Méthodes d'apprentissage

Maouche Dalia Watiotienne Henry

July 14, 2024

Contents

1 Introduction

La classification, dans le domaine de l'apprentissage automatique, implique le tri d'objets en catégories, définies par des propriétés communes. Cette discipline se scinde en deux approches principales : supervisée et non supervisée. Notre projet se concentre sur l'approche supervisée, où nous utilisons une base de données étiquetée pour former des modèles à distinguer les "Malwares" des "goodwares".

L'objectif principal de ce projet est d'identifier l'algorithme le plus efficace pour développer un modèle précis, capable de prédire si un logiciel est un "Malware" ou un "goodware". À cette fin, nous examinerons diverses techniques d'apprentissage,comme la Régression Logistique, les K-NN (K plus proches voisins), les Forêts Aléatoires, les SVM (Support Vector Machines) et les Arbres de Décision. Nous intégrerons également des stratégies d'ensemble telles que le Bagging et le Boosting. En parallèle, différentes méthodes d'échantillonnage, comme le NearMiss (sous-échantillonnage) et le SMOTE (sur-échantillonnage), seront testées pour optimiser la performance des algorithmes dans la classification de la variable 'Label'.

2 Etudes préliminaire

2.1 Description des données

Notre ensemble de données, comprennent un total de 4465 observations et 242 variables. Chaque variable semble être associée à une autorisation ou à une fonctionnalité spécifique, dans le contexte de logiciel. Les valeurs binaires des variables (0 ou 1) indiquent la présence ou l'absence de certaines autorisations ou fonctionnalités pour chaque observation.

Les autorisations couvrent un large éventail de domaines, allant de l'accès aux téléchargements et au système de fichiers cache à la localisation, au réseau, au Wi-Fi, à la caméra... Outre les autorisations, certaines variables semblent refléter des actions spécifiques, telles que l'installation de packages, la gestion de l'alimentation de l'appareil, l'accès à des services système, ou encore des interactions avec d'autres applications.

2.2 Statistique descriptive

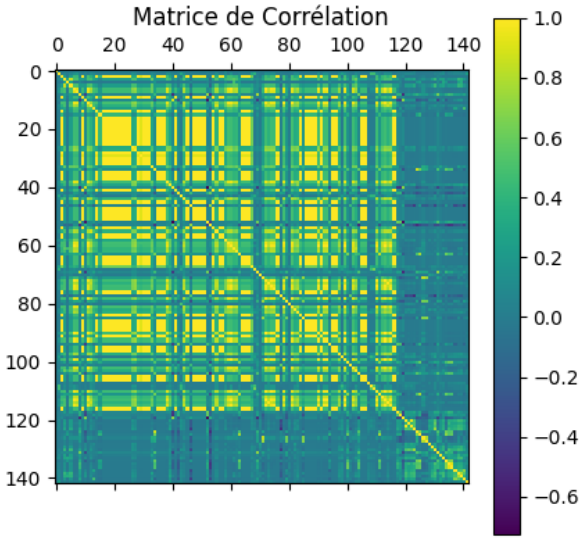
Une étude approfondie de notre jeu de données révèle l'existence de nombreux doublons : 368 classés comme goodware et 3434 comme malwares, ce qui fait un total de 3802 doublons. Ces répétitions pourraient influencer de manière significative les taux d'erreur dans nos analyses futures.

Les statistiques descriptives, consultables via le lien Statistiques Descriptives, on constate le déséquilibre entre les classes, avec 20 % de goodwares contre 80 % de malwares. Pour remédier à ce déséquilibre, des méthodes de rééquilibrage telles que SMOTE ou NEARMISS seront envisagées. Par ailleurs, nous avons observé que plusieurs variables ne contiennent qu'une seule valeur distincte, ce qui les rend inefficaces pour la distinction entre les labels. Une sélection des variables est donc nécessaire.

Face au grand nombre de variables, nous avons opté pour une stratégie de sélection sélective. Nous utilisons la fonction gtsummary de R pour identifier les variables les plus significatives. Cette

fonction applique un test de Pearson pour les échantillons de plus de cinq observations et un test de Fisher dans le cas contraire, permettant d'évaluer l'indépendance entre deux groupes de variables catégorielles. Les résultats de ces tests, y compris les p-values, sont disponibles à l'adresse suivante : p-values des variables.

En établissant un seuil de signification à 5%, nous sélectionnons uniquement les variables dont la p-value est inférieure à ce seuil. Cette méthode nous a permis de réduire le nombre de variables à 142, améliorant ainsi la pertinence et l'efficacité de notre modèle de classification.



La matrice de corrélation suggère une prudence particulière lors de l'utilisation de modèles linéaires tels que la régression linéaire, en raison de la présence notable de multicollinéarité entre les variables. Bien qu'une analyse des correspondances multiples (ACM) puisse être envisagée pour atténuer ces problèmes en réduisant la dimensionnalité et en éliminant la redondance, dans ce contexte précis, nous choisissons de ne pas procéder à une ACM. Cette décision peut être motivée par le désir de conserver l'interprétabilité des variables originales dans le modèle.

Dans le cadre de l'apprentissage autonome, la normalisation est une étape cruciale, particulièrement pour certains algorithmes qui sont sensibles aux échelles des variables. La normalisation permet d'équilibrer l'importance de différentes caractéristiques en les mettant à la même échelle. Cependant, dans notre cas où chaque variable est binaire (valeurs 0 ou 1), la question de la normalisation devient moins critique. Puisque toutes les variables sont déjà sur une échelle commune, l'étape de normalisation pourrait ne pas apporter de valeur ajoutée significative. On choisit donc de ne pas appliquer de normalisation.

