

UNIVERSITÉ MOHAMMED V de Rabat
Faculté des Sciences



Département d’Informatique

Filière Licence fondamentale
en Sciences Mathématiques et Informatique

PROJET DE FIN D’ÉTUDES

intitulé :

**Classification des intrusions dans l’environnement IoT
avec l’algorithme SVM**

Présenté par :
ASSIA EL-ATTARY
FATIMA BOURHIM

soutenu le 20 Juin 2024 devant le Jury

Mme. OUAFAA IBRIHICH Professeur à la Faculté des Sciences - Rabat *Présidente*
Mme. GHIZLANE ORHANOU Professeur à la Faculté des Sciences - Rabat *Encadrante*

Année universitaire 2023-2024

Remerciements

À travers ce mémoire, nous tenons à exprimer nos sincères remerciements à toute l'équipe pédagogique de la FSR et aux intervenants professionnels responsables de la Sciences mathématiques et informatiques pour nous avoir assuré une bonne formation.

Nous exprimons notre profonde gratitude et reconnaissance envers notre encadrante de la FSR, la professeure Mme GHIZLANE ORHANOU qui nous a fait bénéficier de ses conseils appréciables, de sa disponibilité, de son aide et pour l'intérêt manifeste qu'il a porté à ce projet.

Nous exprimons notre gratitude aux membres du jury qui se sont libérés de leurs obligations pour assister à la soutenance et juger notre travail. Nous sommes reconnaissants pour le temps et l'expertise qu'ils ont consacrés à évaluer notre projet, ainsi que pour les conseils et suggestions qui nous aideront à nous améliorer dans nos futures recherches.

Nous souhaitons également adresser nos sincères remerciements à nos chers parents, dont le soutien inébranlable et les sacrifices ont été la force motrice derrière la réalisation de ce travail. Leur amour, leurs encouragements et leurs conseils éclairés ont été pour nous une source infinie d'inspiration.

Enfin, merci infiniment à tous ceux qui ont contribué de près ou de loin à la réussite de ce travail, nous disons merci du fond du cœur.

Résumé

L'arrivée de l'Internet des Objets (IoT :Internet of Things) a modifié notre vie quotidienne en reliant de nombreux objets physiques à notre connexion Internet. Cela comprend des équipements allant des appareils domestiques aux équipements médicaux, en passant par les infrastructures industrielles. La collecte, l'analyse et la gestion autonomes des données sont des capacités remarquables d'IoT, mais cette technologie fait face à des défis importants en termes de sécurité, tels que les risques de vol et de manipulation des données.

Cette étude porte sur la sécurité de l'Internet des objets, en mettant l'accent sur la détection et la classification des intrusions. Grâce à l'algorithme Support Vector Machine (SVM), les scientifiques ont pour objectif de créer des solutions permettant de repérer et de classer les attaques visant les réseaux IoT.

Il y a trois chapitres dans notre projet de fin d'étude : le premier définit l'IoT, présente ses principes technologiques et identifie les attaques informatiques fréquentes contre ces systèmes.

Le deuxième chapitre examine les concepts d'intelligence artificielle et d'apprentissage automatique, en mettant en avant l'algorithme SVM.

Enfin, le troisième chapitre se focalise sur la mise en place de l'algorithme SVM pour la classification des intrusions dans les réseaux IoT, en détaillant les différentes étapes de mise en place.

Mots clés : IoT, SVM, Apprentissage Automatique, dataset, classification.

Abstract

The advent of the Internet of Things (IoT) has changed our daily lives by connecting many physical objects to the Internet connection. This includes equipment ranging from home appliances to medical equipment to industrial infrastructure. Autonomous data collection, analysis and management are remarkable capabilities of IoT, but it faces significant security challenges, such as the risks of data theft and manipulation.

This study focuses on Internet of Things security, with a focus on intrusion detection and classification. Using the Support Vector Machine (SVM) algorithm, scientists aim to create solutions to identify and classify attacks targeting IoT networks.

There are three chapters in our end-of-study project : the first defines IoT, presents its technological principles and identifies frequent computer attacks against these systems. The second chapter examines the concepts of artificial intelligence and machine learning, highlighting the SVM algorithm.

Finally, the third chapter focuses on the implementation of the SVM algorithm for the classification of intrusions in IoT networks, detailing the different implementation steps.

Keywords : IoT, SVM, Machine Learning, dataset, classification.

Table des matières

Remerciements	i
Résumé	ii
Abstract	iii
Introduction	ix
1 Sécurité de l'Internet des Objets (IoT)	2
1.1 L'internet des Objets	2
1.1.1 Les principes technologiques de l'IoT	2
1.1.2 Le modèle de l'IoT	5
1.1.3 Les secteurs de l'utilisation de l'IoT	6
1.2 Définition d'un Objet Connecté	10
1.2.1 Objet Connecté	10
1.2.2 Caractéristiques d'un objet connecté	11
1.2.3 Composants essentiels d'un objet connecté	11
1.3 Attaques informatiques visant l'IoT	12
1.3.1 Définition d'une attaque informatique	12
1.3.2 Types d'attaques sur l'IoT	12
2 Introduction à l'Apprentissage Automatique et l'Algorithme SVM	17
2.1 Intelligence Artificielle (IA)	17
2.1.1 Types d'Intelligence Artificielle	17
2.1.2 Cas d'utilisation de l'intelligence artificielle	18
2.1.3 Avantages de l'intelligence artificielle	19
2.2 L'Apprentissage Automatique	19
2.2.1 Catégories d'apprentissage automatique	20
2.2.2 Types d'algorithme d'apprentissage automatique	21
2.3 Support Vector Machine (SVM)	24
2.3.1 Définition de l'algorithme SVM	24
2.3.2 Types d'algorithmes SVM	24
2.3.3 SVM pour la classification	26
3 La Classification des intrusions avec l'algorithme SVM	33
3.1 Environnement de développement	33
3.1.1 Matériel utilisé	33

3.1.2	Langages de programmation et bibliothèques	33
3.1.3	Importation des bibliothèques	34
3.1.4	Le dataset CICIDS 2017	35
3.1.5	Construction du modèle	36
3.2	Les étapes de pré-traitement du dataset	37
3.2.1	pré-traitement du dataset CICIDS 2017-Friday	37
3.2.2	pré-traitement du dataset CICIDS 2017-Thursday	39
3.3	Entraînement de dataset et évaluation des performances	41
3.3.1	Classification des attaques DDoS du dataset CICIDS 2017-Friday Avec l'algorithme SVM	42
3.3.2	Classification des attaques DDoS du dataset CICIDS 2017- Thursday Avec l'algorithme SVM	46
Conclusion		52
Bibliographie		53

Table des figures

1.1.1 Réseau de capture sans fil	3
1.1.2 Big data	3
1.1.3 Fonctionnement de la technologie RFID	4
1.1.4 Machine To Machine	4
1.1.5 Cloud computing	5
1.1.6 L'architecture de l'Internet des Objets	6
1.1.7 Maison intelligente	7
1.1.8 Agriculture 2.0	7
1.1.9 Les services médicaux	8
1.1.10 Logistique intelligente	8
1.1.11 La ville intelligente	9
1.1.12 Industrie 4.0	9
1.1.13 Véhicule autonome	10
1.2.1 Objet connecté	10
1.3.1 Attaque Man-in-the-middle (MITM)	12
1.3.2 Attaque DDoS	13
1.3.3 Attaque Par force brute	13
1.3.4 Attaque par rejeu	14
1.3.5 Attaque par botnet	14
1.3.6 Attaque dans la couche application	15
1.3.7 Attaques d'usurpation d'adresse IP	15
2.2.1 Catégories d'apprentissage automatique	20
2.2.2 L'apprentissage supervisé et non supervisé	21
2.2.3 Exemple d'apprentissage par renforcement	21
2.2.4 La regression VS la classification	22
2.2.5 Exemple de la classification binaire	23
2.2.6 Exemple de la classification multiclasse	23
2.2.7 Exemple de La régression linéaire	24
2.3.1 Code d'entraînement de dataset avec LinearSVC	26
2.3.2 SVM avec LinearSVC (C=1)	27
2.3.3 SVM avec LinearSVC (C=50)	27
2.3.4 Code de découpage train-test	27
2.3.5 SVM Linéaire	28
2.3.6 Code d'entraînement de dataset avec le noyau polynomial	28
2.3.7 SVM avec noyau polynomial (degré=3)	29
2.3.8 SVM avec noyau polynomial (degré=8)	29

2.3.9 SVM avec Noyau Polynomial (degré=3)	29
2.3.10 Code d'entraînement de dataset avec noyau RBF	30
2.3.11 SVM avec Noyau RBF (gamma=5,c=0.001)	31
2.3.12 SVM avec Noyau RBF (gamma=15,c=0.009)	31
2.3.13 SVM avec Noyau RBF	31
 3.1.1 Caractéristiques du dataset utilisé	35
3.1.2 Chargement du dataset Friday	36
3.1.3 Chargement du dataset Thursday	36
3.1.4 Diagramme de flux de construction du modèle	36
3.2.1 Diagramme de flux de nettoyage des données	37
3.2.2 Code 1 du pré-traitement du dataset Friday	38
3.2.3 Code 2 du pré-traitement du dataset Friday	39
3.2.4 Code 3 du pré-traitement du dataset Friday	39
3.2.5 Code 1 du pré-traitement du dataset Thusday	40
3.2.6 Code 2 du pré-traitement du dataset Thusday	41
3.3.1 Code d'entraînement du dataset Friday avec SVM linéaire	42
3.3.2 Classification des attaques DDoS avec SVM linéaire	43
3.3.3 Code d'entraînement du dataset Friday avec SVM non-linéaire	44
3.3.4 Classification des attaques DDoS avec SVM non linéaire	45
3.3.5 Génération des données pour le SVM linéaire	46
3.3.6 code d'entraînement du dataset Thusday avec SVM lineaire	46
3.3.7 Classification des attaques Web avec algorithme SVM linéaire	47
3.3.8 Génération des données pour le SVM non-linéaire	48
3.3.9 Code d'entraînement Avec SVM non-lineaire	48
3.3.10 Code d'affichage du graphique	49
3.3.11 Classification des attaques Web avec algorithme SVM non linéaire	50

Liste des tableaux

3.1	Caractéristiques des matériels	33
3.2	Résultats des évaluations 1	43
3.3	Matrice de confusion 1	43
3.4	Résultats des évaluations 2	45
3.5	Matrice de confusion 2	45
3.6	Résultats des évaluations 3	47
3.7	Matrice de confusion 3	47
3.8	Résultats des évaluations 4	50
3.9	Matrice de confusion 4	50

Introduction

Au fil des dernières années, Internet a profondément modifié notre façon de vivre et continue de le faire, en particulier grâce à l'augmentation du nombre d'objets/appareils à notre disposition ou à ceux qui nous entourent qui peuvent se connecter à Internet.

De nos jours, l'Internet des Objets, également connu sous le nom d'Internet of Things (IoT) en anglais, fait référence à la fois au processus de connexion d'objets physiques à Internet et au réseau qui les relie.

On entend par « objets » à la fois des appareils de la vie quotidienne (domotique, montre de fitness, etc.) et des dispositifs médicaux, des machines agricoles, des chaînes d'approvisionnement, des robots industriels ou des feux de circulation routière.

En définitive, l'Internet des objets connecte tout objet capable de transmettre des données sur un réseau. Et ce, sans nécessiter d'interactions entre humains ou entre un humain et un ordinateur.

L'IoT possède une grande puissance en raison de sa capacité à communiquer, analyser, traiter et gérer des données de manière autonome. Cependant, de nos jours, il présente également des difficultés en matière de sécurité, les problèmes de sécurité entravent considérablement l'avancement et la mise en place rapide de cette technologie avancée. Les vols de données et la manipulation des données constituent un véritable risque pour ce genre de systèmes.

Toutefois, les experts dans ce domaine cherchent à trouver des solutions afin de garantir la sécurité des appareils connectés.

Dans notre projet, nous nous concentrerons sur la détection et la classification des intrusions en mettant l'accent sur la classification des attaques visant l'IoT en utilisant l'algorithme Support Vector Machine (SVM) avec ses deux versions : SVM Linéaire et SVM Non Linéaire.

Ce rapport est réparti en trois chapitres que nous détaillerons ci-dessous :

Chapitre1 : est consacré à définir l'internet des objets, citer ses principes technologiques, son modèle et les divers secteurs où il est largement utilisé, et il est aussi dédié à déterminer l'objet connecté en présentant ses caractéristiques et ses composants essentiels. En outre, nous décrivons les attaques informatiques ciblant l'IoT en identifiant

leur différents types.

Chapitre2 : est voué à détailler les notions de l'intelligence artificielle et l'apprentissage automatique, Nous mettons l'accent en suite sur l'algorithme Support Vector Machine (SVM).

Chapitre3 : est dédié à la classification des intrusions dans les réseaux IoT en utilisant l'algorithme SVM, en suivant les différente étapes de l'implémentation.

Chapitre 1

Sécurité de l'Internet des Objets (IoT)

1.1 L'internet des Objets

L'Internet des Objets, communément appelé en anglais Internet of Things (IoT) désigne une technologie d'avant-garde, où les objets traditionnellement non connectés qui nous entourent (comme des lampes, machines, vêtements, etc.), qu'ils soient physiques ou virtuels, ont désormais la capacité de communiquer entre eux en temps réel. Ce réseau d'objets connectés permet le partage de leurs données par l'intermédiaire d'une plateforme Cloud et ce, sans intervention humaine. C'est grâce à l'optimisation des interactions entre les humains et les machines et à la multiplication des flux de données, que les objets connectés offrent la possibilité de définir les besoins précis d'un individu, de sorte à lui offrir un bien ou un service unique [33].

1.1.1 Les principes technologiques de l'IoT

L'Internet des Objets (IoT) facilite l'interconnexion entre les divers objets intelligents via Internet. Son fonctionnement repose sur l'utilisation de multiples systèmes technologiques. Pour cela, l'IoT propose un éventail de solutions techniques telles que la RFID (Radio Frequency Identification), le TCP/IP, les technologies mobiles, etc., qui facilitent l'identification, la capture, le stockage, le traitement et le transfert des données, que ce soit dans des environnements physiques ou entre des contextes physiques et virtuels. Parmi les diverses technologies employées dans le cadre de l'IoT, certaines sont considérées comme étant fondamentales. Parmi elles, on retrouve les réseaux de capteurs sans fil (WSN) et la communication machine à machine (M2M), le cloud computing et big data qui seront définies par la suite [33].

— Réseau de capture (WSN - Wireless Sensor Network)

Le réseau de capteurs sans fil (WSN), comme le montre la figure 1.1.1, est constitué un ensemble de petits dispositifs intégrés, munis de capteurs autonomes et distribués dans l'espace afin de surveiller les conditions physiques environnementales. Ces réseaux peuvent collaborer avec les systèmes RFID pour assurer un suivi précis des mouvements, de la pression, de la température et de l'emplacement. La pertinence des réseaux de capteurs sans fil (WSN) dans le domaine de

l'Internet des Objets réside dans leur capacité à fournir des données, lesquelles sont exploitées dans divers secteurs tels que les soins de santé, les services gouvernementaux environnementaux (notamment dans les secours en cas de catastrophe naturelle), la défense(suivi et surveillance militaires), ainsi que dans les systèmes de maintenance [33].

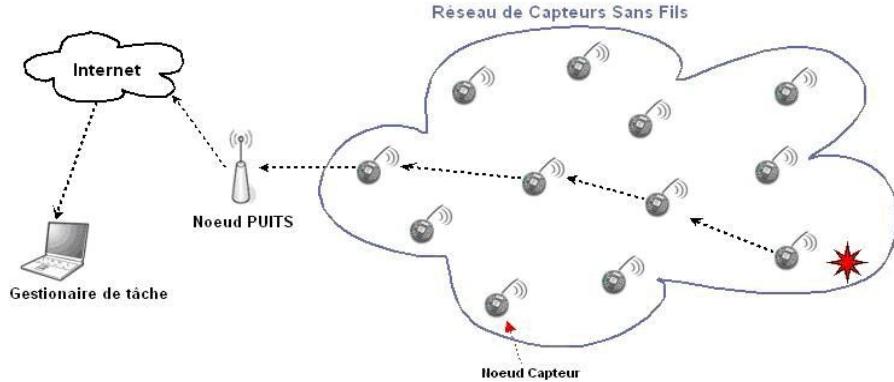


FIGURE 1.1.1 – Réseau de capture sans fil

— Big data

Big data est un système basé principalement sur 3 caractéristiques majeures, comme le montre la figure 1.1.2, qui sont : le volume, vitesse et les variétés connues par 3Vs du big data. Le volume dans big data présente la taille et la quantité de données qui peuvent être traiter à chaque seconde par les utilisateurs dans chaque domaine. Ensuite la variété des données volumineuses indique la complexité des données sous forme de données structurées, semi-structurées, non structurées ou mixtes, et enfin la vitesse qui présente une rapidité croissante dans la création des données les progrès technologiques et la nécessité correspondante de digérer et d'analyser ces données presque en temps réel [33].

