Yoann KERGOSIEN
Alistair RAMEAU

Cybersécurité du logiciel - 5ème année

Intelligence Artificielle et Détection d'Intrusion

École d'ingénieurs
ensibs
Université Bretagne Sud

2021-2022

Abstract

L'utilisation de l'intelligence artificielle connaît un grand essor dans son application pour la détection d'intrusion des systèmes d'information. Nous avons pu, lors de ce TP, implémenter différents algorithmes d'intelligence artificielle pour analyser des logs. Nous avons pu faire un classement des algorithmes en fonction de leurs statistiques sur la détection de logs traduisant une attaque. Deux points clefs que nous avons ainsi relevés pour l'implémentation d'intelligence artificielle sont le choix de l'algorithme et la manière de représenter les données à analyser.

Introduction

L'intelligence artificielle peut avoir beaucoup d'applications, notamment dans la sécurité informatique. Nous pouvons notamment penser à la détection d'attaques réseaux, car les équipements du systèmes d'information peuvent remonter un nombre important de logs, et certains logs traduisant une attaque peuvent être noyés dans ce flot d'informations. Ainsi, l'utilisation d'intelligence artificielle pourra permettre de faire ressortir ces logs aux yeux des équipes de sécurité. Nous allons étudier lors de ce TP la mise en place de telle intelligence artificielle et comprendre les problématique liées au traitements des données, notamment pour la traduction des informations en nombres compréhensibles par l'intelligence artificielle.

Nous allons tout d'abord expliquer comment nous avons configuré notre environnement d'entraînement et de test sur nos intelligences artificielles.

Nous allons ensuite comparer les intelligences artificielles sélectionnées pour l'exercice et expliquer nos résultats.

Enfin, nous allons conclure sur l'utilisation possible de l'intelligence artificielle et les perspectives d'avenir.

Project Structure

Afin de pouvoir utiliser notre intelligence artificielle, il est nécessaire de créer un environnement adapté à celle-ci. Nous voulons travailler sur de la détection de logs indiquant une attaque. Pour ce faire, il nous est fourni une liste de log dans de lourds fichiers xml. Ces logs contiennent beaucoup d'informations, notamment le type de protocole utilisé, ce qui va nous permettre de ne pas mélanger les logs. Ces logs doivent être envoyés à un serveur ElasticSearch afin d'être centralisés. Pour ce faire nous allons

utiliser l'API python d'Elasticsearch. Ainsi, nous avons des fonctions de lecture de fichier, parsing d'xml, d'envoi de données au serveur Elasticsearch.

Une fois les données dans ElasticSearch, nous avons pu nous intéresser à la récupération de données, toujours par l'API python d'Elasticsearch. Cela passe par l'utilisation de requêtes DSL, pour récupérer des données, notamment en fonction des protocoles utilisés.

Une fois que nous avons bien défini cette structure, nous pouvons nous attaquer à l'entraînement de notre intelligence artificielle. Tout d'abord, cela passe par la phase de pre-processing, où nous allons vectoriser nos logs. En effet, pour pouvoir utiliser une intelligence artificielle, il nous est nécessaire de respecter un format pour les valeurs d'entrée. Nous allons donc mettre sous format de vecteur nos logs, chaque champ portant une information. A la fin de cette vectorisation, nos logs composés initialement de 20 champs textuels, produisent un vecteur composé de 164 champs en float. Les entrées de l'intelligence artificielle ne pouvant être qu'en float, nous avons dû convertir des informations textuelles en float, tout en perdant le moins d'information possible, ce qui n'est pas chose aisée. Ainsi, nous avons parfois simplement pu mettre le texte en float, parfois en plusieurs champs comme l'adresse IP, où nous avons 4 champs stockant chaque partie de l'adresse IP. Il n'a pas été possible de conserver toutes les informations non plus, notamment pour les payloads des trames, qui ont été traduites en base 64. Ainsi, nous n'avons pas pu garder la valeur des champs, mais nous avons compté l'occurrence de chaque caractère dans la chaîne.

Une fois la vectorisation réalisée, nous avons créé un nouvel index dans ElasticSearch et envoyé ces vecteurs dedans.

