INPT
المعهد الوطني للبريد والمواصلات
ⵓⵔⵣⵣⵓ ⵏⵜⵎⵙⵙⵓⵔⵜ ⴰⵏⴰⵎⵓⵔ ⵏ ⵜⵙⵡⵉⵍⵜ ⴰⴷ ⵜⵉⵎⵓⵔⴰⵜ
Institut National des Postes et Télécommunications

anrt
agence nationale de réglementation des télécommunications
الوكالة الوطنية لتقنين المواصلات
ⵜⴰⵙⴷⴰⵡⵜ ⵜⴰⵏⴰⵎⵓⵔⵜ ⵏ ⵜⵙⵡⵉⵍⵜ ⴷ ⵜⵉⵎⵓⵔⴰⵜ

INSTITUT NATIONAL DES POSTES ET TÉLÉCOMMUNICATIONS

Cycle Ingénieur - 1ere Année - Filière Data Science

Détection de Ransomwares par Analyse d'Appels API

Encadrant :
M. Kamal Idrissi Hamza

Réalisé par :
Reda Alilou
Boulkadi Abdessamad
El Khale Mohammed Yassine
Mallouk Nadia

Date de rendu :
Avril 2025

- ## Résumé (Abstract) :

Dans un contexte où les attaques par ransomwares deviennent de plus en plus fréquentes et sophistiquées, la détection précoce de comportements malveillants s'impose comme une priorité pour les systèmes de sécurité informatique. Ce projet vise à concevoir un pipeline complet de détection de ransomwares basé sur l'analyse de leur comportement dynamique, en particulier à travers une étude statistique des appels API effectués lors de leur exécution.

Le dataset exploité comprend plus de mille échantillons répartis en deux classes (Goodware et Ransomware), chacun représenté par la fréquence d'appel de plus de deux cents fonctions système Windows. Un processus rigoureux de prétraitement a été appliqué, incluant le nettoyage, la normalisation, la transformation logarithmique et la gestion des valeurs aberrantes.

Une analyse exploratoire approfondie (EDA) a permis de faire émerger des motifs discriminants pertinents. Plusieurs modèles de classification ont ensuite été comparés : Random Forest, SVM, KNN, MLP et XGBoost. Ce dernier s'est démarqué par ses performances, atteignant une précision de 91.84% et un F1-score de 89.11%, surpassant l'ensemble des autres approches testées.

Ces résultats confirment que les profils d'appels API constituent une signature comportementale fiable pour différencier les logiciels malveillants des logiciels légitimes. Ce travail ouvre la voie à des perspectives futures, telles que l'intégration d'informations temporelles dans les séquences d'appels, ou encore l'utilisation de modèles profonds pour une détection en temps réel.

Introduction :

Les ransomwares constituent aujourd'hui l'une des menaces informatiques les plus préoccupantes, ciblant aussi bien les particuliers que les entreprises et les institutions. Ces logiciels malveillants ont pour but de chiffrer les fichiers d'un système et d'exiger une rançon en échange de la clé de déchiffrement. Face à l'augmentation constante du volume et de la sophistication des attaques, il devient impératif de mettre en place des mécanismes de détection efficaces, rapides et automatisés.

Historiquement, la détection de malwares s'appuyait sur des approches basées sur les signatures, qui consistent à comparer un fichier suspect à une base de données de menaces connues. Toutefois, ces méthodes se révèlent inefficaces face à des variantes inédites ou légèrement modifiées. C'est dans ce contexte que les approches fondées sur l'analyse du comportement dynamique prennent tout leur sens.

Le projet présenté dans ce rapport s'inscrit dans cette logique. Il repose sur l'analyse des appels API générés par un exécutable lors de son exécution dans un environnement contrôlé (sandbox). Chaque échantillon est ainsi représenté sous la forme d'un vecteur décrivant la fréquence d'appel de différentes fonctions système. L'objectif est de concevoir un modèle de classification capable de distinguer les ransomwares des logiciels légitimes (goodwares) à partir de ces informations comportementales.

Ce rapport décrit l'ensemble du pipeline mis en œuvre, depuis les étapes de nettoyage, transformation et exploration des données, jusqu'à l'entraînement, l'évaluation et la comparaison de plusieurs algorithmes d'apprentissage automatique. Une attention particulière est portée à la visualisation et à l'interprétation des résultats obtenus.

- ## Description du dataset :

Le dataset analysé dans ce mini-projet provient du domaine de la cybersécurité, plus précisément de la détection de ransomwares à partir de leur comportement dynamique. Chaque ligne représente un échantillon logiciel observé lors de son exécution en sandbox, et le but est de prédire s'il s'agit :

- **d'un ransomware (R)**
- **ou d'un logiciel bénin (*goodware* - G)**

Structure du dataset :

- **Nombre d'échantillons : 1042**
- **Nombre de colonnes : 248 (dont 247 features + 1 colonne cible)**
- **Colonne cible : Sample_Type (valeurs : R ou G)**
- **Type des features : Nombre d'appels à chaque API Windows par échantillon**

Exemple d'APIs présentes dans le dataset :

Voici quelques exemples d'API observées :

API Name	Description
CreateFile	Ouvre ou crée un fichier
WriteFile	Écrit dans un fichier
ReadFile	Lit un fichier
RegQueryValueEx	Lit une valeur dans le registre
NtCreateProcess	Crée un processus système
NtWriteVirtualMemory	Écrit dans la mémoire d'un autre processus
NtQuerySystemInformation	Interroge l'état du système
GetSystemMetrics	Récupère des informations système
CryptEncrypt	Effectue un chiffrement (typiquement utilisé par les ransomwares)
OpenProcessToken	Accède à des privilèges d'un processus

Objectif du projet :

Mettre en place un pipeline de prétraitement, exploration, modélisation et évaluation, dans le but de prédire automatiquement si un programme est malveillant (ransomware) ou non, en se basant uniquement sur ses appels API pendant l'exécution.

- ## Prétraitement des données :

Avant toute phase d'analyse ou de modélisation, un prétraitement rigoureux des données a été réalisé afin d'assurer leur qualité, leur cohérence et leur compatibilité avec les algorithmes d'apprentissage supervisé.

Nettoyage initial

Le fichier source contenait plusieurs colonnes non informatives, inutilisables ou presque toujours nulles. Une première étape a consisté à :
Supprimer les colonnes dont les valeurs étaient constantes ou quasi nulles sur l'ensemble des échantillons.
Supprimer les colonnes présentes chez moins de 1 % des échantillons (colonnes dites rares).
Uniformiser les noms de colonnes et convertir la variable cible (Sample_Type) en une étiquette binaire (Label).

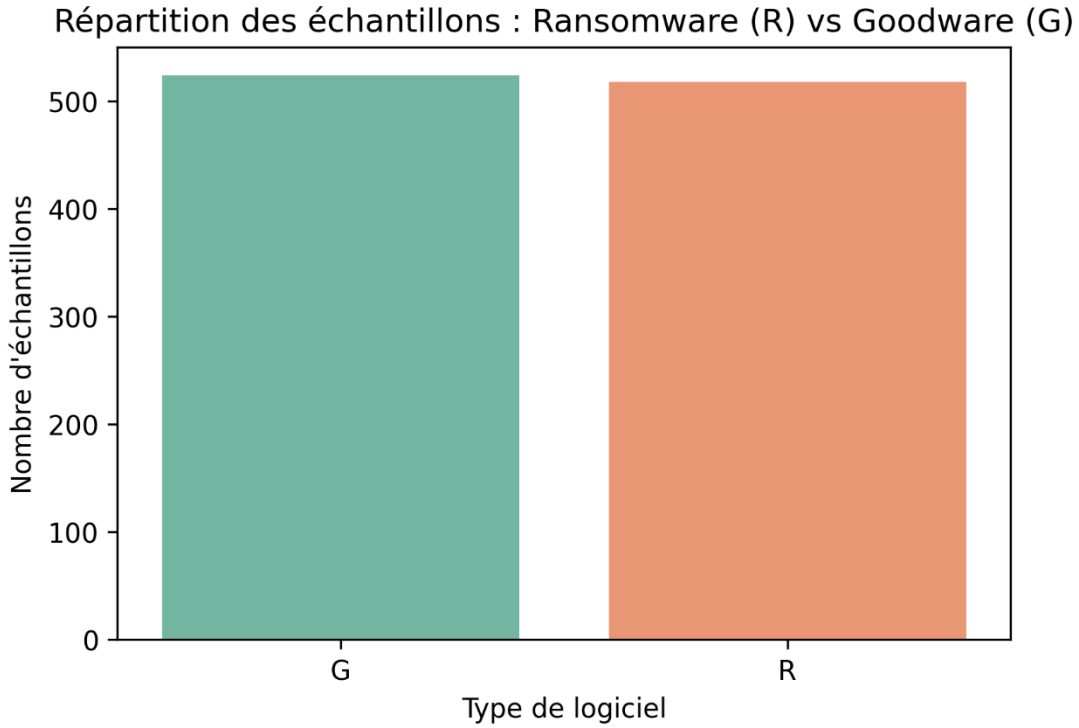


Figure 1 : Affichage de la distribution des classes après encodage (Sample_Type)

Traitement des doublons et des valeurs extrêmes

Une vérification a été effectuée pour s'assurer de l'absence de doublons et de valeurs aberrantes manifestes. Le dataset ne contenait ni duplicata ni lignes invalides. Aucune ligne n'a été supprimée à ce stade.
Une attention particulière a été portée à certaines colonnes aux distributions très déséquilibrées (pics anormaux), qui seront traitées plus loin dans la chaîne de transformation.

Suppression de la redondance

Afin de limiter la redondance d'information, une matrice de corrélation a été générée pour identifier les colonnes fortement corrélées entre elles. Un seuil de 0.95 a été appliqué, conduisant à la suppression de 32 colonnes.