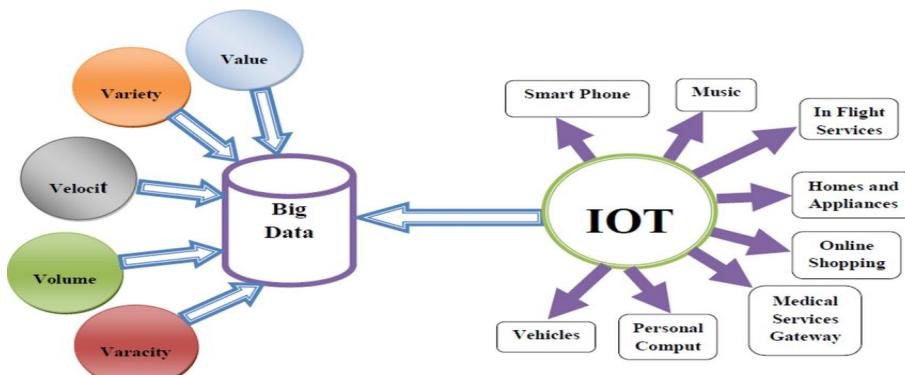


FIGURE 1.1.2 – Big data

— RFID

RFID est l'acronyme de Radio Frequency Identification, ce qui donne en français « identification par radiofréquence ». Il s'agit d'une technique qui exploite les ondes radio haute fréquence, comme le montre la figure 1.1.3, afin de transmettre et de stocker des informations dans le but d'identifier de façon unique les objets, les animaux ou les individus [26].

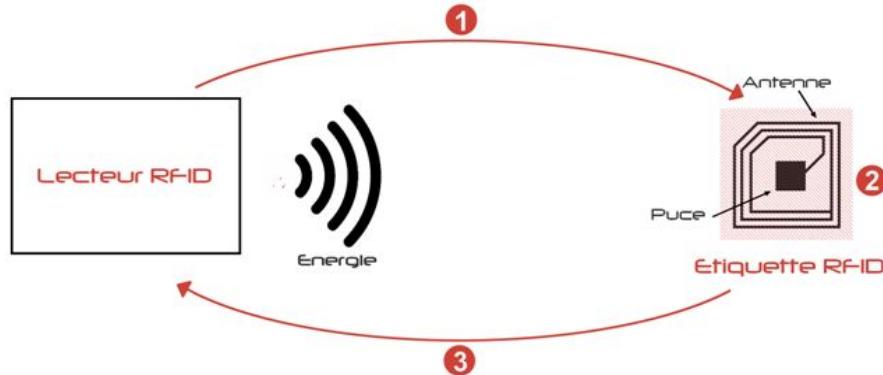


FIGURE 1.1.3 – Fonctionnement de la technologie RFID

— Machine To Machine

L'internet des objets n'est qu'une interaction entre appareils avec un échange de donnée. Pour cela M2M est nécessaire, c'est la communication de données entre les appareils sans la nécessité d'une interaction humaine, comme le montre la figure 1.1.4. M2M est une technologie de futur de l'IoT qui joue le rôle de la connectivité et l'intégration des machines informatisées, telles que les capteurs, les actionneurs, les contrôleurs et les robots [33].



FIGURE 1.1.4 – Machine To Machine

— Cloud Computing

Selon la définition de l’Institut National des Normes et des Technologies (NIST), le Cloud Computing est défini comme un modèle d’accès à la demande permettant d’utiliser un ensemble partagé de ressources configurables telles que des ordinateurs, des réseaux, des serveurs, du stockage, des applications, des services et des logiciels, comme le montre la figure 1.1.5. Ce modèle peut être fourni en tant qu’infrastructure, service ou logiciel, et il est spécifiquement adapté à la gestion d’une énorme quantité de données générées à partir d’appareils connectés à Internet dans le contexte de l’Internet des Objets. Le Cloud computing offre à l’IoT une solution idéale pour gérer des flux massifs de données et les traiter en temps réel, que ce soit pour des appareils IoT ou des utilisateurs humains [33].

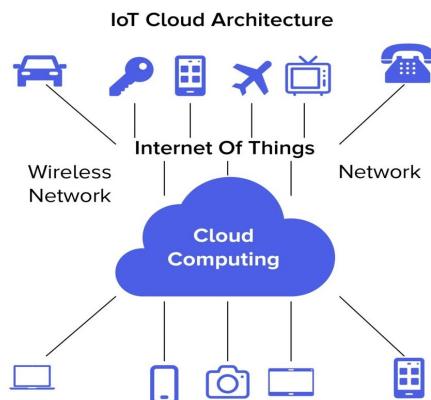


FIGURE 1.1.5 – Cloud computing

1.1.2 Le modèle de l’IoT

Le concept de l’Internet des objets a été l’objet des recherches depuis plus d’une décennie, mais même si, encore de nombreux aspects ne sont pas clairement définis. Par exemple, aujourd’hui il n’y a pas une architecture standardisée et spécifique pour l’IoT. Malgré ce manque de compatibilité, il y a une architecture à trois couches comme on a montré dans la figure 1.1.6 bien connue qui est généralement acceptée, ces couches sont : la couche de perception, la couche réseau et la couche d’application [33].

— La couche de perception

La tâche principale de la couche de perception est de reconnaître les propriétés physiques telles que la température, l’humidité, le niveau de la lumière, la vitesse, etc., par divers dispositifs de détection, et de convertir ces informations en signaux numériques. Les objets de cette couche peuvent avoir des capacités de détection et/ou des capacités d’actionnement. (Un actionneur est un dispositif qui peut recevoir des commandes programmées et effectuer des tâches à des moments précis) [1].

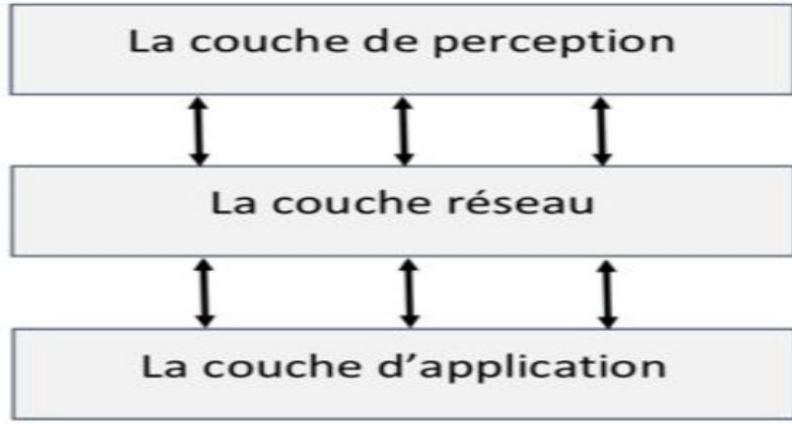


FIGURE 1.1.6 – L’architecture de l’Internet des Objets

— La couche réseau

La couche réseau est la couche responsable de la transmission des données reçues de la couche de perception à une base de données, serveur, ou d'un centre de traitement. Les principales technologies utilisées pour réaliser cette couche sont : les technologies cellulaires 2G / 3G / LTE, Wi-Fi, Bluetooth, Zigbee ou Ethernet, et avec ces différentes technologies on peut donc faire le traitement de plusieurs objets qui seront connectés à l'avenir [1].

— La couche d’application

La couche application analyse les informations reçues de la couche réseau. Cette couche fournit des applications pour toutes sortes de défis technologiques. Ces applications favorisent l’Internet des objets, ce qui explique pourquoi cette couche joue un rôle important dans la propagation de l’IoT [1].

1.1.3 Les secteurs de l’utilisation de l’IoT

— Maison intelligente

Une maison intelligente, ou « Smart Home », se distingue par l’utilisation d’appareils connectés à Internet qui nous offrent la possibilité de les contrôler à distance et de les surveiller, comme le montre la figure 1.1.7. La présence de la technologie intelligente peut varier, allant des lumières connectées intégrées à une enceinte connectée à une maison entièrement connectée grâce à plusieurs appareils intelligents. Un smartphone permet de contrôler à distance la plupart de ces appareils, tels que l’intensité de la lumière, la couleur d’une ampoule LED ou sa surveillance pour vérifier si elle est allumée ou éteinte. Si elle est utilisée avec une enceinte connectée, ces paramètres peuvent également être contrôlés par des commandes vocales [33].



FIGURE 1.1.7 – Maison intelligente

— Agriculture 2.0

L’agriculture intelligente est une méthode de production agricole utilisant des technologies comme l’Internet des Objets, la robotique, les drones et l’intelligence artificielle, afin de gérer les exploitations agricoles, accroître la production et diminuer les conséquences sur l’environnement. Elle vise à améliorer la qualité et la quantité des produits agricoles tout en optimisant le travail humain pour obtenir les meilleurs résultats. Les fermes intelligentes désignent les exploitations et les sites agricoles qui adoptent des méthodes d’agriculture intelligente. En général, Elles font appel à de nouveaux outils tels que des capteurs qui surveillent les conditions météorologiques 24 heures sur 24 et des applications d’inspection mobiles, comme le montre la figure 1.1.8, qui aident les agriculteurs à améliorer leurs méthodes agricoles pour augmenter leur production tout en favorisant la durabilité et en préservant la qualité des produits [6].



FIGURE 1.1.8 – Agriculture 2.0

— Les services médicaux

L’Internet des Objets (IoT) offre de nombreux avantages dans le domaine médical, permettant de réaliser des opérations médicales à distance et facilitant l’examen minutieux des patients. De plus, il a trouvé une application importante dans la gestion des stocks pharmaceutiques grâce aux applications utilisées par les pharmaciens. En outre, l’IoT joue un rôle essentiel dans le secteur de la santé, où il contribue à la collecte et à l’analyse de données à des fins de recherche et de surveillance [33].



FIGURE 1.1.9 – Les services médicaux

— Logistique intelligente

Une autre application possible de l’IoT est la logistique intelligente ou « logistique intelligente ». Il s’agit de l’application de la technologie Internet des objets dans le domaine de la logistique, comme le montre la figure 1.1.10. La « logistique » est la gestion des flux physiques entre tous les acteurs de la chaîne de production jusqu’au client final. Ces processus doivent respecter un cahier des charges adapté à la situation et au type de produit manipulé et déplacé, que ce soit en termes de rapidité, d’hygiène ou de sécurité. La logistique concerne à la fois la manutention des produits et la gestion et l’optimisation des stocks et du transport. En tirant parti des technologies spécifiques à l’IoT trouvées dans d’autres secteurs, il peut surveiller, superviser et optimiser les processus à tous les niveaux de la chaîne d’approvisionnement [2].



FIGURE 1.1.10 – Logistique intelligente

— Les villes intelligentes

Les villes intelligentes sont des villes ou des régions qui font appel à la connectivité, à des capteurs, à l’analyse des big data et à d’autres technologies intelligentes, comme le montre la figure 1.1.11 pour améliorer notre mode de vie, nos logements et déplacements. Alors les capteurs et les organisations de l’IoT peuvent aider à créer ce genre des villes où il existe des lumières intelligentes et une architecture respectueuse de la nature, ainsi que l’atténuation de la circulation et d’évitement des catastrophes naturelles, tout en améliorant les services publics et préservant la vitalité [5].



FIGURE 1.1.11 – La ville intelligente

— Industrie 4.0

On peut définir l'Industrie 4.0 comme la mise en œuvre de technologies numériques intelligentes dans la production et les processus industriels. Elle comprend différentes technologies telles que les réseaux industriels IoT, l'intelligence artificielle, le Big Data, la robotique et l'automatisation. Grâce à l'Industrie 4.0, la production devient plus intelligente et les usines deviennent plus intelligentes, comme le montre la figure 1.1.12. Son objectif est d'accroître la productivité, l'efficacité et la souplesse tout en offrant une prise de décision et une personnalisation plus intelligentes dans les opérations de production et de chaîne d'approvisionnement. En conclusion, il serait également important de souligner que toute définition de l'Industrie 4.0 doit prendre son nom de la quatrième révolution industrielle. Depuis les années 1800, trois révolutions industrielles ont eu lieu. Ce qu'on appelle ces « révolutions » industrielles, c'est que l'innovation qui a donné naissance à ces « révolutions » n'a pas simplement permis d'améliorer un peu la productivité et l'efficacité : elle a révolutionné la production et l'organisation du travail. Actuellement, nous nous trouvons dans la quatrième révolution industrielle, connue sous le nom d'Industrie 4.0 [4].



FIGURE 1.1.12 – Industrie 4.0

— Véhicule autonome

Un véhicule autonome est un type particulier de véhicule automobile qui, grâce à un système de contrôle automatique complexe, est capable de rouler dans des environnements de conduite normaux sans intervention humaine. Il est également capable de se connecter aux appareils à proximité via des réseaux sans fil. Ces véhicules constituent un facteur important dans le développement de l'Internet des objets. Les cas d'utilisation vont d'un système de divertissement connecté au téléphone d'un conducteur, comme on a montré dans la figure 1.1.13 à un véhicule connecté à Internet qui communique dans les deux sens avec d'autres véhicules,

appareils mobiles et intersections de la ville [3].



FIGURE 1.1.13 – Véhicule autonome

1.2 Définition d'un Objet Connecté

Un objet est une entité physique telle qu'un voiture, une couveuse, une machine à café, etc. L'objectif de l'internet des objets (IoT) est de transformer ces objets classiques en objets intelligents, interactifs ou connectés, leur permettant ainsi de communiquer entre eux [7].

1.2.1 Objet Connecté

Objet Connecté : est un objet physique équipé de capteurs ou d'une puce qui lui permettent de transcender son usage initial pour proposer de nouveaux services. Il s'agit d'un matériel électronique capable de communiquer avec un ordinateur, un smartphone ou une tablette via un réseau sans fil (Wi-Fi, Bluetooth, réseaux de téléphonie mobile, réseau radio à longue portée de type Sigfox ou LoRa, etc.), qui le relie à Internet ou à un réseau local [27].



FIGURE 1.2.1 – Objet connecté

On distingue communément deux grands groupes d'objets connectés :

- les objets destinés à **la collecte et l'analyse de données**, dont la mission principale est de collecter et transmettre des informations.
- les objets qui répondent à une logique de **contrôle-commande** et permettent de déclencher une action à distance [14].

1.2.2 Caractéristiques d'un objet connecté

- **Identification**

Chaque objet a son unique identification qui se représente dans un code barre, adresse IP ou puce RFID [33].

- **Sensibilité à son environnement**

Les caractéristiques des objets connectés comprennent la sensibilité de leur environnement par la détection, l'analyse, le traitement et la notification, où ils peuvent mesurer l'humidité, la température, la combustion de gaz et l'énergie consommée [33].

- **Interactivité**

La connexion entre l'objet et le réseau auquel il connecté est basée sur le besoin et la façon de travailler de l'objet, c'est-à-dire soit permanente soit brève [33].

1.2.3 Composants essentiels d'un objet connecté

Pour connecter les objets physiques à Internet, il est nécessaire d'utiliser un ensemble de dispositifs qui recueilleront les informations et l'état de chacun. Dans cette section, nous présentons une série d'équipements indispensables pour un objet connecté [33] :

- **Cartes Arduino** : sont des microcontrôleurs open-source permettant de prototyper et développer facilement des projets électroniques interactifs. Elles offrent une interface de développement intégrée, une simplicité d'utilisation et une grande flexibilité, favorisant la création de dispositifs allant des objets connectés simples aux systèmes plus avancés.
- **Capteurs** : sont des dispositifs électroniques intégrés aux objets physiques qui détectent et mesurent des données spécifiques, comme la température, ces capteurs transmettent ensuite ces données à d'autres appareils via des technologies sans fil telles que le Wi-Fi, le Bluetooth ou des réseaux comme l'IoT (Internet des Objets).
- **Cartes d'essai** : sont des plateformes matérielles préconçues utilisées pour développer des prototypes d'appareils connectés à l'Internet des objets (IoT). Leur but est de simplifier et d'accélérer le processus de développement en fournissant une base sur laquelle les développeurs peuvent expérimenter et tester leurs idées avant la production finale.
- **Les câbles** : sont des composants essentiels qui permettent la connexion et la communication efficace des appareils IoT. Ces câbles peuvent inclure des fonctionnalités telles que des connecteurs spéciaux, des adaptateurs pour différents types de interfaces (comme USB, micro-USB, USB-C, etc.), ou même des câbles combinant des technologies de communication comme l'alimentation électrique et les données sur un seul câble (comme les câbles Ethernet PoE).
- **LED'S** : (Light-Emitting Diodes) font référence à des diodes électroluminescentes intégrées dans des dispositifs connectés pour diverses fonctions visuelles. Ces LEDs peuvent servir à indiquer l'état de fonctionnement, à fournir des notifications ou à signaler des informations spécifiques à l'utilisateur. Elles sont souvent utilisées pour ajouter une interface visuelle intuitive et informative aux appareils connectés, améliorant ainsi l'expérience utilisateur et la gestion des appareils IoT.
- **Les résistances** : sont des composants électroniques utilisés pour limiter le courant électrique et ajuster les niveaux de tension dans les circuits. Elles sont essentielles

pour protéger les composants sensibles et assurer le bon fonctionnement des dispositifs connectés en régulant la quantité de courant qui circule à travers eux.

