

1^{ère} année du cycle d'ingénieur
[CSCC]

Module : Recherche Opérationnelle

Rapport du Mini-Projet

P157

**L'Analyse Et Détection Des Menaces Dans Des
Environnements Cloud Avec Machine Learning**

GROUP ID : CS2C-07

Réalisé par :

- MAZZIR MOHAMED TAHA
- ELMORTAJI AYOUB
- RATBY MOSSAAB

Encadré par : Pr. Abdessamad KAMOUSS

Résumé

Le cloud Computing (CC) est une technologie transformatrice qui permet un accès à la demande aux ressources réseau et informatiques, telles que les services de stockage et de gestion de données, sur la base d'un modèle « **Pay as you go** ». Il améliore l'efficacité et l'évolutivité du système. Cependant, malgré ses nombreux avantages, les fournisseurs de cloud sont confrontés à d'importants défis en matière de sécurité, notamment lorsqu'il s'agit de protéger les environnements et les services cloud. Assurer la sécurité de ces environnements est essentiel, car les vulnérabilités peuvent exposer les données et les systèmes sensibles à des menaces potentielles. Pour répondre à ces préoccupations, diverses solutions, notamment des systèmes de surveillance avancés, ont été mises en œuvre pour améliorer la sécurité du cloud en analysant les ressources, les services et les activités réseau afin de détecter les comportements suspects. Dans ce contexte, les systèmes de gestion des informations et des événements de sécurité (**SIEM**) font partie intégrante de la surveillance du trafic réseau et de l'identification des anomalies pouvant indiquer des menaces potentielles. **Ce mini-projet** propose un modèle de détection des menaces basé sur le cloud qui exploite des algorithmes d'apprentissage automatique, en particulier **Random Forest** (RF) et **Decision Tree** (DT). Le classificateur RF est incorporé pour améliorer la précision (ACC) du modèle de détection. L'efficacité de l'approche proposée a été évaluée à l'aide de deux ensembles de données, atteignant une précision de **86,4 %** avec l'arbre de décision et de **100 %** avec le classificateur Random Forest, démontrant le potentiel de **Machine Learning** dans la détection et l'atténuation des menaces dans **les environnements cloud en temps réel**.

Abstract

Cloud computing (CC) is a transformative technology that allows on-demand access to network and computing resources, such as storage and data management services, based on a "**Pay as you go**" model. It enhances system efficiency and scalability. However, despite its many advantages, cloud providers face significant security challenges, particularly when it comes to safeguarding cloud environments and services. Ensuring the security of these environments is critical, as vulnerabilities can expose sensitive data and systems to potential threats. To address these concerns, various solutions, including advanced monitoring systems, have been implemented to improve cloud security by analyzing resources, services, and network activities to detect suspicious behaviors. In this context, **Security Information and Event Management (SIEM)** systems are integral for monitoring network traffic and identifying anomalies that may indicate potential threats. This mini-project proposes a cloud-based threat detection model that leverages machine learning algorithms, specifically **Random Forest (RF)** and **Decision Tree (DT)**. The RF classifier is incorporated to enhance the accuracy (ACC) of the detection model. The effectiveness of the proposed approach has been evaluated using two datasets, achieving an accuracy of **86.4%** with the Decision Tree and **100%** with the Random Forest classifier, demonstrating the potential of **Machine Learning** in detecting and mitigating threats in **real-time cloud environments**.

Sommaire

Résumé.....	2
Abstract	3
Liste des figures.....	5
Liste des tableaux.....	6
Introduction Générale	7
1.Introduction générale sur l'intérêt de la recherche opérationnelle en domaine d'ingénierie	7
2.Introduction sur l'analyse et détection des menaces dans des environnements cloud avec ML	7
3. Une description de l'organisation des chapitres de ce document	8
Chapitre I : Présentation du contexte du projet	9
I. Présentation de contexte	9
I.1 Processus de fonctionnement	9
I.2 Intérêt et enjeux économiques.....	13
I.3 Quelques chiffres clés	14
I.4 Quelques problématiques	14
I.5 Intérêt de RO dans ce domaine	14
II. Présentation de la problématique	15
Chapitre II : Modélisation	16
Chapitre III : Résolution.....	24
I. Outils utilisés.....	24
II. Résolution détaillée et résultats.....	25
III. Interprétations et critiques	40
Conclusion Générale.....	43
Bibliographie	44

Liste des figures

Figure 1 Les types du Cloud	9
Figure 2 les models du Cloud	11
Figure 3 Architecture de sécurité Cloud Computing	12
Figure 4 Fonctionnement de La solution SIEM dans Cloud	12
Figure 5 c'est quoi Cloud Computing	17
Figure 6 Faux potive / False Positive	17
Figure 7 Classification threshold	18
Figure 8 Confusion matrix	18
Figure 9 les ressource d'un serveur	20
Figure 10 Performance metrics of our model with NSL-KDD and bot-iot datasets	23
Figure 11 Decision Tree	25
Figure 12 Random Forest	26
Figure 13 Model Métriques	34
Figure 14 F1 score interprétation.....	34
Figure 15 Confusion Matrix	35
Figure 16 ROC curve.....	36

Liste des tableaux

Tableau 1: Ressources consommé par model	18
Tableau 2 : Présentation des ressources.....	19
Tableau 3 : Conclusion des résultats	19

Introduction Générale

1. Introduction générale sur l'intérêt de la recherche opérationnelle en domaine d'ingénierie :

Dans le monde actuelle les technologies deviennent de plus en plus complexes et nécessitent une meilleure approche scientifique, c'est pour cela qu'on utilise la recherche opérationnelle car c'est une discipline qui utilise des méthodes analytiques, mathématiques et algorithmiques pour analyser des problèmes complexes, optimiser les performances et améliorer la prise de décision.

Dans le domaine de l'ingénierie, la recherche opérationnelle a un rôle important dans différents domaines, comme par exemples dans l'optimisation des processus de la production, la gestion efficace des projets cela optimisent la planifications des tâches, et aussi la gestion efficace des ressources, la réduction des coûts tout en maintenant une bonne qualité, ou par exemple dans le domaine de la cybersécurité pour optimiser la détection des intrusions, et donc ainsi réagir efficacement contre les attaques.

2. Introduction sur l'analyse et détection des menaces dans des environnements cloud avec Machine Learning :

L'environnement cloud est une infrastructure informatique accessible à distance via internet, qui fournit une gamme de service et de ressource comme le stockage, les bases de données, les serveurs, les réseaux... Le cloud permet donc aux utilisateurs d'utiliser ces ressources à tout moment et n'importe où tant que le connexion internet est disponible, à partir de serveurs distants et non par des ordinateurs locaux, et cela permet une réduction des coûts et une meilleure flexibilité.

Dans le monde d'aujourd'hui connaît une très grande croissance dans l'utilisation des environnements cloud, donc la sécurité des environnements cloud contre les différentes menaces devient une priorité majeure. Les menaces incluent des attaques réseau, comme le déni de service (DDOS), les violations de données et les autres différentes intrusions...

Les méthodes de sécurité traditionnelles sont de plus en plus inefficaces pour détecter les différentes menaces, une solution puissante pour cela est d'utilisé le Machine Learning pour améliorer l'analyse et la détection des menaces dans un environnement cloud, avec le Machine Learning on peut analyser une très

grande quantité de données et identifier des patterns de comportement anormales et donc détecté une menace, et cela grâce à l'apprentissage à partir des incidents passés. Le ML améliore continuellement la détection des incidents ce qui permet de toujours avoir une réponse rapide et plus précise face aux différentes menaces.

