décroissance des rendements liés à l'augmentation du nombre de partitions est plus lente qu'attendu.

Abstract – Benchmarking machine learning algorithms is crucial for scientific progress. Cross-validation, a common evaluation tool, enables repeated performance comparisons across dataset subsets. This study investigates how validation procedures – non-cross-validation, repeated k-fold, and random permutation – affect algorithm ranking reliability and computational cost on a simulated dataset. We analyze the impact of the number of splits on these factors. Results indicate that procedures with equal splits yield similar performance, diminishing returns from increasing splits occurring slower than expected.

1 Introduction

L'évaluation comparative des algorithmes d'apprentissage est essentielle pour le progrès scientifique en apprentissage automatique. Comparer un algorithme récemment développé aux méthodes à l'état de l'art permet d'obtenir une première indication de son efficacité : parmi quelques exemples récents, on peut citer [3] dans le domaine des interfaces cerveau-machine, [9] en adaptation de domaine, ou encore [4] pour l'évaluation comparative des grands modèles de langage.

Plusieurs outils ont été développés afin de faciliter la transparence et la reproducibilité de l'analyse comparative. D'abord, par l'établissement de benchmarks de référence, qui fournissent un cadre standardisé pour évaluer les performances des différentes méthodes [7]. Parmi les plus connus figurent ImageNet pour la classification d'images [5] ou les jeux Atari 2600 pour l'apprentissage par renforcement [1]. Par ailleurs, des plateformes compétitives comme Kaggle ont émergé, permettant une comparaison directe des algorithmes sur un même jeu de données. Des outils comme Benchopt [10] permettent également une évaluation comparative transparente et efficace des méthodes d'optimisation.

Cependant, seul un nombre limité de domaines utilise systématiquement des benchmarks de référence. Dans de nombreuses publications en apprentissage automatique, les processus d'évaluation sont conçus sur mesure pour une étude spécifique. Malheureusement, l'évaluation comparative est un processus intrinsèquement variable à plusieurs niveaux — que ce soit en termes de segmentation des données, d'ordre des échantillons, d'initialisation des poids ou d'optimisation des hyperparamètres [2] — ce qui peut entraîner des classements incohérents et des conclusions trompeuses sur l'efficacité des algorithmes étudiés. Il est donc primordial d'identifier et de prendre en compte avec précision toutes les sources pouvant invalider les comparaisons numériques, afin d'obtenir des résultats fiables et robustes.

Pour tirer des conclusions pertinentes, il est crucial d'adopter une méthodologie valide et rigoureuse. Or, ce n'est pas toujours le cas : dans le domaine de la détection d'anomalies dans les séries temporelles, on observe une utilisation récurrente de métriques d'évaluation défectueuses et de pratiques de benchmarking incohérentes [13].

Un autre défi majeur lié à l'évaluation comparative est son coût computationnel, qui peut augmenter drastiquement lorsque les méthodes existantes doivent être réévaluées à chaque nouvelle avancée. Les benchmarks de référence, en imposant une méthodologie fixe, permettent de réutiliser les évaluations des méthodes existantes, réduisant ainsi le besoin d'effectuer des calculs coûteux à plusieurs reprises. À l'inverse, les nouveaux processus d'évaluation exigent une réévaluation complète de toutes les méthodes selon de nouveaux critères, ce qui accroît le temps et les ressources nécessaires. Ainsi, le développement de benchmarks fiables et largement acceptés est crucial tant pour garantir la précision et la cohérence des évaluations que pour minimiser les coûts et l'empreinte environnementale associés à l'analyse comparative.

Parmi les outils disponibles permettant de réduire la variabilité de l'analyse comparative et de vérifier la capacité d'un algorithme d'apprentissage à généraliser à de nouvelles données figure la validation croisée : l'ensemble de données est partitionné, l'algorithme est entraîné et validé, puis est enfin évalué. On répète cette opération plusieurs fois et on moyenne les résultats obtenus avant de comparer les performances de nos différents algorithmes.

