Détection de Logiciels Malveillants à l'Aide de Graphes de Flux de Contrôle

Kimmeng HONG, Tito KOH, Lamiaa EL OUATILI
Équipe: Hyperparameter
École Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise

Résumé

Ce rapport présente une approche d'apprentissage automatique pour la détection de logiciels malveillants en utilisant des graphes de flux de contrôle (CFG) extraits de fichiers exécutables. La méthodologie implique le traitement de fichiers JSON contenant des données de CFG, l'extraction de caractéristiques sur les séquences d'instructions, la vectorisation, l'application d'une réduction de dimensionnalité avant l'entraînement d'un classifieur XGBoost, et la gestion du déséquilibre des classes. Le modèle subit un réglage d'hyperparamètres par validation croisée randomisée, atteignant une classification efficace des logiciels malveillants.

Mots-clés : Classification multi-label, XGBoost, Tuning d'hyperparamètres, Décomposition en Valeurs Singulières Tronquée (SVD).

1 Introduction

La détection de logiciels malveillants reste un défi crucial en cybersécurité où les méthodes traditionnelles basées sur des signatures peinent souvent face aux nouvelles menaces évolutives. Cette limitation a motivé l'adoption d'approches d'apprentissage automatique capables d'apprendre des motifs à partir des structures de programmes. Notre projet explore la classification de logiciels malveillants utilisant des graphes de flux de contrôle, qui capturent efficacement les motifs structurels et les flux d'exécution des programmes exécutables, fournissant des caractéristiques riches pour l'analyse.

2 Description des données

Le jeu de données se compose de fichiers JSON contenant des CFGs d'échantillons exécutables, ainsi que d'un fichier CSV décrivant plus de 400 étiquettes comportementales par échantillon. Ce problème est formulé comme une classification multi-label. Environ 1% des fichiers JSON n'ayant pas d'entrée correspondante dans le CSV ont été écartés pour garantir la cohérence de l'apprentissage supervisé.

3 Méthodologie

3.1 Prétraitement des données

Le pipeline de prétraitement a commencé par l'alignement des fichiers JSON contenant les séquences d'instructions brutes avec leurs étiquettes correspondantes du fichier CSV (voir figure 1). Chaque séquence d'instructions a été traitée comme un texte complet, préservant tous les détails opérationnels incluant les opérandes, registres et adresses mémoire. Les séquences ont ensuite été vectorisées en utilisant HashingVectorizer configuré avec 2^{14} caractéristiques, transformant efficacement les séquences d'instructions de longueur variable en une représentation numérique cohérente sans surcharge de stockage de vocabulaire. L'étape finale de prétraitement a intégré ces caractéristiques vectorisées avec leurs étiquettes encodées en one-hot parmi les 453 familles de logiciels malveillants.

3.2 Réduction de dimensionnalité

Afin de réduire la dimensionnalité de notre espace de caractéristiques, nous avons appliqué la Décomposition en Valeurs Singulières Tronquée (SVD), bien adaptée aux matrices creuses de grande dimension. Cette méthode préserve les structures linéaires les plus significatives tout en réduisant les coûts computationnels. L'analyse du ratio de variance expliquée a révélé que les 150 premières composantes retenaient environ 99% de la variance totale. Le point d'inflexion observé dans le scree plot confirmait cette borne, permettant de réduire drastiquement le temps d'entraînement sans perte significative de performance pour la classification multi-label sur les 453 familles de logiciels malveillants.

3.3 Entraînement du modèle

3.3.1 Choix de XGBoost

XGBoost (eXtreme Gradient Boosting) est un algorithme d'ensemble reposant sur le principe du *gradient boosting*, dans lequel chaque arbre de décision est entraîné pour corriger les erreurs des précédents. Il optimise une fonction de coût régulière :

$$\mathcal{L}(\phi) = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t)}) + \sum_{k=1}^{t} \Omega(f_k),$$

où $\Omega(f_k)$ régularise la complexité des arbres via des pénalités L1/L2. Ce cadre mathématique robuste, combiné à une implémentation hautement optimisée, fait de XGBoost un modèle performant, rapide à entraîner, capable de gérer efficacement des matrices creuses et résistant au sur-apprentissage.

