République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Akli Mohand Oulhadj de Bouira

Faculté des Sciences et des Sciences Appliquées

Département d'Informatique

Mémoire de Master

en Informatique

Spécialité : *ISIL & GSI*

Thème

Modèle de prédiction et de détection de Malwares informatiques

---

Encadré par

— Dr.AID Aicha

Réalisé par

— BAHRI Rym

— SALMI Nadia

Devant le jury composé de :

BENNOUAR Djamel	*Pr*	*UAMOB*	*Président*
BOUDJELABA Hakim	*DR*	*UAMOB*	*Examinateur*
YAHIAOUI Kais	*DR*	*UAMOB*	*Examinateur*

2019/2020

Remerciements

En tout premier lieu, nous remercions le bon Dieu, tout puissant, de nous avoir donné la force pour survivre, ainsi que l'audace pour dépasser toutes les difficultés.

Nous adressons nos grands remerciements à notre promotrice AID Aicha d'avoir accepté de diriger ce travail , pour tous ses conseils ,sa patience et son orientation tout au long de l'accomplissement de ce travail sans lesquelles ce travail n'aurait pas vu le jour.

Nous adressons aussi nos remerciements au président et aux membres du jury d'avoir accepté le jugement de notre travaille.

Que tous ceux qui sont contribué de_prés ou de-loin à la réalisation de ce travaille trouvent ici nos sincères remerciements.

Dédicaces

Avant toutes choses je tiens à remercier toutes personnes qui a contribuer de prêt ou de loin à la réalisation de ce travail.

Je dédie ce travail à :

Mes chers parents que nul dédicace ne peut exprimer mes sincères sentiments, je les remercie pour leur patience illimitée, leur encouragement continu ainsi que leur aide précieuse, en témoignage de mon profond amour et respect pour leur grands sacrifices.

Mes chers frères et sœurs.

A mon très cher ami Nacer merci pour tes orientations précieuses et tes conseils inestimables.

A ma chère binôme : Nadia tu es été très collaboratrice et c'est un honneur pour moi d'avoir partagé ce travail avec toi.

Mes chers ami(e)s qui sans leur encouragements ce travail n'aura jamais vu le jour.

Et en fin à toute ma famille et à tous ceux que j'aime.

BAHRI Rym.

Dédicaces

Je dédie ce modeste travail à ceux qui ont sacrifié, corps et âmes pour m'offrir un milieu favorable et qui n'ont pas cessé de m'encourager et de me soutenir pour mener à bien ce projet.

A mes très chers parents, qui m'ont donné la vie et m'ont soutenu depuis toujours. Que Dieu les protège.

A mes frères et sœurs Wissam, Malika, Mohamed et Walid ainsi que toute ma famille pour leur confiance et leur amour.

A mes très cher(e)s ami(e)s, hayette selmane , hayet kither .

A l'homme le plus proche de mon cœur Mohammed pour son assistance quotidienne(confiance) et ses encouragements.

En fin, à ma chère collègue Rym.

SALMI Nadia.

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AdaBoost	Adaptive Boosting
API	Application Programming Interface
ATP	Advanced Threat Protection
AUC	Area Under the Curve
AVG	Anti-Virus Guard (computer security software)
BSD	Berkeley Software Distribution
CART	Classification And Regression Trees
CN-G	Common N-Gram
FBI	Federal Bureau of Investigation
FPR	False Positive Rate
GBDT	Gradient Boosting Decision Tree
GBM	Gradient Boosting Machine
Hachage MD5	Hachage Message Digest 5
IDF	inverse document frequency
IM Trojan	Instant Messenger trojan
KNN	k-nearest neighbors
MSN	MicroSoft Network
NAN	Not A Number

NHS	National Health Service
PE	Portable Executable
Reg	regression
ROC	Receiver Operating Characteristic
SHA1	Secure Hash Algorithm 1
SVM	Support Vector Machine
TF	term frequency
VBS	Visual Basic Script
XGBoost	eXtreme Gradient Boosting

Introduction générale

Actuellement, les systèmes informatiques interviennent dans tous les domaines (industrie, éducation , la médecine...etc) , et même dans le quotidien des particuliers.

Avec l'étendu de ces systèmes la sécurité informatique est devenue nécessaire et prioritaire pour maintenir le bon fonctionnement et l'exploitation des systèmes informatiques. Le domaine de sécurité informatique traite les différentes techniques et les moyens nécessaire à mettre en place visant à empêché les utilisations non autorisés , mauvaise usage et les modifications ou détournement d'un système.

L'usage accru d'internet et l'évolution contenu de la technologie ont contribué à l'augmentation du nombre de cybers-attaques , compensée par l'augmentation du nombre de vulnérabilités de sécurité qui offrent des opportunités pour la cybercriminalité et ouvrent un large champ à la programmation de logiciels malveillants qui évolue constamment .

Malware est l'acronyme pour malicious software également appelés logiciels malveillant qui désigne tous les programmes crées dans le but de nuire à un système informatique .Ils sont conçus pour perturber, désactiver ou prendre le contrôle d'un système informatique en utilisant des techniques intelligentes pour tromper les utilisateurs.
Il existe diverses familles de ces logiciels malveillant tel que : virus , vers , chevaux de troie , Rootkits ...etc. souvent, leurs objectifs est de voler et de détruire des informations ainsi que des systèmes corrompus.

La protection contre les attaques , les dommages ou les accès non autorisés a besoin d'un antivirus ou bien un autre outil pour la détection et la protection contre ses malwares par exemple Windows Difender. La détection des logiciels malveillants est un problème de classification binaire qui permet de savoir si un programme donné est un malware ou pas.Il existe deux types d'analyse qui sont : l'analyse statique et dynamique.

L'objectif de notre travail consiste à proposer une solution qui permettra de détecter si une machine windows est infectée par diverses familles de logiciels malveillants en fonction des différentes propriétés de cette machine . Nous avons un but de créer notre propre modèle (**XGBOOST classifier**) qui nous aide à résoudre le problème posé . Les techniques du Machine Learning permettent de résoudre les problèmes de domaine de sécurité informatique avec ses diverses algorithmes .

Afin de mener à bien notre travail, nous l'avons fragmenté en quatre chapitres comme suit :

- Dans le premier chapitre intitulé **" Détection et prédiction de malwares"**, nous avons présenté quelques notion de base sur la sécurité informatique ,en mettant l'accent sur les malwares puis nous définissons les méthodes de détections des malwares à la fin nous citons quelque célèbres attaques.
- Le deuxième chapitre nommé **"Machine learning"** permettra d'avoir une vue globale sur le domaine du machine learning afin de définir ses méthodes.Comme nous l'avons mentionné les méthodes ensemblistes.
- Le troisième chapitre détaille notre approche proposée. Afin d'expliquer les étapes nécessaires pour construire notre classiffieur xgboost puis nous avons amélioré notre modèle avec la technique traditionnel de tuning **(GridSearch)**.
- Le quatrième chapitre : cet ultime chapitre est consacré pour la présentation des différents outils de programmation utilisé et les résultats expérimentaux obtenus et discussion.

Nous terminons par une conclusion générale et quelques perspectives sur tout le travail réalisé.

Chapitre 1

Détection et prédiction de malwares

1.1 Introduction

Les systèmes informatiques occupent une place prédominante dans les entreprises, les administrations et dans le quotidien des particuliers, c'est pour ça la sécurité informatique est devenue primordiale, et d'une priorité absolue pour la bonne exploitation de ces systèmes. Et pour assurer cette sécurité il faut connaitre tous les obstacles que nous pouvons rencontrons.

Parmi ces obstacles nous avons les logiciels malveillants(Malwares), les attaques informatiques qui représentent une réelle menace pour la sécurité des systèmes informatiques.

Dans cette partie nous nous sommes intéressées au Malware qui constitue une menace à croissance rapide dans le monde informatique moderne. La production des logiciels malveillants est devenue une industrie de plusieurs milliards de dollars.

Dans ce chapitre, nous commençons par présenter les différentes propriétés relatives à la sécurité informatique, une classification des malwares informatiques selon différents aspects, ensuite nous citerons quelques célèbres attaques existant dans le monde et enfin, nous abordons les différents moyens et techniques de protection contre les malwares informatiques.

1.2 Notion de base

1.2.1 Sécurité d'un système (informatique ou d'information)

C'est un ensemble de moyens techniques, organisationnels, juridiques et humains nécessaires et mis en place pour réduire la vulnérabilité d'un système contre les menaces accidentels ou intentionnels afin d'assurer les services de sécurité [1].

1.2.2 Services de sécurité

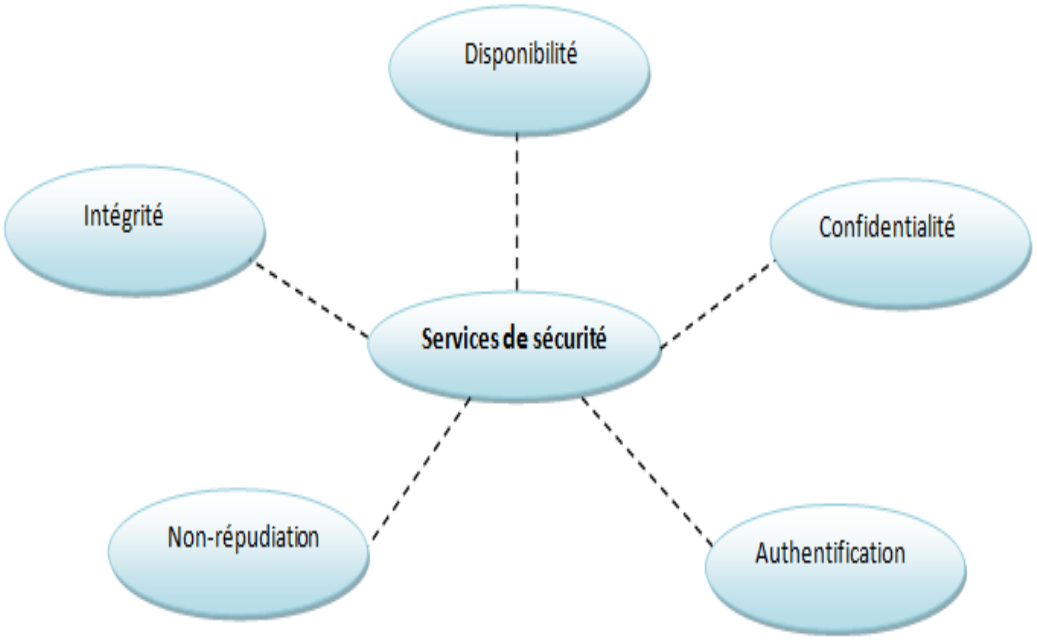

FIGURE 1.1 – Services principaux de la sécurité informatique [2].

— **Intergrité :** l'intégrité est la propriété qu'une information ne soit pas altérée. Cela signifie que le système informatique doit empêcher les modifications par des utilisateurs non autorisés ou une modification incorrecte par des utilisateurs autorisés [2].

- **Disponibilité :** la disponibilité est la propriété qu'une information soit accessible lorsqu'un utilisateur autorisé en a besoin. C'est à dire maintenir le bon fonctionnement du système [2].
- **Confidentialité :** la confidentialité est la propriété qui assure le non divulgation

des données échangées, Cela signifie que le système informatique doit empêcher les utilisateurs de lire une information confidentielle s'ils n'y sont pas autorisés[2].

- **Non-répudiation :** c'est la propriété qui permet de garantir qu'une transaction ne peut être niée[1].
- **Authentification :** : consiste à assurer l'identité d'un utilisateur. Un contrôle d'accès peut permettre l'accès à des ressources uniquement aux personnes autorisées[1].

1.3 Qu'est-ce qu'un malware ?

Malware est l'abréviation de Malicious Software, qui signifie une séquence d'instructions qui exécutent une activité malveillante sur un ordinateur. L'histoire des programmes malveillants a commencé avec «Computer Virus» ce terme introduit pour la première fois par Cohen [1] en 1983 [3].

Un malware ou un logiciel malveillant désigne tous les programmes ou les codes qui sont développés dans le but de nuire à un système informatique .

Aujourd'hui, les logiciels malveillants incluent les virus, les vers, les chevaux de troie,rootkit..etc, et tout autre programme présentant un comportement malveillant.

1.3.1 Types de malwares

Pour mieux comprendre les méthodes et la logique des malwares, il est nécessaire de les classer selon leurs fonctionnements et leurs objectifs. Certains types de logiciels malveillants populaires sont indiqués ci-dessous :

1.3.1.1 Virus

C'est un code reproducteur qui est élaboré pour se répliquer puis se propager en se cachant à l'intérieur d'un programme ou un fichier informatique[4]. Le fichier ou le programme infecté par le virus est appelé hôte. Lorsque cet hôte est exécuté, le virus est également exécuté [13] .

1. Cohen : est un informaticien américain et surtout connu comme l'inventeur des techniques de défense contre les virus informatiques. Il a donné la définition de "virus informatique"

Les virus peuvent [4] :

- Désactiver les systèmes de défenses.
- Voler les données des utilisateurs et dans certains cas ils peuvent même détruire le secteur de démarrage utiliser par les machines pour charger le système d'exploitation et les autres programmes nécessaires.

Les phases d'existence des virus

Il existe trois phases d'existence :

- **Infection :** le virus infecte le système cible. L'infection des fichiers par un virus se fait principalement à l'aide de cinq techniques qui sont : le recouvrement, l'écrasement, l'entrelacement, l'accompagnement de code, et par modification du code source[5].
- **Contamination :** il se duplique et infecte d'autres cibles sans perturber le fonctionnement de système[4].
- **Destruction :** il entre en activité et produit les effets pour lesquels il a été conçu [4].

Les familles des Virus

Il existe 5 familles de Virus [4] :

1. **virus de boot (secteur d'amorçage) :** il remplace ou s'implante lui-même dans le secteur de boot (une partie du disque utilisée lors du démarrage de la machine). Ce type de virus peut empêcher la machine de démarrer.
2. **Virus exécutable :** ils infectent un programme pour être lancés en même temps que lui.
3. **Virus macro :** ils infectent les documents créés par les logiciels de la suite Microsoft Office, sous la forme de macro commandes .
4. **Les virus Win32 :** il s'agit d'un virus exécutable au format 32 bits de Windows. Ce type de virus est le plus dangereux car ils peuvent prendre le contrôle de l'ordinateur .

5. **Les VBS :** les virus écrits en Visual Basic Script parviennent à se reproduire en se recopiant dans les répertoires de démarrage automatique, en modifiant la base de registre ou en se propageant par les e-mails .

1.3.1.2 Ver (virus réseaux)

Le ver informatique (Worms) sont des programmes malveillants indépendants, qui une fois installés dans une machine, utilisent un réseau pour envoyer des répliques d'eux même à d'autres machines . Les vers peuvent : perturber le trafic dans un réseau, supprimer des fichiers, envoyer des documents par courriel, ou bien installer d'autres codes malicieux tels que des Backdoors sur les machines infectée. les ver vont continuer de se répliquer jusqu'à ce qu'un mécanisme arrête le processus [5].

Les catégories des vers :

Catégories	**Description**
Vers de messagerie	Se propagent via les courriels électroniques surtout avec les pièces jointes.
Vers d'internet	Se propagent directement via l'internet en abusant l'accès aux faiblesses du système ouvert.
Vers de réseau	Répartis sur des partages de réseau ouverts et non protégés.

TABLE 1.1 – Catégories des vers [6].

1.3.1.3 Cheval de troie (Trojan Horse, abrévié Trojan)

C'est un virus qui est présenté comme un programme légitime. Lorsque le programme est lancé, les chevaux de troie peuvent offrir au hacker la possibilité d'effectuer des différentes taches à distance dans la machine infectée tel que : la suppression des fichiers, le formatage de disque dur, voler les mots de passe ou encore envoyer des informations confidentielles à son créateur via l'internet. Ce type est très utilisé par l'ingénierie sociale et les hackers [4] .

Les catégories des chevaux de troie

Catégories	**Description**
Proxy Trojan	Conçu pour être utiliser par un serveur proxy d'ordinateur cible qui peut se connecter pour effectuer une multitude d'opérations de manière anonyme.
Password Stealer Trojan	Conçu pour voler les mots de passe des systèmes ciblés. Ce trojan déposera très souvent le composant keylogging dans le périphérique infecté.
IM Trojan	Conçu pour voler des informations de compte ou des données via des programmes de messagerie instantanée tels que SKype ,MSN .
Dropper Trojan	Conçu pour être installer d'autres logiciels malveillants sur les systèmes cibles .Il est généralement utilisé au début d'une attaque de logiciel malveillant.
Game Thief Trojan	Conçu pour voler des informations via un compte du jeu en ligne.
Banker Trojan	Conçu pour voler des informations bancaires en ligne qui permettent aux pirates d'accéder aux informations de compte bancaire ou de carte de crédit.

TABLE 1.2 – Catégories des chevaux de troie [6].

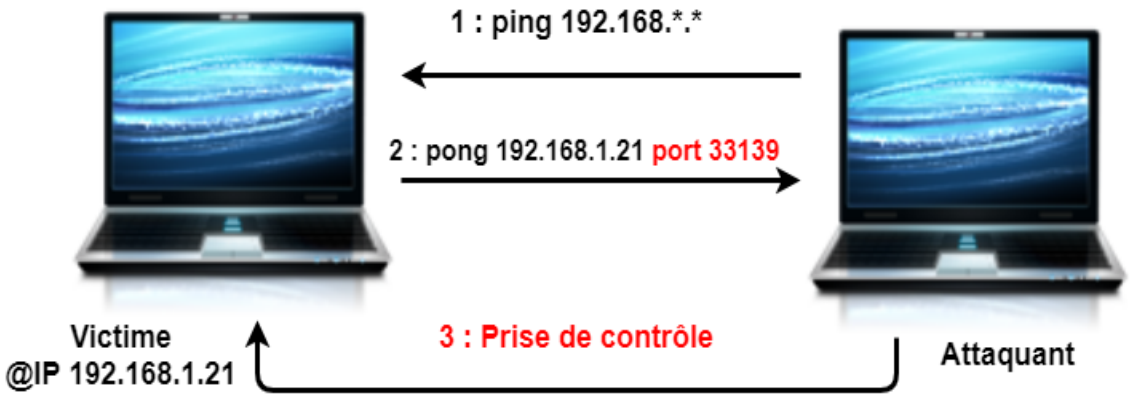

FIGURE 1.2 – Le fonctionnement du cheval de troie [7].

1.3.1.4 Adware

L'adware est un virus qui affiche des publicités sous forme de fenêtres pop-up [4]. En plus d'être une nuisance en raison du nombre d'annonces pop-up qui apparaissent, qui causent une perturbation à l'utilisateur.

1.3.1.5 Rootkit

« Root » désigne chez Unix l'Administrateur, et le mot « kit » signifie « Équipement» ou « Caisse à outils »et le mot composé Rootkit désigne donc un ensemble de logiciels qui permettent aux pirates informatiques d'accéder aux données des utilisateurs afin de les voler sans qu'ils ne soient détectés.
Il existe deux types de rootkit[8] :

- **Les rootkits en mode utilisateur :** ils fonctionnent au sein du système d'exploitation d'un ordinateur.
- **Les rootkits en mode noyau :** ils fonctionnent au niveau le plus profond du système d'exploitation d'ordinateur et donnent au pirate une série de privilèges très puissants.

