Apprentissage automatique pour détecter les intrusions dans les réseaux IoT

Rapport technique

réalisé par
Loïc Sorroche et **Elyes Manai** ;

Québec, Canada

20 avril 2024

Résumé

Ce rapport technique présente une étude approfondie sur l'utilisation des techniques d'apprentissage automatiques pour détecter les intrusions dans les réseaux de l'Internet des Objets (IoT). Avec l'augmentation exponentielle du nombre de dispositifs IoT en service, la sécurité de ces réseaux est devenue une préoccupation majeure. Notre recherche se concentre sur l'implémentation et l'évaluation de plusieurs algorithmes d'apprentissage supervisés et non supervisés, y compris les K-Plus Proches Voisins (KNN), les K-moyens (KMeans), et les autoencodeurs pour la détection d'anomalies. Nous avons utilisé les deux jeux de données Edge-IIoTset et TON-IoT pour tester ces algorithmes.

La méthodologie adoptée consistait à implémenter et tester chaque algorithme sous diverses configurations pour évaluer leur efficacité à identifier les comportements malveillants dans les flux de données IoT. Les défis incluaient la gestion de grandes dimensions de données, l'optimisation des performances des modèles en présence de déséquilibres de classes, et l'interprétation des résultats dans des contextes non supervisés.

Les résultats obtenus démontrent une supériorité notable des approches supervisées, qui ont non seulement réalisé des taux de détection élevés mais ont également offert une robustesse significative face aux variations des types d'attaques. Ces modèles ont prouvé leur capacité à s'adapter et à généraliser à partir de données d'entraînement complexes, résultant en une précision et une réactivité améliorée dans la détection d'anomalies.

Cette recherche contribue à la littérature existante en confirmant l'efficacité des techniques d'apprentissage automatiques supervisées dans la sécurisation des environnements IoT et suggère des directions pour de futures études, notamment l'intégration de l'intelligence artificielle plus avancée pour développer des systèmes de détection d'intrusions plus dynamiques et adaptatifs. La finalité étant de renforcer la résilience des infrastructures IoT face à une panoplie d'attaques cybernétiques de plus en plus sophistiquées.

Table des matières

Liste des tableaux

1 Introduction

L'avènement de l'Internet des Objets (IoT) a marqué une ère de transformation numérique, où les objets physiques sont connectés à Internet et échangent des données de manière autonome. Cette évolution a ouvert un large éventail de possibilités, des applications industrielles aux systèmes domestiques intelligents. Selon Statista [14], environ 8.6 milliards de dispositifs IoT actifs étaient en service dans le monde en 2019 et ce chiffre devrait aller jusqu'à près de 30 milliards d'ici 2030. Cependant, avec cette expansion rapide de l'IoT, des défis significatifs ont émergé, notamment en matière de sécurité.

L'intégration croissante des appareils IoT dans notre vie quotidienne a mis en évidence la nécessité critique de renforcer la sécurité de ces systèmes. Les cyberattaques ciblant les dispositifs IoT peuvent avoir des conséquences graves, allant de l'accès non autorisé aux données sensibles à la compromission de la sécurité physique des individus. Par conséquent, il est impératif de développer des solutions robustes pour protéger les réseaux et les appareils IoT contre les menaces potentielles.

Dans ce contexte, les méthodes de détection d'intrusions en IoT ont gagné en importance. Ces méthodes comprennent une gamme de techniques visant à identifier et à contrer les activités malveillantes sur les réseaux IoT. L'utilisation de capteurs et d'algorithmes spécifiques permet de surveiller en temps réel les comportements suspects et de déclencher des alertes en cas d'anomalies, contribuant ainsi à renforcer la sécurité des systèmes IoT.

Parallèlement, l'intégration de l'intelligence artificielle (IA) dans le domaine de la cybersécurité offre de nouvelles perspectives pour la protection des réseaux IoT. Les techniques d'apprentissage automatique et d'apprentissage profond permettent de détecter les modèles de comportement malveillant avec une précision accrue, tout en réduisant les faux positifs. L'IA offre également la possibilité de développer des systèmes de défense adaptatifs, capables d'anticiper et de contrer les attaques avant qu'elles ne causent des dommages.

Enfin, le mariage entre l'intelligence artificielle et l'Internet des Objets ouvre de nouvelles perspectives dans le domaine de la sécurité. En combinant les capacités d'analyse avancées de l'IA avec la connectivité omniprésente de l'IoT, il est possible de créer des systèmes de surveillance et de protection plus intelligents et plus efficaces. Cette convergence offre un potentiel considérable pour renforcer la sécurité des infrastructures critiques et des applications IoT dans divers domaines, de la santé à l'industrie manufacturière.

Dans ce projet, on va explorer l'efficacité de la d'érection d'anomalie en fonction des modèle d'IA utilisé sur les datasets Edge-IIoTset et TON-IoT. pour se faire nous alons parler du background puis vous expliquez les démarches de nos expérimentations.

2 Contexte et concepts fondamentaux

Dans cette section, nous explorons le contexte et les concepts fondamentaux nécessaires à la compréhension approfondie de ce projet.

2.1 Etapes de pré-traitement

Avant de passer à la modélisation, différentes étapes de pré-traitement sont nécessaires pour préparer les données de manière adéquate. Cette sous-section explore en détail les étapes clés de pré-traitement effectuées sur nos jeux de données.

2.1.1 Standardisation

La standardisation est un processus qui consiste à transformer les données de telle sorte qu'elles présentent une moyenne de zéro et un écart-type de un. Ce processus commence par le calcul de la moyenne (μ) et de l'écart-type (σ) des valeurs de chaque caractéristique dans l'ensemble de données. Ensuite, pour chaque valeur dans une caractéristique donnée, la moyenne de cette caractéristique est soustraite, puis le résultat est divisé par l'écart-type, conformément à la formule suivante :

$$\text{valeur standardisée} = \frac{\text{valeur originale} - \mu}{\sigma}$$

Cette démarche permet de recentrer les données autour de zéro (la moyenne de chaque caractéristique devient 0) et de les mettre à l'échelle de manière à avoir une variance unitaire (l'écart-type devient 1). Ainsi, les caractéristiques deviennent comparables et les écarts d'échelle potentiellement biaisants pour les algorithmes d'apprentissage automatique sont éliminés. Une fois les moyennes et les écarts-types calculés, le standardiseur choisi peut transformer l'ensemble de données en remplaçant chaque valeur originale par sa valeur standardisée correspondante.