• **Les Plateformes d'IoT** : sont des infrastructures logicielles et parfois matérielles qui permettent de gérer, de connecter et de tirer des données des appareils IoT.

1.3 Attaques informatiques visant l'IoT

1.3.1 Définition d'une attaque informatique

Une attaque informatique fait référence à toute action visant à porter atteinte à un système informatique dans un but malveillant. Il s'agit d'ordinateurs ou de serveurs, isolés ou en réseau, connectés ou non à Internet, ainsi que d'équipements périphériques tels que les imprimantes, ou encore d'appareils communicants tels que les téléphones mobiles, les smartphones ou les tablettes [26].

1.3.2 Types d'attaques sur l'IoT

— Attaques l'homme au milieu (Man-in-the-middle)

Une attaque de l'homme du milieu ou Man-in-the-middle (MITM) est un genre de cyberattaque où les attaquants interceptent une conversation ou un transfert de données déjà en cours, soit en écoutant, soit en se faisant passer pour un participant légitime, comme le montre la figure 1.3.1. Selon la victime, il semble qu'il y ait un échange normal d'informations en cours, mais en s'infiltrant au cœur de la conversation ou du transfert de données, l'attaquant peut subtilement détourner des informations [8].



FIGURE 1.3.1 – Attaque Man-in-the-middle (MITM)

— Attaques par déni de service distribué (DDoS)

Les attaques par déni de service distribué (DDoS) sont une cybermenace qui submerge une ressource en ligne de trafic, empêchant le service Web de fonctionner normalement. Cette menace est capable de nuire considérablement à une entreprise, d'empêcher les utilisateurs d'accéder à des sites ou de ralentir considérablement le serveur Web au point qu'il devienne inaccessible, comme le montre la figure 1.3.2.

Ils peuvent se prolonger pendant plusieurs heures et, dans les situations graves, pendant plusieurs jours. Beaucoup d'entreprises et d'organisations ont une grande

dépendance envers leurs plateformes en ligne. Ainsi, cette attaque peut entraîner des répercussions significatives.

Les attaquants ont la capacité de prévoir de manière stratégique les attaques DDoS pendant des périodes clés. Par exemple, il est possible qu'un marchand en ligne soit victime d'une attaque lors d'une journée d'achats à grande échelle telle que le Black Friday [13].

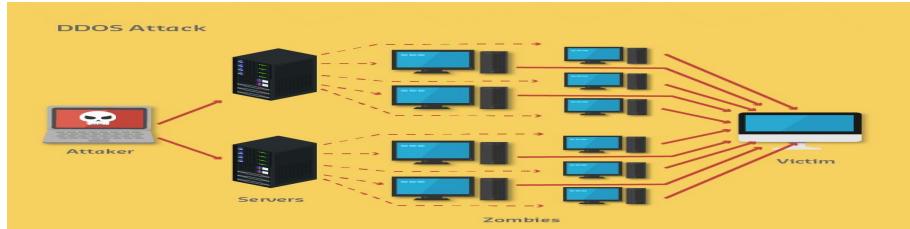


FIGURE 1.3.2 – Attaque DDoS

— Attaques par force brute

Les cybercriminels utilisent une attaque par bruteforce, ou force brute en français, pour déchiffrer les informations d'identification des comptes, notamment les mots de passe. L'attaquant, dans une attaque par force brute, possède habituellement un dictionnaire de termes et de mots de passe courants qu'il utilise pour « déchiffrer » le mot de passe d'un utilisateur. Une fois qu'une liste de mots du dictionnaire est épuisée, l'attaquant utilise des combinaisons de caractères jusqu'à ce qu'il trouve une correspondance. Il est possible que les attaquants aient besoin de plusieurs milliers de tentatives avant qu'un mot de passe ne soit compromis, c'est pourquoi ils utilisent des outils d'automatisation pour effectuer rapidement des milliers de tentatives, comme le montre la figure 1.3.3 [12].



FIGURE 1.3.3 – Attaque Par force brute

— Attaques par rejeu

L'attaque de relecture est une attaque dans laquelle un attaquant répète ou retarde une transmission valide et la retransmet frauduleusement, comme le montre la figure 1.3.4. En utilisant cette approche, un attaquant peut s'authentifier frauduleusement auprès d'un système, bien qu'il ne soit pas autorisé à le faire [9].

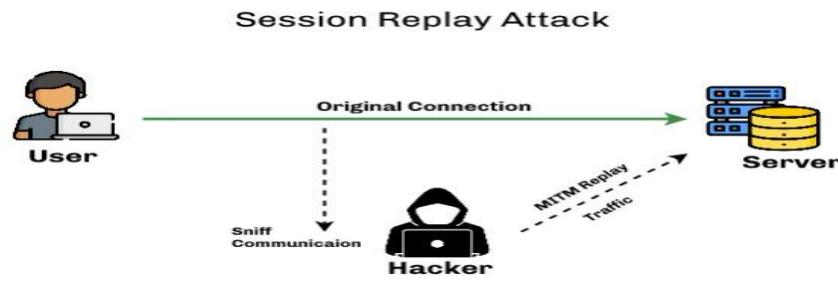


FIGURE 1.3.4 – Attaque par rejeu

— Attaques par Botnet

Une attaque par botnet désigne une attaque informatique à grande échelle qui est réalisée à l'aide d'appareils infectés par des logiciels malveillants et contrôlés à distance. Le contrôleur de botnet transforme les appareils compromis en « robots zombies ». À la différence d'autres logiciels malveillants qui se reproduisent sur une seule machine ou un seul système, les botnets représentent une menace plus significative car ils offrent à un acteur malveillant la possibilité d'accomplir un grand nombre d'actions simultanément. Les attaques de botnet sont similaires à l'intervention d'un acteur dangereux dans le réseau, contrairement à un logiciel malveillant auto-répliquant. Les attaques de logiciels malveillants deviennent de plus en plus complexes que les autres types, car elles peuvent être étendues ou modifiées à la volée afin de causer davantage de dommages. la figure 1.3.5 montre comment un botnet fonctionne [10].

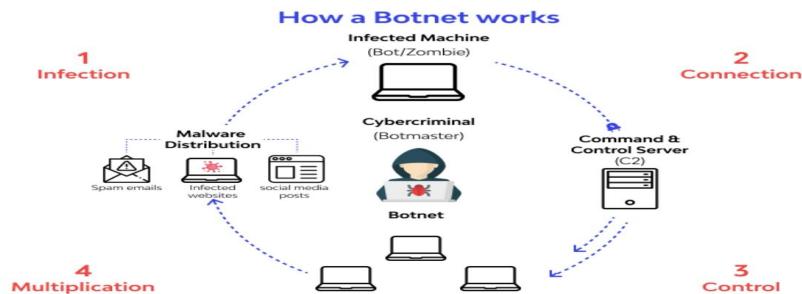


FIGURE 1.3.5 – Attaque par botnet

— Attaques dans la couche application

Différentes formes et méthodes peuvent être utilisées pour attaquer la couche application. Comme le grand public peut accéder aux serveurs web à des adresses de port connues, comme celles définies par des protocoles tels que le protocole HTTP (port 80), les pirates peuvent utiliser cette information pour lancer des attaques capables de contourner les pare-feu, comme le montre la figure 1.3.6. Dans la couche application, les attaques exploitent les lacunes du système d'exploitation et des applications pour accéder aux ressources. Une

configuration et une autorisation erronées peuvent engendrer des vulnérabilités en matière de sécurité [11].

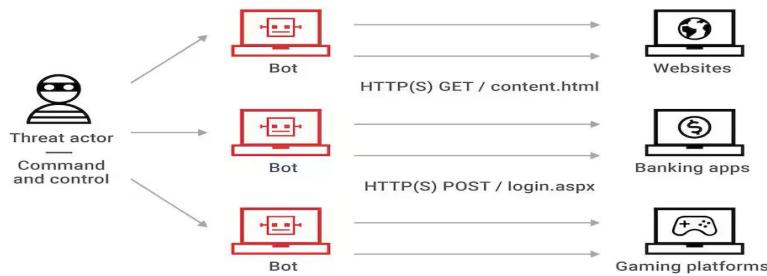


FIGURE 1.3.6 – Attaque dans la couche application

— Attaques d’usurpation d’adresse IP

Les attaques d’usurpation d’adresse IP sont complexes et exploitent les relations de confiance. L’attaquant, en utilisant des méthodes d’usurpation d’identité, utilise les identifiants d’un hôte pour compromettre sa sécurité. Cet hôte croit être en train de discuter avec un hôte de confiance. Au cours de cette attaque, l’attaquant débute en identifiant un hôte de confiance dont il va prendre l’identifiant. Ainsi, il est possible de commencer par identifier les séquences de confiance pour l’hôte. En général, il s’agit d’identifier la plage d’adresses IP à laquelle l’hôte peut avoir confiance. La prochaine étape est de désactiver l’hôte, car l’attaquant va prendre ses identifiants. Dans cette optique, l’attaquant a la possibilité d’utiliser des méthodes comme les attaques par inondation SYN TCP. La facilité de la falsification des adresses IP permet aux attaques par usurpation d’adresse IP de réussir, comme le montre la figure 1.3.7 [11].

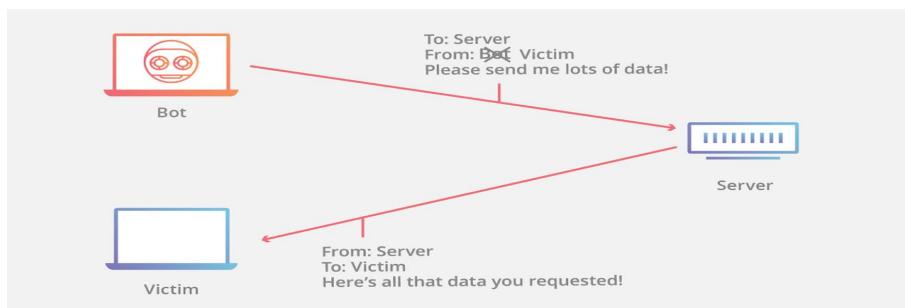


FIGURE 1.3.7 – Attaques d’usurpation d’adresse IP

Conclusion

En résumé, nous avons étudié l’univers de l’Internet des Objets (IoT) en examinant en détail ses caractéristiques technologiques, son modèle et ses nombreuses applications différentes. Au début, nous avons exposé les principes technologiques essentiels qui fondent l’IoT, en décrivant la manière dont les appareils connectés communiquent et interagissent au sein d’un réseau. Par la suite, le modèle de l’IoT

a été exposé, qui organise les diverses couches et interactions au sein de l'écosystème IoT. Nous avons aussi étudié les différents domaines où l'IoT est couramment employé, comme la santé, l'agriculture et les villes intelligentes, mettant en évidence les répercussions importantes de cette technologie sur différentes industries. En même temps, nous avons établi une définition d'un objet connecté, en mettant en évidence ses caractéristiques uniques et en décrivant ses éléments essentiels, tels que les capteurs, les actuateurs, et les modules de communication. En outre, nous avons abordé les risques et les faiblesses propres aux appareils IoT en définissant les attaques informatiques et en répertoriant les types d'attaques les plus fréquentes contre les systèmes IoT. cette analyse incluait une description des techniques utilisées par les cybercriminels pour exploiter les vulnérabilités de sécurité et mettre en péril les appareils connectés, mettant ainsi en évidence l'importance de la sécurité lors de la conception et de la mise en place de solutions IoT.

Chapitre 2

Introduction à l'Apprentissage Automatique et l'Algorithme SVM

2.1 Intelligence Artificielle (IA)

L'IA, également connue sous le nom d'intelligence artificielle, est une technologie qui permet aux ordinateurs et aux machines de reproduire l'intelligence humaine et les compétences en résolution de problèmes.

L'intelligence artificielle peut accomplir des tâches qui auraient autrement besoin de l'intelligence ou de l'intervention humaine. Les assistants virtuels, la navigation GPS, les véhicules autonomes et les outils d'intelligence artificielle générative (tels que Chat GPT d'Open AI) sont autant d'exemples de l'IA dans notre quotidien et notre vie de tous les jours [31].

2.1.1 Types d'Intelligence Artificielle

- **Intelligence artificielle étroite (Narrow AI)** : L'intelligence artificielle étroite, aussi connue sous le nom d'« IA faible », désigne la capacité d'un système informatique à accomplir une tâche spécifiquement définie de manière plus efficace qu'un être humain ne le peut. Le niveau le plus élevé de développement de l'intelligence artificielle que l'humanité a connu jusqu'à présent est l'intelligence artificielle étroite, et chaque exemple d'IA que vous observez dans la réalité est inclus dans cette catégorie, y compris les véhicules autonomes et les assistants numériques personnels. Cela s'explique par le fait que, bien que l'IA semble penser en temps réel pour elle-même, elle mène en réalité plusieurs processus étroits et prend des décisions dans un cadre préétabli. L'intelligence artificielle n'inclut ni la conscience ni l'émotion [15].
- **Intelligence générale artificielle (IA générale)** : L'intelligence artificielle générale (également connue sous le nom d'« IA forte » ou « IA de niveau humain ») désigne la capacité d'un système informatique à dépasser les humains dans toutes les tâches intellectuelles. Ce genre d'intelligence artificielle est présent dans les films où les robots possèdent des pensées sensibles et agissent de manière autonome. Selon la théorie, un système informatique qui a atteint

l'intelligence artificielle générale pourrait résoudre des problèmes extrêmement complexes, faire preuve d'un jugement dans des situations incertaines et intégrer des connaissances préalables à son raisonnement actuel. Sa créativité et son imagination seraient comparables à celles des êtres humains et elle pourrait prendre en charge un éventail de tâches bien plus vaste que l'IA étroite [15].

- **Super intelligence artificielle (ASI)** : Le système informatique équipé d'une intelligence artificielle extrêmement avancée pourrait dépasser les humains dans pratiquement tous les domaines, tels que la créativité scientifique, la sagesse générale et les compétences sociales [15].

2.1.2 Cas d'utilisation de l'intelligence artificielle

- **Reconnaissance vocale** : Également connue sous le nom de reconnaissance automatique de la parole (ASR), reconnaissance de la parole par ordinateur ou speech-to-text. Cet aspect utilise le traitement du langage naturel (NLP) afin de traiter la parole humaine dans un format écrit. Plusieurs smartphones intègrent la reconnaissance vocale dans leur système afin de réaliser des recherches vocales (par exemple Siri) ou d'améliorer l'accessibilité des textes [16].
- **Service client** : Les agents virtuels en ligne prennent le relais des agents humains tout au long de la trajectoire du client. Ils répondent aux questions couramment posées (FAQ) concernant des sujets tels que l'expédition ou offrent des conseils sur mesure, en proposant des produits en vente croisée ou en proposant des tailles pour les utilisateurs, ce qui modifie notre manière d'envisager l'engagement des clients sur les sites web et les plateformes de réseaux sociaux. Parmi les exemples, on retrouve les bots de messagerie sur les plateformes de commerce électronique avec des agents virtuels, les applications de messagerie comme Slack et Facebook Messenger, ainsi que les tâches habituellement accomplies par les assistants virtuels et les assistants vocaux [16].
- **Vision par ordinateur** : Les ordinateurs et les systèmes peuvent utiliser cette technologie d'intelligence artificielle pour extraire des informations importantes des images numériques, des vidéos et d'autres données visuelles, et prendre des mesures en fonction de ces informations. Elle se démarque des tâches de reconnaissance d'images par sa capacité à donner des recommandations. La vision par ordinateur, améliorée grâce aux réseaux neuronaux convolutifs, trouve des utilisations dans le marquage des photos sur les réseaux sociaux, l'imagerie radiologique dans le domaine de la santé et les voitures autonomes dans le domaine commercial [16].
- **Moteurs de recommandation** : Une des utilisations les plus fréquentes de l'intelligence artificielle est de suggérer des éléments en se basant sur les données historiques. Par exemple, lors d'une recommandation d'un service de diffusion multimédia en continu, le moteur de recommandation utilise l'intelligence artificielle pour analyser ce que vous avez regardé ou écouté précédemment, trier

toutes les options disponibles en fonction de leurs caractéristiques et mettre en évidence l'option la plus susceptible de vous divertir. Quand vous réalisez des achats sur un site Web et qu'il vous suggère d'ajouter des accessoires ou des articles connexes à votre panier, il utilise également l'intelligence artificielle de la même façon [15].

- **Analyse des données de santé** : Les institutions de santé à travers le monde font appel à l'intelligence artificielle afin de simplifier la recherche, les tests, le diagnostic, le traitement et la surveillance. Certains recourent à l'intelligence artificielle pour examiner des échantillons de tissus et établir des diagnostics plus précis. L'intelligence artificielle est employée par certaines entreprises afin d'analyser des données cliniques et repérer des faiblesses dans le traitement des patients. L'IA est employée par certaines entreprises pour analyser des milliards de composés, ce qui permet aux chimistes de réaliser des découvertes plus rapidement et de repérer les candidats appropriés pour les essais cliniques [15].