3 Algorithmes de classifications

Dans cette partie, nous nous intéressons aux modèles de régression suivants : régression logistique, CART (arbres de décision), forêt aléatoire et SVM (machines à vecteurs de support). Pour chaque algorithme, nous appliquons une méthode d'ensemble appropriée. Nous appliquerons également une méthode de ré-échantillonnage.

3.1 Méthodologie générale

Expliquons notre méthodologie pour évaluer les performances d'un modèle. Pour chaque modèle cité précédemment, nous allons étudier ses performances avec et sans méthodes d'ensemble. Pour les algorithmes qui présentent un faible biais, par exemple, le CART profond, le SVM, et les K plus proches voisins (avec une valeur de k faible), nous utiliserons la technique du bagging. Le bagging consiste en un processus où plusieurs modèles de base sont entraînés sur des sous-ensembles aléatoires des données d'entraînement, puis leurs prédictions sont combinées pour obtenir une prédiction finale plus robuste. Remarque: La Random Forest est déjà une méthode d'ensemble qui intègre le principe du bagging dans sa conception de base car elle combine plusieurs arbres de décision.

Pour les algorithmes dont la variance est faible, c'est-à-dire ceux qui ne souffrent pas de surapprentissage, comme la régression logistique, nous utiliserons la méthode du boosting. Le boosting est une technique d'apprentissage automatique qui consiste à entraîner une série de modèles faibles, puis à combiner leurs prédictions de manière séquentielle en mettant l'accent sur les erreurs précédentes. Cela permet d'améliorer la performance globale du modèle en corrigeant progressivement ses erreurs antérieures.

Les erreurs données dans la suite représentent un intervalle de confiance à 95% obtenue avec 15 itérations. L'intérêt de réaliser 15 itérations en tirant à chaque fois une nouvelle base de test et d'entraînement réside dans la validation croisée. La validation croisée est une technique essentielle en statistiques et en apprentissage automatique qui permet d'évaluer la performance d'un modèle de manière plus robuste en évitant les biais dus à un découpage aléatoire unique des données. En répétant le processus avec différentes partitions des données, on obtient une estimation plus fiable de la performance du modèle sur l'ensemble du jeu de données, ce qui renforce la confiance dans les résultats obtenus.

Pour chaque algorithme, que ce soit avec ou sans l'utilisation de la méthode d'ensemble, nous avons recours à GridSearchCV pour estimer les meilleurs paramètres. Cependant, en raison de la durée de calcul prolongée, nous ne pouvons pas explorer en détail l'ensemble des hyperparamètres possibles. Par conséquent, nous allons sélectionner un nombre limité d'hyperparamètres à tester.

De plus nous utiliserons deux méthodes de rééchantillonnage, à savoir SMOTE et NEARMIS. SMOTE génère des exemples synthétiques pour les classes minoritaires en interpolant les données

existantes, tandis que NEARMIS sous-échantillonne la classe majoritaire en sélectionnant soigneusement les exemples basés sur leur proximité avec la classe minoritaire. Ces méthodes visent à traiter le déséquilibre des classes dans nos données d'entraînement pour améliorer la performance des modèles.

Nous étudierons la matrice de confusion pour l'algorithme qui présente les meilleurs performance.

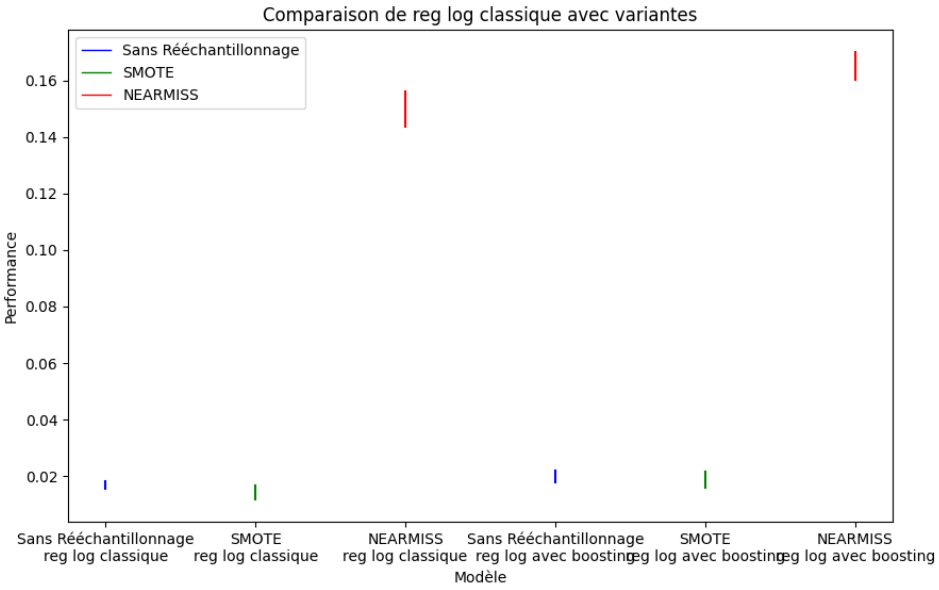
3.2 Régression logistique

La régression logistique est spécifiquement conçue pour les tâches de classification binaire. C'est un algorithme qui ne risque pas le sur-apprentissage, et il est rapide en temps d'exécution. De plus, les coefficients de la régression logistique peuvent être interprétés directement. C'est donc un algorithme incontournable à tester.