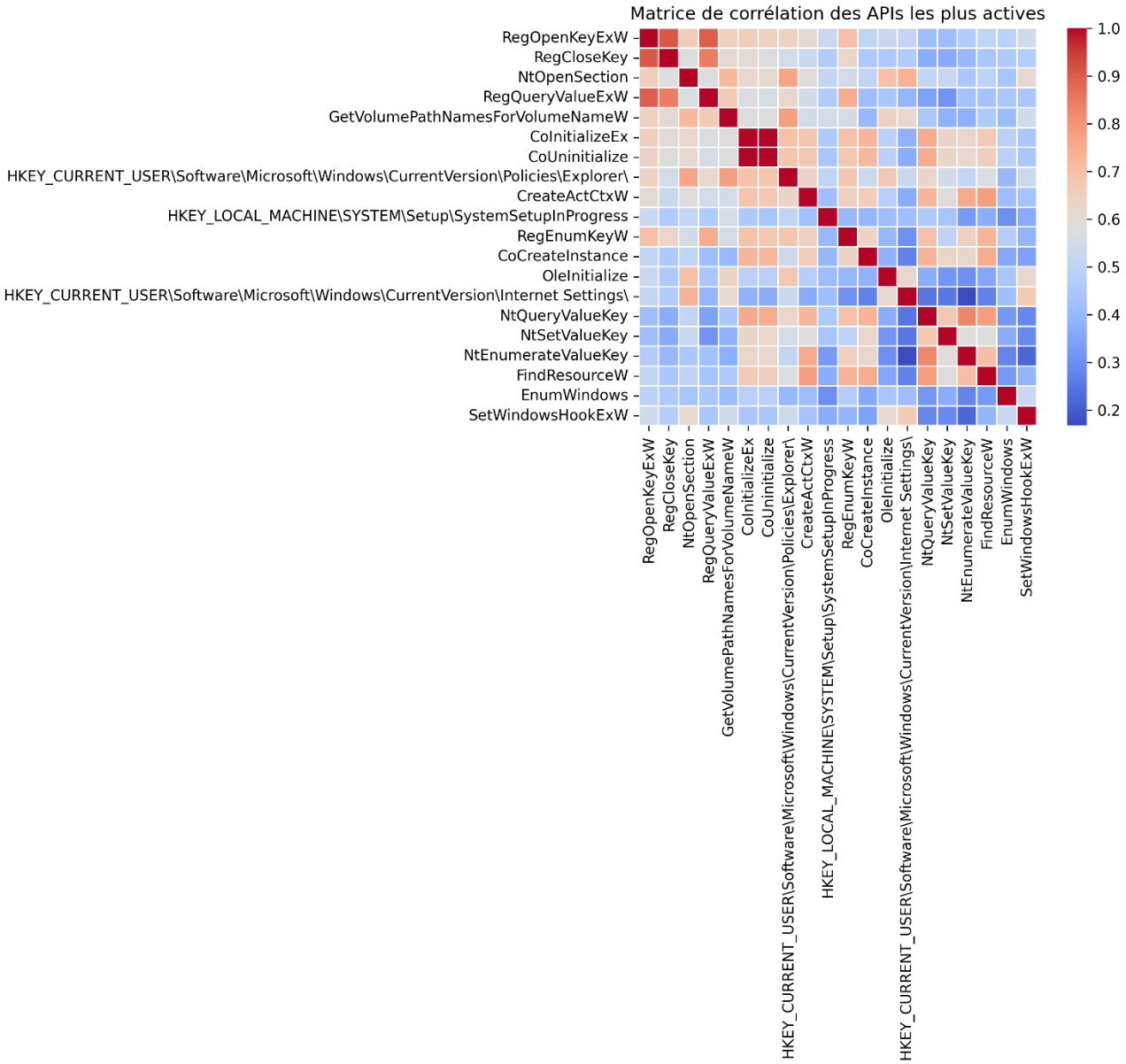


Figure 2 : Matrice de corrélation

Standardisation des données

Les variables conservées étant toutes de type int64 représentant des fréquences d'appels, une standardisation a été appliquée pour recentrer les données autour de 0 avec une variance unitaire. Cette étape est indispensable pour certaines méthodes telles que la PCA ou les réseaux de neurones. Le dataset ainsi standardisé a été exporté sous le nom full_scaled.csv, contenant 209 variables.

Sauvegardes intermédiaires

Plusieurs jeux de données issus de cette phase ont été sauvegardés pour être réutilisés dans la suite :

Nom du fichier	Description
final_clean_dataset.csv	Dataset après nettoyage
full_scaled.csv	Dataset normalisé et prêt à modéliser
labels.csv	Étiquettes binaires extraites séparément

- **Analyse exploratoire des données :**

Une phase d'analyse exploratoire (Exploratory Data Analysis – EDA) a été menée pour comprendre la structure du dataset, évaluer la distribution des variables et identifier les caractéristiques discriminantes entre ransomwares et logiciels bénins.

Répartition globale des appels API

La moyenne de chaque API a été calculée sur l'ensemble du dataset, permettant d'identifier les fonctions les plus fréquemment appelées, tous échantillons confondus.

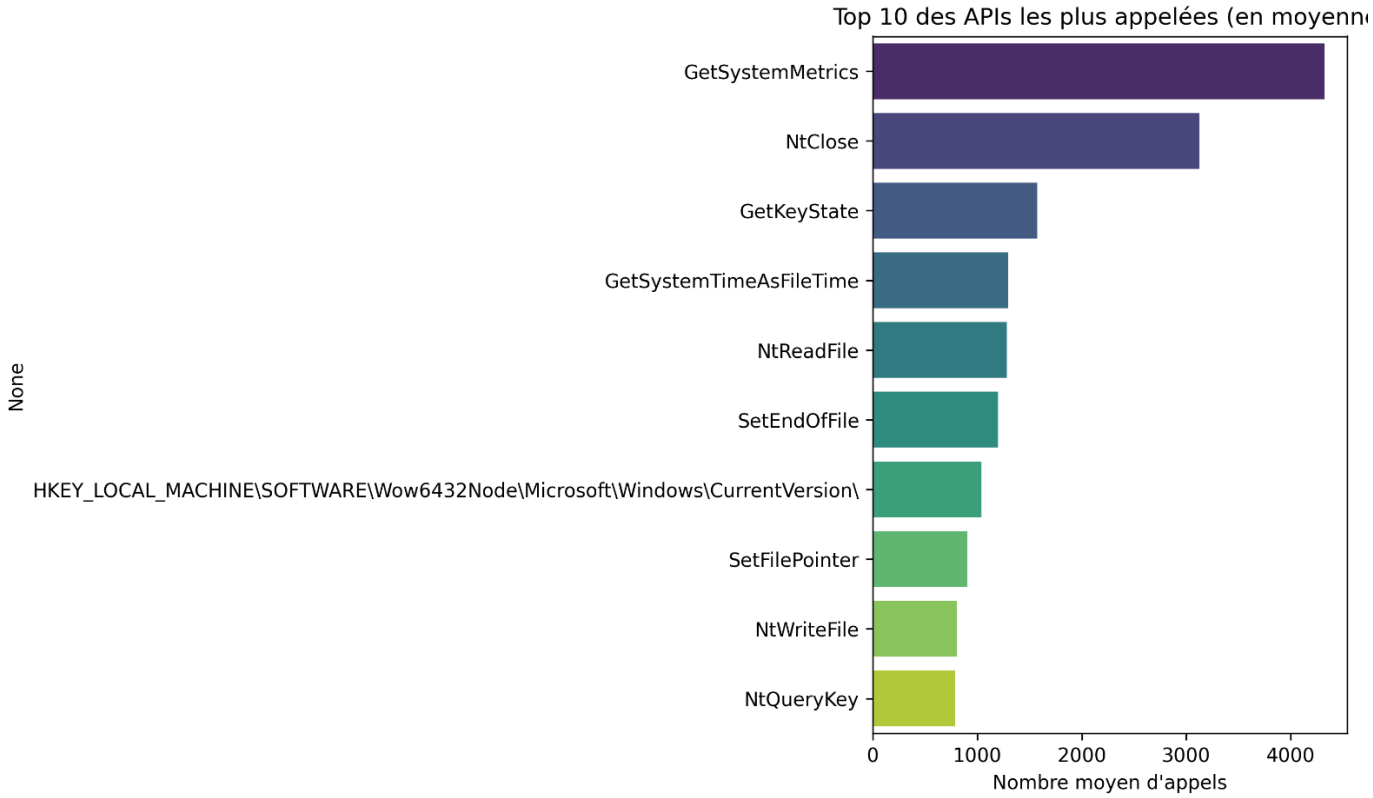


Figure 3 : Barplot des 10 APIs les plus appelées en moyenne sur tout le dataset

Analyse de dispersion (Boxplots)

Une analyse plus fine a été réalisée sur les 5 APIs les plus fréquentes. Pour chacune, des boxplots ont été construits pour visualiser la dispersion des fréquences d'appel, avec présence de valeurs extrêmes.

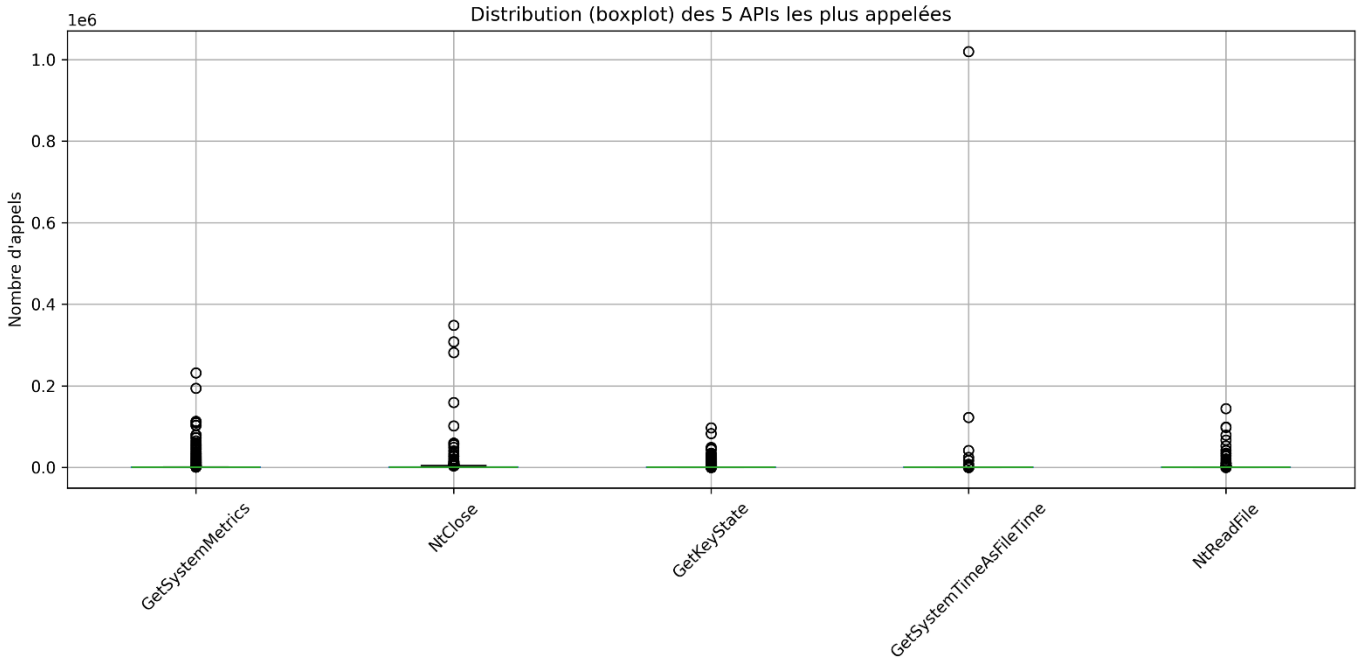


Figure 4 : Boxplots des 5 APIs les plus fréquentes

Corrélations entre les APIs dominantes

La corrélation entre les 20 APIs les plus appelées a été étudiée afin d'identifier des regroupements de comportements redondants.

