UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

L’utilisation de modèles d’intelligence artificielle pour la détection des intrusions dans les réseaux

Rapport Final

présenté à

Karim El Guemhioui, Coordonnateur

Hajar Moudoud, Superviseure

par

Mégane Labelle

Dans le cadre du cours

INF4173 - Projet synthèse

Gatineau

Le 25 avril 2025

Table des matières

1. Introduction 1

2. Collecte et préparation des données 2

2.1 La base de données NSL-KDD 2

2.2 Préparation des données 2

3. Modèles d'IA utilisés 4

3.1 Decision Tree 4

3.2 Random Forest 4

3.3 *K-Nearest Neighbors* (KNN) 5

3.4 Logistic Regression 5

3.5 Naïve Bayes 5

3.6 *Support Vector Machines* (SVMs) 5

4. Évaluation des modèles 6

4.1 Métriques utilisées 6

4.2 Comparaison générale des modèles 7

4.3 Comparaison des modèles pour l'attaque *DoS* 9

4.4 Comparaison des modèles pour l'attaque *Probe* 11

4.5 Comparaison des modèles pour l'attaque *R2L* 12

4.6 Comparaison des modèles pour l'attaque *U2R* 13

5. Conclusion 15

6. Bibliographie 16

1. Introduction

Le but du projet est de réaliser un prototype de système de détection des intrusions (IDS) avec des modèles préexistants d'Intelligence Artificielle (IA), plus précisément d'apprentissage automatique. L'objectif est de comparer leurs performances pour déterminer le modèle le plus efficace pour la détection d'attaques.

L'intelligence artificielle est une technologie puissante, et a une grande utilité dans le domaine de la cybersécurité, en permettant de reconnaitre une cyberattaque, et l'isoler de son environnement pour ne pas cause de dégâts [4], ce qui en fait un outil bien adapté pour la détection d'intrusions.

Pour entrainer les données, la base de données publique NSL-KDD est utilisée. Elle contient des données de trafic réseau, incluant des données normales et d'attaques. Il y a quatre types d'attaques présentes : *DoS*, *Probe*, *R2L* et *U2R*.

Pour réaliser le prototype, le langage Python est utilisé, avec la bibliothèque sklearn, où se trouvent les différents modèles de classification qui sont utilisés et comparés. Ces modèles sont : *Decision Tree*, *Random Forest*, *K-Nearest Neighbors*, *Logistic Regression*, *Naïve Bayes* et *Support Vector Machines*. Ils sont évalués et comparés selon les métriques d'exactitude, précision, rappel et F-mesure. Le graphe de la courbe ROC est également utilisé pour comparer les modèles dans leur précision.

Après la comparaison des modèles, il est clair que c'est *Random Forest* qui est le meilleur pour prédire les attaques en général, ainsi que pour les attaques spécifiques *DoS*, *Probe* et *R2L*, suivi de près par *Decision Tree*. Pour l'attaque *U2R*, il semble que le modèle le plus efficace soit *Logistic Regression*, mais ce n'est pas possible de dire avec certitude, puisque ce type d'attaque n'apparait que très peu dans la base de données, ce qui empêche les modèles d'être bien entrainés sur ce type d'attaque. Le pire modèle est *Naïve Bayes* pour tous les types d'attaques.

2. Collecte et préparation des données

2.1 La base de données NSL-KDD

La base de données choisie pour ce projet est NSL-KDD. C'est une base de données publique qui contient du trafic réseau enregistré par l'Université du Nouveau-Brunswick en 2009 [3]. Bien qu'elle soit assez vieille, elle a également été beaucoup documentée et c'est l'une des plus utilisées pour entrainer des IDS. Les données de trafic réseau contiennent des données normales, et des attaques de différents types. Inclut dans les données sont le type de protocole, le service utilisé, etc. Les différentes attaques et leur nombre dans la base de données sont décrites dans le tableau ci-dessous :

TYPE D'ATTAQUE	NOMBRE
DOS	45927
PROBE	11656
R2L	995
U2R	52
TOTAL	58630

Tableau 1 – Le nombre d'attaques de chaque type dans NSL-KDD

2.2 Préparation des données

Après avoir choisi la base de données, il faut préparer les données. Tout d'abord, il faut vérifier s'il y a des valeurs nulles. Il n'y a aucune valeur nulle dans NSL-KDD, mais s'il y en avait, il aurait fallu soit les remplacer, soit supprimer les lignes de données qui les contiennent.