2.1.2 Encodage par étiquettes

L'encodage par étiquettes (Label Encoding) est un processus qui consiste à convertir des catégories ou des classes en valeurs numériques. Par exemple, si on avait une caractéristique avec comme valeurs "rouge", "vert" et "bleu", cet encodage pourrait les convertir en 0, 1 et 2 respectivement.

Contrairement à la standardisation des caractéristiques, qui recentre les données autour de zéro et les met à l'échelle pour avoir une variance unitaire, l'encodage par étiquettes ne modifie pas les écarts entre les différentes catégories. Il attribue simplement des valeurs numériques uniques à chaque catégorie. Cependant, il faut faire attention, car cela peut introduire des biais si les valeurs numériques attribuées suggèrent un ordre ou une relation entre les catégories qui n'existe pas dans les données originales. Dans de tels cas, d'autres techniques d'encodage comme l'encodage binaire (One-Hot Encoding) peuvent être préférables.

2.2 Algorithmes d'apprentissage supervisés

L'apprentissage supervisé est un paradigme de l'apprentissage automatique où l'algorithme est entraîné sur un ensemble de données étiqueté, signifiant que chaque exemple de formation dans l'ensemble de données est associé à sa cible ou sortie correspondante. L'objectif de l'apprentissage supervisé est d'apprendre une cartographie ou relation entre les caractéristiques d'entrée et leurs étiquettes associées.

2.2.1 Voisins les Plus Proches - K-Plus Proches Voisins (KNN)

KNN classe un point de données x en fonction de la classe majoritaire parmi ses k voisins les plus proches dans l'espace des caractéristiques. La classe $C(x)$ est déterminée par un vote majoritaire parmi les k voisins, souvent en utilisant une métrique de distance telle que la distance euclidienne : $C(x) = argmax_c \sum_{i=1}^{k} \delta(y_i, c)$, où y_i est la classe du i-ème voisin, x représente un point de données pour la classification, k est le nombre de voisins les plus proches à considérer, $C(x)$ est la classe assignée au point de données x, et y_i est la classe du i-ème voisin le plus proche.

2.2.2 Modèles d'arbre - Arbre de Décision (DT) :

Les arbres de décision divisent récursivement les données en fonction des caractéristiques pour créer une structure d'arbre. À chaque nœud, une décision est prise en fonction d'une caractéristique x_i : $x_i \leq t$ ou $x_i > t$, où x_i est une caractéristique utilisée pour la prise de décision à un nœud et t est une valeur seuil pour la caractéristique x_i. Le processus continue jusqu'à ce qu'un critère d'arrêt soit atteint. La décision est prise en parcourant l'arbre de la racine à un nœud feuille.

2.2.3 Modèles de boosting - XGBoost (XGB) :

XGBoost entraîne séquentiellement des apprenants faibles et les combine en un modèle prédictif fort : $F_t(x) = F_{t-1}(x) + \eta h_t(x)$, où $F_t(x)$ est le score prédit à l'itération t, η est le taux d'apprentissage, et $h_t(x)$ est l'apprenant faible. L'objectif est de minimiser une fonction objective régularisée.

2.2.4 Modèles de régression - Régression Logistique (LR)

La régression logistique modélise la probabilité qu'une instance donnée appartienne à une classe particulière en utilisant la fonction logistique : $P(Y = 1|x) = \frac{1}{1+e^{-(w \cdot x + b)}}$, où x représente les caractéristiques d'entrée d'une instance, w est le vecteur de poids, et b est le terme de biais. La frontière de décision est là où $P(Y = 1|x) = 0.5$, et la classe est déterminée en conséquence.

2.2.5 Modèles bayésiens - Bayes Naïf Gaussien (GNB) :

GNB est basé sur le théorème de Bayes. Il suppose que les caractéristiques sont indépendantes et suivent une distribution gaussienne : $P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_{y_i}^2}} e^{-\frac{(x_i-\mu_{y_i})^2}{2\sigma_{y_i}^2}}$, où x_i est la i-ème caractéristique d'un point de données, y est l'étiquette de classe, μ_{y_i} est la moyenne de la i-ème caractéristique pour la classe y, et σ_{y_i} est l'écart type de la i-ème caractéristique pour la classe y. La classe est déterminée en maximisant la probabilité postérieure $P(y|x)$ en utilisant la règle de Bayes.

2.3 Algorithmes d'apprentissage non supervisé

L'apprentissage non supervisé est un paradigme de l'apprentissage automatique où l'algorithme reçoit des données non étiquetées et est chargé de trouver des motifs, des structures ou des relations au sein des données sans guidance explicite. L'algorithme explore la structure inhérente dans les données sans étiquettes de sortie prédéfinies. On utilise 2 types de non supervisé :

1. **Regroupement (Clustering) :** Les algorithmes de regroupement visent à regrouper ensemble des points de données similaires basés sur certaines caractéristiques. Ces méthodes sont employées dans des scénarios d'apprentissage non supervisé, où l'algorithme identifie des motifs ou des structures inhérents au sein des données.
2. **Détection d'anomalies :** La détection d'anomalies, également connue sous le nom de détection de valeurs aberrantes ou de détection de nouveautés, est une branche de l'apprentissage automatique qui se concentre sur l'identification de motifs ou d'instances qui s'écartent de manière significative de la norme ou du comportement attendu au sein d'un ensemble de données. Les anomalies, souvent appelées valeurs aberrantes, sont des points de données qui diffèrent de manière marquée de la majorité des données. La détection d'anomalies est utilisée dans divers domaines, tels que la détection de fraude, la détection de défauts dans les processus industriels, la sécurité réseau et la surveillance de la santé.

