الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية الرياضيات و الإعلام الآلي و الاتصالات السلكية و اللاسلكية

قسم: الإعلام الآلي

Mémoire de Master en informatique

Spécialité : Modélisation Informatique des Connaissances et du Raisonnement

T h è m e

Détection des application Android malveillante

▪ **Présenté par :**

Guerroudj Mouad

Kaddouri Seifeddine

▪ **Dirigé par :**

Mokri Miloud Aboubakeur El Sadek

Année universitaire 2025-2026

REMERCIEMENTS

Avant tout, nous remercions Allah le Tout-Puissant de nous avoir accordé la santé, la volonté et la patience nécessaires pour mener à bien ce travail. Nous tenons à exprimer notre profonde gratitude à notre encadreur, Dr. MOKRI, pour son accompagnement, ses précieux conseils, sa disponibilité et son soutien tout au long de la réalisation de ce mémoire. Ses orientations et son expertise ont été d'une grande importance dans l'aboutissement de ce travail. Nous adressons également nos sincères remerciements aux membres du jury qui ont accepté d'évaluer ce mémoire et de participer à sa soutenance. Nous leur sommes reconnaissants pour le temps consacré à l'examen de notre travail ainsi que pour leurs remarques et suggestions constructives. Nos remerciements s'adressent aussi à l'ensemble des enseignants du département d'Informatique de l'Université Dr. Moulay Tahar de Saïda pour la qualité de la formation qu'ils nous ont dispensée tout au long de notre parcours universitaire. Nous exprimons notre profonde reconnaissance à nos parents et à nos familles pour leurs sacrifices, leur soutien moral et matériel, leurs encouragements permanents et leur confiance inébranlable. Enfin, nous remercions tous nos amis et collègues qui nous ont aidés, de près ou de loin, à réaliser ce travail. À toutes les personnes ayant contribué à l'aboutissement de ce mémoire, nous adressons nos plus sincères remerciements.

DÉDICACES

Je dédie ce modeste travail,

À mes très chers parents,

Aucun mot ne saurait exprimer pleinement ma gratitude, mon amour et mon respect envers vous. Merci pour vos sacrifices, votre patience, vos encouragements constants et votre soutien inconditionnel tout au long de mon parcours. Que Dieu vous accorde santé, bonheur et longue vie.

À mes frères et sœurs,

Pour leur affection, leur présence et leurs encouragements qui m'ont toujours donné la force d'avancer.

À toute ma famille,

Pour leur soutien moral et leurs prières qui ont été une source de motivation permanente.

À mes enseignants,

Qui m'ont transmis le savoir, les valeurs du travail et le goût de la recherche scientifique.

À mes amis et collègues,

Pour les moments partagés, l'entraide et les encouragements qui ont rendu ce parcours plus agréable.

Enfin, je dédie ce travail à toute personne qui a contribué de près ou de loin à ma réussite

Sommaire Générale :

Chapitre 1 : La sécurité informatique 2

- 1.1 Introduction 2
- 1.2 Définition de la sécurité informatique 2
- 1.3 Les objectifs fondamentaux de la sécurité informatique 2
- 1.4 Les principales menaces informatiques 4
- 1.5 Les moyens de protection en sécurité informatique 6
- 1.6 Les services de sécurité 7

Chapitre 2 : Le système Android 09

- 2.1 Historique 12
- 2.2 Présentation du système Android 16

    2.2.1 Introduction .....

    2.2.2 Description .....

    2.3 Open Handset Alliance .....

    2.4 Bugdroïde .....

- 2.5 Architecture Android 19

  2.5.1 Applications .....

  2.5.2 Framework de développement .....

  2.5.3 Le noyau Linux .....

- 2.6 Les permissions dans le système Android 25

    2.6.1 Types de permissions Android .....

- 2.7 La sécurité dans les systèmes Android 27

    2.7.1 Modèle de sécurité Android .....

    2.7.2 Mécanismes de sécurité au niveau de l'application .....

2.7.3 Le modèle des permissions .....

2.7.4 Permissions au niveau des composants .....

2.7.5 Permissions au niveau des processus .....

2.7.6 Permissions sur les fournisseurs de contenu .....

2.7.7 Permissions par URI .....

2.7.8 Composants publics et privés .....

- 2.8 Attaques et vulnérabilités contre le système Android 32
- 2.9 Les applications malveillantes déclarées 33

Chapitre 3 : L'apprentissage automatique et l'apprentissage profond 36

- 3.1 L'apprentissage automatique 39
- 3.2 Types d'algorithmes de Machine Learning 40

3.2.1 Apprentissage supervisé .....

3.2.2 Apprentissage non supervisé .....

3.2.3 Apprentissage par renforcement .....

3.2.4 Apprentissage semi-supervisé .....

- 3.3 Les classificateurs 43

3.3.1 Classificateurs uniques .....

3.3.2 Classificateurs hybrides .....

3.3.3 Classificateurs ensemblistes .....

- 3.4 Deep Learning (Apprentissage profond) 46
- 3.5 Apprentissage automatique vs Apprentissage profond 47

Chapitre 4 : Implémentation et expérimentation 49

- 4.1 Python : Introduction et définition 51
- 4.2 Concernant la détection des applications Android malveillantes 54

4.2.1 Les datasets .....

4.2.2 Les modèles d'apprentissage utilisés .....

4.2.3 Modèle d'apprentissage profond .....

- 4.3 Code final avec méthodes de splitting 72

- 4.4 Output avec résultats finaux 86
- 4.5 Analyse et comparaison 99

Conclusion générale 103

Références bibliographiques 105

Chapitre 01 : La sécurité informatique

Listes des matiéres :

1-introduction.....2

2-définition.....2

3- Les objectifs fondamentaux de la sécurité informatique.....2

4-Les principales menaces informatiques 4

5-Les moyens de protection en sécurité informatique 6

6- les services de sécurité 7

1/Introduction :

Avec l'évolution rapide des technologies numériques et l'expansion d'Internet, les systèmes informatiques occupent aujourd'hui une place centrale dans tous les domaines : entreprises, administrations, banques, hôpitaux et même dans la vie quotidienne des particuliers. Cependant, cette dépendance croissante aux outils informatiques s'accompagne de nombreux risques. Les cyberattaques, les vols de données, les logiciels malveillants et les intrusions sont devenus des menaces fréquentes et dangereuses.

La sécurité informatique est donc devenue un enjeu majeur. Elle regroupe l'ensemble des moyens techniques, organisationnels et humains permettant de protéger les systèmes d'information contre toute forme de menace.**[5.7.10]**

2/Définition de la sécurité informatique :

La sécurité informatique, aussi appelée cybersécurité, désigne l'ensemble des pratiques et des mécanismes mis en place pour garantir la protection des données, des réseaux, des logiciels et des équipements informatiques.

Son objectif principal est d'assurer :

- la confidentialité des informations,
- l'intégrité des données,
- la disponibilité des services,
- l'authenticité des utilisateurs et des échanges.

Ainsi, la sécurité informatique vise à empêcher toute action non autorisée pouvant compromettre le bon fonctionnement d'un système. **[1.2.10]**

3/ Les objectifs fondamentaux de la sécurité informatique :

La sécurité informatique repose sur trois piliers principaux, souvent appelés le modèle **CIA** :

3/ 1- Confidentialité :

La confidentialité consiste à garantir que seules les personnes autorisées peuvent accéder aux informations sensibles.

Exemple : empêcher qu'un hacker puisse consulter les données bancaires d'un utilisateur.

Mécanismes clés pour assurer la confidentialité :

Chiffrement (Cryptographie) : Transformation des données en un format inintelligible, déchiffrable uniquement avec une clé secrète (symétrique ou asymétrique).

Contrôle d'accès : Restriction des accès aux seules personnes nécessaires (principe du moindre privilège).

Authentification : Vérification de l'identité, notamment via l'authentification multifacteur (MFA).

Gestion des permissions : Classification des données et définition de droits d'accès stricts.

Enjeux et menaces :
La confidentialité protège contre l'usurpation d'identité, la vente de données personnelles et les attaques ciblées, des informations volées ayant une grande valeur sur le marché noir. Elle concerne la protection contre des accès non autorisés, qu'ils soient internes ou externes. **[1]**

3/ 2- Intégrité :

L'intégrité assure que les données ne sont pas modifiées ou altérées de manière frauduleuse.
Exemple : éviter la modification d'un dossier médical ou d'un document officiel.

Aspects Clés de l'Intégrité :

Exactitude et Complétude : Les données doivent être précises et complètes, sans erreurs ni corruption.

Intégrité Physique et Logique : La protection contre la détérioration physique (stockage) et la modification logique (utilisations).

Mesures Techniques :

Hachage (Hashing) : Vérification via des signatures numériques ou des empreintes (checksums).

Contrôle d'accès : Limitation des droits de modification.

Chiffrement : Protection des données en transit et au repos.

Signature numérique : Garantie de l'origine et de l'absence de modification. **[1.9]**

3/ 3- Disponibilité :

La disponibilité garantit que les ressources informatiques restent accessibles aux utilisateurs légitimes à tout moment.
Exemple : éviter qu'un site web soit bloqué par une attaque de type déni de service (DDoS).

Principes Clés de la Disponibilité

Accès Fiable : Les utilisateurs autorisés doivent pouvoir accéder aux données de manière rapide et cohérente.

Continuité d'activité : Maintien du fonctionnement des systèmes même face à des attaques (ex: ransomwares) ou des pannes.

Composante de la triade CID : Elle est souvent priorisée car des données confidentielles inutilisables n'ont aucune valeur.

Mesures pour Garantir la Disponibilité

Redondance et Haute disponibilité : Utilisation de serveurs de secours pour prendre le relais en cas de panne.

Sauvegardes : Technique indispensable pour assurer la pérennité et l'accès aux données.

Maintenance préventive : Mise à jour régulière des systèmes et du matériel.

Défense active : Surveillance du trafic pour bloquer les attaques par déni de service (DDoS).

Indicateurs de Mesure

Taux de disponibilité : Souvent exprimé en pourcentage de temps de fonctionnement (ex: 99,99%).

MTTR (Mean Time To Repair) : Temps moyen nécessaire pour réparer un système après une panne.

[1.7]

4/Les principales menaces informatiques :

4/1 -**Concepts Fondamentaux : Risque vs Attaque :**

-**Le Risque Informatique :** Il est défini comme la probabilité qu'une **menace** exploite une **vulnérabilité** (faiblesse technique ou humaine), entraînant un impact négatif sur la confidentialité, l'intégrité ou la disponibilité des données.

-**L'Attaque Informatique (Cyberattaque) :** C'est une action délibérée visant à obtenir un accès non autorisé à un système pour voler, modifier, détruire des données ou paralyser un service.

Les systèmes informatiques peuvent être exposés à plusieurs types de menaces :

4/ 2- Les logiciels malveillants (malwares) :

-Un malware (logiciel malveillant) est un programme conçu pour infecter, endommager ou accéder illicitement à des systèmes informatiques afin de voler des données, extorquer de l'argent ou espionner des utilisateurs. Ces menaces incluent virus, ransomwares, chevaux de Troie et spywares. La protection repose sur des logiciels antivirus, des mises à jour régulières et la vigilance.

-L'un des cas les plus célèbres et les plus fascinants est celui de **Stuxnet**, découvert en 2010. Contrairement aux malwares classiques qui volent des données, Stuxnet était une véritable **arme numérique** conçue pour le sabotage industriel. **[5.7.10]**

4/ 3- Le phishing (hameçonnage) :

-Le phishing est une technique visant à tromper un utilisateur afin qu'il divulgue des informations personnelles (mot de passe, numéro bancaire). Cela se fait généralement via des emails ou des messages frauduleux.

-Un cas de phishing célèbre et marquant est celui de la fraude au président (BEC - Business Email Compromise) qui a ciblé **Google et Facebook entre 2013 et 2015**, orchestrée par le Lituanien Evaldas Rimasauskas. Le fraudeur a usurpé l'identité d'un fabricant asiatique (Quanta Computer) et convaincu des employés de transférer plus de 100 millions USD sur ses comptes. **[5.7.10]**

4/ 4- Les attaques par déni de service (DDoS) :

-Les attaques par déni de service distribué (DDoS) visent à rendre indisponibles des services en ligne, des sites web ou des réseaux en les submergeant d'un volume massif de requêtes provenant de multiples sources (botnets). Contrairement au DoS classique (une source), le DDoS sature la bande passante ou les ressources du serveur, causant pannes ou ralentissements majeurs.

- L'attaque par botnet **Mirai** **en octobre 2016** est l'un des cas de DDoS les plus célèbres, paralysant des sites majeurs (Twitter, Netflix, Reddit) en infectant des appareils IdO (caméras, routeurs). Elle a démontré la puissance des objets connectés non sécurisés, atteignant des volumes records à l'époque. **[5.7.10]**

4/ 5- Les intrusions et piratages :

Un pirate informatique peut exploiter des failles dans un système pour accéder à des informations sensibles ou prendre le contrôle d'un réseau. **[3.4.9]**

5/Les moyens de protection en sécurité informatique :

Pour lutter contre ces menaces, plusieurs solutions existent :

5/ 1- Protection du Réseau et des Accès:

- **Pare-feu (Firewall) :** Agit comme une barrière filtrant le trafic entrant et sortant selon des règles de sécurité précises.
- **VPN (Réseau Privé Virtuel) :** Sécurise les connexions distantes en créant un tunnel chiffré entre l'utilisateur et le réseau de l'entreprise.
- **Authentification Multi-Facteurs (MFA) :** Ajoute une couche de vérification (code SMS, application, empreinte) au-delà du simple mot de passe. **[2.4.6]**

5/ 2- Défense Contre les Menaces Logicielles:

- **Antivirus et EDR :** Les solutions comme Bitdefender ou Microsoft Defender détectent et bloquent les logiciels malveillants (malwares, ransomwares) en analysant les signatures ou les comportements suspects.
- **IDS/IPS (Systèmes de détection/prévention d'intrusion) :** Surveillent les activités suspectes sur le réseau pour alerter ou bloquer automatiquement les attaques en temps réel. **[4.5.10]**

5/ 3- Sécurisation des Données et Systèmes:

- **Chiffrement (Encryption) :** Rend les données illisibles pour toute personne ne possédant pas la clé de déchiffrement, que ce soit pour le stockage ou le transfert.
- **Modèle Zéro Confiance (Zero Trust) :** Principe selon lequel aucun utilisateur ou appareil n'est considéré comme sûr par défaut, exigeant une vérification continue pour chaque accès.
- **Sauvegardes (Backups) :** Mécanisme vital de résilience pour restaurer les données après une attaque ou une panne technique. **[4.6.10]**

5/ 4- Mesures Physiques et Humaines:

- **Sécurité physique :** Verrouillage des serveurs, badges d'accès et protection des ports USB.
- **Sensibilisation :** Formation des utilisateurs pour éviter le **hameçonnage (phishing)** et l'ingénierie sociale, qui restent les principales failles de sécurité.

Tableau 1 : types d'attaque et mécanismes de sécuriité

Type d'Attaque	Description	Mécanisme de Sécurité Approprié
Phishing / Ingénierie Sociale	Tromper l'utilisateur pour voler des identifiants via email/SMS.	**MFA (Authentification Multi-facteurs)** et sensibilisation des utilisateurs.
Ransomware (Rançongiciel)	Chiffrement des données pour exiger une rançon.	**Sauvegardes régulières (3-2-1)** et solutions **EDR** (Endpoint Detection & Response).
Déni de Service (DoS/DDoS)	Saturation d'un serveur par un trafic massif pour le rendre indisponible.	**Filtrage de trafic (Firewall/WAF)** et solutions de protection anti-DDoS.
Injections (SQL, XSS)	Insertion de code malveillant dans des formulaires web.	**Validation des entrées** et développement logiciel sécurisé.
Man-in-the-Middle (MitM)	Interception des communications entre deux parties.	**Chiffrement des données (TLS/SSL)** et utilisation de VPN.

[2.4.10]

6 / les services de sécurité :

6/ 1- Service d'Authentification :

Ce service assure que l'identité d'une entité (utilisateur ou système) est bien celle qu'elle prétend être.

- **Authentification de l'entité paire :** Vérification de l'identité lors d'une connexion active.
- **Authentification de l'origine des données :** Garantie que la source des données reçues est authentique.

6/ 2- Service de Contrôle d'Accès :

Il protège contre l'utilisation non autorisée des ressources. Ce service détermine qui est autorisé à accéder à quoi, et sous quelles conditions (lecture, écriture, exécution).

6/ 3- Service de Confidentialité des Données :

Ce service protège les données contre la divulgation non autorisée. Il peut s'appliquer à un flux entier de messages, à des messages individuels ou même à des champs spécifiques au sein des données.

6/ 4- Service d'Intégrité des Données :

Il garantit que les données n'ont pas été modifiées, insérées, supprimées ou rejouées de manière non autorisée. Il permet de détecter toute altération du contenu durant le transfert ou le stockage

6/ 5- Service de Non-Répudiation :

Ce service empêche un émetteur ou un récepteur de nier avoir envoyé ou reçu un message.

[4.6.8.9]

Chapitre 02 : Le système Android

Tables Des Matiers :

1-Historique.....12

2-Présentation du systéme Android 16

3-Architecture Android.....19

4-Les permissions dans le système Android.....25

5-la Sécurité dans les systèmes android.....27

Liste des figures et tableau :

Figure 1: Le personnage Bugdroid. 18

Figure 2: Architecture d'Android 19

Figure 3: Architecture d'Android « Kernel linux » 20

Figure 4: Architecture d'Android « Librairies C/C++].....21

Figure 5: Architecture d'Android « Android runtime 21

Figure 6: Architecture d'Android « Application framework 22

Figure 7: Architecture d'Android « Application ». 22

Figure 8: Déclaration de la permission d'accéder à l'emplacement GPS …….....28

Tableau 2 : Vérification des permissions au niveau des composants.....30

Figure 9: Arbre de classification des applications Android.....33

1/ Historique

Android a débuté avec la sortie de la version 1.0 « Apple pie » en septembre 2008. Android a connu plusieurs mises à jour depuis sa première version. Ces mises à jour servent généralement à corriger des bugs et à ajouter de nouvelles fonctionnalités. Dans l'ensemble, chaque version est développée sous un nom de code basé sur des desserts. Ces noms de codes suivent une logique alphabétique, en voici quelques-unes :

Android 1.5 « Cupcake » a apporté de nombreuses améliorations. La première grande étape fut une mise à jour du noyau vers la version 2.6.27 qui a rendu le système plus stable et corrigea le manque d'API et rendit le système plus utilisable.

Android 1.6 « Donut », Android2.0 et **2.1 « Eclair »** ont apporté d'importantes améliorations respectivement sur les fonctionnalités et sur l'interface graphique du système.

Android 2.2 « Froyo » a fortement mis l'accent sur la synergie avec Internet. L'envoi d'applications et de liens instantanés depuis un ordinateur est désormais possible. Aussi, Google annonce-t-elle que le navigateur chrome intégré à Android 2.2 est le navigateur mobile le plus rapide au monde grâce à l'intégration du moteur JavaScript V8.

Android 2.3 « Ginerbread » Dernière version dédiée uniquement aux smartphones. Cette version est parfois utilisée sur de petites tablettes. Gingerbread était un système très réussi, qui devint à l'époque le système mobile le plus populaire.

Android 3.0 « Honeycomb » est spécialement étudié pour les tablettes tactiles. Les premiers modèles devraient être annoncés au CES 2011. On y apprend quelques nouveautés comme la prise en charge de la vidéo-conférence via Gtalk, la nouvelle interface Gmail ou encore le lecteur de livre électronique Google. La refonte graphique de l'interface utilisateur est assez réussie, plus d'informations devraient suivre dont surement des éclaircissements sur l'intégration ou non de l'interface de cette version d'Android sur les futurs smartphones.

Android 4.0 « Ice Cream Sandwich » devrait arriver très vite (mi 2011) pour rajouter encore plus de fonctionnalités aux terminaux. Pour le développement, ces nouvelles versions d'Android devraient proposer de nouveaux composants permettant de réaliser des applications avec une ergonomie plus adaptée aux tablettes tactiles.

Android 3.0 et **Android 4.0** devraient apporter plus d'outils aux constructeurs leur permettant de proposer des tablettes tactiles, qui seront capables de rivaliser (surtout au niveau de l'ergonomie) avec Ipad.

Android 4.1 « Jelly Bean » Il ajoute un système de notification améliorée, la reconnaissance vocale sans connexion internet, et le « Project Butter » qui augmente la fluidité d'Android. **Android 4.4 « KitKat »** Consommation en ressource moins élevée nécessitant moins de RAM, nouvelles icônes plus soignées, la barre du bas et celle de statut deviennent transparentes sur certains menus et changent de couleur en fonction du contenu affiché

Android 5.0 « Lollipop »
Cette version du système d'exploitation mobile de Google est apparue à la fin de l'année 2014. Elle introduit le concept de « Material Design », un nouveau langage graphique visant à offrir une expérience utilisateur plus moderne, cohérente et intuitive sur l'ensemble des appareils Android. Android Runtime (ART) remplace définitivement Dalvik afin d'améliorer les performances des applications. Le projet Volta permet une meilleure gestion de l'autonomie de la batterie. Le système de notifications est entièrement repensé et devient plus interactif. Cette version ajoute également la prise en charge des processeurs 64 bits, des technologies OpenGL ES 3.1 et Android Extension Pack, ainsi que des fonctionnalités telles que Tap and Go et Android Smart Lock.