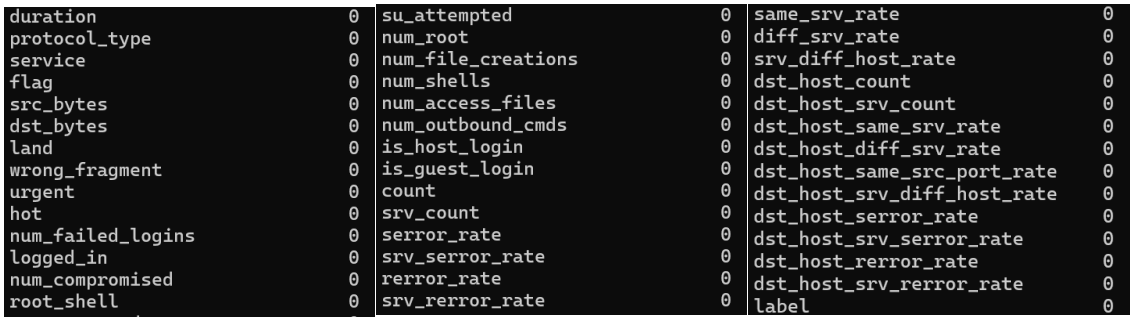

Figure 1 – Nombre de valeurs nulles dans chaque colonne de NSL-KDD

Ensuite, il faut vérifier s'il y a des lignes qui sont des duplicatas. Il y en a 9 dans la base de données, alors elles sont supprimées.

```
#enlever les duplicats
print(df.duplicated().sum())
df = df.drop_duplicates()
print(df.duplicated().sum())
```

Figure 2 – Code Python pour afficher le nombre de duplicatas et les enlever

Il faut ensuite convertir les colonnes en quelque chose que les modèles peuvent comprendre, ce qui est soit des nombres, soit des vrais /faux. Il y a donc deux approches possibles ; ce sera celle avec les valeurs booléennes qui sont choisies. Cette méthode s'appelle *get_dummies()*, et permet de créer une nouvelle colonne pour chaque valeur de chaque colonne. C'est cette méthode qui a été choisie pour le prototype. Ci-dessous se trouve un exemple de cette transformation subite avec un extrait de la base de données NSL-KDD.

duration	protocol_type	service
0	tcp	ftp_data
0	udp	other
0	tcp	private
0	tcp	http

Tableau 2 – Extrait de la base de données NSL-KDD.

duration	protocol_ type_tcp	protocol_ type_udp	service_ ftp_data	service_ other	service_ private	service_ http
True	True	False	True	False	False	False
True	False	True	False	True	False	False
True	True	False	False	False	True	False
True	True	False	False	False	False	True

Tableau 3 – Extrait de la base de données NSL-KDD avec la transformation get_dummies.

Pour les étiquettes d'attaques, elles sont transformées manuellement en 0 pour normal, et 1 pour attaque, avec la fonction *replace*.

Finalement, la dernière étape est de normaliser les données. Ceci est fait pour améliorer la performance et la précision des modèles qui vont être testés [8]. Ci-dessous se trouve l'extrait de la base NSL-KDD, normalisée.

duration	protocol_ type_tcp	protocol_ type_udp	service_ ftp_data	service_ other	service_ private	service_ http
-0.110249	0.476175	-0.367555	4.166943	0.189322	0.458130	0.686328
-0.110249	2.100067	2.720684	0.239984	5.281999	0.458130	0.686328
-0.110249	0.476175	-0.367555	0.239984	0.189322	2.182788	0.686328
-0.110249	0.476175	-0.367555	0.239984	0.189322	0.458130	1.457030

Tableau 4 – Extrait de la base de données NSL-KDD avec la transformation get_dummies normalisé.

3. Modèles d'IA utilisés

Il existe différents types de modèles d'apprentissage automatique. Dans le cas de ce projet, NSL-KDD contient les données qui sont étiquetées avec le nom de l'attaque ou « normal ». Également, ces étiquettes, et donc les classes à prédire, ne sont pas continues. Donc, le type de modèle d'apprentissage automatique utilisé sera « classification », qui consiste à prédire une classe à partir de données d'entrainement étiquetées.

Il existe différents modèles de classification, chacun avec leurs propres caractéristiques. Ces modèles sont décrits dans les sections ci-dessous.

3.1 Decision Tree

Le modèle *Decision Tree* est représenté par une structure arborescente où chaque nœud est des tests sur des caractéristiques, et chaque feuille, une classe prédite. Pour prendre une décision sur la classe, les données fournies partent de la racine de l'arbre, et chaque test oriente l'exploration vers le prochain nœud jusqu'à atteindre une feuille, qui dictera la classe à prédire pour ces données. Pour déterminer les caractéristiques à évaluer dans chaque nœud, la formule du gain d'information ou l'indice de Gini est utilisé pour maximiser la séparation des classes, et donc faire une séparation dans l'arbre [12].

3.2 Random Forest

Le modèle *Random Forest* combine plusieurs Décision Tree en un. Chaque arbre est donné un sous-groupe du jeu de données. À la place de calculer le gain d'information avec toutes les caractéristiques comme au modèle précédent, ce n'est qu'un sous-ensemble des caractéristiques choisi aléatoirement qui est utilisé pour les séparations dans l'arbre. Chaque arbre va arriver à une de ses feuilles, qui va être leurs décisions. Pour choisir la

décision finale du *Random Forest*, le vote majoritaire est utilisé, c'est-à-dire la décision qui a été la plus choisie parmi les arbres [13].