3. Une description de l'organisation des chapitres de ce document :

On va voir dans ce document 3 Chapitre en tout qui sont :

Dans le Chapitre 1 on vas voir une présentation du contexte du mini-projet, tels que le processus de fonctionnement, Intérêt et enjeux économique, Quelques chiffres clés et Quelques problématique.

Puis on va voir le Chapitre 2 qui est porter sur la modélisation, puis finalement le Chapitre 3 (Résolution) on va voir les outils utilisés durant ce mini-projet, une résolution détaillée et les résultats obtenues, puis finalement une critique sur les résultats obtenues et les recommandations pour améliorer notre solution.

Chapitre I : Présentation du contexte du projet

Introduction

Les nouvelles technologies cloud ont révolutionner la gestion des données en donnant aux entreprises des solutions flexibles pour stocker, traiter et analyser leur information. Cependant cette révolution s'accompagne de nouveaux défis qui concerne la cybersécurité. Et parmi les meilleures solutions pour cela, on a le Machine Learning qui s'impose comme un outil prometteur pour l'analyse et la détection des menaces dans ces environnements complexes.



Figure 1 Cloud Security

I. Présentation de contexte

I.1 Processus de fonctionnement

Le Cloud Computing (CC), une technologie qui offre des ressources informatiques telles que le stockage et la sécurité sur Internet selon les besoins avec un modèle de tarification « **Pay as you go** ». Le Cloud Computing présente de grands avantages : une évolutivité souple qui permet d'adapter les ressources en fonction des besoins fluctuants ; des coûts réduits grâce à la suppression de l'infrastructure physique ; une accessibilité à distance encourageant la collaboration au travail ; et des mises à jour automatiques assurant des services constamment à jour. C'est pourquoi le cloud est si attrayant pour les entreprises cherchant flexibilité et efficacité. Ainsi le cloud Computing est composé de trois types principaux :

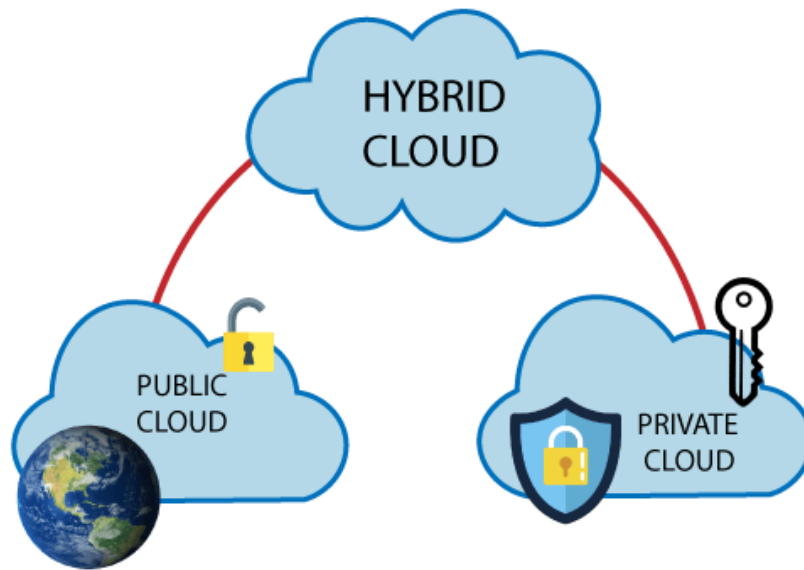


Figure 2 Les Types du Cloud

- **Cloud Public** : Ce modèle implique le partage des ressources entre divers clients grâce à un prestataire externe tel qu'AWS ou Microsoft Azure, il offre souplesse et évolutivité à moindre coût et convient aux applications moins critiques.
- **Cloud privé** : Ce modèle propose une infrastructure réservée uniquement à une entreprise pour offrir un contrôle accru et une sécurité renforcée tout en permettant une personnalisation des services selon les besoins spécifiques de l'organisation, ce service peut être géré en interne ou confié à un prestataire externe.
- **Cloud Hybride** : En associant les atouts du cloud privé et du cloud public, ce concept permet aux entreprises de stocker leurs données sensibles en toute sécurité sur un cloud privé tout en tirant parti d'un cloud public pour des applications plus souples et économiques.

D'autre part Le Cloud Computing fournit trois types de modèles qui sont :

IaaS (Infrastructure as a Service) : Ce modèle offre des ressources informatiques fondamentales comme des serveurs, du stockage et des réseaux à la demande. L'utilisateur a la possibilité de configurer et de gérer l'infrastructure sans avoir à se préoccuper de la gestion physique des serveurs. Ex : Microsoft Azure, IBM Cloud.

PaaS (Platform as a Service) : PaaS fournit une infrastructure pour le développement, le déploiement et la gestion d'applications sans se préoccuper de l'infrastructure sous-jacente. Il aide les développeurs à se focaliser sur le développement et la créativité. Ex : Google Cloud Platform, Microsoft Azure.

SaaS (Software as a Service) : Ce modèle offre des applications logicielles accessibles sur Internet, via un navigateur. L'utilisateur n'a pas à s'occuper de la gestion ou de l'entretien de l'infrastructure ou de la plateforme, puisque tout est pris en charge et supervisé par le fournisseur. Ex : Microsoft 365, HubSpot.

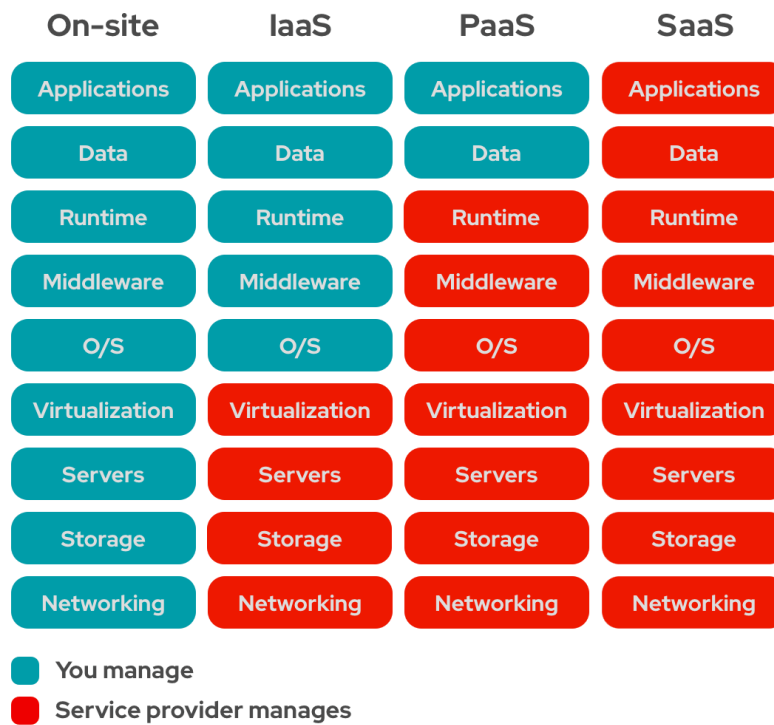


Figure 3 les models du Cloud

La sécurité du cloud est un ensemble de procédures et de technologies conçues pour faire face aux menaces externes et internes pesant sur la sécurité des entreprises. Les entreprises ont besoin de sécuriser le cloud à mesure qu'elles déploient leurs stratégies de transformation numérique et qu'elles intègrent, dans leur infrastructure, des outils et services basés sur le cloud. Parmi les éléments essentiels de cette sécurité, **l'IAM (Gestion des Identités et des Accès)**, les **IDS (Systèmes de Détection d'Intrusion)** et les **pares-feux** occupent une place centrale. **L'IAM** permet de réguler qui a accès aux ressources cloud, en s'assurant que seules les personnes ou entités habilitées possèdent les droits adéquats pour interagir avec les données sensibles, grâce à des systèmes d'authentification et de gestion des rôles. Les **IDS** surveillent l'intégralité du réseau et des systèmes cloud, identifiant les comportements inhabituels ou les tentatives d'intrusion, ce qui autorise une réaction rapide face aux menaces. Enfin, les **pares-feux** filtrent le trafic entrant et sortant, protégeant le réseau des accès non autorisés en bloquant les menaces potentielles, en particulier à travers des pare-feux applicatifs qui se concentrent sur les attaques visant les applications web hébergé dans un environnement cloud.