Android 6.0 « Marshmallow »
Lancée en octobre 2015, cette version met l'accent sur la sécurité et la gestion des autorisations. Les utilisateurs peuvent désormais accorder ou refuser les permissions aux applications individuellement. Android Marshmallow introduit également la prise en charge de l'USB Type-C et des capteurs d'empreintes digitales. La fonctionnalité Doze permet de réduire la consommation énergétique lorsque l'appareil est inactif. Cette version améliore également la gestion du stockage externe et intègre Google Now on Tap, offrant des informations contextuelles directement à partir de l'écran.

Android 7.0 « Nougat »
Déployée en 2016, cette version apporte de nombreuses améliorations en matière de performances et de productivité. Android Nougat introduit la plateforme de réalité virtuelle Daydream ainsi qu'une meilleure optimisation du système grâce au compilateur Just In Time (JIT). L'API graphique Vulkan est intégrée afin d'améliorer les performances des applications et des jeux. De plus, Chrome est utilisé comme composant WebView, permettant une meilleure gestion de la mémoire et des ressources du système.

Android 8.0 « Oreo »
Lancée en août 2017, cette version marque une évolution importante avec l'introduction du projet Treble, qui facilite et accélère les mises à jour Android par les fabricants. Android Oreo

améliore les performances générales du système et propose une version allégée appelée Android Go Edition destinée aux appareils à faible mémoire vive. Cette version introduit également Google Play Protect pour renforcer la sécurité des utilisateurs et améliore la gestion des connexions Wi-Fi et de la confidentialité.

Android 9.0 « Pie »
Disponible depuis août 2018, Android Pie se distingue par l'intégration de l'intelligence artificielle dans plusieurs fonctionnalités du système. La batterie adaptative et la luminosité adaptative permettent d'optimiser automatiquement l'utilisation des ressources selon les habitudes de l'utilisateur. Cette version introduit également Digital Wellbeing, un ensemble d'outils visant à mieux contrôler le temps passé sur les applications et à favoriser une utilisation plus équilibrée du smartphone.

Android 10
Lancé officiellement en septembre 2019, Android 10 apporte des améliorations importantes en matière d'intelligence artificielle, de sécurité et de confidentialité. Cette version généralise le mode sombre à l'ensemble du système et améliore les réponses intelligentes aux notifications et aux messages. Android 10 renforce également la protection des données personnelles grâce à un contrôle plus précis des autorisations accordées aux applications.

Android 11
Publiée en 2020, cette version améliore la prise en charge des smartphones pliables et des réseaux 5G. Android 11 introduit de nouvelles fonctionnalités de gestion des conversations et renforce la sécurité grâce à des permissions temporaires permettant d'accorder l'accès à certaines données uniquement pendant l'utilisation d'une application. Cette version améliore également les performances multimédias et la compatibilité avec les nouveaux formats d'images.

Android 12
Déployé à partir de 2021, Android 12 propose une refonte visuelle importante avec l'introduction du design « Material You », permettant une personnalisation avancée de l'interface utilisateur. Cette version améliore également la confidentialité grâce au tableau de bord de confidentialité qui permet de surveiller l'accès des applications à la caméra, au microphone et à la localisation. Android 12 introduit aussi de nouvelles fonctionnalités destinées aux jeux et à la navigation gestuelle.

Android 13
Lancé en 2022, Android 13 poursuit l'amélioration de la personnalisation du système avec de nouveaux thèmes dynamiques et un lecteur multimédia modernisé. La sécurité et la confidentialité sont renforcées grâce à une gestion plus détaillée des autorisations accordées aux applications. Le presse-papiers bénéficie également d'améliorations visant à protéger davantage les données personnelles des utilisateurs.

Android 14

Disponible depuis octobre 2023, Android 14 met l'accent sur l'optimisation des performances, la stabilité et la sécurité du système. Cette version améliore la gestion des ressources afin d'offrir une meilleure autonomie de la batterie et une exécution plus fluide des applications. Android 14 introduit également des mécanismes avancés de protection des données, un contrôle plus précis des autorisations et de nouvelles mesures de sécurité destinées à lutter contre les logiciels malveillants et les cyberattaques.

.

.

.

Android 17 (Cinnamon Bun)

Est la version la plus récente , axé sur l'intégration native de l'intelligence artificielle, l'amélioration de l'expérience utilisateur et la connectivité inter-appareils. [AOSP]

Voici les nouveautés marquantes introduites dans cette version :

IA et Personnalisation

Widgets auto-générés par Gemini : Vous pouvez décrire textuellement le widget souhaité, et l'IA s'occupe de le créer de toutes pièces (suivi d'événements, compte à rebours, etc.).

Génération d'icônes par IA : Possibilité de créer des packs d'icônes personnalisés à l'aide de l'intelligence artificielle pour uniformiser toutes vos applications.

Saisie vocale "Rambler" : L'ancien moteur de dictée vocale est remplacé par Gemini pour une reconnaissance ultra-naturelle et multilingue en simultané. [Google Developer Guide for On-Device Inference with Gemini.]

Interface et Multitâche

Retour des widgets sur l'écran de verrouillage : Un espace dédié permet d'épingler de nouveau des widgets tiers sans déverrouiller le téléphone.

Fonction "Handoff" universelle : Une API qui vous permet de commencer une activité sur votre téléphone et de la poursuivre exactement là où vous en étiez sur une tablette Android compatible.

Bulles d'applications étendues : Vous pouvez désormais transformer presque n'importe quelle application en bulle flottante pour améliorer le multitâche. [**Guide de l'architecture des applications**]

Outils pour les Créateurs

Screen Reactions : Lors des enregistrements d'écran, vous pouvez incruster votre visage en direct avec un détourage automatique du fond (effet fond vert intégré).

Qualité vidéo constante (CQ) : Amélioration technique pour stabiliser la qualité visuelle lors des enregistrements vidéo à la volée. [Android Developpers : **Fonctionnalités et API**]

Sécurité et Bien-être numérique

Pause Point : Pour lutter contre le visionnage compulsif (*doom scrolling*), le système propose des alternatives plus saines ou un minuteur lorsque vous tentez d'ouvrir une application jugée trop distrayante.

Confidentialité réseau et localisation : Nouvelles restrictions pour empêcher les applications de scanner votre réseau local ou votre position précise sans autorisation explicite.

[32]

2/ Présentation du systéme Android

2/1.Introduction

Android est une excellente opportunité pour appréhender le développement d'applications mobiles ambitieuses.

Rappelons les points clés d'Android en tant que plate-forme :

ϖ elle est innovante car toutes les dernières technologies de téléphonie y sont intégrées : écran tactile, accéléromètre, GPS, appareil photo numérique etc. ;

ϖ elle est accessible car en tant que développeur vous n'avez pas à acheter de matériel spécifique (si vous voulez aller plus loin que l'utilisation d'un émulateur, un téléphone Android pour effectuer vos tests vous sera toutefois nécessaire), ni à connaître un langage peu utilisé ou spécifique : le développement sur la plate-forme Android est en effet réalisé en langage Java, un des langages de programmation les plus répandus ;

ϖ elle est ouverte parce que la plate-forme Android est fournie sous licence open source, permettant à tous les développeurs – et constructeurs – de consulter les sources et d'effectuer les modifications qu'ils souhaitent.

Ce chapitre est une introduction à la plate-forme, aux outils et à la configuration de notre environnement de travail.

2/2.Description

Android est un système d'exploitation pour Smartphones et tablettes tactiles conçu par Android. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs, des radio-réveils ou des autoradios et même des voitures. Il a été développé par une petite startup qui fut acheté en 2007 par Google qui poursuit activement son développement avec l'Open Handset Alliance, Android est distribué sous licence open source depuis 2008. Ce système est assez nouveau auprès des programmeurs.

Selon Google qui est un majeur distributeur, Android est une plateforme puissante, moderne, sécurisée et ouverte. Basé sur le kernel Linux2.6 (noyau Linux 2.6) et utilisant la plateforme java pour ses applications. Il est entièrement gratuit et sa plateforme très flexible ce qui permet aux développeurs d'intégrer, d'agrandir et de replacer les composants existants et d'adapter les applications aux besoins du client ou les remplacer entièrement, l'utilisateur peut donc personnaliser facilement son appareil.

De plus, il n'y a pas de distinction entre les applications natives et les applications qui sont développées par les développeurs, toutes sont disponibles sur l'Android Market(maintenant appelé Google Play Store).

En termes d'applications, Android a intégré plusieurs services de Google pour accéder rapidement aux services d'internet comme Gmail, You Tube, Google Talk, Google Calendar et Google Maps.

Android offre des avantages non négligeables, que ce soit pour les développeurs, ou pour les utilisateurs qui bénéficient grâce aux dernières versions de nombreuses fonctionnalités et d'un confort d'utilisation de qualité. Android est : [3]

Open source : Le contrat de licence pour Android respecte les principes de l'open source, c'est-à-dire que c'est possible de télécharger à tout moment les sources et les modifier. Android utilise des bibliothèques open source puissantes, comme par exemple SQLite pour les bases de données et OpenGL pour la gestion d'images 2D et 3D.

Gratuit (ou presque) : Android est gratuit, autant pour les utilisateurs que pour les constructeurs. La publication d'applications sur le Google Play1 , coûte la modique somme de 25$. Ces 25$ permettent de publier des applications à l'infini, contrairement à IPhone ou il faut verser une somme de 100$ par an.

Facile à développer : Toutes les API mises à disposition facilitent et accélèrent grandement le travail. Ces APIs sont très complètes et très faciles d'accès. De manière un peu caricaturale, on peut dire qu'il est possible d'envoyer un SMS en seulement deux lignes de code.

Facile à vendre : Le Play Store (anciennement Android Market) est une plateforme immense et très visitée ; c'est donc une mine d'opportunités pour quiconque possède une idée originale ou utile.

Flexible : Le système est extrêmement portable, il s'adapte à beaucoup de structures différentes. Les Smartphones, les tablettes, la présence ou l'absence de clavier ou de trackball, différents processeurs... On trouve même des fours à micro-ondes qui fonctionnent à l'aide d'Android. Non seulement c'est une immense chance d'avoir autant d'opportunités, mais en plus Android est construit de manière à faciliter le développement et la distribution en fonction des composants en présence dans le terminal (si une application nécessite d'utiliser le Bluetooth, seuls les terminaux équipés de Bluetooth pourront la voir sur le Google Play).

Ingénieux : L'architecture d'Android est inspirée par les applications composites, et encourage par ailleurs leur développement. Ces applications se trouvent essentiellement sur internet et leur principe est la possibilité de combiner plusieurs composants totalement différents pour obtenir un résultat surpuissant. Par exemple, si on combine l'appareil photo avec le GPS, on peut poster les coordonnées GPS des photos prises.

2/3.Open Handset Alliance

L'Open Handset Alliance (abrégé OHA) est un consortium Regroupant de grands constructeurs et développeurs de logiciels dont le but est de développer des normes ouvertes pour les appareils de téléphonie mobile.

Le consortium a été créé le 5 novembre 2007 à l'initiative de Google qui a su fédérer autour de lui 34 compagnies. On note ainsi la présence d'opérateurs comme NTT DoCoMo, Sprint Nextel, Telecom Italia ou Bouygues Telecom. Des équipementiers tels que Samsung Electronics, Motorola Mobility ou LG Electronics. Des semi-conducteurs comme Intel ou Nvidia. Et enfin des incontournables de l'Internet dont eBay. On remarquera quelques grands absents dont Nokia.

2/4.Bugdroïde

Le personnage nommé Bugdroid est le petit robot vert utilisé par Google pour présenter Android. Ce personnage est sous licence « creativecommons by (3.0) » et peut donc être utilisé librement.

Le site Engadget annonce que Bugdroid, le logo d'Android, serait en fait un personnage d'un jeu des années 1990 sur Atari : Gauntlet: The ThirdEncounter.



Figure 1: le personnage Bugdroïde

3/ Architecture Android

L'architecture du système Android est organisée en plusieurs couches logicielles qui travaillent ensemble pour assurer le fonctionnement des applications et la gestion des ressources matérielles de l'appareil. Cette architecture modulaire permet une meilleure flexibilité, sécurité et performance du système.

Au niveau le plus bas se trouve le Linux Kernel, qui constitue le noyau du système. Il assure la gestion de la mémoire, des processus, de la sécurité, des pilotes matériels (Wi-Fi, Bluetooth, caméra, écran, etc.) ainsi que la communication entre le matériel et les logiciels.

Au-dessus du noyau se trouvent les Libraries et l'Android Runtime (ART). Les bibliothèques natives fournissent des fonctionnalités essentielles telles que le traitement graphique, la gestion des bases de données SQLite, le rendu web et les services multimédias. Android Runtime permet l'exécution des applications Android et optimise leurs performances.

La couche suivante est l'Application Framework, qui fournit aux développeurs un ensemble d'API et de services permettant de créer des applications. Cette couche comprend notamment

le gestionnaire d'activités (Activity Manager), le gestionnaire de fenêtres (Window Manager), le gestionnaire de notifications (Notification Manager) et les services de localisation.

Enfin, la couche supérieure est constituée des Applications, qui regroupent les applications système préinstallées ainsi que les applications développées par les utilisateurs ou les développeurs tiers. Ces applications utilisent les services offerts par les couches inférieures pour fonctionner efficacement.

Ainsi, l'architecture d'Android repose sur une organisation en couches qui garantit la portabilité, la sécurité, la stabilité et la facilité de développement des applications mobiles**.**

Le diagramme suivant illustre les composants principaux du système d'exploitation Android. Chaque section sera décrite dans ce qui suit :

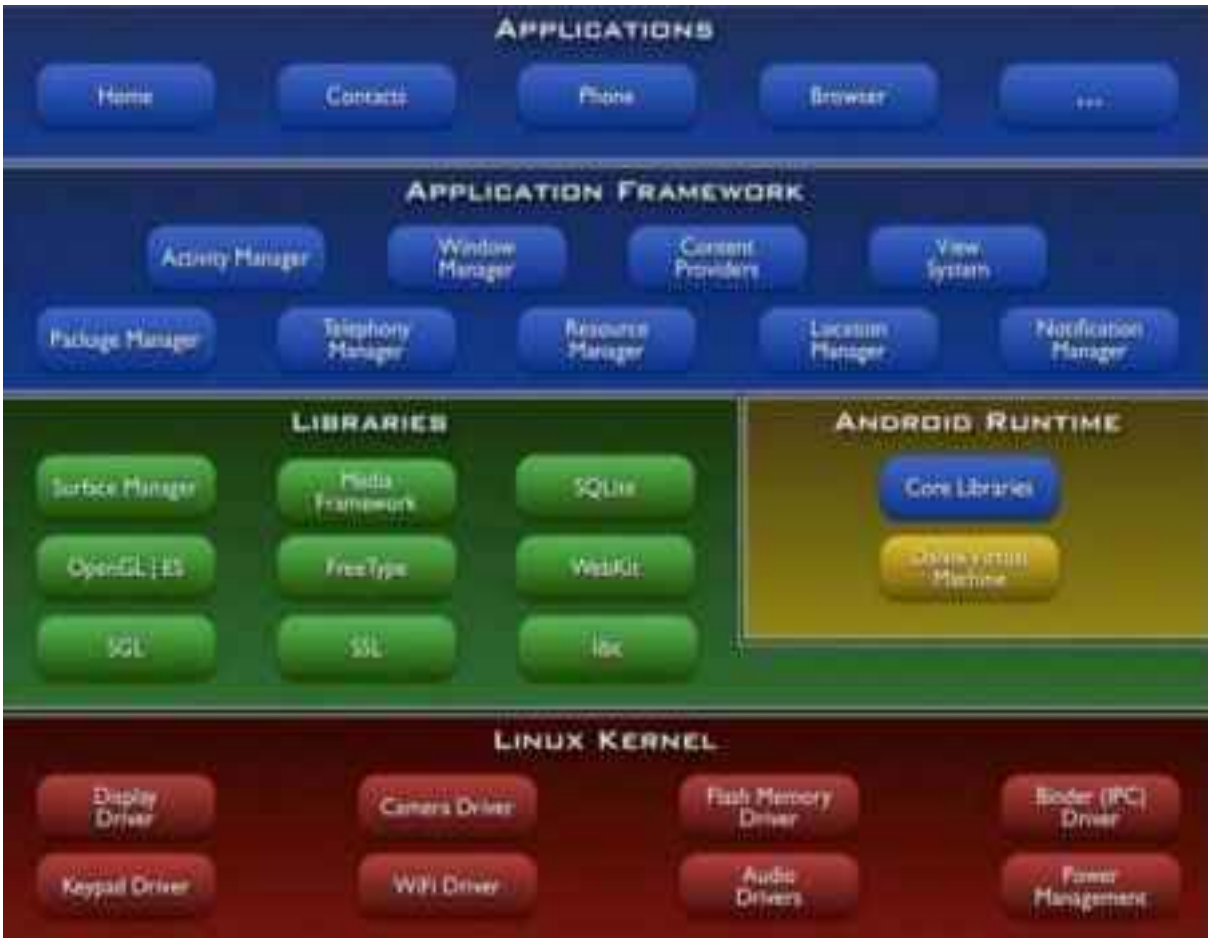

Figure 2:Architecture d'Android

Android est basé sur un kernel linux 2.6.xx.



Figure 3:Architecture d'Android « Kernel linux

Au-dessus de cette couche, on retrouve les librairies C/C++ utilisées par un certain nombre de composants du système Android.



Figure 4:Architecture d'Android « Librairies C/C++ »

Au-dessus des librairies, on retrouve l'Android Runtime. Cette couche contient les librairies cœurs du Framework ainsi que la machine virtuelle exécutant les applications.



Figure 5: Architecture d'Android « Android runtime »

Au-dessus de la couche "Android Runtime" et des librairies cœurs, on retrouve le Framework permettant au développeur de créer des applications.



Figure 6: Architecture d'Android « Application framework »

Enfin au-dessus du Framework, il y a les applications .



Figure 7: Architecture d'Android « Application »

3/1.Applications

java « avant l'apparition du langage Kotlin en 2017 qui est devenu officiellement le langage de programmation voulu et recommandé par le géant américain Google ».

Elle sont Alors c'est la couche ApplicationJ. C'est un ensemble des programmes écrits généralement en généralement distribuées depuis des plate formes de téléchargement telles que l'AppStore (plateforme d'Apple), le GooglePlay (plateforme de Google / Android), ou encore le Windows Phone Store (plateforme de Microsoft).

Une application Android peut être composée des éléments suivants:

• **Activité « Activity »** : une interface graphique « une vue » (équivalent à une fenêtre ou boite de dialogue en java). • Service : un programme qui s'exécute en arrière plan.

• **Fournisseurs de contenus « Provider »** : un mécanisme (une Class) permettant le partage d'informations entre applications. • Widgets : une vue accrochée au bureau d'Android.

• **Notifications** : permet de notifier l'utilisateur de la survenue d'événements.

• **Intention « Intent** » : un mécanisme (une Class) peut être utilisé pour transiter des données entre deux activités.

• **Récepteurs d'Intents « BoradcastReceiver** »

3/2.Framework de développement

En fournissant une plateforme de développement ouverte, le système Android offre aux développeurs la possibilité de créer des applications extrêmement riches et innovantes. Les développeurs sont libres de profiter du matériel périphérique et informations sur la localisation d'accès, exécuter des services d'arrière-plan, définir des alarmes, ajouter des notifications à la barre d'état, etc.

Les développeurs ont un accès complet au même framework API utilisé par les applications de base. L'architecture d'application est conçue pour simplifier la réutilisation des composants; n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités (soumis à des contraintes de sécurité appliquées par le framework). Ce même mécanisme permet aux composants d'être remplacés par l'utilisateur. Toutes les applications sous-jacentes forment un ensemble de services et de systèmes, y compris:

- Un jeu extensible de vues qui peuvent être utilisées pour construire une application.

- Des fournisseurs de contenu qui permettent aux applications d'accéder aux données des autres applications (telles que les Contacts), ou de partager leurs propres données.

- Un gestionnaire de ressources.

- Un gestionnaire de notification qui permet à toutes les demandes d'afficher des alertes personnalisées dans la barre d'état.