3.3.2 Stratégie d'entraînement et optimisation

Le processus d'entraînement a débuté par une évaluation initiale avec les paramètres par défaut afin d'obtenir une per-

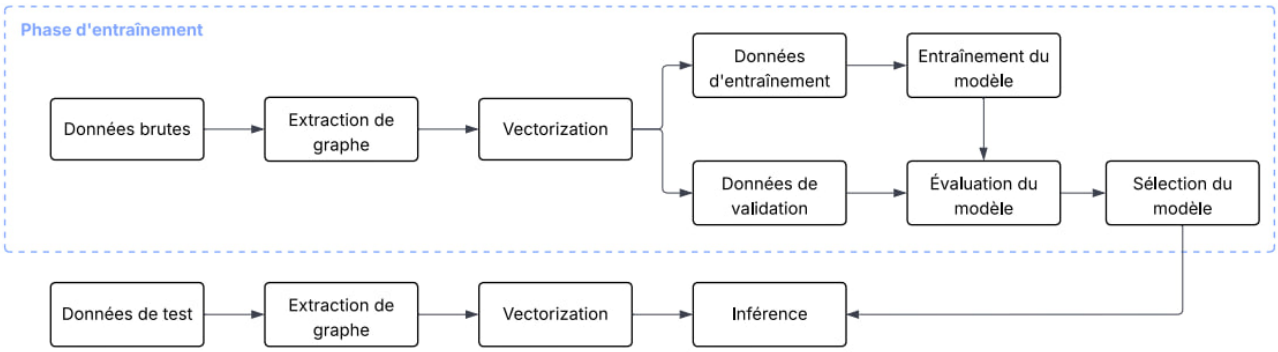

Figure 1: Détection des logiciels malveillants

formance de base. Nous avons ensuite appliqué une validation croisée aléatoire pour effectuer un réglage automatique des hyperparamètres. L'espace exploré incluait le taux d'apprentissage, la profondeur des arbres, les ratios de sous-échantillonnage horizontal et vertical, les termes de régularisation L1/L2 et le nombre d'estimateurs. Ce tuning a permis d'atteindre une configuration optimisée assurant un bon compromis entre biais, variance et performance.

3.4 Équilibrage des classes

Pour atténuer le fort déséquilibre entre les 453 familles de logiciels malveillants, une stratégie d'apprentissage pondéré a été adoptée. Des poids d'échantillons ont été calculés pour chaque étiquette à l'aide de la méthode `compute_sample_weight` de scikit-learn avec la stratégie balanced, puis moyennés sur l'ensemble des labels. Cette pondération a permis de survaloriser les familles rares tout en réduisant l'influence des classes dominantes, améliorant ainsi la détection des minorités sans nuire à la performance globale du classifieur XGBoost.

4 Évaluation et Résultats

Compte tenu de la nature multiclasse de notre problème avec 453 catégories potentielles, nous nous sommes concentrés sur le score F1 macro comme métrique d'évaluation principale. Notre modèle XGBoost optimisé a atteint un score F1 macro de 0.5852 sur l'ensemble de test. La figure 2 montre une courbe ROC utilisée pour évaluer la performance du modèle de classification. L'AUC micro-moyenne est de 0.98, indiquant une excellente capacité de discrimination, tandis que l'AUC macro-moyenne est de 0.90 en ne considérant que les classes qui ont au moins une prédiction positive (51 classes ont été ignorées), ce qui suggère une légère variation des performances entre les classes. La figure 3 illustre la diminution de la log-loss au fil des itérations. La perte de validation se stabilise autour de 0.06914 à l'itération 610, indiquant un bon apprentissage sans surajustement. L'écart réduit entre les courbes d'entraînement et de validation montre une bonne généralisation du modèle.

5 Discussion

La combinaison de la vectorisation par hachage, de la SVD tronquée et de l'approche d'apprentissage équilibré s'est avérée particulièrement efficace pour traiter les séquences d'instructions brutes parmi les 453 familles de logiciels malveillants. La réduction de dimensionnalité a atteint un équilibre entre efficacité computationnelle et performance du modèle, réduisant le temps d'entraînement de 85%.

Plusieurs résultats clés sont ressortis de notre approche. Premièrement, les séquences d'instructions brutes contenaient suffisamment d'information discriminante pour une classification fine des familles de logiciels malveillants, sans nécessiter d'analyse plus coûteuse des graphes de flux de contrôle. Deuxièmement, l'efficacité mémoire de la vectorisation par hachage a permis le traitement de grands jeux de données qui seraient impraticables avec des méthodes de vectorisation traditionnelles basées sur le comptage. Troisièmement, la SVD tronquée a non seulement réduit la dimensionnalité mais a aussi aidé à atténuer le bruit potentiel dans les caractéristiques hachées, conduisant à une classification plus robuste.

6 Conclusion

Ce travail établit que le traitement léger des séquences d'instructions permet une classification évolutive des logiciels malveillants sans sacrifier la granularité. Notre représentation optimisée à 150 caractéristiques atteint un équilibre efficace entre précision et efficience, avec un entraînement du modèle se terminant en moins d'une heure. Deux directions prometteuses pour des recherches futures incluent : (1) des stratégies de classification hiérarchique pour les familles rares, et (2) des techniques de réduction de dimensionnalité adaptatives. Les résultats démontrés fournissent une base pratique pour construire des systèmes d'analyse de logiciels malveillants à grande échelle capables de gérer des centaines de familles distinctes.