La dernière étape avant l'entraînement de l'intelligence artificielle est la normalisation des vecteurs. En effet, lorsque nous récupérons nos vecteurs d'ElasticSearch, nous pouvons réduire les possibles variations et bruits des vecteurs en vue de l'entraînement de l'intelligence artificielle. En effet, certaines intelligence artificielle se basant sur la distance entre les champs du vecteur, il est nécessaire de réduire leur valeur pour diminuer le poids que certains pourraient apporter. Nous pouvons principalement penser aux champs date-time. En effet, lors de la vectorisation, nous avons transformé les champs de dates, qui étaient dans un format textuel, en timestamp, soit le nombre tic 1970. Ce qui fait que nous avions des valeurs énormes par rapport au reste des valeurs du vecteur, ce qui peut apporter un poids supplémentaire lors de la corrélation des valeurs. Nous avons donc réduit la valeur de ces champs par soustraction et division afin d'avoir des valeurs plus cohérentes par rapport au vecteur.

Enfin, nous pouvons passer au choix de nos intelligences artificielles. Nous avons choisi de faire le benchmark de plusieurs intelligences artificielles, à savoir Gaussian Bayes, KNN, et Random Forest. Pour pouvoir les tester, nous récupérons les vecteurs, que nous séparons en 5 groupes, avec la même proportion d'attaques dans chaque groupe. Ensuite, nous donnons en entraînement 4 de ces 5 groupes, et nous effectuons

un test de prédiction par l'intelligence artificielle pour le dernier set de données. Nous faisons tourner les sets de données pour prédire chacun des sets, afin d'avoir un plus grand nombre de résultats et obtenir un benchmark plus pertinent.

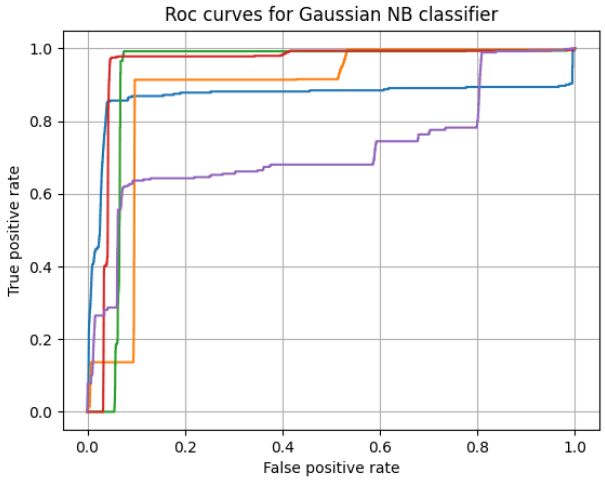
Évaluation

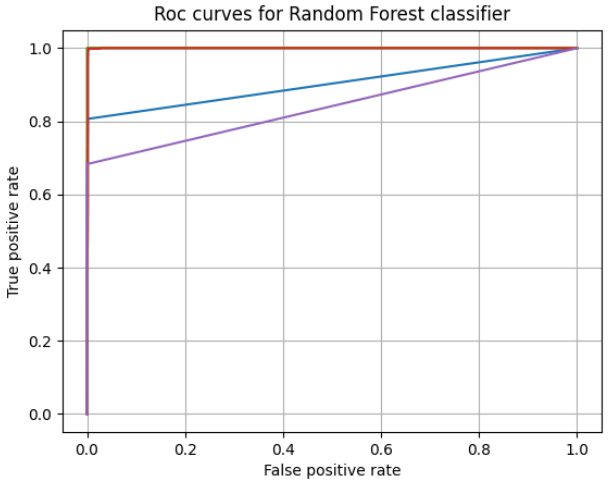
Pour l'évaluation, nous avons donc nos résultats obtenus par le benchmark des différentes intelligences artificielles. Pour rappel, nous allons comparer la Gaussian Bayes, la vérification par K voisins (KNN), ainsi que la Random Forest. Les métriques utilisées sont la précision de la prédiction, le taux de rappel, la courbe ROC ainsi que l'aire sous la courbe (AUC) de cette courbe. Nous avons aussi différencié en fonction du protocole, car la quantité de données disponibles est différente pour chacun d'eux.