- Un gestionnaire d'activité qui gère le cycle de vie des applications et propose une navigation commune

3/3 Le noyau linux :

Le système d'exploitation d'Android se basait sur Linux, c'est le noyau («kernel » en anglais) de Linux qui est utilisé. Le noyau est l'élément du système d'exploitation qui permet de faire le pont entre le matériel et le logiciel. Par exemple les pilotes Wifi

permettent de contrôler la connexion Wifi. La version du noyau utilisée avec Android est une version conçue spécialement pour l'environnement mobile, avec une gestion avancée de la batterie et une gestion particulière de la mémoire. C'est cette couche qui fait en sorte qu'Android soit compatible avec tant de supports différents

4/ Les permissions dans le système Android

Les permissions servent à protéger la confidentialité d'un utilisateur dans un appareil Android.

Les applications Android doivent demander l'autorisation d'accéder aux données sensibles de l'utilisateur. Parmi ces données sensibles, on peut citer les contacts.

Les applications Android doivent demander aussi l'autorisation d'accéder à certaines fonctionnalités du système comme l'appareil photo, l'internet, etc.

Selon la fonctionnalité, le système peut accorder la permission automatiquement ou peut inviter l'utilisateur à approuver la demande.

Aucune application n'est autorisée à effectuer des opérations qui auraient un impact négatif sur les autres applications, le système d'exploitation ou l'utilisateur

On peut citer par exemple la lecture ou l'écriture des données privées de l'utilisateur (telles que les contacts ou les courriels), la lecture ou l'écriture des fichiers d'une autre application, la création d'un accès réseau, le maintien de l'appareil en veille, etc.

4/1.Types de permissions Android

Permission normale

Une permission est dite normale quand elle représente un risque très faible pour la confidentialité de l'utilisateur ou le fonctionnement d'autres applications.

La permission d'accéder à l'internet est une autorisation normale.

Si une application déclare dans son manifeste qu'elle a besoin d'une permission normale, le système lui attribue automatiquement cette permission au moment de l'installation.

Le système ne demande pas à l'utilisateur d'accorder les permissions normales.

Les utilisateurs ne peuvent pas révoquer ces permissions.

Permission de signature

Le système accorde ces permissions à l'application au moment de l'installation, mais uniquement lorsque l'application qui tente de les utiliser est signée par le même certificat que l'application qui définit ces permissions.

C'est le type le plus "robuste" de signature, car elle nécessite l'utilisation d'une clé cryptographique.

Les applications utilisant ce type de permissions sont toutes contrôlées par le même développeur.

C'est le système qui décide s'il est nécessaire de faire intervenir l'utilisateur.

Ces permissions sont généralement utilisées par les applications du système.

Permission à l'exécution (« runtime »)

Elles sont dites aussi dangereuses.

Elles souhaitent obtenir des données ou des ressources impliquant les informations privées de l'utilisateur ou susceptibles d'affecter les données stockées de l'utilisateur ou le fonctionnement d'autres applications.

Par exemple, la possibilité de lire les contacts de l'utilisateur.

Si une application déclare avoir besoin d'une permission dangereuse, l'utilisateur doit explicitement l'accorder.

Jusqu'à ce que l'utilisateur approuve la permission, votre application ne peut pas fournir de fonctionnalités qui dépendent de cette permission.

Pour utiliser une permission dangereuse, votre application doit inviter l'utilisateur à accorder cette permission au moment de l'exécution.

5/ la Sécurité dans les systèmes android

5/1. introduction :

Aujourd'hui, la convergence des technologies a transformé ce qui était autrefois un simple téléphone portable en un véritable ordinateur de poche, au sens propre comme au figuré. Ce que nous appelons dans nos poches un smartphone, ou « ordiphone », mérite bien son nom.

D'un point de vue utilisateur, ces appareils servent à jouer, consulter ses mails, naviguer sur le web, et bien plus encore. Du point de vue technique, ils exploitent désormais des processeurs comparables à ceux des netbooks et disposent de systèmes d'exploitation complexes remplaçant le firmware traditionnel.

Cependant, le téléphone portable dépasse la simple fonction de communication : il nous accompagne toute la journée, suit nos interactions et habitudes, et permet même l'accès à des services payants.

C'est dans ce contexte qu'en 2008, Android a été lancé. Basé sur Linux et open source, ce système d'exploitation a rapidement conquis les développeurs grâce à la disponibilité de téléphones relativement ouverts et à la possibilité de personnaliser le système.

Le ministère de la Défense s'est naturellement intéressé aux possibilités offertes par Android,

5/2. modéles de sécurité ANDROID :

Comme vu précédemment, Android est construit sur un noyau Linux. Par conséquent, il hérite de l'architecture de sécurité de Linux ainsi que de son modèle de contrôle d'accès. Le noyau Linux fait d'Android un système multiprocessus exécutant chaque application dans son propre processus. La sécurité d'Android est implémentée par différents mécanismes et à différents niveaux. Celle-ci est renforcée, en grande partie, au niveau du noyau Linux à travers des mécanismes de Linux tels que le UID et les GID assignés à chaque application (Asaf et al., 2010). Au niveau des applications, Android propose un système de permissions en vue de restreindre l'accès à des composants spécifiques appartenant au système et aux applications installées sur le téléphone. En outre, un mécanisme basé sur les signatures numériques force les développeurs à signer leurs applications pour qu'elles puissent être installées. D'autre part, Android met en place des mécanismes visant principalement à protéger les données manipulées par

l'application. Dans le présent chapitre, nous présentons les détails de l'architecture de sécurité d'Android au niveau du noyau Linux et celui des applications. **[11]**

5/3. Mécanisme de sécurité au niveau de l'application :

3.1- Signature numérique du package de l'application

Android exige que chaque application soit signée numériquement par une clé privée associée à un cert ificat numérique (Shabtai et al., 2009), faute de quoi elle ne peut être installée sur l'appareil.

Cependant, aucune autorité de certification (CA) n'est requise, permettant ainsi aux développeurs d'utiliser des certificats auto-signés. Par conséquent, la signature ne sert qu'à associer l'application à son développeur, sans pour autant servir à vérifier sa vraie identité. Au moment de l'installation de l'application, Android utilise le certificat du développeur (fourni avec l'application) afin de vérifier l'intégrité du package de l'application. Si le contenu du package de l'application a été modifié, alors la signature devient invalide et l'installation de l'application est annulée.

Les signatures sont aussi utilisées afin de vérifier si deux applications ou plus appartiennent au même développeur – en fait, signées en utilisant le même certificat digital. Si tel est le cas, elles peuvent utiliser le même ID à travers la fonctionnalité shareUserId, utiliser le même ensemble de permissions et acquérir les signatures de type Signature et SignatureOrSystem accordées systématiquement au moment de l'installation. Plus de détails seront présentés dans la prochaine section. **[12]**

3.2- Le modèle des permissions

Par défaut, Android ne permet l'accès qu'à des parties restreintes des ressources système, sauf pour les applications système. En effet, Android comporte un certain nombre d'API sensibles qui sont destinées à l'usage exclusif des applications système ainsi que les applications auxquelles l'utilisateur accorde sa confiance (AOSP, 2008f). Ces API renferment des fonctionnalités sensibles telles que les fonctions de téléphonie, SMS/MMS, GPS, appareil photo, etc., et sont protégées par le mécanisme des permissions.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.example.android.exampleApplication">
   <uses-sdk android:minSdkVersion="10" />

<application   android:label="Example application" >
      <activity android:name=".ControllerApp" >   </activity>

     <provider android:name="Fournisseur Adresse"
               android:authorities="com.example.android.exampleApplicationProvider">
     </provider>