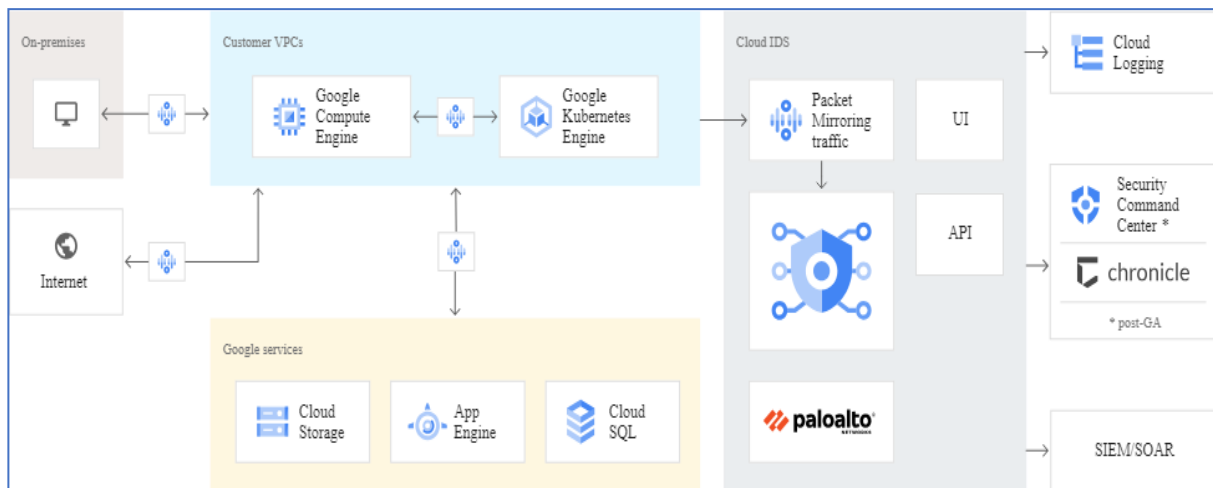


Figure 4 Architecture de sécurité Cloud Computing

On trouve aussi **SIEM** qui est une solution très puissante dans la sécurité cloud. La [gestion des informations et des événements liés à la sécurité \(SIEM\)](#) est une solution complète d'orchestration de la sécurité qui automatise la surveillance et la détection des menaces dans les environnements cloud, ainsi que la réponse à ces menaces. En utilisant des technologies basées sur **l'intelligence artificielle (IA)** pour mettre en corrélation les données des journaux sur plusieurs plateformes et actifs numériques. SIEM a fourni des **rapports de sécurité** et assure une **surveillance continue**, permettant une gestion proactive des risques et la conformité aux normes de sécurité dans le cloud.

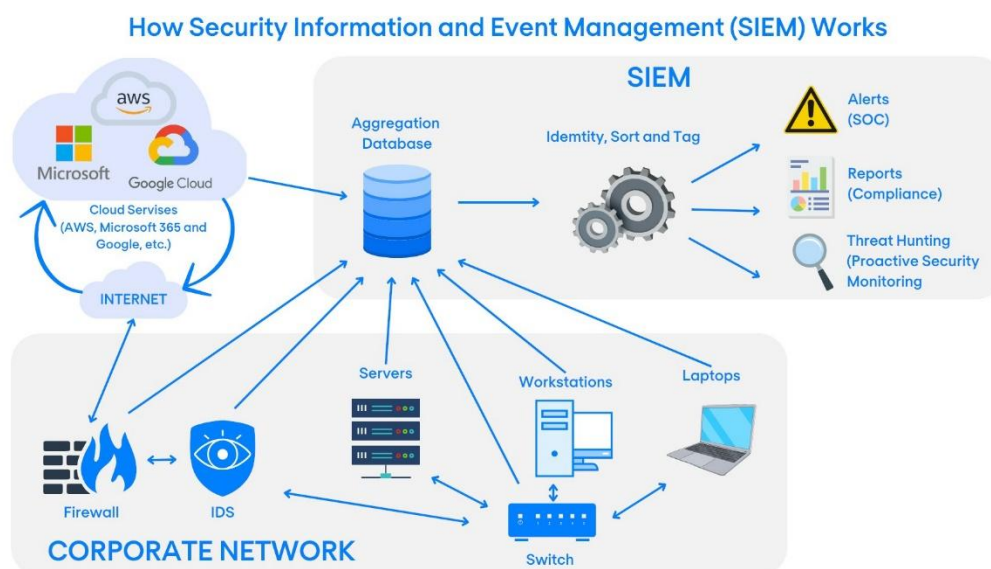


Figure 5 Fonctionnement de La solution SIEM dans Cloud

I.2 Intérêt et enjeux économiques

Les technologies cloud ont transformé la manière dont les entreprises gèrent leurs données. Cependant, cette transition majeure soulève des grands enjeux économiques liés à la sécurité.

- **Coût des cyberattaques** : Les cyberattaques coûtent en moyenne **4,35** millions de dollars par incident, ce qui met en danger la stabilité financière des entreprises, et donc une détection des menaces est très importante pour éviter de grande perte.
- **La croissance du marché** : Le marché de la sécurité cloud à l'échelle mondiale a été évalué en 2022 à **48,57** milliards de dollars, et les études estiment que cela va atteindre **116,25** milliards de dollars en 2028, ce qui montre que la sécurité cloud sera un domaine très sérieux dans les années avenir.
- **Optimisation** : Avec l'intégration du Machine Learning, cela permet l'automatisation des processus de sécurité et donc réduire fortement les interventions humaines, et donc améliorer grandement la rentabilité.

I.3 Quelques chiffres clés

- **Croissance du marché** : Le marché mondiale de la sécurité était évalué à **48,58** milliards de dollars en 2022, et il est prévue qu'il atteigne **116,25** milliards d'ici 2028, avec un taux de croissance composé de **15,5 %**.
- **Défis majeurs** : Les études disent que près de **47 %** des données stocker dans le cloud sont considérés comme sensibles. Et que **81%** des organisations ont signalé au moins une fois au cours des 12 derniers mois un incident de sécurité lié au cloud.
- **Chiffres liés à la sécurité** : Dans les entreprises actuelles au moins **10%** chiffrent **80%** de leurs données sensibles dans le cloud, et cela donc laisse une grande partie des informations exposées. Et aussi les chiffres disent que plus de la moitié des organisations ont une confiance faible ou moyenne dans leur capacité a sécurisé des données.