1.3.1.6 Backdoor (porte dérobée)

Une porte dérobée (Backdoor) peut être définir comme étant un programme qui permet à des attaquants de contourner les mesures de sécurité d'un système, afin d'y obtenir l'accès au système [9].L'installation d'une porte dérobée requiert un accès à la machine cible. De ce fait, l'installation doit se faire soit manuellement en transmettant le malware à l'utilisateur (ex. par courriel) et le convaincre d'installer le programme (méthode basée sur l'ingénierie sociale), soit automatiquement en utilisant un autre code malicieux tel un ver, ou un cheval de troie. Une fois installée, la porte dérobée va donner un accès à l'attaquant. Cela se fait généralement par l'ouverture d'un port réseau et en acceptant des commandes à provenance de la machine de l'attaquant généralement via le protocole http distant via Internet. L'attaquant peut ensuite effectuer différentes tâches sur la machine infectée telles que l'obtention de privilèges administrateur et le contrôle à distance via l'exécution de commandes [5].

1.3.1.7 Keylogger

(littéralement « enregistreurs de touches »), sont des logiciels qui lorsqu'ils sont installés sur le poste de l'utilisateur, permettent d'enregistrer les frappes de claviers saisies par l'utilisateur. Un keylogger peut être installé par un administrateur dans une entreprise afin de contrôler les activités des employés, comme il peut être installé par un Hacker afin d'obtenir des informations sur ses victimes[5].
La figure suivante illustre le mécanisme de fonctionnement d'un Keylogger .

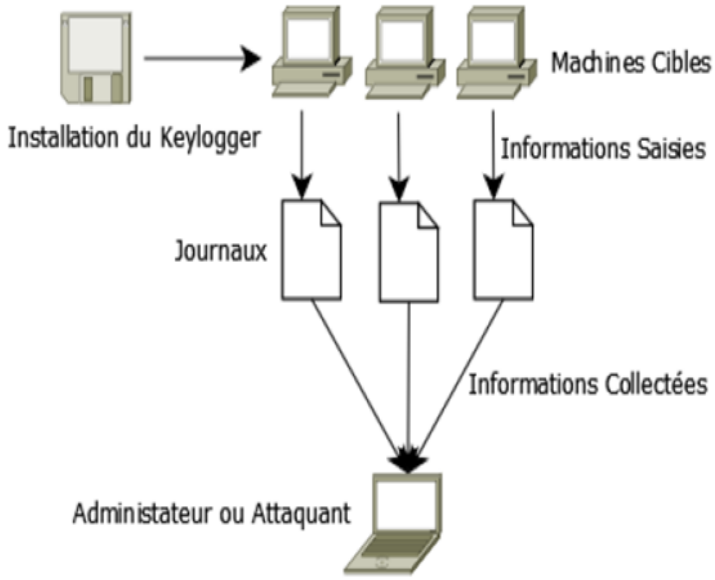

FIGURE 1.3 – Fonctionnement de keylogger [5].

1.3.1.8 Ransomware(Rançongiciels)

définir tout type de malware conçu dans le but d'exiger une certaine somme d'argent en contrepartie de la restitution d'une information ou fonctionnalité dérobée. Ce type de malware vise à crypter toutes les données de la machine et à demander à une victime de transférer de l'argent pour obtenir la clé de décryptage [5].

1.3.1.9 Remote Administration Tools (RAT)

Ce type de logiciel malveillant permet à un attaquant d'accéder au système et d'apporter des modifications possibles comme s'il y était physiquement accessible[10].

1.3.1.10 BotNet

BotNet est l'abréviation de robot network. Le terme robot, ou bot, est un terme générique pour les programmes automatisés qui exécutent des tâches sans intervention de l'utilisateur. Les botnets sont les menaces les plus importantes qui menace l'internet aujourd'hui. Un botnet est un réseau de machines compromises qui peuvent être coordonnées à distance par un attaquant pour remplir une directive malveillante. Les botnets ont de nombreux autres noms parmi ce sont : l'armée bot, le troupeau bot, la horde de zombies,et réseau de zombies[11].

1.3.1.11 Riskware

Riskware c'est le nom donné aux programmes légitimes qui peuvent causer des dommages s'ils sont exploités par des utilisateurs malveillants afin de supprimer, bloquer, modifier ou copier des données et perturber les performances des ordinateurs ou des réseaux [12].

1.3.1.12 Spyware

c'est un type de virus qui s'installe dans un système dans le but de collecter et transférer des informations de l'appareil où il s'est installé [4].

1.3.2 Techniques d'analyse et de détection des malwares

Le processus de détection d'un malware nécessite le passage par une phase d'analyse du code, celle-ci va permettre d'en extraire différents attributs, qui permettrons la classification du fichier. On distingue deux types d'analyses, qui sont l'analyse statique et l'analyse dynamique [5].

1.3.2.1 Analyse statique

L'analyse statique des logiciels malveillants permet d'examiner tout échantillon de logiciel malveillant donné sans réellement exécuter le fichier ou le code . Cette technique nous donnons une idée approximative de la façon dont elle affectera l'environnement lors de l'exécution sans être réellement exécutée[13].

L'avantage principal de l'analyse statique est la capacité de découvrir tous les scénarios comportementaux possibles,et que la recherche du code permet au chercheur de voir tous les moyens d'exécution de logiciels malveillants. De plus, ce type d'analyse est plus sûr que dynamique, car le fichier n'est pas exécuté et il ne peut pas entraîner de mauvaises conséquences pour le système. L'inconvénient majeur d'analyse statique c'est le temps, pour ça, il n'est généralement pas utilisé dans des environnements dynamiques du monde réel, tels que les systèmes antivirus, mais il est souvent utilisé à des fins de recherche, par exemple : lors du développement de signatures pour les logiciels malveillants.
L'analyse statique peut comprendre diverses techniques [10] :

1. Inspection du format de fichier

Les métadonnées de fichier peuvent fournir des renseignements utiles. Par exemple, Windows PE (exécutable portable) fichiers peuvent fournir beaucoup d'informations sur le temps de compilation, les fonctions importées et exportées, etc.

2. Extraction de la chaîne de caractères

Il s'agit de l' examination de la sortie du logiciel (messages d'état ou d'erreur) et l'inférence d'information sur l'opération du programme malveillant.

3. Empreintes digitales

Cela comprend le calcul du hachage cryptographique, la recherche des artefacts environnementaux, comme le nom d'utilisateur codé, le nom de fichier, les chaînes de registre.

4. Analyse d'antivirus

Si le fichier inspecté est un malware bien connu, très probablement tous les scanners anti-virus seront en mesure de le détecter. Bien que cela puisse sembler non pertinent, cette méthode de détection est souvent utilisée par les fournisseurs de matériel audiovisuel ou les bacs à sable(SandBox) pour confirmer leurs résultats.

5. Démontage

Il s'agit d'inverser le code machine au langage d'assemblage et d'en déduire la logique et les intentions du logiciel. C'est la méthode d'analyse statique la plus courante et la plus fiable.

1.3.2.2 Analyse dynamique

Ce type d'analyse est contrairement à l'analyse statique, ici le comportement du fichier est surveillé pendant qu'il est en cours d'exécution et les propriétés du fichier sont déduites de cette information. Avec cette technique nous observons toutes les fonctionnalités du malware et son effet sur l'environnement lors de son exécution. Habituellement, le fichier est exécuté dans un environnement virtuel, par exemple dans le bac à sable (sandbox)[10].

Sandbox est un mécanisme de sécurité permettant d'exécuter des programmes non approuvés dans un environnement sans craindre de nuire aux systèmes réels[13]. Avec cette analyse il est possible de trouver tous les attributs comportementaux, tels que les fichiers ouverts, les mutexes créés, etc , et aussi il plus rapide que l'analyse statique[10].

Néanmoins, cette approche a deux inconvénients. D'une part, une seule trace n'est pas représentative du code et une analyse négative ne garantit en rien que le code soit sain. D'autre part, en n'analysant que le comportement passé, cette approche est souvent insuffisante pour endiguer à temps une activité malicieuse[14].

1.3.3 Méthodes de détection des malware

Les méthodes de détections des logicielles malveillantes sont fondamentalement classées en différentes catégories selon des différents points de vue. Les méthodes sont illustrées à la figure **1.4** :

1.3.3.1 Méthodes basées signature

Sont largement utilisées par les programmes antivirus commerciaux. Ce genre de méthodes repose sur la représentation de chaque malware en utilisant une signature. Cette signature est une séquence d'octet qui est unique pour chaque fichier « malware », quelque chose comme une empreinte digitale d'un exécutable par exemple : Hachage MD5 ou SHA1, chaînes statiques, métadonnées de fichier [10][15].

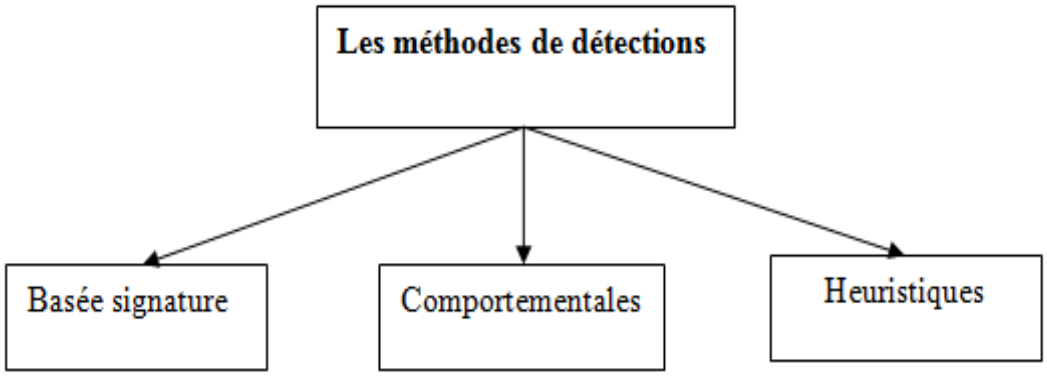

FIGURE 1.4 – Méthodes de détection de malware[15].

À titre d'exemple, le malware Chernobyl/CIH [2] possède la signature suivante : E8000000005B 8D4B42515050 0F014C24FE5B 83C31CFA8B2B[16]. Après leurs extractions, les signatures sont ensuite stockées dans ce qu'on appelle une base virale, et à chaque fois qu'un fichier est analysé, sa signature est générée et est comparées à toutes les autres présentes dans cette base.

Les méthodes basées sur les signatures utilisent les modèles extraits du divers malwares pour les identifier et sont plus efficaces et plus rapide que toute autre méthode. L'inconvénient de ces méthodes c'est que ces méthodes sont incapables de détecter les variantes logiciels malveillants inconnues et nécessitent beaucoup de temps et d'argent pour extraire les signatures uniques et l'incapacité de faire face contre les malwares qui mutent leurs codes à chaque infection comme polymorphe et métamorphique[15].

De toute évidence, ces logiciels malveillants ne peuvent pas être détectés à l'aide des techniques qui base uniquement sur les signatures. De plus, les nouveaux types de logiciels malveillants ne peuvent pas être détectés à l'aide de signatures tant que les signatures n'ont pas été crées. Par conséquent, vendors (les fournisseurs audiovisuelles) ont dû trouver un autre moyen de détection basé sur le comportement également appelé analyse basée sur l'heuristique [10].

2. Chernobyl qui Connu aussi sous le nom de Tchernobyl est un virus informatique créé pour infecter les systèmes d'exploitation Microsoft Windows. CIH doit son nom initial à son inventeur taïwanais Cheng Ing-Hau. Il a été détecté pour la première fois en juin 1998.

1.3.3.2 Les méthodes heuristiques

Tentent de trouver des caractéristiques comportementales et/ou structurelles relatives aux fichiers malicieux et qui vont permettre leur distinction des fichiers bénins[9]. Dans cette approche, une valeur est attribué à chaque fonctionnalité similaire à un logiciel malveillant. Voici quelques-unes des fonctionnalités des logiciels malveillants connus [5] :

- Enregistrement des coups de touches.
- L'écriture dans des fichiers exécutables.
- Suppression de nouvelles clés de registre à des endroits particuliers du registre Windows.
- Suppression de fichiers sur le disque dur.
- Activité réseau suspecte.

Dans le but d'automatiser ce type de méthodes, les experts en sécurité informatique ont utilisé des techniques d'apprentissage automatique (machine Learning) et plus précisément l'apprentissage supervisé afin de construire un modèle de classiffieur permettant de distinguer les malwares des fichiers bénins[15].

1.3.3.3 Méthode comportementale

Cette technique de détection des logiciels malveillants basées sur le comportement d'un programme pour déterminer s'il est malveillant ou non . Étant donné que les techniques basées sur le comportement observent ce que fait un fichier exécutable, elles ne sont pas sensibles aux défauts des fichiers basés sur la signature.

Un détecteur basé sur le comportement se compose essentiellement des composants suivants [17] :

- **Collecteur de données :** la collecte d'informations se fait d'une manière statique ou dynamique.
- **Interprète :** consiste à analyser les données collectées durant la phase précédente afin d'en extraire celles jugées comme étant les plus pertinentes.
- **Matcher :**il est utilisé pour comparer cette représentation avec les signatures de comportement.

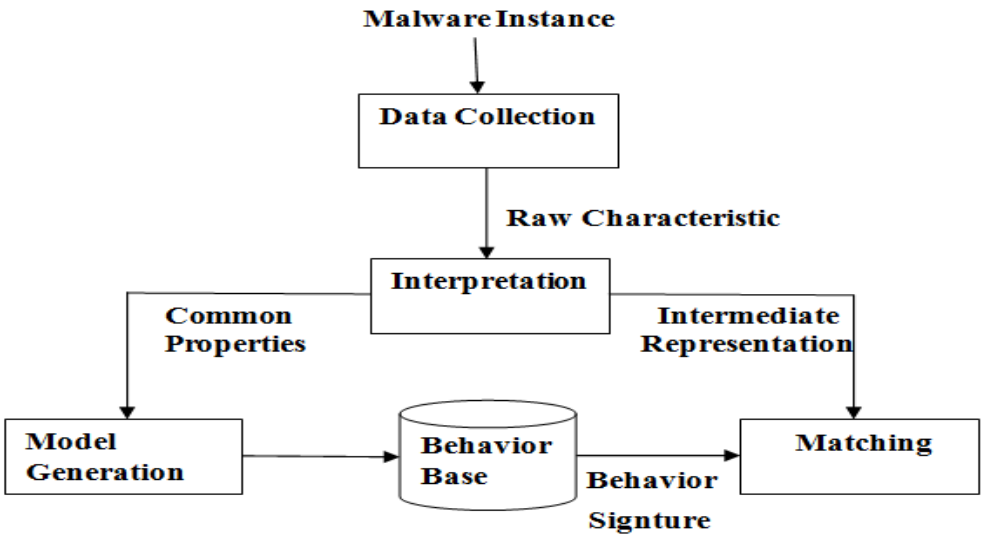

FIGURE 1.5 – Conception fonctionnelle d'un détecteur comportementale [17].

1.3.4 Mécanismes de protection contre les malwares informatiques

1.3.4.1 Anti-malwares

Un système de détection de malwares (antimalware) est un programme qui permet de détecter la présence de codes malicieux sur un ordinateur et de les supprimer. L'éradication d'un malware se fait de plusieurs façons qui sont [5] :

- La désinfection du fichier infecté en supprimant le code malveillant.
- La suppression du fichier infecté entièrement.
- La mise en quarantaine du fichier infecté, ce qui implique son déplacement vers un endroit où il ne pourra pas être exécuté.

Les anti_malwares utilisent différentes techniques pour la détection des codes malicieux telles que les techniques basées signatures (qui sont souvent utilisées par les antivirus commerciaux),les techniques comportementales, et les techniques heuristiques [13].

Antivirus résidant

Un antivirus " résident " est installé sur l'ordinateur comme n'importe quel autre programme classique. Il doit démarrer en même temps que l'ordinateur et rester actif durant tout le temps que dure la session de travail[18].

1.3.4.2 Fonctionnement

Les antivirus sont composé de trois parties ayant chacune un rôle essentiel[19] :

- **Un moteur** qui a pour rôle la détection des virus.
- **Une base de données** contenant des informations sur les virus connus. C'est cette base de données qu'il faut maintenir à jour le plus régulièrement possible, afin de permettre à l'antivirus de connaître les virus les plus récents.
- **Un module de nettoyage** qui a pour but de traiter le fichier infecté.

.

A chaque fichier testé, si le programme pense voir un virus,il regarde dans sa base de données si le virus est connu (chaque virus ainsi que ses variantes a une signature particulié, et c'est cette signature qui est comparée avec la base).

1.3.5 Célèbres attaques de malwares dans le monde

1.3.5.1 Robinn Hood

C'est une attaque qui concerne les ordinateurs du gouvernement de la ville de Baltimore. En avril 2018, les responsables de la ville de Baltimore ont découvré que les fichiers de leurs ordinateurs avaient été volées par des inconnus. Les attaquants ont demandé une rançon numérique de trois bitcoins (environ 17000 $) pour l'échange des informations volé. D'autres villes tels que Greenville, Amarillo et Atlanta, ont également touché par des attaques similaires[21].

1.3.5.2 WannaCry

WannaCry est un rançongiciel (ransomware) qui a défrayé la chronique en 2017,il a touché environ 230 000 machines dans 150 pays et il a laissé des dommages financiers estimées aux 4 milliards de dollars.

L'épidémie WannaCry se propage via des courriels piégés,puis s'installe sur la machine de la victime ensuite il chiffre les données du disque dur[22].

WannaCry a pris le contrôle des serveurs de pays, et aussi il a chiffré tous les appareils de certains hôpitaux et certaines usines ont été contraintes d'arrêter leur production.

Les créateurs de WannaCry ont demandé une somme d'argent en contrepartie de la restitution des informations cryptées[21].

1.3.5.3 My Doom

En 2004, un virus dénommé MyDoom fait son apparition. Le malware servait à prendre le contrôle des ordinateurs infectés pour envoyer de spam mais aussi pour lancer des attaques contre des entreprises comme Google et Microsoft.

Google était tellement préoccupé par le ver il a offert une récompense de 250 000 $ à toute personne qui peut fournir des informations sur le créateur.Microsoft dépensa 5 millions de dollars pour aider le FBI à trouver le créateur et plus 250.000$ de récompense à quiconque donnerait des informations sur lui[20].

MyDoom est considéré le virus le plus couteux car il a laissé des dommages financiers estimées au 38 milliards de dollars[21].

1.4 Conclusion

Dans ce chapitre nous avons présenté de manière globale les différentes concepts relative au domaine de sécurité informatique , et nous avons essayé de mettre l'accent sur les différentes types de malwares existant , leurs fonctionnements , et nous avons terminé par les techniques et les mécanismes utilisé pour la détection et la protection contre les malwares. Dans le chapitre suivant nous allons intéresser au domaine de machine Learning avec ces différents méthodes tel que les méthodes ensemblistes et les mesures d'évaluation(rappel, précision, F1-mesure..) .

Chapitre 2

Machine Learning

2.1 Introduction

L'intelligence artificielle est une vaste branche de l'informatique qui consiste à construire des machines capables de simuler l'intelligence.L'apprentissage automatique est une branche de l'intelligence artificielle,a pour l' objectif d'extraire et d'exploiter automatiquement l'information présente dans un jeu de données.Il existe plusieurs façons d'apprendre automatiquement à partir des données dépendamment des problèmes à résoudre et des données disponibles(apprentissage supervisé, non supervisé ..etc).