Nous allons utiliser la fonction GridSearchCV En pratique, ajuster C aide à trouver un équilibre entre la simplicité du modèle (prévention du surajustement) et son adaptation aux données d'entraînement (capacité de généralisation). Nous n'avons pas cherché à optimiser les paramètres comme le solver qui peut influencer la vitesse de convergence ou le type de régularisation (l1, l2, elasticnet, none). Chaque type a un impact différent sur la pénalisation des coefficients du modèle.

Résultats :

Erreur en %	Sans rééchantillonnage	SMOTE	NEARMISS
Régression logistique avec boosting	[1,799,2,215]	[1.615,2.149]	[16.023,16.999]
Régression logistique classique	[1,543,1.826]	[1.183,1.661]	[14.376,15.599]



En examinant les résultats, on constate une tendance sera applicable à tous nos modèles : L'emploi de NEARMISS entraîne une baisse importante des performances.

A l'inverse ici l'utilisation de SMOTE réduit les erreurs, bien que l'amélioration ne soit pas significative compte tenu du chevauchement des intervalles. Concernant le boosting, il apparaît qu'il n'apporte pas d'avantage significatif, tendant même à légèrement diminuer la performance.

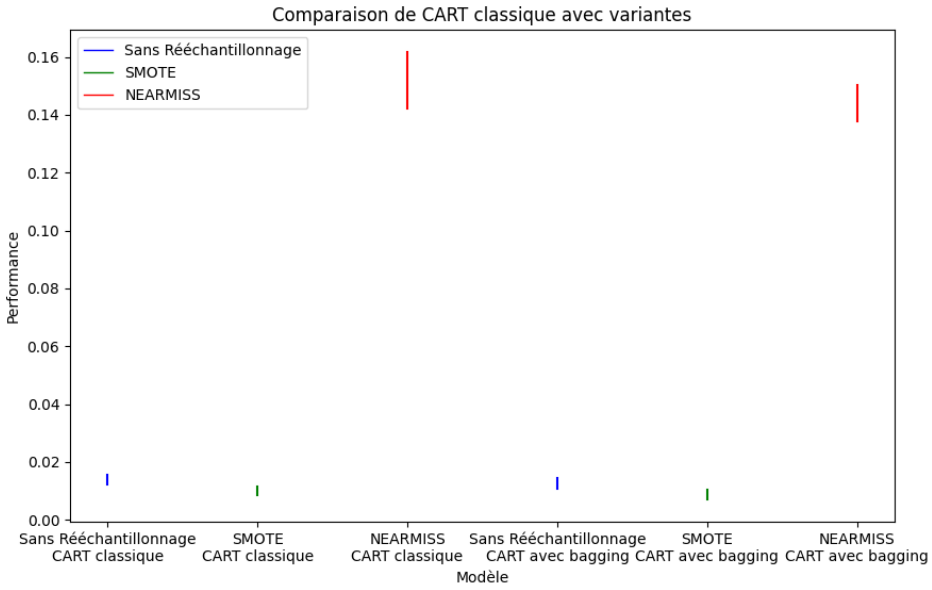
3.3 CART

L'algorithme CART présente plusieurs avantages dans notre contexte de classification. Tout d'abord, sa capacité à modéliser des relations complexes permet de capturer les schémas non linéaires et les interactions entre les différentes caractéristiques des logiciels. Par ailleurs, la structure arborescente aux arbres de décision offre une interprétation aisée des critères de décision, ce qui facilite la clarté des processus décisionnels. La robustesse aux valeurs aberrantes et la facilité d'optimisation des hyperparamètres font de CART un choix interréssant pour la classification, offrant une combinaison équilibrée entre performances prédictives et interprétabilité.

Nous avons cherché a optimiser le paramétre max_deep avec GridSearchCV. C'est un hyperparamètre qui définit la profondeur maximale de l'arbre. En ajustant max_depth, on cherche la profondeur optimale qui équilibre la généralisation et la capacité du modèle à apprendre des détails à partir des données d'entraînement. Il existe d'autre paramétre que nous n'avons pas cherché à optimiser comme le nombre minimum d'échantillons requis pour diviser un nœud. Un nombre plus élevé peut prévenir le surajustement, ou le nombre minimum d'échantillons requis pour être à un nœud feuille.

Résultats :

	Sans rééchantillonnage	SMOTE	NEARMISS
CART avec bagging	[1.062,1.447]	[0.707,1.038]	[13.782,15.035]
CART classique	[1.210,1.550]	[0.855,1.140]	[14.222,16.172]



Les observations pour SMOTE et NEARMISS sont identique que précédemment. Cependant, dans ce contexte, l'amélioration apportée par SMOTE est significative. Par ailleurs, bien que le bagging tende à réduire l'erreur, cette diminution n'est pas statistiquement significative.