      <service android:name="ExampleService"
           android:exported = "false">
      </service>
   </application>
 <uses-permission android:name = "android.permission.ACCESS-FINE_LOCATION"
</manifest>
```

Figure 8: Déclaration de la permission d'accéder à l'emplacement GPS

3.2.1 Déclaration des permissions

Afin de pouvoir utiliser les fonctionnalités protégées du système, une application doit déclarer dans son fichier AndroidManifest.xml les permissions requises à cet effet (voir figure 3.1). Lors de l'installation de l'application, le système affiche une boîte de dialogue décrivant les permissions sollicitées et demande à l'utilisateur s'il souhaite continuer l'installation (voir figure 3.2). Si l'utilisateur souhaite continuer, le système accorde toutes les permissions demandées à l'application. Par contre, Android n'offre pas la possibilité d'accorder ou de refuser des permissions individuelles, ce qui force l'utilisateur à accorder ou refuser toutes les permissions demandées en bloc. Une fois installée, l'application conserve ces permissions jusqu'à ce qu'elle soit désinstallée par l'utilisateur. De plus, il n'existe aucun moyen de révoquer ces permissions mise à part la désinstallation de l'application.

3.2.2 Les permissions au niveau des composantes

Comme nous l'avons vu au chapitre 2, une application Android est un ensemble de composants de différents types pouvant interagir les uns avec les autres en utilisant le mécanisme d'intention. Les permissions sont également utilisées pour aider les développeurs à limiter l'accès aux composants de leurs applications en définissant leurs propres permissions dans le fichier Manifest. Les permissions définies par l'utilisateur servent à protéger les composants de l'application contre toute utilisation abusive. Les permissions sont définies en utilisant la balise , et doivent être mentionnées dans

l'attribut Android:permission du composant devant être protégé. Une application désirant accéder à un composant protégé par une permission, doit explicitement demander la permission requise à cet effet dans son fichier Manifest Chaque permission déclarée par le développeur possède un niveau de protection qui lui est associé (Shabtai et al., 2010) :

Normal : Ce type de permissions est accordé de façon automatique sans demander l'approbation de l'utilisateur.

Dangerous : Le système demande à l'utilisateur s'il souhaite accorder ce type de permission à l'application qui la demande.

Signature: Ce type de permissions est accordé automatiquement par le système aux applications ayant été signées par la même clé privée et donc appartenant au même développeur.

SignatureOrSystem : Ce type de permissions n'est accordé qu'aux applications signées par la même clé privée que l'image système (donc signées par Google) ou aux applications signées par la même clé privée qu'une application déjà installée

Il appartient au développeur d'attribuer le niveau de permissions pour chaque permission qu'il déclare. Les permissions de type Normal impliquent un risque mineur et ne servent qu'à informer l'utilisateur que l'application qu'il est sur le point d'installer demande l'accès à une fonctionnalité donnée (p. ex., utiliser le vibreur). D'autre part, les permissions de type Dangerous sont utilisées pour protéger des composants offrant des fonctionnalités impliquant un risque plus important. Les permissions de type Signature sont utilisées pour garantir que seules les applications appartenant au même développeur peuvent interagir avec le composant protégé par la permission.

3.2.3 Usage des permissions au niveau des composants

Les permissions sont utilisées afin de protéger les composants du système ainsi que les composants appartenant aux applications. Les restrictions d'accès diffèrent selon le type de composant. Le tableau 3.1 montre comment les différents types de composants sont protégés, le moment de vérification et l'action prise par le système dans le cas où une application ne disposerait pas des permissions nécessaires**.**

3.2.4 Usage des permissions au niveau des intentions

Android offre la possibilité aux développeurs de spécifier quelle application peut recevoir et lire le contenu des intentions qu'elle envoie. Ce type de restriction s'applique aux intentions implicites décrites dans la section 2.4. En passant une permission comme paramètre à la méthode sendBroadcast (intention, permission), seules les applications qui détiennent cette permission particulière sont autorisées à recevoir l'intention envoyée. Lorsqu'une application envoie une intention, le système vérifie d'abord si l'application appelée détient la permission de la recevoir. Par la suite, il vérifie si l'application appelante a le droit d'envoyer des intentions à l'application destinatrice

Tableau 2:Vérification des permissions au niveau des composants

Composant Android	Actions restreintes	Permissions vérifiées lors de	Action déclenchée par le manque de permissions
Activité	Démarrer l'activité	*startActivity(), ActivityForResult()*	Une exception de sécurité est renvoyée à l'appelant
Service	Démarrer ou se lier à un service	*startService(), stopService(), bindService()*	Une exception de sécurité est renvoyée à l'appelant
Récepteur de diffusion	Envoi de diffusion au récepteur de diffusion	*sendBroadcast()*	L'intention n'est pas livrée au destinataire, sans exception de sécurité
Fournisseur de contenu	Accéder aux données détenues par le fournisseur de contenu	*ContentResolver.query(), ContentResolver.insert(), ContentResolver.update(), ContentResolver.delete()*	Une exception de sécurité est renvoyée à l'appelant

3.2.5 Vérification des permissions au niveau des processus

En plus des vérifications de permissions effectuées par le système lors des interactions entre les composantes des applications, Android offre la possibilité de vérifier les permissions des autres applications à n'importe quel moment durant l'exécution de l'application. La méthode checkCallingPermissions() peut être utilisée par une application afin de vérifier les permissions de l'application qui l'appelle. Également, la méthode checkPermissions() est invoquée lors qu'on désire vérifier les permissions accordées à une application donnée en utilisant le PID du processus dans lequel elle s'exécute. La dernière forme de vérification se base sur le nom d'un package (application) donnée en utilisant PackageManager.checkPermission(). **[13]**

3.3- Les permissions sur les fournisseurs de contenu :

L'utilisation des permissions pour les fournisseurs de contenu peut s'avérer insuffisante (AOSP, 2008f). Un développeur pourrait exprimer le besoin de définir son application comme étant la seule capable de mettre à jour le contenu de son fournisseur de contenu, et limiter l'accès des autres applications à la lecture seulement (Enck, Ongtang et McDaniel, 2009). Afin de combler ce besoin, Android permet de séparer les droits de lecture et d'écriture pour chaque fournisseur de contenu ce qui augmente le degré de contrôle sur les données de l'application

3.4- Les permissions par URI :

Android permet une forme de délégation momentanée de permissions qui, lorsqu'utilisée, permet à une application d'accéder à un URI appartenant à une autre application sans avoir demandé la ou les permissions au préalable (Enck, Ongtang et McDaniel, 2009). En effet, une application peut envoyer une intention qui contient l'URI d'une donnée quelconque, avec le flag Intent.FLAG_GRANT_READ_URI_PERMISSION. L'application destinatrice pourrait alors lire le contenu de l'URI passé dans l'intention, peu importe si elle dispose ou non des permissions pour accéder au fournisseur de contenu de l'application appelante. Par exemple, une application de messagerie électronique peut utiliser le mécanisme décrit cidessus afin de permettre à une deuxième application de consulter le contenu d'un courriel en particulier sans pour autant lui permettre d'accéder au contenu de tous les courriels

3.5- composant publique et composant privé :

Les applications Android comportent souvent des composants qui ne doivent jamais être accessibles par le reste des applications (Enck, Ongtang et McDaniel, 2009). En effet, les composants peuvent manipuler des informations sensibles et retourner des résultats (tels que 32 les mots de passe et les numéros de carte de crédit) à d'autres composants. Android permet de mettre ce type de composant à l'abri des autres applications, en définissant l'attribut exported dans la déclaration du composant dans le fichier AndroidManifest.xml. Si exported est true, le composant est accessible à toutes les applications. Si la valeur est false, alors le composant ne peut être accessible qu'aux composants de la même application ou des applications ayant le même ID**.**

Par ailleurs, la valeur par défaut de l'attribut exported est déterminée par l'existence ou l'absence des filtres d'intention dans la déclaration du composant. Si aucun filtre d'intention n'est attaché au composant, celui-ci est considéré comme privé (AOSP, 2008c). Cependant, il est toujours possible de l'activer s'il est désigné par son nom tel qu'il apparait dans le fichier Manifest. D'autre part, l'existence des filtres d'intention implique que le composant est public et accessible aux autres applications. L'accès est alors régi par le mécanisme de permissions.**[14]**

5/4. ATTAQUES ET VULNÉRABILITÉS CONTRE LE SYSTÈME ANDROID:

Les applications malveillantes Android exploitent de nombreux types de vulnérabilités. La forme la plus simple des applications malveillantes ciblant Android tire parti de

l'inadvertance des utilisateurs des appareils Android, pour atteindre leurs objectifs. Un type plus sophistiqué d'applications malveillantes exploite des vulnérabilités de la plateforme mobile. Ces vulnérabilités peuvent être des vulnérabilités de la plateforme Android ou des vulnérabilités spécifiques au noyau Linux sur lequel est construit Android. Dans ce chapitre, nous proposons une nouvelle classification qui partage les différents types des applications Android malveillantes en se basant sur la nature des vulnérabilités exploitées. Nous illustrons cette classification en présentant des exemples d'applications malveillantes pour chaque catégorie.

Cette nouvelle classification est présentée schématiquement dans la figure 9

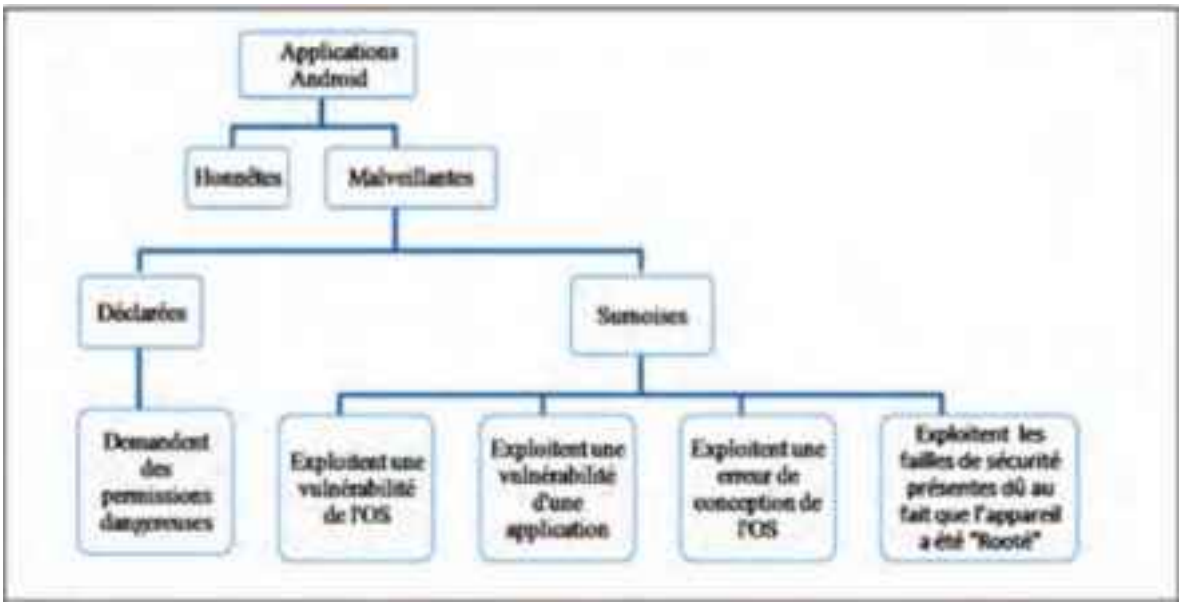

Figure 9: Arbre de classification des applications Android

[15]

5/5. Les applications malveillantes déclarées

Comme nous l'avons vu précédemment, la sécurité d'Android est essentiellement basée sur le modèle de permissions, qui compte sur l'utilisateur pour donner les permissions nécessaires à l'application qui les demande. Cependant, l'efficacité du système de permissions est fortement dépendante des connaissances de l'utilisateur, en plus de sa compréhension du modèle de permissions. Alors qu'une minorité d'utilisateurs d'Android peut avoir une très bonne connaissance liée aux systèmes d'exploitation, la grande majorité de cette communauté n'a pas les connaissances nécessaires liées à la plate-forme mobile, et encore moins sur la sécurité d'Android. Par ailleurs, la plupart des utilisateurs sont insensibles aux risques et aux impacts que la décision d'octroi de permissions peut avoir sur la confidentialité de leurs données

personnelles ainsi que sur la fiabilité de leur téléphone mobile Android. Les applications malveillantes déclarées tirent avantage de l'inadvertance de la plupart des utilisateurs d'Android quand il s'agit de l'octroi de permissions aux applications. Nous classons ce type d'applications malveillantes comme déclarées puisqu'elles demandent des permissions qui sont inutiles pour leur fonctionnement et donc elle déclare implicitement ses intentions malveillantes.

L'application malveillante classée sous cette catégorie se présente comme une application légitime, qui prend souvent la forme d'un simple jeu. Au moment de l'installation, elle requiert une ou plusieurs permissions dangereuses. Des exemples de permissions dangereuses seraient :

- effectuer des appels téléphoniques
- envoyer des SMS
- obtenir la localisation GPS
- obtenir la liste des contacts de l'utilisateur

Toutefois, rien dans la description de l'application n'indique que l'application aurait besoin de ces privilèges pour accomplir les tâches qu'elle prétend réaliser. Ces applications sont plus susceptibles d'être des applications malveillantes que des applications légitimes. Une application malveillante de ce genre contient un code caché qui effectue des tâches en silence à l'insu des utilisateurs, et en l'absence de moyen d'avertissement (p. ex., notifications d'envoi de message), l'utilisateur ne pourrait éviter l'impact d'une telle attaque à un stade précoce.

La version gratuite du logiciel malveillant était conçue pour envoyer les données GPS de l'utilisateur à un serveur externe toutes les 15 minutes, tandis que la version payante, appelée GPSSpy, permettait à son utilisateur de retracer l'emplacement d'un utilisateur spécifique. Supposons que nous voulons suivre quelqu'un. Nous n'avons qu'à inciter cette personne à installer l'application sur son téléphone. À la première utilisation, l'utilisateur sera invité à saisir son adresse de courriel. En utilisant la version payante, nous utilisons la même adresse

ourriel avec l'application GPSSpy payante pour suivre l'historique des emplacements géographiques visités par l'utilisateur espionné. Suite à la découverte de l'application malveillante, Google l'a retiré d'AndroidMarket, et a procédé à sa désinstallation à distance des appareils infectés. Cependant, quelques versions de l'application malveillante continuent à exister sur les marchés d'applications parallèles et sont disponibles pour le téléchargement.



Figure 10: fonctionnement d'Application GPSS

[16]

Chapitre 03 : L’ apprentissage

Table Des Matiers :

1 machine Learning(Apprentissage automatique).....…….39

2 Types d'Algorithmes Machine Learning 40

3 Classificateurs 43

4 Deep learning (Apprentissage profond) 46

5 Apprentissage automatique vs Apprentissage profond 47

Liste des figures et tableau :

Figure 11 : Classification de l'apprentissage automatique..... 40

Figure 12: Répartition annuelle des articles pour les types de conception de classificateurs.....43

Figure 13: Un exemple de classification par k-plus proches voisins pour k=5 44

Figure 14 : Différence entre l'apprentissage automatique et l'apprentissage profond ..48

1/ Apprentissage automatique :

1/ Introduction

Le Apprentissage automatique est une branche de l'Intelligence Artificielle qui permet aux ordinateurs d'apprendre automatiquement à partir de données d'entraînement, puis d'améliorer Introduction L'apprentissage automatique est un sous-domaine de l'informatique qui s'intéresse à la construction d'algorithmes qui, pour être utile, s'appuie sur un ensemble d'exemples de certains phénomènes. Ces exemples peuvent provenir de la nature, être fabriqués à la main par l'homme ou générés par un autre algorithme. L'apprentissage automatique peut également être défini comme le processus de résolution d'un problème pratique par 1) la collecte d'un ensemble de données, et 2) la construction algorithmique d'un modèle statistique basé sur cet ensemble de données. Ce modèle statistique est censé être utilisé d'une manière ou d'une autre pour résoudre le problème pratique.

Une définition un peu plus générale :[L'apprentissage automatique est le domaine d'étude qui donne aux ordinateurs la possibilité d'apprendre sans être explicitement programmés.][1] Et une autre plus technique : [on dit qu'un programme informatique apprend de l'expérience E en ce qui concerne une tâche T et une mesure de performance P, si sa performance sur T, mesurée par P, s'améliore avec l'expérience E.]**[18]**

Dans ce mémoire, nous nous intéressons à l'estimation des valeurs des propriétés immobilières, c'est pour cela que nous devons nous demander quelles sont les grandes lignes de l'apprentissage automatique ?

Dans ce chapitre, nous présentons un état de l'art de l'apprentissage automatique. Tout d'abord, nous donnons un bref aperçu historique allant des prémisses à l'essor de ce domaine. Ensuite, nous donnons ses différents types et certains algorithmes utilisés. Avant de conclure ce chapitre, nous décrivons la manière de tester et de valider un modèle d'apprentissage automatique.

2/Définition de L'apprentissage automatique

L'apprentissage automatique également connu sous le nom de "Machine Learning", est un domaine de recherche en intelligence articielle qui vise à extraire des connaissances pertinentes à partir de grandes quantités de données. Cette approche repose sur l'utilisation d'algorithmes d'apprentissage automatique qui permettent de modéliser le processus d'apprentissage. Ces modèles peuvent être utilisés pour prédire la classe ou le groupe auquel appartiendront de

nouvelles instances de données. En somme, l'objectif de l'apprentissage automatique est de fournir des connaissances utiles à partir de données en utilisant des algorithmes sophistiqués. **[19]**

2/Types d'Algorithmes Apprentissage automatique

Les algorithmes de Apprentissage automatique peuvent être classés en plusieurs catégories principales : l'apprentissage supervisé, l'apprentissage non supervisé et l'apprentissage par renforcement.

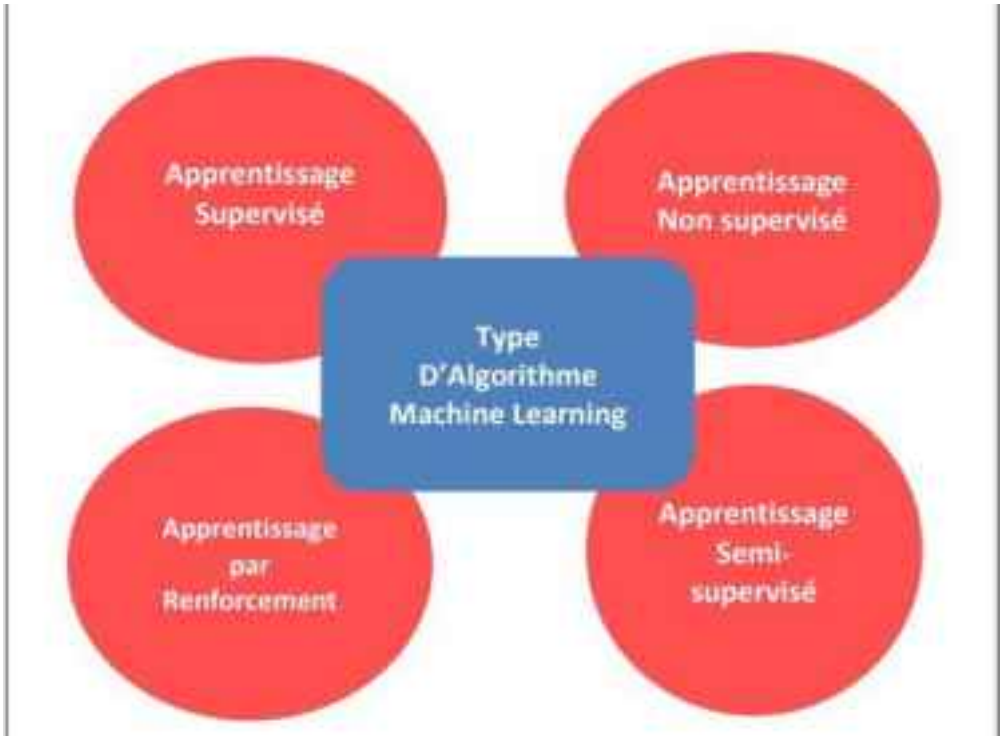

Figure 11: Classification de l'apprentissage automatique

3/1.Apprentissage supervisé

L'objectif de l'apprentissage supervisé est d'utiliser que des données étiquetées pour apprendre. Il existe principalement deux types d'algorithmes d'apprentissage supervisé : les algorithmes de régression, qui peuvent convertir l'espace d'entrée en un domaine de valeurs réelles (par exemple, la régression bayésienne et la régression par vecteur de support), et les algorithmes de classication, qui peuvent convertir l'espace d'entrée en classes prédénies (comme les réseaux neuronaux articiels, les arbres de décision, les forêts d'arbre et les machines à vecteur de support SVM) **[20]**

1.1.Classification en apprentissage automatique supervisé.

Il existe plusieurs algorithmes de classification utilisés en apprentissage supervisé, parmi lesquels les machines à vecteurs de support et les classificateurs bayésiens naïfs sont les plus courants. Dans les tâches de classification, la valeur de sortie est une catégorie avec un nombre fini d'options.

1.2.régression en apprentissage supervisé en Machine Learning

En apprentissage supervisé, la régression consiste à prédire une valeur numérique continue, telle que la probabilité qu'un événement se produise, qui peut avoir n'importe quelle valeur numérique dans une certaine plage.

2/2. Apprentissage non supervisé

Les algorithmes d'apprentissage non supervisé découvrent des informations et des relations dans des données non étiquetées. Dans ce cas, les modèles sont alimentés en données d'entrée, mais les résultats désirés sont inconnus, ils doivent donc tirer des conclusions à partir d'indices circonstanciels, sans aucune guidance ou formation. Les modèles ne sont pas entraînés avec la « bonne réponse », ils doivent donc trouver des motifs par eux-mêmes. L'un des types les plus courants d'apprentissages non supervisés est le clustering, qui consiste à regrouper des données similaires. Cette méthode est principalement utilisée pour l'analyse exploratoire et peut aider à détecter des modèles ou des tendances cachées. Par exemple, l'équipe marketing d'une entreprise de commerce électronique pourrait utiliser le clustering pour améliorer la segmentation des clients. Avec un ensemble de données de revenus et de dépenses, un modèle d'apprentissage automatique peut identifier des groupes de clients ayant des comportements similaires. La segmentation permet aux marketeurs d'adapter les stratégies pour chaque marché

clé. Ils pourraient offrir des promotions et des remises aux clients à faible revenu qui dépensent beaucoup sur le site, afin de récompenser la fidélité et d'améliorer la rétention.

2/3. Apprentissage par renforcement

L'apprentissage par renforcement (RL) concerne la façon dont un agent logiciel (ou un programme informatique) devrait agir dans une situation pour maximiser la récompense. En bref, les modèles d'apprentissage renforcé tentent de déterminer le meilleur chemin possible à suivre dans une situation donnée. Ils le font par essais et erreurs. Comme il n'y a pas de données d'entraînement, les machines apprennent de leurs propres erreurs et choisissent les actions qui mènent à la meilleure solution ou à la récompense maximale.

Cette méthode d'apprentissage automatique est principalement utilisée dans la robotique et les jeux. Les jeux vidéo démontrent une relation claire entre les actions et les résultats, et peuvent mesurer le succès en gardant une note. Par conséquent, ils constituent un excellent moyen d'améliorer les algorithmes d'apprentissage par renforcement.

2/4. Apprentissage semi-supervisé

L'apprentissage semi-supervisé est une solution alternative pour pallier le manque de données annotées. Cette méthode d'entraînement permet de combiner des données étiquetées et non étiquetées pour entraîner des modèles. Cette particularité en fait une option intéressante pour les cas où les données classiées sont limitées, pouvant servir d'alternative aux méthodes d'apprentissage supervisé. **[21]**

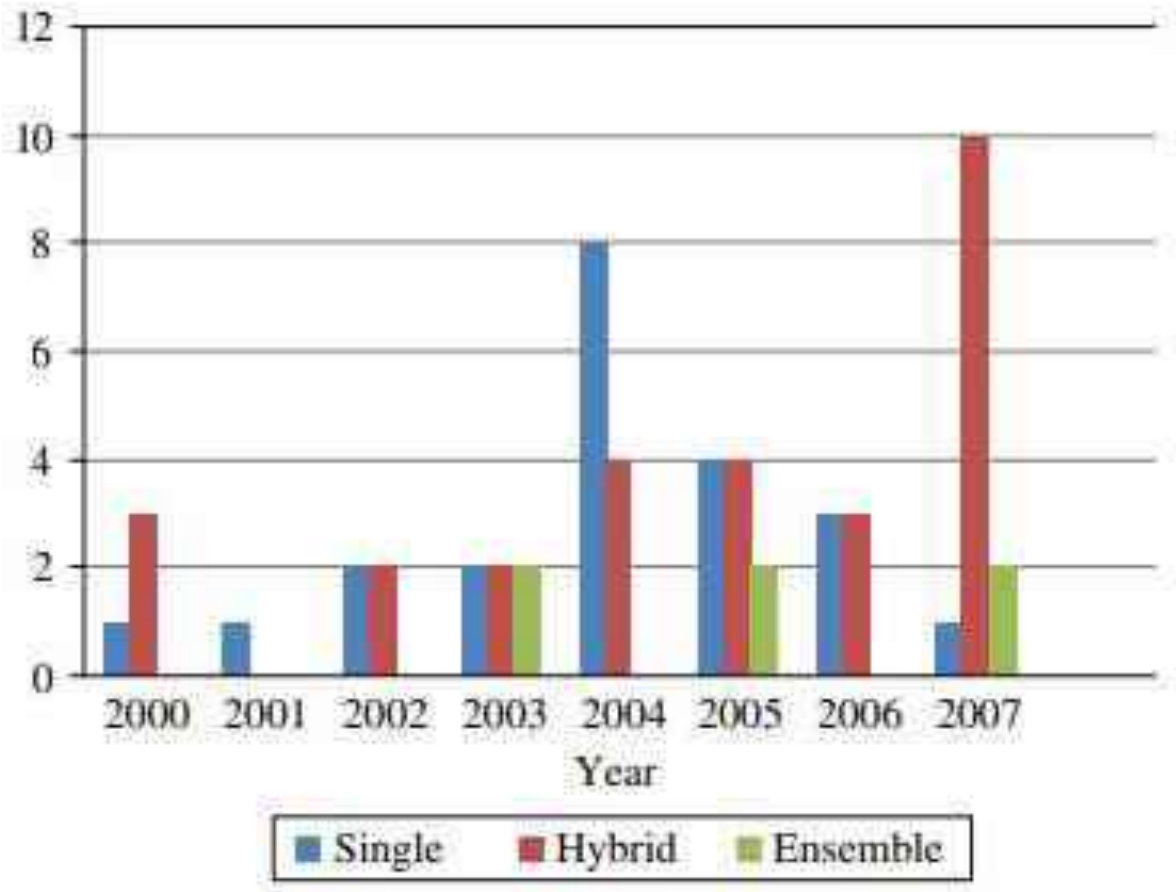

Figure 12: Répartition annuelle des articles pour les types de conception de classificateurs

3/ Classificateurs

Les classificateurs sont des algorithmes d'apprentissage automatique (machine learning) qui sont utilisés pour classer des données dans des catégories spécifiques en fonction de leurs caractéristiques. Les classificateurs sont utilisés dans de nombreux domaines, tels que la reconnaissance d'images, la reconnaissance vocale, l'analyse de texte, la détection de spam et la prédiction des résultats financiers.

3/1. Classificateurs uniques

Un classificateur développé par un algorithme d'apprentissage machine simple est connu sous le nom de classificateur unique ou classificateur simple. Les techniques d'apprentissage machine suivantes sont souvent utilisées comme classificateurs uniques.

a) Arbre de décision (Decision tree)

Un arbre de décision classe une étiquette inconnue à partir d'une liste de décisions. L'arbre de décision est très simple et facile à mettre en oeuvre. Il est souvent utilisé comme classificateur unique. Un arbre de décision est divisé en arbre de régression et arbre de classification.

b) Naive Bayes

Naive Bayes montre de meilleures performances en classification en raison des relations simples. Il ne réalise qu'un seul balayage des données et la classification est donc facile. Naive Bayes effectue une probabilité conditionnelle en fonction de l'étiquette de classe donnée.

c) Random Forest

Le Random Forest est un algorithme d'apprentissage supervisé qui crée un ensemble de modèles d'arbres de décision et agrège leurs prédictions pour améliorer la précision globale et la stabilité du modèle. Chaque arbre de décision est construit à partir d'un sous-ensemble aléatoire des données d'entraînement et d'un sousensemble aléatoire des variables explicatives, ce qui permet de réduire l'overfitting et d'augmenter la généralisation du modèle.

d) K-plus proches voisins (KNN)

KNN est l'approche la plus simple pour la classification d'échantillons. Ici, différentes mesures de distance sont utilisées pour classer les échantillons. Le K-plus proches voisins trouve le nombre d'échantillons à partir des données d'entraînement qui sont proches de l'échantillon de test et attribue l'étiquette de classe la plus fréquente.