Dans le cadre de ce projet, on va utiliser K-Means et BIRCH comme algorithmes de regroupement et les auto-encodeurs comme algorithmes de détection d'anomalies. On a aussi essayé de travailler avec les Forêts d'Isolation, le Facteur d'aberration locale, et le SVM à Une Classe comme algorithmes de détection d'anomalies ainsi que DBScan, MeanShift, OPTICS, et le regroupement agglomératif comme algorithmes de regroupement. Ces derniers par contre ont été abandonnés car ils étaient trop gourmands en resources et demandaient des temps de traitement exponentiels qui montaient jusqu'à des dizaines d'heures.

2.3.1 Regroupement - K-Means

K-Means est un algorithme de regroupement largement utilisé qui divise un ensemble de données en K groupes. L'algorithme attribue chaque point de données au groupe dont la moyenne est la plus proche, optimisant la similarité au sein de chaque groupe. Mathématiquement, cela peut être décrit comme suit :

Soit X l'ensemble de données avec n points de données, $x_1, x_2, \ldots, x_n$, et $C = \{c_1, c_2, \ldots, c_K\}$ l'ensemble des centroïdes de groupes. L'objectif est de minimiser la somme des carrés intra-groupe :

$$J(C) = \sum_{i=1}^{n} \min_j \|x_i - c_j\|^2$$

L'algorithme met à jour itérativement les centroïdes des groupes et attribue les points de données au groupe à la moyenne la plus proche jusqu'à convergence. L'optimisation implique deux étapes :

1. **Étape d'attribution :** Attribuer chaque point de données au groupe avec le centroïde le plus proche.

$$\text{Pour chaque } i, \quad \text{minimiser} \quad \|x_i - c_j\|^2 \quad \text{et attribuer } x_i \text{ au groupe } j.$$

2. **Étape de mise à jour :** Recalculer les centroïdes sur la base des attributions actuelles.

$$\text{Pour chaque } j, \quad c_j = \frac{\sum_{i \text{ dans le groupe } j} x_i}{\text{nombre de points dans le groupe } j}$$

Ces étapes sont répétées jusqu'à la convergence, et le résultat est un partitionnement de l'ensemble de données en K groupes.

2.3.2 Regroupement - BIRCH

Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) est un algorithme de regroupement hiérarchique conçu pour gérer efficacement de grands ensembles de données. Il construit de manière incrémentielle une structure d'arbre pour représenter la hiérarchie de regroupement, le rendant adapté pour des tâches de regroupement évolutives.

L'algorithme utilise une structure de données basée sur un arbre appelée Arbre de Caractéristiques de Groupe (CF Tree). Chaque nœud de l'arbre représente un groupe et contient un résumé des points de données au sein de ce groupe. Le CF Tree est mis à jour de manière incrémentielle

à mesure que de nouveaux points de données sont introduits, permettant à BIRCH de s'adapter aux motifs changeants dans les données.

Au cœur de BIRCH, il y a un processus en deux étapes pour le regroupement :

1. **Construction de Caractéristique de Regroupement :** L'algorithme construit une Caractéristique de Regroupement (CF) pour chaque point de données entrant. La CF résume les informations statistiques sur les points de données au sein d'un groupe, incluant le nombre de points (N), la somme linéaire (LS), et la somme au carré (SS) de chaque caractéristique.
$$CF = (N, LS, SS)$$
2. **Regroupement Structuré en Arbre :** Les CF sont organisées dans une structure d'arbre, et l'algorithme fusionne récursivement les nœuds de l'arbre pour créer une hiérarchie de groupes. Cette représentation hiérarchique permet une gestion efficace et évolutive des grands ensembles de données.

BIRCH est particulièrement efficace dans des scénarios où l'ensemble de données est trop volumineux pour tenir entièrement en mémoire, car il traite les données de manière incrémentielle et maintient un résumé compact des groupes à l'aide du CF Tree.

2.3.3 Détection d'anomalies - Auto-encodeurs

Les auto-encodeurs sont une classe de réseaux de neurones utilisée dans la détection d'anomalies qui apprend à coder les données d'entrée en une représentation compressée (espace latent) pour ensuite les reconstruire avec une perte minimale d'information. Cette méthode se base sur la construction d'une architecture comprenant principalement trois parties : l'encodeur, le code (représentation latente), et le décodeur.

- **Encodeur :** Transforme les données d'entrée en une représentation latente.
- **Espace latent (Code) :** Représentation compressée des données d'entrée, captant les caractéristiques essentielles pour la reconstruction.
- **Décodeur :** Tente de reconstruire les données d'entrée à partir de la représentation latente.

L'objectif est de minimiser l'erreur de reconstruction, généralement calculée comme l'erreur quadratique moyenne (MSE) entre l'entrée originale x et sa reconstruction $\hat{x}$, selon la formule suivante :

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x}_i)^2$$

où n représente le nombre d'échantillons dans l'ensemble de données.

Dans le contexte de la détection d'anomalies, on suppose que l'auto-encodeur, après un entraînement sur des données considérées comme normales, sera moins capable de reconstruire précisément des anomalies. Ces dernières peuvent donc être détectées par une erreur de reconstruction significativement plus élevée par rapport aux données normales.

Les auto-encodeurs offrent une flexibilité importante et peuvent être adaptés à travers différentes variantes, telles que les auto-encodeurs variationnels (VAE) et les auto-encodeurs empilés (Stacked Auto-encoders), chacun présentant des caractéristiques propres adaptées à divers types de données et d'applications.

Cette approche est particulièrement utile pour traiter des ensembles de données complexes et de grande dimension, où la distinction entre les comportements normaux et anormaux n'est pas évidente. Les auto-encodeurs excèlent dans l'apprentissage de représentations latentes non linéaires, ce qui les rend efficaces pour identifier des schémas complexes et subtiles indiquant des anomalies.