2.1.3 Avantages de l'intelligence artificielle

- **Communication à grande échelle** : Les entreprises peuvent offrir des conseils et une assistance à un plus grand nombre de personnes et d'endroits simultanément grâce aux robots et aux agents virtuels [15].
- **Automatisation des tâches répétitives** : En utilisant l'IA pour accomplir des tâches répétitives et chronophages, les employés des entreprises peuvent se focaliser sur un travail plus stratégique et plus impactant [15].
- **Des décisions plus rapides et plus précises** : L'intelligence artificielle permet de diminuer les erreurs humaines, ce qui la rend pratique pour prendre des décisions éclairées par des données et impliquant de nombreux calculs détaillés [15].

2.2 L'Apprentissage Automatique

L'Apprentissage Automatique (Machine Learning (ML)) implique l'utilisation de modèles mathématiques de données afin d'assister un ordinateur dans son apprentissage sans avoir besoin d'instructions directes. Il est perçu comme un élément essentiel de l'intelligence artificielle. Les algorithmes utilisés dans le ML permet de repérer des patterns dans les données, puis de les utiliser pour développer un modèle de données capable de faire des prédictions. Grâce à l'augmentation des données et de l'expérience, les résultats du ML sont plus précis, tout comme les êtres humains s'améliorent avec l'expérience. Le ML est une option idéale dans les situations où les données évoluent constamment, lorsque la nature de la demande ou de la tâche est toujours modifiée, ou lorsque

le codage d'une solution est toujours en constante évolution [17].

2.2.1 Catégories d'apprentissage automatique

Il existe trois catégories d'apprentissage automatique :

l'apprentissage supervisé, l'apprentissage non-supervisé, l'apprentissage par renforcement, comme illustré sur 2.2.1 ci-dessous .

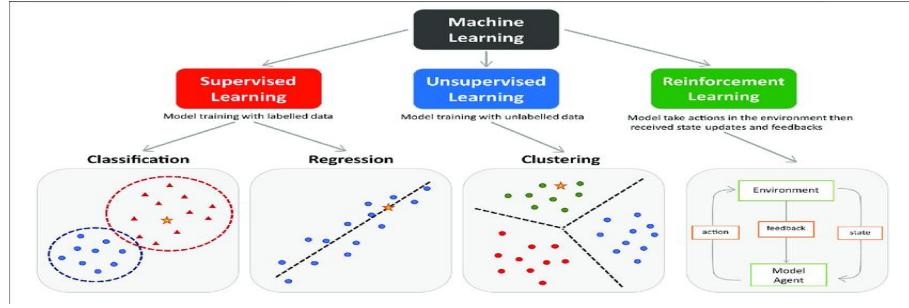


FIGURE 2.2.1 – Catégories d'apprentissage automatique

Apprentissage automatique supervisé :

Les algorithmes d'apprentissage automatique supervisé sont les plus répandus. Grâce à ce type de modèles, un chercheur en données joue le rôle de guide et enseigne à l'algorithme les conclusions qu'il doit tirer. De même qu'un enfant apprend à reconnaître les fruits en les mémorisant dans un livre d'images, donc l'algorithme est entraîné par un ensemble de données déjà étiqueté et avec un résultat préétabli. Ces algorithmes comprennent la régression linéaire et logistique, la classification multiclasse et les machines vectorielles de support [18].

Apprentissage automatique non supervisé :

L'apprentissage automatique non supervisé adopte une méthode plus autonome, où un ordinateur acquiert la capacité d'identifier des processus et des modèles complexes sans une supervision humaine étroite et continue. L'apprentissage automatique non supervisé consiste en une apprentissage basé sur des données qui ne présentent ni étiquettes ni résultat précis et défini. il est utilisé pour : le classement des données, le calcul approximatif de la densité de distribution, la réduction des dimensions.

Exemple : vous fournissez des données clients et souhaitez créer un segment de clients qui préfèrent des produits similaires. Les données que vous fournissez ne sont pas étiquetées, les étiquettes dans les résultats sont générées en fonction des similitudes entre les points de données détectés.

Parmi les algorithmes d'apprentissage automatique non supervisés, on peut citer :

le clustering k-moyennes, l'analyse des composants principaux et indépendants, ainsi que les algorithmes de clustering [18]. la figure 2.2.2 illustre la différence entre apprentissage supervisé et non supervisé.

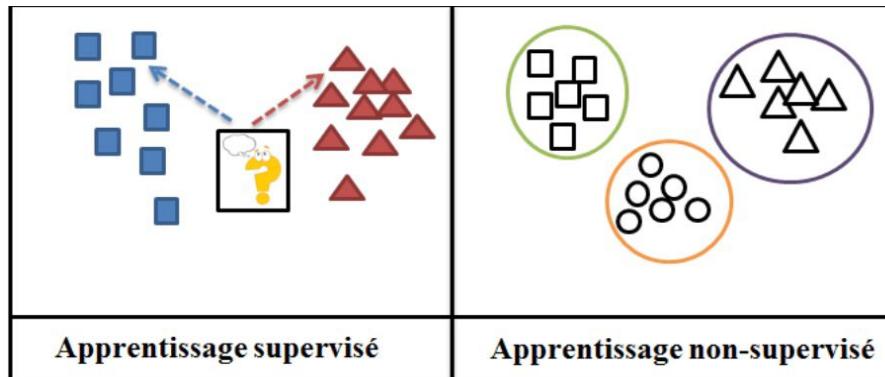


FIGURE 2.2.2 – L'apprentissage supervisé et non supervisé

Apprentissage par renforcement :

L'apprentissage par renforcement utilise des algorithmes qui apprennent des résultats et décident des actions à entreprendre ensuite. Après chaque action, l'algorithme reçoit un retour qui détermine si le choix effectué était correct, neutre ou incorrect, comme le montre la figure 2.2.3. C'est une solution idéale pour les systèmes automatisés qui doivent prendre un grand nombre de petites décisions sans instructions humaines.

Exemple : vous concevez une voiture autonome et vous voulez vous assurer qu'elle est conforme à la réglementation et assure la sécurité des passagers. Au fur et à mesure que la voiture acquiert de l'expérience et développe un historique de renforcement, elle apprend à rester dans sa voie, à respecter les limites de vitesse et à freiner pour laisser passer les piétons [17].



FIGURE 2.2.3 – Exemple d'apprentissage par renforcement

2.2.2 Types d'algorithme d'apprentissage automatique

Les algorithmes d'intelligence artificielle sont des éléments de programmation qui offrent aux utilisateurs la possibilité d'explorer, d'analyser et de déterminer

le sens de jeux de données complexes. Chaque algorithme consiste en une série restreinte d'instructions pas à pas claires qu'un ordinateur peut suivre afin d'atteindre un objectif spécifique. Dans un modèle d'apprentissage automatique, le but est de créer ou de trouver des modèles que les utilisateurs peuvent exploiter afin de prédire ou de classer des informations. Les paramètres qu'ils utilisent reposent sur les données d'apprentissage, un sous-ensemble de données qui constitue l'ensemble le plus significatif. Plus les données d'apprentissage s'étendent pour donner une représentation plus réaliste du monde, plus l'algorithme calcule des résultats précis.

On peut classer les algorithmes de machine learning en deux catégories :

- **Régression** :permettent de prédire une valeur précise,par exemple : Quel sera le cours des actions d'Apple demain.

- **Classification** :des méthodes permettant de prédire une catégorie , par exemple : si le cours de l'action d'Apple sera supérieur ou inférieur à celui d'aujourd'hui).

La figure 2.2.4 explique la différence entre la régression et la classification.

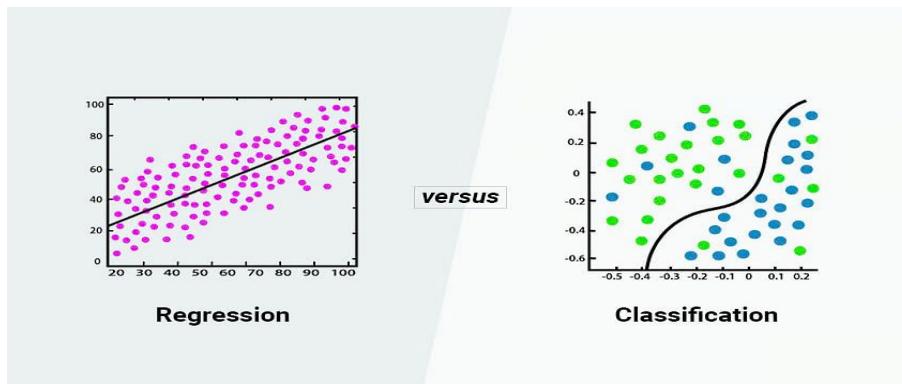


FIGURE 2.2.4 – La regression VS la classification

La classification « Donne-moi une catégorie »

La classification consiste en une méthode d'apprentissage automatique qui classe des éléments d'une collection en fonction de catégories ou de classes cibles.

l'objectif de cet algorithme consiste à anticiper de manière précise la classe cible pour chaque situation dans les informations. Prenons l'exemple d'un modèle de classification qui permet de repérer les demandeurs de prêt ayant un risque de crédit faible, moyen ou élevé [20].

Il existe deux principaux types de classification dans le Machine Learning :

- **les algorithme de classification à deux classes (binaires)** :

Les algorithmes de classification binaires permettent de séparer les données en deux groupes distincts. Ils sont bénéfiques pour les questions où seules deux réponses

sont possibles et s'excluent mutuellement, en particulier pour les questions de type oui/non [19]. la figure 2.2.5 montre comment le hyperplan sépare les deux classes binaire pour faire la classification.

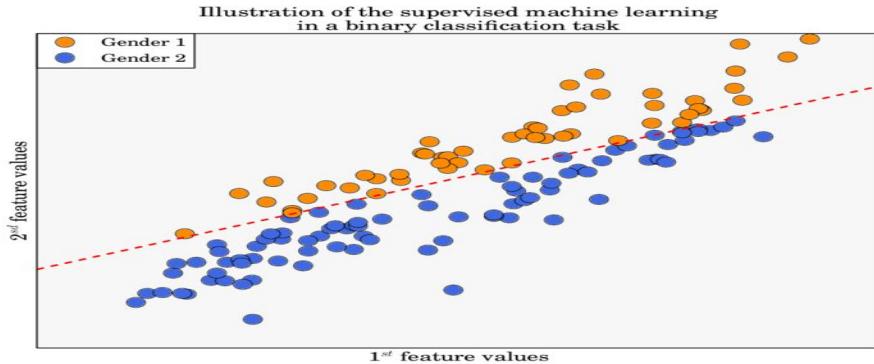


FIGURE 2.2.5 – Exemple de la classification binaire

- **Les algorithmes de classification multiclassse (multimultinomiale) :** Les méthodes de classification multiclassse (multimultinomiale) permettent de classer les données en trois ou plusieurs catégories, comme le montre la figure 2.2.6. Ils sont bénéfiques pour les questions avec au moins trois réponses possibles qui sont exclues les unes des autres [19].

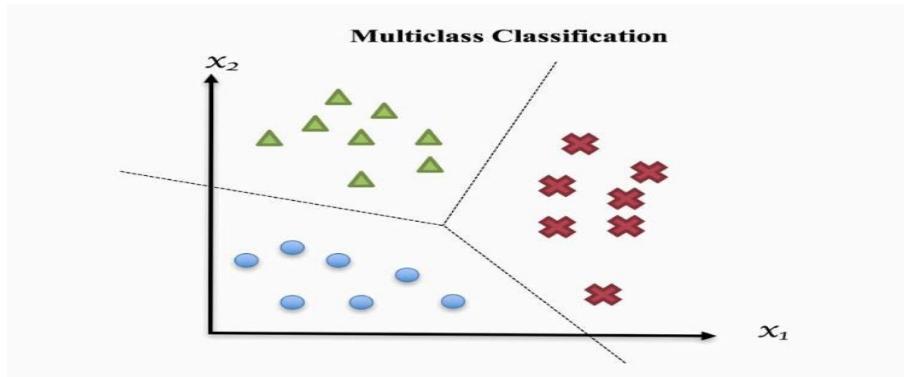


FIGURE 2.2.6 – Exemple de la classification multiclassse

La régression « Donne-moi un nombre »

Les algorithmes de régression utilisent des données d'historique pour anticiper la valeur d'un nouveau point de données. Ils abordent des interrogations comme : Quel sera le prix d'un appartement de trois pièces dans ma ville l'année prochaine ? Quelle sera la fréquentation de la clinique mardi ?

Parmis les types de régression dans le machine learning on trouve :

- **La régression linéaire :**

La régression linéaire est une technique statistique fréquemment utilisée dans le domaine de l'apprentissage automatique, puis a été enrichie par de nombreuses

nouvelles techniques d'ajustement de ligne de régression et de mesure des erreurs. En résumé, la régression désigne l'estimation d'une cible numérique. Il est toujours judicieux de choisir la régression linéaire lorsque vous désirez obtenir un modèle simple pour une tâche prédictive de base. Il est également courant que la régression linéaire fonctionne bien avec des jeux de données éparques et de grande taille peu complexes [21].

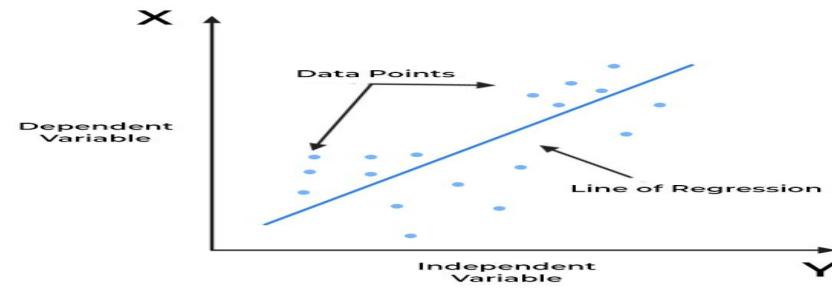


FIGURE 2.2.7 – Exemple de La régression linéaire

- **La régression logistique :**

La régression logistique est une méthode statistique bien établie qui permet de représenter les résultats binaires. elle est employée de diverses façons dans le domaine de la recherche statistique, avec des méthodes d'apprentissage variées. Une version de l'algorithme Microsoft Neural Network a été utilisée pour développer l'algorithme de régression logistique Microsoft. Cet algorithme ressemble à de nombreux réseaux neuronaux, mais son apprentissage est plus simple [22].

2.3 Support Vector Machine (SVM)

2.3.1 Définition de l'algorithme SVM

Le modèle SVM est très courant, et son fonctionnement repose sur la recherche d'un hyperplan adéquat pour diviser les échantillons de données collectés. La segmentation consiste à maximiser l'espace et à le transformer en un problème de programmation quadratique spécifique à résoudre. Les modèles principaux sont les suivants : En cas de partage de temps linéaire de l'échantillon d'apprentissage, il est recommandé d'utiliser la machine à vecteurs de support séparables linéaires afin de maximiser l'intervalle dur. En cas de partage de temps approximativement linéaire de l'échantillon d'apprentissage, il est recommandé d'utiliser la machine à vecteurs de support linéaire en maximisant l'intervalle logiciel et en choisissant la fonction de noyau adéquate. Si l'échantillon d'apprentissage est non partagé en temps linéaire, cela permet de maximiser l'intervalle logiciel et de choisir la meilleure option [23].

2.3.2 Types d'algorithmes SVM

Il existe deux types d'algorithme SVM : SVM linéaire et SVM non linéaire :

SVM linéaire

Le SVM linéaire est employé pour les données séparables de manière linéaire. Cela implique que si un ensemble de données peut être divisé en deux classes en utilisant une seule ligne droite, ces données sont désignées comme séparables de manière linéaire et le classificateur utilisé est appelé classificateur SVM linéaire [23].

On peut définir et entraîner le modèle SVM linéaire par deux méthodes :

Méthode 1 :

```
svm_clf_soft = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C=0.1, loss="hinge"))
])
```

Avec :

c est le l'hyperparamètre qui contrôle la force de régularisation..
loss ="hing" indique que la fonction d perte utilisée pour l'entraînement du modèle est la perte de charnière (hinge loss).

Méthode 2 :

```
pipeline = Pipeline([
    ("scaler", StandardScaler()),
    ("svm", SVC(kernel='linear'))
])
```

SVM non linéaire

Le SVM non linéaire est utilisé pour séparer les données d'une manière non linéaire. Cela implique que si un ensemble de données ne peut pas être classé à l'aide d'une ligne droite, ces données sont désignées comme données non linéaires et le classificateur utilisé est désigné comme non-linéaire [23].

Il contient deux noyaux de fonction : kernel polynomial et kernel rbf.

Polynomial kernel : Il s'agit d'une fonction non linéaire de noyau qui utilise des fonctions polynomiales pour transporter les données d'entrée dans un espace de fonctionnalités large.

Le noyau polynomial se définit par la fonction : $((\gamma(\mathbf{x}, \mathbf{x}') + \mathbf{r})^d)$, où \mathbf{d} est spécifié par le paramètre degré et \mathbf{r} par coef0.

La limite de décision d'un SVM à noyau polynomial pourrait générer des relations plus complexes entre les caractéristiques d'entrée, car il s'agit d'un hyperplan non linéaire. Son degré non-linéarité est déterminé par le degré du polynôme [25].