3.3 *K-Nearest Neighbors* (KNN)

Le modèle KNN se contente de sauvegarder les données d'entrainement, et donc tout le calcul se passe lorsqu'il prédit une classe. Lorsque cela se passe, il regarde les classes k données d'entrainement les plus proches selon des formules de distance. Avec les classes de ces voisins, il effectue un vote majoritaire pour prédire la classe des données. [14]

3.4 Logistic Regression

Le modèle de *Logistic Regression* fonctionne sur une base de probabilité. À partir des données d'entrainement, le modèle utilise une fonction sigmoïde pour déterminer les probabilités d'appartenance à une classe selon les caractéristiques. Lors de la prédiction, cette fonction sigmoïde est utilisée pour déterminer la classe. Généralement, c'est la classe qui a une probabilité d'au-dessus de 50% qui est choisie [15].

3.5 Naïve Bayes

Le modèle *Naïve Bayes* repose sur le principe que chaque caractéristique est indépendante les unes des autres, ce qui n'est pas généralement le cas dans la réalité. Pendant l'entrainement, le modèle calcule les probabilités de chacune des classes, et la probabilité conditionnelle pour chaque caractéristique étant donné une classe. Lors de la prédiction, le modèle calcule le produit de la probabilité de chacune des classes avec la probabilité des caractéristiques des données de test. La classe avec la plus haute probabilité est celle qui est choisie comme prédiction [16].

Le modèle utilisé dans ce projet est plus précisément *Gaussian Naïve Bayes*. C'est une variante qui suppose que les données sont continues et normalisées, ce qui a été fait à l'étape de pré-traitement des données.

3.6 *Support Vector Machines* (SVMs)

Le modèle SVMs permet de séparer linéairement à l'aide d'une ligne, d'un plan ou d'un hyperplan les données des différentes classes. Cette séparation est faite en maximisant la

marge de séparation entre les classes, c'est-à-dire la distance entre l'hyperplan et les points les plus proches de chaque classe, ce qui permet de faire de meilleures prédictions. Lorsqu'un plan est non-linéairement séparable, une fonction kernel est utilisée pour transformer les données dans un plus grand espace de dimension pour permettre la séparation linéaire [17].

4. Évaluation des modèles

Les modèles qui sont évalués sont ceux décris à la section précédente :

NUMÉRO	MODÈLE
1	*Decision Tree*
2	*Random Forest*
3	*K-Nearest Neighbors (KNN)*
4	*Logistic Regression*
5	*Naïve Bayes*
6	*Support Vector Machines (SVMs)*

Tableau 5 – Liste des modèles utilisés et leur numéro.

Le programme utilisé pour évaluer ces modèles peut se trouver dans ce répertoire : https://github.com/labm1/Projet-Synthese-IDS.

Les modèles seront d'abord évalués pour prédire n'importe quel type d'attaques de la base de données. Ensuite, ils seront évalués sur leur capacité à prédire les attaques spécifiques présentes dans NSL-KDD, qui sont *DoS*, *Probe*, *R2L* et *U2R*.

4.1 Métriques utilisées

Dans cette section, les métriques utilisées pour l'évaluation des modèles sont décrites.

Exactitude : Proportion des classifications qui sont correctes, qu'elles soient négatives ou positives [9].

$$Exactitude = \frac{Classifications\ correctes}{Total\ de\ classifications}$$

$$Exactitude = \frac{Vrais\ Positifs + Vrais\ Négatifs}{Vrais\ Positifs + Vrais\ Négatifs + Faux\ Positifs + Faux\ Négatifs}$$

Précision : Proportion de toutes les classifications positives qui sont réellement positives [9].

$$Précision = \frac{Classifications\ Correctes\ Positives}{Classifications\ Positives}$$

$$Précision = \frac{Vrais\ Positifs}{Vrais\ Positifs + Faux\ Positifs}$$

Rappel : Proportion des positifs qui ont été classés comme des positifs [9].

$$Rappel = \frac{Classifications\ Correctes\ Positives}{Tous\ les\ positifs}$$

$$Rappel = \frac{Vrais\ Positifs}{Vrais\ Positifs + Faux\ Négatifs}$$

F-mesure : Valeur entre 0 et 1. Plus le nombre de faux positifs et faux négatifs est élevé par rapport aux vrais positifs, plus la valeur est basse, et vice-versa [10].

$$F - Mesure = 2 * \frac{Précision * Rappel}{Précision + Rappel}$$

$$F - Mesure = \frac{2 * Vrais\ Positifs}{(2 * Vrais\ Positifs) + Faux\ Positifs + Faux\ Négatifs}$$

4.2 Comparaison générale des modèles

	DECISION TREE	RANDOM FOREST	KNN	LOGISTIC REGRESSION	NAÏVE BAYES	SVMs
Exactitude	0.9981	0.9985	0.9959	0.9737	0.8429	0.9918
Précision	0.9981	0.9993	0.9963	0.9784	0.9976	0.9900
Rappel	0.9978	0.9975	0.9949	0.9646	0.6629	0.9922

F-mesure	0.9979	0.9984	0.9956	0.9714	0.7965	0.9911
Faux-positifs	33	12	65	373	28	175
Faux-négatifs	39	44	89	621	5909	136
Temps d'entrainement (s)	1.16	5.97	0.02	1.90	0.11	320.52

Tableau 6 – Comparaison générale des métriques des différents modèles.