2.4 Évaluation

Pour l'évaluation de ces jeux de données, et pour les deux types d'apprentissages, on va utiliser les mêmes métriques de performance. Ces métriques sont connues et basées sur la matrice de confusion :

Une matrice de confusion est une disposition spécifique de tableau qui permet de visualiser la performance d'un algorithme, typiquement un algorithme d'apprentissage supervisé. Elle

est particulièrement utile dans l'analyse de classification, où elle aide à montrer les écarts entre les classifications réelles et prédites. La matrice compare les valeurs cibles réelles avec celles prédites par le modèle d'apprentissage automatique, fournissant un aperçu du nombre de prédictions correctes et incorrectes faites par le modèle, ventilé par chaque classe :

- **Vrais Positifs (VP) :** Instances correctement identifiées comme positives par un modèle de classification.
- **Vrais Négatifs (VN) :** Instances correctement identifiées comme négatives par un modèle de classification.
- **Faux Positifs (FP) :** Instances incorrectement identifiées comme positives par un modèle de classification.
- **Faux Négatifs (FN) :** Instances incorrectement identifiées comme négatives par un modèle de classification.

Un exemple de matrice de confusion binaire est montré dans le Tableau 1.

Tableau 1 – Exemple de matrice de confusion

	Négatifs Prédits	**Positifs Prédits**
Négatifs Réels	87	15
Positifs Réels	10	92

Un exemple de matrice de confusion multi-classes est montré dans le Tableau 2.

Tableau 2 – Matrice de Confusion

	Prédit : A	**Prédit : B**	**Prédit : C**
Réel : A	10	192	34
Réel : B	1	123	67
Réel : C	45	1	14

Ces 4 valeurs permettent le calcul de métriques telles que la précision, la spécificité, le rappel et le score F1 (expliqués dans la partie expérimentation). Elles sont donc une pierre angulaire de l'évaluation de performance pour les modèles ML.

- **Précision :** Représente le rapport des instances correctement classées parmi le total des instances, fournissant une mesure globale de la justesse d'un modèle. Son équation est :

$$Prcision = \frac{VP + VN}{VP + VN + FP + FN} \tag{1}$$

- **Précision (Exactitude) :** Représente le rapport des observations positives correctement prédites au total des positifs prédits, mettant l'accent sur la justesse des prédictions positives. Son équation est :

$$Prcision = \frac{VP}{VP + FP} \tag{2}$$

- **Rappel :** Représente le rapport des observations positives correctement prédites à tous les positifs réels, mettant en évidence la capacité d'un modèle à capturer toutes les instances pertinentes. Son équation est :

$$Rappel = \frac{VP}{VP + FN} \tag{3}$$

- **Score F1 :** Équilibre la précision et le rappel, fournissant une moyenne harmonique entre les deux métriques et offrant une évaluation complète de la performance d'un modèle sur des tâches de classification binaire. Son équation est :

$$F1 = \frac{2 * Prcision * Rappel}{Prcision + Rappel} \tag{4}$$

3 Expérimentations

Cette section expose l'environnement de travail, détaille les phases de prétraitement et de modélisation entreprises, et présente les résultats expérimentaux obtenus.

3.1 Eléments en commun

Dans cette partie, on présente les éléments de notre travail qui ont été partagés par les membres de l'équipe et appliqués sur les deux jeux de données de ce projet.

3.1.1 Environnement de travail

Notre environnement de travail physique comporte :

- **Pour Edge-IIoTset :** Un ordinateur portable doté d'un CPU Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz, 24Go RAM et une carte graphique RTX-3060.
- **Pour TON-IoT :** Un ordinateur portable doté d'un CPU AMD RAZEN(TM) 9 5900HX @ 3.30GHz, 16Go RAM et une carte graphique RTX-3060.
- **Pour les deux :** Un environnement Cloud Google Colab avec 12 GB RAM et une carte graphique T4.

Cette information est importante à déclarer car plusieurs algorithmes d'apprentissage prenaient beaucoup trop de temps et de mémoire pour cet environnement et on a du les abandonner.

3.1.2 Pré-traitement en commun

Plusieurs étapes de pré-traitement ont été partagés par les 2 jeux de données, notamment :

1. La supression de dupliqués en gardant que la première occurence.
2. La suppression de lignes où il existe au moins 1 valeur manquante.
3. Un mélange aléatoire des lignes.
4. Encodage des données catégorielles avec Label Encoding de Scikit-learn [12]. Cet encodage consiste à représenter chaque catégorie par un nombre entier unique, de sorte que chaque label est unique et ordonné.
5. La standardisation des valeurs avec StandardScaler de Scikit-learn [12], en raison de l'écart considérable entre les plages de valeurs des différentes colonnes.

On détaille dans les sections suivantes les étapes de pré-traitement supplémentaires appliquées pour chacun des deux jeux de données.

3.1.3 Choix et conception des modèles

Nous avons utilisé les mêmes modèles ML pour modéliser nos deux jeux de données, notamment KNN, DT, XGB, LR, K-Means ainsi que les auto-encodeurs. Par contre, Edge-IIoT a en plus été modélisé avec GNB et BIRCH car ces modèles ont pris trop de temps pour s'entrainer sur ToN-IoT, ce qui nous a obligé de les abandonner pour ce dernier. Pour

les modèles KNN, GNB, DT et LR, nous les avons initialisés avec les valeurs par défaut disponibles dans la bibliothèque Scikit-learn [12]. Pour KNN, cette valeur est fixée à 5.

Pour XGBoost, nous avons utilisé 200 estimateurs d'une profondeur maximum égale à 5 chacun, un taux d'apprentissage de 0.05 et un objectif de scoring de 'binary :logistic' ou 'multi :softmax' selon la version du jeu de données. Finalement, on a utilisé tree_method = 'gpu_hist' pour pouvoir l'entrainer sur notre GPU.