On peut le définir par :

```
poly_kernel_svm_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("svm_clf", SVC(kernel="poly", degree=n, coef0=m, C=i))
]).
```

Avec :

kernel="poly" signifie le noyau polynomial.
degree=n indique le degré du polynôme.
c indique le paramètre de régularisation .
coef c'est le paramètre qui contrôle la constante dans la fonction de noyau polynomial.

Kernel RBF : Le noyau gaussien, également connu sous le nom de noyau de fonction de base radiale (RBF), est une fonctionnalité de noyau couramment employée dans le domaine de l'apprentissage automatique, notamment dans les machines de support vectoriel. Ce type de fonction de noyau non linéaire utilise une fonction gaussienne pour organiser les données d'entrée dans un espace de fonctionnalités de dimension supérieure. Le noyau gaussien se définit par la fonction : $\exp(-\gamma\|x-x'\|^2)$, où γ est spécifié par le paramètre 'gamma' qui doit être supérieur à 0.

On peut le définir par :

("scaler", StandardScaler()),
("svm_clf", SVC(kernel="rbf", gamma=n, C=m)) .

Avec :

kernel="rbf" indique le noyau rbf.

gamma est le paramètre qui contrôle la largeur de la fonction de noyau gaussien.

c est le paramètre de régularisation .

2.3.3 SVM pour la classification

— SVM linéaire

Dans cette partie, nous avons utilisé le code Scikit-Learn sur la figure 2.3.1 qui charge le dataset iris.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import LinearSVC
# Chargement du dataset Iris :
iris = datasets.load_iris()
X = iris["data"][:, (2, 3)] # Utilisation des caractéristiques petal length et petal width
y = (iris["target"] == 2).astype(np.float64) # Classe binaire : 1 si Iris-Virginica, sinon 0
# Définition et entraînement du modèle SVM linéaire avec StandardScaler :
svm_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C=1, loss="hinge")),
])
svm_clf.fit(X, y)
# Plot des données et de la frontière de décision :
plt.figure(figsize=(8, 6))
# Création d'une grille pour le tracé
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.linspace(x_min, x_max, 100),
                      np.linspace(y_min, y_max, 100))
# Prédiction pour chaque point de la grille :
Z = svm_clf.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Tracé de la frontière de décision et des données :
plt.contourf(xx, yy, Z, cmap=plt.cm.bwr, alpha=0.8)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.bwr, edgecolors='k')
plt.xlabel('Petal Length')
plt.ylabel('Petal Width')
plt.title('Linear SVM Decision Boundary')
plt.colorbar(label='Predicted Class')
plt.show()
```

FIGURE 2.3.1 – Code d'entraînement de dataset avec LinearSVC

Puis Nous avons crée un pipeline où les caractéristiques sont d'abord mises à l'échelle avec StandardScaler avant d'être utilisées pour entraîner un modèle SVM linéaire (LinearSVC) avec un hyperparamètre de régularisation C de 1 et une fonction de perte hinge pour détecter les fleurs d'Iris-Virginica. Après nous avons changé le paramètre de régularisation C de 50. Les figures 2.3.2 et 2.3.3 montrent la différence entre les deux cas :

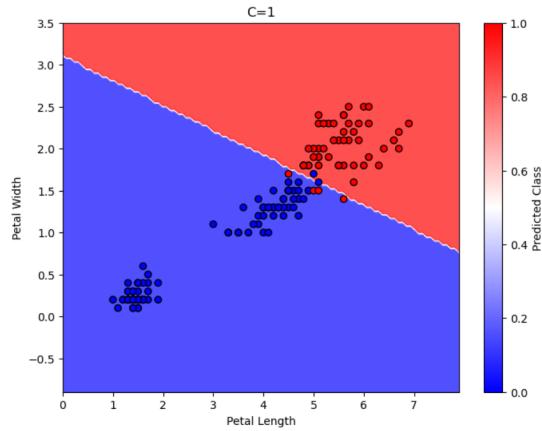


FIGURE 2.3.2 – SVM avec LinearSVC (C=1)

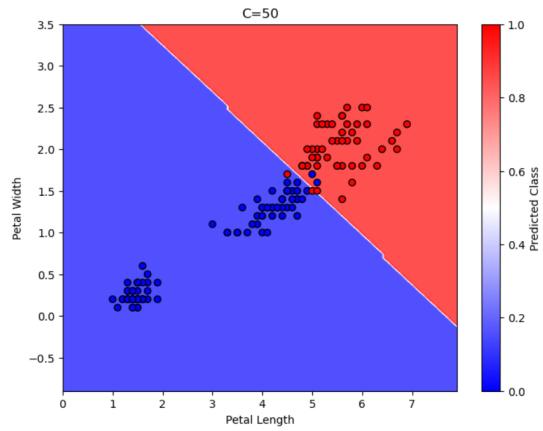


FIGURE 2.3.3 – SVM avec LinearSVC (C=50)

Après nous avons séparé le dataset 70% pour l'entraînement et 30% pour le test. Nous utilisons une technique appelée validation croisée ou découpage train-test. Grâce à cette technique, nous pouvons évaluer les résultats du modèle sur des données invisibles (ensemble de tests) après avoir été entraîné sur un ensemble différent (ensemble d'entraînement), comme le montre la figure 2.3.4.

```
# Division des données en ensembles d'entraînement (70%) et de test (30%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Définition et entraînement du modèle SVM linéaire avec StandardScaler :
svm_clf = Pipeline([
    ("scaler", StandardScaler()),
    ("linear_svc", LinearSVC(C=100, loss="hinge")),
])
svm_clf.fit(X_train, y_train)
# Prédiction sur l'ensemble de test :
y_pred = svm_clf.predict(X_test)
# Calcul de la précision du modèle :
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision du modèle : {accuracy:.2f}")
```

FIGURE 2.3.4 – Code de découpage train-test

Finalement, après l'entraînement du dataset nous avons affiché le graphique (figure 2.3.5) :

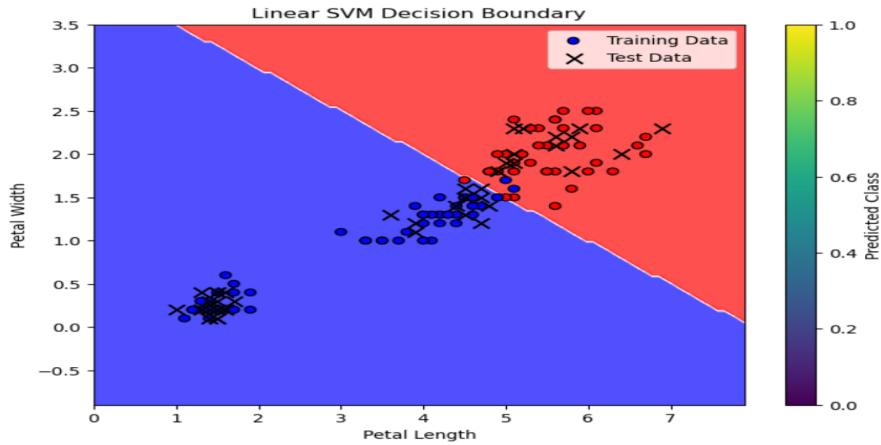


FIGURE 2.3.5 – SVM Linéaire

— SVM non linéaire

Nous avons implémenter et charger le dataset avec le noyau polynomial.

Polynomial Kernel (Noyau polynomial) :

La figure 2.3.6 représente le code où nous avons chargé le dataset. Puis, nous avons créé un pipeline qui commence par mettre à l'échelle les fonctionnalités avec StandardScaler() afin d'assurer une distribution uniforme. Ce modèle est utilisé pour des problèmes de classification linéaires où les classes ne peuvent pas être séparées linéairement dans l'espace des caractéristiques d'origine.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.datasets import make_moons
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
#Générer un jeu de données synthétique :
X, y = make_moons(n_samples=100, noise=0.15, random_state=42)
# Définir et entraîner le modèle SVM avec noyau polynomial :
poly_kernel_svm_clf = Pipeline([
    ("scaler", StandardScaler()), # Étape de mise à l'échelle des fonctionnalités
    ("svm_clf", SVC(kernel="poly", degree=3, coef0=1, C=5)) # Modèle SVM avec noyau polynomial
])
```

FIGURE 2.3.6 – Code d'entraînement de dataset avec le noyau polynomial

Pour effectuer une classification, on utilise un algorithme SVM avec un noyau polynomial de degré 3, un coefficient0 de 1 et un paramètre de régularisation

'C' de 5, ensuite on change les paramètre de degré 8 . Les figues 2.3.7 et 2.3.8 montrent la différence entre les deux cas :

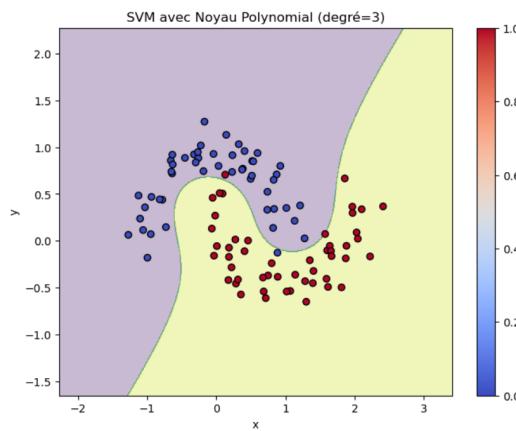


FIGURE 2.3.7 – SVM avec noyau polynomial (degré=3)

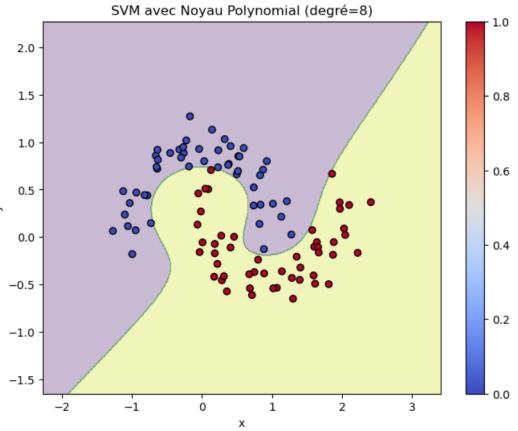


FIGURE 2.3.8 – SVM avec noyau polynomial (degré=8)

Après la séparation de dataset, 70% pour l'entraînement et 30% pour le test par la technique de écoupage train-test, nous aurons le graphique suivant :

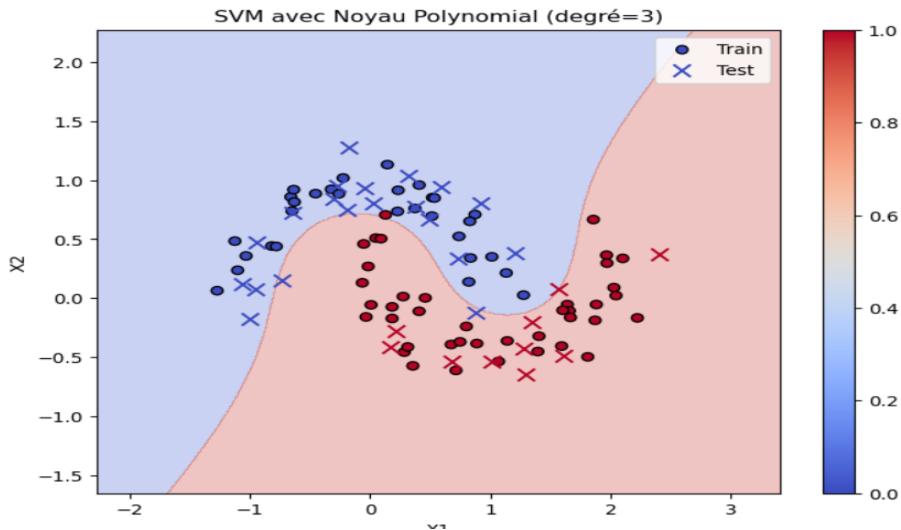


FIGURE 2.3.9 – SVM avec Noyau Polynomial (degré=3)

Gaussian (RBF) Kernel :

Dans la figure 2.3.10, Nous avons créé un pipeline qui effectue la mise à l'échelle des fonctionnalités pour entraîner les données synthétique, après il utilise un algorithme SVM avec un noyau RBF, ce dernier est capable de modéliser des relations non linéaires en projetant les données dans un espace de caractéristiques de dimension supérieur où les classes peuvent être séparées de manière linéaire.

```

X, y = make_moons(n_samples=100, noise=0.15, random_state=42)
# Division des données en ensembles d'entraînement (70%) et de test (30%):
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Définir le pipeline pour un SVM avec noyau RBF et mise à l'échelle des données :
rbf_kernel_svm_clf = Pipeline([
    ("scaler", StandardScaler()), # Étape de mise à l'échelle des fonctionnalités
    ("svm_clf", SVC(kernel="rbf", gamma=5, C=0.001)) # SVM avec noyau RBF
])
# Entraîner le modèle sur les données :
rbf_kernel_svm_clf.fit(X, y)
# Prédire les étiquettes sur l'ensemble de test avec les meilleurs paramètres :
y_test_pred = grid_search.best_estimator_.predict(X_test)
# Calculer l'exactitude (accuracy) sur l'ensemble de test :
accuracy_test = accuracy_score(y_test, y_test_pred)
print("Exactitude (Accuracy) sur l'ensemble de test : {:.2f}".format(accuracy_test))
# Créer un meshgrid pour visualiser la frontière de décision
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.01),
                      np.arange(y_min, y_max, 0.01))
Z = grid_search.best_estimator_.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)
# Afficher le graphique avec la frontière de décision, les données d'entraînement et de test :
plt.figure(figsize=(8, 6))
plt.contourf(xx, yy, Z, alpha=0.3, cmap=plt.cm.coolwarm) # Contourf pour la frontière de décision
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.coolwarm, edgecolors='k', label='Training Data')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=plt.cm.coolwarm, marker='x', s=80, label='Test Data') # Pour les points de test
plt.title("SVM avec Noyau RBF (gamma=5 , C=0,001)")
plt.xlabel("X1")
plt.ylabel("X2")
plt.colorbar()
plt.legend()
plt.show()

```

FIGURE 2.3.10 – Code d'entraînement de dataset avec noyau RBF

Pour effectuer une classification pour l'ensemble d'entraînement, on utilise un algorithme SVM avec un noyau RBF avec les hyperparamètres 'gamma' égale à 5 et 'c' égale à 0.001, ensuite on change les hyperparamètre à 'ganma' égale à 15 et 'c' égale à 0.009. Les figures 2.3.11 et 2.3.12 montrent la différence entre les deux cas :

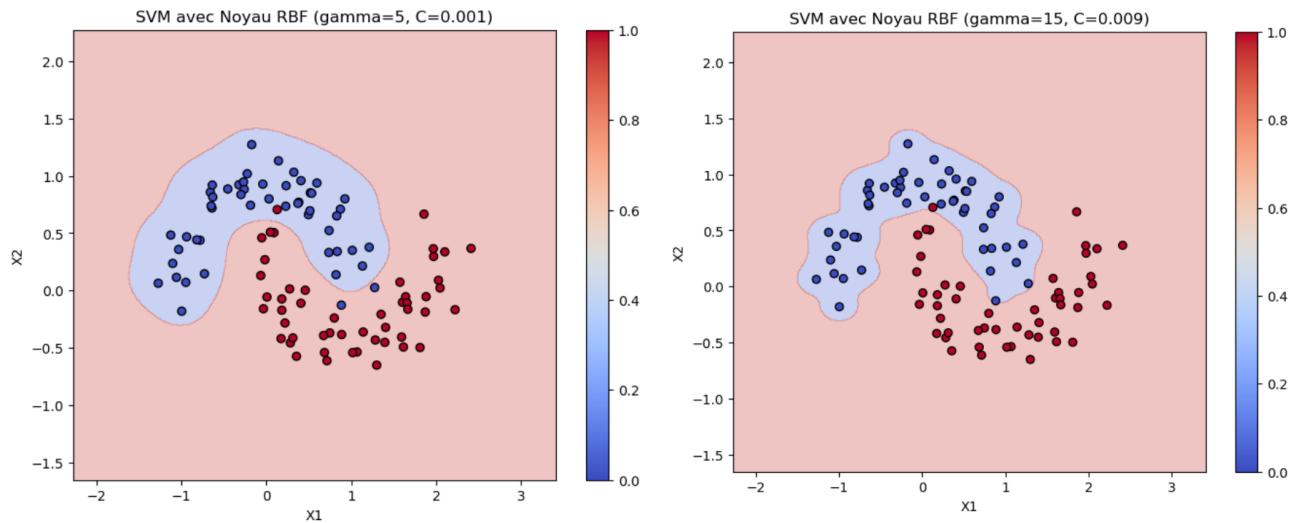


FIGURE 2.3.11 – SVM avec Noyau RBF (gamma=5,c=0.001)

FIGURE 2.3.12 – SVM avec Noyau RBF (gamma=15,c=0.009)

Finalement nous avons séparé le dataset 70% pour l'ensemble d'entraînement et 30% pour le test. Cela nous donne ce résultat illustré sur la figure 2.3.13 :