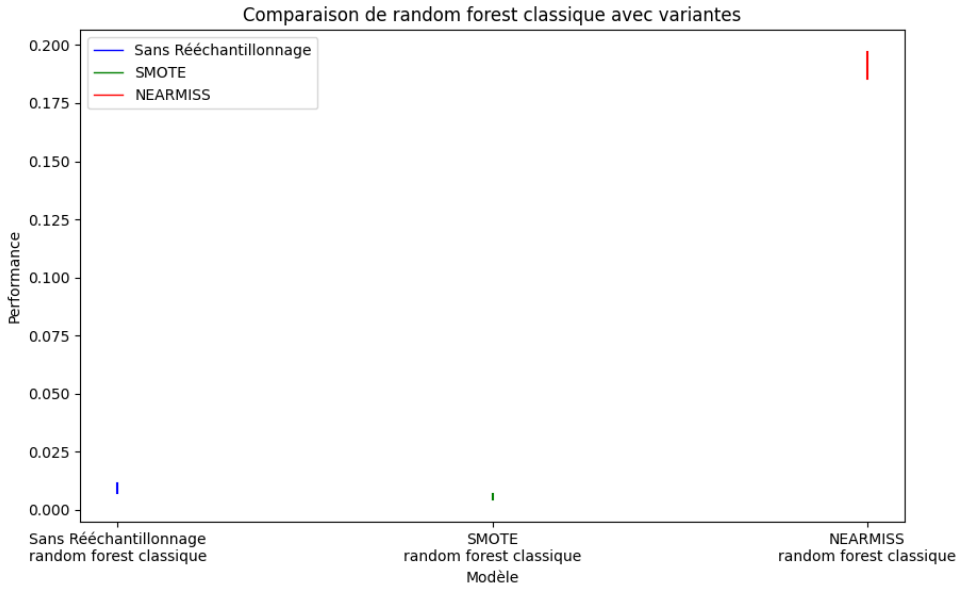
3.4 Forêt aléatoire

La forêt aléatoire tire son premier avantage de sa capacité à renforcer la robustesse et la stabilité des prédictions en agrégeant les résultats de plusieurs arbres individuels. Elle préserve également les avantages des arbres de décision, tels que la modélisation efficace de relations complexes. Elle est simple à implémenter et présente généralement de très bons résultats. La forêt aléatoire s'adapte bien à des ensembles de données complexes comme le nôtre, où de nombreuses variables interagissent pour déterminer la nature des logiciels (malveillants ou non).

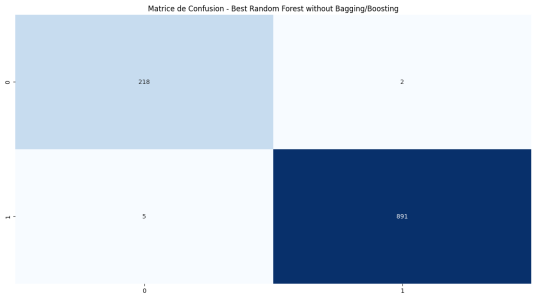
Avec GridSearchCV nous cherchons également à optimiser le paramètre max_deep, il joue le même role que dans CART.

Résultats :

	Sans rééchantillonnage	SMOTE	NEARMISS
Random Forest	[0.743,1.157]	[0.455,0.692]	[18.546,19.697]



Même remarque que précédemment pour NEARMISS, et SMOTE améliore significativement les performances.



La matrice de confusion révèle le faible taux d'erreur : seulement 7 individus sont mal classifiés. Les erreurs sont réparties entre faux positifs et faux négatifs. De plus, le modèle commet moins d'erreurs en classant des individus comme goodware (0) lorsqu'ils sont en réalité des malwares (1), ce qui est une erreur plus grave que l'erreur inverse.

3.5 SVM (Support Vector Machine)

Les machines à vecteurs de support (SVM) sont particulièrement efficace dans un contexte de classification. L'algorithme SVM vise à trouver l'hyperplan optimal qui sépare efficacement les différentes classes de logiciels (malware et goodware) dans un espace multidimensionnel.

L'une des forces majeures des SVM réside dans leur capacité à traiter des espaces de caractéristiques de grande dimension, ce qui notre cas.

Principaux hyper paramètres de cette méthode : le choix du noyau(kernel) pour la redimensionnement d'espace,on trouve plusieurs types de fonction noyau et ce qu'on va traiter dans notre modèle le noyau Gaussien(ou RBF) qui utilise deux paramètres 'C' et 'gamma' et le noyau linéaire

qui utilise le paramètre 'C'.Pour ces deux types de paramètres on va les testez pour différente valeurs afin de trouver les meilleures configurations pour notre modèle.

Pour des raisons de temps de calcul trop important nous ne cherchons pas à optimiser les paramètre. Mais il est possible de faire varier les paramétres suivant :

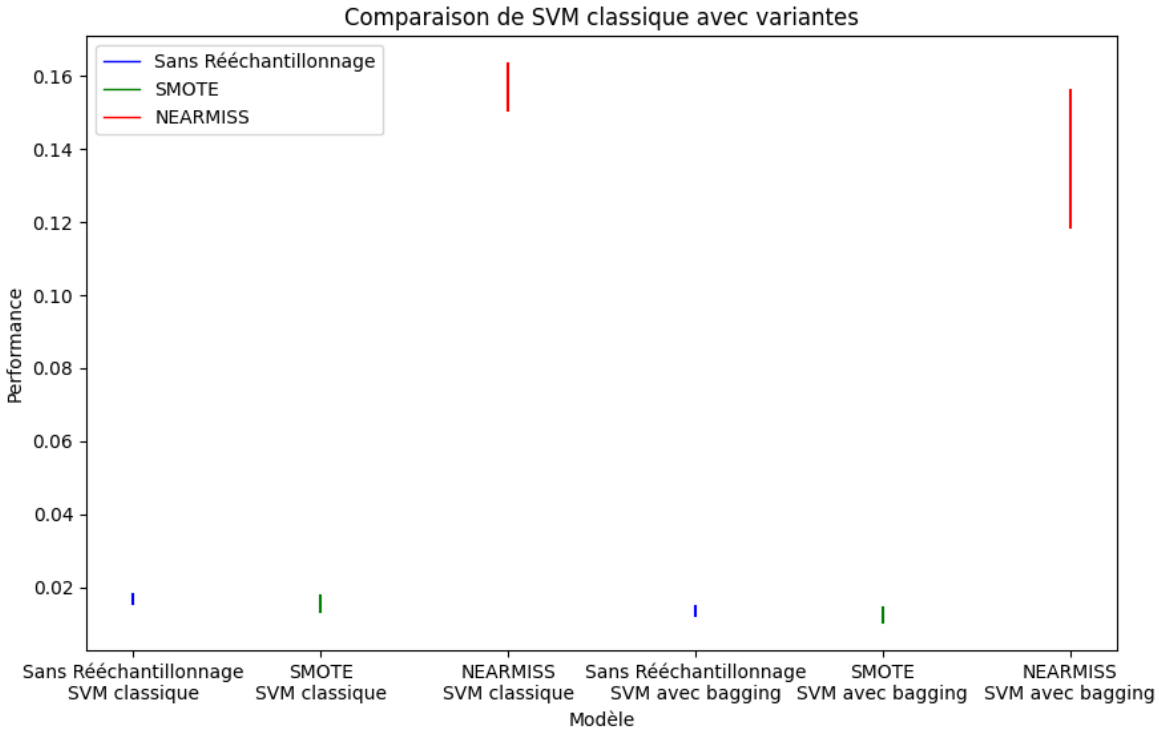
- C: C'est le paramètre de pénalité de l'erreur de classification.un C élevé vise à minimiser les erreurs de classification au détriment de la marge, ce qui peut conduire à un surajustement.