Concernant K-Means et BIRCH, nous avons également utilisé les valeurs par défaut, mais nous avons ajusté le nombre de clusters à 2 pour les problèmes binaires et à 15 pour les problèmes multiclasse.

Finalement, pour le auto-encodeur, l'architecture séquentielle est la suivante :

1. Couche d'entrée avec dimensions égale au nombre de colonnes qu'on a
2. Couche encodeur de type Dense avec 14 dimensions et activation "tanh"
3. Couche encodeur de type Dense avec 7 dimensions et activation "relu"
4. Couche encodeur de type Dense avec 7 dimensions et activation "relu"
5. Couche encodeur de type Dense avec 14 dimensions et activation "tanh"

On a optimizé l'auto-encodeur avec l'optimizateur Adam (par défaut et généralement bon) et on a choisi l'erreur quadratique moyenne (Mean Squared Error) comme métrique de performance pour évaluer la reconstruction.

3.1.4 Evaluation

Pour l'évaluation de nos modèles grâce aux métriques d'accuracy, précision, recall et f1 on a fait le suivant :

- Pour évaluer les algorithmes supervisés :
  1. Nous entraînons le modèle supervisé sur les données d'entraînement.
  2. Nous utilisons le modèle pour prédire les labels sur les données de test.
  3. Nous comparons les prédictions avec les vrais labels pour évaluer les performances du modèle à l'aide des métriques ci-dessus.
- Pour évaluer un modèle non supervisé, nous suivons les étapes suivantes :
  1. Entraîner le modèle non supervisé sur les données X.
  2. Extraire les numéros de cluster pour chaque point de données.
  3. Utiliser les clusters extraits comme étiquettes de prédiction et les comparer à la vérité terrain (étiquettes Y) en utilisant les métriques ci-dessus.
- Pour les auto-encodeurs, les étapes sont un peu différentes :
  1. Nous utilisons l'auto-encodeur pour reconstruire les données d'entrée.
  2. Nous calculons l'erreur de reconstruction pour chaque exemple.
  3. En utilisant une certaine valeur seuil pour l'erreur de reconstruction, nous classifions les exemples comme anomalies ou non.
  4. Nous évaluons ensuite les performances du modèle avec les métriques traditionnels.

3.2 Le jeu de données Edge-IIoTset

Cette partie se concentre sur le jeu de données Edge-IIoTset, son pré-traitement spécifique ainsi que ses résultats de modélisation ML.

3.2.1 Présentation du jeu de données

L'Edge-IIoTset [5] est un jeu de données de cybersécurité conçu spécifiquement pour les applications IoT et IIoT, visant à renforcer les systèmes de détection d'intrusion basés sur l'apprentissage automatique à travers les paradigmes d'apprentissage centralisé et fédéré. Il a été généré via un ensemble de 18 outils open source (tels que Nod Red, Zeek et Spyder) et incorpore une vaste gamme de dispositifs, capteurs, protocoles et configurations à travers les environnements de cloud computing et edge computing. Le jeu de données se distingue par l'inclusion de données IoT provenant de plus de 10 types de dispositifs (par exemple, des capteurs pour surveiller la température, l'humidité, le niveau d'eau, le pH, l'humidité du sol, la fréquence cardiaque) et englobe 14 types d'attaques segmentées en 6 catégories telles que les attaques DoS/DDoS, la collecte d'informations, l'homme du milieu, les attaques par injection et les attaques par logiciels malveillants. De plus, il introduit 61 nouvelles caractéristiques identifiées à forte corrélation, sélectionnées à partir d'un pool initial de 1176 caractéristiques, dérivées de diverses sources, y compris les alertes, les ressources système, les journaux et le trafic réseau. L'accès public au jeu de données est facilité via `http://ieee-dataport.org/8939`, en améliorant son utilité pour les chercheurs et les praticiens en cybersécurité.

L'évaluation de la performance de ce jeu de données a été menée en utilisant à la fois des approches d'apprentissage centralisé et fédéré avec des algorithmes d'apprentissage profond et d'apprentissage automatique traditionnel, notamment l'arbre de décision (DT), la forêt aléatoire (RF), la machine à vecteurs de support (SVM), le K-Plus proches voisins (KNN), ainsi que le réseau de neurones profond (DNN). L'expérimentation a été structurée autour de trois scénarios de classification : binaire, à 6 classes, et à 14 classes, pour évaluer à la fois la prédictibilité du trafic et l'efficacité de la détection de diverses cyberattaques et modèles de menace.

Dans le scénario de classification binaire, le modèle vise à distinguer entre le trafic normal et les activités malveillantes sans distinction entre les types d'attaques. Les résultats obtenus dans cette configuration fournissent une mesure de base de la capacité des modèles à discriminer entre les comportements bénins et malicieux. Pour la classification à 14 classes, le but est de différencier entre le trafic normal et les treize types spécifiques d'attaques décrits dans le jeu de données. Ce scénario offre une compréhension plus profonde de la capacité des modèles à identifier des attaques spécifiques, permettant une réaction plus ciblée et informée aux incidents de sécurité.

Deux approches d'apprentissage automatique ont été testées : d'une part, l'apprentissage centralisé, exploitant les ressources étendues du cloud pour contourner les limitations en termes de ressources, et d'autre part, l'apprentissage fédéré, visant à évaluer la précision de la détection tout en tenant compte des problématiques de confidentialité, d'hétérogénéité et de disponibilité des données. L'apprentissage fédéré offre l'avantage de permettre aux modèles d'apprendre à partir de données distribuées sans nécessiter leur centralisation, abordant ainsi les préoccupations relatives à la confidentialité des données.

3.2.2 Pré-traitement spécifique

Les auteurs du jeu de données ont partagé, sur le site web du projet, un code Python de pré-traitement de données prêt à être utilisé qui comprenait les étapes 1, 2 et 3 du pré-traitement en commun. On a décidé de faire les étapes 4 et 5 après analyse du jeu de données.