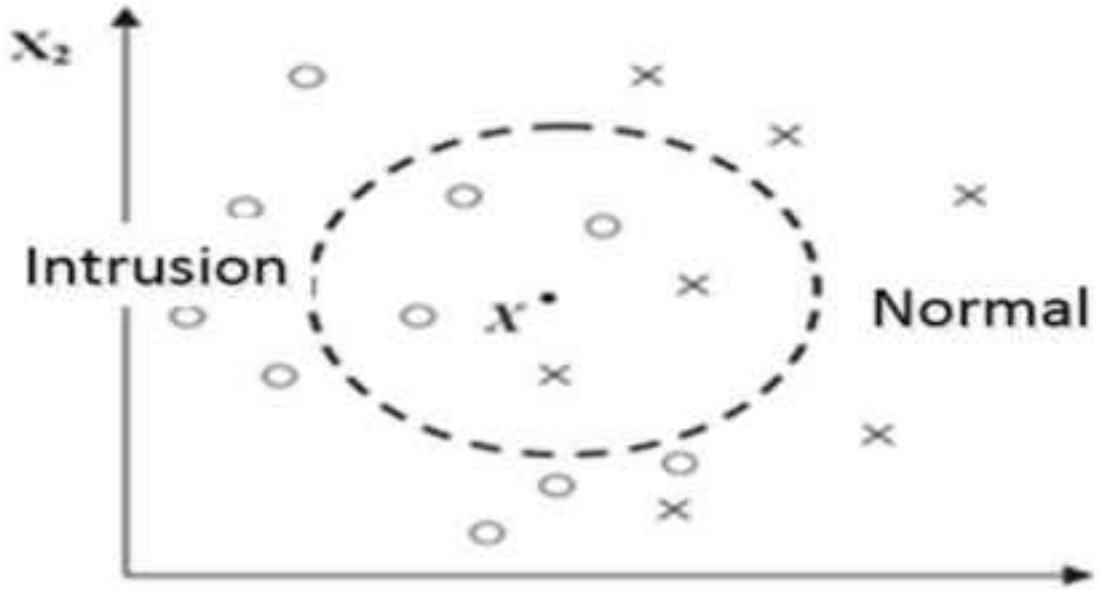

Figure 13: Un exemple de classification par k-plus proches voisins pour k=5 [6].

e) Réseau de neurones artificiels

Le concept de réseau de neurones artificiels (ANN) est pris de la matière Biologie où un réseau neuronal joue un rôle important dans notre corps humain. ANN est un système informatique contenant une grande collection d'unités qui sont interconnectées d'une manière qui permet la communication entre les unités. Ces unités sont appelées des noeuds ou des neurones qui sont de simples processeurs fonctionnant en parallèle. Dans ANN, les couches d'entrée sont connectées aux couches cachées qui sont associées à des poids permettant le traitement. Ces couches cachées sont connectées aux couches de sortie pour obtenir les résultats.

3/2. Classificateurs hybrides

La combinaison d'un ou plusieurs algorithmes d'apprentissage automatique pour améliorer les performances d'un système est appelée classificateur hybride. Les algorithmes supervisés ou non supervisés entrent dans la catégorie des classificateurs hybrides. Un classificateur est conçu sur la base de techniques de regroupement afin d'éliminer les échantillons étiquetés inconnus et de reconnaître les motifs. Les principaux algorithmes utilisés dans les classificateurs hybrides sont l'arbre de décision, l'algorithme génétique, l'algorithme du plus proche voisin et la machine à vecteurs de support. Les classificateurs hybrides permettent d'améliorer la précision et les performances de classification en combinant plusieurs techniques d'apprentissage automatique

3/3. Classificateurs ensemblistes

La combinaison de plusieurs "apprenants faibles" pour améliorer les performances du classificateur est appelée classificateur ensembliste. Boosting, stacking et Bagging sont des méthodes pour créer des classificateurs ensemblistes. Certaines applications de classificateurs ensemblistes sont la reconnaissance d'écriture manuscrite, l'évaluation de la qualité des données, la gestion des données de capteurs manquantes, la cartographie des habitats et la prédiction de la croissance des algues.

4/ Apprentissage profond

4/1. Introduction au Apprentissage profond

L'apprentissage profond est un domaine de recherche récent en apprentissage automatique qui vise à rapprocher la discipline de son but ultime : la création d'une intelligence articielle. Il se caractérise par l'utilisation d'algorithmes inspirés du fonctionnement et de la structuree du cerveau humain pour apprendre à modéliser des relations complexes entre des données à plusieurs niveaux de représentation. En résumé, l'apprentissage profond se concentre sur le développement d'algorithmes sophistiqués capables d'apprendre de manière autonome à partir de données complexes et qui pourraient contribuer de manière signicative à la création d'une intelligence articielle puissante**[22]**

4/2.Les types de réseaux de neurones profonds

Les deux types de réseaux de neurones profonds utilisés dans le domaine de l'apprentissage en profondeur (deep learning) les plus connus sont :

2.1Le réseau de neurones convolutif (CNN)

Un réseau de neurones convolutif est constitué d'une ou plusieurs couches de convolution, suivies d'une ou plusieurs couches complètement connectées comme dans un réseau de neurones multicouche standard (Vasan et al., 2020b). Un réseau de neurones convolutif contient une couche d'entrée et une couche de sortie, ainsi que plusieurs couches cachées. Les couches cachées d'un CNN contiennent généralement une séquence decouches de convolution qui convoluent avec une multiplication. Un CNN reçoit une entrée 2D et extrait des caractéristiques de haut niveau via une séquence de couches cachées. Les CNN, qui améliorent l'architecture des réseaux de neurones courants, bénéficient des caractéristiques spatiales (Vasan et al., 2020c). Les caractéristiques spatiales sont généralement des types de caractéristiques de trafic appliquées dans le domaine des IDS. Lors de l'application de caractéristiques spatiales, le trafic réseau est transformé en images de trafic ; ensuite, la technique de classification d'image est utilisée pour catégoriser les images de trafic, ce qui permet finalement de détecter le trafic intrusif. Cette technique est relativement récente, mais de nombreuses études récentes prouvent son grand potentiel. Par exemple, Vasan, et al. (Vasan et al., 2020b) ont adopté des techniques

de CNN et ont transformé le binaire de logiciel malveillant brut en images en niveaux de gris et en couleur, puis ont appliqué une architecture de CNN fine-tunée.

2.2Le réseau de neurones récurrent (RNN)

Le réseau de neurones récurrent peut fonctionner efficacement sur une série de données de longueurs variables. Cela signifie que les RNN utilisent l'information de leur état antérieur comme entrée pour leur prédiction actuelle, et nous pouvons répéter ce processus pour un nombre arbitraire d'étapes permettant au réseau de propager l'information via son état caché à travers le temps. C'est essentiellement comme donner à un réseau de neurones une mémoire à court terme. Cette fonctionnalité rend les RNN très efficaces pour travailler avec des séquences de données qui se produisent au fil du temps. Yin, et al. (Yin et al., 2017) [9] ont proposé une approche d'apprentissage profond pour la détection d'intrusion en utilisant des réseaux de neurones récurrents (RNN-IDS). Leurs résultats expérimentaux montrent que RNN-IDS est très approprié pour créer des IDS avec une grande précision et que ses performances sont supérieures à celles des méthodes de classification d'apprentissage machine traditionnelles, à la fois pour la classification binaire et multiclasse.

5/Apprentissage automatique vs Apprentissage profond

Les experts en apprentissage automatique et en apprentissage profond n'ont pas encore trouvé de consensus sur ces concepts. Dans ce contexte, de nouvelles idées sont discutées presque chaque jour. L'apprentissage automatique est un concept plus ancien que l'apprentissage profond. L'apprentissage profond peut également être appelé une technique qui effectue de l'apprentissage automatique. Les différences sont énumérées cidessous :

1) En apprentissage profond, beaucoup de données sont nécessaires pour amener la structure de l'algorithme à l'idéal. En apprentissage automatique, le problème peut être

résolu avec beaucoup moins de données, car la personne donne des fonctionnalités spécifiques à l'algorithme.

2) Les algorithmes d'apprentissage profond tentent d'extraire des fonctionnalités des données. En apprentissage automatique, les fonctionnalités sont déterminées par l'expert.
3) Alors que les algorithmes d'apprentissage profond travaillent sur des machines à haute performance, les algorithmes d'apprentissage automatique peuvent fonctionner sur des CPUs ordinaires.
4) En apprentissage automatique, le problème est généralement divisé en morceaux, ces parties sont résolues les unes après les autres, puis les solutions sont formées en résultat des solutions. En apprentissage profond, le problème est résolu de bout en bout.
5) Il faut beaucoup de temps pour entraîner des algorithmes d'apprentissage profond

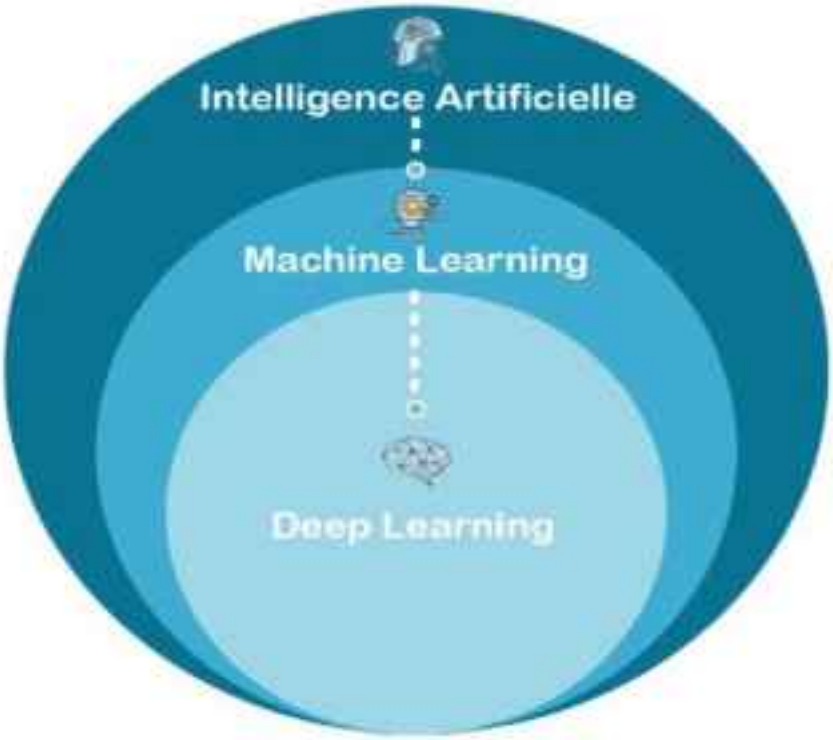

Figure 14: Différence entre l'apprentissage automatique et l'apprentissage profond

Chapitre 04 : implémentation et expérimentation

Tables Des Matiers :

1-Python.....51

2- Concernant la détection des application Android malveillantes 54

a-datasets

b-les models d'apprentssage utilisés

c-model d'apprentissage profond

3-code final avec méthodes de splitting (expliquation) 72

4- Output avec résultat final 86

5-Analyse et Comparison.....99

Listes Des Tableau et Figures :

Figure 15: KAGGLE Android Malware Dataset for machine learning.....54

Figure 16: KAGGLE Drebin_DATASET 55

Figure 17:regréssion logistque exemple concret 61

Figure 18:SVM exemple concret.....65

Figure 19:random forest exemple concret.....68

Figure 20:Python Libraries.....73

Figure 21:KFold cross validation iinsertion.....74

Tableau 03 :KAGGLE Android Malware Detection DATACARD 55

Tableau 04 :Drebin_DATASET Description.....56

Tableau 05 :Logistic Regression Hyperparametres.....60

Tableau 06 :les fonctions Kernel.....62

Tableau 07 :SVM hyperparamétres.....63

Tableau 08 :random forest classification vs régression.....66

Tableau 09 :random forest hyperparamétres.....67

Tableau 10 :caractéristique d'MLP.....70

Tableau 10 :MLP Hyperparamétres config.....71

1/Python (Introduction et Définition) :

-Python est un langage de programmation de haut niveau, polyvalent et interprété, reconnu pour son accent sur la lisibilité du code, sa simplicité et sa facilité d'écriture grâce à l'utilisation d'indentation significative, des noms en « anglais clair » et une vaste bibliothèque standard (« batteries included »).

-Python a été créé par **Guido van Rossum** à la fin des années 1980 (le développement a commencé en décembre 1989) au Centrum Wiskunde & Informatica (CWI) aux Pays-Bas, comme successeur du langage ABC. La première version (0.9.0) a été publiée en février 1991. Le nom « Python » provient de la série comique britannique *Monty Python's Flying Circus*, et non du serpent.

1/1.Caractéristiques principales :

Langage interprété : Le code est exécuté ligne par ligne sans compilation préalable, ce qui accélère le développement et le prototypage.

Syntaxe lisible : L'indentation (espaces significatifs) remplace les accolades, rendant le code proche de la langue naturelle.

Typage dynamique : Pas besoin de déclarer explicitement les types de variables ; gestion automatique de la mémoire via le ramassage de déchets (garbage collection).

Multi-paradigmes : Supporte la programmation procédurale, orientée objet (OOP), fonctionnelle et même impérative.

Portabilité : Fonctionne sur Windows, macOS, Linux et bien d'autres plateformes.

Écosystème riche : Bibliothèque standard complète + des dizaines de milliers de packages via pip (PyPI).

Selon la documentation officielle, Python est un langage puissant et facile à apprendre, avec des structures de données de haut niveau efficaces et une approche simple de la programmation orientée objet. Il est idéal pour le scripting et le développement rapide d'applications.

Évolution : Guido van Rossum a été le « Benevolent Dictator For Life » (BDFL) jusqu'en 2018. Aujourd'hui, le développement est géré par la Python Software Foundation et une communauté active. Les versions majeures incluent Python 2 (obsolète) et Python 3 (actuel, avec des améliorations majeures comme Unicode natif). **[22.23]**

1/ 2.Concepts Fondamentaux :

Variables et Types de Données : Entiers, flottants, chaînes, listes, tuples, dictionnaires, ensembles. Exemple : ma_liste = [1, 2, 3].

Structures de Contrôle : if-elif-else, boucles for et while, compréhensions puissantes ([x**2 for x in range(10)]).

Fonctions : Fonctions de première classe, lambdas, décorateurs, générateurs (yield).

Programmation Orientée Objet : Classes, héritage, polymorphisme, encapsulation.

Modules et Paquets : Organisation du code (import math).

Gestion des Exceptions : try-except-finally pour une robustesse accrue.

Autres : Gestion de fichiers, concurrence (threading, asyncio), intégration avec C/C++ pour les parties critiques en performance. **[22.23]**

1/3. Pourquoi Python est un Choix Raisonnable (et Souvent Préféré) en Apprentissage Automatique (Machine Learning) et Apprentissage Profond (Deep Learning) :

Python est devenu le *langage standard de facto* dans le domaine de l'IA pour de multiples raisons documentées :

3.1.Simplicité et Productivité :

Sa syntaxe claire permet aux chercheurs et développeurs de se concentrer sur les algorithmes plutôt que sur le code boilerplate. Cela accélère considérablement le prototypage, l'expérimentation et l'itération — essentiels en ML où des centaines de modèles sont testés rapidement.

3.2.Écosystème Exceptionnel de Bibliothèques :

-Manipulation de données : NumPy, Pandas, SciPy.

-Visualisation : Matplotlib, Seaborn, Plotly.

-ML classique : scikit-learn.

-Deep Learning : TensorFlow/Keras, PyTorch (très apprécié en recherche grâce à son style impératif et ses graphes dynamiques), JAX.

-Ces bibliothèques offrent des APIs de haut niveau tout en s'appuyant sur des backends optimisés (C/C++/CUDA/Fortran).

3.4.Communauté et Support :

Communauté open-source massive, tutoriels abondants, modèles pré-entraînés (Hugging Face), et code de recherche souvent publié directement en Python. Soutien fort des grandes entreprises (Google, Meta, Microsoft, OpenAI).

3.5.Flexibilité et Évolutivité :

Facile à intégrer avec d'autres langages/outils, passage du prototypage à la production (avec FastAPI, Flask, Docker, etc.), et indépendance des plateformes. Faible barrière d'entrée pour les scientifiques non-informaticiens.

3.6.Performance Gérée Efficacement : Bien que Python soit plus lent pour les boucles pures, les bibliothèques déportent les calculs intensifs vers du code optimisé. Les compilateurs JIT et les GPU (via CUDA) comblent largement l'écart.

Statistiques récentes (2024-2025) : Python domine les offres d'emploi en IA (plus de 199 000 mentions aux États-Unis en 2024), est utilisé par ~41 % des développeurs Python pour le ML, et reste en tête des classements de popularité (IEEE, JetBrains, Stack Overflow, Kaggle). **[22.23]**

2/ Concernant la détection des application Android malveillantes :

2/1. Les datasets :

1.1.Android malware dataset for machine learning :



Figure 15: KAGGLE Android Malware Dataset for machine learning

Description Kaggle :

Context

"Mobile malware is malicious software that targets mobile phones or wireless-enabled Personal digital assistants (PDA), by causing the collapse of the system and loss or leakage of confidential information. As wireless phones and PDA networks have become more and more common and have grown in complexity, it has become increasingly difficult to ensure their safety and security against electronic attacks in the form of viruses or other malware."

Content

Dataset consisting of feature vectors of 215 attributes extracted from 15,036 applications (5,560 malware apps from Drebin project and 9,476 benign apps). The dataset has been used to develop and evaluate multilevel classifier fusion approach for Android malware detection, published in the IEEE Transactions on Cybernetics paper 'DroidFusion: A Novel Multilevel Classifier Fusion Approach for Android Malware Detection. The supporting file contains the description of the feature vectors/attributes obtained via static code analysis of the Android apps.

Tableau 3 :KAGGLE Android Malware Detection DATACARD

Data Card | Code (29) | Discussion (0) | Suggestions (0)

transact	API call signature
215 unique values	Manifest Permiss... 53% API call signature 33% Other (30) 14%
onServiceConnected	API call signature
bindService	API call signature
attachInterface	API call signature
ServiceConnection	API call signature
android.os.Binder	API call signature
SEND_SMS	Manifest Permission
Ljava.lang.Class.getCanonicalName	API call signature
Ljava.lang.Class.getMethods	API call signature
Ljava.lang.Class.cast	API call signature

2 .2 .Drebin Dataset :



Figure 16: KAGGLE Drebin_DATASET

Ce dataset est utilisé pour la **détection de malwares Android** par apprentissage automatique. Il permet de :

Comparer différentes approches de détection de malwares

Développer des modèles de classification malveillant/bénin

Évaluer la performance des détecteurs de malwares Android

Performance du détecteur Drebin

Le méthode Drebin originale atteint **94% de détection** des malwares avec peu de faux positifs sur un grand ensemble de données. Certaines études rapportent jusqu'à **99% de précision** avec ce dataset.

Tableau 4 :Drebin_DATASET Description

Caractéristique	Détails
Nom du dataset	`drebin_dataset` sur Kaggle `kaggle`
Traducteur	LikhithAduRusoju (2022) `academia`
Nombre d'applications malveillantes	5 560 applications malveillantes (S) `academia`
Nombre d'applications bénignes	1 694 applications bénignes (B) `academia`
Total d'applications	7 254 applications Android au total `academia`
Familles de malwares	179 familles différentes de malwares `drebin.misec`
Période de collecte	Août 2010 à octobre 2012 `drebin.misec`
Format du fichier	CSV (disponible sur Kaggle) `papers.xxx`
Source originale	Site officiel du projet : drebin.misec.org `papers.xxx`

Ce dataset est considéré comme une **référence clé** dans la littérature de recherche sur la détection de malwares Android.

2/2. Les modéles d'apprentissage automatique utilisé :

2.1. XGBoost :

- *difinition* (pour *Extreme Gradient Boosting*) :

est l'un des algorithmes de Machine Learning les plus populaires et les plus performants pour les données tabulaires ou structurées. Il est fréquemment utilisé pour remporter des compétitions de Data Science (comme sur Kaggle) en raison de sa précision et de sa vitesse.

fonctionnement de XGBoost

XGBoost est basé sur une méthode d'apprentissage ensembliste appelée le **Boosting de Gradient** (Gradient Boosting).

> **Le principe :** Au lieu d'utiliser un seul modèle complexe, l'algorithme combine une multitude de modèles simples appelés "apprenants faibles" (souvent des arbres de décision).

La méthode séquentielle : Ces arbres ne sont pas créés indépendamment. Le modèle crée un premier arbre, puis calcule ses erreurs (les résidus). L'arbre suivant est spécifiquement entraîné pour **corriger les erreurs** de l'arbre précédent.

L'agrégation : Ce processus est répété des centaines de fois. La prédiction finale est la somme (pondérée) des prédictions de tous les arbres construits.

La régularisation : La particularité "extrême" de XGBoost est d'intégrer une régularisation mathématique pour pénaliser les modèles trop complexes et éviter le surapprentissage (overfitting).

Exemple concret : Prédire le prix d'une maison

Imaginons que vous souhaitiez prédire le prix de vente d'une maison en fonction de sa surface, du nombre de chambres et de son année de construction.

1. **Prédiction initiale (Arbre 1) :** XGBoost commence par faire une prédiction de base, qui est simplement la moyenne des prix de toutes les maisons (par exemple, 200 000 €).
2. **Calcul des erreurs :** Si la première maison vaut en réalité 250 000 €, l'erreur (le résidu) est de +50 000 €.
3. **Apprentissage de l'erreur (Arbre 2) :** Un deuxième arbre est créé spécifiquement pour essayer de prédire cette erreur. Il va utiliser les caractéristiques des maisons pour identifier, par exemple, que les grandes surfaces ont tendance à générer une erreur positive de cette ampleur. Si l'arbre prédit +40 000 € pour cette maison, la nouvelle prédiction devient 200 000 + 40 000 = 240 000 €.
4. **Itérations et raffinage :** Le processus se répète. Un troisième arbre essaie de prédire la nouvelle erreur restante (ici, +10 000 €).
5. **Prédiction finale :** La prédiction finale du modèle correspond à la somme du prix initial et des prédictions de toutes ces petites corrections.

Hyperparamètres

-Pour optimiser un modèle XGBoost, il faut ajuster ses paramètres. Ces réglages contrôlent l'apprentissage, évitent le surapprentissage et accélèrent le calcul.

Voici les hyperparamètres les plus importants, classés par catégorie.

1. Les paramètres de contrôle de l'apprentissage

Ces réglages influencent la manière dont le modèle progresse au fil des étapes.

eta (ou learning_rate) : Le taux d'apprentissage. Il réduit le poids de chaque nouvel arbre (valeur typique : 0.01 à 0.3). Une valeur plus faible exige plus d'arbres mais améliore la précision.

n_estimators : Le nombre total d'arbres à construire. Si learning_rate est bas, il faut augmenter ce nombre. On utilise souvent l'arrêt précoce (*early stopping*) pour le stopper automatiquement dès que le modèle n'évolue plus.

2. Les paramètres de complexité des arbres

Ces réglages limitent la taille de chaque arbre de décision pour éviter le surapprentissage (*overfitting*).

max_depth : La profondeur maximale d'un arbre. Une valeur trop élevée crée des arbres trop complexes qui apprennent par cœur (valeur typique : 3 à 10).

min_child_weight : Le poids minimal requis dans un nœud pour créer une nouvelle séparation. Plus cette valeur est grande, plus le modèle est conservateur et évite de créer des règles pour quelques individus seulement.

gamma (ou min_split_loss) : La réduction minimale de la perte requise pour effectuer une nouvelle séparation dans un arbre. Une valeur élevée rend l'algorithme plus strict.

3. Les paramètres d'échantillonnage (Robustesse)

Ces options ajoutent du hasard durant l'entraînement pour rendre le modèle plus robuste face aux données bruitées.

subsample : La proportion de lignes (observations) lues au hasard pour construire chaque arbre. Si fixé à 0.8, l'algorithme utilise 80 % des données à chaque étape.

colsample_bytree : La proportion de colonnes (variables/features) sélectionnées au hasard pour la construction de chaque arbre.

4. La Régularisation mathématique

Ce sont les paramètres qui pénalisent la complexité des feuilles des arbres, caractéristiques uniques de la fonction objectif de XGBoost.

- **alpha** : Régularisation L1 (équivalent au modèle Lasso). Utile pour rendre le modèle plus simple en annulant le poids des variables inutiles.
- **lambda :** Régularisation L2 (équivalent au modèle Ridge). Aide à réduire l'impact global de toutes les variables de manière homogène.

2.2.Logistic regression :

Définition

La régression logistique est un modèle statistique de **classification supervisée** qui estime la probabilité qu'une observation appartienne à une classe spécifique. Bien que son nom contient "régression", elle est principalement utilisée pour des problèmes de **classification binaire** (deux classes), mais peut aussi être étendue à la classification multinomiale (plus de deux classes).

Concept

Fonction mathématique :

La fonction de lien est la **fonction logit** : $\text{logit}(p) = \ln\left(\frac{p}{1-p}\right)$

La fonction sigmoïde transforme la combinaison linéaire en probabilité : $\sigma(z) = \frac{1}{1+e^{-z}}$

Où $z = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_n X_n$

Optimisation :

Utilise la **maximisation de la vraisemblance** (Maximum Likelihood Estimation - MLE)

La fonction de perte est l'**erreur logistique** (log loss) : $\text{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1-y_i)\log(1-p_i)]$

Algorithmes d'optimisation : Newton-Raphson, IRLS (Iteratively Reweighted Least Squares), LBFGS

Types de régression logistique :

Binaire : deux classes (0/1, oui/non)

Multinomiale : plus de deux classes sans ordre

Binomiale : plus de deux classes avec ordre (ex: faible/moyen/fort)

Hyperparamètres

Tableau 5 :Logistic Regression Hyperparametres

Hyperparamètre	Description	Valeurs recommandées
penalty	Type de régularisation	'L1' (sparseté), 'L2' (réduction), 'elasticnet', 'none'
C	Inverse de la force de régularisation	0.01 à 10 (plus petit = plus forte régularisation)
solver	Algorithme d'optimisation	'lbfgs', 'liblinear', 'sag', 'saga', 'newton-cg'
max_iter	Nombre maximum d'itérations	100 à 1000 (plus pour convergence)
class_weight	Poids des classes	'balanced' (pour données déséquilibrées), None
tol	Critère d'arrêt	1e-4 à 1e-3

Régularisation :

L1 (Ridge) : Produit des modèles sparses, sélectionne des caractéristiques

L2 (Lasso) : Réduit les coefficients sans les éliminer complètement

Exemple

Prédiction de diabète (Dataset Pima Indians Diabetes)