- kernel: Le choix du noyau transforme les données d'entrée dans un espace de caractéristiques plus dimensionnel où une séparation linéaire est possible. Les options courantes incluent linear, poly, rbf et sigmoid. Le noyau RBF est souvent un choix par défaut efficace pour les problèmes non linéaires.

- gamma: Pour les noyaux non linéaires comme RBF, gamma détermine l'influence d'un seul exemple d'entraînement. Une valeur élevée de gamma signifie que les points de données doivent être très proches pour influencer l'un l'autre, ce qui peut conduire à un surajustement.

Résultats :

	Sans rééchantillonnage	SMOTE	NEARMISS
SVM avec Bagging	[1.212,1.471]	[1.038,1.447]	[11.849,15.617]
SVM classique	[1.549,1.820]	[1.330,1.753]	[15.054,16.343]



Les constatations concernant l'utilisation de SMOTE et NEARMISS demeurent inchangées par rapport aux observations précédentes. L'application du bagging semble réduire l'erreur, mais cette amélioration n'est statistiquement significative que dans les cas sans SMOTE.

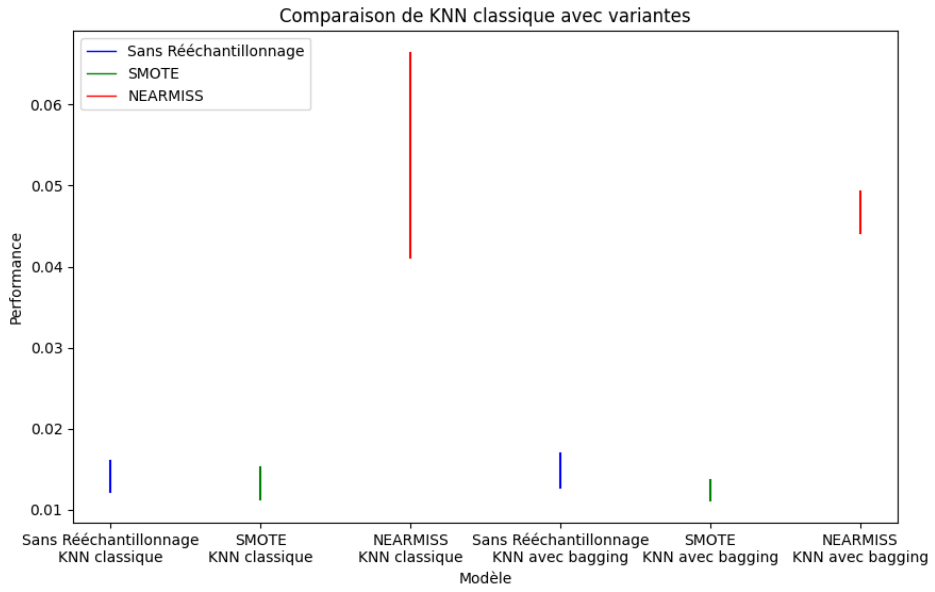
3.6 K plus proche voisin

Ce modèle consiste à choisir les K données les plus proches du point étudier afin d'en prédire sa valeur; pour en fait déterminer la classe à laquelle appartient un cas, il calcul pour chaque point du graphique le nombre d'éléments majoritaire autour de lui.
Le K-NN est un des plus simple algorithme à mettre en oeuvre ,de plus il s'adapte facilement aux changements des données. Donc le choix de l'algorithme K-NN semble être pertinent dans le cadre de notre étude, il pourrait être bénéfique d'ajuster ce modèle à nos données.

Cette méthode utilise deux principaux hyper paramètre :la fonction de similarité pour comparer les individus dans l'espace de caractéristiques en mesurant la similarité entre deux vecteurs à l'aide de distances telles que la distance Euclidienne, distance de Manhattan, distance maximale qui sont les fréquemment utiliser; et le nombre K qui décide combien de voisins influencent la classification (le choix de K est important: un K trop bas peut rendre notre modèle sensible au bruit,tandis qu'une valeur trop élevée peut introduire un lissage excessif). C'est ce paramètre que nous avons essayé d'optimiser.

Résultats :

	Sans rééchantillonnage	SMOTE	NEARMISS
KNN avec bagging	[1.279,1.700]	[1.116,1.370]	[4.4410,4.932]
KNN classique	[1.217,1.603]	[1.135,1.529]	[4.117,6.634]



L'amélioration apportée par SMOTE n'est pas statistiquement significative. De même, bien que l'utilisation de NEARMISS augmente l'erreur, son impact est moins prononcé comparé à d'autres algorithmes. Concernant le bagging, il ne semble pas modifier de manière significative les résultats.