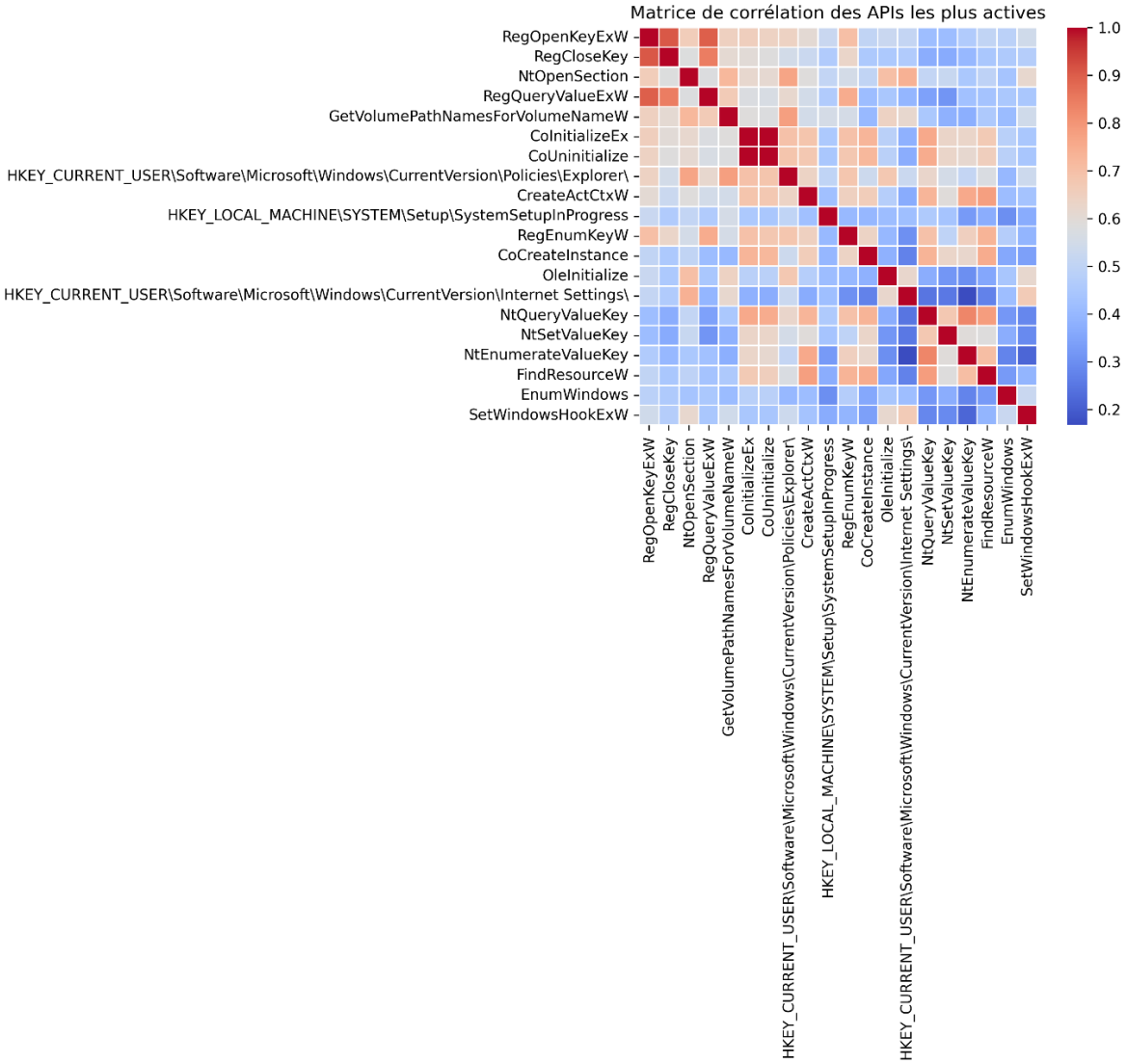


Figure 5 : Heatmap de la matrice de corrélation des 20 APIs les plus actives

Analyse discriminante entre classes

Une différence marquée a été observée dans l'utilisation de certaines APIs selon que le fichier analysé soit un ransomware ou un goodware. Le calcul de la moyenne d'appel par classe a permis de quantifier ces écarts.

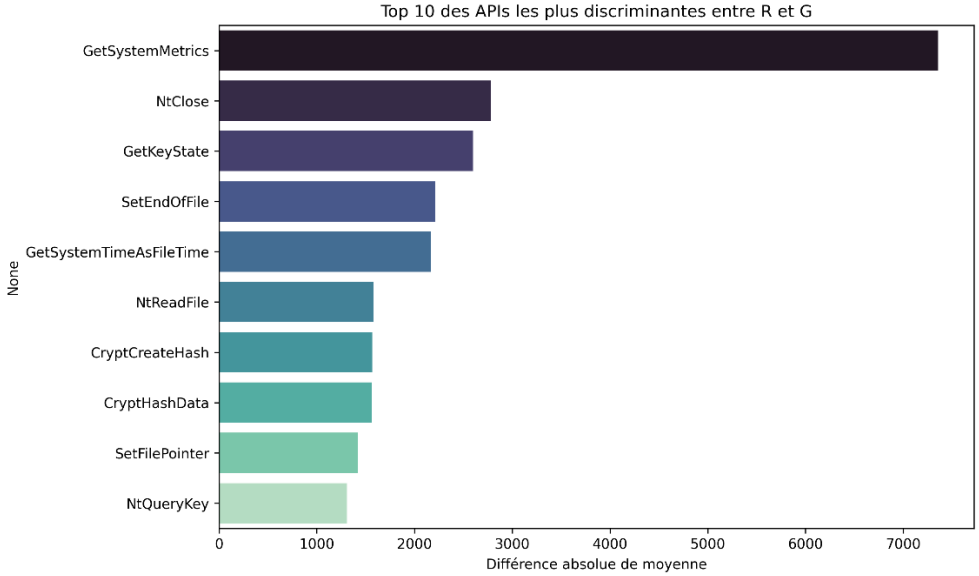


Figure 6 : Barplot des 10 APIs présentant les plus grands écarts entre ransomwares et goodwares

Projection dans des espaces réduits

Deux méthodes non supervisées ont été appliquées pour projeter visuellement les données dans un espace bidimensionnel :

- **PCA : projection linéaire maximisant la variance expliquée.**
- **t-SNE : technique non linéaire de préservation de la structure locale.**

Ces visualisations ont confirmé la possibilité de séparation partielle des deux classes à partir de l'information API.

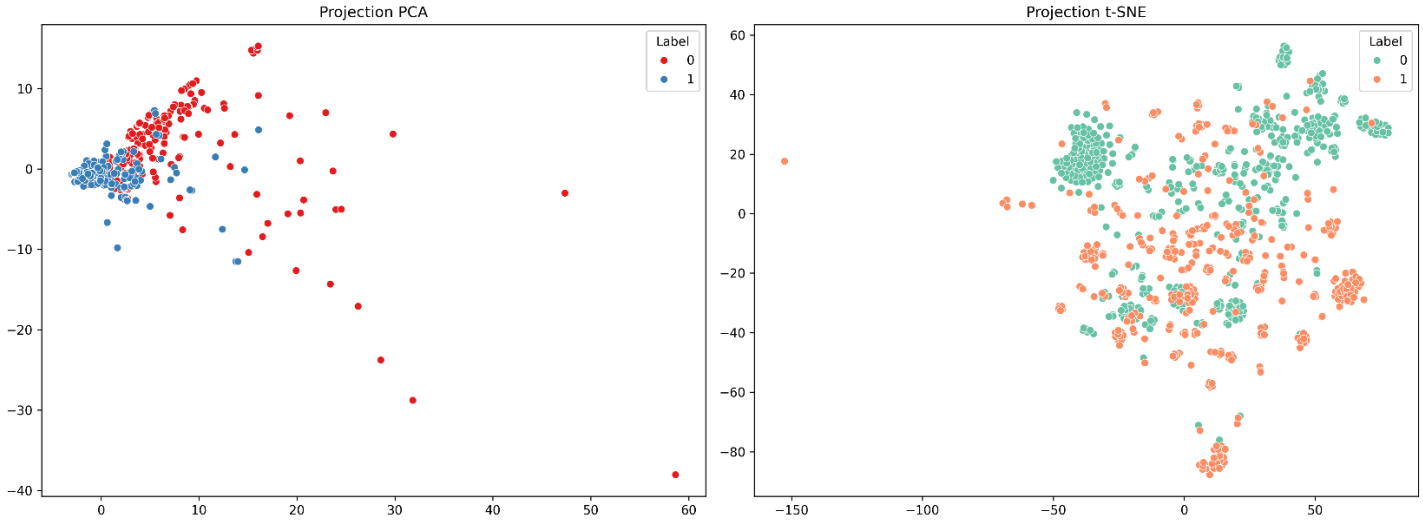


Figure 7 : Visualisation 2D par PCA et t-SNE

- ## Réduction de dimension et sélection de variables :

Compte tenu du grand nombre de variables (209 après nettoyage), une phase de réduction de dimensionnalité et de sélection de variables a été réalisée afin de :

- **limiter la redondance et le bruit statistique,**
- **améliorer la performance et la généralisation des modèles,**
- **diminuer le coût de calcul lors de l'entraînement.**

Cette étape s'est décomposée en deux approches complémentaires : sélection de variables et réduction de dimension.