Dans ce papier, nous proposons une méthode pour évaluer la stabilité des classements calculés par des procédures de validation, et nous l'évaluons dans un cadre de régression sur des données synthétiques. Nos résultats suggèrent que les procédures de validation avec le même nombre de partitions (« splits ») ont des performances similaires, et qu'augmenter le nombre de partitions ralentit moins les rendements qu'attendu.

2 Comment classer ?

Dans cette section, nous nous intéressons à la définition des classements des fonctions de décision et des algorithmes d'apprentissage.

2.1 Fonctions de décision

Soit $\mathcal{D}$ un ensemble de données composé de paires $(X, Y) \in \mathcal{X} \times \mathcal{Y}$. Typiquement, $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \{0, \dots, n-1\}$ en classification et $\mathcal{X} \times \mathcal{Y} = \mathbb{R}^d \times \mathbb{R}$ en régression.

Une fonction g qui associe chaque $X \in \mathcal{X}$ à $g(X) \in \mathcal{Y}$ est appelée une fonction de décision – ou un *estimateur*. Plus précisément, elle peut être appelée un classifieur ou un régresseur. *Évaluer* les fonctions de $\mathcal{Y}^{\mathcal{X}}$ (l'ensemble des fonctions ayant pour ensemble de départ $\mathcal{X}$ et pour ensemble d'arrivée $\mathcal{Y}$) consiste à les classer en leur attribuant des scores relatifs.

Intuitivement, nous voulons que notre classement de fonctions soit robuste : il ne doit pas varier en fonction des caractéristiques des données telles que la taille du jeu de données, par exemple. Un classement sera considéré comme robuste s'il respecte l'ordre attendu qui serait obtenu si nous avions accès à la distribution de la population.

Soit $m : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ une métrique. Nous noterons :

$$S_{\mathcal{D}}(g) = \frac{1}{|\mathcal{D}|} \sum_{(X,Y)\in\mathcal{D}} m(g(X), Y) \tag{1}$$

la fonction de score associée à m et à un jeu de données $\mathcal{D} \subset \mathcal{X} \times \mathcal{Y}$, qui prend en entrée une fonction de décision g et retourne le score de g sur le jeu de données $\mathcal{D}$. Le *score oracle*, $S^*(g)$, est défini comme le score attendu sur la distribution de la population d, c'est-à-dire :

$$S^*(g) = \mathbb{E}_{\mathcal{D}\sim d}[S_{\mathcal{D}}(g)] \tag{2}$$

où d représente la distribution sous-jacente de la population. En régression, la métrique la plus utilisée est le coefficient de détermination R^2, tandis que c'est l'exactitude (« accuracy ») en classification.

Le *classement oracle* des fonctions de décision évaluées est le classement respectant l'ordre des scores oracles. Si nous évaluons trois fonctions de décision g_i, g_j, g_k avec des scores oracles $S^*(g_i) < S^*(g_j) < S^*(g_k)$, alors le classement oracle sera :

$$1.g_k \qquad 2.g_j \qquad 3.g_i$$

Le classement oracle est précisément « l'ordre qui serait obtenu si nous avions accès à la distribution de la population » mentionné plus haut.

Classer les estimateurs est une tâche non triviale. La principale difficulté est d'évaluer les fonctions en se basant uniquement sur quelques observations des valeurs de ces fonctions, et non sur l'espérance sur la distribution de la population. On parle d'*erreur de généralisation*.

Ce problème de classement peut être vu comme un problème d'optimisation : nous voulons choisir comme meilleur estimateur la fonction g^* qui maximise la métrique (ou la minimise – dans ce cas, la métrique considérée est appelée une *perte*) sur la distribution de la population. Formellement :

$$g^* \in \underset{g\in\mathcal{Y}^{\mathcal{X}}}{\operatorname{argmax}}\, S^*(g) \tag{3}$$

Mais empiriquement, nous ne pouvons aborder que l'approximation de ce problème sur un jeu de données $\mathcal{D}$:

$$\hat{g} \in \underset{g\in\mathcal{Y}^{\mathcal{X}}}{\operatorname{argmax}}\, S_{\mathcal{D}}(g) \tag{4}$$

en espérant que $\hat{g}$ sera d'une manière ou d'une autre proche de g^*.