Nous allons tout d'abord montrer nos résultats pour le protocole HTTPWeb en fonction des algorithmes de deep learning.

	Gaussian Bayes	KNN	Random Forest
F1	0.197454	0.665530	0.4985273
Precision	0.502747	0.719941	0.497063
Recall	0.578350	0.640943	0.50000
Area Under Curve	0.858514	0.640943	0.970782

Nous obtenons des courbes ROC pour chacun nos algorithmes.

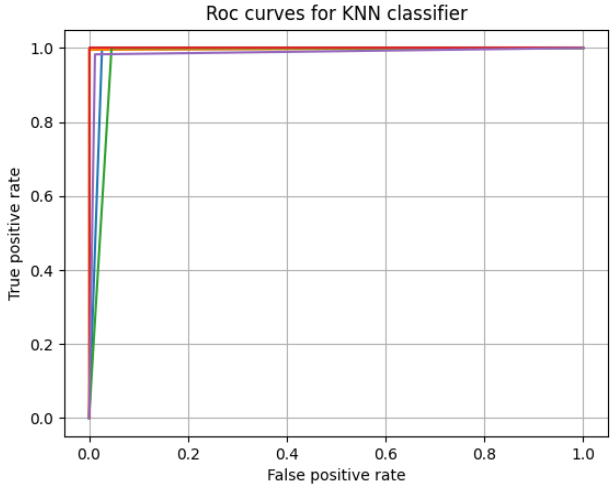


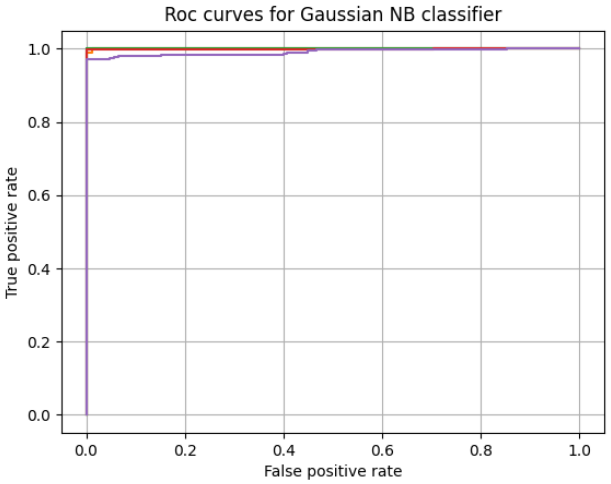


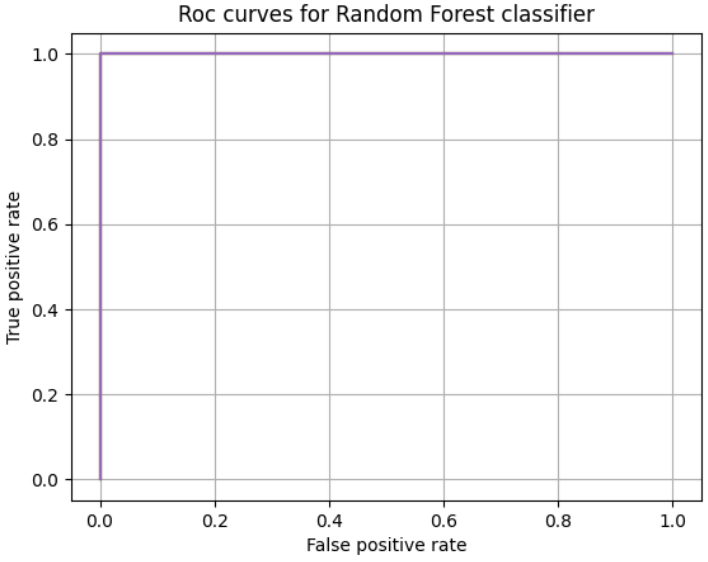
Nous n'avons pas pu faire ce diagramme pour KNN à cause du coût de calcul qui était trop important au vu du nombre de logs du protocole HTTPWeb et des performances de la méthode.