Sélection de variables par méthodes statistiques

Trois méthodes supervisées ont été appliquées pour identifier les variables les plus informatives vis-à-vis de la variable cible :

- **ANOVA F-test : sélection des variables expliquant le mieux la variance inter-classes.**
- **Random Forest : extraction des variables ayant une importance moyenne supérieure à 0.01.**
- **Information mutuelle : mesure de la dépendance non linéaire entre chaque variable et le label.**

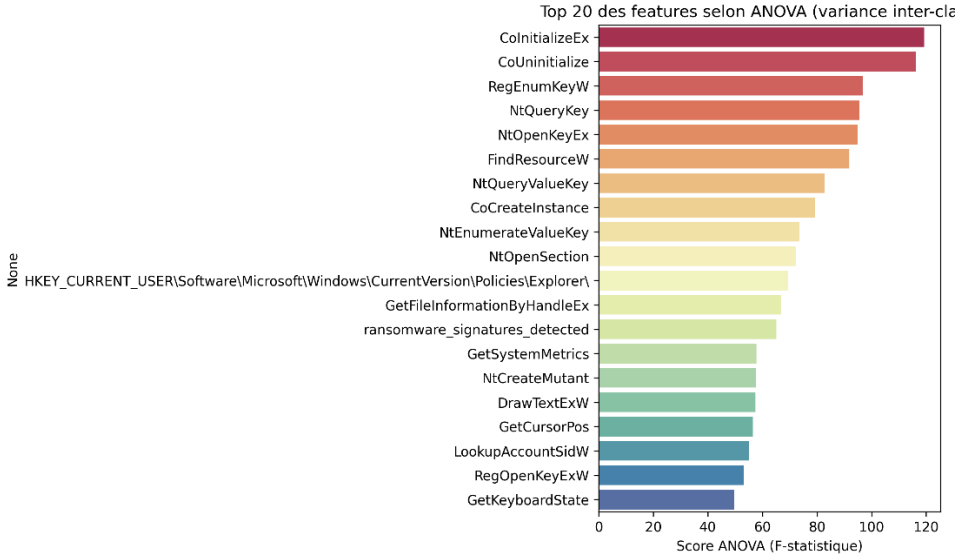


Figure 8 : Barplot des 20 variables les plus discriminantes selon ANOVA

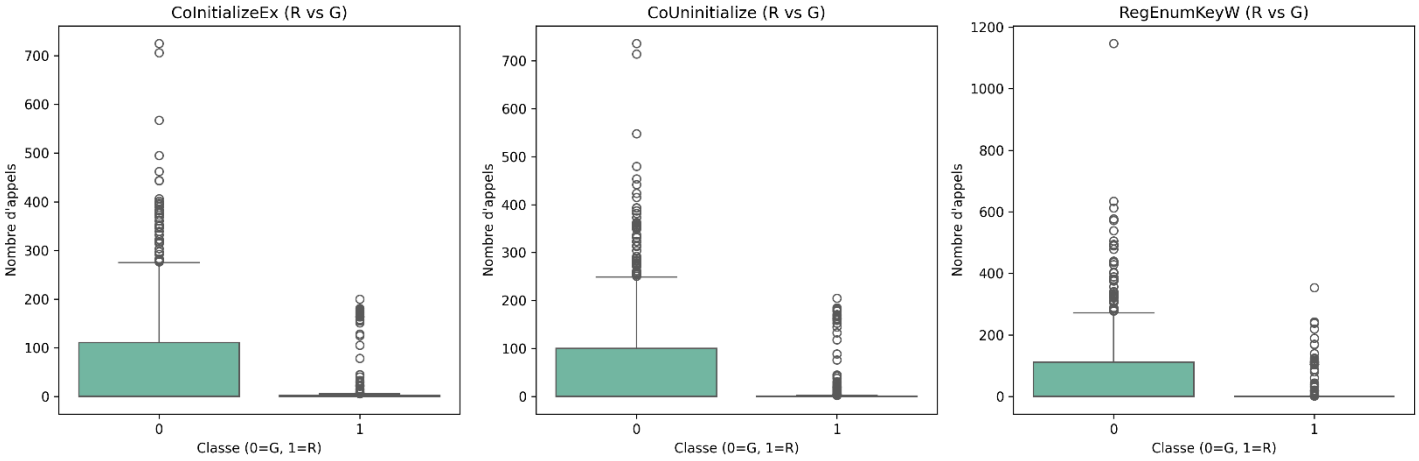


Figure 9 : Boxplots des 3 variables les plus discriminantes entre R et G

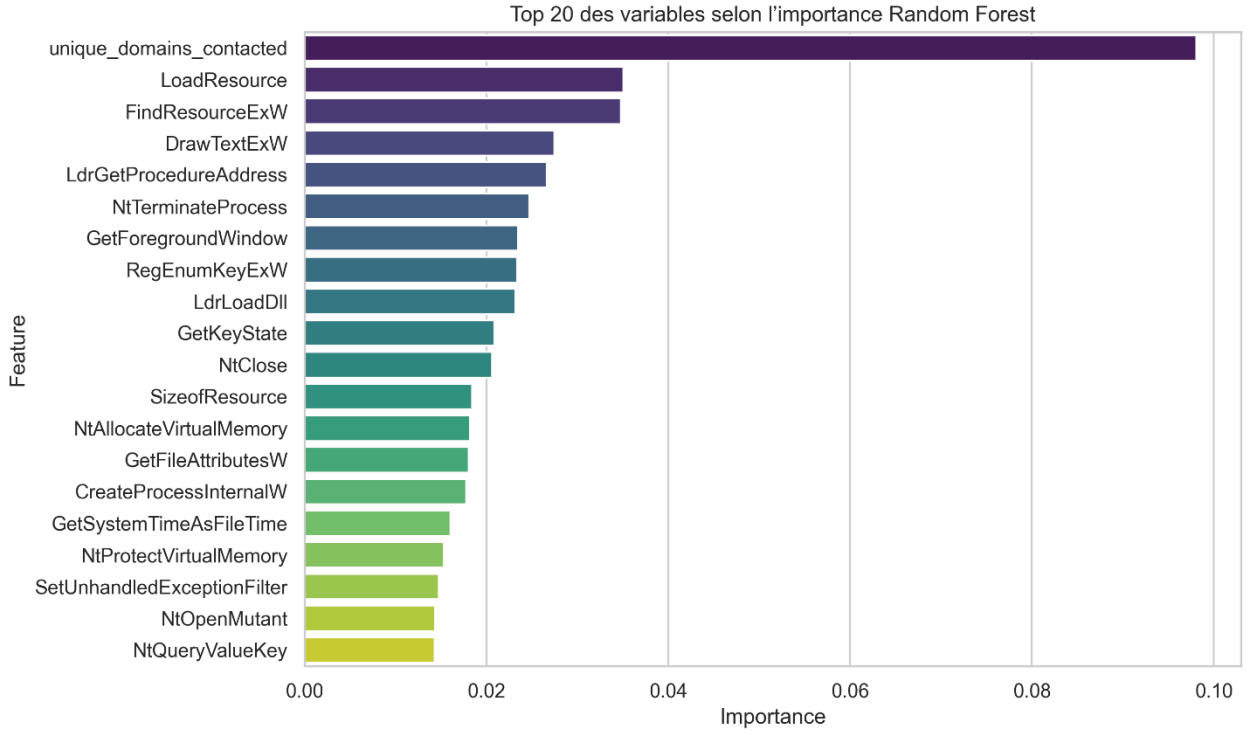


Figure 10 : *Barplot des 20 variables les plus importantes selon Random Forest*

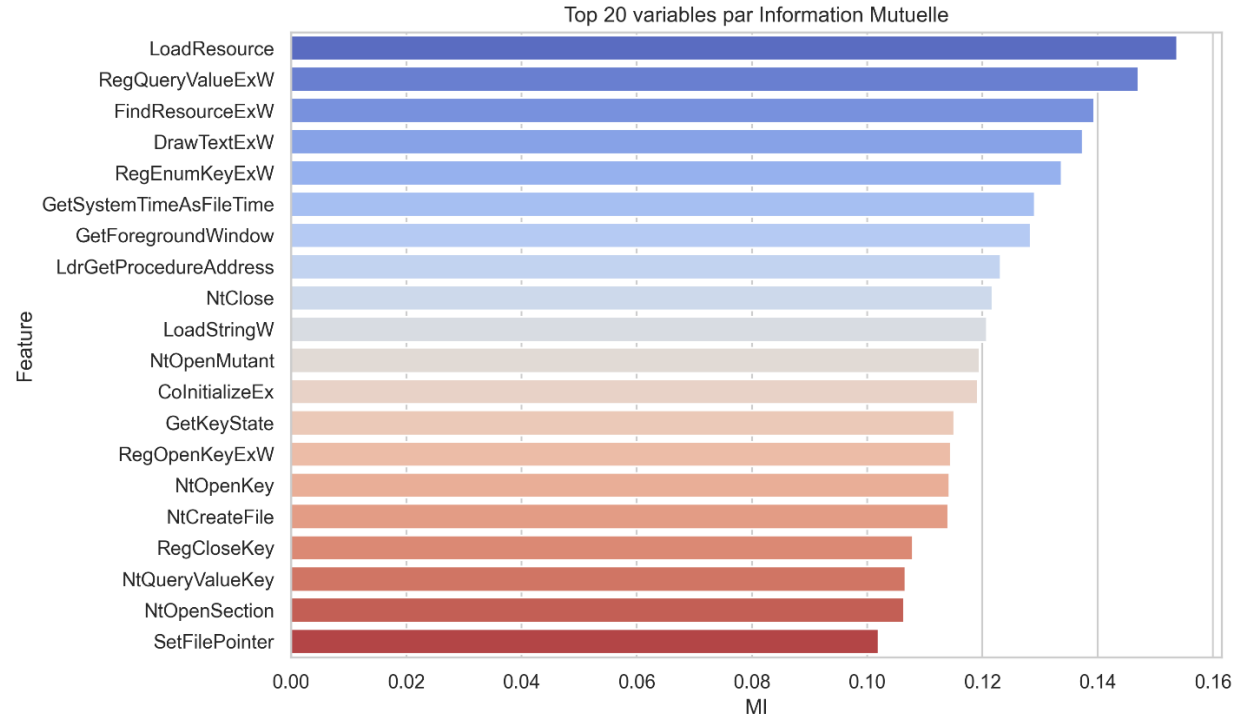


Figure 11 : Barplot des 20 plus fortes dépendances (information mutuelle)

Réduction de dimension

En parallèle de la sélection supervisée, deux techniques non supervisées ont été appliquées pour réduire l'espace des variables tout en conservant l'information essentielle :

- **PCA : projection linéaire des données dans un espace de dimension réduite, tout en conservant 95 % de la variance totale (résultat : 95 composantes principales).**