Dans ce chapitre,nous avons vu quelques définitions du machine learning, nous détaillerons par la suite ses différents types.

2.2 Machine Learning

2.2.1 Définition

L'apprentissage automatique est un sous-domaine de l'intelligence artificielle, où le terme fait référence à la capacité des systèmes informatiques à trouver indépendamment des solutions aux problèmes en reconnaissant les modèles dans les bases de données. En d'autres termes : le Machine Learning permet aux systèmes informatiques de reconnaître des modèles sur la base d'algorithmes et d'ensembles de données existants et de développer des concepts de solutions adéquats.

En 1959, **Arthur Samuel**[1] a défini l'apprentissage automatique comme un "do-

1. Arthur Samuel : est un pionnier américain du jeu sur ordinateur, de l'intelligence artificielle et de

maine d'étude qui donne aux ordinateurs la capacité d'apprendre sans être explicitement programmé". Par conséquent, dans le Machine Learning, la connaissance artificielle est générée sur la base de l'expérience.

Afin de permettre au logiciel de générer indépendamment des solutions, l'action préalable des personnes est nécessaire. Par exemple, les algorithmes et les données nécessaires doivent être introduits dans les systèmes à l'avance et les règles d'analyse respectives pour la reconnaissance des modèles dans data set doivent être définies. Une fois ces deux étapes terminées, le système peut effectuer les tâches suivantes par Machine Learning [23] :

- Trouver, extraire et résumer les données pertinentes ;
- Faire des prédictions sur la base des données d'analyse ;
- Calcul des probabilités pour des résultats spécifiques .

L'apprentissage automatique est un domaine d'expertise assez vaste dont l'application dans plusieurs domaines, dont celle de secteur de banque et assurance(prédire qui un client va quitter sa banque,détection des fraudes..etc), ainsi que la sécurité informatique (détection des malwares).

2.2.2 Méthodes de Machine Learning

Il existe plusieurs façon d'apprendre automatiquement,ces façons peuvent être divisés en différentes catégories :

2.2.2.1 Apprentissage supervisé

Au cours de l'apprentissage supervisé, des exemples de modèles sont définis à l'avance. Afin d'assurer une allocation adéquate des informations aux groupes de modèles respectifs des algorithmes, ceux-ci doivent alors être spécifiés. En d'autres termes, le système apprend sur la base de paires d'entrée et de sortie données. Au cours d'un apprentissage contrôlé, un programmeur, qui agit comme une sorte d'enseignant, fournit les valeurs appropriées pour une entrée particulière. L'objectif est de former le système dans le cadre de calculs successifs avec différentes entrées et sorties et d'établir des connexions [23].

L'apprentissage supervisé est principalement divisé en deux sous catégories permettant de classer des objets dans des classes à partir de variables qualitatives**(classification)** ou

l'apprentissage automatique

quantitatives**(régression)** caractérisant ces objets.

Régression

Dans l'apprentissage automatique, le but de la régression est d'estimer une valeur (numérique) de sortie à partir des valeurs d'un ensemble de caractéristiques en entrée. Par exemple, estimer le prix d'une maison en se basant sur sa surface, nombre des étages, son emplacement, etc. Donc, le problème revient à estimer une fonction de calcul en se basant sur des données d'entrainement [39].

Il existe plusieurs algorithmes pour la régression [39] :

— Régression linéaire.

— Régression polynomiale.

— Régression logistique.

— Régression quantile.

Classification

Quand la variable à prédire prend une valeur discrète, on parle d'un problème de classification. Parmi les algorithmes de classification, on retrouve : Support Vector Machine (SVM), Naïve Bayes,K plus proches voisins,les arbres de décision..etc ,les méthodes ensemblistes(bagging,boosting,Random forest).

Chacun de ses algorithmes a ses propres propriétés mathématiques et statistiques. En fonction des données d'entrainement (Training set), et nos features, on optera pour l'un ou l'autre de ces algorithmes. Toutefois, la finalité est la même : pouvoir prédire à quelle classe appartient une donnée (ex : un nouveau email est il spam ou non) [40].

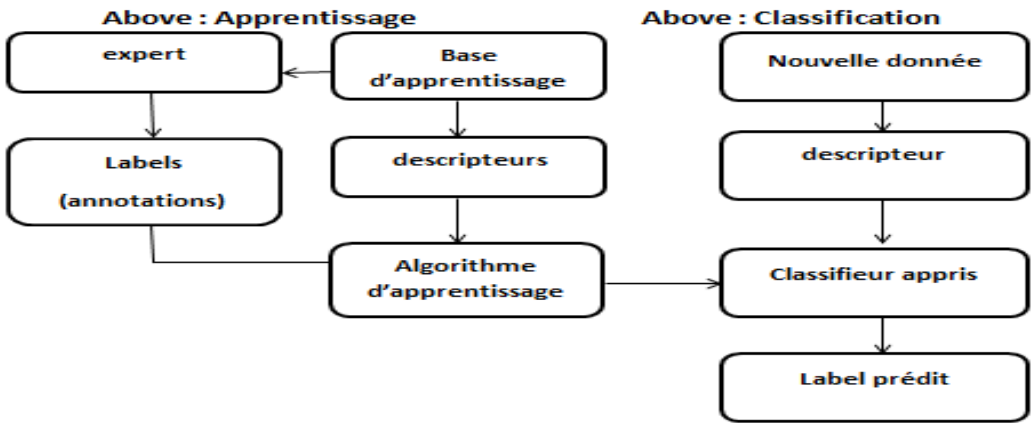

FIGURE 2.1 – Schéma général de l'apprentissage supervisé [40].

2.2.2.2 Apprentissage non supervisé

Dans l'apprentissage non supervisé, l'intelligence artificielle apprend sans valeurs cibles prédéfinies et sans récompenses. Il est principalement utilisé pour l'apprentissage de la segmentation (clustering). La machine essaie de structurer et de trier les données saisies selon certaines caractéristiques. Par exemple, une machine pourrait (très simplement) apprendre que des pièces de couleurs différentes peuvent être triées selon la "couleur" caractéristique afin de les structurer[23] .

2.2.2.3 Apprentissage partiellement supervisé

L'apprentissage partiellement supervisé est une combinaison d'apprentissage supervisé et non supervisé[23].

2.2.2.4 Apprentissage par Renforcement

Les données en entrée sont les mêmes que pour l'apprentissage supervisé, cependant l'apprentissage est guidé par l'environnement sous la forme de récompenses ou de pénalités données en fonction de l'erreur commise lors de l'apprentissage[24].

2.2.3 Quelques algorithmes du Machine Learning

La méthode des K plus proches voisins

Définition

KNN est un algorithme de reconnaissance des formes qui peut être utilisé autant pour la classification que pour la régression. C'est l'une des techniques non paramétriques fréquemment utilisée en prédiction financière non linéaire. Cette préférence est dû principalement à deux raisons [29] :

— **premièrement**, la simplicité algorithmique de la méthode comparée aux autres méthodes globales telles que les réseaux de neurones ou les algorithmes génétiques.

— **deuxièmement**, la méthode KNN a démontré empiriquement une importante capacité de prédiction.

Les étapes de réalisation

Pour appliquer cette méthode, les étapes à suivre sont les suivantes [32] :

- On fixe le nombre de voisins k.
- On détecte les k-voisins les plus proches des nouvelles données d'entrée que l'on veut classer.
- On attribue les classes correspondantes par vote majoritaire.

La détermination du plus proche voisin est basée sur un fonction distance arbitraire d(x,y) . La distance euclidienne ou dissimilarité entre deux individus caractérisés par p covariables est définie par[30] :

$d((x_1, x_2, ., x_p), ((y_1, y_2, ., y_p) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + ...(x_p - y_p)^2}$

Avantages [31]

Avantages	**Inconvénients**
- Pas d'hypothèse sur les distributions.	- Temps de calcul important (recherche des kNN).
- Simple à mettre en œuvre.	- Place mémoire (stockage de l'ensemble des prototypes).
- Donne une probabilité d'erreur faible.	-Le choix de la valeur de k (le nombre de voisins le plus proche)

TABLE 2.1 – Avantages et inconvénients de l'algorithme KNN .

Classification Naïve Bayésienne [33]

Définition

C'est un type de classification Bayésienne probabiliste simple basée sur le théorème de Bayes avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires (dont le rôle est de classer dans des groupes (des classes) les échantillons qui ont des propriétés similaires, mesurées sur des observations) .

Principe

La construction d'un classifieur naïf de bayes est comme suite :

- On estime la probabilité jointe P(X,Y).
- On calcule la probabilité conditionnelle de la classe c :
  $P(Y = c|X = x) = \frac{P(X=x,Y=c)}{P(X=x)}$

Avantages et inconvénients [34]

Avantages	**Inconvénients**
Il n'est pas sensible aux caractéristiques non pertinentes	Il implique que chaque fonctionnalité soit indépendante .
Il est facile à former, même avec un petit jeu de données .	
C'est relativement simple à comprendre et à construire .	

TABLE 2.2 – Avantages et inconvénients de l'algorithme naïf de bayes .

2.3 Réduction de dimension

L'augmentation du nombre des variables (caractéristiques) qui modélisent le problème introduit des difficultés à plusieurs niveaux comme la complexité, le temps de calcul ainsi que la détérioration du système de résolution en présence de données bruitées. Une méthode de réduction de la dimensionnalité consiste à trouver une représentation des données initiales dans un espace plus réduit. Les méthodes de réduction de la dimensionnalité sont généralement classées dans deux catégories [25] :

- Une réduction basée sur une sélection de caractéristiques qui consiste à **sélectionner les caractéristiques** les plus pertinentes à partir de l'ensemble de données des variables décrivant le phénomène étudié.
- Une réduction basée sur une transformation des données appelée aussi **une extraction de caractéristique**s et qui consiste à remplacer l'ensemble initial des données par un nouvel ensemble réduit, construit à partir de l'ensemble initial de caractéristiques.

2.4 Features Extraction

L'extraction de caractéristiques est un processus de réduction de dimensionnalité par lequel un ensemble initial de données brutes est réduit à des groupes plus gérables pour le traitement. Une caractéristique de ces grands ensembles de données est un grand nombre de variables qui nécessitent beaucoup de ressources informatiques pour être traitées. L'extraction de caractéristiques est le nom des méthodes qui sélectionnent et / ou combinent des variables dans des caractéristiques, réduisant efficacement la quantité de données qui doivent être traitées, tout en décrivant de manière précise et complète l'ensemble de données d'origine[26].

2.4.1 Méthode d'extraction des caractéristiques

Le prétraitement des données pour le machine learning implique à la fois l'extraction des données et des caractéristiques. L'extraction des données consiste à convertir des données brutes en données préparées. L'extraction de caractéristiques ajuste ensuite les données préparées de manière à créer les caractéristiques attendues par le modèle machine learning.

Les diverses méthodes d'extraction de caractéristiques ont été proposé dans la littérature sont :

- **Extraction des caractéristiques binaires :** La méthode de base pour extraire des entités d'une séquence consiste à identifier tous les éléments distincts trouvés dans cette séquence. La séquence peut alors être représentée comme un vecteur binaire de la même longueur que le terme dictionnaire S telle que chaque caractéristique du vecteur signifie la présence ou l'absence du terme de dictionnaire correspondant dans la séquence. Le vecteur caractéristique résultant peut être représenté par VSb = (bs1 ; bs2 ; : : : ; bsn), où bsi vaut 1 si S contient au moins une instance de si et 0 sinon, et n est la taille de S . Cette méthode a été utilisée pour des séquences d'octets de code , d'appels d'API , etc [26].
- **Extraction des caractéristiques de fréquence :**
  Dans cette méthode d'extraction des caractéristiques, le nombre d'occurrences d'un terme de dictionnaire dans la séquence est utilisé à la place de sa présence ou de son absence [26].
- **Extraction des caractéristiques de poids de fréquence (TF/IDF) :** Des méthodes de pondération de fréquence telles que le terme fréquence-fréquence de document inverse (TF-IDF) ont également été utilisé pour générer des vecteurs de caractéristiques à partir de séquences . TF-IDF est une statistique largement utilisée dans les domaines de recherche d'information et d'exploration de texte pour calculer les pondérations basées sur la fréquence des termes dans un document afin de évaluer leur importance relative [25].

2.5 Features Selection

L'objectif de la sélection des caractéristiques est de supprimer les caractéristiques non importantes. Sa suppression n'affectera en rien la précision. Nous distinguons trois classes générales de méthodes de la sélection des caractéristiques et qui sont [26] :

- **Méthode de filtrage (Filter methods) :** Les méthodes de filtrage notent statistiquement les fonctionnalités. Les entités avec des scores plus élevés sont conservées dans l'ensemble de données, tandis que les entités avec des scores faibles sont supprimées .

- **Méthode d'encapsulation (Wrapper methods) :** faire une combinaison (conduit à la plus grande précision).
- **Méthode d'intégration (Embedded methods) :** évaluent les fonctions utilisés lors de la création de modèle.

2.6 Méthodes ensemblistes

Une méthode ensembliste selon la définition de Dietterich [2] est un ensemble de classificateurs individuels qui sont divers, mais précis, et dont les décisions sont combinées par moyenne ou par vote(la décision la plus populaire) pour donner une décision beaucoup plus précise. Selon Dietterich,un classificateur est précis si son taux d'erreur est plus faible qu'une classification aléatoire , et deux classificateurs sont divers s'ils font des erreurs différentes sur de nouvelles données. Notons que ses méthodes peuvent être appliquées à tous algorithmes instables, ceux pour lesquels un petit changement dans les données d'entraînement induit un grand changement dans le classificateur final [27].

2.6.1 Quelques techniques ensemblistes

2.6.1.1 Bagging

Bagging est une méthode d'ensemble introduite par Breiman [3] en1996 est basée sur les concepts de Bootstrapping et d'aggregating. Le Bootstrap est un principe de rééchantillonnage statistique traditionnellement utilisé pour l'estimation de grandeurs ou de propriétés statistiques[28]. Le bootstrapping est conçu pour générer au hasard et avec remise L copies indépendantes de S objets appelées bootstrap à partir de l'ensemble initial des échantillons d'apprentissage de taille S. Un objet de la base initiale peut être sélectionné plusieurs fois comme il peut être absent dans les copies générées. Le même classifieur est appris sur chacune des copies. On obtient par la suite L classifieurs avec des performances différentes [27].

2. G. Dietterich est professeur émérite d'informatique à l'Oregon State University. Il est l'un des fondateurs du domaine de l'apprentissage automatique.

3. Leo Breiman (17 janvier 1928 – 5 juillet 2005) était un statisticien de renom à l'université de Californie.Sa contribution la plus importante concerne son travail sur les arbres de régression et de classification et les ensembles d'arbres taillés pour les échantillons traités par les techniques de bootstrap.

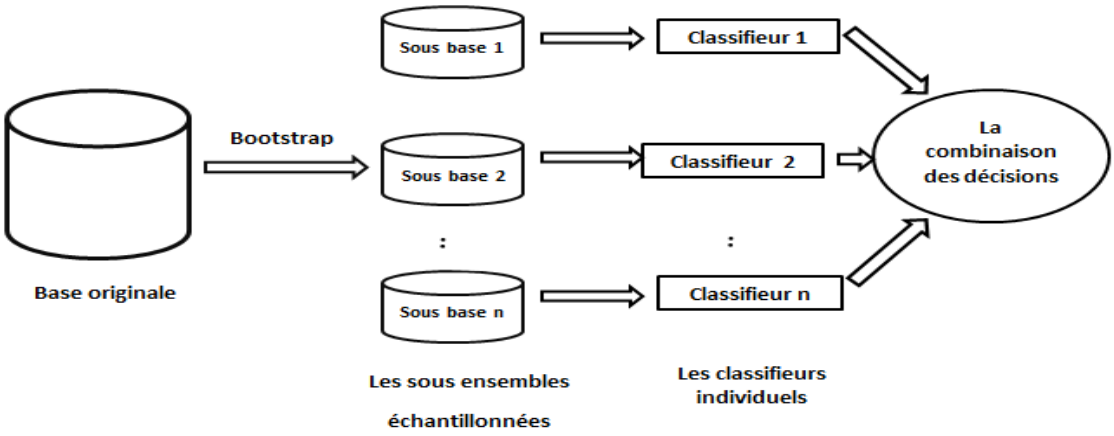

FIGURE 2.2 – Principe de Bagging [27].

Forêts aléatoires

Définition

Une forêt aléatoire Breiman est le mélange des deux techniques de "Bagging" et de "Random SubSpace"appliqués sur des arbres de décisions. A chaque itération, un échantillon "boostrap" est tiré aléatoirement afin de construire un arbre de décision binaire.

L'espace de recherche pour la construction des nœuds de l'arbre est limité par P caractéristiques tirées aléatoirement. La performance de la méthode dépend directement du paramètre P. Une petite valeur de P risque de dégrader les performances du classificateur. Dans (Breiman), l'auteur a montré empiriquement que la valeur optimale de P est :$P = \sqrt{N}$ ou N est le nombre total de caractéristiques [25].

Principe

Les forêts aléatoires sont basées sur [37] :

- Utiliser un grand nombre d'arbres de décision construits chacun avec un sous-échantillon différent de l'ensemble des données (bootstraping).
- Pour chaque construction d'arbre, la décision à un nœud (répartition des objets) est faite en fonction d'un sous-ensemble de variables tirées au hasard : on tire

aléatoirement m variables parmi les p disponibles et on cherche parmi celles-ci la meilleure coupure (avec toujours le même critère).

- Utiliser l'ensemble des arbres de décision produits pour faire la prédiction/choisir le nombre de groupes, avec un choix fait à la majorité.
- Classification : variable prédite est de type facteur.
- Régression : variable prédite de type numérique .

Avantages et inconvénients [35]

Avantages	**Inconvénients**
Efficace sur inputs de grande dimension.	Apprentissage souvent long.
Reconnaissance trés rapide , multi-classes par nature.	Valeurs extrêmes souvent mal estimées dans cas de régression .

TABLE 2.3 – Avantages et inconvénients de forêts aléatoires .

2.6.1.2 Boosting

Le "Boosting" (Schapire) [4] désigne un principe général d'apprentissage permettant d'améliorer la précision d'un algorithme d'apprentissage donné. Le principe général est de combiner linéairement des résultats de classificateurs dits "faibles" afin de construire un classificateur "fort" d'apprentissage à partir de l'ensemble original et une méthode de combinaison de classificateurs construits à partir de chaque nouvel ensemble[25].

Pour définir sa nouvelle technique de "Boosting", Shapire se base sur l'idée que tout classificateur faible capable d'apprendre avec une certaine confiance et une erreur de classification inférieure à (0.5), peut être transformer en un classificateur plus confiant et avec une erreur de classification aussi petite que désirée. En d'autres termes, un classificateur faible donnant de meilleurs résultats qu'un simple pile ou face (50% de risque) peut être la base pour construire un ensemble de classificateurs. A chaque itération, l'algorithme cherche à trouver un classificateur faible qui peut corriger au mieux les erreurs des classi-

4. Robert Elias Schapire est professeur et chercheur en informatique. Son travail est centré autour de l'apprentissage automatique. Il est connu pour son travail avec Yoav Freund sur l'algorithme Adaboost qui leur a valu le prix Gödel en 2003

ficateurs obtenus aux itérations précédentes. Dans le principe de "Boosting", cet objectif est réalisé à l'aide d'une pondération des données d'apprentissage[25].

La version de boosting qui est implantée est une variante de celle proposée par Freund [5] & Shapire[1996],appelée Adaboost [27].