Nous obtenons donc une approximation du classement oracle. Si nous utilisons les mêmes notations que ci-avant, nous nous attendrions ici à ce que notre classement soit $S_{\mathcal{D}}(g_i) < S_{\mathcal{D}}(g_j) < S_{\mathcal{D}}(g_k)$.

Mais la généralisation de (4) à (3) peut être compliquée par plusieurs facteurs – « compliquée » signifiant que le classement empirique des fonctions peut être très différent de leur classement oracle.

2.2 Algorithmes d'apprentissage

Les fonctions de décision peuvent être générées à l'aide d'*algorithmes d'apprentissage*. Comme les algorithmes d'apprentissage peuvent avoir des comportements stochastiques, leur fonction de décision en sortie peut varier d'une itération de l'algorithme à une autre [2, 12]. On peut alors s'intéresser à quel algorithme d'apprentissage produira des fonctions plus robustes sur un ensemble de données spécifique [6].

Il y a deux difficultés principales dans l'évaluation des performances des algorithmes d'apprentissage :

- L'évaluation des algorithmes d'apprentissage n'est pas aussi directe que l'évaluation des fonctions de décision : le classement des algorithmes d'apprentissage doit être approximé à travers des classements des fonctions de décision.
- Généralement, le classement oracle des algorithmes d'apprentissage étudiés est inconnu.

3 Configuration expérimentale

L'objectif de nos expériences était de tester différentes configurations de validation croisée et de quantifier les avantages ou inconvénients du choix d'une procédure par rapport à une autre. Nous avons réalisé nos expériences dans un cadre de régression, sur un jeu de données synthétiques à distribution gaussienne.

Dans ces expériences, nous avons divisé le jeu de données en deux ensembles : l'*ensemble d'étude* et l'*ensemble de référence*, en faisant varier la taille du premier pour différentes expériences, et en fixant une taille de 100 000 échantillons pour le second. L'ensemble d'étude a ensuite été subdivisé en ensembles d'entraînement, de test et de validation. Cela est expliqué dans la figure 1. Nous avons choisi une taille de validation et une taille de test de 20% de l'ensemble d'étude chacune dans chaque expérience.

Nous avons considéré différentes procédures pour subdiviser l'ensemble d'étude : *train_test_split* (partitionnement aléatoire), *RepeatedKFold* (partitionnement du jeu en k blocs, utilisés un à un pour test et les autres entraînement ; on répète l'opération et moyenne les résultats) et *ShuffleSplit* (partionnement aléatoire répété) avec un nombre variable de partitions et de répétitions.

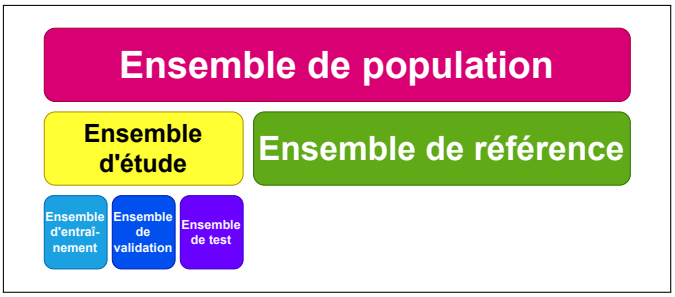


FIGURE 1 : Diagramme expliquant la décomposition du jeu de données dans nos expériences.

Nous avons comparé les performances des fonctions de décision générées par les algorithmes d'apprentissage ExtraTrees [8], avec son hyperparamètre $n_{estimators}$ prenant des valeurs entre 2, 20 et 200. Ainsi, le classement oracle des algorithmes d'apprentissage étudiés est connu :

1. ExtraTrees($n_{estimators} = 200$)
2. ExtraTrees($n_{estimators} = 20$)
3. ExtraTrees($n_{estimators} = 2$)

Ce classement sera désigné ci-après par 200-20-2, et on généralisera cette notation aux autres classements possibles.