I.4 Quelques problématiques

- **Faux positifs et négatifs** : Les systèmes basés sur le Machine Learning peuvent générer des alertes non importants ou même manquer des menaces critiques.
- **Coûts associés à la sécurité** : Pour les entreprises développer, déployer et maintenir des solutions de sécurité avancées représente une charge financière importante.
- **Complexité des environnements cloud** : Par exemple la gestion des environnements multicloud (AWS, Azure, Google Cloud) rend difficile l'intégration des solutions de sécurité.

I.5 Intérêt de RO dans ce domaine

Dans le domaine de la détection des logs et des menaces dans un environnement Cloud une seule erreur peut avoir de graves conséquences.

C'est là qu'entre en jeu la recherche opérationnelle (RO), car l'optimisation du temps et des ressources est essentielle dans cette situation. La RO utilise des techniques avancées pour analyser et optimiser les performances des modèles de Machine Learning en prenant en compte plusieurs paramètres tels que le temps de réponse, la précision des prédictions et l'allocation efficace des ressources.

Un modèle optimisé, par exemple, peut réduire le nombre de faux positifs dans une analyse de logs tout en permettant une détection rapide des menaces, ce qui est essentiel pour une réponse efficace aux incidents.

II. Présentation de la problématique

Avec l'adoption massive du cloud, les entreprises dépendent de plus en plus de ces infrastructures pour gérer leur opérations critiques. Bien que cette transition soit bénéfique, elles exposent les systèmes à des menaces croissant. D'où l'importance d'utiliser les technologies comme le machine Learning pour automatiser la détection des menaces et analyser les comportements suspects. Et dans notre mini-projet on va essayer de construire un **système de détection des menaces basé sur les données en temps réel avec un faible taux de faux positifs**. Et c'est ce qu'on va voir dans les chapitres qui suivent.

Conclusion :

Pour les entreprises la sécurité cloud est devenue l'une des priorités les plus importante, mais grâce à l'intégration du Machine Learning, les solutions de sécurité gagne en efficacité, permettant une meilleure détection des menaces, avec une plus grande précision. Donc on peut dire que la combinaison du cloud et Machine Learning forme une étape clé dans la lutte contre les cyberattaques, malgré les défis qui existent comme les couts élevés, la complexité des environnements multi-cloud et les faux-positifs.

Chapitre II : Modélisation

Introduction :

Que ce que nous voulons optimiser ?

Notre objectif est d'utiliser les techniques de recherche opérationnelle afin d'optimiser l'emploi des modèles de Machine Learning dans des environnements cloud et pour cela, nous allons aborder ce problème sous deux angles :

1. Les ressources utilisées par le serveur ou on va déployer notre système de détection de Machine Learning.
2. Les faux positifs du modèle de Machine Learning dans la prédiction des intrusions dans les logs.

1) Les ressources utiliser par le serveur

Les serveurs cloud fonctionnent de la même manière que les serveurs traditionnels, car ils fournissent à la fois la puissance de traitement, les applications et le stockage. Toutefois, comme les serveurs cloud sont accessibles à distance, ils sont généralement plus stables et plus sécurisés que les serveurs traditionnels.

La principale différence entre un serveur cloud et un serveur traditionnel est qu'un serveur cloud peut être partagé entre de nombreux utilisateurs sur une plateforme accessible, souvent via un réseau tel que Internet. Un serveur traditionnel (dédié) est accessible uniquement par une société ou une entité donnée. Bien que les serveurs cloud exécutent les mêmes fonctions que les serveurs physiques, les serveurs cloud sont hébergés et remis sur un réseau plutôt que d'être configurés et gérés sur site.

Une autre différence entre les serveurs cloud et les serveurs physiques est que les serveurs cloud offrent une capacité de calcul illimitée, mais que les serveurs physiques sont limités à leur infrastructure ou capacité de calcul existante.

Donc on peut traiter un serveur cloud comme une ferme de serveurs que le client peut allouer comme il veut dépendant de son besoin.

Donc dans cette partie nous allons prendre en considération que le nombre de ressource qui ne sont pas limité mais seulement le cout augmente **Pay-As-You-Go (Païement à l'utilisation)**

Les ressource Essentials que on va traiter sont :

- CPU
- RAM

- Bandwidth
- Taux de Faux-positif

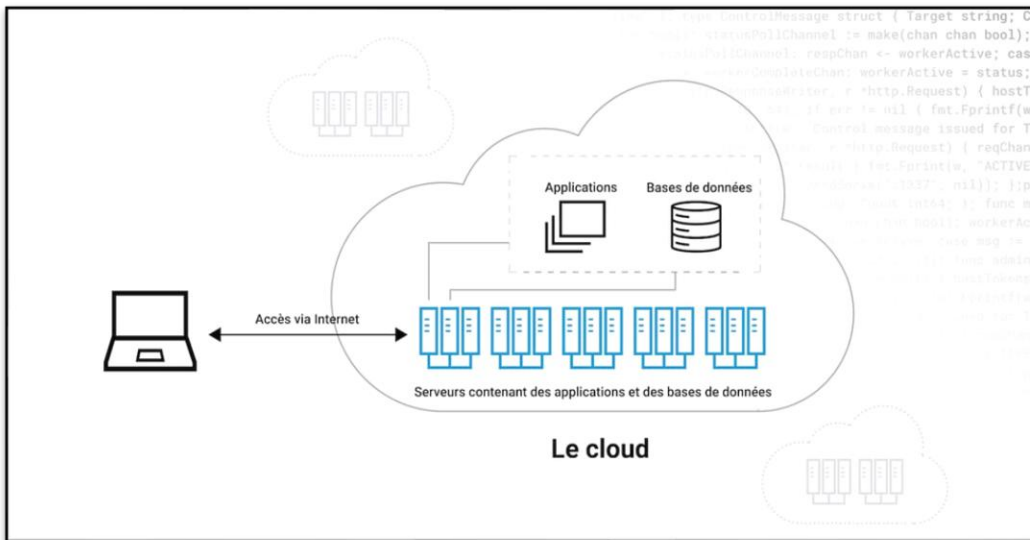


Figure 6 c'est quoi Cloud Computing

2) Les faux positifs

Le taux de faux positifs (TPF) est une mesure de la proportion de cas positifs qui ont été incorrectement identifiés ou classés comme positifs lors d'un test. Ou, en termes simples, de fausses alertes. Le taux de faux positifs peut être utilisé pour mesurer les problèmes de contexte binaire.

Il est calculé comme :

$$FPR = FP / (FP + TN)$$

- FP (False Positive) – The *positive* instances *incorrectly* classified.
- TN (True Negative) – The *negative* instances *correctly* classified.

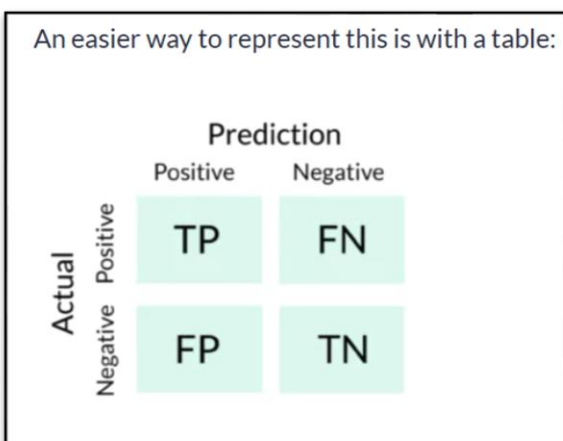


Figure 7 Faux potive / False Positive

On pose que :

- **Anomalie = positif**
- **Normal = négatif**
 - Si on avait prédit l'incident est une anomalie tant qu'il est Safe donc c'est un **Faux-Positif**

Un modèle parfait ne comporterait aucun faux positif et donc un taux de faux positifs de 0, c'est-à-dire un taux de fausses alarmes de 0 %.