Adaboost

AdaBoost (Adaptive Boosting) est une méthode algorithmique connue dans le domaine de l'apprentissage. Cette méthode a été initialement conçue par Robert Schapire et Yoav Freund en 1995 . Cette approche est la dérivée la plus partiquée de la méthode du Bossting qui vise à stimuler la performance de l'algorithme d'apprentissage. Adaboost consiste à transformer, d'une manière efficace, un classifieur « faible » en un classifieur « fort » en réduisant les taux d'erreur.

L'algorithme d'Adaboost, appelle à chaque itération, un algorithme d'apprentissage qui entraînent les instances à classifier. Par la suite, Adaboost définit une nouvelle distribution de probabilité pour les instances d'apprentissage en fonction des résultats de l'algorithme à l'itération précédente tout en augmentant le poids des instances qui ne sont pas correctement classées. A la fin, Adaboost combine les données faibles par un vote pondéré pour en déduire un classifieur fort. Dans cette optique, AdaBoost essaye, de trouver un classifieur optimal à partir de la combinaison d'un ensemble de données d'apprentissage faible [41].

5. Yoav Freund est un chercheur d'informatique théorique et professeur de l'Université de Californie à San Diego (UCSD) qui travaille principalement dans le domaine de l'apprentissage automatique et la théorie des probabilités.

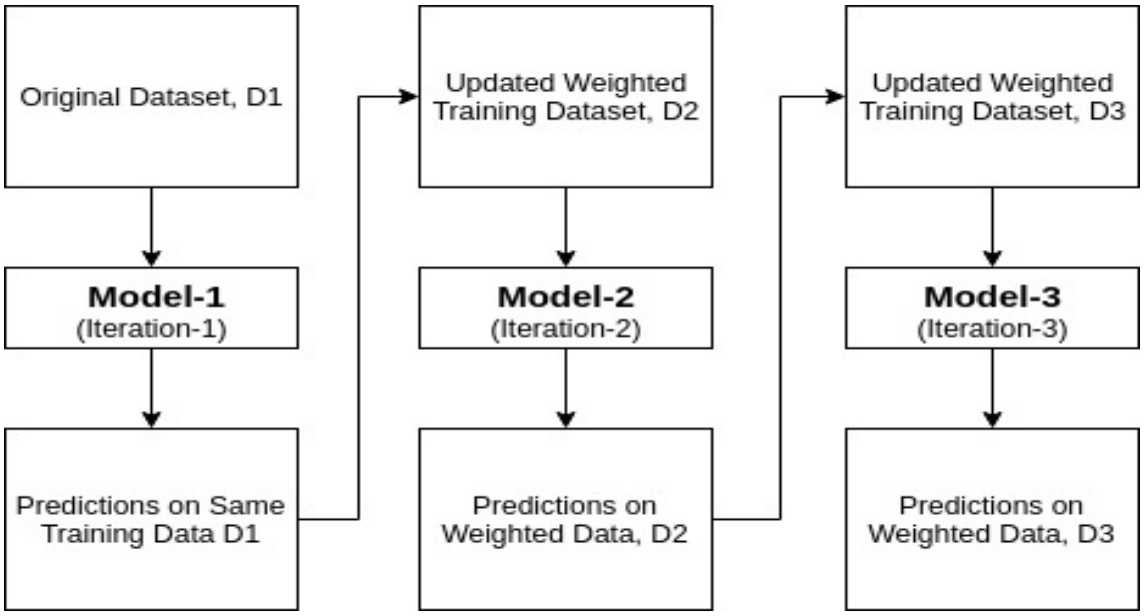

FIGURE 2.3 – Principe d'AdaBoost[42].

Avantages [43]

- Rapide ,simple et facile à implémenter .
- Applicable à de nombreux domaines par un bon choix de classifieur faible .
- Pas de sur_apprentissage par la maximisation de la marge .

Inconvénients [44]

- Parfois particulièrement sensible au bruit présent dans les données.
- Importance du choix des classifieurs faibles,en particulier il ne faut pas que chaque classifieur faible pris individuellement soit trop "bon".

Gradient Boosting

Le Gradient Boosting est une technique d'apprentissage automatique utilisée pour des problèmes de régression ou de classification.

Comme les autres méthodes de boosting, le Gradient Boosting optimise les performances d'un ensemble de modèles de prédiction dits « faibles » en les assemblant en un modèle final. On appelle modèle de prédiction « faible », une méthode de classification ou de régression qui est à peine plus efficace qu'un tirage aléatoire. Le modèle de prédiction faible généralement utilisé avec un Gradient Boosting est un arbre de décision CART. De manière plus concrète, cette méthode du Gradient Tree Boosting consiste alors à réaliser

une succession d'arbres de décision où chaque modèle est construit sur l'erreur résiduelle du précédent [45].

1. XGBoost

L'algorithme Extreme Gradient Boosting (XGBoost) est similaire à l'algorithme du Gradient boosting, néanmoins il est plus efficace et plus rapide puisqu'il est composé à la fois d'un modèle linéaire et des modèles d'arbres. Cela en plus de sa capacité à effectuer des calculs parallèles sur une seule machine. Les arbres construits dans un algorithme du gradient boosting sont construits en serie pour qu'une étape de descente du gradient puisse être effectuée afin de minimiser une fonction de perte. Contrairement aux Random Forest, l'algorithme du XGBoost, construit l'arbre lui-même d'une maniere parallèle. Essentiellement les informations contenues dans chaque colonne correspondant a une variable peuvent avoir des statistiques calculées en parallèle. L'importance des variables est calculée de la même manière que pour les forêtes aléatoires, en calculant et moyennant sur les valeurs par laquelle une variable diminue l'impurte de l'arbe à chaque étape [38].

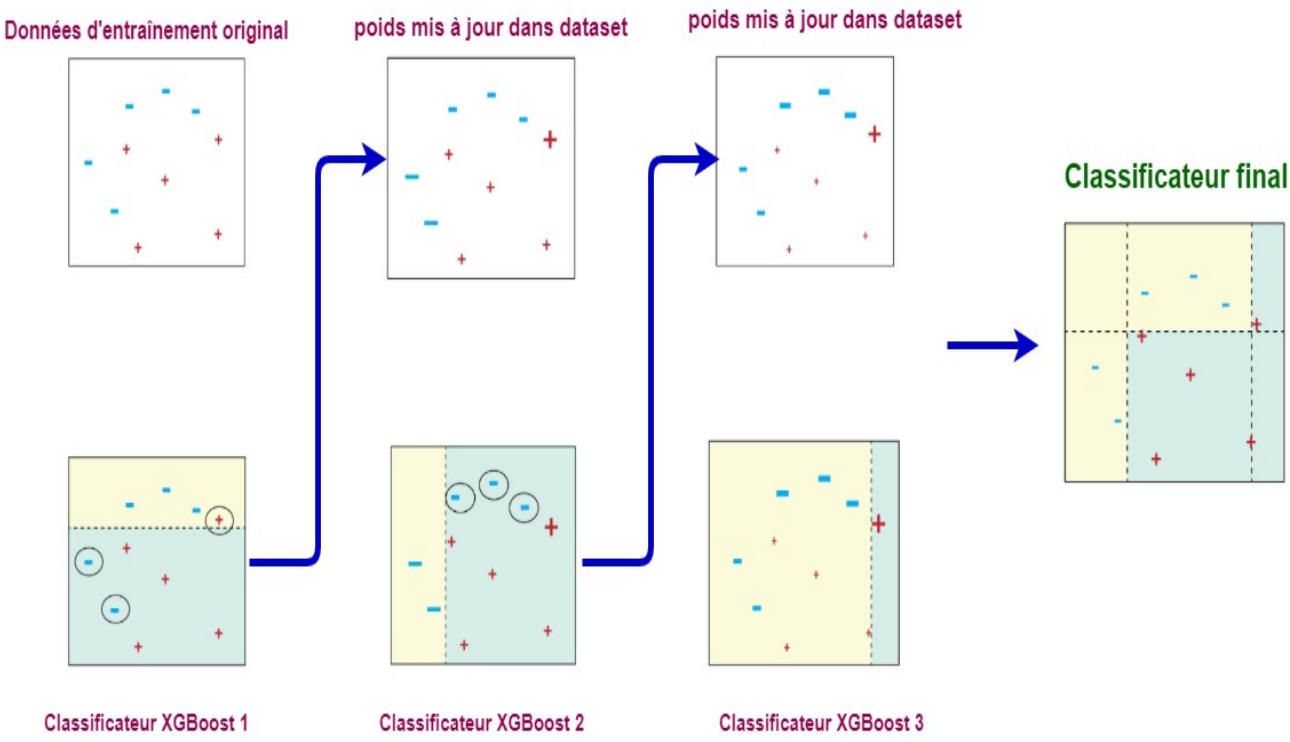

FIGURE 2.4 – Simple exemple sur le fonctionnement de l'algorithme XGBoost [38].

2. LightGBM

Abréviation de Light Gradient Boosted Machine, c'est l'une des méthodes ensemblistes basés sur les arbres de décision pour augmenter l'efficacité du modèle . LightGBM divise l'arbre par feuille, par opposition à d'autres algorithmes de gradient boosting qui poussent au niveau de l'arbre. Il choisit la feuille avec une perte maximale pour se développer. Étant donné que la feuille est fixe, l'algorithme feuille à feuille a une perte inférieure par rapport à l'algorithme par niveau. La croissance des arbres par feuilles peut augmenter la complexité du modèle et entraîner un sur_apprentissage dans des petits ensembles de données[67].

Comme le montre la figure **2.5** suivante, LightGBM agrandit toujours son arbre à la manière des feuilles. Il choisit la feuille qui a donné la perte maximale à la précision du modèle.

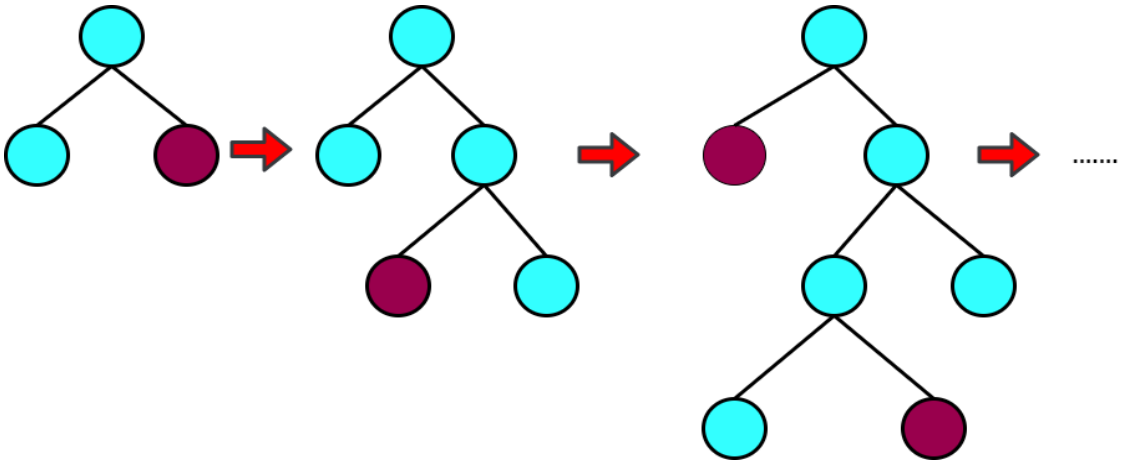

FIGURE 2.5 – Croissance par feuille de LightGBM [67].

2.7 Méthodes d'évaluation

2.7.1 Matrice de confusion

La matrice de confusion, appelée également matrice d'erreur, est un tableau qui présente différentes prévisions et résultats de tests, en les comparant avec des valeurs réelles. Ces matrices sont utilisées dans les statistiques, le data mining, les modèles d'apprentissage automatique et d'autres applications de l'intelligence artificielle [36].

Les quatre données trouvées dans la matrice de confusion sont :

- ***TP(Correct Positive) :*** classe positive considérée positive.
- **TN(Correct Negative) :** classe negative considérée negative.
- **FP(False Positive) :** classe negative considérée positive.
- **FN(False Negative) :** classe positive considérée negative.

	Classe 1	Classe 0
Classe 1	TN	FP
Classe 0	FN	TP

TABLE 2.4 – Matrice de confusion.

Notons bien que c'est avec ces quatre paramètres que toutes les autres mesures sont calculées.

2.7.1.1 Précision

précision (ou valeur prédictive positive) est la proportion des items pertinents parmi l'ensemble des items proposés .

$$Précision = \frac{TP}{TP+FP}.$$

2.7.1.2 Rappel(sensibilité)

le rappel (ou sensibilité) est la proportion des items pertinents proposés parmi l'ensemble des items pertinents.

$$Rappel(Se) = \frac{TP}{TP+FN}$$

2.7.1.3 Moyenne harmonique

Moyenne harmonique de la précision et du rappel mesure la capacité du système à donner toutes les solutions pertinentes et à refuser les autres.

$$F1 - Score = \frac{2*TP}{2*TP+FP+FN} = \frac{2*(Précision*Rappel)}{Précision+Rappel}$$

2.7.1.4 Accuracy

Nombre relatif d'exemples correctement classés ou en d'autres termes pourcentage de prédictions correctes.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

2.8 Méthodes de validation

Pour avoir une estimation correcte de l'erreur de classification, il faut recourir à un ensemble d'exemples qui n'ont pas servi pour l'apprentissage.Il s'agit de l'ensemble de test.

Nous citerons deux méthodes de validation généralement utilisées : validation par le test, validation croisée.

2.8.1 validation croisée

Les techniques de validation croisée (ou cross validation), permettent d'obtenir une estimation des performances du prédicteur, en exploitant la totalité du jeu de données. Ceci est obtenu en faisant plusieurs tests sur différents ensembles d'apprentissage et de test, et en faisant la moyenne des résultats. En effet, si l'on obtient une bonne précision en moyenne, ainsi qu'un écart-type faible, notre méthode de prédiction pourra être considérée comme robuste [46].

2.8.1.1 Validation Croisée : K-Fold

L'algorithme de validation croisée **K-Fold** consiste à segmenter aléatoirement les exemples du jeu de données initial en k sous-ensembles disjoints numérotés de **1 à k**. Ensuite, on va écarter le 1 er bloc qui nous servira d'ensemble de test, et utiliser les **k-1** autres blocs afin de constituer l'ensemble d'apprentissage [24].

2.9 Conclusion

Nous avons présenté à travers ce chapitre,de façon générale,l'apprentissage automatique avec ses différents types.Comme nous avons cité les méthodes du machine learning en définissant quelques algorithmes les plus utilisés.Aussi,nous avons présenté quelques types des méthodes ensemblistes(Bagging,Boosting..).

À la fin,nous avons vu les techniques qui peuvent être utilisés pour l'évaluation de la performance des modèles d'apprentissage automatique(Précision,F1-Score,..) .

Chapitre 3

Approche et Solution

3.1 Introduction

Avec l'évolution de la technologie,les cybercriminels essayent toujours de développer des nouveaux malwares (logiciel malveillant) qui représentent une réelle menace pour la sécurité des systèmes informatiques, ces derniers peuplent notre quotidien et interviennent dans toutes nos activités .

Malheureusement ,ils profitent chaque fois des bugs existant dans nos systèmes pour lancer des attaques avec divers familles des logiciels malicieuses .

Un malware(logiciel malicieux) est un type de logiciel ou code dont l'objectif est de voler les informations privées ou des données de systèmes sans prendre une autorisation de l'utilisateur,l'industrie de ces logiciels reste un marché organisé dédié à la rupture des systèmes de sécurités .Bref,la qualité et la diversité des malwares rendent les défenses inefficaces.

Par ailleurs, la détection de ces logiciels reste classique et incapable de protégé nos machines car elle utilise des anti_virus commerciaux qui nécessite par fois des mise à jour en ligne .

Dans ce chapitre, nous présentons notre problématique avec quelques travaux connexes. Par la suite nous ajoutons une analyse exploratoire des données ,vient ensuite l'étape de préparation des données et en fin nous abordons notre proposition avec une architecture globale de notre approche.

3.2 Problématique étudiée

La détection et la prédiction des malwares dans une machine c'est l'un des problèmes les plus soulevées par la sécurité informatique , mais nous pouvons les résoudre avec diverses méthodes de machine learning (classification binaire) .

La détection des malwares sur une machine windows dépend des techniques classiques ,les anti_malwares sont incapable de fournir une protection efficace et performante pour nos machines surtout tant que la machine est connectée , la détection de ces logiciels devient très compliquée et nos ordinateurs deviennent vulnérable aux dommages.

Pour cela Microsoft est intéressé à cette problématique afin de prédire si une machine sera bientôt touchée par des malwares ou pas dans le but de bien sécuriser nos données .

L'objectif de notre travail est de prédire et détecter à l'aider d'un modèle ensembliste (XGboost) la probabilité qu'une machine windows soit infectée par divers familles de logiciels malveillants , il nous est demandé d'évaluer la vulnérabilité de la machine en fonction de sa configuration.

3.3 Travaux connexes

Dans cette section , nous discuterons quelques travaux similaires sur l'analyse et la détection des malwares .

1. **Gavrilut et al**,(2009)[48] , ont expérimenté une technique d'apprentissage automatique supervisé en ligne pour classer les fichiers comme bénins ou malveillants. Contrairement aux techniques d'apprentissage automatique hors ligne, qui entraînent des modèles à l'aide d'un ensemble d'entraînement, puis évaluent les résultats à l'aide de l'ensemble de test, les techniques en ligne apprennent lorsque les données transitent par l'algorithme dans un flux. Le modèle prédit le résultat de l'observation réussi, il évalue ensuite le résultat par rapport au résultat réel. Le modèle apprend des erreurs commises. Pour cette expérience, les auteurs ont choisi de sélectionner une combinaison de 308 caractéristiques pour représenter les caractéristiques géométriques ou comportementales des logiciels malveillants. En utilisant ces caractéristiques, les auteurs ont décidé de compter toutes les observations qui génèrent la même valeur qu'une. Ainsi, la réduction du nombre

d'observations de fichiers bénins et malveillants. Le nouvel ensemble de données après avoir supprimé la similitude de comportement est plus biaisé pour favoriser la détection des logiciels malveillants , il s'agit d'un effet secondaire du choix des fonctionnalités qui représentent les caractéristiques géométriques ou comportementales des logiciels malveillants. Il n'est cependant pas clair comment les auteurs ont défini les caractéristiques géométriques et comportementales des logiciels malveillants et comment ces fonctionnalités ont été extraites. Ainsi, l'évaluation de cette expérience est très limitée et ne peut être reproduite par manque d'informations.