Finalement, quelques colonnes n'avaient qu'une seule valeur pour toutes les lignes et on les a du coup supprimé. Ces colonnes sont : 'frame.time', 'ip.src_host', 'ip.dst_host', 'arp.src.proto_ipv4', 'arp.dst.proto_ipv4', 'http.file_data', http.request.full_uri', 'mqtt.msg', 'icmp.transmit_timestamp', 'http.request.uri.query', 'tcp.options', 'tcp.payload', 'tcp.srcport', 'tcp.dstport', 'udp.port'

3.3 Résultats expérimentales de la modélisation

Nous comparons les performances de nos modèles à celles présentées par les auteurs. Toutefois, il est à noter que les auteurs n'ont utilisé qu'une seule métrique, à savoir l'accuracy. En commenceant par la version binaire du jeu de données, on montre les résultats de notre modélisation binaire dans le Tableau 3.

TABLEAU 3 – Comparaison des performances des méthodes supervisées et non supervisées pour la détection d'intrusion binaire

Type	Modèle	Accuracy	Precision	Recall	F1
Supervisé	KNN	**1**	1	1	1
	XGBoost	**1**	1	1	1
	DT	**1**	1	1	1
	LR	**1**	1	1	1
	GNB	**1**	1	1	1
Non Supervisé	Kmeans	0.673	0	0	0.577
	Birch	0.716	0	0	0.598
	Auto Encodeur	0.724	0.879	0.031	0.618
Travail Auteurs	DT	99.98	N/A	N/A	N/A
	RF	99.99	N/A	N/A	N/A
	SVM	99.99	N/A	N/A	N/A
	KNN	99.99	N/A	N/A	N/A
	DNN	99.99	N/A	N/A	N/A

On peut voir que toutes nos méthodes supervisées ont eu des valeurs parfaites de performance, ce qui n'est pas surprenant puisque les auteurs du jeu de données ont eu les mêmes résultats. Par contre, les méthodes non supervisées ont été largement inférieurs en termes de performances, avec Kmeans et Birch ayant un score de précision et recall de 0, insinuant qu'ils n'ont pas du tout réussi à détecter les intrusions. L'autoencodeur s'est montré plus efficace dans la précision et l'excactitude, mais pas assez pour compéter avec les méthodes supervisées. Les méthodes de travail des auteurs montrent des performances presque parfaite, ce compense le de valeurs pour le reste des métriques.

En passant à la modélisation multi-classe, on montre les résultats multi-classes dans le Tableau 4.

Tableau 4 – Comparaison des performances des méthodes supervisées et non supervisées pour la détection d'intrusion multi-classes

Modèle	Model	Accuracy	Precision	Recall	F1
Supervisé	KNN	**0.953**	0.953	0.953	0.953
	XGBoost	**0.985**	**0.987**	**0.985**	**0.985**
	DT	**0.981**	0.981	0.981	0.981
	LR	**0.95**	0.95	0.95	0.95
	GNB	**0.902**	0.942	0.902	0.89
Non Supervisé	Kmeans	0.001	0.000	0.001	0
	Birch	0.012	0.027	0.012	0
	Auto Encodeur	0.012	0	0.012	0
Travail Auteurs	DT	0.671	N/A	N/A	N/A
	RF	0.808	N/A	N/A	N/A
	SVM	0.776	N/A	N/A	N/A
	KNN	0.7918	N/A	N/A	N/A
	DNN	0.947	N/A	N/A	N/A

Les modèles supervisés KNN, XGBoost, DT, LR et GNB ont des scores élevés sur pour toutes les métriques, avec des valeurs allant de 0.902 à 0.985, ce qui suggère qu'ils sont efficaces dans la détection d'intrusions multi-classes. Les modèles non-supervisés Kmeans, Birch et l'auto-encodeur ont des performances considérablement inférieures, avec des scores d'exactitude très bas, proches de 0, indiquant une incapacité à détecter correctement les intrusions multi-classes. Il est important de noter que nous avons obtenu des performances nettement meilleures que celles des auteurs en utilisant des algorithmes presque identiques. Cette disparité peut s'expliquer par plusieurs facteurs :

1. Les modèles utilisés par les auteurs pourraient avoir des paramètres différents des nôtres. Les auteurs n'ont pas partagé les détails spécifiques de leurs modèles, ce qui rend difficile une comparaison précise.
2. Les auteurs n'ont pas précisé comment ils ont réparti les données entre les ensembles d'entraînement et de test. Cette répartition pourrait avoir un impact significatif sur les performances des modèles, expliquant ainsi les variations observées dans les scores entre les différents modèles.

3.3.1 Discussion

D'après ce travail, on peut dire que les méthodes supervisées sont nettement supérieures aux méthodes non supervisées pour la détection binaire et multi-classe des intrusions dans des données IoT. Par contre, la distribution des données dans Edge-IIoT est linéaire et trop facile à modéliser avec des méthodes même traditionnelles, ce qui rend difficile de tirer une conclusion car ca ne reflète pas le monde réel. Plus de tests sur d'autres jeux de données sont requis pour pouvoir analyser les nuances entre algorithmes. En plus, une analyse visuelle et statistique pourrait être intéressante pour comprendre pourquoi les méthodes non supervisées ont eu des mauvais résultats sur ce jeu de données. Cette analyse pourrait nous aider à

comprendre quelles étapes supplémentaires pourraient être prises pour rendre ces méthodes plus performantes.