3.7 Comparaison des Résultats

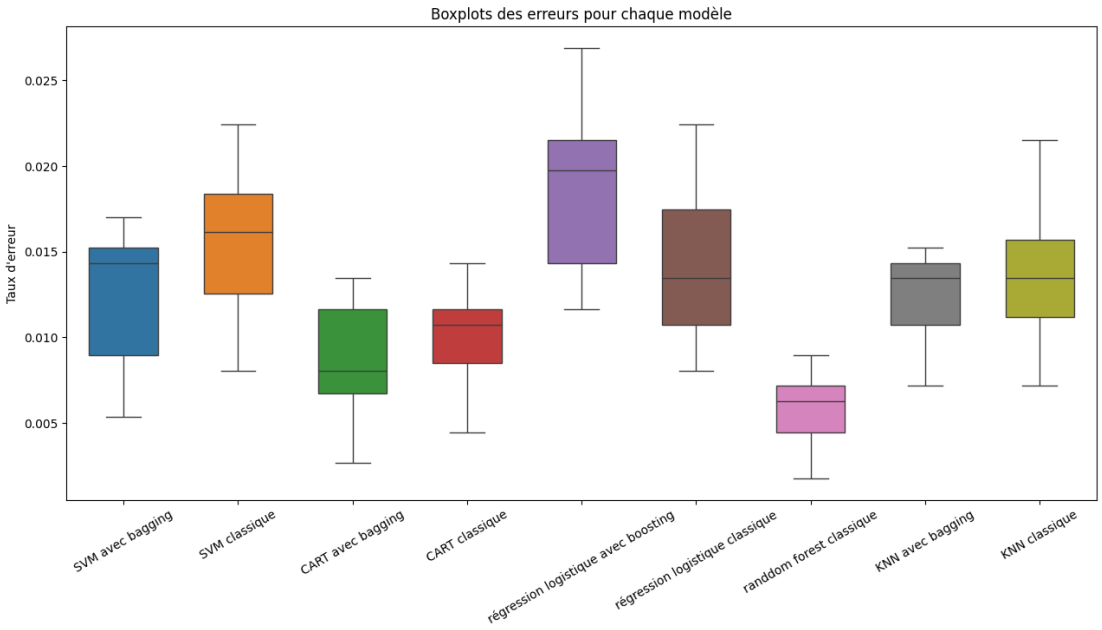


Figure 3.7.1: Erreur avec SMOTE

Tous nos modèles ont des efficacités comparables ; cependant, la forêt aléatoire semble se distinguer et être le modèle le plus efficace. L'erreur se situe entre 1 et 2 % pour tous les modèles, sauf pour la forêt aléatoire où l'on tombe en dessous de 1 %. Ces résultats très élevés peuvent s'expliquer par la forte redondance des individus dans la base de données. Ces doublons facilitent grandement la classification.

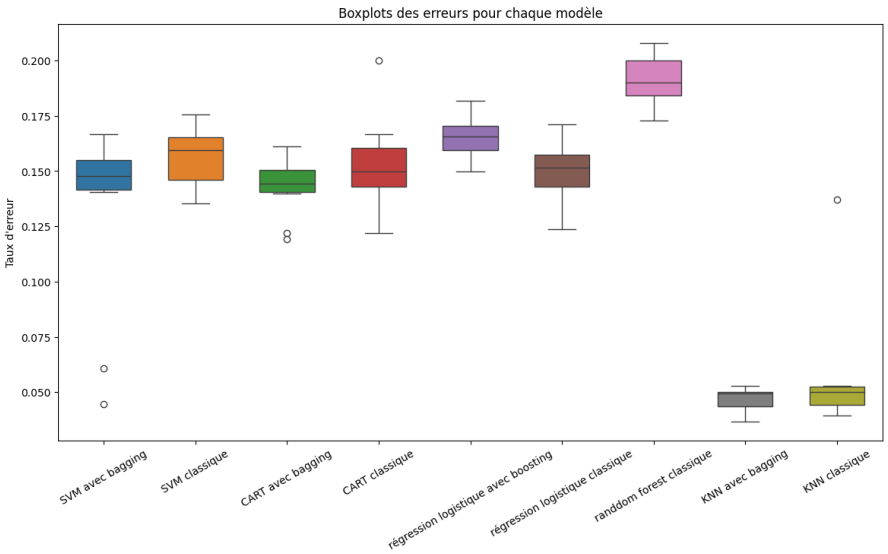


Figure 3.7.2: Erreur avec NEARMISS

Les erreurs avec NEARMISS sont considérablement élevées. Face à des classes déséquilibrées (80/20), un algorithme naïf qui classerait tous les individus dans la classe majoritaire aboutirait à une erreur de 20 %. Comparée à ce seuil, la performance de nos algorithmes semble insuffisante. Pour le KNN, cependant, l'augmentation de l'erreur est moins prononcée. Cette hausse général du taux d'erreur peut être attribuée au sous-échantillonnage effectué par NEARMISS, qui équilibre les deux classes de la base d'entraînement en se basant sur l'effectif le plus faible, entraînant ainsi une perte significative d'information.