2. **Tony Abou-Assaleh et al** [49],ont proposé un modèle qui appliquait le classificateur k-plus proche voisin (KNN) avec la méthode d'analyse Common N-Gram (CN-G) pour extraire et sélectionner les fonctionnalités de fichier . D'où l'idée de cette recherche est venue de la méthode (CN-G) généralement appliquée dans la classification de texte et le traitement du langage naturel. En appliquant la manière d'un octet à la fois de fenêtre glissante sur fichier, les auteurs ont rassemblé des octets n-grammes qui se chevauchaient sous-chaînes, ainsi, des statistiques de sous-chaînes de longueur n et les fréquences de sous-chaînes plus longues ont été collectées. Des N-Gram très fréquents ont été produits par analyse N-Gram et représentaient des signatures. Par conséquent, les n-grammes pourraient être utilisés pour prédire un programme invisible comme un programme malveillant ou bénin en fonction de la similitude des fonctionnalités avec les catégories d'échantillons connues antérieures. Le modèle de caractéristiques était implicite dans les n-grammes sélectionnés. Par conséquent, les auteurs de virus ont une tâche complexe d'écrire des virus qui peuvent tromper l'analyse de n-gramme, bien qu'ils connaissaient ou pouvaient accéder à l'algorithme de détection. Cependant, le profil de classe généré à partir des n-grammes les plus fréquents avec leurs fréquences normalisées qui ont été recueillies à partir de l'étape de date de formation, les paramètres des profils de classe étaient la longueur du profil et la taille des n-grammes. Code invisible détecté comme malveillant ou bénin selon la classe la plus similaire par utilisation dans l'algorithme KNN avec $k = 1$. L'ensemble de données des chercheurs comprenait 40 exécutables Windows bénins et 25 vers provenant d'emails infectés. Les résultats des chercheurs étaient une précision moyenne de 98% avec une validation croisée

triple et une précision de 100% pour la formation avec une certaine disposition des paramètres.

3. **Schultz et al**[51], ont mis en place le premier système de détection de malwares basé sur des techniques d'apprentissage automatique. Les auteurs ont étudié différentes informations contenues dans le fichier PE tels que des chaines de caractères, les APIs, et la séquences d'octets. Ils ont utilisé une méthode de classification basée sur un algorithme Bayésien naif, et ils ont obtenu une précision globale de 97,11% en utilisant les chaines de caractères comme attributs.

3.4 EDA - Exploratory Data Analysis, analyse et exploration des données (dataset)

L'analyse exploratoire des données (AED) est un processus ouvert dans le cadre duquel nous calculons des statistiques et établissons des chiffres pour trouver des tendances, des anomalies, des modèles ou des relations au sein des données.La faculté de AED à analyser et représenter les données en détailles que offre au data_scientiste une capacité à délivrer rapidement des résultats vis-à vis des métiers et à établir des hypothèses avant de lancer une analyse prédictives. Le but de l'AED est d'apprendre ce que nos données peuvent nous dire. nous commençons par une vue d'ensemble de haut niveau, puis se réduit à des domaines spécifiques car nous trouvons des zones intrigantes des données. Les résultats peuvent être intéressants en eux-mêmes, ou ils peuvent être utilisés pour éclairer nos choix de modélisation, notamment en nous aidant à choisir les fonctionnalités à utiliser.

3.4.1 Jeu de données Microsoft Malware Prediction

Microsoft Malware Prediction[1] est un ensemble de données de classification binaire(HasDetections '0' ou '1'),il contient environ 16.774.736 machines et 81 attributs et (identifient de la machine et la cible 'HasDetections') , dont 52 attributs catégoriques (23 attributs sont codes numériquement pour protéger la confidentialités des données).

Ce jeu de données a été préparé par Microsoft et Windows Defender ATP Research avec des contributions supplémentaires de l'université de boston Northeastern et le Georgia

1. https ://www.kaggle.com/c/microsoft-malware-prediction/data

Tech Institute for Information Security & Privacy.

Dans le but de prédire la probabilité qu'une machine Windows soit infectée par diverses familles de logiciels malveillants, en fonction des différentes propriétés de cette machine. La méthodologie d'échantillonnage utilisée pour créer **Microsoft Malware Prediction** a été conçue pour répondre à certaines contraintes commerciales, tant en ce qui concerne la confidentialité des utilisateurs que la période pendant laquelle la machine était en marche. Cet ensemble de données n'est pas représentatif des machines des clients Microsoft dans la nature,il a été échantillonné pour inclure une proportion beaucoup plus importante de machines malveillantes.

Cette base est devisé en deux sous-ensembles,une base d'entraînement (**train.csv**) et une base de test (**test.csv**).

- **Ensemble d'entraînement :** contenant des informations sur 8.921.483 machines avec 83 attributs (décrivant diverses informations concernant le hardware et le software des machines).

La figure **3.1** représente une partie de cet ensemble d'entraînement :

base entrainement: (8921483, 83)

	MachineIdentifier	ProductName	EngineVersion	AppVersion	AvSigVersion	IsBeta	RtpStateBitfiel
0	0000028988387b115f69f31a3bf04f09	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1735.0	0	7
1	000007535c3f730efa9ea0b7ef1bd645	win8defender	1.1.14600.4	4.13.17134.1	1.263.48.0	0	7
2	000007905a28d863f6d0d597892cd692	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1341.0	0	7
3	00000b11598a75ea8ba1beea8459149f	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1527.0	0	7
4	000014a5f00daa18e76b81417eeb99fc	win8defender	1.1.15100.1	4.18.1807.18075	1.273.1379.0	0	7

5 rows × 83 columns

FIGURE 3.1 – Ensemble d'entraînement.

- **Ensemble de test :** contenant des informations sur 7.853.253 machines avec 82 attributs (décrivant diverses informations concernant le hardware et le software des machines).

Le tableau **3.1** suivant illustre les attributs du data-set que nous avons utilisé :

Attribut	**Description**
AvSigVersion	Version de la signature antivirus.
AVProductStatesIdentifier	L'identifiant pour la configuration spécifique du logiciel antivirus d'un utilisateur.
Census_FirmwareManufacturerIdentifier	Identifiant du fabricant du micrologiciel.
Census_FirmwareVersionIdentifier	Identificateur de version du micrologiciel.
Census_InternalPrimaryDiagonalDisplaySizeInInches	Taille de l'écran diagonale principale interne en pouces .
Census_OEMNameIdentifier	Identifiant du nom de fabrication d'équipement d'origine .
Census_OEMModelIdentifier	Identificateur de modèle de fabrication d'équipement d'origine .
Census_OSBuildRevision	Révision de build du système d'exploitation .
Census_OSInstallTypeName	Description conviviale de l'installation utilisée sur la machine, c'est-à-dire propre .

Census_OSUILocaleIdentifier	Identificateur de paramètres régionaux de l'interface utilisateur du système d'exploitation.
Census_ProcessorModelIdentifier	Identificateur de modèle de processeur.
Census_PrimaryDiskTotalCapacity	Quantité d'espace disque sur le disque principal de la machine en Mo. Les tailles de disque les plus courantes sont de 500 Go et 1 To.
Census_SystemVolumeTotalCapacity	Taille de la partition sur laquelle le volume système est installé en Mo .
CountryIdentifier	Identifiant du pays dans lequel se trouve la machine .
GeoNameIdentifier	Identificateur de la région géographique dans laquelle se trouve la machine .
LocaleEnglishNameIdentifier	Nom anglais de l'identificateur de paramètres régionaux de l'utilisateur actuel.
SmartScreen	La valeur activée pour l'écran intelligent du registre.
UacLuaenable	Cet attribut indique si le type d'utilisateur "administrateur en mode d'approbation administrateur" est désactivé ou activé dans UAC.
Wdft_IsGamer	Indique si l'appareil est un joueur dispositif ou non en fonction de sa combinaison matérielle.
Wdft_RegionIdentifier	La documentation Microsoft mentionne un identifiant régional.

TABLE 3.1 – Attributs et leurs explications.

3.5 Visualisation des données

3.5.1 Visualisation de la cible (HasDetections)

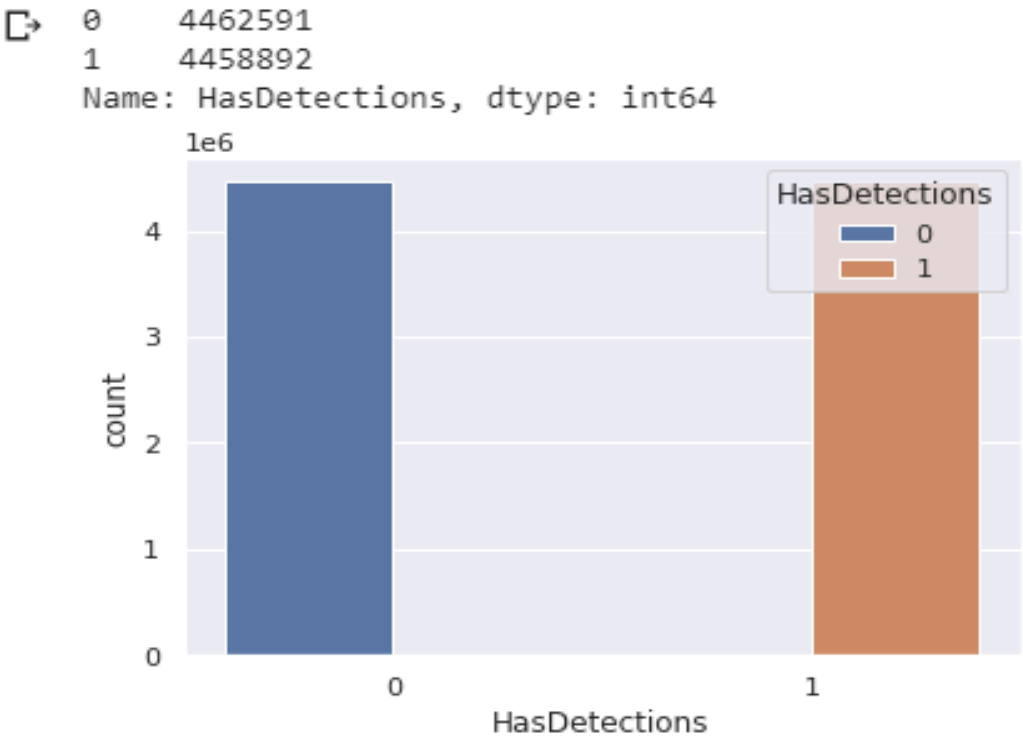

FIGURE 3.2 – Cible **HasDetection**.

D'après ces informations , nous voyons qu'il s'agit d'un dataset équilibré **(4.462.591 pour la classe 0 et 4.458.892 pour la classe 1)** tel que , l'étiquette **1** indique que ***des logiciels malveillants*** ont été détectés sur la machine .

3.5.2 Analyse des machines infecté en fonction de "Platform"

Platform est un attribut catégorique , il a quatre valeurs possibles : windows10 , windows8, windows7 ,windows2016.

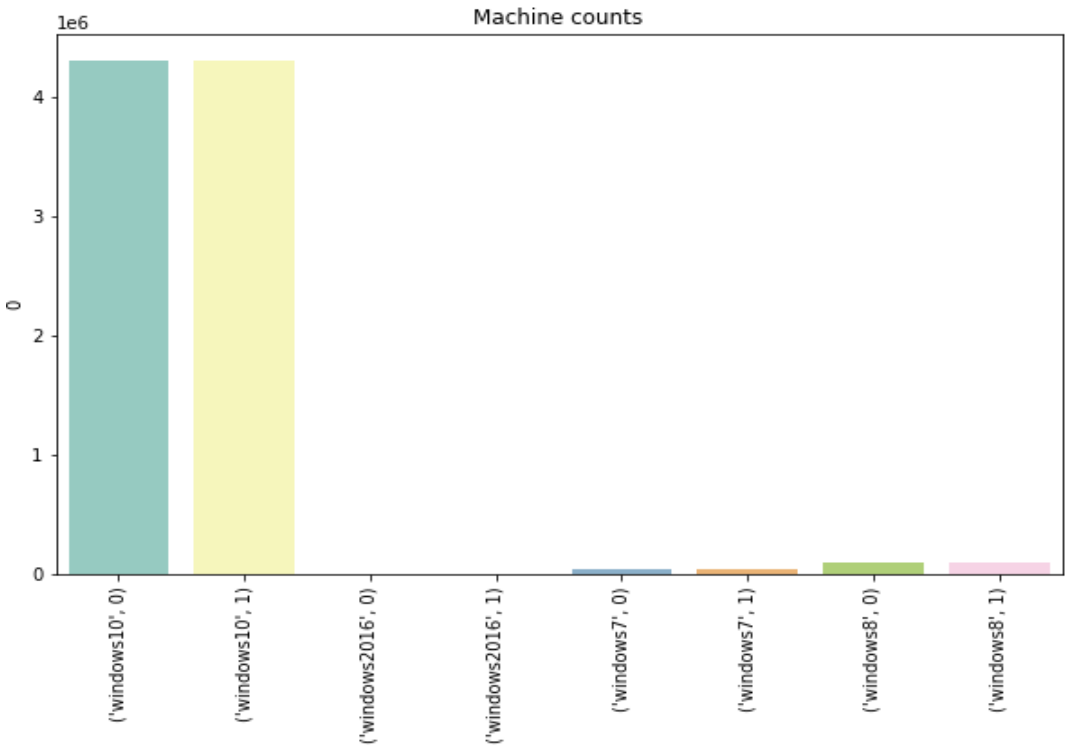

FIGURE 3.3 – Platform des machines.

D'après le résultat obtenu , Windows10 est la plate-forme la plus infectée avec plus de 4 millions de machines .

3.6 Pré_traitement des données

Le Prétraitement des données (Preprocessing) c'est l'étape qui consiste à préparer nos données avant de les fournir à la machine pour son apprentissage , le but est mettre les données dans un format propice au machine learning afin d'améliorer la performance de ses modèles.

Parmi les opérations de prétraitement les plus importantes,que nous avons appliqué à l'ensemble de données **Microsoft malware prediction** :

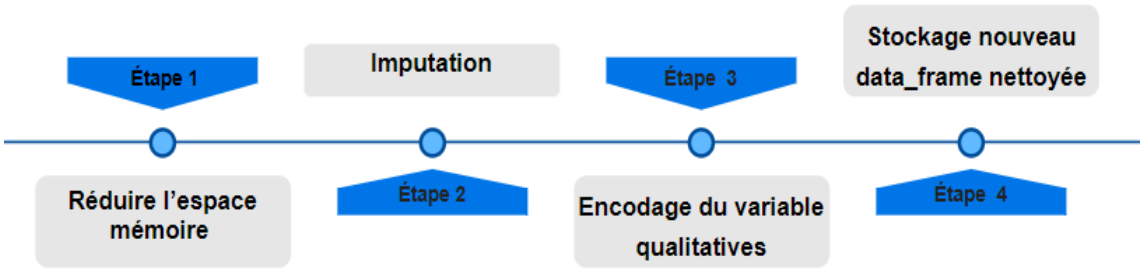

FIGURE 3.4 – Pré_processing de données .

- **Économiser de l'espace mémoire après importation des données**

  Avec ce jeu de données , le chargement des données était difficile car il est volumineux . Aussi il peut entraîner un dysfonctionnement de certaines machines lors du téléchargement, c'est pour cela nous devions réduire sa taille en convertissant par exemple les types int64 en int 16, etc.

  Avant cette étape la lecture de jeu du données il prend plus de 12Go. Le tableau suivant Clarifie avant/après la réduction de mémoire :

	Avant	Après
Train_set	+ 5.5 Go	1.7 Go
test_set	+ 4.8 Go	1.5 Go

TABLE 3.2 – Espace mémoire avant/après la réduction .

- **Imputation (élimination des valeurs manquantes ou aberrantes)**

  Nous examinons le nombre et le pourcentage des valeurs manquantes dans chaque colonne , et nous remarquons plus de 44 colonnes qui ont des valeurs manquantes ,entre eux il y'a environ de 7 colonnes ont plus de 50% de leur données manquantes. Nous déterminons un seuil **na_rate_threshold = 0.9** pour supprimer les Valeurs NAN qui dépassent ce taux aussi un autre seuil pour supprimer les colonnes avec des caractéristiques déséquilibrées à leurs valeurs . à la fin nous Remplissons les valeurs NAN avec le mode statistique.

  Le schéma suivante montre tout ce que nous avons fait dans cette étape :

- **L'Encodage des variables qualitatives**

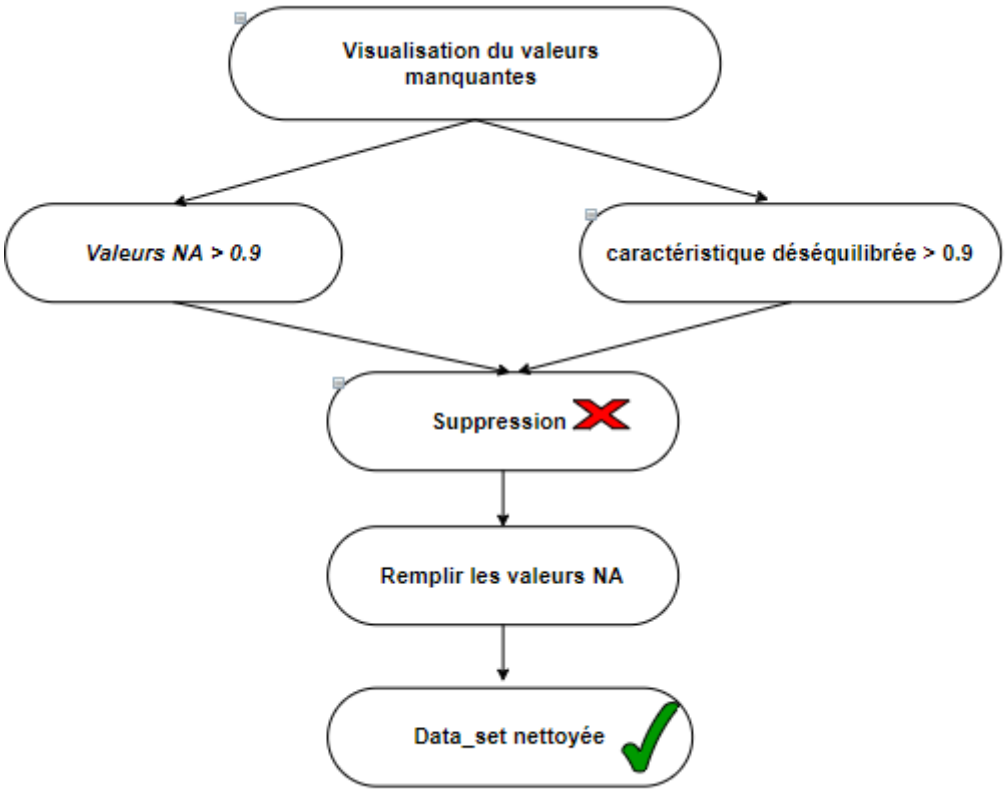

FIGURE 3.5 – Imputation et traitement des valeurs manquantes.

Avant d'aller plus loin, nous devons traiter des variables catégoriques . Un modèle d'apprentissage automatique ne peut malheureusement pas traiter les variables catégorielles (à l'exception de certains modèles tels que LightGBM mais il nécessite plus de 16 go Ram). Par conséquent, nous devons trouver un moyen de coder (représenter) ces variables sous forme de nombres avant de les transférer au modèle. Il y a deux manières principales de mener à bien ce processus :

1.LabelEncoder : attribue chaque catégorie unique dans une variable catégorielle avec un entier.

2.One-hot encoding : crée une nouvelle colonne pour chaque catégorie unique dans une variable catégorielle.

Notre approche d'encodage

Nous utiliserons la politique suivante : pour toute variable catégorielle ayant plus de 2 catégories uniques nous utiliserons le codage d'étiquette (label encoding).

- **Sauvegarder le jeu de données pré_traité**

Après les trois étapes précédent , notre jeu de données est est prêt à être passé à la phase de modélisation. Par conséquent, nous l'avions stocké dans un nouveau dataset nettoyé.