Ci-dessus se trouve les comparaisons des différentes métriques. Dans le jeu de données NSL-KDD, il y a 67343 données qui sont normales, et 58630 qui sont des attaques. Il est possible de voir dans les métriques que *Random Forest* a les meilleures métriques, qui est seulement quelques points plus haut que *Decision Tree*. Ce sont donc ces deux modèles qui sont les meilleurs pour prédire les attaques en général. Autre fait à noter, le modèle *Naïve Bayes* a un très haut nombre de faux négatifs (5909), ce qui explique la métrique très bas pour le rappel à 0.66. Il est clair que ce modèle est le pire à prédire les attaques. On peut aussi noter que le modèle *Support Vector Machines* a un très long temps d'entrainement (320 secondes) du modèle comparé aux autres.

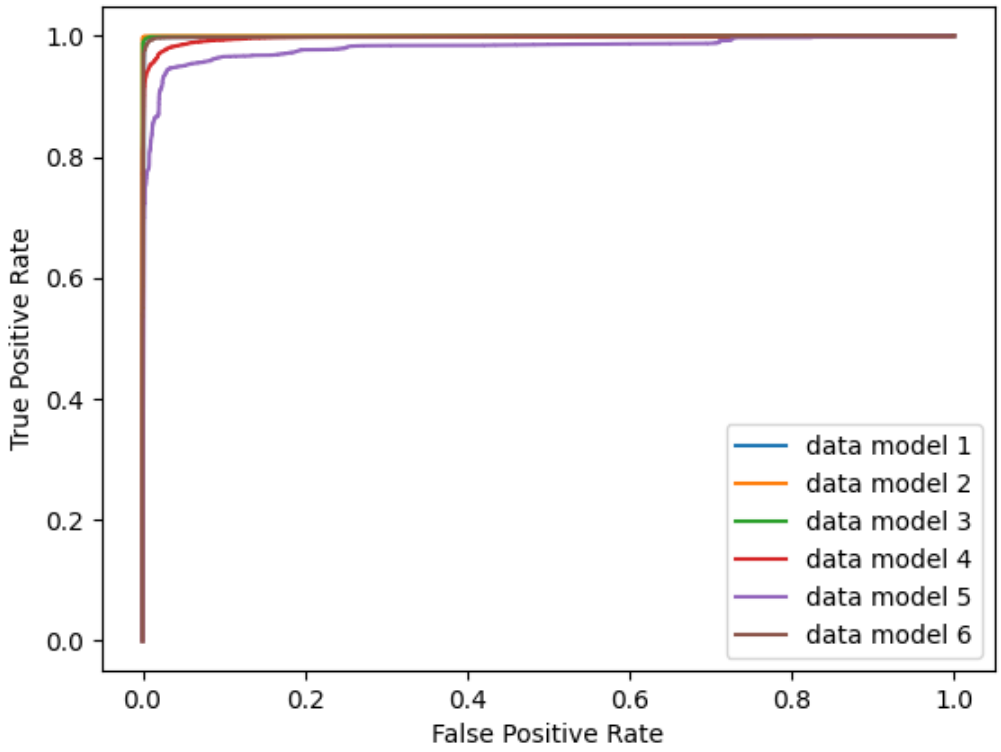

Figure 3 – Courbe ROC des modèles pour la prédiction de toutes les attaques.

Ci-dessus est le graphe de courbe ROC. Le plus rapidement que la ligne atteint le 1, le plus précis que le modèle est. Une ligne en diagonale signifierait que le modèle est aussi bon qu'une prédiction aléatoire [11]. On peut voir la ligne mauve, qui représente Naive Bayes, qui est en-dessous des autres. Les lignes bleue et orange sont toutes les deux les premières à atteindre le 1, ce qui montre la précision de prédiction aux modèles *Decision Tree* et *Random Forest*.

4.3 Comparaison des modèles pour l'attaque *DoS*

	DECISION TREE	RANDOM FOREST	KNN	LOGISTIC REGRESSION	NAÏVE BAYES	SVMs
Exactitude	0.9998	0.9999	0.9991	0.9979	0.9545	0.9982
Précision	0.9997	0.9999	0.9983	0.9966	0.9961	0.9961
Rappel	0.9997	0.9999	0.9993	0.9981	0.8913	0.9995
F-mesure	0.9997	0.9999	0.9988	0.9974	0.9408	0.9978
Faux-positifs	4	1	23	47	48	54
Faux-négatifs	4	2	9	26	1499	7
Temps d'entrainement (s)	0.85	4.63	0.02	1.13	0.12	107.41

Tableau 7 – Comparaison des métriques pour l'attaque DoS des différents modèles.