3.4 TON-IoT

Cette partie se concentre sur le jeu de données Edge-IIoTset, son pré-traitement spécifique ainsi que ses résultats de modélisation ML

3.4.1 Présentation du jeu de données

Le TON_IoT [2] est un ensemble de jeux de données séparées pour chaque appareil tel que : le frigo, le garage, le traqueur GPS, le Modbus, les lumières, le thermostat, la station météo et le réseau. il y a aussi des données sur des appareils linux et windows qui ne nous seront pas utiles pour cette étude. ici nous avons travaillé avec le jeu de donné réseau possèdent 211 044 lignes et 44 colonnes

Le TON_IoT Telemetry Dataset est un ensemble de données IoT et IIoT conçu pour renforcer les systèmes de détection d'intrusion basés sur l'apprentissage automatique. Il a été créé à partir d'un banc d'essai représentatif et intègre des données de télémétrie, des journaux de systèmes d'exploitation et du trafic réseau . Ce jeu de données se distingue par sa représentation réaliste des réseaux IoT/IIoT et inclut des scénarios normaux et d'attaque, ainsi que des sous-classes d'attaques pour la classification multi-classe .

Les principales contributions de cette étude comprennent la création d'un banc d'essai moyen échelle pour la génération des données, l'introduction de nouveaux ensembles de données TON-IoT basés sur ce banc d'essai, et l'évaluation de sept méthodes d'apprentissage automatique ainsi qu'un modèle d'apprentissage profond sur ces ensembles de données . Ces ensembles de données sont disponibles publiquement pour la communauté de recherche, offrant ainsi une base solide pour le développement et l'évaluation de solutions de défense IoT/IIoT .

L'évaluation des performances des méthodes de détection d'intrusion sur le TON_IoT Telemetry Dataset a montré que les méthodes RF et CART ont obtenu les meilleurs scores dans tous les critères d'évaluation, tant sur les ensembles de données par appareil que sur l'ensemble combiné . Ces résultats indiquent que les ensembles de données proposés peuvent être efficacement utilisés pour former et mettre en œuvre diverses méthodes d'apprentissage automatique pour la détection d'anomalies dans les applications IoT et IIoT.

En conclusion, le TON_IoT Telemetry Dataset représente une ressource précieuse pour la recherche en matière de sécurité des systèmes IoT et IIoT, offrant des données réalistes et diversifiées pour l'évaluation des méthodes de détection d'intrusion et ouvrant la voie à de futures avancées dans ce domaine

3.4.2 Preprocessing spécifique

Afin d'effectuer les expérimentations et comparatifs entre les modèles d'entraînement à utiliser, un preprocessing du dataset devait être effectué. Pour ce faire :

1. J'ai ajouté la valeur id à chaque ligne du tableau de données afin de les différencier.
2. Puis nous avons supprimé toutes les colonnes qui avaient pour titre "unnamed".
3. En suite nous avons supprimé toutes les lignes qui dans la colonne type avaient pour valeurs les strings : '1', '0', '-', ' WOW64)', 'sqlmap/1.2#stable (http ://sqlmap.org)', '37'. Qui était des erreurs de recensement de données.

4. Pour finir le dataset a subi le preprocessing commun.

3.4.3 Résultats expérimentales de la modélisation

Après avoir parcouru les dix publications les plus récentes en rapport avec TON-IoT [4, 1, 6, 8, 11, 10, 9, 3, 7, 13] publié entre 2022 et fin 2023, je me suis aperçu qu'aucun tableau comparatif de performance n'a été publié. Je vais donc seulement comparer mes resultas obtenue entre eux.

Tableau 5 – Comparaison des performances des méthodes supervisées et non supervisées pour la détection d'intrusion binaire détection d'intrusion

méthode d'apprentissage	**modèle**	**Accuracy**	**Precision**	**Recall**	**F1 Score**
supervisé	**XGBoost**	**1**	**1**	**1**	**1**
supervisé	knn	0.975	0.974	0.975	0.974
supervisé	**dt**	**1**	**1**	**1**	**1**
supervisé	lr	0.899	0.896	0.899	0.895
non-supervisé (avec des données non standardisées)	KMeans	0.23	0.23	0.23	0.23
non-supervisé (avec des données standardisées)	KMeans	0.646	0.814	0.695	0.750
non-supervisé (avec des données non standardisées)	autoencoder	Erreur	Erreur	Erreur	Erreur
non-supervisé (avec des données standardisées)	autoencoder	0.246	0.937	0.012	0.11

Il est observé que les modèles supervisés produisent des résultats nettement supérieurs, notamment XGB et DT, qui atteignent un score général de 100% dans chaque domaine d'évaluation. Par conséquent, il est judicieux d'opter pour l'un de ces modèles. De plus, les méthodes d'apprentissage supervisé nécessitent moins de temps de traitement que les méthodes non supervisées pour atteindre ces résultats, ce qui implique une consommation d'énergie moindre. Or, la gestion énergétique est l'un des défis de l'IoT.

3.4.4 Discussion

D'après ce travail, on peut dire que les méthodes supervisées sont nettement supérieures aux méthodes non supervisées pour la détection multi-classe des intrusions dans des données IoT. Par contre, la distribution des données dans ToN_IOT est trop facile à modéliser avec des méthodes même traditionnelles, ce qui rend difficile de tirer une conclusion. Plus de tests sur d'autres jeux de données sont requis pour pouvoir analyser les nuances entre algorithmes. En plus, une analyse visuelle et statistique pourrait être intéressante pour comprendre pourquoi les méthodes non supervisées ont eu des mauvais résultats sur ce jeu de données, car ici seul KMeans avec des données standardisées obtiens de bonnes valeurs parmi les modèle non-supervisé.

De plus on peut voir que la distribution de ce jeu de données est assez linéaire à cause des résultats de lr mais elle aussi assez dense avec existence de regroupements à cause des

résultats de knn mais il existe quand même de l'anormalité puisque ces 2 algos n'ont pas un score parfait, cette anormalité a été ramassée par les algos d'arbres c'est pour quoi on peut en déduire que la distribution est assez anormale. Cette analyse pourrait nous aider à comprendre quelles étapes supplémentaires pourraient être prises pour rendre ces méthodes plus performantes.