3.7 Approche et solution proposée

Notre solution consiste à construire un **Classifieur XGBoost** sur l'ensemble de données **Microsoft malware prediction** et d'obtenir des probabilités du prédiction ***"prédiction des occurrences de logiciels malveillants "***.En utilisant l'attribut cible **HasDetections** (0 non / 1 oui) pour distignuer les machines infecté parmi les machines bienveillant .Notre modèle construit sur un nombre limité des features (Méthode intégrée (Embedded methods)).Pour les méthodes intégrées combinent les types de la sélection **"wrapper"** et **"filter"**,ils sont implémentés par des algorithmes d'apprentissage par exemple **random forest** qui ont leurs propres méthodes de sélection des attributs intégrées. Nous utilisons cette algorithme car elle est caractérisées par leur robustesse et leur bonne précision.

Le but de notre approche est ce qui suit :

- **La prédiction :** grâce à cette étape ,il est possible de prendre des mesures et pré_protection pour éviter les dommages ou moins mieux s'y prépares .

La figure **3.6** illustre la structure du notre proposition comprend ces phases :

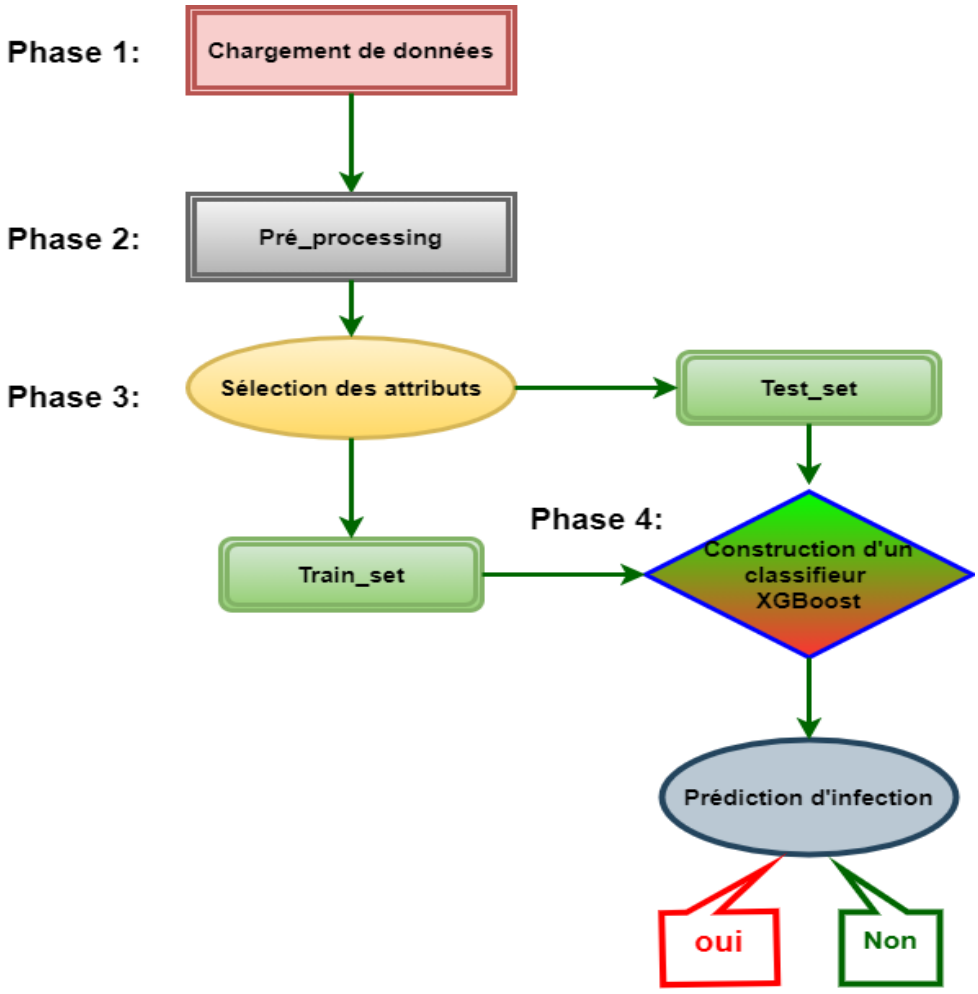

FIGURE 3.6 – Architecture globale de la solution.

- **Phase 1 :** cette phase permet de charge notre jeu de donnée .
- **Phase 2 :** l'étape de pré_traitement est discuté dans la section **3.6** .
- **Phase 3 :** La sélection des attributs est une étape importante pour notre recherche.Elle permet de filtrer les attributs non pertinents de notre jeu de données.Pour notre approche nous choisissons la sélection des attributs à l'aide des forêts aléatoires. Nous sélectionnons les attributs de l'ensemble d'entraînement , puis nous transferons les modifications à l'ensemble de test ceci pour éviter l'over_fitting.
  La figure **3.7** représente les attributs sélectionnes :

La sélection à l'aide d'une forêt aléatoire

Random forest se compose de 4 à 120 arbres décision , chacun d'eux étant construit sur une extraction aléatoire des observations de l'ensemble de données et une extraction aléatoire des attributs, tous les arbres ne voient pas touts les attri-

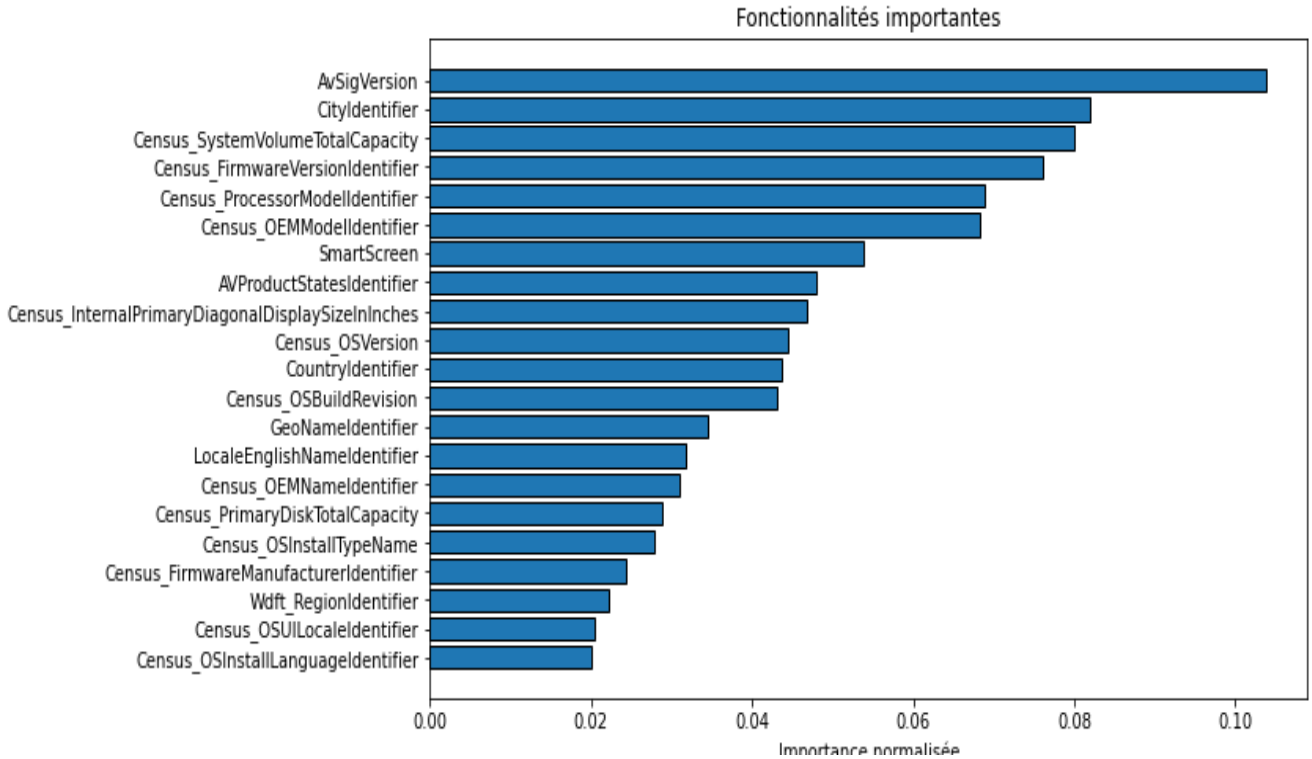

FIGURE 3.7 – Attributs sélectionnes.

buts ou tous les observations ce qui garantit que les arbres sont dé-corrélés et donc moins enclin à l'over_fitting.Pour donner une meilleure intuition, les attributs sélectionnés au sommet des arbres sont en général plus importantes que les attributs sélectionnées aux nœuds d'extrémité des arbres, car généralement les divisions supérieures conduisent à des gains d'informations plus importants[56].

- **Phase 4 :**cette phase correspond à la réalisation de notre approche qu'elle illustrée dans la section suivante.

3.7.1 Création de notre modèle

Dans cette partie , nous allons aborder la création de notre modèle et son implémentation. D'abord,avant de construire le modèle nous devons redéfinir et expliquer le fonctionnement de **XGBoost** puis fournir ses hyperparameters que nous avons utiliser.

Algorithme XGBoost

XGBoost est l'acronyme pour eXtreme Gradient Boosting.XGBoost est une implémentation open-source populaire et efficace de gradient boosting. ce dernier est un algorithme d'ap-

prentissage supervisé qui tente de prédire avec précision une variable cible en combinant un ensemble d'estimations à partir d'un ensemble de modèles plus simples et plus faibles. L'algorithme XGBoost fonctionne bien dans les compétitions d'apprentissage automatique en raison de sa gestion robuste d'une variété de types de données, de relations, de distributions et de la variété de hyperparamètres que vous pouvez affiner [57].

Il a sa propre bibliothèque également appelée XGBoost :

- XGBoost est une bibliothèque optimisée de gradient boosting distribuée conçue pour être très efficace, flexible et portable. Il implémente des algorithmes d'apprentissage automatique dans le cadre de Gradient Boosting. XGBoost fournit un boost d'arborescence parallèle (également connu sous le nom de GBDT, GBM) qui résout de nombreux problèmes de science des données de manière rapide et précise[58].

Raison du choix

Nous choisissons cette algorithme à cause de :

1. Nous disposant d'un grand nombre d'échantillons d'entraînement .
2. La diversité des hyper_paramètres (XGBoost propose un panel d'hyperparamètres très important) , permet d'avoir un contrôle total sur l'implémentation du Gradient Boosting .
3. XGBoost est souvent l'algorithme gagnant des compétitions Kaggle. C'est un algorithme hautement flexible, rapide et polyvalent qui peut fonctionner à travers la classification et d'autres problèmes.
4. La vitesse d'exécution et la performance de modèle par rapport d'autres algorithmes :

XGBOOST	Algorithme optimisé de gradient boosting grâce à la régular_ isation de l'élagage des arbres à traitement parallèle évitant le over-fitting/ biais
Random Forest	Le nombre d'observations de l'ensemble d'entraînement n'est pas infini . les "base learners" auront trop peu de données et de mauvaises performances.
Gradient Boosting Machine	- Pas de sortie anticipée. - Entraînement plus lent . - Moins précis.

TABLE 3.3 – Comparaison de XGBOOST avec d'autres algorithmes.

En divisant notre approche en deux, partie modèle de base, et partie tuning :

Modèle de base

Dans notre modèle de base , nous utilisons les paramètres de l'algorithme étant fixé avant la phase d'apprentissage .
Les paramètres sont divisés en trois catégoriques :

1. **Paramètres généraux :**
   - **silent :** le mode silencieux est activé est réglé sur 1, c'est-à- dire qu'aucun message en cours ne sera imprimé. Il est généralement bon de le garder à 0 car les messages peuvent aider à comprendre le modèle [53] .
2. **Paramètres Booster :**
   - **learning_rate :** le taux d'apprentissage c'est une méthode supplémentaire pour éviter le sur_apprentissage . Plus le taux d'apprentissage est bas, plus le modèle sera robuste [53] .
   - **max_depth :** fait référence à la profondeur d'un arbre. Il définit le nombre maximum de nœuds pouvant exister entre la racine et la feuille la plus éloignée[53]. il est utilisé pour contrôler le sur-apprentissage car une profondeur plus élevée permettra au modèle d'apprendre des relations très spécifiques à un échantillon particulier.Les valeurs typiques sont entre 1 et 30 [52].
   - **colsample_bytree :**représente les caractéristiques à prendre en compte à chaque

construction d'une arbre , et donc cela se produit une fois pour chaque arbre construit.C'est une techniques pour éviter le sur_ajustement et pour améliorer la vitesse de calcul [53].Les valeurs typiques sont entre 0.1 et 1 .

- **subsample :** pourcentage d'échantillons utilisés par arbre. Une valeur faible peut entraîner un sous-apprentissage [54].
- **min_child_weight :**définit la somme minimale des poids de toutes les observations requises chez un enfant. Ceci est similaire à **min_ child_leaf** dans GBM mais pas exactement. Cela fait référence à la «somme des poids» minimale des observations tandis que le GBM a un «nombre d'observations» minimal. Utilisé pour contrôler le sur-apprentissage.

  Des valeurs plus élevées empêchent un modèle d'apprendre des relations qui pourraient être très spécifiques à l'échantillon particulier sélectionné pour un arbre.
- **gamma :** si la valeur est définie sur 0, cela signifie qu'il n'y a pas de contrainte. S'il est défini sur une valeur positive, cela peut aider à rendre l'étape de mise à jour plus conservatrice[52].

3. **Paramètres de tâche d'apprentissage :**
   - **objective :**détermine la fonction de perte à utiliser comme [54] :
     (a) **reg :linear :**pour les problèmes de régression.
     (b) **reg :logistic :** logistique pour les problèmes de classification avec une seule décision.
     (c) **binary :logistic :**logistique pour les problèmes de classification avec probabilité.
   - **eval_metric :** La métrique à utiliser pour les données de validation[52].Il existe plusieurs métriques .

En outre, il existe trois autres paramètres [55] :

1. **num_boost_round :**nombre d'itérations boostantes.Une valeur trop faible entraîne un sous-apprentissage, qui sont des prédictions inexactes sur les données d'entraînement et les données de test. Une valeur trop élevée entraîne un sur_apprentissage, c'est-à-dire des prédictions précises sur les données d'entraînement, mais des prédictions inexactes sur les données de test (ce qui nous importe). Les valeurs typiques vont de 100 à 10000.

2. **verbose_eval :**si verbose_eval est un entier, alors la métrique d'évaluation sur l'ensemble de validation est affichée à chaque étape de boosting verbose_eval donnée. La dernière étape de boosting / l'étape de boosting trouvée en utilisant early_stopping_rounds est également imprimée. Exemple : avec verbose_eval = 4 et au moins un élément dans evals, une métrique d'évaluation est imprimée toutes les 4 étapes de boost, au lieu de chaque étape de boost.
3. **early_stopping_rounds :**l'argument early_stopping_rounds offre un moyen de trouver automatiquement la valeur idéale. L'arrêt précoce entraîne l'arrêt de l'itération du modèle lorsque le score de validation cesse de s'améliorer, même si nous ne sommes pas à l'arrêt dur pour num_boost_round . Il est judicieux de définir une valeur élevée pour num_boost_round , puis d'utiliser early_stopping_rounds pour trouver le moment optimal pour arrêter l'itération.

Le tableau **3.4** représente les valeurs de nos paramètres et comment ont été choisis dans cette partie :

Silent = 0	Pour afficher les itérations d'entraînement parce qu'elles peuvent aider à comprendre notre modèle.
learning rate=0.03	Si sa valeur est faible, cela conduit à accroître le nombre d'arbres mais entraîne généralement une amélioration de la qualité de prévision.
max depth = 8	Notre profondeur est petit pour un simple modèle car plus cette profondeur est grand, plus complexe est le modèle.
colsample bytree =0.2	Pour éviter l'over_fitting avant le tuning .
subsample = 0.7	Plus le pourcentage des observations prises aléatoirement à chaque instance est élevé, plus l'observation est précise .
min child weight = 4	Valeur faible pour ne pas empêcher notre modèle d'apprendre des relations qui pourraient être très spécifiques à l'échantillon particulier sélectionné pour un arbre.
gamma =0.2	Ne prendre pas la valeur par défaut qui est de 0 (pas de régularisation).

'objective' :'binary :logistic'	Car notre problème est de classification avec probabilité .
'eval_metric' :'auc'	Nous choisissons la courbe Roc comme un indice d'évaluation de notre modèle de prédiction .
num boost round = 4000	Valeur recommandée pour éviter le under_fitting et over_fitting.
verbose eval =100	Afin de réduire le nombre d'impressions volumineuses (c_à_d l'AUC est imprimée tous les **100** étapes de boost au lieu de chaque étape de boost.

TABLE 3.4 – Hyper_paramètres choisis pour notre modèle .

2.Optimisation de notre modèle (tuning)

XGBoost Hyperparameter Tuning / Hyperparameter Optimization : c'est à dire modifier, simuler, et choisir les meilleurs hyperparamètres de l'algorithme ,ces valeurs peuvent être optimisées permettant d'augmenter le score du notre modèle.

La technique choisie est la méthode de ***Grid Search*** qui va nous permettre de tester toutes les combinaisons possibles de paramètres et de comparer les performances pour en déduire le meilleur paramétrage.

Le tableau suivant représente les valeurs qui ont été attribuées :

	Ensemble tester	**Valeur optimisée**
max_depth	[3, 5, 7, 9, 11,13]	13
min_child_weight	[1, 3, 5, 7, 9,11]	9
subsample	[0.4, 0.6, 0.8, 1]	0.8
colsample_bytree	[0.2, 0.4, 0.6, 0.8]	0.4

TABLE 3.5 – Hyperparameter Optimization .

3.8 Conclusion

Dans ce chapitre , nous avons présenté des travaux similaires pour la détection des malwares puis nous avons déterminé notre problématique et nous définissons le jeu de données utilisé .Aussi , nous avons présenté notre processus du pré_traitement .En fin, nous avons présenté notre proposition pour résoudre le problème de prédiction des malwares pour une machine Windows .

Dans le chapitre suivant, nous discuterons les résultats obtenus .

Chapitre 4

Résultat et Discussion

4.1 Introduction

Après avoir décrit notre approche dans le chapitre précédent, dans cette partie nous allons parlé d'une manière générale sur les différents moyens techniques (langages, bibliothèques et environnements) utilisés pour implémenter notre solution.En suite nous ajoutons le critère choisi pour évaluer la performance de notre modèle **(courbe Roc)** . Puis , nous présentons également une architecture détaillée de notre implémentation .

Enfin, nous allons discuter sur les résultats obtenus.

4.2 Environnement de développement

4.2.1 Langage du développement

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

La dernière version de Python est la version 3. Plus précisément, la version 3.7 a été publiée en juin 2018. La version 2 de Python est désormais obsolète et cessera d'être maintenue après le 1er janvier 2020. Dans la mesure du possible évitez de l'utiliser.