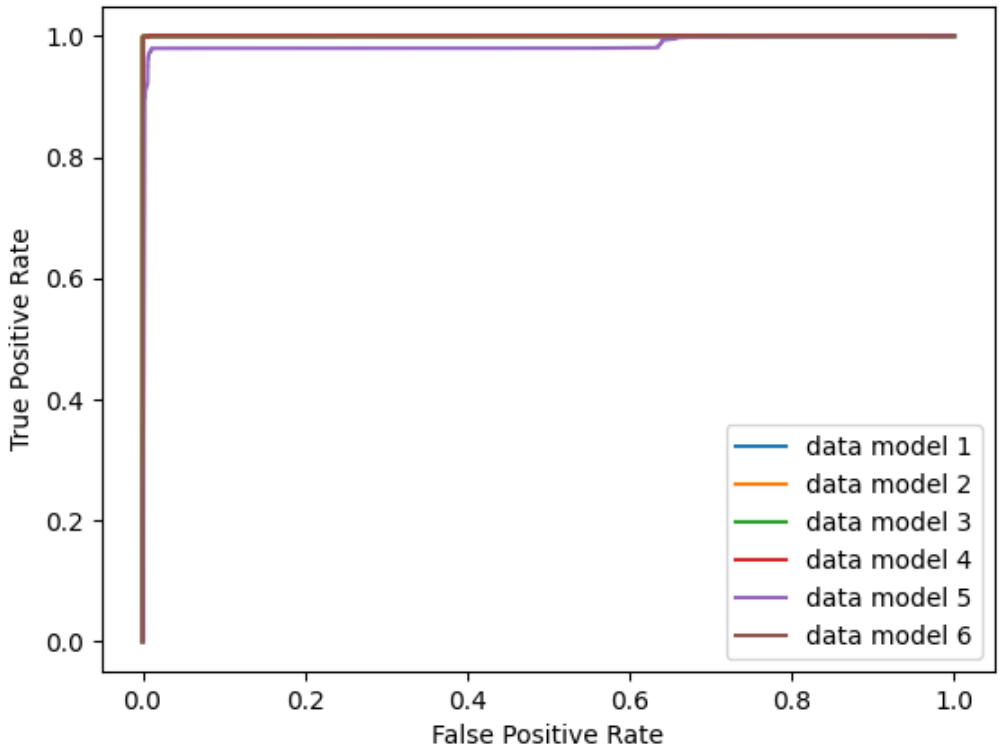

Figure 4 – Courbe ROC des modèles pour la prédiction de l'attaque DoS.

Pour la prédiction d'attaques *DoS*, c'est encore le *Random Forest* et le *Decision Tree* qui ont les meilleures métriques, ce qui n'est pas surprenant en considérant que les attaques *DoS* forment 78,3% des attaques de la base de données NSL-KDD, donc les résultats se doivent d'être similaires que ceux réunissant toutes les attaques. KNN est aussi très proche dans ses métriques aux deux autres modèles, mais il possède nettement plus de faux-positifs et faux-négatifs que ceux-ci. Cependant, tous les modèles sans exceptions semble être plus performants lors de la prédiction d'attaques *DoS* que toutes les attaques en général. Cela peut être dû au petit nombre de données disponibles pour les autres types d'attaques comparé au *DoS*. La courbe ROC montre une précision comparable entre cinq des modèles. Seul le modèle *Naïve Bayes* se trouve précisément en dessous des autres, tout autant dans le graphe que dans les métriques du tableau.

4.4 Comparaison des modèles pour l'attaque *Probe*

	DECISION TREE	RANDOM FOREST	KNN	LOGISTIC REGRESSION	NAÏVE BAYES	SVMs
Exactitude	0.9987	0.9990	0.9968	0.9921	0.8737	0.9939
Précision	0.9953	0.9980	0.9938	0.9776	0.9630	0.9765
Rappel	0.9961	0.9955	0.9852	0.9702	0.1737	0.9836
F-mesure	0.9957	0.9968	0.9895	0.9739	0.2943	0.9800
Faux-positifs	17	7	22	80	24	85
Faux-négatifs	14	16	53	107	2968	59
Temps d'entrainement (s)	0.44	3.21	0.01	0.51	0.08	71.91

Tableau 8 – Comparaison des métriques pour l'attaque Probe des différents modèles.