Ensuite, nous avons fait les mêmes benchmarks pour le protocole SSH.

	Gaussian Bayes	KNN	Random Forest
F1	0.807603	0.989891	0.998223
Precision	0.819402	0.990848	0.998190
Recall	0.826104	0.989133	0.998267
Area Under Curve	0.997975	0.989133	1.0







Par rapport à tous ces résultats, nous pouvons sortir 3 points.

Premièrement, la méthode la plus efficace d'un point de vue précision semble être KNN. Pour les flows SSH, la précision se compare avec celle de la méthode Random Forest, mais pour les flows HTTP Web la précision était bien supérieure à chaque fois. Le même constat peut être fait pour F1 et le Recall, statistiques qui sont toutes les 2 supérieures en moyenne pour KNN.

Le deuxième point notable est lui que d'un point de vue aire sous la courbe, la méthode la plus intéressante semble être de loin celle du Random Forest, surtout par rapport à la méthode KNN qui pèche dans ce milieu. Cela peut sembler étrange de prime abord, mais s'explique par le fait que les prédictions de la méthode Random Forest n'engrangent que peu de faux positifs, et presque chaque fois que la méthode indique un flow positif signe d'une attaque, celui-ci se révèle bel et bien être une attaque. Cette propriété semble particulièrement intéressante pour un algorithme d'un tel type car cela permet de ne pas relever de trop nombreux faux positifs et donc un trop grand nombre de logs signifiant une attaque, mais au contraire de limiter le logging aux flows qui sont réellement des attaques et qui sont donc pertinents à monitorer. Il faut cependant bien prendre en considération que certaines attaques non-évidentes pourraient ne pas être détectées. Il faudrait donc d'autres statistiques sur plus de cas pour bien se rendre compte du nombre de flows positifs mis de côtés à tort, et si ce pourcentage est dérangeant d'un point de vue sécurité.

Un troisième point qui n'est pas visible ici dans les graphiques mais qui a bien été observé est le temps de training et de détection des méthodes. Et le résultat est ici sans appel : la méthode KNN est très lente par rapport aux 2 autres, à un point où il nous a été impossible de tester un entraînement avec KNN avec l'intégralité des 500 000 flows HTTP Web, pour ensuite analyser les 191000 flows inconnus. Au contraire, les méthodes Naive Bayes et surtout Random Forest se sont révélées très rapides, surtout pour la 2ème méthode qui possède en plus des statistiques de détection intéressantes. Cette lenteur de KNN, malgré ses performances statistiques attrayantes, rendrait compliquée son utilisation dans le cadre d'un système temps réel.

Conclusion et Perspectives

Lors de ce TP, nous avons pu comprendre le fonctionnement du deep learning, et voir une de ses possibles applications dans le domaine de la sécurité. Les deux points clefs de la bonne implémentation d'une intelligence artificielle sont le choix même du type d'intelligence artificielle ainsi que la manière de représenter les données à analyser. Tout d'abord, la performance, que ce soit en coût de calcul ou en précision des prédictions, va être fortement dépendant de l'algorithme d'intelligence artificielle

choisie. Nous avons vu dans ce TP que KNN est très gourmand au niveau ressource contrairement aux Gaussian Bayes et Random Forest. Cependant, il est aussi nécessaire d'adapter la préparation et le format des données d'entrées à l'algorithme voulu. En effet, certains algorithmes vont accorder plus de poids sur certaines différences et ne vont pas gérer les corrélations de la même manière. Ainsi, la partie pre-processing et normalisation des données est une phase très importante de l'implémentation d'un système d'intelligence artificielle.

L'intelligence artificielle est un outil très puissant pour faire des corrélations et des prédictions sur des gros jeux de données. C'est pourquoi son utilisation dans l'analyse de logs est très prometteuse. La quantité des logs d'un SIEM pouvant être très conséquent, il peut être difficile pour les analystes de remarquer une attaque si elle est noyée dans la masse de logs normaux. L'utilisation d'intelligence artificielle permet donc de déjà trier les logs et de faire remonter les plus suspects. L'évolution des technologies de deep learning va donc avoir un impact direct sur la performance des détections d'attaques.