```python
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler

# Données : âge, poids, glycémie, pression artérielle, etc.
# Classe : 1 (diabétique), 0 (non diabétique)

model = LogisticRegression(
    penalty='l2',
    C=1.0,
    solver='lbfgs',
    max_iter=1000,
    class_weight='balanced'
)

model.fit(X_train, y_train)
probabilities = model.predict_proba(X_test)
predictions = model.predict(X_test)

# Coefficients montrent l'importance
print(model.coef_)  # Ex: [0.5, -0.3, 0.8, ...]
```

Figure 17 :regréssion logistque exemple concret

Interprétation : Un coefficient de 0.5 pour "glycémie" signifie qu'à chaque augmentation d'1 unité de glycémie, le log-odds de diabète augmente de 0.5.

[24.25]

2.3.SVM(Support Vector Machine / Machine à Vecteurs de Support) :

Définition

Les SVM sont des algorithmes d'apprentissage supervisee basés sur la **théorie de l'apprentissage** qui trouvent l'hyperplan optimal pour séparer les données en classes. Ils sont particulièrement efficaces pour la **classification binaire**, mais peuvent être étendus à la classification multinomiale et à la régression (SVR).

Concept

Sépare les données en classes avec une **frontière optimale** (hyperplan)

Maximalise la **marge** (distance entre l'hyperplan et les points les plus proches)

Géométrie de séparation :

Hyperplan : $w^T x + b = 0$ où w est le vecteur normal et b le biais

Marge : distance entre l'hyperplan et les points les plus proches (vecteurs de support)

Objectif : Maximiser la marge $\frac{2}{\|w\|}$ tout en minimisant les erreurs

Formulation mathématique :

Problème d'optimisation original (primal) :

$$\min_{w,b} \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{N} \xi_i$$

$$\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

où ξ_i sont les variables de marge (soft margin) et C le paramètre de régularisation

Problème dual (avec multiplicateurs de Lagrange) :

$$\max_{\alpha} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{s.t. } 0 \leq \alpha_i \leq C, \sum_{i=1}^{N} \alpha_i y_i = 0$$

Les **vecteurs de support** sont les points de données les plus proches de la frontière

Utilise des **fonctions de noyau** (kernel) pour transformer les données dans un espace de plus haute dimension quand la séparation n'est pas linéaire

Tableau 6 :les fonctions Kernel

Kernel	Fonction	Paramètres	Usage
Linear	$K(x_i, x_j) = x_i^T x_j$	None	Données linéairement séparables
Polynomial	$K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$	degreeId), gamma, r	Relations polynomialles
RBF	$K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$	gamma	Données non-linéaires (le plus utilisé)
Sigmoid	$K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$	gamma, r	Similarité neural network

Soft Margin vs Hard Margin :

Hard margin : Aucune erreur tolérée (C → ∞), risque de surapprentissage

Soft margin : Erreurs tolérées (C fini), meilleur pour données bruitées

Hyperparamètres

Tableau 7 :SVM hyperparamétres

Hyperparamètre	Description	Valeurs recommandées
kernel	Type de fonction de noyau	'linear', 'poly', 'rbf', 'sigmoid'
C	Paramètre de régularisation (soft margin)	0.1 à 100 (plus grand = moins d'erreurs tolérées)
gamma	Influence d'un seul point de training (RBF/Poly)	0.001 à 10 ('scale', 'auto', 'qr')
degree	Degré du noyau polynomial	2 à 5 (généralement 3)
coef0	Terme indépendant dans kernel	0 à 1 (pour poly/sigmoid)
class_weight	Poids des classes	'balanced', None
tol	Critère d'arrêt	1e-3 à 1e-4

Guides de réglage :

C grand : Modèle complexe, risque de surapprentissage

C petit : Modèle simple, risque de sous-apprentissage

gamma grand : Chaque point a influence locale (frontière complexe)

gamma petit : Influence globale (frontière simple)

Exemple

```python
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler

# Données : longueur/largeur des pétales et feuilles
# Classes : 3 espèces d'iris (setosa, versicolor, virginica)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

model = SVC(
    kernel='rbf',
    C=10.0,
    gamma='scale',
    class_weight='balanced'
)

model.fit(X_train_scaled, y_train)
predictions = model.predict(X_test)

# Vecteurs de support
print(model.support_vectors_)  # Points critiques pour la frontière
```

Figure 18 :SVM exemple concret

Visualisation de la frontière :

```python
# Pour 2 caractéristiques, créer une grille
xx, yy = np.meshgrid(np.linspace(X.min(), X.max(), 100),
                     np.linspace(X.min(), X.max(), 100))
Z = model.predict(np.c_[xx.ravel(), yy.ravel()])
```

[26.27]

2.4.Random Forest(Forêt aléatoire) :

Définition

Le Random Forest est un algorithme d'apprentissage **d'ensemble** (ensemble learning) qui combine **multiple arbres de décision** entraînés indépendamment. Développé formellement par **Leo Breiman en 2001** (bien que les concepts initiaux de Taiga Ho en

1995), il utilise le **bagging** (bootstrap aggregating) et la sélection aléatoire de caractéristiques pour réduire la variance et éviter le surapprentissage.

Concept

Principe fondamental :

Ensemble d'arbres : $f(x) = \frac{1}{B}\sum_{b=1}^{B} T_b\ (x)$ où T_b est le b-th arbre

Bagging : Chaque arbre est entraîné sur un sous-ensemble aléatoire du dataset (bootstrap)

Sélection de caractéristiques : À chaque division, seulement m caractéristiques sont considérées (généralement $m = \sqrt{p}$ pour classification)

Algorithmes de construction :

#Pourchaquearbreb=1aB

1.Creerunsous-ensemblebootstrapZ_b（sizeN,avecremise

2.Entrainerunarbredededecisionsur Z_b:

-Achagueneud.selectionnermcaracteristiguesaleatoires

-Trouverlameiileuredivisionparmicesmcaracteristigues

-Diviseriusau'aprofondeurmaxouneudsminimums

3.Nepasfairedepruning(lesarbressontgrands

Classification vs Régression :

Tableau 8 :random forest classification vs régression

Type	Prédiction	Variance réduite
Classification	Mode des prédictions : $\hat{y} = \text{mode}(T_1(x), \ldots, T_B(x))$	$\text{Var} = \rho\sigma^2 + \frac{1-\rho}{B}(1-\rho)\sigma^2$
Régression	Moyenne : $\hat{y} = \frac{1}{B}\sum_{b=1}^{B} T_b(x)$	$\text{Var} = \rho\sigma^2 + \frac{\sigma^2}{B}$

où ρ = corrélation entre arbres, σ^2 = variance individuelle

Importance des caractéristiques :

Calculée par la réduction moyenne du gini (ou entropie) :

$$\text{Importance}(X_j) = \frac{1}{B}\sum_{b=1}^{B}\sum_{t\in\text{nœuds}(X_j)}\Delta G(t)$$

où $\Delta G(t)$ est la réduction de l'indice de gini au nœud t

Matrice de proximité (Proximity) :

Deux observations sont "proximes" si elles tombent souvent dans le même nœud final

Utilisée pour clustering, détection d'anomalies

Hyperparamètres

Hyperparamètres principaux à optimiser :

Tableau 9 : random forest hyperparamétres

Hyperparamètre	Description	Valeurs recommandées
n_estimators	Nombre d'arbres dans la forêt	100 à 500 (plus = plus stable, mais plus lent) stat4decision
max_depth	Profondeur maximale des arbres	5 à 20 (None = arbres complets) stat4decision
max_features	Nombre de caractéristiques par division	'auto' ($\sqrt{p}$), 'sqrt', 'log2', fraction stat4decision
criterion	Fonction pour mesurer la qualité	'gini', 'entropy', 'log_loss' (classification) stat4decision
min_samples_split	Minimum d'observations pour diviser	2 à 10 (plus grand = arbre plus simple) stat4decision
min_samples_leaf	Minimum d'observations dans un nœud feuille	1 à 10 (plus grand = régularisation) stat4decision
bootstrap	Utiliser bootstrap pour under-sets	True (standard), False (Poisson RF) stat4decision
class_weight	Poids des classes	'balanced', 'balanced_subsample' stat4decision

Guides de réglage :

n_estimators : Augmenter jusqu'à stabilisation de la performance (courbe d'apprentissage)

max_features : Plus petit = plus de variance réduite mais arbres plus corrélés

max_depth : Control principal du surapprentissage (plus petit = moins de surapprentissage)

Avancés :

max_leaf_nodes : Limiter le nombre de nœuds feuilles

warm_start : Ajouter des arbres à une forêt existante

oob_score : Estimation de la performance sur données hors-boostrap

Exemple

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split

# Données : surface, nombre de chambres, localisation, âge, etc.
# Target : prix de la maison

rf = RandomForestRegressor(
    n_estimators=200,
    max_depth=15,
    max_features='sqrt',
    min_samples_split=5,
    min_samples_leaf=2,
    bootstrap=True,
    random_state=42
)

rf.fit(X_train, y_train)

# Prédictions
predictions = rf.predict(X_test)

# Importance des caractéristiques
importances = rf.feature_importances_
for feat, imp in zip(feature_names, importances):
    print(f"{feat}: {imp:.4f}")

# Sortie exemple :
# surface: 0.3521
# chambres: 0.1843
# localisation: 0.2156
```

Figure 19 :random forest exemple concret

[28.29]

2.3. Le modéle d'apprentissage profond utilisé :

- Deep Multy Layer Perception(MLP) :

Le modèle Deep Multi-Layer Perceptron (MLP) est une architecture fondamentale des réseaux de neurones artificiels, caractérisée par une succession de couches entièrement connectées qui permettent de modéliser des relations non linéaires complexes entre les données d'entrée et les sorties. Il est largement utilisé en apprentissage supervisé pour des tâches de classification et de régression.

Structure du MLP

Couches d'entrée : reçoivent les données brutes (par exemple, des vecteurs numériques).

Couches cachées : composées de neurones interconnectés, elles appliquent des transformations linéaires suivies de fonctions d'activation non linéaires (ReLU, sigmoid, tanh).

Ces couches permettent au réseau de capturer des représentations hiérarchiques et abstraites.

Couche de sortie : produit la prédiction finale (par exemple, une probabilité pour chaque classe en classification).

Profondeur : un Deep MLP se distingue par la présence de plusieurs couches cachées, ce qui accroît sa capacité de modélisation mais augmente aussi le risque de surapprentissage et les besoins computationnels.

Fonctionnement

Propagation avant (Forward pass) : les données traversent successivement les couches, chaque neurone calculant une combinaison linéaire des entrées suivie d'une activation.

Fonction de coût : mesure l'écart entre la prédiction et la vérité terrain (ex. entropie croisée pour la classification).

Rétropropagation (Backpropagation) : calcule les gradients de la fonction de coût par rapport aux paramètres du réseau.

Optimisation : mise à jour des poids via des algorithmes comme Stochastic Gradient Descent ou Adam.

Caractéristiques principales

Tableau 10 :caractéristique d'MLP

Aspect	Description
Capacité de représentation	Peut approximer toute fonction continue (théorème d'approximation universelle).
Profondeur	Plus de couches = plus de puissance expressive, mais aussi plus de complexité d'entraînement.
Applications	Reconnaissance d'images, traitement du langage naturel, prévisions financières, biomédecine.
Limites	Sensible au surapprentissage, nécessite beaucoup de données et de puissance de calcul.

[30]

Hyperparamètres

Les **hyperparamètres d'un Deep Multi-Layer Perceptron (MLP)** sont des paramètres définis *avant* l'entraînement et qui influencent directement la performance, la vitesse de convergence et la capacité de généralisation du modèle. Voici une présentation académique des principaux hyperparamètres :

Hyperparamètres essentiels

Nombre de couches cachées (depth) Détermine la profondeur du réseau. Plus de couches permettent de capturer des représentations complexes, mais augmentent le risque de surapprentissage et la difficulté d'optimisation.

Nombre de neurones par couche (width) Contrôle la capacité de représentation. Trop peu de neurones limitent la puissance du modèle, trop de neurones entraînent une complexité excessive.

Fonction d'activation

- ReLU : rapide et efficace, réduit le problème du gradient qui disparaît.

- Sigmoid/Tanh : utiles pour certaines tâches mais sujets à saturation.

- Variantes modernes (Leaky ReLU, GELU, Swish) améliorent la stabilité.

Taux d'apprentissage (learning rate) Paramètre critique de l'optimisation. Un taux trop élevé empêche la convergence, trop faible ralentit l'entraînement.

Taille du batch (batch size) Influence la stabilité du gradient et la vitesse d'entraînement. Petits batchs → plus de bruit mais meilleure généralisation ; grands batchs → calculs plus rapides mais risque de surapprentissage.

Nombre d'époques (epochs) Définit combien de fois le réseau voit l'ensemble des données. Trop d'époques → surapprentissage ; trop peu → sous-apprentissage.

Algorithme d'optimisation

- SGD : simple mais sensible au choix du taux d'apprentissage.
- Adam, RMSProp : adaptatifs, souvent plus efficaces pour les MLP profonds.

Régularisation

- Dropout : réduit le surapprentissage en désactivant aléatoirement des neurones.
- L2/L1 weight decay : pénalise les poids trop grands.
- Early stopping : arrête l'entraînement avant surapprentissage.

Exemple de configuration typique

Tableau 11 :MLP Hyperparamétres config

Hyperparamètre	Valeur courante
Couches cachées	3–5
Neurones/couche	128–512
Activation	ReLU
Learning rate	0.001 (Adam)
Batch size	32–128
Régularisation	Dropout (0.2–0.5)

[31]

3/code final avec méthodes de splitting (expliquation) :

3/1. Introduction au plateforme KAGGLE :

Kaggle est une plateforme en ligne communautaire incontournable dédiée à la science des données et à l'intelligence artificielle. Acquise par Google, elle sert de point de rencontre mondial pour les chercheurs, développeurs et entreprises, permettant de s'entraîner, de trouver des bases de données et de participer à des concours de modélisation prédictive.

Voici les piliers qui définissent la plateforme :

Les Compétitions : Des entreprises et organisations publient des problèmes réels (ex: prédire des pannes, détecter des fraudes) et offrent souvent des récompenses financières ou des opportunités de recrutement. Les participants s'affrontent pour créer le modèle d'intelligence artificielle le plus précis possible.

Les Jeux de données (Datasets) : La plateforme héberge une immense bibliothèque de données publiques et gratuites (allant de la finance à la santé) que vous pouvez utiliser pour vos propres projets.

Les Notebooks : Kaggle fournit des environnements de travail dans le cloud (avec des outils comme Jupyter, Python ou R) prêts à l'emploi. Cela permet de coder sans surcharger son propre ordinateur et de partager ses codes avec la communauté.

La Formation : Le site propose de nombreux cours en ligne gratuits pour apprendre les bases du Machine Learning, du Deep Learning et de la manipulation de données.

3/2. Les bibliothéques utilisées :

```python
import numpy as np
import pandas as pd
import os
from sklearn.model_selection import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import (
    accuracy_score, precision_score, recall_score,
    f1_score, roc_auc_score, confusion_matrix, classification_report
)
import xgboost as xgb
import warnings
warnings.filterwarnings('ignore')
```

Figure 20 :Python Libraries

numpy (importé en tant que **np**) : C'est la librairie de référence pour le calcul numérique en Python. Elle permet de manipuler facilement des tableaux (arrays) multidimensionnels et d'effectuer des opérations mathématiques complexes très rapidement.

pandas (importé en tant que **pd**) : Utilisée pour la manipulation et l'analyse de données. Elle structure les données sous forme de "DataFrames" (des tableaux similaires à ceux d'Excel), ce qui facilite le filtrage, le nettoyage et la préparation des données.

sklearn.model_selection.StratifiedKFold : Un outil de validation croisée. Il divise les données en K parties (plis) tout en s'assurant que la proportion de chaque classe (ex: nombre de malwares vs logiciels sains) est conservée dans chaque pli.

sklearn.ensemble.RandomForestClassifier : Implémente l'algorithme de "Forêt Aléatoire" (Random Forest), un modèle d'apprentissage automatique qui combine plusieurs arbres de décision pour améliorer la précision et éviter le surapprentissage.

sklearn.svm.SVC : Implémente les "Machines à Vecteurs de Support" (SVM), un algorithme puissant pour trouver la frontière optimale qui sépare les différentes classes.

sklearn.linear_model.LogisticRegression : Implémente la Régression Logistique. Malgré son nom, c'est un algorithme de classification (binaire ou multiclasse) qui modélise la probabilité qu'une donnée appartienne à une classe.

sklearn.metrics : Un ensemble de fonctions pour évaluer les performances des modèles :

accuracy_score : Le pourcentage de prédictions correctes.

precision_score : Parmi les prédictions positives, combien étaient vraiment positives.

recall_score : Parmi les vrais positifs, combien ont été correctement identifiés.

f1_score : La moyenne harmonique de la précision et du rappel (excellent pour les données déséquilibrées).

roc_auc_score : L'aire sous la courbe ROC, mesurant la capacité du modèle à distinguer les classes.

confusion_matrix : Un tableau résumant les vrais positifs, faux positifs, vrais négatifs et faux négatifs.

xgboost (importé en tant que **xgb**) : "eXtreme Gradient Boosting". Une librairie très performante et rapide basée sur les arbres de décision, souvent utilisée pour gagner des compétitions de Machine Learning.

warnings : Permet de gérer les avertissements (warnings) générés par Python. Ici, on les désactive pour garder la console propre lors de l'exécution.

3/3.Méthodes de splitting :

- **KFold Cross Validation :**

```
def run_kfold(model_name, X, y, k=5):
    kf = StratifiedKFold(n_splits=k, shuffle=True, random_state=42)
    oof_preds = np.zeros(len(y), dtype=int)
    oof_probs = np.zeros(len(y))
    metrics = {'acc': [], 'prec': [], 'rec': [], 'f1': [], 'auc': []}
```

Figure 21 :KFold cross validation iinsertion

Prépare la validation croisée en 5 plis.

Crée deux tableaux vides de la taille de **y** pour stocker les prédictions finales "Out-Of-Fold" (OOF) : une pour les classes prédites (0 ou 1), l'autre pour les probabilités (ex: 0.85).

Initialise un dictionnaire pour stocker les métriques de chaque pli.

```
for fold, (train_idx, val_idx) in enumerate(kf.split(X, y)):
    model = get_model(model_name)
    model.fit(X.iloc[train_idx], y[train_idx])
    preds = model.predict(X.iloc[val_idx])
    probs = model.predict_proba(X.iloc[val_idx])[:, 1]
```

Figure 22 :stockage des métrique

Boucle sur les 5 plis. À chaque tour :

Récupère un modèle neuf.

L'entraîne sur les données d'entraînement de ce pli (**train_idx**).

Prédit les classes et les probabilités sur les données de validation de ce pli (**val_idx**).

```
    oof_preds[val_idx] = preds
    oof_probs[val_idx] = probs

    metrics['acc'].append(accuracy_score(y[val_idx], preds))
    # ... (idem pour prec, rec, f1)
    metrics['auc'].append(roc_auc_score(y[val_idx], probs) if len(np.unique(y[val_idx])) == 2 else np.nan)
return oof_preds, oof_probs, metrics
```

Figure 23 :OOF tableau remplissement (predictions,probabilités,metrics)

Remplit les tableaux OOF aux bonnes positions (ainsi, à la fin, on a une prédiction pour chaque ligne du dataset sans fuite de données).

Calcule les métriques pour ce pli et les ajoute au dictionnaire. (Le **if** sur l'AUC empêche le code de planter si par hasard un pli ne contenait qu'une seule classe).

Retourne tout à la fin de la boucle.

- **Holdout (70train 15test 15val) :**

```
def run_presplit(model_name, X_train, y_train, X_test, y_test):
    model = get_model(model_name)
    model.fit(X_train, y_train)

    preds = model.predict(X_test)
    probs = model.predict_proba(X_test)[:, 1]
```

Figure 24 :Holdout methode

Plus simple : récupère un modèle(déjà divisé a l'aide de la méthode **train_test_split**), l'entraîne strictement sur le jeu d'entraînement (70%), et prédit strictement sur le jeu de test (15%).