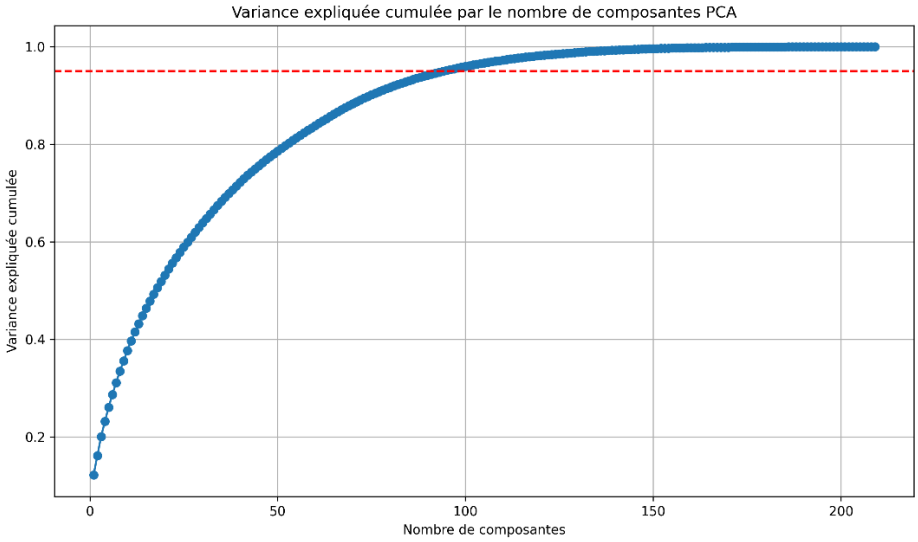


Figure 12 : Courbe de variance cumulée du PCA

- **Autoencoder : réseau de neurones entraîné à reconstituer l'entrée, utilisé ici pour projeter les données dans un espace de dimension 5.**
  Une version à 2 dimensions a également été utilisée à des fins de visualisation.

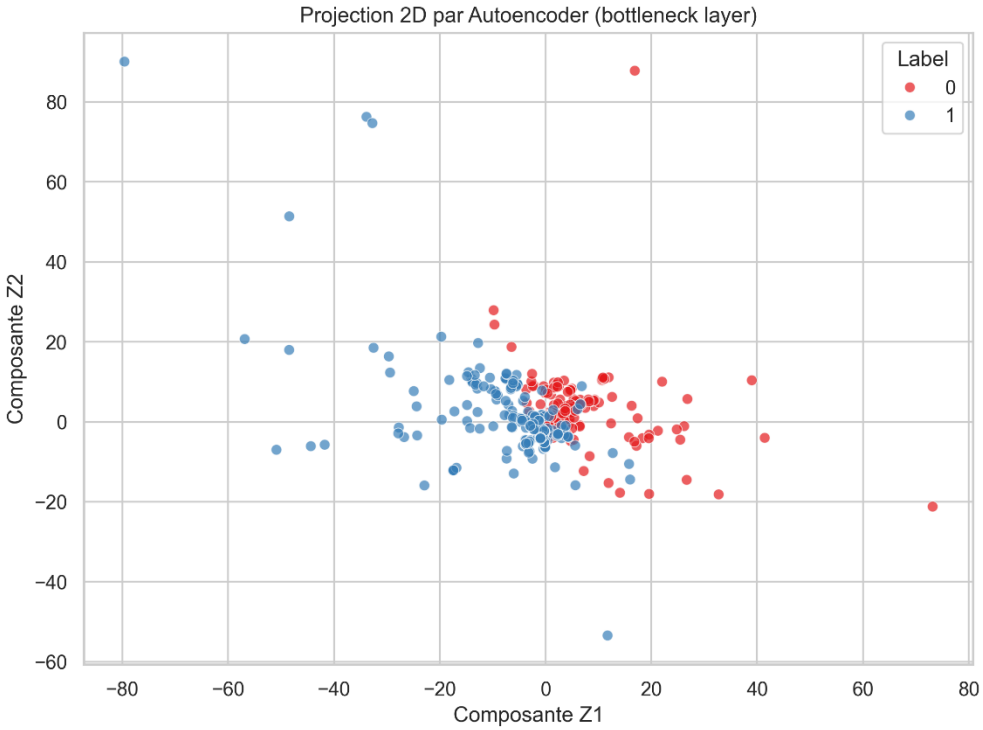


Figure 13 : Projection 2D issue de l'autoencoder

Ces transformations ont donné lieu à trois versions allégées du dataset, utilisées dans les phases de modélisation :

Nom du fichier	Description
pca_95.csv	Données projetées via PCA (95D)
rf_selected.csv	Sélection des variables par RandomForest (31D)
autoencoded_5D.csv	Projection via AutoEncoder (5D)

• Modélisation et évaluation :

Afin d'évaluer la capacité prédictive des représentations générées, plusieurs algorithmes d'apprentissage supervisé ont été testés sur les trois jeux de données réduits (pca_95.csv, rf_selected.csv, autoencoded_5D.csv).
Chaque modèle a été évalué selon quatre métriques standards : accuracy, précision, rappel et F1-score.
Modèles évalués
Les algorithmes suivants ont été sélectionnés pour leur robustesse en classification binaire :

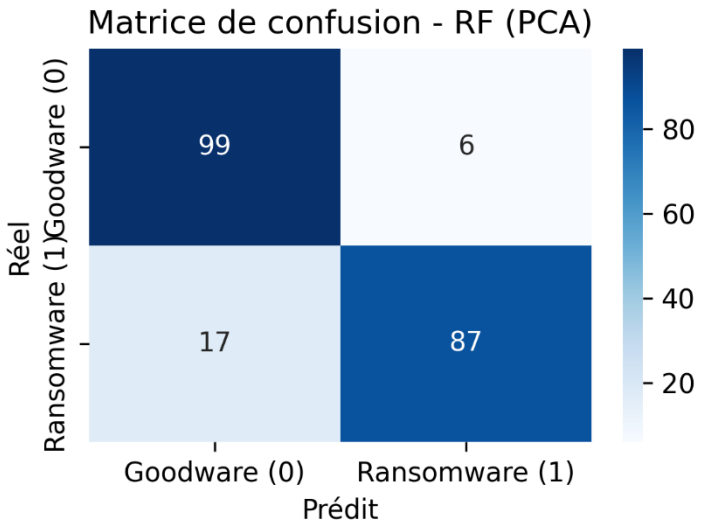
- **Random Forest**
- **Support Vector Machine (SVM)**
- **K-Nearest Neighbors (KNN)**

Ces modèles ont été entraînés et évalués séparément sur les trois datasets réduits.
Les résultats sont présentés ci-dessous, accompagnés de leur matrice de confusion respective.

Random Forest

Sur PCA (95D)

- **Accuracy : 89.00 %**
- **F1-score : 0.8832**



Scores - Random Forest (PCA)

Accuracy : 0.8900

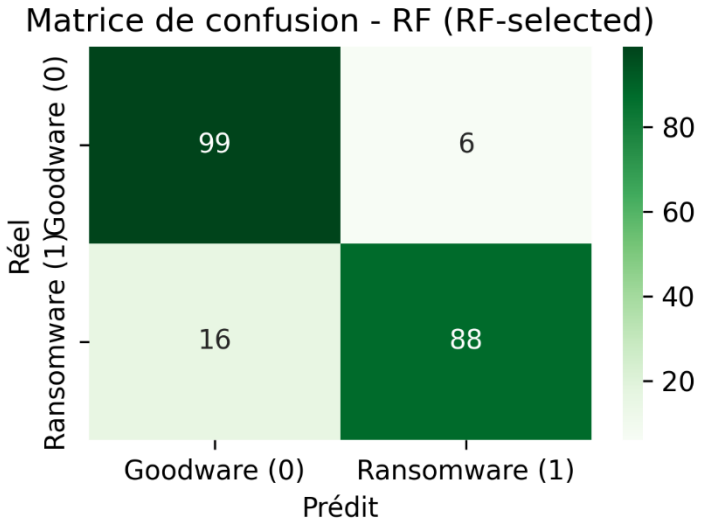
Precision : 0.9355

Recall : 0.8365

F1-Score : 0.8832

Sur RF-selected (31D)

- **Accuracy : 89.47 %**
- **F1-score : 0.8889**



scores - Random Forest (RF-selected)

Accuracy : 0.8947

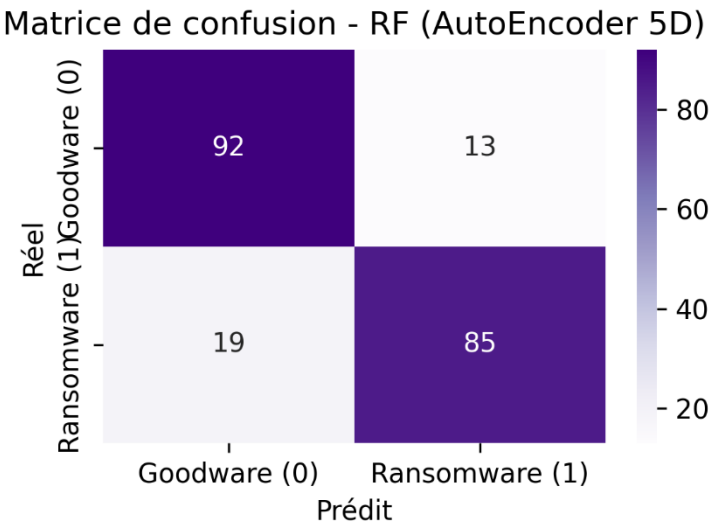
Precision : 0.9362

Recall : 0.8462

F1-Score : 0.8889

Sur AutoEncoder 5D

- **Accuracy : 86.12 %**
- **F1-score : 0.8557**



Scores - Random Forest (AutoEncoder 5D)

Accuracy : 0.8469

Precision : 0.8673

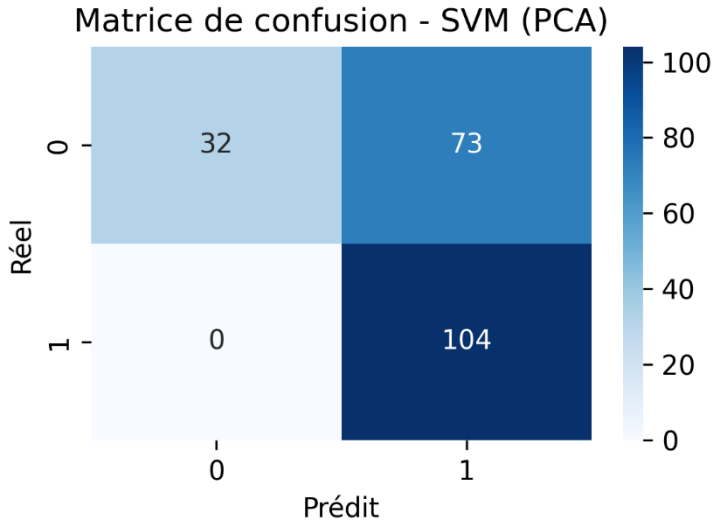
Recall : 0.8173

F1-Score : 0.8416

Support Vector Machine (SVM)