Pour effectuer cette conversion, vous choisissez une probabilité de seuil, appelée seuil de classification. Les exemples dont la probabilité est supérieure à la valeur du seuil sont ensuite attribués à la classe positive, la classe que vous testez. Les exemples dont la probabilité est inférieure sont attribués à la classe négative, la classe alternative.

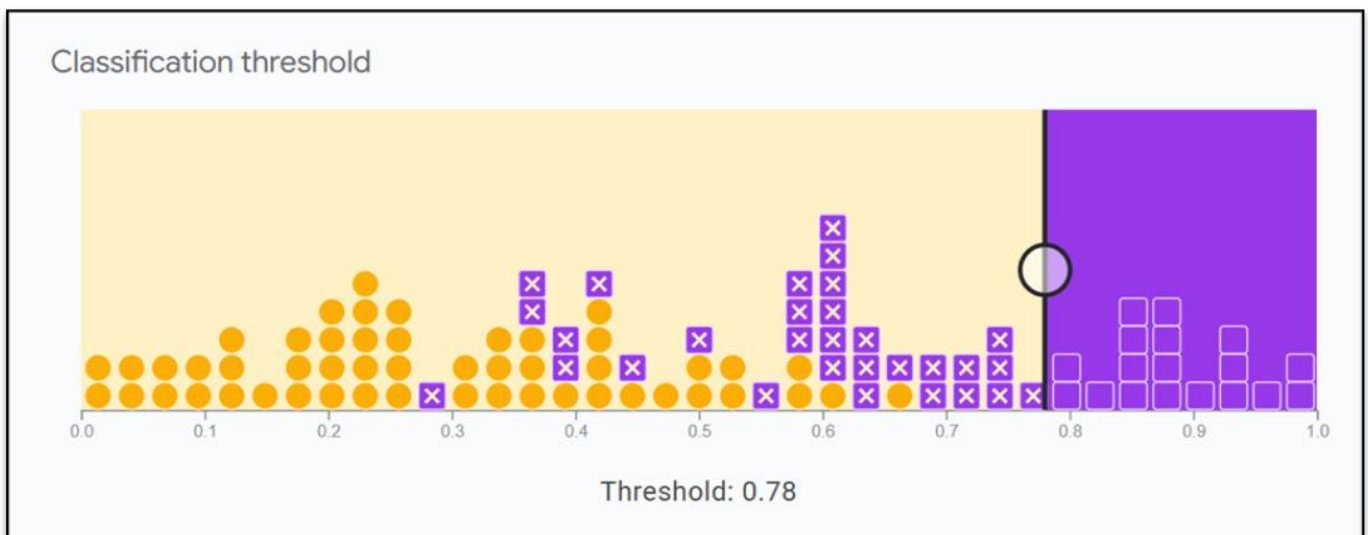


Figure 8 Classification threshold





Confusion matrix		
	Actually positive	Actually negative
Predicted positive	 TP=18	 FP=0
Predicted negative	 FN=30	 TN=51

Figure 9 Confusion matrix

I. Choix de la modélisation appropriée

Comme notre problème est consacré sur minimiser les ressources utiliser pour notre modèle de Machine Learning dans un environnement cloud donc minimiser le cout. Nous allons utiliser l'approche d'un programme linéaire

Pour cela nous allons proposer deux modèles chacun avec un :

- Nombre de ressources utiliser
- Temps d'exécution
- Un taux de faux positifs

Et nous allons essayer de trouver la quantité des paquets donner à chaque model afin d'optimiser le cout et le taux de faux-positifs.

Par exemple si on pose les ressources consommées par un *nombre unitaire de paquets

Model	RAM (GB)	CPU (%)	FPR (%)	*Bandwidth needs (MB/s)	Temps d'exécution (s)
Model 1	1.0	60	0.2	10	2
Model 2	1.4	40	0.1	15	3

Tableau 1 Ressource consommé par model

* Bandwidth needs : mesure le bandwidth que notre model a besoin pour donner une fonctionnalite desirable

* nombre unitaire de paquets : ce nombre de paquets est donnée au model dans un temp bien defini donc il faut bien respecter les constraints d'usage de RAM / CPU / ect ...

II. Variables de décisions

Les variables de décision qu'on propose sont :

- $Nbr_{packet1}$: quantité de paquet unitaire traiter par model 1
- $Nbr_{packet2}$: quantité de paquet unitaire traiter par model 2

Ce choix de variables de décisions nous permet de contrôler directement la distribution des paquets entre les modèles tout en optimisant les ressources du serveur.

III. Contraintes

On suppose que notre serveur a des capacités bien défini et on suppose que dans ce cas on ne peut pas les changer

Ram	$Ram_{serveur}$
Cpu	$Cpu_{serveur}$
Bandwith	$bandwidth_{serveur}$

Tableau 2 Présentation de ressource

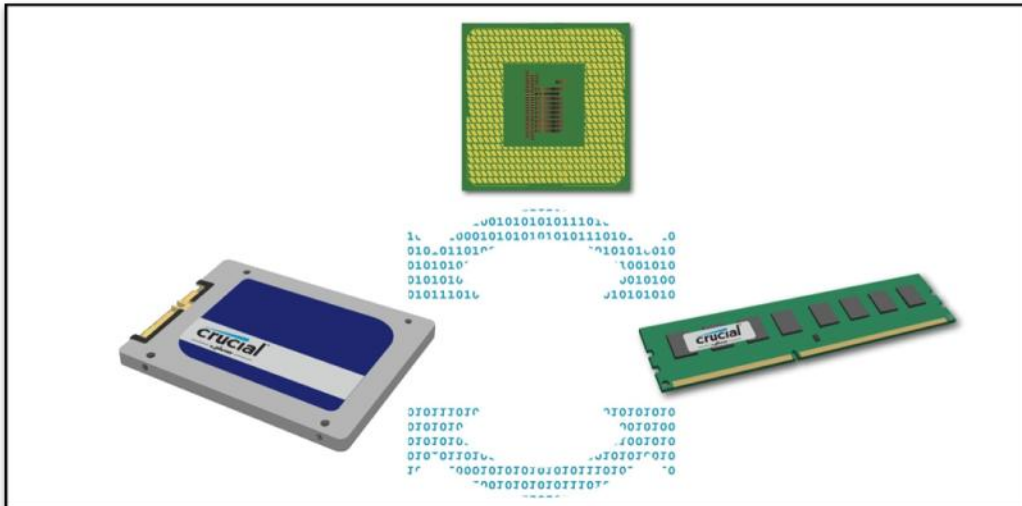


Figure 10 les ressources d'un serveur

Donc les constraints sont :

1. Limite de RAM

$$\bullet \quad Nbr_{packet1} * Ram_{M1} + Nbr_{packet2} * Ram_{M2} \leq Ram_{serveur}$$

2. Limite de CPU

$$\bullet \quad Nbr_{packet1} * Cpu_{M1} + Nbr_{packet2} * Cpu_{M2} \leq Cpu_{serveur}$$

3. Limite de Bandwidth

- $Nbr_{packet1} * bandwidth_{M1} + Nbr_{packet2} * bandwidth_{M2} \leq bandwidth_{serveur}$

4. On pose que la contrainte du taux de faux-positive qu'on désire ne doit pas dépasser une FPR_{max} :

- $Nbr_{packet1} * FPR_{M1} + Nbr_{packet2} * FPR_{M2} \leq FPR_{max}$

5. Contrainte de non-négativité:

- $Nbr_{packet1}, Nbr_{packet2} \geq 0$

IV. Fonction objectif

L'objectif est toujours de minimiser le cout qu'on peut modéliser par la somme des ressources utilisées.