3/4.Expliquation du code :

1 – Apprentssage Automatique :

- **Fonction de nettoyage des données**

```
def clean_drebin(df):
    TARGET = 'class'
```

Figure 25 :fnct nettoyage de données

Crée une fonction pour nettoyer le dataset DREBIN. Définit le nom de la colonne cible (ce qu'on veut prédire) comme étant 'class'.

```
df[TARGET] = df[TARGET].astype(str).str.strip()
valid_classes = ['S', 'B', '1', '0', 1, 0]
df = df[df[TARGET].isin(valid_classes)].copy()
```

Figure 26 :filtrage de données

Convertit la colonne cible en texte et enlève les espaces inutiles. Filtre le DataFrame pour ne garder que les lignes où la classe fait partie des valeurs valides ('S' pour Malware/Suspicious, 'B' pour Benign/Bon, ou leurs équivalents numériques 1 et 0).

```
class_map = {'B': 0, '0': 0, 0: 0, 'S': 1, '1': 1, 1: 1}
df[TARGET] = df[TARGET].map(class_map)
df = df.dropna(subset=[TARGET])
df[TARGET] = df[TARGET].astype(int)
```

Figure 27 :standardisement des étiquettes

Standardise les étiquettes : transforme tout en 0 (Bénin) ou 1 (Malware). Supprime les lignes où il y aurait des valeurs manquantes dans la cible, puis convertit la colonne en nombres entiers.

```
X = df.drop(columns=[TARGET])
X = X.apply(pd.to_numeric, errors='coerce')
X = X.dropna(axis=1, how='all')
X = X.fillna(0).astype(np.int8)
y = df[TARGET].values
return X, y
```

Figure 28 :séparation et nettoyage des caractérstique

Sépare les caractéristiques (**X**) de la cible (**y**).

Force toutes les colonnes de **X** à être numériques (les erreurs deviennent des valeurs nulles **NaN**).

Supprime les colonnes de **X** qui sont entièrement vides.

Remplace les **NaN** restants par des 0, et convertit les données en **int8** (très léger en mémoire, parfait pour des données binaires comme les permissions Android).

Retourne **X** (les features) et **y** (les étiquettes).

- **Usine à modèles (Factory)**

```
def get_model(model_name):
    if model_name == 'RandomForest':
        return RandomForestClassifier(n_estimators=200, max_depth=15, n_jobs=-1, random_state=42)
    elif model_name == 'LogisticRegression':
        return LogisticRegression(C=1.0, max_iter=1000, n_jobs=-1, random_state=42)
    elif model_name == 'SVM':
        return SVC(kernel='linear', C=1.0, probability=True, random_state=42)
    elif model_name == 'XGBoost':
        return xgb.XGBClassifier(n_estimators=300, learning_rate=0.05, max_depth=6,
                                 subsample=0.8, colsample_bytree=0.8, tree_method='hist',
                                 eval_metric='logloss', random_state=42, n_jobs=-1)
```

Figure 29 :Usine a modéles

Une fonction qui prend le nom d'un modèle en texte et retourne l'objet algorithmique correspondant, pré-configuré avec des hyperparamètres optimisés pour ce type de tâche. **random_state=42** assure que les résultats sont reproductibles, **n_jobs=-1** utilise tous les cœurs du processeur pour aller plus vite.

1.Random Forest (Forêt Aléatoire)

n_estimators=200 : C'est le nombre d'arbres de décision qui composeront la forêt. Plus il y a d'arbres, plus le modèle est stable et précis, mais plus il prend du temps à calculer. 200 est un excellent compromis pour ce type de dataset (la valeur par défaut est 100).

max_depth=15 : C'est la profondeur maximale de chaque arbre. Sans cette limite, l'arbre pourrait créer une branche pour chaque ligne exacte du dataset

(surapprentissage absolu). Limiter à 15 force l'arbre à trouver des règles générales, ce qui améliore la capacité du modèle à généraliser sur de nouveaux malwares inconnus.

n_jobs=-1 : Indique à Scikit-Learn d'utiliser tous les cœurs du processeur disponibles sur la machine en parallèle. Étant donné que chaque arbre est construit indépendamment, c'est un énorme gain de temps.

random_state=42 : Fixe la graine aléatoire. L'algorithme fait des tirages aléatoires (pour sélectionner les features à chaque nœud). Mettre 42 assure que si vous relancez le code, vous aurez exactement les mêmes résultats.

2. Régression Logistique

C=1.0 : C'est le paramètre de régularisation (l'inverse de la force de régularisation

). Dans un contexte avec des milliers de features (comme les permissions Android), il y a du bruit. **C=1.0** est la valeur par défaut qui offre un bon équilibre. *(Note : si C était très petit, ex: 0.01, le modèle ignorerait presque toutes les features, ce qui est trop simple. Si C était grand, ex: 100, il essaierait de tout utiliser, même le bruit)*.

max_iter=1000 : La régression logistique résout un problème d'optimisation mathématique de manière itérative. La valeur par défaut (100) s'arrête souvent trop tôt sur des datasets complexes et lance un avertissement **ConvergenceWarning**. Mettre 1000 donne largement assez de temps à l'algorithme pour trouver la meilleure solution.

n_jobs=-1 : Parallélise le calcul derrière les scènes (utile surtout si le solver gère le multi-cœur).

random_state=42 : Assure la reproductibilité du mélange des données en arrière-plan.

3. SVM (Support Vector Machine)

kernel='linear' : C'est le choix le plus crucial ici. Le noyau "linéaire" cherche à tracer une ligne droite (ou un hyperplan) pour séparer les malwares des bons logiciels. Sur les datasets DREBIN (des milliers de colonnes binaires 0/1), le noyau linéaire est presque toujours le meilleur. Un noyau "rbf" (qui fait des cercles) prendrait un temps infini à calculer sans meilleur résultat.

C=1.0 : Comme pour la régression logistique, c'est le compromis entre avoir une "marge" large et bien classer les points d'entraînement. 1.0 est standard.

probability=True : Très important dans votre code. Par défaut, un SVM ne renvoie que la classe prédite (0 ou 1). Pour calculer l'AUC plus tard dans votre code, vous avez besoin de la *probabilité* (ex: 85% de chance d'être un malware). Ce paramètre active une méthode interne (Platt Scaling) pour estimer ces probabilités (au prix d'un peu de temps de calcul supplémentaire).

random_state=42 : Utilisé ici pour le calcul interne des probabilités.

4. XGBoost

n_estimators=300 : Le nombre d'arbres consécutifs. Contrairement à Random Forest où les arbres sont indépendants, ici le 2ème arbre essaie de corriger les erreurs du 1er, le 3ème les erreurs des deux précédents, etc. 300 est un nombre assez élevé nécessaire quand on utilise un petit **learning_rate**.

learning_rate=0.05 (aussi appelé *eta*) **:** Le taux d'apprentissage. Au lieu de corriger les erreurs à 100% à chaque arbre, on les corrige à 5%. Cela rend l'apprentissage lent mais extrêmement précis et empêche le modèle de "brûler les étapes" et de surapprendre. (Valeur par défaut souvent autour de 0.3, mais 0.05 est un choix d'expert pour la performance pure).

max_depth=6 : La profondeur des arbres. Dans le Boosting, les arbres doivent rester "faibles" (shallow trees) pour que le système fonctionne bien. 6 est la valeur par défaut de XGBoost et se révèle quasi toujours optimale.

subsample=0.8 : À chaque création d'un nouvel arbre, XGBoost ne prendra que 80% des lignes au hasard. Cela introduit du "bruit" contrôlé qui force le modèle à ne pas se fier à des lignes spécifiques, réduisant considérablement le surapprentissage (technique du *Stochastic Gradient Boosting*).

colsample_bytree=0.8 : Similaire à **subsample**, mais pour les colonnes. À chaque arbre, il n'utilisera que 80% des features (permissions) au hasard. Cela empêche quelques permissions très dominantes d'écraser toutes les autres dans la prise de décision.

tree_method='hist' : L'algorithme de construction des arbres. **'hist'** (histogramme) est une méthode rapide récente. Au lieu de tester chaque valeur possible pour couper un nœud, il regroupe les valeurs en "bacs" (histogrammes). C'est beaucoup plus rapide et utilise beaucoup moins de mémoire RAM que la méthode exacte, particulièrement vital sur des gros datasets comme le 15k.

eval_metric='logloss' : La fonction de perte (Loss function). C'est la formule mathématique que le modèle essaie de minimiser à chaque étape. La *log-loss* (ou entropie croisée) punit sévèrement le modèle lorsqu'il est très confiant mais se trompe (ex: prédire 99% de probabilité que c'est un malware alors que c'est bénin).

n_jobs=-1 et **random_state=42** : Même utilisation que pour les autres modèles (parallélisation et reproductibilité).

En rlation avec la méthode kfold :

```
metrics = {
    'acc': [accuracy_score(y_test, preds)],
    # ... (idem pour prec, rec, f1)
    'auc': [roc_auc_score(y_test, probs) if len(np.unique(y_test)) == 2 else np.nan]
}
return preds, probs, metrics
```

Figure 30 :calcule des métrics pour KFold config

Calcule les métriques sur le test set*, elles sont mises dans des listes **[...]** pour garder le même format de données que la méthode K-Fold et pouvoir utiliser les mêmes fonctions d'affichage.*

- **Fonction d'affichage**

```
def print_best_results(ds_name, method_name, best_model_name, y_true, y_pred, metrics):
    # ... Prints décoratifs ...
    print(f"Accuracy:  {np.nanmean(metrics['acc']):.4f}")
    # ... (idem pour les autres métriques)
```

Figure 31 :affichage

Affiche un joli résumé pour le meilleur modèle d'un dataset. **np.nanmean** calcule la moyenne des métriques en ignorant les valeurs nulles (utile pour l'AUC si jamais il y en avait une).

```
cm = confusion_matrix(y_true, y_pred)
print(f"                Pred Benign  Pred Malware")
print(f"True Benign    {cm[0,0]:<12} {cm[0,1]:<12}")
print(f"True Malware   {cm[1,0]:<12} {cm[1,1]:<12}\n")
```

Figure 32 :Matrice de Confusion

Calcule la matrice de confusion et l'affiche sous forme d'un tableau formaté texte pour voir rapidement les Faux Positifs et Faux Négatifs.

- **Initialisation globale**

```
final_comparison = []
MODELS = ['RandomForest', 'LogisticRegression', 'SVM', 'XGBoost']
```

Figure 33 :Initialisation globale

Crée une liste vide qui servira à construire le tableau récapitulatif final.

Définit la liste des 4 modèles à tester.

- **SECTION 1 : Exécution du K-Fold**

```
print("\n" + "="*80)
print("          SECTION 1: METHODOLOGY 1 - K-FOLD CROSS VALIDATION")
print("="*80)

for ds_name, path in ORIGINAL_PATHS.items():
    print(f"\n▶▶▶ Processing: {ds_name}")
    X, y = clean_drebin(pd.read_csv(path))
```

Figure 34 :Exécution du Section 1 Méthode 1

Affiche l'en-tête de la section 1.

Boucle sur les datasets originaux (7k et 15k). Pour chacun, il lit le CSV et le nettoie avec la fonction créée plus tôt.

```
best_f1 = 0
best_results = None

for m_name in MODELS:
    print(f"  Running {m_name}...", end=" ")
    oof_preds, oof_probs, metrics = run_kfold(m_name, X, y)
    mean_f1 = np.nanmean(metrics['f1'])
    print(f"F1: {mean_f1:.4f} | AUC: {np.nanmean(metrics['auc']):.4f}")
```

Figure 35 :Stockage Des Données de modéles

Initialise des variables pour garder en mémoire le meilleur modèle de ce dataset.

Boucle sur les 4 modèles. Lance la validation croisée, extrait le F1-score moyen et l'affiche en temps réel.

```
    if mean_f1 > best_f1:
        best_f1 = mean_f1
        best_results = (m_name, oof_preds, oof_probs, metrics)

print_best_results(ds_name, "K-Fold Method", best_results[0], y, best_results[1], best_results[3])
```

Figure 36 :Comparison Entre Modéles

Si le modèle actuel a un meilleur F1-score que les précédents, il devient le "best_results".

Une fois les 4 modèles testés, appelle la fonction d'affichage pour montrer les détails du vainqueur.

```
final_comparison.append({
    "Dataset": ds_name, "Method": "1. K-Fold", "Best Model": best_results[0],
    "Accuracy": np.nanmean(best_results[3]['acc']), "Precision": np.nanmean(best_results[3]['prec']),
    "Recall": np.nanmean(best_results[3]['rec']), "F1-Score": np.nanmean(best_results[3]['f1']),
    "AUC": np.nanmean(best_results[3]['auc'])
})
```

Figure 37 :Meilleur Modéle

Ajoute un dictionnaire contenant un résumé des performances du meilleur modèle dans la liste **final_comparison** (qui servira pour le tableau final).

- **SECTION 2 : Exécution du Pre-Split (70/15/15)**

La logique est rigoureusement la même que la Section 1, seules les différences sont commentées ci-dessous

```
X_train, y_train = clean_drebin(pd.read_csv(paths["train"]))
X_test, y_test   = clean_drebin(pd.read_csv(paths["test"]))

print(f"  Train size: {len(X_train)} | Test size: {len(X_test)}")
```

Figure 38 :Exécution du Holdout Méthode

Ici, au lieu de charger un seul fichier, on charge séparément les fichiers d'entraînement et de test, et on affiche leur taille pour vérifier que le split est bien respecté.

```
test_preds, test_probs, metrics = run_presplit(m_name, X_train, y_train, X_test, y_test)
```

On appelle la fonction **run_presplit** au lieu de **run_kfold**. Le reste de la boucle (comparaison des F1, sauvegarde dans **final_comparison**) est identique.

- **SECTION 3 : Verdict Final**

```
results_df = pd.DataFrame(final_comparison)
float_cols = ['Accuracy', 'Precision', 'Recall', 'F1-Score', 'AUC']
results_df[float_cols] = results_df[float_cols].round(4)

print(results_df.to_string(index=False))
```

Figure 39 :Tableau de Comparison

Transforme la liste final_comparison en un vrai tableau Pandas (DataFrame).

Arrondit les colonnes de métriques à 4 chiffres après la virgule pour que ce soit lisible.

Affiche le tableau complet dans la console sans l'index numérique de Pandas.

```
best_idx = results_df['F1-Score'].idxmax()
best_row = results_df.iloc[best_idx]
```

Figure 40 :Meilleur Score

Cherche l'index de la ligne qui contient le F1-score le plus élevé de tout le tableau (toutes méthodes et datasets confondus).

Extrait cette ligne spécifique.

```
print("\n" + "="*80)
print(" 🏆 ABSOLUTE BEST RESULT 🏆")
print("="*80)
print(f"Dataset:        {best_row['Dataset']}")
# ... (imprime les autres champs du meilleur résultat)
print("="*80)
```

Figure 41 :Affichage Final (Résultat Absolu)

Affiche le verdict final de manière très visuelle en imprimant le nom du dataset gagnant, la méthode gagnante, le modèle gagnant et ses scores.

4/Output avec comparaison final :

```
================================================================================
==========
      SECTION 1: METHODOLOGY 1 - K-FOLD CROSS VALIDATION
================================================================================
==========

▶▶▶ Processing: Dataset 1 (7k)
 Running RandomForest... F1: 0.9809 | AUC: 0.9961
 Running LogisticRegression... F1: 0.9832 | AUC: 0.9944
 Running SVM... F1: 0.9833 | AUC: 0.9936
 Running XGBoost... F1: 0.9873 | AUC: 0.9972

************************************************************
BEST MODEL for Dataset 1 (7k) [K-Fold Method]: XGBoost
```

****

Accuracy: 0.9806

Precision: 0.9857

Recall: 0.9890

F1-Score: 0.9873

AUC: 0.9972

Confusion Matrix:

	Pred Benign	Pred Malware
True Benign	1614	80
True Malware	61	5499

►►► Processing: Dataset 2 (15k)

Running RandomForest... F1: 0.9735 | AUC: 0.9976

Running LogisticRegression... F1: 0.9675 | AUC: 0.9960

Running SVM... F1: 0.9705 | AUC: 0.9956

Running XGBoost... F1: 0.9814 | AUC: 0.9982

****

BEST MODEL for Dataset 2 (15k) [K-Fold Method]: XGBoost

****

Accuracy: 0.9864

Precision: 0.9880

Recall: 0.9750

F1-Score: 0.9814

AUC: 0.9982

Confusion Matrix:

	Pred Benign	Pred Malware
True Benign	9410	66
True Malware	139	5421

==========

SECTION 2: METHODOLOGY 2 - PRE-SPLIT (70/15/15 HOLDOUT)

==========

▶▶▶ Processing: Dataset 1 (7k)

Train size: 5077 | Test size: 1089

Running RandomForest... F1: 0.9792 | AUC: 0.9969

Running LogisticRegression... F1: 0.9857 | AUC: 0.9969

Running SVM... F1: 0.9850 | AUC: 0.9968

Running XGBoost... F1: 0.9868 | AUC: 0.9981

****

BEST MODEL for Dataset 1 (7k) [70/15/15 Method]: XGBoost

****

Accuracy: 0.9798

Precision: 0.9868

Recall: 0.9868

F1-Score: 0.9868

AUC: 0.9981

Confusion Matrix:

	Pred Benign	Pred Malware
True Benign	243	11
True Malware	11	824

▶▶▶ Processing: Dataset 2 (15k)

Train size: 10525 | Test size: 2256

Running RandomForest... F1: 0.9701 | AUC: 0.9978

Running LogisticRegression... F1: 0.9637 | AUC: 0.9958

Running SVM... F1: 0.9655 | AUC: 0.9953

Running XGBoost... F1: 0.9789 | AUC: 0.9980

****

BEST MODEL for Dataset 2 (15k) [70/15/15 Method]: XGBoost

****

Accuracy: 0.9845

Precision: 0.9854

Recall: 0.9724

F1-Score: 0.9789

AUC: 0.9980

Confusion Matrix:

	Pred Benign	Pred Malware
True Benign	1410	12
True Malware	23	811

FINAL COMPARISON: SPLITTING METHODS VERDICT

Dataset	Method	Best Model	Acc	Pre	Recall	F1	AUC
Dataset1(7k)	K-Fold	XGBoost	0.9806	0.9857	0.9890	0.9873	0.9972
Dataset2(15k)	K-Fold	XGBoost	0.9864	0.9880	0.9750	0.9814	0.9982
Dataset1	Pre-Split	XGBoost	0.9798	0.9868	0.9868	0.9868	0.9981
Dataset2	Pre-Split	XGBoost	0.9845	0.9854	0.9724	0.9789	0.9980

🏆 ABSOLUTE BEST RESULT 🏆

Dataset	Dataset1
Winning method	KFold
Winning model	XGBoost
F1-Score	0.9873
AUC	0.9972

2 – Apprentssage Profond :

Nous avons utilisé les étape et les méme méthode de Splitting

```
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'

import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings('ignore')

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
```

Figure 42 :MLP Code Lib

os : Une librairie standard de Python permettant d'interagir avec le système d'exploitation. Ici, elle est utilisée pour configurer une variable d'environnement afin de contrôler le comportement de TensorFlow.

tensorflow (importé en tant que **tf**) : La librairie de référence créée par Google pour le calcul numérique et l'apprentissage profond (Deep Learning).

tensorflow.keras.models.Sequential : C'est le conteneur principal. Un modèle "Séquentiel" empile les couches de neurones les unes après les autres de manière linéaire (comme un sandwich).

tensorflow.keras.layers : Contient les différents types de couches :

- **Dense** : La couche de neurones classique (chaque neurone est connecté à tous les neurones de la couche précédente).
- **Dropout** : Une couche de régularisation (expliquée plus bas).
- **BatchNormalization** : Une couche qui normalise les données (expliquée plus bas).

tensorflow.keras.callbacks.EarlyStopping : Un "callback" (une fonction déclenchée automatiquement à un moment précis) qui permet d'arrêter l'entraînement du réseau de neurones avant la fin s'il n'apprend plus rien.

*(Les librairies **numpy, pandas, warnings** et les métriques de **sklearn** sont les mêmes qu'avant).*

Explication du code :

Configuration de l'environnement :

```
python

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '3'
```

Cache les messages d'avertissement de bas niveau ("INFO" et "WARNING") générés par le moteur C++ de TensorFlow. Cela permet de garder une console propre, car TensorFlow est très bavard au démarrage.

Modification dans le nettoyage des données :

```
python

  X = X.fillna(0).astype(np.float32)
```

Différence majeure avec le ML classique : Au lieu de convertir les features en **int8** (comme pour Random Forest), on les convertit en **float32** (nombres à virgule). Les réseaux de neurones effectuent des multiplications de matrices avec des "poids" qui sont des décimales ; ils ont absolument besoin de floats pour fonctionner. Le **float32** est le standard (il prend 4 octets en mémoire, contrary au **float64** qui en prendrait 8 pour un gain de temps minime ici).

Création de l'Architecture du Réseau de Neurones (MLP) :

```
python

def create_mlp(input_dim):

  """Creates the Deep Learning Architecture"""

  model = Sequential([
```

Crée un modèle séquentiel vide. "MLP" signifie *Multi-Layer Perceptron*, le nom technique d'un réseau de neurones classique (entièrement connecté).

```
python

    Dense(256, activation='relu', input_dim=input_dim),

    BatchNormalization(), Dropout(0.4),
```

Dense(256...) : Première couche cachée avec 256 neurones. **input_dim** dit au réseau combien de features (permissions) il va recevoir en entrée. **relu** est la fonction d'activation (elle met à 0 les valeurs négatives, ce qui accélère l'apprentissage).