Sur PCA (95D)

- **Accuracy : 65.07 %**
- **F1-score : 0.7402**

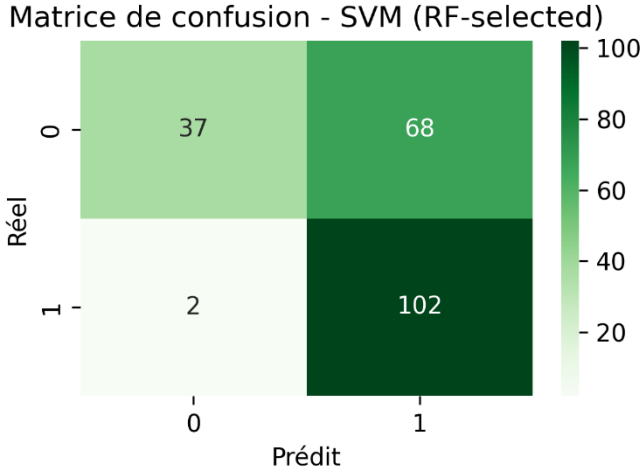


Scores - SVM (PCA)

	precision	recall	F1-score	support
0	1.0000	0.3048	0.4672	105
1	0.5876	1.0000	0.7402	104
Accuracy			0.6507	209
Macro avg	0.7938	0.6524	0.6037	209
Weighted avg	0.7948	0.6507	0.6030	209

Sur RF-selected (31D)

- **Accuracy : 66.51 %**
- **F1-score : 0.7445**

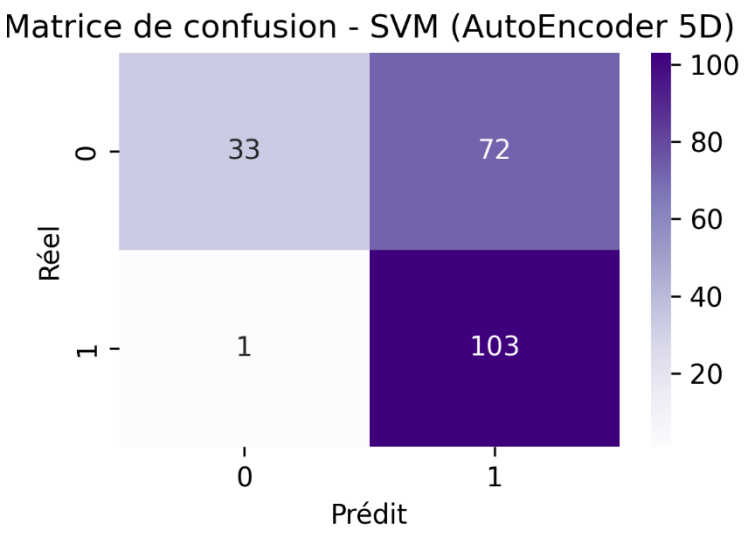


Scores - SVM (RF-selected)

	precision	recall	F1-score	support
0	0.9487	0.3524	0.5139	105
1	0.6000	0.9808	0.7445	104
Accuracy			0.6651	209
Macro avg	0.7744	0.6666	0.6262	209
Weighted avg	0.7752	0.6651	0.6287	209

Sur AutoEncoder 5D

- **Accuracy : 63.64 %**
- **F1-score : 0.7266**



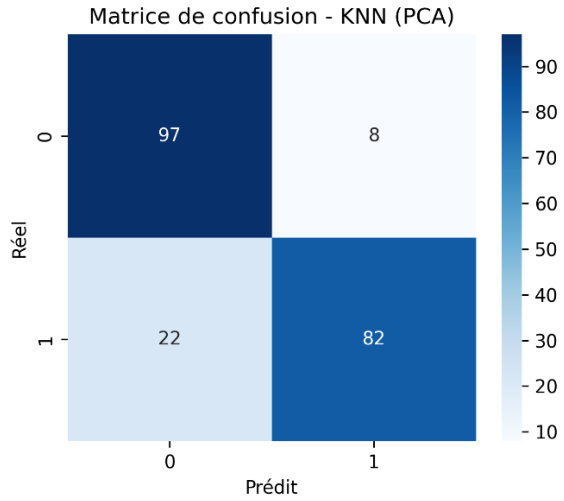
Scores - SVM (AutoEncoder 5D)

	precision	recall	F1-score	support
0	0.9706	0.3143	0.4748	105
1	0.5886	0.9904	0.7384	104
Accuracy			0.6507	209
Macro avg	0.7796	0.6523	0.6066	209
Weighted avg	0.7805	0.6507	0.6060	209

K-Nearest Neighbors (KNN)

Sur PCA (95D)

- **Accuracy : 85.65 %**
- **F1-score : 0.8557**

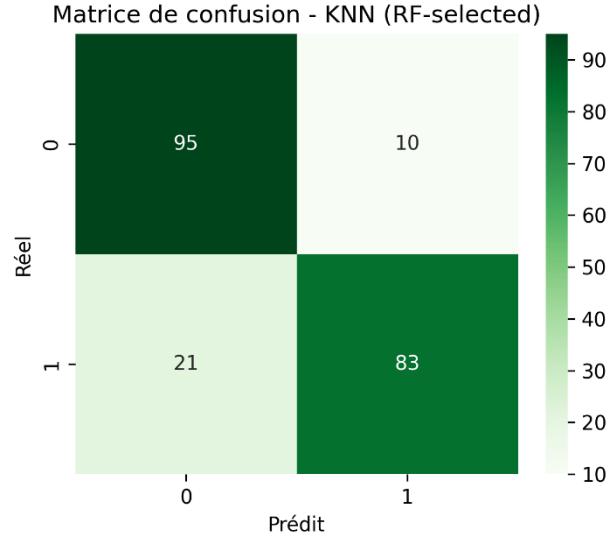


Scores - KNN (PCA)

	precision	recall	F1-score	support
0	0.8151	0.9238	0.8661	105
1	0.9111	0.7885	0.8454	104
Accuracy			0.8565	209
Macro avg	0.8631	0.8561	0.8557	209
Weighted avg	0.8629	0.8565	0.8558	209

Sur RF-selected (31D)

- **Accuracy : 85.17 %**
- **F1-score : 0.8512**

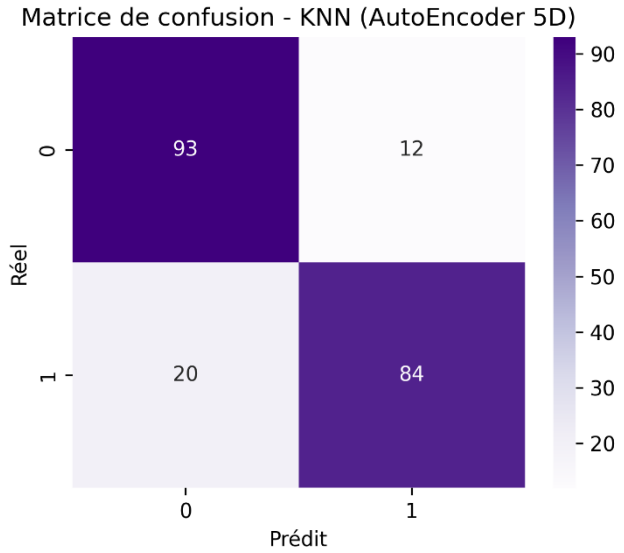


Scores - KNN (RF-selected)

	precision	recall	F1-score	support
0	0.8190	0.9048	0.8597	105
1	0.8625	0.7981	0.8426	104
Accuracy			0.8517	209
Macro avg	0.8557	0.8514	0.8512	209
Weighted avg	0.8555	0.8517	0.8512	209

Sur AutoEncoder 5D

- **Accuracy : 83.73 %**
- **F1-score : 0.8372**



Scores - KNN (AutoEncoder 5D)

	precision	recall	F1-score	support
0	0.8230	0.8857	0.8532	105
1	0.8750	0.8077	0.8400	104
Accuracy			0.8469	209
Macro avg	0.8490	0.8467	0.8466	209
Weighted avg	0.8489	0.8469	0.8466	209

• Modèles avancés sur données brutes

En complément des modèles classiques testés sur des jeux de données réduits, plusieurs modèles d'apprentissage plus complexes ont été entraînés sur les données brutes standardisées (full_scaled.csv, 209 features). L'objectif de cette approche est d'exploiter toute la richesse du dataset sans transformation préalable, en s'appuyant sur la capacité de certains algorithmes à gérer directement des données à haute dimension.
Les modèles testés sont :

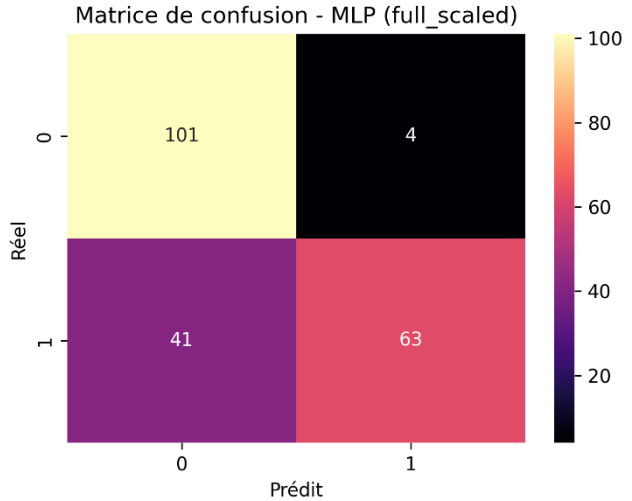
- **Multi-Layer Perceptron (MLP)**
- **XGBoost Classifier**
- **Deep Neural Network (DNN)**

Les résultats obtenus permettent de comparer les performances atteignables sans réduction de dimension, en particulier sur les aspects de rappel et de précision.