Chaque configuration expérimentale a été exécutée 1 000 fois, conduisant à 1 000 classements calculés sur l'ensemble de test, et 1 000 classements calculés sur l'ensemble de référence.

Nous avons également réalisé exactement les mêmes expériences avec trois algorithmes d'apprentissage ExtraTrees avec un $n_{estimators}$ constant de 10, mais en changeant uniquement le $random\_state$ entre 0, 42 et 120. Ces expériences sont appelées le *contrôle négatif* et servent à vérifier notre configuration expérimentale, car parmi ces trois algorithmes d'apprentissage, aucun ne devrait se démarquer : leur classement oracle est une égalité entre les trois.

Ces expériences ont été implémentées en Python, utilisant les algorithmes d'apprentissage et les procédures de partitionnement des données de la bibliothèque scikit-learn [11]. Le code est disponible à l'adresse `https://github.com/ceelestin/benchmark_regression`.

4 Résultats

Pour chaque procédure, nous déterminons la taille minimale de l'ensemble d'étude à laquelle un classement particulier est établi dans 95% des exécutions. La figure 2 fournit deux exemples de cette analyse, avec deux procédures différentes.

Nous observons d'abord que, pour chaque procédure – que ce soit sur l'ensemble de test ou sur l'ensemble de référence – le classement oracle se distingue clairement.

De plus, le contrôle négatif montre que lorsqu'il n'existe pas d'ordre établi parmi les algorithmes testés, aucun classement ne se distingue ni sur l'ensemble de test ni sur l'ensemble de référence, ce qui valide notre configuration empirique.

Ensuite, plus le nombre total de partitions (c'est-à-dire le nombre de partitions multiplié par le nombre de répétitions) dans une procédure est élevé, plus la taille de l'ensemble d'étude nécessaire pour atteindre une proportion de classement de 95% est faible. Ce comportement est attendu par la nature même de la validation croisée.

Nous observons également que les classements calculés sur l'ensemble de référence convergent plus rapidement vers le classement oracle que ceux calculés sur l'ensemble de test. Rappelons que l'ensemble de référence a une taille fixe de 100 000 échantillons, tandis que la taille de l'ensemble de test varie et représente toujours 20% de l'ensemble d'étude. Par conséquent, l'ensemble de référence reste nettement plus grand que l'ensemble de test et sert de *quasi-oracle* pour le classement des fonctions de décision de sortie.

Globalement, ces observations préliminaires conduisent à la même conclusion : un ensemble de données plus grand et un nombre plus élevé de partitions en validation croisée améliorent la robustesse du classement. Cependant, l'augmentation du nombre de partitions augmente également le temps de calcul. La sélection du nombre optimal de partitions nécessite donc un équilibre entre la confiance dans le classement et le coût computationnel. La figure 3 reflète ce compromis.

Enfin, ces expériences nous permettent de quantifier combien d'échantillons la validation croisée permet d'économiser. En d'autres termes, elles déterminent combien de données supplémentaires sont nécessaires pour qu'une procédure train_test_split produise des conclusions aussi robustes qu'une procédure de validation croisée donnée avec un nombre spécifique d'échantillons de données. Cela est illustré dans la figure 4. On parle de « gain d'échantillons » : partitionner 20 fois l'ensemble d'étude équivaut à multiplier conceptuellement le nombre d'échantillons de données disponibles par 10. Cette figure, ainsi que la figure 3, illustrent qu'à nombre total de partitions égal, les deux procédures que sont la validation croisée à k blocs répétée et la validation croisée par permutations aléatoires affichent des performances comparables.