BatchNormalization() : Normalise les sorties de la couche précédente pour qu'elles aient une moyenne proche de 0 et un écart-type proche de 1. Cela stabilise énormément l'entraînement des réseaux profonds.

Dropout(0.4) : À chaque passage d'une donnée (batch), cette couche "éteint" aléatoirement 40% des neurones. Cela empêche le réseau de mémoriser par cœur le dataset (surapprentissage/overfitting) et le force à trouver des motifs généraux.

python

```
    Dense(128, activation='relu'),
    BatchNormalization(), Dropout(0.3),
    Dense(64, activation='relu'),
    BatchNormalization(), Dropout(0.2),
```

Deux autres blocs identiques. On remarque que l'architecture est en entonnoir (256 -> 128 -> 64). Le réseau compresse progressivement l'information pour en extraire l'essence. Le Dropout diminue aussi (0.4 -> 0.3 -> 0.2) car la couche de sortie n'a pas besoin d'être trop perturbée.

python

```
    Dense(1, activation='sigmoid')
])
```

Couche de sortie : 1 seul neurone, car on veut une prédiction binaire (Malware ou Non). La fonction d'activation **sigmoid** écrase la sortie finale pour qu'elle soit toujours comprise strictement entre 0 et 1. Cette sortie est interprétée comme une probabilité (ex: 0.92 = 92% de chances d'être un malware).

python

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
              loss='binary_crossentropy', metrics=['accuracy'])
```

```
return model
```

Prépare le modèle pour l'entraînement.

Adam : L'optimiseur le plus utilisé. Il ajuste les poids du réseau intelligemment. **learning_rate=0.001** est la taille du pas à chaque ajustement.

binary_crossentropy : La fonction de perte (Loss). C'est la formule mathématique qui calcule l'erreur entre la probabilité prédite (ex: 0.85) et la vraie étiquette (1). Le but du réseau est de minimiser ce chiffre.

Exécution de la Section 1 (K-Fold avec DL) :

```
python

    X_tr, X_va = X.iloc[train_idx].values, X.iloc[val_idx].values
```

.values : Transforme les DataFrames Pandas en tableaux Numpy purs. C'est obligatoire pour TensorFlow, qui n'accepte pas les DataFrames en entrée (pour des raisons de performance).

```
python

    tf.keras.backend.clear_session()
```

INDISPENSABLE EN DL : Dans une boucle K-Fold, on crée 5 modèles successivement. Les réseaux de neurones prennent énormément de mémoire (GPU/RAM). Cette ligne vide la mémoire à chaque fin de tour pour éviter un debordement de mémoire (Out Of Memory - OOM error).

```
python

    model = create_mlp(X_tr.shape[1])

    model.fit(X_tr, y_tr, epochs=30, batch_size=64, verbose=0)
```

Lance l'entraînement.

epochs=30 : Le réseau va parcourir l'intégralité des données d'entraînement 30 fois de suite.

batch_size=64 : Au lieu de regarder toutes les données d'un coup (trop lourd), le réseau prend des petits paquets de 64 lignes, ajuste ses poids, puis prend les 64 suivantes.

verbose=0 : Cache la barre de progression de l'entraînement (très longue si on l'affiche 5 fois par modèle).

```
python

    va_probs = model.predict(X_va, verbose=0).flatten()

    va_preds = (va_probs >= 0.5).astype(int)
```

model.predict() génère les probabilités sur le jeu de validation. **.flatten()** transforme le résultat en une simple liste à 1 dimension.

Seuil de décision : Si la probabilité est >= 0.5, on arrondit à 1 (Malware), sinon à 0 (Bénin).

Exécution de la Section 2 (Pre-Split avec DL) :

```
python

  early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True, verbose=0)
```

C'est ici que le jeu de validation (15%) prend tout son sens en Deep Learning.

monitor='val_loss' : Surveille l'erreur (loss) sur le jeu de validation à la fin de chaque *epoch*.

patience=5 : Si l'erreur du jeu de validation augmente (le modèle commence à faire du surapprentissage) pendant 5 époques consécutives, l'entraînement s'arrête automatiquement.

restore_best_weights=True : C'est le plus important. Si le modèle s'arrête à l'époque 40 (parce qu'il a empiré pendant 5 époques), cette instruction restaure les poids (la "mémoire") du modèle tels qu'ils étaient à la meilleure époque (la 35ème).

```
python
  model.fit(X_train, y_train, validation_data=(X_val, y_val),
        epochs=50, batch_size=64, callbacks=[early_stop], verbose=0)
```

L'entraînement est lancé pour un maximum de 50 époques. On lui fournit **validation_data** pour qu'il puisse calculer le loss de validation à chaque epoch. Le **callbacks=[early_stop]** est activé pour appliquer la règle d'arrêt définie au-dessus.

Note :

(La fin du code, l'évaluation sur le test set et la création du tableau comparatif final, utilisent les exactes mêmes logiques que dans notre précédent code d'Apprentissage auto)

Output :

```
==========================================================================
            SECTION 1: 5-FOLD CROSS VALIDATION
==========================================================================

▶▶▶ Working on: Dataset 1 (7k Original) ◀◀◀
Data Shape: (7254, 215) | Classes: [1694 5560]
  Fold 1: Acc=0.9793 | F1=0.9866 | AUC=0.9950
  Fold 2: Acc=0.9883 | F1=0.9924 | AUC=0.9987
  Fold 3: Acc=0.9869 | F1=0.9915 | AUC=0.9983
  Fold 4: Acc=0.9869 | F1=0.9914 | AUC=0.9932
  Fold 5: Acc=0.9786 | F1=0.9861 | AUC=0.9975

  MEAN RESULTS:
  Accuracy:  0.9840
```

Precision: 0.9882

Recall: 0.9910

F1-Score: 0.9896

AUC: 0.9965

Confusion Matrix (OOF):

	Pred Benign	Pred Malware
True Benign	1628	66
True Malware	50	5510

▶▶▶ Working on: Dataset 2 (15k Original) ◀◀◀

Data Shape: (15036, 215) | Classes: [9476 5560]

Fold 1: Acc=0.9910 | F1=0.9878 | AUC=0.9972

Fold 2: Acc=0.9927 | F1=0.9901 | AUC=0.9978

Fold 3: Acc=0.9920 | F1=0.9892 | AUC=0.9985

Fold 4: Acc=0.9864 | F1=0.9815 | AUC=0.9987

Fold 5: Acc=0.9900 | F1=0.9865 | AUC=0.9988

MEAN RESULTS:

Accuracy: 0.9904

Precision: 0.9897

Recall: 0.9844

F1-Score: 0.9870

AUC: 0.9982

Confusion Matrix (OOF):

	Pred Benign	Pred Malware
True Benign	9419	57
True Malware	87	5473

```
======================================================================
            SECTION 2: PRE-SPLIT EVALUATION (70/15/15)
======================================================================

▶▶▶ Working on: Dataset 1 (7k Pre-Split) ◀◀◀
Train: 5077 | Val: 1088 | Test: 1089
Training model (watching Val set for Early Stopping)...
Evaluating on unseen Test set...

 TEST SET RESULTS:
 Accuracy:  0.9844
 Precision: 0.9892
 Recall:    0.9904
 F1-Score:  0.9898
 AUC:       0.9989
 Confusion Matrix (Test Set):
          Pred Benign  Pred Malware
True Benign   245         9
True Malware  8           827

▶▶▶ Working on: Dataset 2 (15k Pre-Split) ◀◀◀
Train: 10525 | Val: 2255 | Test: 2256
Training model (watching Val set for Early Stopping)...
Evaluating on unseen Test set...

 TEST SET RESULTS:
 Accuracy:  0.9898
```

Precision: 0.9915

Recall: 0.9808

F1-Score: 0.9861

AUC: 0.9982

Confusion Matrix (Test Set):

	Pred Benign	Pred Malware
True Benign	1415	7
True Malware	16	818

```
#######################################################################
#          FINAL COMPREHENSIVE COMPARISON TABLE          #
#######################################################################
```

Dataset	Method	Accuracy	Precision	Recall	F1-Score	AUC
Dataset 1 (7k Original)	5-Fold CV	0.9836	0.9891	0.9896	0.9893	0.9959
Dataset 2 (15k Original)	5-Fold CV	0.9899	0.9893	0.9833	0.9863	0.9976
Dataset 1 (7k Pre-Split)	Pre-Split	0.9844	0.9916	0.9880	0.9898	0.9985
Dataset 2 (15k Pre-Split)	Pre-Split	0.9880	0.9927	0.9748	0.9837	0.9969

```
#######################################################################
```

Figure 43 :Table de Comparison final MLP entre KFold et Holdout

5/Analyse et Comparison :

5/1.Discussion des résultats du modèle XGBoost

Les résultats du modèle **XGBoost** montrent des performances très élevées sur les deux jeux de données (7k et 15k) avec les deux méthodes de validation.

1. Validation croisée (K-Fold)

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dataset1 (7k)	0.9806	0.9857	0.9890	0.9873	0.9972
Dataset2 (15k)	0.9864	0.9880	0.9750	0.9814	0.9982

Analyse

- L'accuracy dépasse 98 %, indiquant une excellente capacité de classification.
- Le rappel (Recall) est particulièrement élevé sur Dataset1 (0.9890), ce qui signifie que très peu d'échantillons positifs sont manqués.
- Les valeurs de F1-score supérieures à 0.98 démontrent un bon équilibre entre précision et rappel.
- Les AUC proches de 1 (0.9972 et 0.9982) indiquent une excellente séparation entre les classes.

2. Pré-division (Pre-Split)

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dataset1	0.9798	0.9868	0.9868	0.9868	0.9981
Dataset2	0.9845	0.9854	0.9724	0.9789	0.9980

Analyse

Les performances restent très proches de celles obtenues avec la validation croisée :

- L'écart d'accuracy est inférieur à 1 %.
- Les valeurs d'AUC demeurent supérieures à 0.998, ce qui confirme la robustesse du modèle.
- La stabilité des résultats montre que XGBoost généralise efficacement et n'est pas affecté par le choix de la méthode de validation.

5/2.Discussion des résultats du modèle MLP

Validation croisée (5-Fold CV)

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dataset1 (7k)	0.9836	0.9891	0.9896	0.9893	0.9959

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dataset2 (15k)	0.9899	0.9893	0.9833	0.9863	0.9976

Pré-Split

Dataset	Accuracy	Precision	Recall	F1-score	AUC
Dataset1 (7k)	0.9844	0.9916	0.9880	0.9898	0.9985
Dataset2 (15k)	0.9880	0.9927	0.9748	0.9837	0.9969

Analyse

- Le réseau de neurones MLP atteint également des performances très élevées.
- Les accuracies varient entre 98.36 % et 98.99 %.
- Les valeurs de précision sont légèrement supérieures à celles de XGBoost, atteignant jusqu'à 99.27 %.
- Les F1-scores sont proches de 0.99, révélant un excellent compromis entre précision et rappel.
- Les AUC restent supérieures à 0.995, ce qui traduit une forte capacité discriminante.

5/3.Comparaison entre XGBoost et MLP

Accuracy

Dataset	XGBoost	MLP
7k CV	0.9806	**0.9836**
15k CV	0.9864	**0.9899**
7k Pre-Split	0.9798	**0.9844**
15k Pre-Split	0.9845	**0.9880**

Le MLP présente une accuracy légèrement supérieure sur tous les jeux de données.

Precision

Le MLP obtient les meilleures valeurs :

- Maximum de **0.9927** contre 0.9880 pour XGBoost.

Cela signifie que le MLP produit moins de faux positifs.

Recall

Dataset	XGBoost	MLP
7k CV	0.9890	**0.9896**
15k CV	0.9750	**0.9833**
7k Pre-Split	0.9868	0.9880
15k Pre-Split	0.9724	**0.9748**

Le MLP obtient généralement un rappel légèrement supérieur.

F1-score

Les deux modèles présentent des F1-scores très élevés (>0.978), mais le MLP possède un léger avantage :

- F1 maximal MLP : **0.9898**
- F1 maximal XGBoost : 0.9873

AUC

XGBoost :

- 0.9972 – 0.9982

MLP :

- 0.9959 – 0.9985

Les deux modèles montrent une excellente capacité de discrimination. La différence est très faible.

Conclusion

Les résultats montrent que les deux approches offrent d'excellentes performances, avec des accuracies proches de 99 % et des valeurs d'AUC proches de 1.

- **XGBoost** présente une grande stabilité et une excellente capacité de généralisation. Il est robuste et fournit des résultats cohérents indépendamment de la méthode de validation utilisée.
- **MLP** obtient globalement les meilleures valeurs d'accuracy, de précision, de rappel et de F1-score. Ses performances sont légèrement supérieures à celles de XGBoost.

Ainsi, **le modèle MLP apparaît comme le modèle le plus performant dans cette étude**, tandis que **XGBoost constitue une alternative très robuste et compétitive**. Les écarts entre les deux modèles restent toutefois faibles, ce qui indique que les deux méthodes sont capables de fournir une classification fiable et précise.

Conclusion générale

Au terme de ce travail, nous avons étudié la problématique de la détection des applications Android malveillantes, qui constitue aujourd'hui un enjeu majeur de la cybersécurité en raison de la popularité croissante du système Android et de l'évolution constante des menaces informatiques.

Dans un premier temps, nous avons présenté les fondements de la sécurité informatique et les principaux mécanismes de protection des systèmes d'information. Nous nous sommes ensuite intéressés à l'environnement Android, à son architecture, à son modèle de sécurité ainsi qu'aux différentes vulnérabilités pouvant être exploitées par les logiciels malveillants.

Par ailleurs, nous avons étudié les concepts de l'apprentissage automatique et de l'apprentissage profond, ainsi que plusieurs méthodes de classification utilisées dans le domaine de la détection des malwares. Une partie pratique a été consacrée à la mise en œuvre et à l'évaluation de différents modèles, notamment XGBoost et le réseau de neurones MLP. Les résultats expérimentaux obtenus ont montré que ces approches sont capables d'identifier efficacement les applications malveillantes avec un niveau de précision élevé. L'analyse comparative réalisée a permis de mettre en évidence les avantages et les limites de chaque modèle et de déterminer le modèle offrant les meilleures performances sur les données étudiées.

Ainsi, les objectifs fixés au début de ce mémoire ont été atteints. Nous avons réussi à comprendre les mécanismes de sécurité d'Android, à étudier les techniques d'apprentissage automatique adaptées à la détection des malwares et à développer des modèles capables de distinguer les applications malveillantes des applications légitimes avec des résultats satisfaisants.

Cependant, face à l'apparition continue de nouvelles familles de logiciels malveillants et à la sophistication croissante des attaques, la détection des malwares Android demeure un domaine de recherche en constante évolution. À ce titre, plusieurs perspectives peuvent être envisagées pour améliorer ce travail :

- exploiter des bases de données plus importantes et plus récentes afin d'augmenter les capacités de généralisation des modèles ;
- intégrer des caractéristiques statiques et dynamiques des applications afin d'obtenir une détection plus robuste ;
- expérimenter des modèles plus avancés de Deep Learning tels que CNN, LSTM ou Transformers ;
- développer des approches hybrides combinant plusieurs algorithmes pour améliorer les performances de classification ;
- mettre en œuvre un système de détection en temps réel capable d'analyser les applications avant leur installation sur les appareils Android ;
- étudier des méthodes d'intelligence artificielle explicable (Explainable AI) permettant de mieux comprendre les décisions prises par les modèles.

En définitive, ce travail met en évidence l'apport considérable de l'intelligence artificielle et de l'apprentissage automatique dans la lutte contre les menaces visant les systèmes Android. Ces techniques représentent aujourd'hui des solutions prometteuses pour renforcer la sécurité des utilisateurs et faire face aux défis futurs de la cybersécurité.

Références bibliographiques :

ANSSI (France) : Le Guide d'hygiène informatique reste la référence absolue pour les 42 mesures de base, incluant les règles sur les **mots de passe** et les **mises à jour**.**[1]**

CNIL : Pour les aspects de **confidentialité** et de conformité liés au RGPD, ainsi que les recommandations sur l'authentification forte. **[2]**

NIST (États-Unis) : Les publications de la série **SP 800** (notamment le NIST SP 800-207 pour le **Zero Trust**) définissent les standards internationaux d'architecture de sécurité. **[3]**

ANSSI. (2025). *Guide d'hygiène numérique : 42 mesures pour sécuriser son système d'information*. Agence nationale de la sécurité des systèmes d'information. cyber.gouv.fr**[4]**

IBM. (2025). *Qu'est-ce qu'une cyberattaque ?*. Récupéré de ibm.com**[5]**

NIST. (2020). *Zero Trust Architecture* (Special Publication 800-207). National Institute of Standards and Technology. doi.org/10.6028/NIST.SP.800-207**[6]**

Oodrive. (2025). *Cyberattaques 2025 : les 10 menaces clés à surveiller*. oodrive.com **[7]**

ISO. (1989). *Systèmes de traitement de l'information — Interconnexion de systèmes ouverts — Modèle de référence de base — Partie 2: Architecture de sécurité* (ISO 7498-2:1989). Organisation internationale de normalisation. iso.org**[8]**

NIST. (2025). *Security Service - Glossary*. National Institute of Standards and Technology. csrc.nist.gov**[9]**

Ferrag, M. A. (2021). *Sécurité Informatique - Cours et TD*. ResearchGate. researchgate.net**[10]**

Asaf, S., Shabtai, A., Fledel, Y., & Elovici, Y. (2010). *Securing Android-powered mobile devices using SELinux*. IEEE Security and Privacy Workshops. **[11]**

Shabtai, A., Fledel, Y., & Elovici, Y. (2009). *Securing Android-powered mobile devices*. IEEE Security and Privacy, 8(3), 36–44. **[12]**

Enck, W., Ongtang, M., & McDaniel, P. (2009). *On lightweight mobile phone application certification*. Proceedings of the 16th ACM Conference on Computer and Communications Security (CCS). **[13]**

Enck, W., Ongtang, M., & McDaniel, P. (2009). *Understanding Android security*. IEEE Security & Privacy, 7(1), 50–57. **[14]**

Felt, A. P., Chin, E., Hanna, S., Song, D., & Wagner, D. (2011). *Android permissions demystified*. Proceedings of the 18th ACM Conference on Computer and Communications Security (CCS). **[15]**

Zhou, Y., & Jiang, X. (2012). *Dissecting Android malware: Characterization and evolution*. IEEE Symposium on Security and Privacy, 95–109. **[16]**

L. Samuel, "Some studies in machine learning using the game of checkers," IBM Journal of research and development, vol. 3, no. 3, pp. 210–229, 1959. **[17]**

T. M. Mitchell, "Machine learning, volume 1 of 1," 1997. **[18]**

Dif, Nassima: L'apprentissage profond pour le traitement des images, 2020. **[19]**

Zhang, Haining et Seung Ki Moon: Reviews on machine learning approaches for process optimization in noncontact direct ink writing. ACS Applied Materials & Interfaces, 13(45) :5332353345, 2021. **[20]**

Dif, Nassima: L'apprentissage profond pour le traitement des images, 2020**[21]**

Raschka, S., Patterson, J., & Nolet, C. (2020). *Machine Learning in Python: Main Developments and Technology Trends in Data Science, Machine Learning, and Artificial Intelligence*. *Information*, 11(4), 193. (Article très cité qui explique la domination de Python grâce à sa productivité et à son écosystème.) Disponible en open access : https://www.mdpi.com/2078-2489/11/4/193. [22**]**

Sarker, I. H. (2021). *Machine Learning: Algorithms, Real-World Applications and Research Directions*. *SN Computer Science*. (Vue d'ensemble complète des techniques ML/DL, avec contexte sur les outils Python.) **[23]**

Hosmer, D. W., Lemeshow, S., & Stur, D. A. (2013). *Applied Logistic Regression* (3rd ed.). Wiley. ISBN: 978-0471617758**[24]**

Cox, D. R. (1958). "The Regression Analysis of Binary Sequences". *Journal of the Royal Statistical Society, Series B*, 20(2), 325-342. DOI: 10.1111/j.2517-6166.1958.00021.x] **[25]**

Cortes, C., & Vapnik, V. (1995). "Support-Vector Networks". Machine Learning, 20(3), 273-297. DOI: 10.1007/BF00994018**[26]**

Schölkopf, B., & Smola, A. J. (2002). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press. ISBN: 978-0262184528].**[27]**

Breiman, L. (2001). "Random Forests". *Machine Learning*, 45(1), 5-32. DOI: 10.1023/A101093301**[28]**

Ho, T. K. (1995). "Random Decision Forests". *Proceedings of the 3rd International Conference on Document Analysis and Recognition*, 278-282. DOI: 10.1109/ICDAR.1995.598994] **[29]**

Barbier, J., Camilli, F., Nguyen, M.-T., Pastore, M., & Skerk, R. (2022). Statistical physics of deep learning: Optimal learning of a multi-layer perceptron. arXiv:2206.04642. **[30]**

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press**[31]**

[https://developer.android.com/][32]