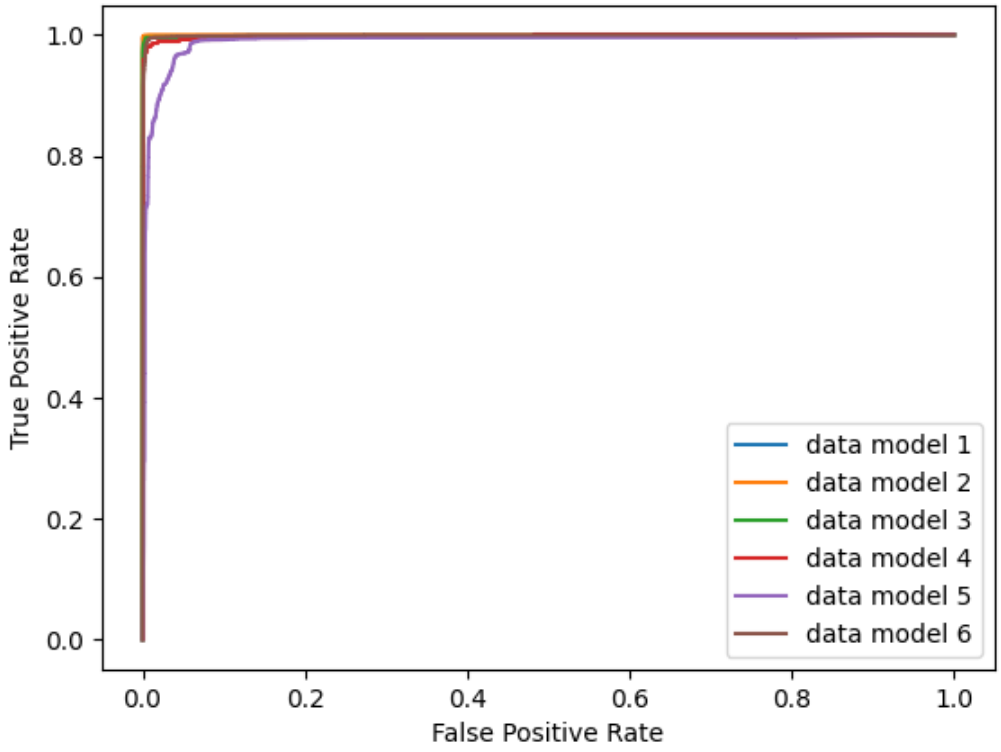

Figure 5 – Courbe ROC des modèles pour la prédiction de l'attaque Probe.

Les attaques *Probe* représentent 19,9% des attaques présentes dans la base de données NSL-KDD. *Random Forest* et *Decision Tree* sont encore les modèles avec les plus hautes métriques, avec KNN très proche derrière. *Naïve Bayes* est encore une fois le pire modèle, avec des mesures pour le Rappel et F-mesure de 0,2 et 0,3 respectivement, ce qui indique un grand nombre de faux négatifs par rapport aux vrais positifs. Avec la courbe ROC, on peut

voir que *Logistic Regression* et *Naïve Bayes* ont toute les deux des courbes en dessous de celle des autres. Les autres modèles ont des courbes ROC semblables, et sont donc très proche en termes de précision.

4.5 Comparaison des modèles pour l'attaque *R2L*

	DECISION TREE	RANDOM FOREST	KNN	LOGISTIC REGRESSION	NAÏVE BAYES	SVMs
Exactitude	0.9989	0.9991	0.9977	0.9946	0.5878	0.9956
Précision	0.9525	0.9874	0.9348	0.8484	0.0370	0.9100
Rappel	0.9817	0.9572	0.9205	0.8043	0.9939	0.8043
F-mesure	0.9669	0.9720	0.9276	0.8257	0.0714	0.8539
Faux-positifs	16	4	21	47	8448	26
Faux-négatifs	6	14	26	64	2	64
Temps d'entrainement (s)	0.36	2.35	0.01	0.76	0.07	42.17

Tableau 9 – Comparaison des métriques pour l'attaque R2L des différents modèles.