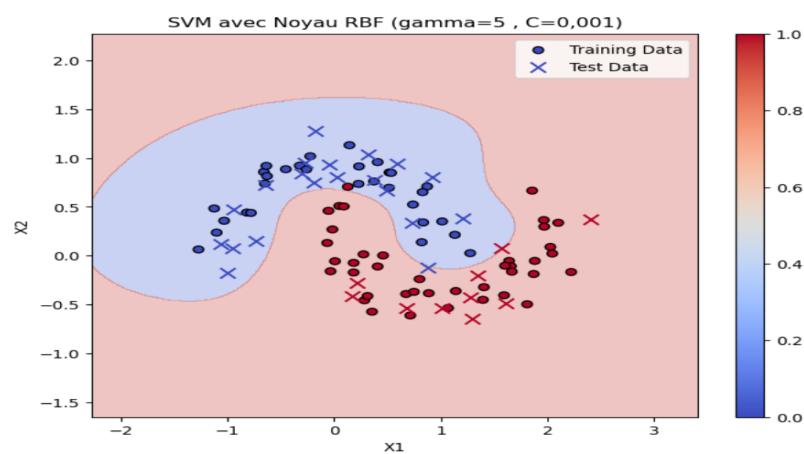


FIGURE 2.3.13 – SVM avec Noyau RBF

Conclusion

Dans ce chapitre, nous avons exploré le domaine fascinant de l'intelligence artificielle (IA) et de l'apprentissage automatique (ML). Nous avons commencé par définir l'IA comme des programmes conçus pour imiter l'intelligence humaine à travers des algorithmes de calcul, mettant en lumière son développement rapide depuis 2010, largement stimulé par le big data. En examinant les différentes formes d'IA, nous avons distingué l'IA étroite, générale et superintelligente, en illustrant chacune avec des exemples concrets de leur utilisation dans divers domaines tels que la reconnaissance vocale, le service client, la vision par ordinateur et les moteurs de recommandation. Nous avons également souligné les avantages de l'IA, notamment sa disponibilité 24 heures sur 24, sa capacité à communiquer à grande échelle et son rôle essentiel dans l'automatisation des tâches répétitives, conduisant ainsi à des prises de décision plus rapides et plus précises. En abordant l'apprentissage automatique, nous avons examiné ses différents types, notamment supervisé, non supervisé et par renforcement, ainsi que les algorithmes associés tels que la régression linéaire, la régression logistique et les SVM. Dans la première partie, nous avons exploré l'utilisation des algorithmes SVM pour la classification, en mettant l'accent sur le SVM linéaire. Nous avons utilisé le jeu de données Iris pour entraîner un modèle SVM linéaire avec différents paramètres de régularisation et avons visualisé les résultats à l'aide de graphiques. Ensuite, dans la deuxième partie, nous nous sommes tournés vers le SVM non linéaire en utilisant des noyaux polynomiaux et gaussiens (RBF). Toujours en utilisant le même jeu de données Iris, nous avons entraîné des modèles SVM avec différents paramètres de degré pour le noyau polynomial et différents paramètres de gamma pour le noyau RBF. Enfin, Les résultats ont été visualisés à l'aide de graphiques pour illustrer la capacité de séparation des différents modèles.

Chapitre 3

La Classification des intrusions avec l'algorithme SVM

3.1 Environnement de développement

3.1.1 Matériel utilisé

Dans ce qui suit, nous allons présenter les configurations des machines sur lesquelles notre système a été développé :

Matériels	Caractéristiques
PC 1	Processeur : Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz Mémoire (RAM) : 16 Go Disque Dur : 256 Go SSD Système d'exploitation : Windows 10 Professionnel
PC 2	Processeur : Intel(R) Core(TM) i5-3320M CPU @ 2.60GHz Mémoire (RAM) : 4 Go Disque Dur : 196 Go SSD Système d'exploitation : Windows 10 Pro

TABLE 3.1 – Caractéristiques des matériels

3.1.2 Langages de programmation et bibliothèques

Nous avons utilisé Jupyter notebooks dans l'environnement Anaconda, et le language Python pour développer nos modèles :

Anaconda : est une distribution gratuite et open source des langages de programmation Python et R appliquée au développement d'applications dédiées à l'apprentissage automatique et à la science des données. Elle fournit un environnement complet et préconfiguré avec de nombreuses bibliothèques et outils essentiels pour ces domaines [29].

Jupyter : a été créé en 2014 par une équipe de développeurs dirigée par Fernando Pérez, un physicien et informaticien. C'est un environnement de développement interactif largement utilisé pour l'exécution de code, la visualisation de données et l'analyse de données. Il permet de créer et de partager des documents appelés "notebooks" qui contiennent à la fois du code exécutable, des résultats, des visualisations, du texte explicatif et des éléments interactifs. L'un des principaux avantages de Jupyter est sa prise en charge de multiples langages de programmation, notamment Python, R, Julia et d'autres [30].

Python : est un langage de programmation interprété, orienté objet, de haut niveau et à sémantique dynamique. La syntaxe de Python, simple et facile à apprendre, privilégie la lisibilité et réduit donc le coût de la maintenance des programmes. IL prend en charge les modules et les packages, ce qui encourage la modularité des programmes et la réutilisation du code [32].

3.1.3 Importation des bibliothèques

Nous commencerons par importer les bibliothèques nécessaires suivantes :

- **Pandas** : permet la manipulation et l'analyse des données et offre des structures de données et des opérations pour travailler avec des tableaux de chiffres.
- **Numpy** : est une bibliothèque Python pour le calcul numérique qui permet la manipulation rapide et efficace de tableaux multidimensionnels.
- **Scikit-Learn** : nous permet d'expérimenter différentes techniques et algorithmes d'apprentissage automatique et d'analyser les données prédéfinis rapidement et facilement.
- **matplotlib** (importé sous le nom plt) pour tracer les graphiques et les figures dans Python.
- **sklearn.preprocessing** : pour la normalisation des données.
- **sklearn.model_selection** : pour la division des données en ensembles d'entraînement et de test, ainsi que la validation croisée.
- **sklearn.metrics** : pour l'évaluation des modèles.
- **sklearn.svm import SVC, LinearSVC, SVR** : Ces classes sont utilisées pour des tâches de classification et de régression à l'aide de machines à vecteurs de support (SVM).
- **sklearn.datasets import make_classification** : Cette fonction est utilisée pour générer des ensembles de données de classification synthétiques avec des paramètres spécifiés.
- **sklearn.impute import SimpleImputer** : est utilisée pour imputer les valeurs manquantes dans les ensembles de données.
- **sklearn.preprocessing import LabelEncoder** : est utilisée pour encoder les variables catégorielles en étiquettes numériques.
- **precision_score, recall_score, f1_score** : Ces fonctions sont utilisées pour évaluer les performances des modèles d'apprentissage automatique, y compris les tâches de régression (mean_squared_error) et de classification (accuracy_score, confusion_matrix, precision_score, recall_score, f1_score) [34].

Le choix des bases de données(dataset) à utiliser est l'une des étapes les plus cruciales pour évaluer et valider les méthodes de détection d'attaques basées sur l'apprentissage automatique en cybersécurité.

Dans la sous-section suivante, nous allons présenté la base de données utilisée pour réaliser notre implémentation.

3.1.4 Le dataset CICIDS 2017

Il s'agit d'un ensemble de données créé pour la détection des attaques dans l'internet des objets. Il intègre également les résultats de l'analyse du trafic réseau à l'aide de CICFlowMeter (CICFlowMeter est un générateur et analyseur de flux de trafic réseau) avec des flux étiquetés en fonction de l'horodatage, des adresses IP source et de destination, des ports source et de destination, des protocoles et des attaques (fichiers CSV).

Les attaques mises en œuvre incluent Force Brute FTP, Force Brute SSH, DoS, Heartbleed, Web Attack, Infiltration, Botnet et DDoS [26].

Le tableau ci-dessous contient les informations nécessaire (nom du dataset, sa date, sa taille et sa description) pour les parties de la base de données CICIDS2017 que nous avons utilisées dans notre implémentation :

Dataset	date	Taille	description
CICIDS 2017-Friday	2017	(225745, 79) Environ 2Go.	Il contient des enregistrement de trafic réseau tel que (Destination Port, ' Flow Duration', ' Total Fwd Packets', 'Total Backward Packets',etc..) qui sont capturé durant l'après-midi d'un vendredi de travail .celui-ci inclut un volume significatif de trafic lié à des attaques de scan de ports et des trafic normale.
CICIDS 2017-Thursday	2017	(3394 , 79) Environ 2Go.	Il contient des enregistrement de trafic réseau tel que (Destination Port, ' Flow Duration', ' Total Fwd Packets', 'Total Backward Packets',etc..) qui sont capturé Durant une matinée de travail,avec un accent particulier sur les attaques web(brute force,XSS, SQL Injection).

FIGURE 3.1.1 – Caractéristiques du dataset utilisé

Nous chargeons les données du dataset à partir de fichiers CSV. Nous avons utilisé les fichiers suivants :

'Friday_WorkingHours_Afternoon_DDoS.pcap_ISCX.csv' : il concerne l'attaque DDoS, comporte 2 classes, trafic normal Benign et l'attaque DDoS. La figure suivante montre comment on a chargé ce dataset dans jupyter :

```
file_path = 'Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv'
data = pd.read_csv(file_path)
```

FIGURE 3.1.2 – Chargement du dataset Friday

'Thursday_WorkingHours_Morning_WebAttacks.pcap_ISCX.csv' : Il concerne les attaques Web, comporte 4 classes, trafic normal, Benign et 3 attaques (Brute Force, XSS, Sql Injection). La figure suivante montre comment on a chargé ce dataset dans jupyter :

```
file_path = 'Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv'
data = pd.read_csv(file_path)
```

FIGURE 3.1.3 – Chargement du dataset Thursday

3.1.5 Construction du modèle

Dans cette sous-section, nous présentons un diagramme explicatif des étapes suivies pour la réalisation de notre implémentation.

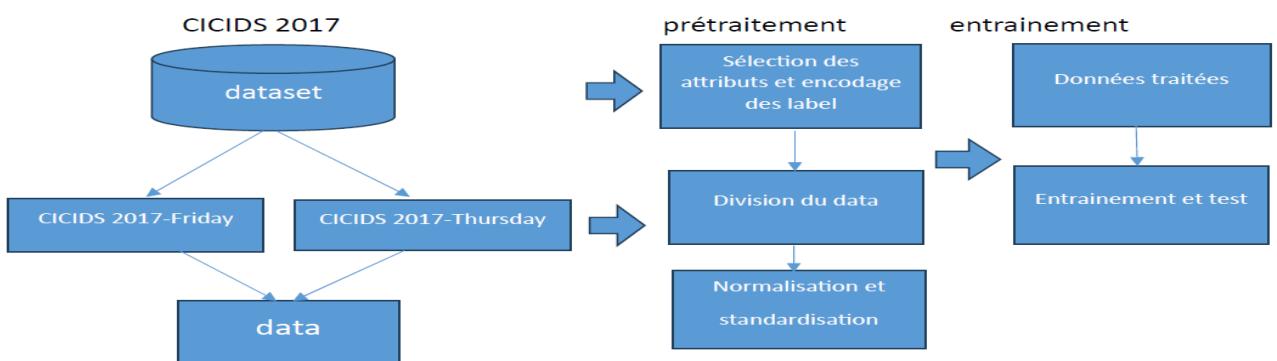


FIGURE 3.1.4 – Diagramme de flux de construction du modèle

3.2 Les étapes de pré-traitement du dataset

Dans cette section, nous décrivons en détail les étapes de pré-traitement que nous avons suivi pour nettoyer notre dataset.

La première étape pour préparer les ensembles de données pour l'apprentissage automatique consiste à nettoyer les données en supprimant les valeurs aberrantes, infinies et NaN (non numériques).

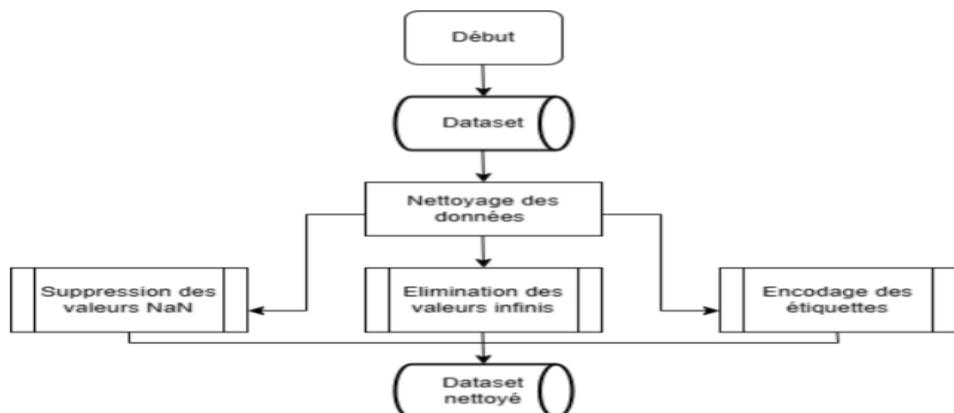


FIGURE 3.2.1 – Diagramme de flux de nettoyage des données

3.2.1 pré-traitement du dataset CICIDS 2017-Friday

Tout d'abord, après le nettoyage du dataset CICIDS 2017-Friday, nous affichons les types de données de chaque colonne en utilisant `data.dtypes` nous trouvons deux types (`int64`,`float64`).

les dix premières lignes en utilisant `data.head(10)`.

la description statistique des données avec `data.describe()`.

les noms des colonnes à l'aide de `data.columns` .

le nombre total des valeurs manquantes dans notre dataset avec `data.isnull().sum().sum()`.

Ensuite, nous convertissons le dataset pour un affichage tabulaire avec `shape_data=pd.DataFrame(shape_data)`.

De plus, nous identifions les caractéristiques numériques et catégorielles, puis nous imputons les valeurs manquantes en utilisant **La médiane** et les caractéristiques catégorielles en les remplaçant par "**None**".

Après cela, nous divisons les données en caractéristiques (X) et étiquettes (y). En plus, nous avons divisé nos données en ensembles de formation et de test à l'aide de la fonction `train_test_split`). Cela nous permet de séparer une partie de nos données pour l'évaluation de notre modèle. Nous avons spécifié une partie de 70% pour l'ensemble de formation et 30% pour l'ensemble de test. et affichons les formats d'ensemble d'entraînement et du test.

Enfin, nous avons vérifié les dimensions de nos ensembles d'étiquettes `y_train` et `y_test` pour nous assurer que le prétraitement a été effectué correctement. les figures 3.2.2 ci-dessous montrent ce que on a expliqué dans ce paragraphe :

6 Affichage du nombre totale des valeurs manquant dans le dataset :

```
data.isnull().sum().sum()
```

4

7 Affichage de la forme du dataset:

```
data.shape
```

(225745, 79)

```
print("Colonnes avec des valeurs nulles :")
[features for features in data.columns if data[features].isnull().sum()>0]
```

Colonnes avec des valeurs nulles :

`['Flow Bytes/s']`

```
print("Colonnes avec des valeurs infinies :")
[feature for feature in data.columns if data[feature].isin([np.inf, -np.inf]).any()]
```

Colonnes avec des valeurs infinies :

`['Flow Bytes/s', ' Flow Packets/s']`

8 Remplacement des valeurs infinies par NaN dans Dataset et suppression des valeurs NaN

```
data.replace([np.inf, -np.inf], np.nan, inplace=True)
```

```
data.dropna(inplace=True)
```

FIGURE 3.2.2 – Code 1 du pré-traitement du dataset Friday

9 Séparation des colonnes numériques et catégorielles :

```
num_features = data.select_dtypes(include=['int64','float64']).columns  
cat_features=data.select_dtypes(include=['object']).columns
```

10 Encodage des variables catégorielles en valeurs numériques :

```
scale = LabelEncoder()  
for i in cat_features:  
    data[i] = scale.fit_transform(data[i].astype(str))
```

FIGURE 3.2.3 – Code 2 du pré-traitement du dataset Friday

11 Séparation des variables indépendantes (features) et dépendantes (target):

```
X= data.drop('Label', axis=1)  
y=data['Label']  
  
# Générer des données avec seulement deux fonctionnalités informatives  
X, y = make_classification(n_samples=200, n_features=2, n_informative=2, n_redundant=0, n_repeated=0, n_classes=2, random_state=42)  
  
# Diviser les données en ensembles d'entraînement et de test  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

FIGURE 3.2.4 – Code 3 du pré-traitement du dataset Friday

3.2.2 pré-traitement du dataset CICIDS 2017-Thursday

D'abord, après le chargement du dataset CICIDS 2017-Thursday, nous affichons les types de données de chaque colonnes en utilisant **data.dtypes** nous trouvons deux types (int64 , , float64), les dix première ligne en utilisant **data.head(10)**, la description statistique des données avec **data.describe()**, les nomes des colonnes a l'aide de **data.columns** et finalement le nombre totale des valeurs manquantes dans notre dataset avec **data.isnull().sum().sum()**.

Ensuite, nous obtenons la forme du dataset et la stocke dans un dictionnaire que nous convertissons en dataset pour un affichage tabulaire avec **shape_data=pd.DataFrame(shape_)**.

Ensuite, nous identifions les caractéristiques numériques et catégorielles,

puis nous imputons les valeurs manquantes en utilisant **La mediane** et les caractéristiques catégorielles en les remplaçants par "**None**".

Après cela, nous divisons les données en caractéristiques (X) et étiquettes (y). puis, nous effectuons une imputation des valeurs manquantes en utilisant **la moyenne**.

Nous mettons ensuite à l'échelle les données en utilisant **StandardScaler** pour les rendre comparable et transforme les caractéristiques pour qu'elles aient une moyenne de "0" et un écart-type de "1".

En plus, nous avons divisé nos données en ensembles de formation et de test à l'aide de la fonction **train_test_split**). Cela nous permet de séparer une partie de nos données pour l'évaluation de notre modèle. Nous avons spécifié un ratio de 70% pour l'ensemble de formation et 30% pour l'ensemble de test. et affichons les formats d'ensemble d'entraînement et du test.

Enfin, nous avons vérifié les dimensions de nos ensembles d'étiquettes **y_train** et **y_test** pour nous assurer que le prétraitement a été effectué correctement. les figures 3.2.5 ci-dessous montre ce que on a expliqué dans ce paragraphe :

8 Affichage du nombre totale des valeurs manquantes dans le dataset :