On modélise ces couts par :

$$[min]w = \sum \text{ressource utilisée}$$

$$w = Nbr_{packet1} * (RamM1 + CpuM1 + bandwidthM1) \\ + Nbr_{packet2} * (RamM2 + CpuM2 + bandwidthM2)$$

Mais d'une façon plus convaincante de traiter ce problème c'est de maximiser le nombre de paquet traiter donc on considère la fonction objective :

$$[Max]z = Nbr_{packet1} + Nbr_{packet2}$$

V. Modélisation

Le modele complet du programme linaire est :

$$[max]z = Nbr_{packet1} + Nbr_{packet2}$$

- $Nbr_{packet1} * Ram_{M1} + Nbr_{packet2} * Ram_{M2} \leq Ram_{serveur}$
- $Nbr_{packet1} * Cpu_{M1} + Nbr_{packet2} * Cpu_{M2} \leq Cpu_{serveur}$
- $Nbr_{packet1} * bandwidth_{M1} + Nbr_{packet2} * bandwidth_{M2} \leq bandwidth_{serveur}$
- $Nbr_{packet1} * FPR_{M1} + Nbr_{packet2} * FPR_{M2} \leq FPR_{max}$
- $Nbr_{packet1}, Nbr_{packet2} \geq 0$

Conclusion

Cette partie illustre comment la recherche opérationnelle peut être utilisée pour optimiser l'utilisation des ressources dans un environnement cloud tout en réduisant le taux de faux positifs des modèles de Machine Learning. Grâce à une modélisation basée sur la programmation linéaire, nous avons proposé une solution efficace qui équilibre le traitement maximal de paquets et la minimisation des coûts.

Cependant, l'optimisation peut être améliorée en prenant en considération plusieurs autres facteurs, notamment :

1. Les types de modèles choisis
2. Les DataSets utilisées à trainer le modelé
3. Utiliser des techniques pour améliorer le taux de faux positif d'un modèle Machine Learning

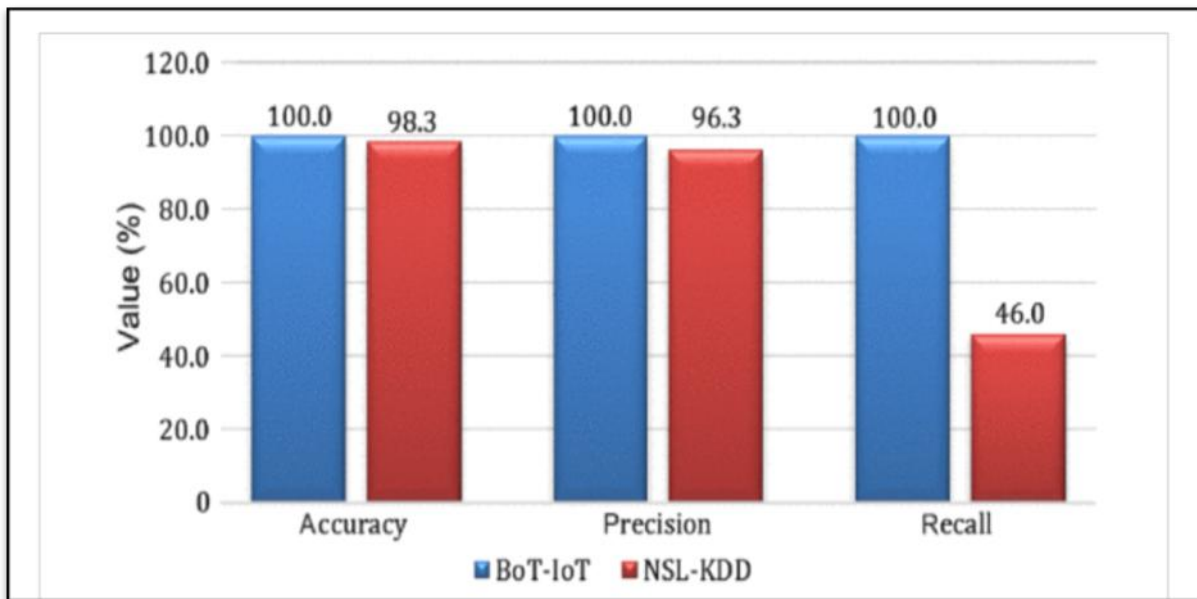


Figure 11 Performance metrics of our model with NSL-KDD and bot-iot datasets

Chapitre III : Résolution

Introduction :

Durant cette partie on vas montrer comment on a créer les deux modèles de Machine Learning : Décision Tree et Random Forest, ces modèles vont à la fin nous donner l'Accuracy après leur entraînement sur un ensemble de données spécifiques, et on vas voir un exemple à la fin qui montre comment on vas utiliser la RO pratiquement pour mieux gérer la répartition d'une dataset sur les deux modèles, afin d'avoir une gestion plus efficace et une meilleure performance.

I. Outils utilisés :

Le premier outils utilisé c'est le langage de programmation Python:



On a aussi utilisé plusieurs bibliothèques afin de faciliter le traitement des données, créer nos deux modèles de Machine Learning et l'analyse des résultats :

```
Importing Libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

[1] Python

Comme on le voit ici on a importé plusieurs bibliothèques tels que :

1- **NumPy** : Elle est utilisée pour des calculs numériques avec des tableaux multidimensionnels et des fonctions mathématiques performantes.

2- **Pandas** : La bibliothèque Pandas est utilisée pour la manipulation et l'analyse des données tabulaires, et elle est essentiel pour importer, nettoyer et aussi la transformation et l'analyse des datasets.

3- **Matplotlib** : Cette bibliothèque est utilisée pour la visualisation des données, c'est à dire la génération des graphiques.

4- **Seaborn** : Elle est basée sur Matplotlib, elle rend la visualisation des données plus facile, et elle simplifie la création des graphiques complexes.

5- **Scikit-learn(sklearn)** : C'est une bibliothèque importante pour le Machine Learning, car elle permet de fournir plusieurs outils nécessaires pour la préparation des données, l'entraînement des modèles et leur évaluation.

Maintenant on va voir les modèles Décision Tree et Random Forest :

1- Le modèle Decision Tree :

Un arbre de décision est un algorithme d'apprentissage supervisé utilisé en ML, cet algorithme permet de prédire en fonction des données d'entrée des résultats. Il a une forme d'une structure arborescente où chaque noeud est représentée par une condition ou un test basé sur une caractéristique des données, et chaque branche représente une issue possible du test, puis finalement chaque feuille donne une prédiction finale.

Les Decisions Trees sont des modèles utilisés pour les problèmes de classification, voici un petit exemple qui montre le principe du Décision Tree :

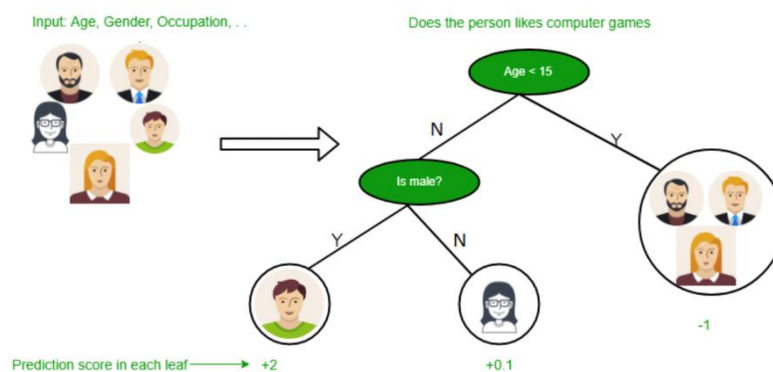


Figure 12 Decision Tree

2- Le modèle Random Forest :

Le Random Forest est aussi un modèle de machine learning puissante qui repose sur l'utilisation de plusieurs Decision Tree, et chaque arbre est créé à partir d'un échantillon aléatoire de données, et il sélectionne aussi à chaque division des nœuds un sous ensemble aléatoire des caractéristiques disponibles. Ceci augmente la diversité entre les arbres, ce qui améliore les performances globales du modèle.

Le Random Forest est un modèle utilisé pour les problèmes de classification et aussi la régression. Lors de la prédiction dans le cas de la classification, chaque Decision Tree prédit une classe, et la classe finale est déterminée par le vote majoritaire des arbres. Et dans le cas de la régression il calcule la moyenne des prédictions des arbres.

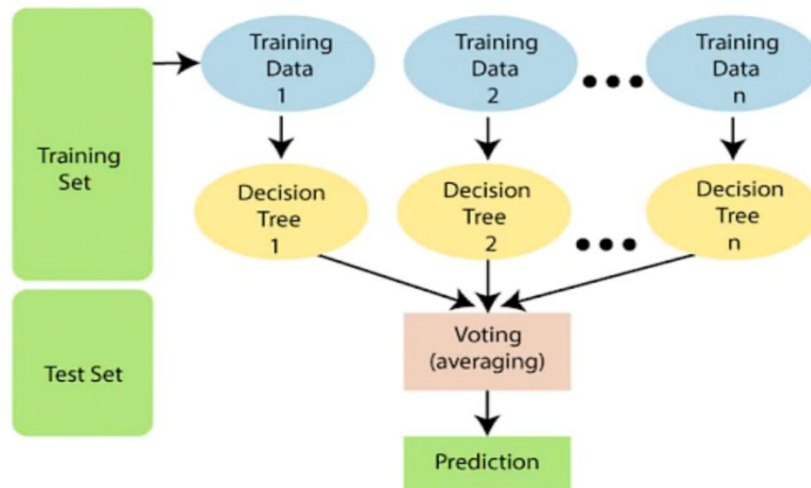


Figure 13 Random Forest

II. Résolution détaillée et résultats :

Tout d'abord commençons par voir la DATASET utilisée dans les deux modèles Machine Learning (Decision Tree et Random Forest) qu'on vas créer.

Il faut d'abord savoir que cette dataset a été prise du site **Kaggle**, et elle divisée en deux une partie d'entrainement et une partie test, voici un aperçu de cette dataset :

```

Trained_Data = pd.read_csv("KDDTrain+.txt", sep = ",", encoding = 'utf-8')
Tested_Data = pd.read_csv("KDDTest+.txt", sep = ",", encoding = 'utf-8')

```

Python

```

Trained_Data.head()

```

Python

	0	tcp	ftp_data	SF	491	0.1	0.2	0.3	0.4	0.5	...	0.17	0.03	0.17.1	0.00.6	0.00.7	0.00.8	0.05	0.00.9	normal	20
0	0	udp	other	SF	146	0	0	0	0	0	...	0.00	0.60	0.88	0.00	0.00	0.00	0.0	0.00	normal	15
1	0	tcp	private	S0	0	0	0	0	0	0	...	0.10	0.05	0.00	0.00	1.00	1.00	0.0	0.00	neptune	19
2	0	tcp	http	SF	232	8153	0	0	0	0	...	1.00	0.00	0.03	0.04	0.03	0.01	0.0	0.01	normal	21
3	0	tcp	http	SF	199	420	0	0	0	0	...	1.00	0.00	0.00	0.00	0.00	0.00	0.0	0.00	normal	21
4	0	tcp	private	REJ	0	0	0	0	0	0	...	0.07	0.07	0.00	0.00	0.00	0.00	1.0	1.00	neptune	21

5 rows x 43 columns

```

Tested_Data.head()

```

Python

	0	tcp	private	REJ	0.1	0.2	0.3	0.4	0.5	0.6	...	0.04.1	0.06.1	0.00.3	0.00.4	0.00.5	0.00.6	1.00.2	1.00.3	neptune	21
0	0	tcp	private	REJ	0	0	0	0	0	0	...	0.00	0.06	0.00	0.00	0.00	0.0	1.00	1.00	neptune	21
1	2	tcp	ftp_data	SF	12983	0	0	0	0	0	...	0.61	0.04	0.61	0.02	0.00	0.0	0.00	0.00	normal	21
2	0	icmp	eco_i	SF	20	0	0	0	0	0	...	1.00	0.00	1.00	0.28	0.00	0.0	0.00	0.00	saint	15
3	1	tcp	telnet	RSTO	0	15	0	0	0	0	...	0.31	0.17	0.03	0.02	0.00	0.0	0.83	0.71	mscan	11
4	0	tcp	http	SF	267	14515	0	0	0	0	...	1.00	0.00	0.01	0.03	0.01	0.0	0.00	0.00	normal	21

5 rows x 43 columns

Comme on le remarque ici la dataset n'a pas les noms des colonnes de base mais d'après la documentation, les colonnes sont :

```
# Modify Columns:

Columns = (['duration', 'protocol_type', 'service', 'flag', 'src_bytes', 'dst_bytes', 'land', 'wrong_fragment', 'urgent', 'hot',
            'num_failed_logins', 'logged_in', 'num_compromised', 'root_shell', 'su_attempted', 'num_root', 'num_file_creations',
            'num_shells', 'num_access_files', 'num_outbound_cmds', 'is_host_login', 'is_guest_login', 'count', 'srv_count',
            'error_rate', 'srv_error_rate', 'error_rate', 'srv_error_rate', 'same_srv_rate', 'diff_srv_rate', 'srv_diff_host_rate',
            'dst_host_count', 'dst_host_srv_count', 'dst_host_same_srv_rate', 'dst_host_diff_srv_rate', 'dst_host_same_src_port_rate',
            'dst_host_srv_diff_host_rate', 'dst_host_error_rate', 'dst_host_srv_error_rate', 'dst_host_error_rate',
            'dst_host_srv_error_rate', 'attack', 'level'])