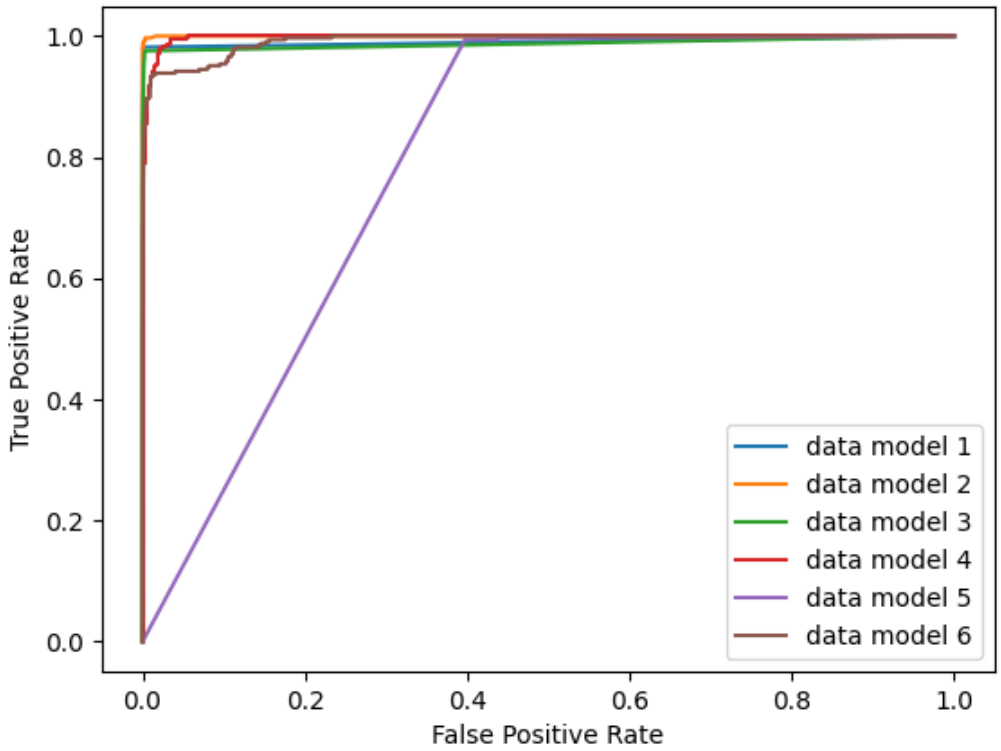

Figure 5 – Courbe ROC des modèles pour la prédiction de l'attaque R2L.

Les attaques *R2L* représentent 1,7% des attaques, avec 995 lignes de données dans NSL-KDD. Avec peu de données, il serait normal de voir une baisse dans les métriques, puisque

les modèles ont moins d'exemples d'attaque pour faire leur entrainement. En regardant les métriques, il est possible de voir qu'elles ont une légère baisse comparée à la prédiction des attaques générales pour tous les modèles. *Random Forest* ainsi que *Decision Tree* sont encore les meilleurs algorithmes pour prédire les attaques *R2L* selon les métriques. Cependant, sur le graphe de courbe ROC, la courbe de *Random Forest* est clairement plus élevée que les autres, ce qui la rendrait légèrement meilleure que *Decision Tree*. En regardant la courbe ROC de *Naïve Bayes*, il est possible de voir qu'il s'agit plus d'une ligne droite que d'une courbe, ce qui est sûrement dû au petit nombre d'attaques.

4.6 Comparaison des modèles pour l'attaque *U2R*

	DECISION TREE	RANDOM FOREST	KNN	LOGISTIC REGRESSION	NAÏVE BAYES	SVMs
Exactitude	0.9995	0.9997	0.9995	0.9996	0.9262	0.9995
Précision	0.6429	1.0000	0.7500	0.7778	0.0080	1.0000
Rappel	0.6429	0.5000	0.4286	0.5000	0.8571	0.2857
F-mesure	0.6429	0.6667	0.5455	0.6087	0.0158	0.4444
Faux-positifs	5	0	2	2	1490	0
Faux-négatifs	5	7	8	7	2	10
Temps d'entrainement (s)	0.26	1.79	0.01	0.23	0.06	21.05

Tableau 7 – Comparaison des métriques pour l'attaque U2R des différents modèles.

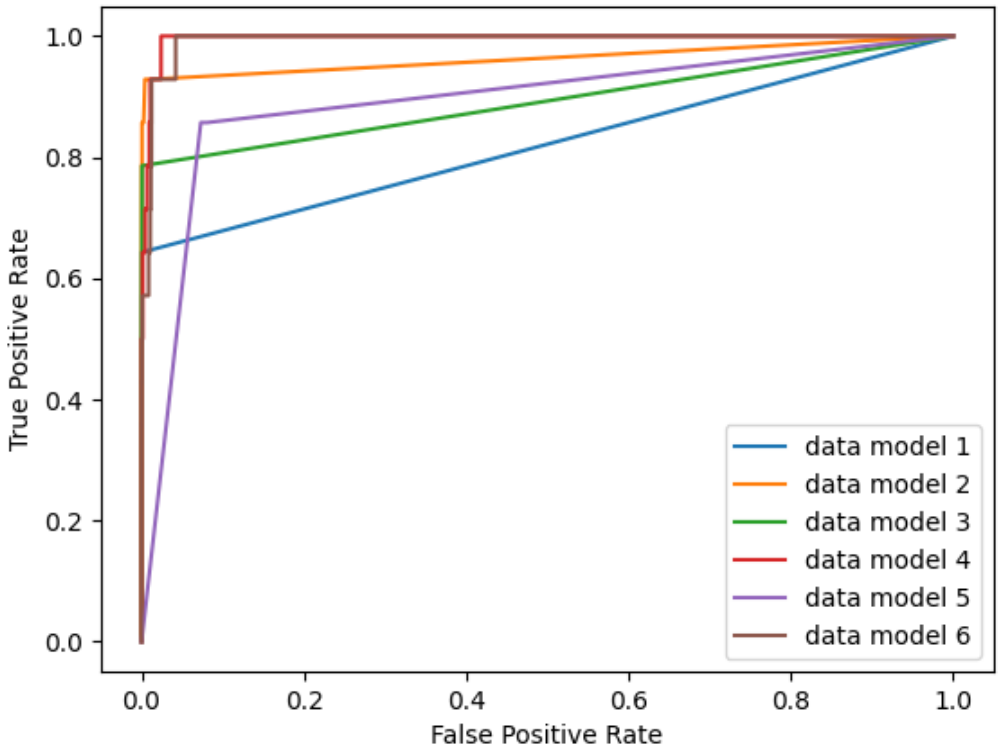

Figure 6 – Courbe ROC des modèles pour la prédiction de l'attaque U2R.