```
data.isnull().sum().sum()
```

53

9 Affichage de la forme du dataset (Nombre de lignes et de colonnes) :

```
# Obtenir la forme du DataFrame
shape_data = [
    ["Nombre de lignes", "Nombre de colonnes"],
    [data.shape[0], data.shape[1]]
]

# Convertir en DataFrame pour un affichage tabulaire
shape_data = pd.DataFrame(shape_data)

# Afficher le tableau
print(shape_df)
```

Dimension	Valeur
0 Nombre de lignes	3394
1 Nombre de colonnes	79

10 Séparation des colonnes numériques et catégorielles :

```
num_features = data.select_dtypes(include=['int64', 'float64']).columns
cat_features = data.select_dtypes(include='object').columns
```

FIGURE 3.2.5 – Code 1 du pré-traitement du dataset Thusday

14 Encodage des variables catégorielles en valeurs numériques :

```
scale = LabelEncoder()  
  
for i in cat_features:  
    data[i] = scale.fit_transform(data[i])  
  
data.head(10)
```

15 Séparation des variables indépendantes (features) et dépendantes (target):

```
X = data.drop(['Label'], axis = 1)  
y = data['Label']
```

16 Imputation des valeurs manquantes avec la moyenne :

```
imp = SimpleImputer(missing_values=np.nan, strategy='mean')  
imp = imp.fit(data)  
data = imp.transform(data)  
  
X = data[:, :-1]  
y = data[:, -1]  
  
X  
print(X.shape)  
y  
print(y.shape)  
  
(3394, 78)  
(3394,)
```

17 Mise à l'échelle des données :

```
sc = StandardScaler()  
X = sc.fit_transform(X)  
  
# Générer des données :  
X, y = make_classification(n_samples=100, n_features=78, n_informative=2,  
                           n_classes=2, random_state=42)
```

18 Division des données en ensembles d'entraînement et de test :

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,  
                                                   random_state=42, shuffle=True)
```

FIGURE 3.2.6 – Code 2 du pré-traitement du dataset Thusday

3.3 Entraînement de dataset et évaluation des performances

Nous avons utilisé l'algorithme SVM pour la classification des attaques DDoS et Web avec un noyau linéaire et un noyau gaussien pour le cas non linéaire. Deux modèles de classification sont instanciés : **SVC(kernel='linear')** pour SVM linéaire et **SVC(kernel='rbf')** pour SVM non linéaire.

3.3.1 Classification des attaques DDoS du dataset CICIDS 2017-Friday Avec l'algorithme SVM

Cas de SVM linéaire

Dans le code sur figure3.3.1 ci-dessous nous avons expliqué ce que nous avons fait.

Tout d'abord, les données sont séparées en 70% pour l'ensemble d'entraînement et de 30% pour le test en utilisant la fonction *train_test_split*. Cela assure que le modèle est évalué sur des données qu'il n'a pas observée pendant l'entraînement, ce qui permet d'évaluer sa performance à partir de données nouvelles.

```
# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Entrainer le modèle SVM avec un noyau linéaire sur l'ensemble d'entraînement
model = SVC(kernel='linear')
model.fit(X_train, y_train)
# Effectuer des prédictions sur l'ensemble de test
y_pred = model.predict(X_test)
# Calculer les métriques d'évaluation
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='weighted')
recall = recall_score(y_test, y_pred, average='weighted')
f1 = f1_score(y_test, y_pred, average='weighted')
# Afficher les métriques d'évaluation
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
# Préparer les points de la grille pour la prédiction
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
# Prédire les étiquettes pour chaque point de la grille
grid_points = np.c_[xx.ravel(), yy.ravel()]
Z = model.predict(grid_points)
Z = Z.reshape(xx.shape)
# Afficher le graphique d'entraînement avec les différentes classes
plt.contourf(xx, yy, Z, cmap=plt.cm.coolwarm, alpha=0.8)
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap=plt.cm.coolwarm, edgecolors='k')
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap=plt.cm.coolwarm, marker='x', s=100)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Decision boundary with Training and Test data')
plt.show()
```

FIGURE 3.3.1 – Code d'entraînement du dataset Friday avec SVM linéaire

Par la suite, on initialise et entraîne un modèle SVM avec un noyau linéaire sur l'ensemble d'entraînement en utilisant la méthode *fit*. Les relations entre les caractéristiques et les étiquettes de classe sont étudiées par le modèle en utilisant les caractéristiques (*X_train*) et les étiquettes de classe correspondantes (*y_train*). Après avoir entraîné le modèle, des prédictions sont réalisées sur

l'ensemble de test en utilisant la méthode predict. On compare ces prévisions (y_pred) aux étiquettes de classe réelles (y_test) afin d'évaluer les résultats du modèle.

Puis on a calculé les métriques d'évaluation, telles que la précision, accuracy, recall et le score F1, sont calculées à l'aide des fonctions de scikit_learn correspondantes. cela nous donne le résultat qui est dans le tableau suivant :

Métrique	Valeurs
Accuracy	0.775
Precision	0.77625
Recall	0.775
F1 Score	0.775140712945591

TABLE 3.2 – Résultats des évaluations 1

puis, nous avons calculé la matrice de confusion pour tout le dataset. Afin

	Non Attaque	Attaque-DDos
Non Attaque	32543	1246
Attaque-DDos	2365	31570

TABLE 3.3 – Matrice de confusion 1

de représenter la limite de décision du modèle SVM, on élabore une grille de points en utilisant np.c_, puis on la colore en utilisant plt.contourf. Les points d'entraînement sont identifiés par leurs classes, tandis que les points de test sont indiqués par 'x', avec des étiquettes d'axes et un titre ajoutés pour rendre les choses plus claires. plt.show() montre la manière dont le modèle distingue les classes dans l'espace des caractéristiques. la figure 3.3.2 affiche le graphique final :

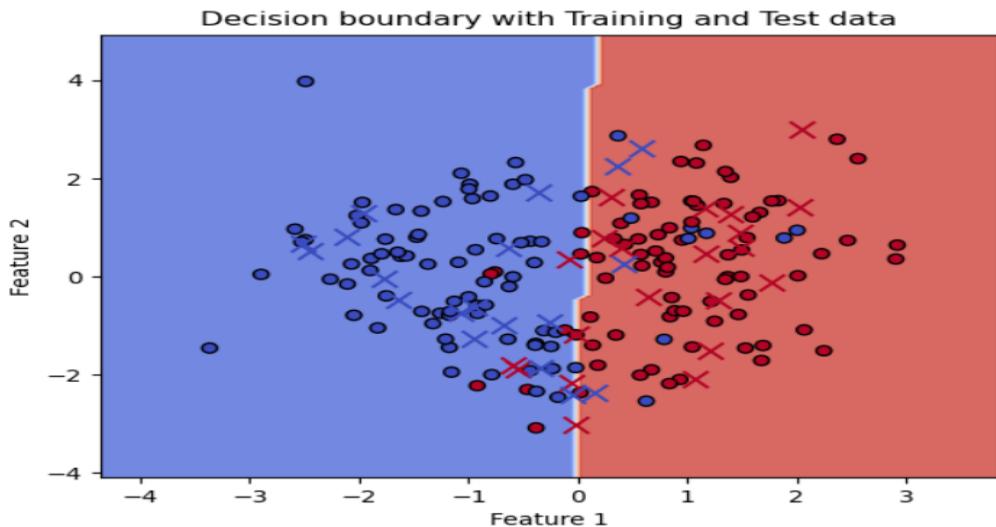


FIGURE 3.3.2 – Classification des attaques DDoS avec SVM linéaire

Cas de SVM non-linéaire

Au cours de cette partie de code, nous constatons une étape essentielle dans la création d'un modèle de classification qui utilise un Support Vector Classifier (SVC) avec un noyau gaussien. Dans un premier temps, nous commençons notre modèle en utilisant la classe SVC de la bibliothèque Scikit-learn. La spécification du paramètre 'kernel' est 'rbf', ce qui implique que nous utilisons un noyau gaussien (ou fonction de base radiale) pour séparer les classes dans l'espace des caractéristiques. Le paramètre 'gamma' est défini sur 'auto', ce qui implique que le modèle déterminera automatiquement la valeur de gamma. Par la suite, nous conduisons notre modèle à l'entraînement en utilisant la méthode fit, en fournissant les informations d'entraînement X_train et les étiquettes correspondantes y_train. Cela ouvre la voie à notre modèle pour explorer les relations entre les caractéristiques et les étiquettes, afin de pouvoir classer de nouvelles données avec précision.

```
# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
# Entraîner le modèle SVC avec un noyau gaussien
model = SVC(kernel='rbf', gamma='auto')
model.fit(X_train, y_train)
# Définir la plage des valeurs x et y pour créer une grille de points
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
# Créer une grille de points avec un pas de 0.1 pour les deux fonctionnalités
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
# Faire des prédictions sur cette grille de points avec seulement deux fonctionnalités
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
# Remodeler les prédictions pour qu'elles correspondent à la grille de points
Z = Z.reshape(xx.shape)
# Créer un nuage de points pour la visualisation
plt.figure(figsize=(13, 8))
# Tracer la séparation de la classe
plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')
# Tracer les points pour les données d'entraînement
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='coolwarm', marker='o', label='Entraînement')
# Tracer les points pour les données de test
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='coolwarm', marker='x', label='Test')
plt.title("Classification avec SVM non linéaire (2 premières caractéristiques)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
# Ajouter une légende pour les attaques (1) et les non-attaques (0)
plt.scatter([], [], color='red', label='Attaque')
plt.scatter([], [], color='blue', label='Non Attaque')
plt.legend()

plt.show()
```

FIGURE 3.3.3 – Code d'entraînement du dataset Friday avec SVM non-linéaire

Puis, nous avons calculé les métriques des évaluations. Voici le résultat présenté sur le tableau ci-dessous.

Métrique	Valeurs
Accuracy	0.81
Precision	0.79
Recall	0.82
F1 Score	0.80

TABLE 3.4 – Résultats des évaluations 2

Puis, nous avons calculé la matrice de confusion pour toute la base de données.

	Non Attaque	Attaque-DDos
Non Attaque	32543	1246
Attaque-DDos	2365	31570

TABLE 3.5 – Matrice de confusion 2

Finalement, Le modèle SVM est visualisé en créant une grille de points en se basant sur les données d'entraînement, prédite par le SVM afin de représenter visuellement la séparation des classes à l'aide de contourf. On distingue visuellement les points d'entraînement et de test grâce à un scatter, avec des titres, des axes et des légendes ajoutés pour éclaircir. Le diagramme définitif, affiché avec plt.show(), met en évidence la manière dont le SVM partitionne de manière efficace les classes dans l'espace des caractéristiques. Voici le graphique 3.3.4 final :

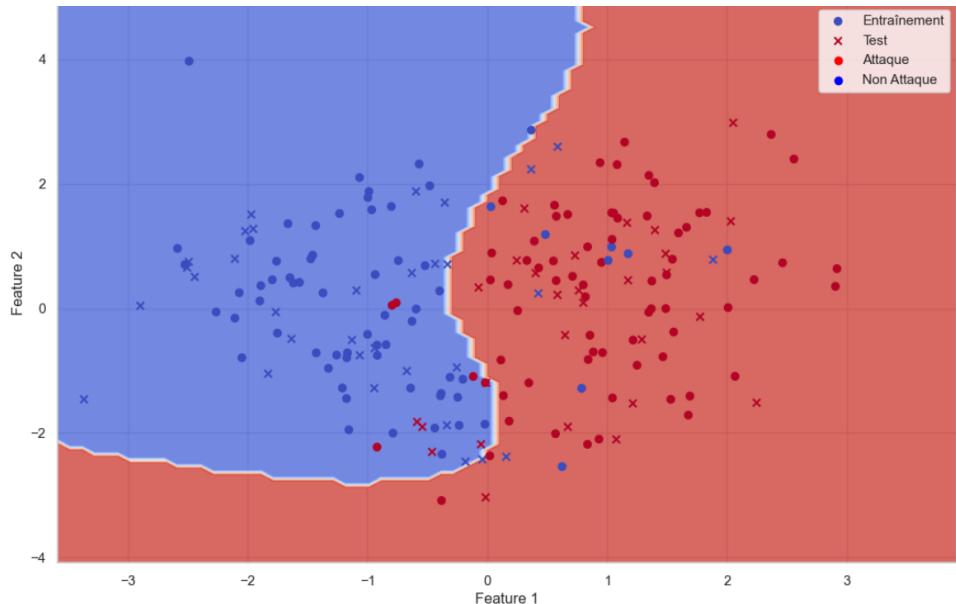


FIGURE 3.3.4 – Classification des attaques DDoS avec SVM non linéaire

3.3.2 Classification des attaques DDoS du dataset CICIDS 2017-Thursday Avec l'algorithme SVM

Cas de SVM linéaire

Dans notre implémentation nous avons pris 200 instances pour les générer, comme le montre la figure 3.3.5.

```
# Générer des données avec seulement deux fonctionnalités informatives
X, y = make_classification(n_samples=200, n_features=2, n_informative=2, n_redundant=0, n_repeated=0, n_classes=2, random_state=42)
# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

FIGURE 3.3.5 – Génération des données pour le SVM linéaire

```
# On entraîne le modèle sur l'ensemble d'entraînement mis à l'échelle :
model.fit(X_train, y_train)
# Définir la plage des valeurs x et y pour créer une grille de points
x_min, x_max = X_train[:, 0].min() - 1, X_train[:, 0].max() + 1
y_min, y_max = X_train[:, 1].min() - 1, X_train[:, 1].max() + 1
# Créer une grille de points avec un pas de 0.1 pour les deux fonctionnalités
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))
# Faire des prédictions sur cette grille de points avec seulement deux fonctionnalités
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
# Remodeler les prédictions pour qu'elles correspondent à la grille de points
Z = Z.reshape(xx.shape)
# Créer un nuage de points pour la visualisation
plt.figure(figsize=(13, 8))
# Tracer la séparation de la classe
plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')
# Tracer les points pour les données d'entraînement
plt.scatter(X_train[:, 0], X_train[:, 1], c=y_train, cmap='coolwarm', marker='o', label='Entraînement')
# Tracer les points pour les données de test
plt.scatter(X_test[:, 0], X_test[:, 1], c=y_test, cmap='coolwarm', marker='x', label='Test')
plt.title("Classification avec SVM linéaire (2 premières caractéristiques)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
# Ajouter une légende pour les attaques (1) et les non-attaques (0)
plt.scatter([], [], color='red', label='Attaque')
plt.scatter([], [], color='blue', label='Non Attaque')
plt.legend()
```

FIGURE 3.3.6 – code d'entraînement du dataset Thusday avec SVM lineaire

D'abord, Nous avons entraîné ce dataset CICIDS 2017-thursday avec l'algorithme SVM linéaire. La figure 3.3.6 montre les étapes suivies.

Puis, nous avons calculé les métriques d'évaluation. Cela nous donne les résultats présentés sur le tableau.

Métrique	Valeurs
Accuracy	0.8166666666666667
Precision	0.7931034482758621
Recall	0.8214285714285714
F1 Score	0.8070175438596492

TABLE 3.6 – Résultats des évaluations 3

Finalement, nous avons calculé la matrice de confusion pour tous les lignes du dataset.

	Non Attaque	Attaque-Web
Non Attaque	24352	1296
Attaque-Web	1122	24340

TABLE 3.7 – Matrice de confusion 3

Après l'entraînement et le test, nous avons affiché finalement le graphique 3.3.7 suivant :

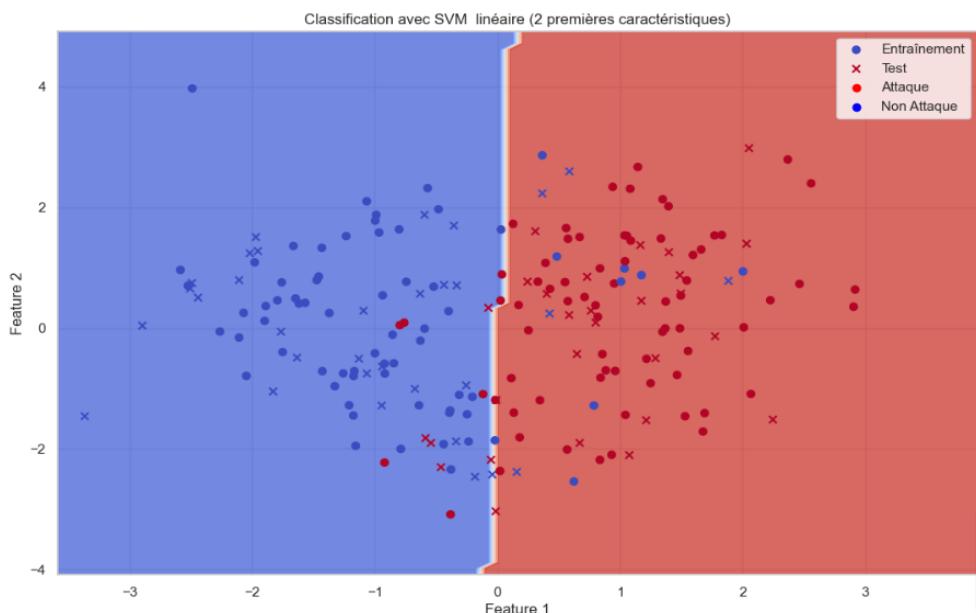


FIGURE 3.3.7 – Classification des attaques Web avec algorithme SVM linéaire

Cas de SVM non-linéaire

Dans notre implémentation nous avons pris 100 instances pour les générer, comme le montre la figure 3.3.8.