4 Conclusion générale

Le projet a abordé la détection des intrusions dans les réseaux IoT en utilisant des méthodes d'apprentissage automatique, à la fois supervisées et non supervisées. Les expérimentations, basées sur les jeux de données Edge-IIoTset et TON-IoT, ont révélé des résultats prometteurs, en particulier avec l'application de modèles supervisés comme XGBoost, KNN, DT, et LR, qui ont montré une efficacité notable dans la distinction entre le trafic normal et les activités malveillantes. Par contre, bien que les modèles supervisés tels que XGBoost, KNN, DT, et LR aient démontré une efficacité remarquable, atteignant presque la perfection dans la distinction entre le trafic normal et les activités malveillantes sur les jeux de données Edge-IIoTset et TON-IoT, cette performance exceptionnelle soulève des questions quant à la complexité des données utilisées. En effet, la distribution relativement simple des jeux de données a favorisé les méthodes supervisées, rendant la comparaison avec les méthodes non supervisées quelque peu injuste.

Les résultats obtenus indiquent que, bien que les méthodes non supervisées semblent moins prometteuses dans ce contexte spécifique, leur potentiel ne doit pas être sous-estimé, surtout dans des scénarios impliquant des données plus complexes et moins structurées. Pour une évaluation équitable et exhaustive des capacités des différentes approches d'apprentissage automatique dans la détection d'intrusions IoT, il est impératif de se tourner vers des jeux de données plus complexes et représentatifs des défis réels auxquels les systèmes IoT sont confrontés aujourd'hui.

Ainsi, Il est à reconnaître que, bien que les résultats actuels soient prometteurs, ils ne permettent pas de tirer des conclusions définitives sur la supériorité des méthodes supervisées sur les non supervisées dans tous les contextes de sécurité IoT. La complexité et l'évolution constante des menaces cybernétiques nécessitent une exploration continue de données plus complexes, reflétant avec précision la réalité des attaques et des vulnérabilités dans les réseaux IoT.

En conséquence, bien que ce projet ait apporté des contributions significatives à la compréhension de l'application de l'apprentissage automatique dans la détection des intrusions IoT, il est clair que de futures recherches sont indispensables. L'exploration de jeux de données plus sophistiqués et représentatifs est essentielle pour une compréhension complète des capacités et des limites des différentes approches d'apprentissage automatique. Cela représente un champ prometteur pour les efforts futurs, visant à renforcer la résilience des systèmes IoT contre des adversaires de plus en plus avancés. La sécurité IoT est un domaine dynamique, où l'innovation continue est la clé pour rester en avance sur les menaces émergentes.

Bibliographie

[1] Alyazia Aldhaheri, Fatima Alwahedi, M. Ferrag, and A. Battah. Deep learning for cyber threat detection in iot networks : A review. *Internet of Things and Cyber-Physical Systems*, null :null, 2023.

[2] Abdullah Alsaedi, Nour Moustafa, Zahir Tari, Abdun Mahmood, and Adnan Anwar. Ton_iot telemetry dataset : A new generation dataset of iot and iiot for data-driven intrusion detection systems. *IEEE Access*, 8 :165130–165150, 2020.

[3] Juan Ignacio Iturbe Araya and H. Rifà-Pous. Anomaly-based cyberattacks detection for smart homes : A systematic literature review. *Internet Things*, 22 :100792, 2023.

[4] Cristiano Antonio de Souza, Carlos Becker Westphall, R. B. Machado, Leandro Loffi, C. Westphall, and G. Geronimo. Intrusion detection and prevention in fog based iot environments : A systematic literature review. *Comput. Networks*, 214 :109154, 2022.

[5] Mohamed Amine Ferrag, Othmane Friha, Djallel Hamouda, Leandros Maglaras, and Helge Janicke. Edge-iiotset : A new comprehensive realistic cyber security dataset of iot and iiot applications : Centralized and federated learning. 2022.

[6] Safwana Haque, F. El-Moussa, N. Komninos, and R. Muttukrishnan. A systematic review of data-driven attack detection trends in iot. *Sensors (Basel, Switzerland)*, 23 :null, 2023.

[7] Barjinder Kaur, S. Dadkhah, Farzaneh Shoeleh, E. P. Neto, Pulei Xiong, Shahrear Iqbal, Philippe Lamontagne, S. Ray, and A. Ghorbani. Internet of things (iot) security dataset evolution : Challenges and future directions. *Internet Things*, 22 :100780, 2023.

[8] François De Keersmaeker, Yinan Cao, Gorby Kabasele Ndonda, and R. Sadre. A survey of public iot datasets for network security research. *IEEE Communications Surveys Tutorials*, 25 :1808–1840, 2023.

[9] Jesús Fernando Cevallos Moreno, A. Rizzardi, S. Sicari, and A. Coen-Porisini. Deep reinforcement learning for intrusion detection in internet of things : Best practices, lessons learnt, and open challenges. *Comput. Networks*, 236 :110016, 2023.

[10] M. Papaioannou, G. Mantas, Aliyah Essop, V. Sucasas, N. Aaraj, and Jonathan Rodriguez. Risk estimation for a secure usable user authentication mechanism for mobile passenger id devices. *2022 IEEE 27th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, null :173–178, 2022.

[11] M. Papaioannou, G. Zachos, G. Mantas, and Jonathan Rodriguez. Novelty detection for risk-based user authentication on mobile devices. *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, null :837–842, 2022.

[12] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn : Machine learning in Python. *Journal of Machine Learning Research*, 12 :2825–2830, 2011.

[13] Mohammad Shahin, F. F. Chen, Hamed Bouzary, Aliakbar Hosseinzadeh, and Rasoul Rashidifar. A novel fully convolutional neural network approach for detection and classification of attacks on industrial iot devices in smart manufacturing systems. *The International Journal of Advanced Manufacturing Technology*, 123 :2017 – 2029, 2022.

[14] Lionel Sujay Vailshery. Number of internet of things (iot) connected devices worldwide from 2019 to 2023, with forecasts from 2022 to 2030. `https://www.statista.com/statistics/1183457/iot-connected-devices-worldwide/`, Last visited : 14 April 2024.