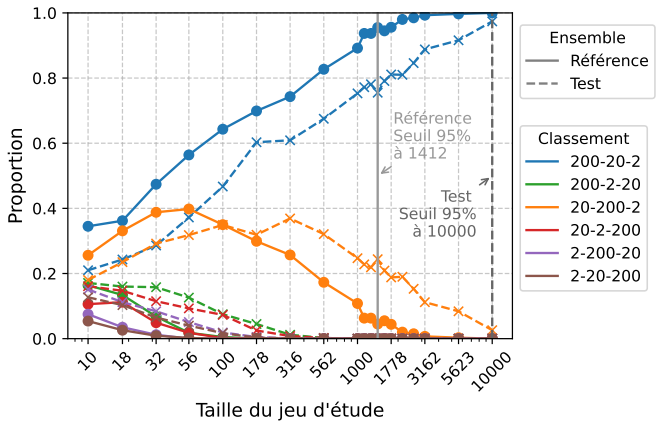
Enfin, une observation intéressante est que les rendements décroissants ne sont pas aussi sévères que l'on pourrait s'y attendre. Même avec une taille de l'ensemble de test fixe de 20% de l'ensemble d'étude, l'ajout de plus de cinq partitions apporte encore des améliorations significatives.

5 Conclusion : cross-validez !

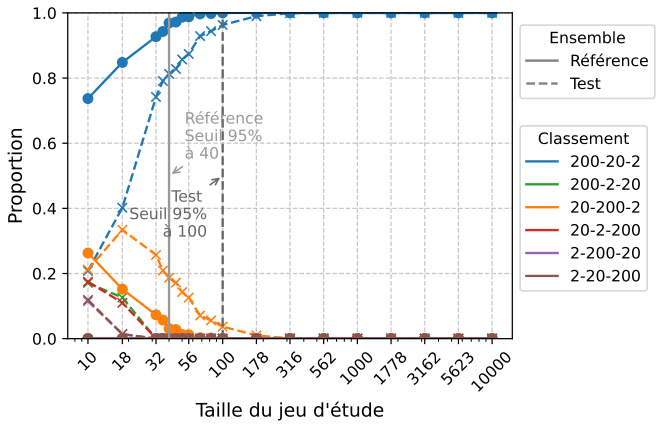
En analysant différentes procédures de validation croisée, nous avons démontré que l'augmentation du nombre de partitions améliore la robustesse du classement des algorithmes d'apprentissage, bien que cela entraîne un coût computationnel accru. Les résultats montrent que, malgré les rendements décroissants, les gains en fiabilité restent importants, encourageant ainsi l'utilisation de configurations plus robustes lorsque les ressources le permettent. L'utilisation d'un ensemble de référence de grande taille a permis de simuler un environnement quasi-oracle, renforçant la confiance dans nos conclusions.

Cette étude a pour limite de ne s'intéresser qu'à un unique jeu de données simulé : afin de généraliser nos résultats, nous devrons à l'avenir mener des expériences similaires sur des jeux de données réels.

Cette recherche souligne l'importance d'un équilibre entre la taille des données, le nombre de partitions, et les ressources computationnelles pour obtenir des classements fiables et robustes. Mais si vous le pouvez : cross-validez !



(a) Procédure train_test_split.



(b) Procédure RepeatedKFold avec 20 répétitions.

FIGURE 2 : Évolution des proportions des classements sur les ensembles de test et de référence. Les seuils correspondent aux premières tailles du jeu d'étude où un classement est trouvé dans 95% des itérations sur l'un ou l'autre de ces deux ensembles.

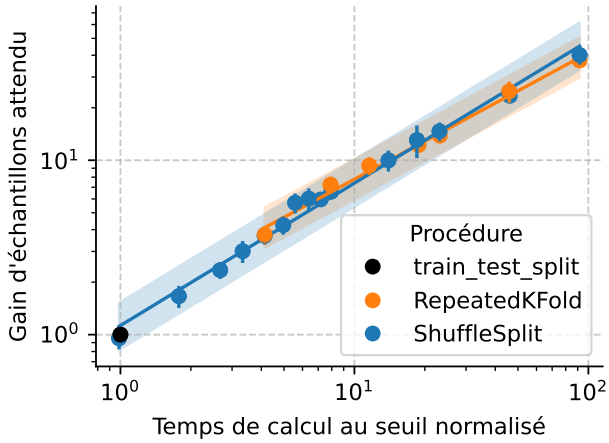


FIGURE 3 : Pour différentes procédures, taille de l'ensemble d'étude atteignant le seuil de 95% sur l'ensemble de référence et temps de calcul de la validation croisée. Chaque axe est normalisé par la valeur donnée par la procédure train_test_split.