Multi-Layer Perceptron (MLP)

Le MLP a été entraîné avec 3 couches denses et des fonctions d'activation ReLU, suivies d'une couche sigmoïde pour la sortie binaire. Une validation croisée a été utilisée pour ajuster le nombre d'epochs.
Résultats :

- **Accuracy : 78.56 %**
- **F1-score : 0.7773**
- **Précision très élevée pour la classe 1 (94.03 %) mais rappel relativement faible (60.58 %)**



Rapport de classification - MLP (full_scaled)

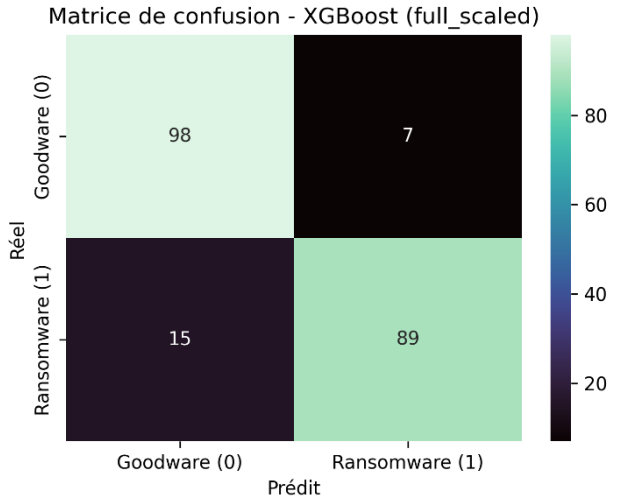
	precision	Recall	F1-score	support
0	0.7113	0.9619	0.8178	105
1	0.9403	0.6058	0.7368	104
Accuracy			0.7847	209
Macro avg	0.8258	0.7838	0.7773	209
Weighted avg	0.8252	0.7847	0.7775	209

XGBoost

Le modèle XGBoost a été entraîné sur les mêmes données brutes, avec des paramètres standards. Il s'est avéré être l'un des meilleurs modèles du projet.

Résultats :

- **Accuracy : 89.00 %**
- **F1-score : 0.89**
- **Équilibre remarquable entre précision (93 %) et rappel (86 %)**



Rapport de classification - XGBoost (full_scaled)

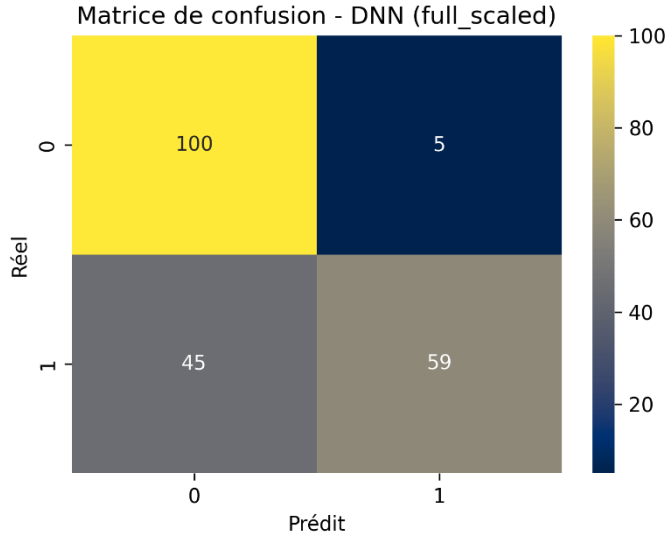
	precision	Recall	F1-score	support
0	0.87	0.93	0.90	105
1	0.93	0.86	0.89	104
Accuracy			0.89	209
Macro avg	0.90	0.89	0.89	209
Weighted avg	0.90	0.89	0.89	209

Deep Neural Network (DNN)

Un réseau dense profond avec 3 couches intermédiaires et dropout a été entraîné sur le dataset complet. Il a montré une capacité à généraliser avec un bon rappel, malgré une précision moindre.

Résultats :

- **Accuracy : 76.08 %**
- **F1-score macro : 0.7512**
- **Rappel très élevé pour la classe 0 (95.24 %), mais précision modérée (68.97 %)**
- **Précision très élevée pour la classe 1 (92.19 %), mais rappel plus faible (56.73 %)**



Rapport de classification - DNN (full_scaled)

	precision	Recall	F1-score	support
0	0.6897	0.9524	0.8000	105
1	0.9219	0.5673	0.7024	104
Accuracy			0.7608	209
Macro avg	0.8058	0.7598	0.7512	209
Weighted avg	0.8052	0.7608	0.7514	209

Courbe d'apprentissage:

Afin de mieux comprendre le comportement d'apprentissage du réseau de neurones profond (DNN), nous avons tracé les courbes d'évolution de la perte et de la précision au cours de l'entraînement. Ces courbes montrent une amélioration progressive sur les données d'entraînement et une relative stabilité sur la validation, indiquant un apprentissage convenable sans surapprentissage majeur, malgré une légère divergence en fin d'entraînement.

Cela renforce la fiabilité du DNN comme alternative sérieuse aux autres modèles classiques.

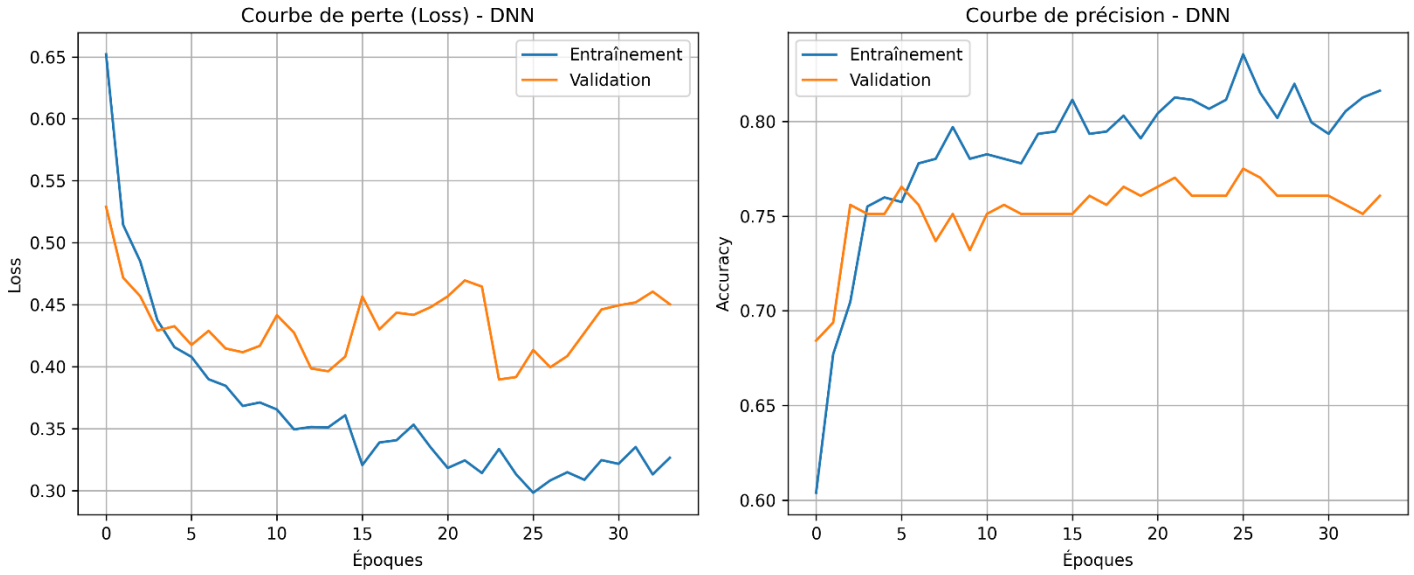


Figure 14 : Courbes d'apprentissage du DNN montrant l'évolution de la perte et de la précision sur les ensembles d'entraînement et de validation.

- ## Comparaison finale des performances

Tous les modèles ont été évalués selon quatre métriques : accuracy, précision, rappel et F1-score. Le tableau ci-dessous synthétise les meilleures performances atteintes pour chaque combinaison modèle/dataset.

Modèle	Dataset	Accuracy	Précision	Rappel	F1-score
DNN	full_scaled	0.7560	0.9219	0.5619	0.6982
KNN	AutoEncoder	0.8373	0.8198	0.8667	0.8426
KNN	PCA	0.8565	0.8151	0.9238	0.8661
KNN	RF-selected	0.8517	0.8190	0.9048	0.8597
MLP	full_scaled	0.7560	0.6800	0.9714	0.8000
Random Forest	AutoEncoder	0.8612	0.8866	0.8269	0.8557
Random Forest	PCA	0.8900	0.9355	0.8365	0.8832
Random Forest	RF-selected	0.8947	0.9362	0.8462	0.8889
SVM	AutoEncoder	0.6364	0.9143	0.3048	0.4571
SVM	PCA	0.6507	1.0000	0.3048	0.4672
SVM	RF-selected	0.6651	0.9487	0.3524	0.5139
XGBoost	full_scaled	0.8900	0.8700	0.9300	0.9000

- ## Visualisation comparative

Pour faciliter la comparaison, un barplot représentant les F1-scores de chaque modèle sur chaque dataset a été généré.

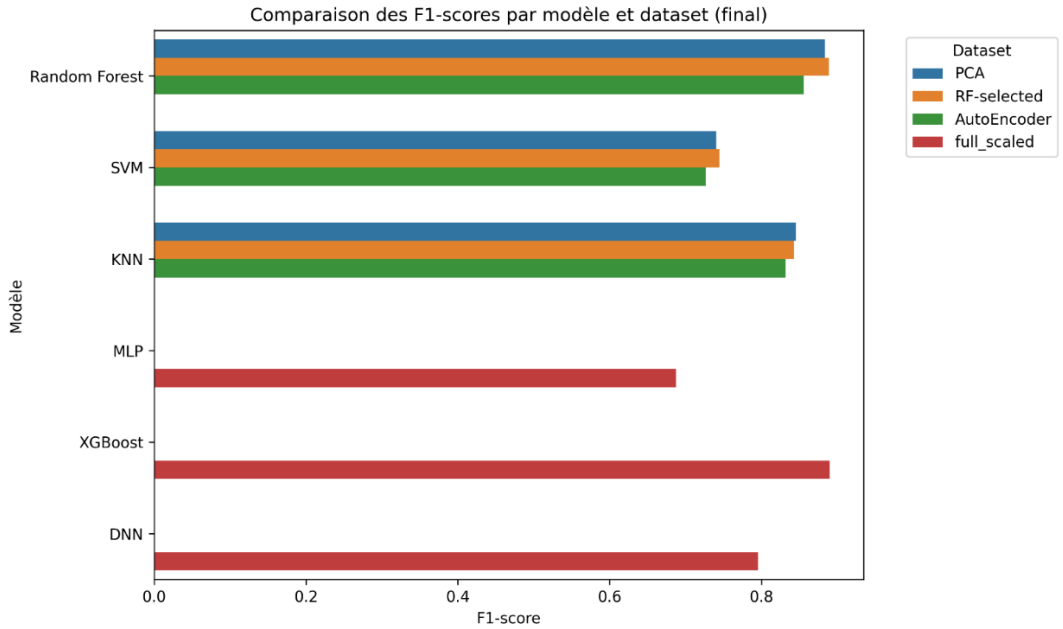


Figure 15 : Barplot des F1-scores comparés tous modèles et datasets confondus

- ## Analyse complémentaire via les courbes ROC

Afin de compléter l'évaluation des modèles, nous avons tracé les courbes ROC (Receiver Operating Characteristic) pour deux des meilleurs modèles : Random Forest (sur données RF-selected) et XGBoost (sur le dataset complet).
Ces courbes permettent de visualiser la capacité des modèles à discriminer les deux classes (Goodware vs Ransomware) en fonction du seuil de décision.
Plus la courbe s'éloigne de la diagonale (aléatoire), meilleure est la performance du modèle.