Le dernier type d'attaque est *U2R*, qui représente 0,09% des attaques, avec 52 lignes de données. C'est un très petit nombre, ce qui veut dire que les modèles seront très imprécis, et que leurs métriques peuvent varier grandement. À cause de cela, il n'est pas possible de dire exactement lequel des modèles est le meilleur pour prédire ces attaques. En regardant le tableau, il est possible de voir que tous les modèles ont des métriques très basses. Certains ont cependant une précision de 100% à cause du nombre de faux-positifs qui est à 0, ce qui ne devrait pas être possible dans un modèle qui est bien entrainé, qui nécessite plus de données. Il est cependant clair que *Naïve Bayes* est le pire pour prédire cette attaque, à cause de son bas taux de précision de 0,9%, ce qui représente un très grand nombre de faux-positifs par rapport aux vrais positifs. En regardant le graphe de courbes ROC, on peut voir les lignes droites, représentatives du petit nombre de données. On peut voir les courbes de *Logistic Regression* et de SVMs au-dessus des autres. Également, selon les courbes ROC, c'est le modèle *Decision Tree* qui est le moins précis.

5. Conclusion

En conclusion, le meilleur modèle d'apprentissage automatique pour prédire les attaques de la base de données NSL-KDD est généralement *Random Forest*, suivi de près par *Decision Tree*. Le pire modèle pour prédire les attaques est *Naïve Bayes*. Bien que la précision indiquée pour toute la base de données soit au-dessus de 99% pour *Random Forest*, il n'est pas envisageable de l'utiliser dans une réelle situation d'attaque sur le réseau, puisque NSL-KDD est une vieille base de données avec des attaques qui ne sont pas à jour, et elle est créée de toutes pièces, ce qui n'est pas forcément représentatif des attaques dans les vrais réseaux. Il est aussi difficile de prédire certains types d'attaques, comme U2R, avec les modèles à cause du manque de données sur ce type d'attaque dans la base de données NSL-KDD.

6. Bibliographie

[1] M, R. (2024, 23 janvier). How to Classify Data In Python using Scikit-learn. ActiveState. https://www.activestate.com/resources/quick-reads/how-to-classify-data-in-python/

[2] Bönninghausen, F. F. P. (s. d.). *All datasets*. COMIDDS. https://fkie-cad.github.io/COMIDDS/content/all_datasets/

[3] *UNB*. (s. d.). *ISCX NSL-KDD dataset 2009*. https://www.unb.ca/cic/datasets/nsl.html

[4] *Qu'est-ce que l'IA pour la cybersécurité ? | Sécurité Microsoft*. (n.d.). https://www.microsoft.com/fr-ca/security/business/security-101/what-is-ai-for-cybersecurity

[5] Eneskosar. (2024, 8 mai). *Intrusion Detection System [NSL-KDD]*. Kaggle. https://www.kaggle.com/code/eneskosar19/intrusion-detection-system-nsl-kdd/notebook

[6] *TP1.ipynb*. (s. d.). https://colab.research.google.com/drive/14STkRMRNxqyzKsVxXUG6plCsT9lxfyaN?usp=sharing

[7] Badgujar, H. (2023, 19 juillet). What is IDS (Intrusion Detection System) & How it works ? *Medium*. https://medium.com/@hrushikeshbadgujar/what-is-ids-intrusion-detection-system-how-it-works-732d81a13fb5

[8] Jaiswal, S. (2024, 14 novembre). *Qu'est-ce que la normalisation dans l'apprentissage automatique ? Un guide complet sur le redimensionnement des données*. https://www.datacamp.com/fr/tutorial/normalization-in-machine-learning

[9] *Classification: Accuracy, recall, precision, and related metrics*. (s.d.). Google for Developers. https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[10] *f1_score*. (s.d.). Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[11] *Machine Learning - AUC - ROC Curve*. (s.d.). W3Schools. https://www.w3schools.com/python/python_ml_auc_roc.asp

[12] *What is a decision tree?*. (s.d.). IBM. https://www.ibm.com/think/topics/decision-trees

[13] *What is random forest?*. (s.d.). IBM. https://www.ibm.com/think/topics/random-forest

[14] *What is the k-nearest neighbors (KNN) algorithm?*. (s.d.). IBM. https://www.ibm.com/think/topics/knn

[15] StatQuest with Josh Starmer. (2018, March 5). *StatQuest: Logistic Regression* [Video]. YouTube. https://www.youtube.com/watch?v=yIYKR4sgzI8

[16] *What are Naïve Bayes classifiers?*. (s.d.). IBM. https://www.ibm.com/think/topics/naive-bayes

[17] *What are support vector machines (SVMs)?*. (s.d.). IBM. https://www.ibm.com/think/topics/support-vector-machine