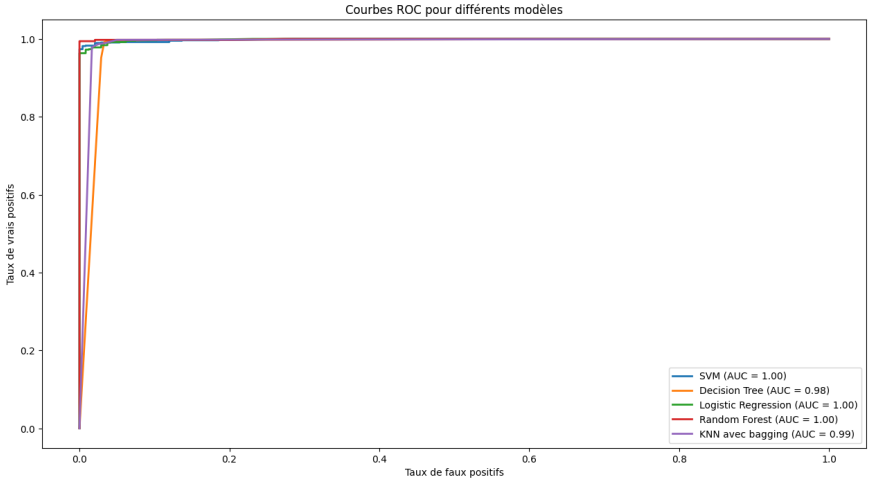
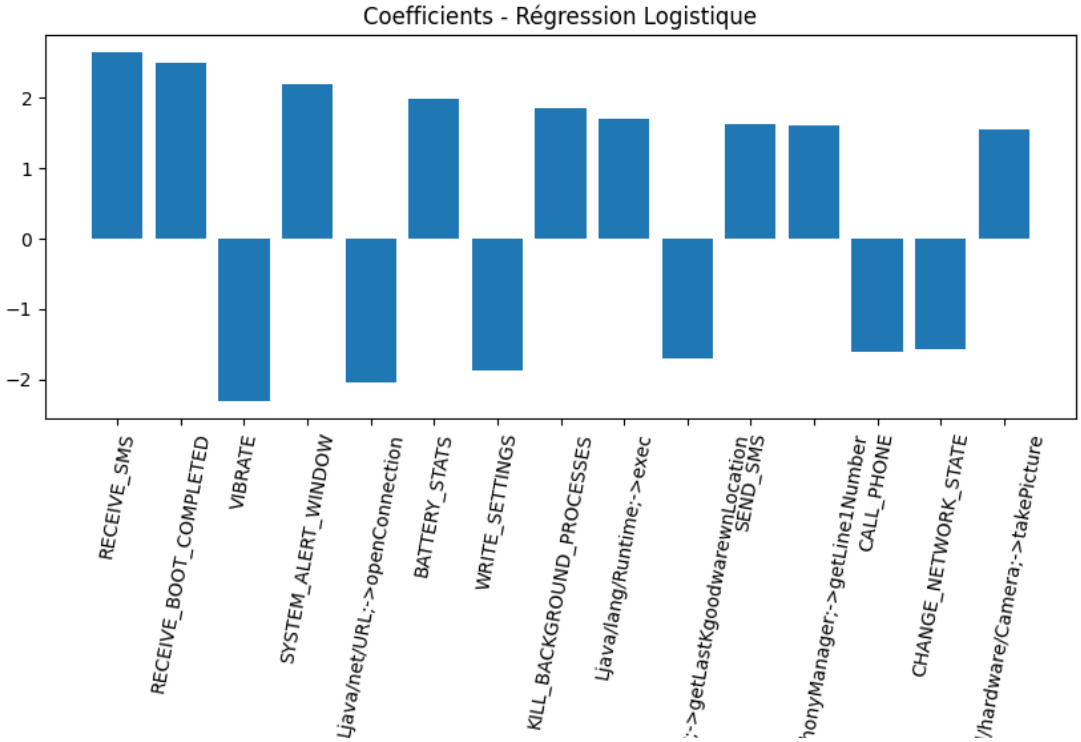