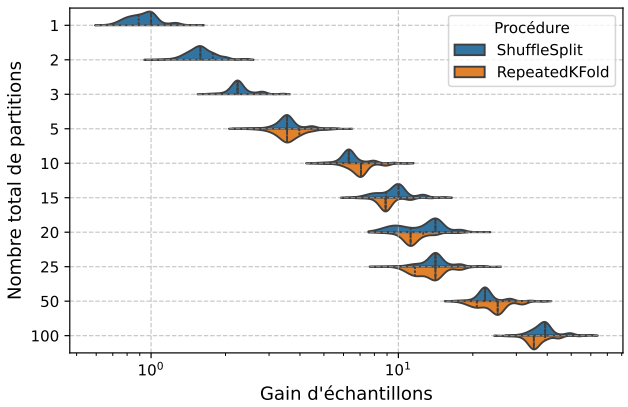


FIGURE 4 : Diagrammes en violon du seuil de 95% sur l'ensemble de référence de la procédure train_test_split divisé par les seuils de 95% sur l'ensemble de référence d'autres procédures.

Références

[1] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The Arcade Learning Environment : An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47 :253–279, 2013.

[2] X. Bouthillier, P. Delaunay, M. Bronzi, A. Trofimov, B. Nichyporuk, J. Szeto, N. Mohammadi Sepahvand, E. Raff, K. Madan, V. Voleti, S. Ebrahimi Kahou, V. Michalski, D. Serdyuk, T. Arbel, C. Pal, G. Varoquaux, and P. Vincent. Accounting for Variance in Machine Learning Benchmarks. *Proceedings of Machine Learning and Systems*, 3 :747–769, 2021.

[3] S. Chevallier, I. Carrara, B. Aristimunha, P. Guetschel, S. Sedlar, B. J. Lopes, S. Velut, S. Khazem, and T. Moreau. The largest EEG-based BCI reproducibility study for open science : the MOABB benchmark. 2024. arXiv :2404.15319.

[4] W.-L. Chiang, L. Zheng, Y. Sheng, A. N. Angelopoulos, T. Li, D. Li, H. Zhang, B. Zhu, M. Jordan, J. E. Gonzalez, and I. Stoica. Chatbot Arena : An Open Platform for Evaluating LLMs by Human Preference. In *ICML*, 2024.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet : A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[6] T. G. Dietterich. Approximate Statistical Tests for Comparing Supervised Classification Learning Algorithms. *Neural Computation*, 10(7) :1895–1923, 1998.

[7] D. Donoho. Data Science at the Singularity. *Harvard Data Science Review*, 6(1), 2024. Publisher : The MIT Press.

[8] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1) :3–42, 2006.

[9] Y. Lalou, T. Gnassounou, A. Collas, A. de Mathelin, O. Kachaiev, A. Odonnat, A. Gramfort, T. Moreau, and R. Flamary. SKADA-Bench : Benchmarking Unsupervised Domain Adaptation Methods with Realistic Validation, 2024. arXiv :2407.11676.

[10] T. Moreau, M. Massias, A. Gramfort, P. Ablin, P.-A. Bannier, B. Charlier, M. Dagréou, T. D. L. Tour, G. Durif, C. F. Dantas, Q. Klopfenstein, J. Larsson, E. Lai, T. Lefort, B. Malézieux, B. Moufad, B. T. Nguyen, A. Rakotomamonjy, Z. Ramzi, J. Salmon, and S. Vaiter. Benchopt : Reproducible, efficient and collaborative optimization benchmarks. *NeurIPS*, 2022.

[11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau. Scikit-learn : Machine Learning in Python. *MACHINE LEARNING IN PYTHON*, 2011.

[12] D. Picard. Torch.manual_seed(3407) is all you need : On the influence of random seeds in deep learning architectures for computer vision, 2023. arXiv :2109.08203.

[13] M. S. Sarfraz, M.-Y. Chen, L. Layer, K. Peng, and M. Koulakis. Position : Quo Vadis, Unsupervised Time Series Anomaly Detection ? In *ICML*, 2024.