```
# Générer des données :  
X, y = make_classification(n_samples=100, n_features=78, n_informative=2,  
                           n_classes=2, random_state=42)
```

FIGURE 3.3.8 – Génération des données pour le SVM non-linéaire

Comme nous avons expliqué dans la figure 3.3.9 de notre entraînement. Nous avons entraîné l'algorithme svm non-lineaire (Noyau de Gaussien) avec le dataset CICIDS 2017-thusday, en utilisant la méthode " SVM\$CLASSIFIER = SVC(kernel='rbf')".

```
# Instanciation du classificateur SVM avec le noyau gaussien :  
svm_classifier = SVC(kernel='rbf') # Utilisation du noyau gaussien pour le SVM  
#non linéaire  
svm_classifier.fit(X_train, y_train)  
  
# Prédire les étiquettes sur l'ensemble de train et de test  
y_train_pred = svm_classifier.predict(X_train)  
y_test_pred = svm_classifier.predict(X_test)
```

FIGURE 3.3.9 – Code d'emtrainement Avec SVM non-lineaire

la figure 3.3.10 explique les étapes que nous avons utilisé pour afficher notre graphique en indiquant l'attaque, non attaque, ensemble d'entraînement et l'ensemble du test.

```

# Définir la plage des valeurs x et y pour créer une grille de points
x_min, x_max = X_train_2d[:, 0].min() - 1, X_train_2d[:, 0].max() + 1
y_min, y_max = X_train_2d[:, 1].min() - 1, X_train_2d[:, 1].max() + 1

# Créer une grille de points avec un pas de 0.1
xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.1),
                     np.arange(y_min, y_max, 0.1))

# Faire des prédictions sur cette grille de points
Z = svm_classifier_2d.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Créer un nuage de points pour la visualisation
plt.figure(figsize=(13, 8))

# Tracer la séparation de la classe
plt.contourf(xx, yy, Z, alpha=0.8, cmap='coolwarm')

# Tracer les points pour les données d'entraînement
plt.scatter(X_train_2d[:, 0], X_train_2d[:, 1], c=y_train, cmap='coolwarm',
            marker='o', label='Entraînement')

# Tracer les points pour les données de test
plt.scatter(X_test_2d[:, 0], X_test_2d[:, 1], c=y_test, cmap='coolwarm',
            marker='x', label='Test')

plt.title("Classification avec SVM non linéaire (2 premières caractéristiques)")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()

# Ajouter une légende pour les attaques (1) et les non-attaques (0)
plt.scatter([], [], color='red', label='Attaque')
plt.scatter([], [], color='blue', label='Non Attaque')
plt.legend()

```

FIGURE 3.3.10 – Code d'affichage du graphique

Ensuite, la résultat de notre d'entraînement se présente dans le graphique 3.3.11.

les points rouges représentent l'ensemble d'entraînement, X représentent l'ensemble du test, les points oranges représentent les attaques et les points bleus représentent les trafic normal.

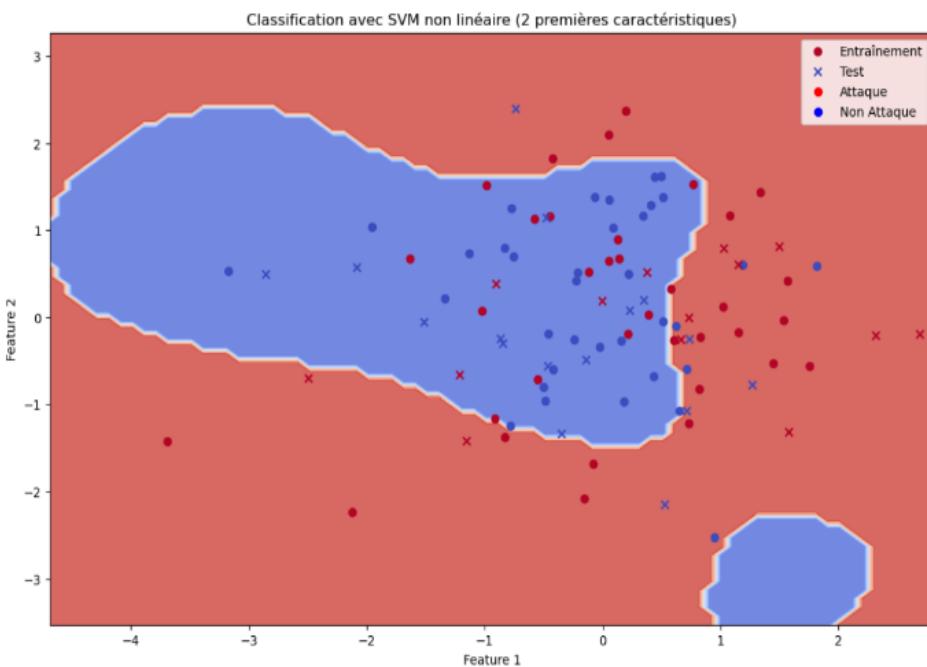


FIGURE 3.3.11 – Classification des attaques Web avec algorithme SVM non linéaire

Puis, nous avons calculé les métriques d'évaluation. Cela nous donne le résultat présenté sur le tableau.

Métrique	Valeurs
Accuracy	0.9
Precision	0.8235294117647058
F1 Score	0.903225806451612

TABLE 3.8 – Résultats des évaluations 4

Finalement, nous avons calculé la matrice de confusion pour tous les lignes de ce dataset.

	Non Attaque	Attaque-Web
Non Attaque	25010	638
Attaque-Web	510	24952

TABLE 3.9 – Matrice de confusion 4

Selon la matrice de confusion présentée dans le tableau 3.9, notre modèle a prouvé de manière correcte 25010 exemples de la classe "Non Attaque" 638 exemples de la classe "Attaque". Par contre, il a commis 510 erreurs en prédisant à tort des exemples de la classe "Non Attaque" comme étant de la classe "Attaque", et 24952

erreurs en prédisant des exemples de la classe "Attaque" comme étant de la classe "Non Attaque".

Conclusion

L'algorithme de classification par Machines à Vecteurs de Support (SVM) a été employé afin de détecter les intrusions en utilisant les données CICIDS 2017. Nous avons utilisé un matériel spécifique dans notre environnement de développement, ainsi que des langages de programmation et des bibliothèques adaptées pour le traitement des données. Dans un premier temps, nous avons importé les bibliothèques requises avant de commencer le pré-traitement des datasets CICIDS 2017, notamment les données des jours "Vendredi" et "Jeudi". Pour nettoyer et préparer les données, nous avons mis en place une série d'étapes de prétraitement, y compris la gestion des valeurs manquantes et la normalisation des caractéristiques. Dans un second temps, nous avons élaboré notre modèle SVM et l'avons entraîné sur les datasets prétraités. Pour évaluer les résultats de notre modèle, nous avons utilisé des métriques standard, telles que la précision, le rappel et le score F1. En particulier, nous avons focalisé nos efforts sur la classification des attaques DDoS dans un dataset "Friday" et attaques web dans un dataset "Thursday", démontrant ainsi la capacité de notre modèle SVM à identifier efficacement ces types d'intrusions.

Conclusion

En conclusion, notre projet de fin d'études a exploré de manière approfondie la sécurité de l'Internet des Objets (IoT), mettant en évidence les difficultés et les possibilités de protection de ces systèmes complexes et interconnectés. Au début, nous avons étudié les principes technologiques essentiels de l'IoT, son modèle de fonctionnement et les domaines où cette technologie est largement utilisée. Les objets connectés ont été clairement définis et leurs caractéristiques et leurs composants essentiels ont été analysés afin de mieux appréhender les vulnérabilités qui leur sont propres.

Ensuite, nous avons examiné les attaques informatiques contre l'IoT, en donnant une définition des diverses formes d'attaques et de leurs conséquences sur la sécurité des réseaux IoT. Cette étude a souligné l'importance essentielle de la mise en place des mesures de sécurité robustes pour prévenir et détecter les intrusions.

Au cours de la deuxième partie du projet de fin d'études, nous avons abordé les notions d'intelligence artificielle (IA) et d'apprentissage automatique, notamment les algorithmes de Machines à Vecteurs de Support (SVM). Nous avons examiné en détail les différents types et bénéfices de l'intelligence artificielle, les différentes catégories d'apprentissage automatique, ainsi que les différents types d'algorithmes employés dans ce domaine. Cette section pose les bases théoriques nécessaires pour comprendre comment SVM peut être utilisé pour la détection d'intrusion. En détaillant les principes des SVM, nous expliquons leur application dans l'identification de comportements anormaux, permettant de détecter des intrusions dans les systèmes informatiques.

Enfin, nous avons effectué la classification des intrusions en utilisant l'algorithme SVM. Nous avons élaboré un modèle de détection en utilisant les datasets CICIDS 2017 en mettant l'accent sur le prétraitement des données.

Bibliographie

- [1] Ouali Alami Mohammed, La conception d'une prise connectée basée sur la technologie d'IoT(Internet of Things), FST Fès, Mémoire de maîtrise, 2016, <https://www.electronique-mixte.fr/>, [Accès le Avril 2024].
- [2] L'Equipe Kaliop, Smart logistics :quand l'IoT révolutionne la logistique des entreprises, 24 Août 2022, <https://www.kaliop.com/fr/smart-logistics-quand-liot-revolutionne-la-logistique-des-entreprises/>, [Accès le Avril 2024].
- [3] Jayna Locke, Qu'est-ce que la technologie des véhicules connectés et quels sont les cas d'utilisation ?, 24 Juin 2020, <https://fr.digi.com/blog/post/what-is-connected-vehicle-technology-and-use-cases>, [Accès le Avril 2024].
- [4] L'Equipe de SAP,Qu'est-ce que l'Industrie 4.0 ?, <https://www.sap.com/france/products/scm/industry-4-0/what-is-industry-4>, [Accès le Avril 2024].
- [5] JOHNSON CONTROLS, Qu'est-ce qu'une maison intelligente et qu'est-ce qui la caractérise ?, 2024, <https://www.hitachiclimat.fr/actualites/maison-intelligente-connectee-climatiseurs>, [Accès le Avril 2024].
- [6] L'Equipe de safetyculture, Smart framing, 15 Janvier 2024, <https://safetyculture.com/fr/themes/smart-farming/>, [Accès le Avril 2024].
- [7] Tebib et Soualah, La Sécurité Des Données Dans L'Internet Des Objets(IoT), Université de Tebessa, Mémoire de Master, 2017.
- [8] Panda Security Mediagcenter, Qu'est-ce qu'une attaque man-in-the-middle (MITM) ? Définition et prévention, 12 Avril 2012, <https://www.pandasecurity.com/fr/mediacenter/attaque-man-in-the-middle/>, [Accès le Avril 2024].
- [9] Amirat Mitra, What are replay attacks ?, 7 Mars 2017, <https://www.thesecuritybuddy.com/vulnerabilities/what-is-replay-attack/>, [Accès le Avril 2024].
- [10] Michelle Greenlee, What Is a Botnet Attack ? A Guide for Security Professionals, 27 Septembre 2021, <https://securityintelligence.com/articles/what-is-botnet-attack/>, [Accès le Avril 2024].
- [11] Union International des Télécommunications, Présentation générale de la cybersécurité, Avril 2008, https://www.itu.int/rec/dologin_pub.asp?lang=eid=T-REC-X.1205-200804-I!!PDF-Ftype=items, [Accès le Avril 2024].

- [12] proofpoint, Qu'est-ce qu'une attaque par bruteforce (Force brute) ?, <https://www.proofpoint.com/fr/threat-reference/brute-force-attack>, [Accès le Avril 2024].
- [13] Coursera Staff, What Is a DDoS Attack ?, 29 Nouvembre 2023, <https://www.coursera.org/articles/ddos-attack> ?, [Accès le Avril 2024].
- [14] EY, Définition autour des objets connectés, 6 Mars 2017, <https://www.smartgrids-cre.fr/>, [Accès le Avril 2024].
- [15] Qu'est-ce que l'intelligence artificielle ?, <https://azure.microsoft.com>, [Accès le Mai 2024].
- [16] Qu'est-ce que l'intelligence artificielle (IA) ?, <https://www.ibm.com/fr-fr/topics/artificial-intelligence>, [Accès le Mai 2024].
- [17] Présentation de Machine Learning, <https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-is-machine-learning-platform/>, [Accès le Mai 2024].
- [18] What Is Machine Learning ?, <https://www.oracle.com/artificial-intelligence/machine-learning/what-is-machine-learning/>, [Accès le Mai 2024].
- [19] Algorithmes d'apprentissage automatique, <https://azure.microsoft.com/fr-fr/resources/cloud-computing-dictionary/what-are-machine-learning-algorithms/>, [Accès le Mai 2024].
- [20] Oracle, Machine Learning for SQL-About Classification,September 2022, <https://docs.oracle.com/en/database/oracle/machine-learning/>, [Accès le Mai 2024].
- [21] Composant Régression linéaire, 1 Juin 2023, <https://learn.microsoft.com/fr-fr/azure/machine-learning/component-reference/linear-regression?view=azureml-api-2>, [Accès le Mai 2024].
- [22] Algorithme MLR (Microsoft Logistic Regression), 31 Octobre 2023, <https://learn.microsoft.com/fr-fr/analysis-services/data-mining/microsoft-logistic-regression-algorithm?view=asallproducts-allversions>, [Accès le Mai 2024].
- [23] Support Vector Machine Algorithm, <https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm>, [Accès le Mai 2024].
- [24] Essam Wisam, Easily Implement Multiclass SVM From Scratch in Python, 4 novembre 2023, <https://towardsdatascience.com/implement-multiclass-svm-from-scratch-in-python>, [Accès le Mai 2024].
- [25] Major Kernel Functions in Support Vector Machine <https://www.javatpoint.com/major-kernel-functions-in-support-vector-machine>, [Accès le Mai 2024].
- [26] OCI, Êtes-vous prêt pour la prochaine Cyberattaque ?, <https://www.oracle.com/fr/security/cyberattaque/>, 2024, [Accès le Mai 2024].
- [27] Tebib Samia et Soualah Abdelghani, La Sécurité Des Données Dans L'Internet Des Objets (IoT) , FST Fès, Mémoire de master, Mai 2017, [Accès le Mai 2024]

- [28] Hamouda Djallel, “Un système de détection d’intrusion pour la cybersécurité”, Mémoire de fin de cycle, Université de 8 Mai 1945 – Guelma -, 2020, [Accès le Mai 2024].
- [29] Thabet Ilyas, “Recalage des images médicales par apprentissage profond”, Mémoire de fin de cycle, Université Larbi Tebessi, 2021, [Accès le Mai 2024].
- [30] Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB), Intrusion detection evaluation dataset (CIC-IDS2017), <https://www.unb.ca/cic/datasets/ids-2017.html>, [Accès le Mai 2024].
- [31] Qu'est-ce que l'intelligence artificielle (IA) ?, <https://www.ibm.com/topics/artificial-intelligenceWhat+is+AI>, [Accès le Mai 2024].
- [32] KERDOUD Fateh, “Un système de compression d’images basé sur les réseaux de neurones profonds”, Mémoire de master, Université de Tébessa, 2021, [Accès le Mai 2024].
- [33] Aissam Outchakocht, “Mise en place d’environnement sécurisé dans l’internet des objets”, Mémoire de master, Faculté des sciences, 2016, [Accès le Mai 2024].
- [34] ABBACHE Imene et AGUEMAL Karima, “Implémentation d’un IDS basé sur des approches de machine et deep learning pour la détection d’attaques de botnets dans l’iot”, Mémoire de fin de cycle, Université A. MIRA-BEJAIA, 2023.