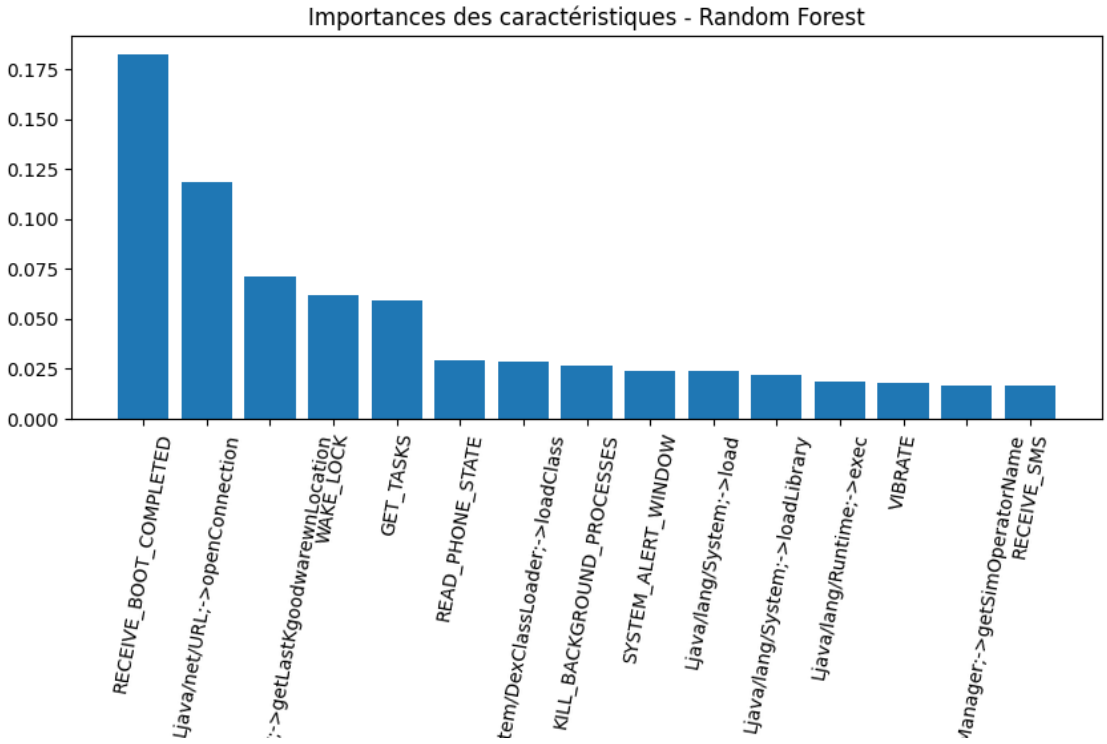
Figure 3.7.3: Avec NEARMISS

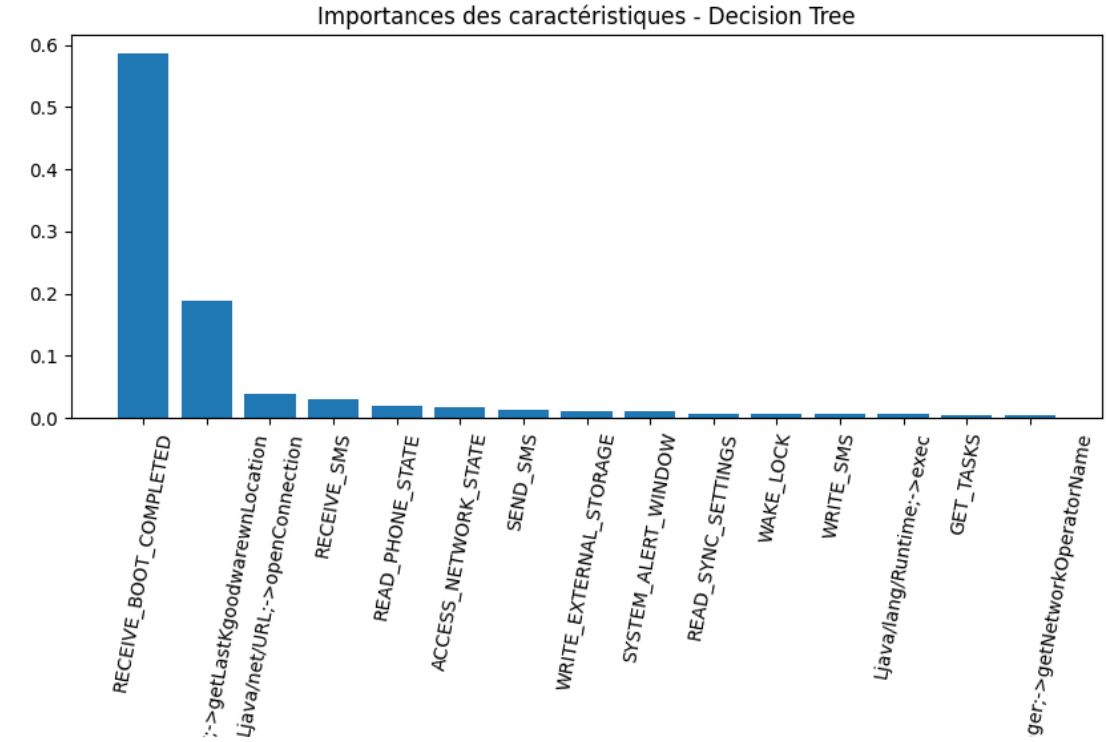
Les courbes ROC sont quasiment parfaites mais on identifie a nouveau la Random Forest comme étant l'algorithme le plus performant (la courbe la plus en haut à gauche).

3.8 Variable d'importance

Certains algorithmes permettent d'interpréter le rôle des variables, comme c'est le cas pour la régression logistique, CART et la forêt aléatoire. On représente pour ces 3 algorithmes les 15 variables les plus influentes.







Nous observons que certaines variables, telles que 'RECEIVE_BOOT_COMPLETED', 'SYSTEM_ALERT_WINDOW' ou 'RECEIVE_SMS', figurent avec différents degrés d'importance dans

les trois graphiques. Leur présence récurrente est cohérente avec les résultats des tests statistiques antérieurs, où elles ont montré des p-values inférieures à 0.01, ce qui confirme que ces variables sont efficaces pour discriminer les labels. L'ensemble des variables présentes dans ce graphique ont aussi des p-value très faible.

4 Conclusion

En conclusion, cette étude a exploré l'utilisation de diverses méthodes de classification supervisée pour distinguer les logiciels malveillants (malware) des logiciels légitimes (goodware). L'analyse a inclus des algorithmes telles que la Régression Logistique, CART, Forêt Aléatoire, SVM et K-NN, en effectuant d'éventuelles méthodes de ré échantillonnage.

Parmi tous les modèles testés, la Forêt Aléatoire s'est révélée être la plus performante. Ce modèle également permis d'identifier les variables les plus influentes dans la distinction entre malwares et goodwares. Cette identification des variables clés est cruciale, car elle offre des informations sur les caractéristiques qui sont les plus pertinentes pour la classification.

Cependant, il est crucial de noter que les taux d'erreur extrêmement faibles observés dans l'étude sont principalement attribuables à la présence de doublons dans l'ensemble des données. Cette redondance a probablement facilité la classification, conduisant à une amélioration artificielle des performances des modèles. Cette observation souligne l'importance de considérer la qualité et l'unicité des données lors de l'évaluation des méthodes de classification en apprentissage automatique, en particulier dans le domaine de la détection de logiciels malveillants.