La Python Software Foundation 1 est l'association qui organise le développement de Python et anime la communauté de développeurs et d'utilisateurs. Ce langage de programmation présente de nombreuses caractéristiques intéressantes[59] :

1. Il est multiplateforme. C'est-à-dire qu'il fonctionne sur de nombreux systèmes d'exploitation : Windows, Mac OS X, Linux, Android, iOS, depuis les mini-ordinateurs Raspberry Pi jusqu'aux supercalculateurs.
2. Il est gratuit. Vous pouvez l'installer sur autant d'ordinateurs que vous voulez (même sur votre téléphone !).
3. C'est un langage de haut niveau. Il demande relativement peu de connaissance sur le fonctionnement d'un ordinateur pour être utilisé.
4. C'est un langage interprété. Un script Python n'a pas besoin d'être compilé pour être exécuté, contrairement à des langages comme le C ou le C++.
5. Il est orienté objet. C'est-à-dire qu'il est possible de concevoir en Python des entités qui miment celles du monde réel (une cellule, une protéine, un atome, etc.) avec un certain nombre de règles de fonctionnement et d'interactions.
6. Enfin, il est très utilisé en bioinformatique et plus généralement en analyse de données. Toutes ces caractéristiques font que Python est désormais enseigné dans de nombreuses formations, depuis l'enseignement secondaire jusqu'à l'enseignement supérieur.

4.2.2 Bibliothèque utiliser

Matplotlib

Matplotlib est une bibliothèque de traçage disponible pour le langage de programmation Python utilisé pour créer des visualisations statiques, animées et interactives[60].

Numpy

NumPy (Numerical Python) est la bibliothèque la plus populaire de calcul scientifique en Python qui permet d'effectuer les calculs scientifiques. Elle propose des fonctions mathématiques complètes, des générateurs de nombres aléatoires, des routines d'algèbre linéaire, des transformées de Fourier, etc [61]

Pandas

pandas est une bibliothèque open source sous licence BSD fournissant des structures de données et des outils d'analyse de données hautes performances et faciles à utiliser

pour le langage de programmation Python [62].

Seaborn

Est une bibliothèque pour créer des graphiques statistiques en Python. Il s'appuie sur matplotlib et s'intègre étroitement aux structures de données des pandas. Il fournit une interface de haut niveau pour dessiner des graphiques statistiques attractifs et informatif[63].

Scikit-learn

Scikit-learn est une bibliothèque libre en Python destinée à l'apprentissage automatique. Le grand avantage de scikit-learn est sa courbe d'apprentissage rapide, aussi elle comprend des fonctions pour estimer des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support. Elle construit sur NumPy, SciPy et matplotlib [64].

4.2.3 Plateforme et environnement de développement

Google Colab [1] ou Colaboratory est un service cloud, offert par Google (gratuit), basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur [65].

4.2.3.1 Jupyter Notebook Jupyter Notebook est une application Web Open Source permettant de créer et de partager des documents contenant du code (exécutable directement dans le document), des équations, des images et du texte. Avec cette application il est possible de faire du traitement de données, de la modélisation statistique, de la visualisation de données, du Machine Learning etc... Elle est disponible par défaut dans la distribution Anaconda [65].

1. https ://colab.research.google.com

4.2.4 Métrique de performance Roc curve

La courbe ROC (Receiving Operating Characteristics) qui signifie la fonction d'efficacité de l'observateur , c'est l'une des outils de visualisation d'organisation et de sélection de classifieurs basé sur leur performance c'est-à-dire leur capacité à départager les différents types d'instances.

Initialement, cette représentation a été utilisée dans le domaine du traitement du signal afin de déterminer le seuil permettant de séparer le signal du bruit. Plus tard cette représentation a été largement étendue au domaine de l'apprentissage automatique .

Une courbe ROC est une représentation graphique de la sensibilité (sensitivity), c'est-à-dire la capacité de détecter des instances positives, par rapport à 1 - spécificité, c'est-à-dire 1 - la capacité de détecter des instances négatives, pour un classifieur binaire.Il indique dans quelle mesure le modèle est capable de distinguer les classes.Plus l'AUC est élevé, mieux le modèle est prédire les négatives comme négative et les positives comme positive [67].

4.3 Architecture de l'implémentation

Grâce à notre étude approfondie nous avons crée plusieurs modèles de détection qui appartiennent au même catégorie (méthodes ensemblistes),m ais notre approche est basée sur le classifieur **XGBOOST**.

Nous avons divisé notre implémentation en deux parties :

1. Modèle de base : C'est le modèle que nous avons expliqué dans le chapitre précédent avec les hyper_paramètres que nous avons choisis .
2. Modèle optimisé :Il est basé sur la même architecture que le modèle de base, il ne diffère que par les hyper_paramètres définis par la méthode de ***Grid Search***.

L'architecture des deux modèles est représentée dans la figure **4.1** suivante :

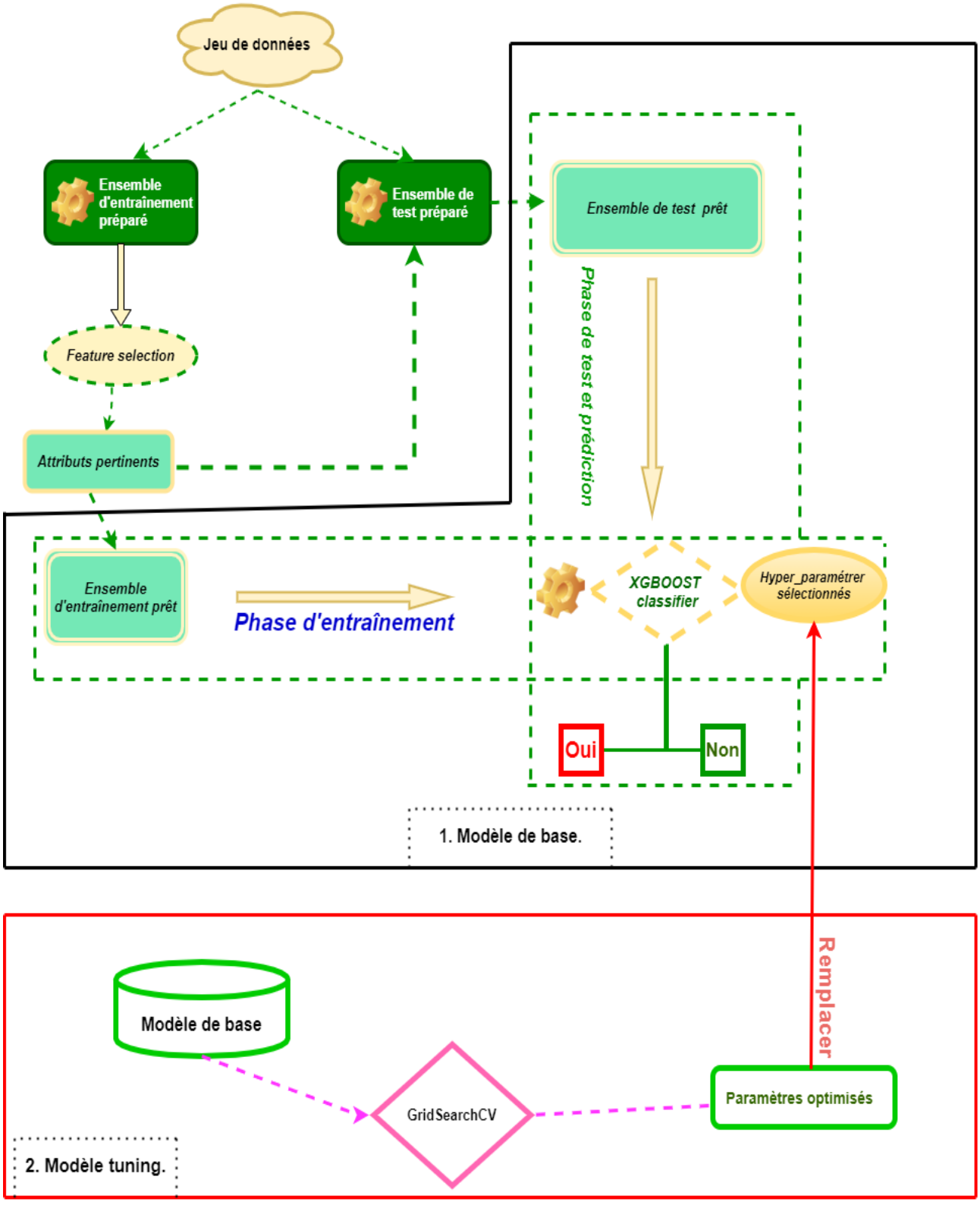

FIGURE 4.1 – Architecture du modèle d'implémentation.

La réalisation du modèle de base passe par quatre phases :

1. **Phase d'entraînement et de validation :** nous choisissons notre dictionnaire des hyper_paramètres , puis nous définissons notre classiffieur qui fonctionne séquentiellement c'est à dire classiffieur fort basé sur des classifieurs faibles . Faible et forte désignent dans notre cas , une mesure de la correlation entre les learners et le target ***HasDetections***.

   Les erreurs de modèle précédent sont corrigées par le prédicateur suivant . Dans notre implémentation , nous avons intégré la validation croisé pour décrémenter l'over_fitting ainsi d'obtenir une estimation plus robuste.

2. **Phase de test et de prédiction :** nous avons utilisé notre ensemble de test prêt pour évaluer et former une prédiction du modèle d'entraînement .

   Dans notre cas , nous avons utilisé les deux fonctions ***predict / predict_proba*** pour donner les probabilités de prédiction .Où que si la valeur de prédiction est supérieur à 0.5 signifie que sa classe est 1 (malveillant), si non elle est 0 (bénigne).

	MachineIdentifier	HasDetections
0	0000010489e3af074adeac69c53e555e	0.593659
1	00000176ac758d54827acd545b6315a5	0.634720
2	0000019dcefc128c2d4387c1273dae1d	0.489733
3	0000055553dc51b1295785415f1a224d	0.331173
4	00000574cefffeca83ec8adf9285b2bf	0.573225

FIGURE 4.2 – Résultats de prédiction.

4.4 Résultats expérimentaux obtenus et discussion

4.4.1 Courbes ROC pour notre modèle

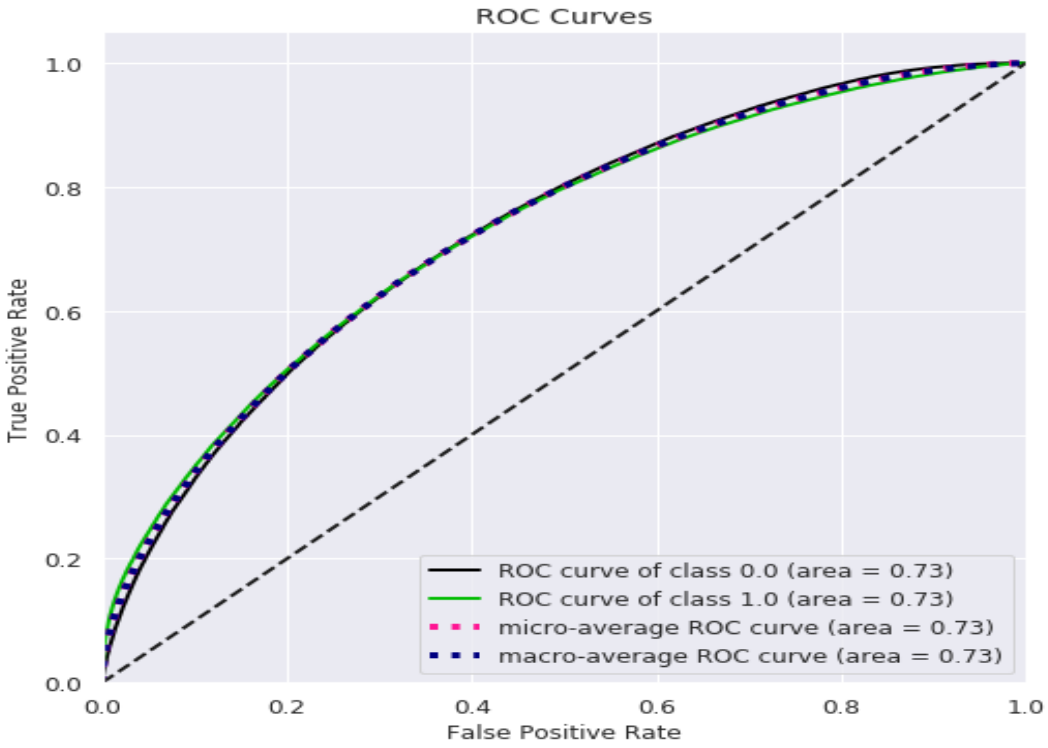

FIGURE 4.3 – Courbes ROC pour notre modèle.

4.4.1.1 Discussion

D'après la figure **4.3** qui représente le taux de vrai positif(sensibilité) en fonction du taux faux positif(1-spécificité). ces taux sont calculées à partir de la matrice de confusion où :

$$\text{FPR} = \text{1- spécificité}$$

Nous remarquons que nous avons trois courbes :

1. La courbe noire pointillée : correspond à des choix aléatoires dans la probabilité =50% , c'est à dire le modèle n'a aucune capacité de séparation entre la classe 1 et la classe 0.
2. La courbe noire (Ligne continue) : représente l'aire sous la courbe ROC de la classe **0** tandis que aire = 0.73.

3. La courbe verte : représente l'aire sous la courbe ROC de la classe **1** tandis que aire = 0.73.

Nous remarquons que les deux courbes de classe positive et négative sont situées au dessus de la ligne pointillée.Nous pouvons pas savoir laquelle fonctionne le mieux car quand :

1. FPR inférieur à 0.15 : la classe positive est supérieure .
2. FPR entre [0.15 - 0.38] : nous observons que les deux classes chevauchant et dans ce cas la prédiction ce fait de façon aléatoire.
3. FPR supérieur à 0.38 : la classe négative est supérieure.

afin d'indiqué la qualité de notre modèle nous devons calculer AUC.

Nous avons notre aire entre [0.7_ 0.8]et ça veut dire que le fonctionnement de notre modèle est bon et qu'il peut faire des prédictions au future sur des nouvelles données.

4.4.2 Courbes Précision / Rappel

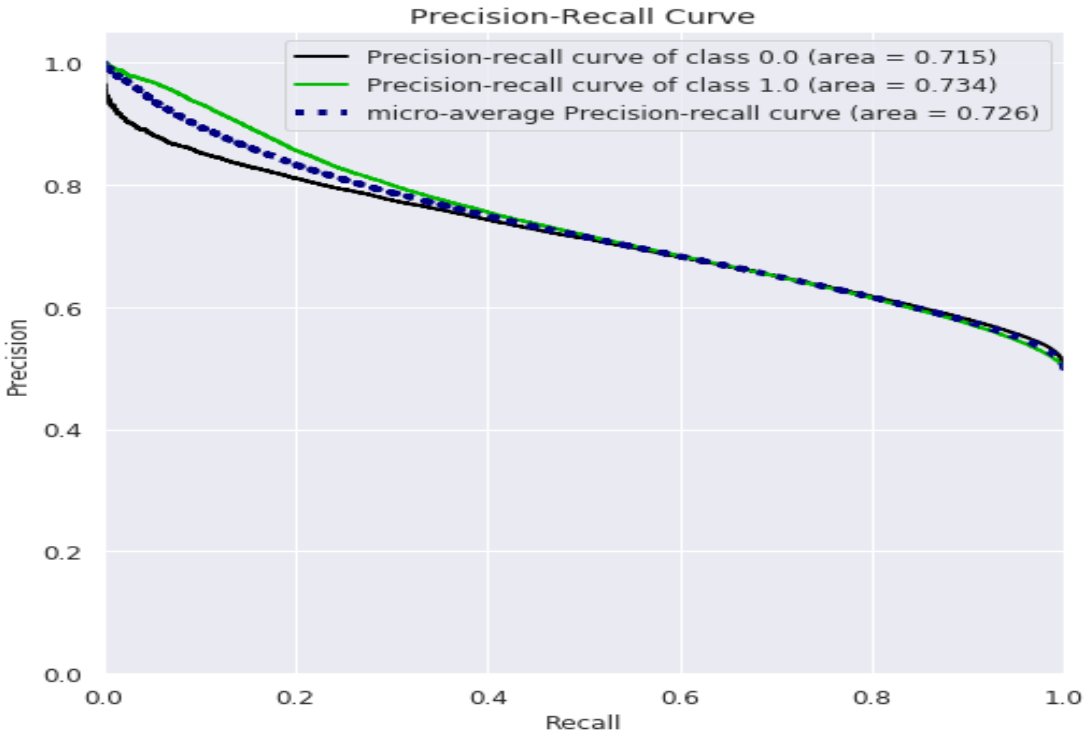

FIGURE 4.4 – Courbes Précision / Rappel .

4.4.2.1 Discussion

Dans cette partie , nous représentons le graphe précision par rapport au rappel qui est l'un des mesures de performances utile pour les problèmes de classification . Cette courbe visualise tous les seuils possibles de notre classiffieur.

D'après la figure **4.4** nous remarquons que :

— Plus la valeur de précision est élevée et la valeur de rappel est faible plus les données de prédiction de classe 1 mieux.

— Plus la valeur de précision est faible et la valeur de rappel est élevée plus les données de prédiction de classe 0 mieux.

4.4.3 Discussion des résultats obtenus

4.4.3.1 Modèle de base

Le tableau ci dessus résume les différents résultats obtenus après l'entraînement de notre modèle de base , qui a été divisé en 5 Flod. Il entraîne sur les quatre premiers

Kfold	**AUC**	**Précision**	**Rappel**
fold 0	72.94%	66.10%	66.80%
fold 1	72.76%	66.00%	66.50%
fold 2	72.79%	66.00%	66.50%
fold 3	72.55%	66.20%	66.85%
fold 4	72.78%	65.50%	66.60%
Mean kflod	72.83%	66.16%	66.63%

TABLE 4.1 – Résultats obtenus pour notre modèle de base.

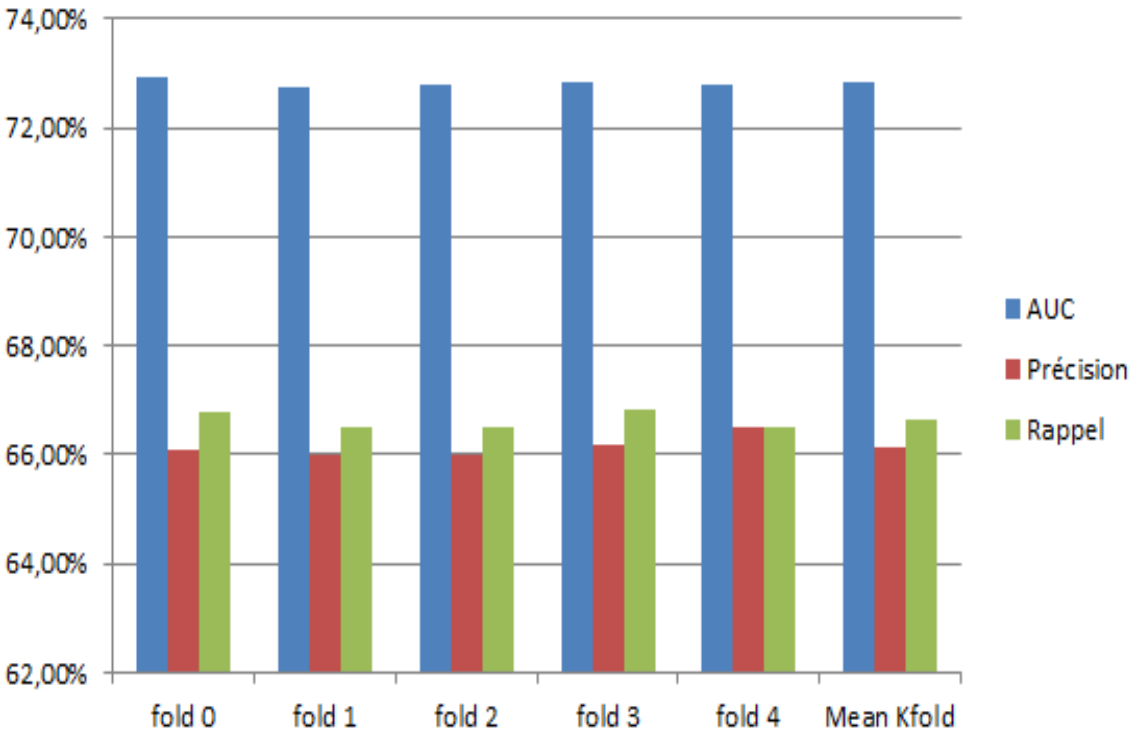

FIGURE 4.5 – Résultat obtenu pour notre modèle de base .

parties puis le validé sur la 5 ème partie . À la fin , il fera la moyenne des scores que nous avons obtenu .