A Annexe

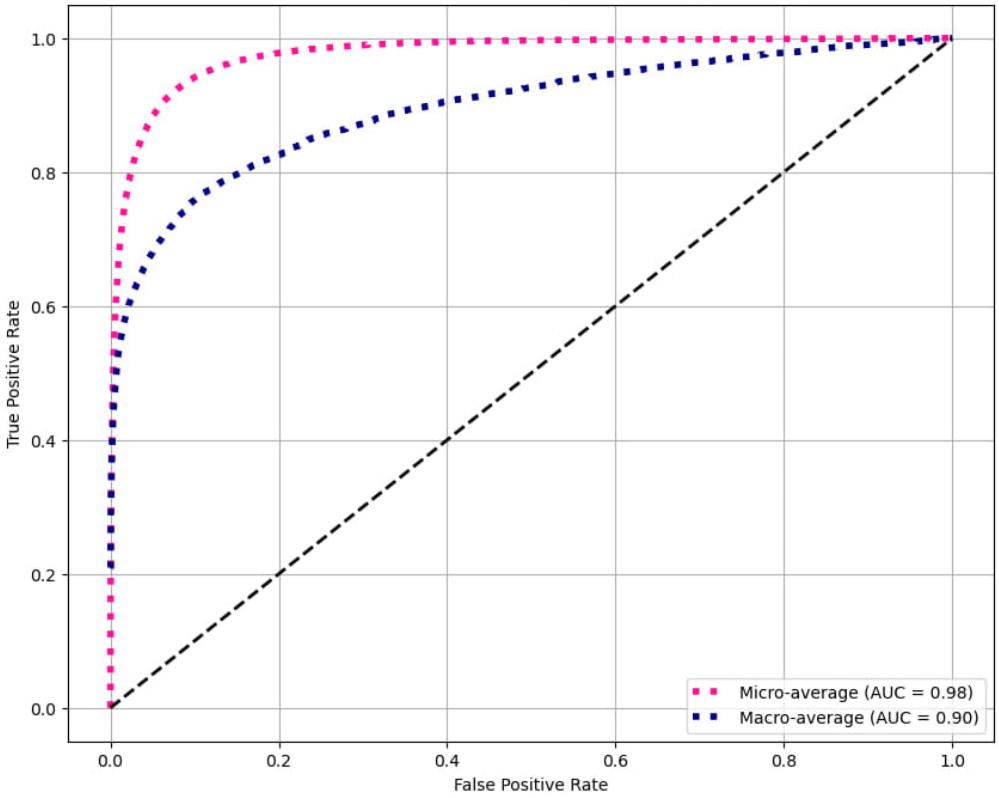

Figure 2: Courbe ROC de XGBoost

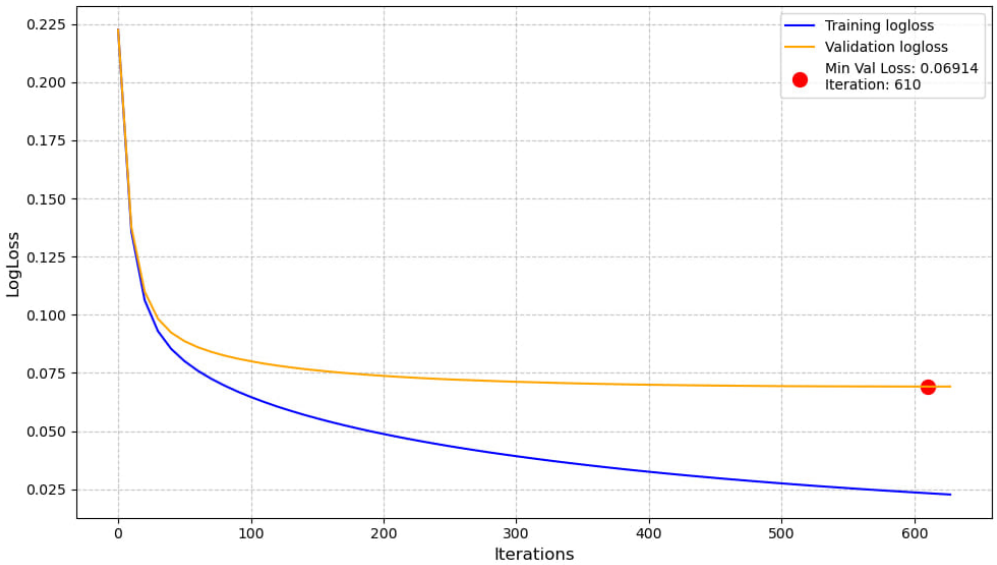

Figure 3: Courbe d'apprentissage de XGBoost