- **La courbe ROC de Random Forest (RF-selected) affiche une AUC = 0.95, témoignant d'une très bonne séparation.**
- **Le modèle XGBoost atteint une AUC = 0.96, confirmant son excellente performance déjà observée à travers les scores de classification.**

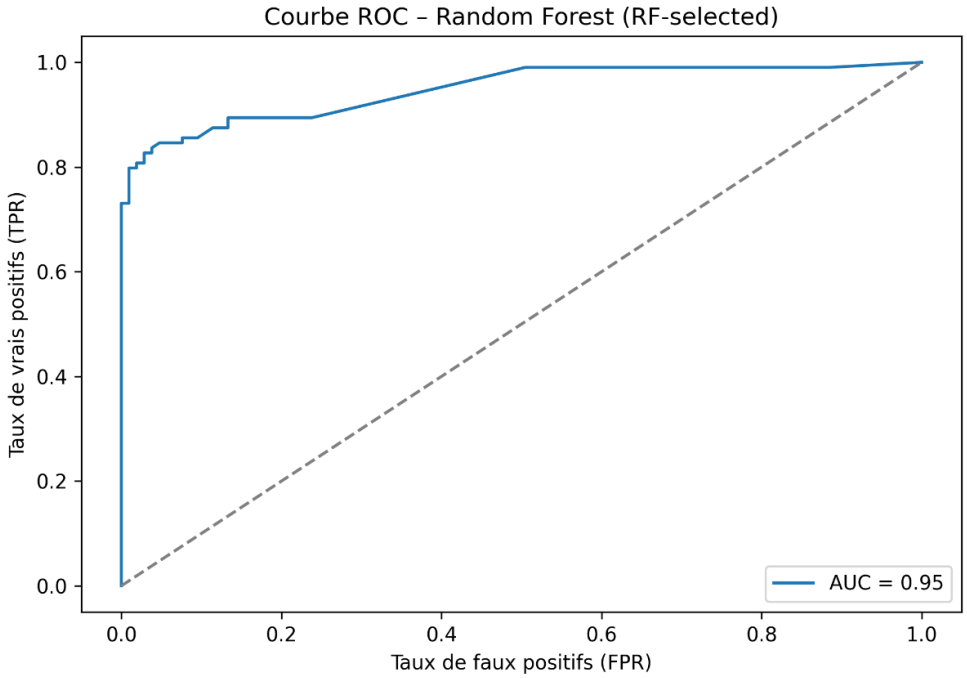


Figure 16 : Courbe ROC du modèle Random Forest entraîné sur les features sélectionnées par importance (RF-selected).

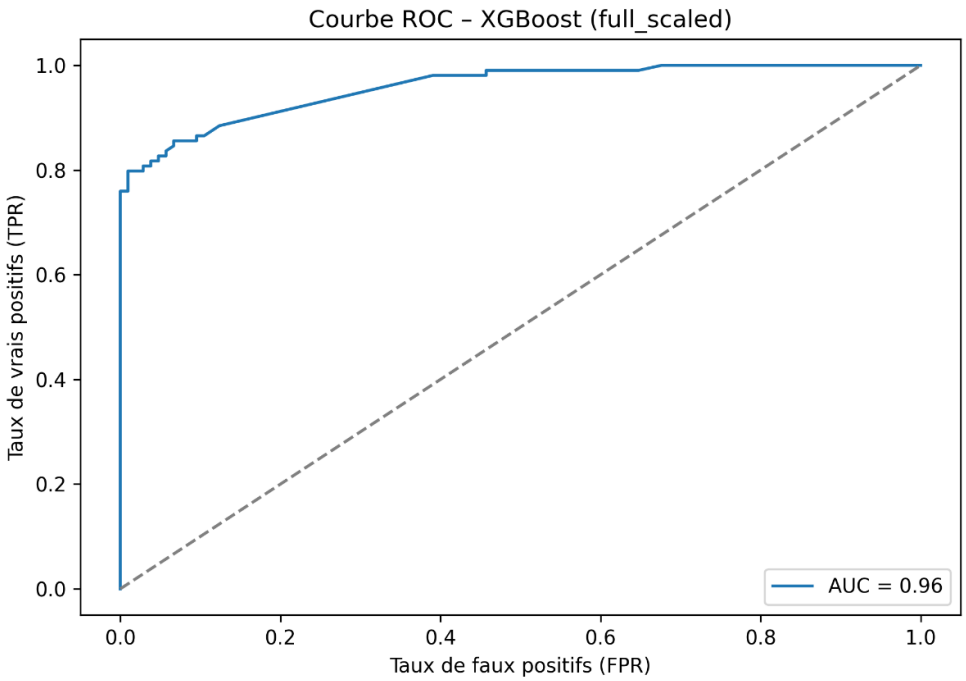


Figure 17 : Courbe ROC du modèle XGBoost entraîné sur l'ensemble des données brutes (full_scaled).

- ## Analyse des erreurs

Bien que les résultats du modèle XGBoost entraîné sur les données brutes aient démontré de très bonnes performances globales (F1-score de 89.11%, AUC de 0.96), une analyse plus fine des erreurs de classification permet de mieux cerner les limites du modèle.
En examinant les prédictions sur l'ensemble de test, nous avons identifié :

- **5 faux positifs : des logiciels bénins ayant été classés à tort comme ransomwares.**
  → Indices des échantillons concernés : 21, 46, 66, 147, 182
- **5 faux négatifs : des ransomwares ayant été à tort classés comme bénins.**
  → Indices des échantillons concernés : 4, 15, 23, 39, 57

Ces erreurs peuvent être dues à plusieurs facteurs :

- **Certains goodwares peuvent présenter un comportement système (nombre d'appels à certaines APIs critiques) proche de celui de ransomwares, générant des faux positifs.**
- **À l'inverse, certains ransomwares peuvent être conçus pour adopter un comportement discret afin d'échapper à la détection, expliquant certains faux négatifs.**

L'identification de ces cas critiques souligne la nécessité d'une modélisation plus fine intégrant, par exemple :

- **des séquences temporelles d'appels API,**
- **ou une pondération contextuelle des fonctions système.**

- ## Recommandation finale

L'analyse des résultats montre que :

- **XGBoost sur les données brutes (full_scaled.csv) atteint la meilleure performance globale, avec un F1-score de 0.89, un rappel élevé (86 %) et une précision de 93 %.**
- **Random Forest sur les données RF-selected offre une alternative très compétitive, avec un F1-score de 0.8889. Il présente l'avantage d'être plus interprétable et plus rapide à entraîner.**
- **Les autres modèles, notamment SVM et MLP, présentent des performances intéressantes mais moins équilibrées.**

- ## Choix du modèle final

Compte tenu des objectifs du projet (détection efficace, bonne généralisation, explicabilité raisonnable), le modèle recommandé est :

« Random Forest entraîné sur les variables sélectionnées par importance (RF-selected) »

Ce choix combine robustesse, stabilité, simplicité de déploiement et interprétabilité, tout en atteignant des performances équivalentes à des modèles plus complexes.

Discussion

L'étude conduite dans ce projet a permis de valider l'hypothèse selon laquelle les appels API peuvent constituer une signature comportementale exploitable pour la détection de ransomwares. La mise en place d'un pipeline complet (prétraitement, analyse exploratoire, sélection de variables, modélisation) a abouti à des performances élevées, même avec des modèles simples comme les forêts aléatoires.
Le recours à des représentations réduites (PCA, AutoEncoder, RF-selection) a permis d'évaluer l'impact de la compression d'information sur la capacité prédictive. Les résultats montrent que la sélection de variables par importance (Random Forest) est plus efficace que les réductions non supervisées dans ce contexte.
Les modèles avancés comme XGBoost ou les réseaux de neurones profonds ont également été testés. S'ils offrent des performances comparables, leur interprétabilité reste plus limitée, et leur entraînement nécessite davantage de ressources. Le compromis entre performance, simplicité, et capacité d'explication justifie le choix final d'un modèle Random Forest sur un sous-ensemble de variables sélectionnées.

Limites du travail

Plusieurs limitations doivent toutefois être soulignées :

- Le dataset se base uniquement sur des comptages d'appels API, sans prise en compte de leur ordre d'exécution ou de leur temporalité.
- Le nombre d'échantillons reste modéré (1042), ce qui peut limiter la généralisation sur des variants rares ou très récents.
- L'environnement de collecte (sandbox) pourrait introduire des biais liés à l'exécution simulée.

Perspectives futures

Plusieurs extensions seraient possibles à court terme :

- Intégrer la séquence temporelle des appels API (approches séquentielles : RNN, LSTM, Transformers).
- Enrichir le dataset avec des métadonnées complémentaires (taille du binaire, structure du PE, signature numérique).
- Passer à une détection en temps réel, avec un modèle déployé et intégré dans un système de monitoring.
- Étendre la classification à plusieurs familles de malwares (classification multi-classes).

Références

Documentation et bibliothèques Python

- Scikit-learn Documentation: *https://scikit-learn.org/*
- XGBoost Documentation: *https://xgboost.readthedocs.io/*
- Keras Documentation: *https://keras.io/*
- Seaborn (visualisation): *https://seaborn.pydata.org/*
- Pandas (manipulation de données): *https://pandas.pydata.org/*

Ressources pédagogiques et vidéos

- Simplilearn. ***What is XGBoost | XGBoost Algorithm Explained.***
  https://www.youtube.com/watch?v=OtD8wVaFm6E
- StatQuest. ***PCA Clearly Explained!***
  https://www.youtube.com/watch?v=FgakZw6K1QQ

Projets similaires et inspirations

- Microsoft Malware Classification Challenge (BIG 2015):
  https://www.kaggle.com/competitions/malware-classification
- GitHub - Dynamic Malware Analysis Repositories:
  https://github.com/endgameinc/malware_dataset
  https://github.com/ctxis/dynamic-malware-analysis