Nous remarquons que les scores(AUC , Rappel , Précision) sont variés dans chacun de ces flods.

Kfold	**AUC**	**Précision**	**Rappel**
fold 0	73.30%	67.00%	66.70%
fold 1	73.14%	66.50%	65.50%
fold 2	73.20%	66.80%	66.80%
fold 3	73.23%	66.85%	66.85%
fold 4	73.22%	66.75%	66.76%
Mean kflod	73.22%	66.78%	66.72%

TABLE 4.2 – Résultats obtenus après l'optimisation des hyper_paramètres.

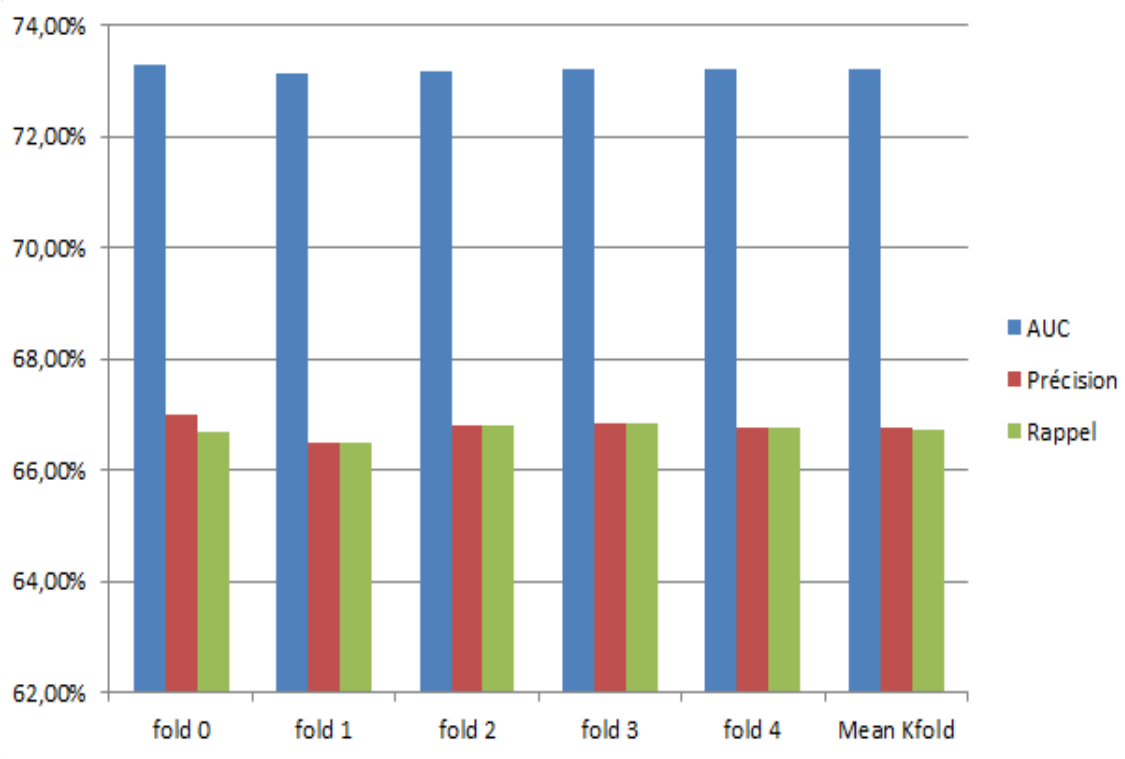

FIGURE 4.6 – Résultats obtenus après l'optimisation des hyper_paramètres

4.4.3.2 Modèle optimisé

D'après la figure et le tableau ci dessus qui nous montrent les différents résultats obtenus après la phase d'optimisation des hyper_paramètres .Nous observons que les résultats(AUC,Précision,Rappel) sont augmentées avec un pourcentage de 1%. Malgré que cette amélioration semble petite mais il garantit le bon fonctionnement de notre prédiction aussi cela peut être un raison pour gagner l'une des compétitions .

4.5 Comparaison des résultats

	Modèle de base	Modèle 1	Modèle 2
max depth	8	13	8
subsample	0.7	0.8	0.7
min child weight	4	9	4
learning rate	0.03	0.03	0.03
colsample bytree	0.2	0.4	0.2
n_estimators	4000	4000	4000

TABLE 4.3 – Hyper_paramètres pour les quatre modèles.

	Modèle de base	Modèle 1	Modèle 2
AUC	72.83%	73.22%	70.5%
Précision	66.16%	66.78%	64.6%
Rappel	66.63%	66.72%	64.4%
Temps d'exécution	1h 2min 14s	59min 6s	1h 21min 42s

TABLE 4.4 – Tableau de comparaison des résultats "XGBoost".

Les deux tableaux **4.3** et **4.4** montrent une comparaison entre les trois algorithmes qui sont :

- Modèle de base XGBOOST qui a été réaliser à l'aide de scikit-learn.
- Modèle 1 représente notre modèle de base optimisé.
- Modèle 2 c'est le modèle XGBOOST qui a été réaliser sans scikit-learn (xgb.train)

.

Le tableau **4.3** représente les divers paramètres primordiale lors de la construction de ces trois modèles .

Où le tableau **4.4** illustre les résultats obtenus en fonction des ces hyper_paramètres , tel que chaque valeur peut être affecté par des effets négatifs ou positifs sur la performance . A la fin , nous choisissons le **modèle 1** comme le modèle qui fonctionne le mieux pour notre ensemble de données avec un AUC égale à 73.22%.

	Modèle 1	LightGBM	Random Forest
AUC	73.22%	72.75%	63.5%
Précision	66.78%	66.26%	64.60%
Rappel	66.72%	65.80%	64.40%
Temps d'exécution	59min 6s	30min 4s	6h 30min 42s

TABLE 4.5 – Tableau de comparaison des résultats .

Le tableau **4.5** montre les résultats obtenus pour chaque modèle (AUC Précision , Rappel) ainsi que leur temps d'exécution .Nous remarquons que **le modèle 1** avait un **AUC** plus élevé par rapport aux deux autres classifieurs (LightGBM , Random Forest). Malgré LightGBM est plus rapide que XGBoost mais dans notre étude nous intéressons à la valeur de **AUC** .

Alors nous concluons que le modèle optimisé avait une prédiction supérieur aux autres modèles.

4.6 Conclusion

Nous avons présenté dans ce dernier chapitre les différents langages et les outils de développement utilisés afin d'implémenter notre solution. Après, nous avons cité les indicateurs de performances de classification. Ensuite nous avons décrit l'architecture de notre modèle de façon générale. Finalement nous avons effectué une comparaison entre notre modèle et d'autres modèles, à fin de discuter sur les résultats obtenus dans ce travail.

Conclusion générale

Au cours de ce mémoire, nous avons présenté les différentes étapes de la réalisation de notre modèle de prédiction (détection)à l'aide de méthode ensembliste (apprentissage supervisé) .

La capacité massive de notre ensemble de données nous a incités à utiliser l'une des méthodes de sélection des attributs , dans notre cas nous avons utiliser random forest pour sélectionner les meilleurs attributs pour gagner le temps d'entraînement et obtenir des meilleurs résultats.

Pour arriver à notre objectif, nous avons utilisé un algorithme qui effectué une classification des machines windows (infecté ou non) sur un jeu de donnée volumineux avec une large panel d'attributs qui collecte toutes les configurations d'une machine windows.Pour la réalisation de notre modèle de base nous avons utilisé l'algorithme xgboost de gradient boosting avec un choix précise des hyper_parameters puis nous avons sélectionné les meilleurs paramètres à l'aide de la fonction GridSearch à fin d'obtenir des bons résultats par rapport au premier.

Perspectives :

Comme perspectives nous pouvons citer : nous essaierons d'appliquer une autre machine apprentissage des algorithmes et améliorer les scores. Nous procéderons également au réglage des hyper_paramètres pour obtenir de meilleurs résultats.

En dehors de cela, nous verrions quelle différence serait faite si l'itération augmentait

en termes d'augmentation de la précision.

Bibliographie

[1] AMAD Mourad.*Sécurité des Systémes Informatiques et de Communications. Principes et Concepts Fondamentaux de la Sécurité des Systéemes Informatiques.*2017, pages 3-12-13 ,Université Bouira.

[2] Fernand Lone Sang.*Protection des systèmes informatiques contre les attaques par entrées-sorties*. PhD thesis, Toulouse, INSA, 2012.

[3] Ravindar Reddy Ravula. *Classification of malware using reverse engineering and data mining techniques.* PhD thesis, University of Akron, 2011.

[4] AMAD Mourad.*Sécurité des Systémes Informatiques et de Communications. Sécurité des Programmes.* 2017, pages 3-7-8-11-16 ,Université Bouira.

[5] Mohamed BELAOUED. *Approches Collectives et Coopératives en Sécurité des Systèmes Informatiques.* PhD thesis, Skikda, 2016.

[6] Muhammad Sarfraz. *Developments in Information Security and Cybernetic Wars. IGI Global,* 2019, pages 13.

[7] Éric Filiol.*Les virus informatiques : théorie, pratique et applications.*Springer Science & Business Media, 2009,pages 129.

[8] https ://www.kaspersky.fr/blog/quest-ce-quun-rootkit/750/, consulté le 27/03/2020.

[9] Ed Skoudis and Lenny Zeltser. *Malware : Fighting malicious code.* Prentice Hall Professional, 2004, pages 164.

[10] Kateryna Chumachenko.*Machine learning methods for malware detection and classification.*phD thesis, Kaakkois-Suomen ammattikorkeakoulu, 2017.

[11] Mikko Elisan, Christopher C et Hypponen.*Malware, rootkits & botnets : guide du débutant.*McGraw-Hill New York, 2013, pages 56-57.

[12] https ://usa.kaspersky.com/resource- center/threats/riskware, consulté le 28/03/2020.

[13] Mohit Singhal. *Analyse et catégorisation des logiciels malveillants de téléchargement Drive-by à l'aide de Sandboxing et de l'ensemble de règles Yara.* PhD thesis,Texas, 2019.

[14] Philippe Beaucamps. *Analyse de programmes malveillants par abstraction de comportements.* PhD thesis, france , INPL, 2011.

[15] Zahra Bazrafshan, Hashem Hashemi, Seyed Mehdi Hazrati Fard, and Ali Hamzeh. *A survey on heuristic malware detection techniques.* The 5th Conference on Information and Knowledge Technology,IEEE,113–120 , 2013.

[16] Annie H Toderici and Mark Stamp. *Chi-squared distance and metamorphic virus detection.*Journal of Computer Virology and Hacking Techniques-Springer-, 9(1) :1–14,2013.

[17] Grégoire Jacob, Hervé Debar, and Eric Filiol. *Behavioral detection of malware : from a survey towards an established taxonomy.*Journal in computer Virology-Springer-, 4(3) :251–266, 2008.

[18] Web sécurité : http ://www.vulgarisation-informatique.com/,consulté le 29/03/2020 .

[19] J. LEGRAND . *Virus et Antivirus.*STS Informatique de Gestion.

[20] https ://www.institut-pandore.com/hacking/les-3-virus-informatiques-connus-dangereux/ , consulté le 04/04/2020.

[21] https ://www.popularmechanics.com/ , consulté le 04/04/2020.

[22] https ://www.futura-sciences.com/tech/definitions/securite-wannacry-16434/ , consulté le 04/04/2020.

[23] https ://www.spotlightmetal.com/machine-learning–definition-and-application-examples-a-746226, consulté le 20/02/2020.

[24] Mokhtar TAFFAR. *Initiation à l'apprentissage automatique.*Page 13 , université de Jijel .

[25] Hassan CHOUAIB. *Sélection de caractéristiques : méthodes et applications* .PhD thesis, Paris Descartes University : Paris, France, 2011.

[26] Mohammad IMRAN , Muhammad Tanvir AFZAL, Muhammad Abdul QADIR. *A comparison of feature extraction techniques for malware analysis* . Turkish Journal of Electrical Engineering & Computer Sciences,(2017) 25 : 1173–1183,2017.

[27] Mélanie LABARRE. *Comparaison de méthodes ensemblistes.*Maître en science ,Université de Montréal ,2003.

[28] Efron, B. and Tibshirani . *An introduction to the bootstrap.* Chaman and Hall ,New York,1993.

[29] Rachid MIFDAL .*Application des techniques d'apprentissage automatique pour la prédiction de la tendance des titres financiers.*Université du Québec,MONTRÉAL,2019.

[30] Eve Mathieu-Dupas.*Algorithme des k plus proches voisins pondérés et application en diagnostic.*Hal ,France,2010.

[31] Younès BENNANI .*Traitement Informatique des Données* .page 27,Université Paris 13.

[32] https ://medium.com/@kenzaharifi/bien-comprendre-lalgorithme-des-k-plus-proches-voisins-fonctionnement-et-implémentation-sur-r-et-a66d2d372679,consulté le 24/03/2020.

[33] https ://tadg5.files.wordpress.com/2015/03/classificateur-du-reseau-bayesien-naif.pdf,consulté le 24/03/2020.

[34] https ://le-datascientist.fr/les-algorithmes-de-naives-bayes,consulté le 23/03/2020.

[35] Pr. Fabien Moutarde.*Arbres de décision et Forêts aléatoires* .CAOR, MINES ParisTech, PSL,Paris ,2017 .

[36] https ://www.lemagit.fr/definition/Matrice-de-confusion,consulté le 25/02/2020.

[37] https ://studylibfr.com/doc/2194961/chapitre-4—random-forest-fichier,consulté le 25/03/2020.

[38] Abdoun Sihem ,Belkham Fella , Medjkoune Nawel . *Rapport final - Apprentissage avance Selection attributs pour la prédiction d'une réponse biologique*,2013.

[39] https ://github.com/projeduc/intro_apprentissage_automatique/blob/ master/regression.md ,consulté le 06/04/2020.

[40] https ://mrmint.fr/apprentissage-supervise-machine-learning , consulté le 03/04/2020.

[41] Rimah Amami, *Dorra Ben Ayed, Noureddine Ellouze . Application de la Méthode Adaboost à la Reconnaissance Automatique de la Parole*.e-STA ,Volume 8 ,N 1 ,pp 53-60 ,2011.

[42] https ://www.datacamp.com/community/tutorials/adaboost-classifier-python ,consulté le 04/04/2020.

[43] Kévin Bailly.*Méthodes d'ensembles : Boosting, bagging et random forests*.Page 29, 2004 .

[44] Sébastien Gambs.*Survol de l'apprentissage machine*. Université de Rennes ,France ,2011.

[45] Jennifer PARIENTE.*Modélisation du risque géographique en assurance habitation*.Université Paris Dauphine,2017.

[46] Vincent Gardeux. *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension : application à l'analyse de données de puces à ADN*.PhD thesis, Université de Paris-est Créteil,2012.

[47] BEKADDOURI Nadia , MOHAMMED SEGHIR Nadia .*Modèles d'apprentissage automatique pour l'analyse des sentiments*.Université ABDELHAMID IBN BADIS MOSTAGANEM 2012/2013.

[48] Mohamad Baset . *MACHINE LEARNING FOR MALWARE DETECTION* .Heriot-Watt University.Page 18 ,2016.

[49] Mustafa A.Ali , wesam S.Bhaya. *Review on Malware and Malware Detection Using Data Mining Techniques*.Journal of University of Babylon for Pure and Applied Sciences ,November 2017.

[50] Matthew S Webb . *Evaluation tool based automated malware analysis through persistence mechanism Detection.* B.S., Kansas State University,2018.

[51] Matthew G. Schultz and Eleazar Eskin ,Erez Zadok,Salvatore J. Stolfo. *Data Mining Methods for Detection of New Malicious Executables*. IEEE Xplore ,Conference Paper 10.1109/SECPRI.2001.924286 ,2001.

[52] https ://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/,consulté le 25/06/2020.

[53] https ://towardsdatascience.com/from-zero-to-hero-in-xgboost-tuning-e48b59bfaf58,consulté le 15/07/2020.

[54] https ://www.datacamp.com/community/tutorials/xgboost-in-python, consulté le 20/07/2020.

[55] https ://xgboost.readthedocs.io/en/latest/python/python_api.html, consulté le 02/08/2020.

[56] https ://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f,consulté le 07/08/2020.

[57] https ://xgboost.readthedocs.io/en/latest/ , consulté le 01/10/2020.

[58] Amazon SageMaker Developer Guide , 2020, page 665.

[59] Patrick Fuchs et Pierre Poulain,Cours de Python , Universite Paris Diderot-Paris 7, Paris, France ,Page 8 , 22 juillet 2020.

[60] https ://matplotlib.org/, consulté le 08/08/2020.

[61] https ://numpy.org ,consulté le 08/08/2020.

[62] https ://pandas.pydata.org/docs/ ,consulter 08/08/2020.

[63] https ://seaborn.pydata.org/ ,consulté le 08/08/2020.

[64] https ://scikit-learn.org/,consulté le 08/08/2020.

[65] https ://ledatascientist.com/google-colab-le-guide-ultime/, consulté le 08-08-2020.

[66] Da vid Gadoury . *Distributions d'auto-amorçage exactes ponctuelles des courbes ROC et des courbes de coûts* .Université de Montréal ,PhD thesis ,2009.

[67] https ://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/,consulté le 20/09/2020.

Abstract

Nowadays, malware is a real threat to computer security. Unfortunately, commercial antiviruses are unable to provide complete and perfect protection against its malware variants. To solve this problem many solutions are proposed, among its solutions we have the detection and prediction of malware.

The detection and prediction of computer malware is a major challenge in the security of computer systems.

The objective of our work is the creation of a supervised machine learning model using an XGBoost set method in order to predict the probability that a Windows machine will be infected with malware in the future or not.

Key words : Malware, Detection, Prediction, Malware, Supervised Machine Learning, XGBoost, Windows Machine ...

Résumé

De nos jours, les logiciels malveillants représentent une réelle menace pour le domaine de sécurité informatique. Malheureusement, les antivirus commerciaux sont incapables de fournir une protection complète et parfaite contre ses variantes malwares. Pour résoudre ce problème beaucoup de solutions sont proposées, parmi ses solutions nous avons la détection et la prédiction des **malwares**.

La **détection** et la **prédiction** des malwares informatique est un défi majeur dans la sécurité des systèmes informatique.

L'objectif de notre travail c'est la création d'un modèle d'**apprentissage automatique supervisé** à l'aide d'une méthode ensembliste **XGBoost** afin de prédire la probabilité qu'une **machine Windows** soit infectée par un logiciel malveillant au future ou pas.

Mots clés : Logiciels Malveillants, Détection, Prédiction , Malwares , Apprentissage Automatique Supervisé , XGBoost , Machine Windows. ...