Trained_Data.columns = Columns
Tested_Data.columns = Columns
```

Donc les variables de notre modèle sont les suivants (on a 43 colonnes en tout) :

Devant chaque variable il y'a son champs de valeurs, et une explication pour les colonnes les plus importante

Duration : real

Explication : Cette donnée indique combien de temps une connexion réseau est restée active. Elle peut être utile pour distinguer les connexions normales des attaques potentiellement éphémères, comme celles utilisées dans des tentatives d'intrusion ou des scans de port.

protocol_type : { tcp , udp , icmp' }

Explication : Le type de protocole réseau employé pour la communication (par exemple TCP, UDP ou ICMP). Chaque protocole a ses propres caractéristiques de fonctionnement. Par exemple, TCP est fiable, tandis qu'UDP est plus rapide mais moins sécurisé, ce qui peut influencer le type d'attaque ou de trafic observé.

service: { aol , auth , bgp , courrier , csnet_ns , ctf , daytime , discard , domain , domain_u , echo , eco_i , ecr_i , efs , exec , finger , ftp , ftp_data , gopher , harvest , hostnames , http , http_2784 , http_443 , http_8001 , imap4 , IRC , iso_tsap , klogin , kshell , ldap , link , login , mtp , name , netbios_dgm , netbios_ns , netbios_ssn , netstat , nnspp , nntpp , ntp_u , other , pm_dump , pop_2 , pop_3 , printer , private , red_i , remote_job , rje , shell , smtp , sql_net , ssh , sunrpc , supdup , systat , telnet , tftp_u , tim_i , time , urh_i , urp_i , uucp , uucp_path , vmnet , whois , X11 , Z39_50 }

Explication : Il s'agit du service ou de l'application auquel une connexion réseau est dédiée (par exemple, HTTP pour la navigation web ou FTP pour le transfert de fichiers). Identifier le service utilisé peut révéler des vulnérabilités spécifiques à certains services, qui sont souvent la cible d'attaques.

flag: { OTH , REJ , RSTO , RSTOS0 , RSTR , S0 , S1 , S2 , S3 , SF , SH }

Explication : Cette colonne fournit des informations sur l'état de la connexion, indiquant si elle a été réussie, rejetée ou si un problème s'est produit. Certains états, comme S0 (absence de réponse) ou SF (connexion réussie), peuvent donner des indices précieux sur la nature de la tentative de connexion ou l'attaque éventuelle.

src_bytes: real

dst_bytes: real

Explication : Cela représente la quantité d'octets envoyés vers la destination.

land: { 0 , 1 }
 wrong_fragment: real
 urgent: real
 hot: real
 num_failed_logins: real
 logged_in: {'0', '1'}
 num_compromised: real
 root_shell: real
 su_attempted: real
 num_root: real
 num_file_creations: real
 num_shells: real
 num_access_files: real
 num_outbound_cmds: real
 is_host_login: { 0 , 1 }
 is_guest_login: { 0 , 1 }
 count: real

Explication : Ce chiffre montre le nombre total de connexions à une même hôte. Une valeur élevée pourrait être un signe de trafic anormal, ce qui est souvent le cas lors d'attaques par déni de service (DoS) ou d'activités comme des scans de vulnérabilité.

srv_count: real

Explication : Indique combien de connexions ont été établies avec un service particulier. Si ce nombre est anormalement élevé pour un service donné, cela peut être le signe d'une exploitation abusive du service ou d'une tentative de brute-force.

serror_rate: real
 srv_serror_rate: real
 rerror_rate: real
 srv_rerror_rate: real
 same_srv_rate: real
 diff_srv_rate: real
 srv_diff_host_rate: real
 dst_host_count: real
 dst_host_srv_count: real
 dst_host_same_srv_rate: real
 dst_host_diff_srv_rate: real
 dst_host_same_src_port_rate: real
 dst_host_srv_diff_host_rate: real
 dst_host_serror_rate: real
 dst_host_srv_serror_rate: real
 dst_host_rerror_rate: real
 dst_host_srv_rerror_rate: real
 attack
 level

Voici a quoi ressemble notre dataset (on a pris ici Trained_Data) après ce traitement des colonnes :

Exploring Data

Trained_Data

[6]

Python

...	ost_same_srv_rate	dst_host_diff_srv_rate	dst_host_same_src_port_rate	dst_host_srv_diff_host_rate	dst_host_serror_rate	dst_host_srv_serror_rate	dst_host_error_rate	dst_host_srv_error_rate	attack	level
	0.00	0.60	0.88	0.00	0.00	0.00	0.00	0.00	normal	15
	0.10	0.05	0.00	0.00	1.00	1.00	0.00	0.00	neptune	19
	1.00	0.00	0.03	0.04	0.03	0.01	0.00	0.01	normal	21
	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	normal	21
	0.07	0.07	0.00	0.00	0.00	0.00	1.00	1.00	neptune	21

	0.10	0.06	0.00	0.00	1.00	1.00	0.00	0.00	neptune	20
	0.96	0.01	0.01	0.00	0.00	0.00	0.00	0.00	normal	21
	0.12	0.06	0.00	0.00	0.72	0.00	0.01	0.00	normal	18
	0.03	0.05	0.00	0.00	1.00	1.00	0.00	0.00	neptune	20
	0.30	0.03	0.30	0.00	0.00	0.00	0.00	0.00	normal	21

Maintenant on va entamer la partie qui concerne data Cleaning :

1- On vas chercher les valeurs NULL via la commande suivante :

Data Cleaning

Trained_Data.isnull().sum()

[21]

Python

...	duration	0
	protocol_type	0
	service	0
	flag	0
	src_bytes	0
	dst_bytes	0
	land	0
	wrong_fragment	0
	urgent	0
	hot	0
	num_failed_logins	0
	logged_in	0
	num_compromised	0
	root_shell	0
	su_attempted	0
	num_root	0
	num_file_creations	0
	num_shells	0
	num_access_files	0
	num_outbound_cmds	0
	is_host_login	0
	is_guest_login	0
	count	0
	srv_count	0
	serror_rate	0
	...	
	dst_host_srv_error_rate	0
	attack	0
	level	0
	attack_state	0
	dtype: int64	

La même chose pour Tested_Data :

Tested_Data.isnull().sum()

Python

On vas maintenant utiliser la méthode `get_dummies` de la bibliothèque `pandas` pour transformer les valeurs des colonnes contenant des valeurs catégoriels sans ordre en valeurs réel compréhensible par le modèle de Machine Learning, et il est préférable d'utiliser one-hot encoding (via `pd.get_dummies`)

```
Trained_Data = pd.get_dummies(Trained_Data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
Tested_Data = pd.get_dummies(Tested_Data,columns=['protocol_type','service','flag'],prefix="",prefix_sep="")
```

Python

Voici comment la dataset se présente après l'application de `get_dummies`:

Trained_Data

Python

	duration	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	num_failed_logins	logged_in	num_compromised	...	REJ	RSTO	RSTOS0	RSTR	S0
0	0	146	0	0	0	0	0	0	0	0	...	False	False	False	False	False
1	0	0	0	0	0	0	0	0	0	0	...	False	False	False	False	True
2	0	232	8153	0	0	0	0	0	1	0	...	False	False	False	False	False
3	0	199	420	0	0	0	0	0	1	0	...	False	False	False	False	False
4	0	0	0	0	0	0	0	0	0	0	...	True	False	False	False	False
...
125967	0	0	0	0	0	0	0	0	0	0	...	False	False	False	False	True
125968	8	105	145	0	0	0	0	0	0	0	...	False	False	False	False	False
125969	0	2231	384	0	0	0	0	0	1	0	...	False	False	False	False	False
125970	0	0	0	0	0	0	0	0	0	0	...	False	False	False	False	True
125971	0	151	0	0	0	0	0	0	1	0	...	False	False	False	False	False

125972 rows x 125 columns

Maintenant on vas utiliser le Label Encoding pour convertir les labels catégorielles en des valeurs numériques, plus précisément on vas utiliser la class `LabelEncoder()` de `sklearn.preprocessing` pour encoder la colonne « attaque » dans `Trained_Data` et `Tested_Data`.

```
LE = LabelEncoder()
attack_LE= LabelEncoder()
Trained_Data['attack'] = attack_LE.fit_transform(Trained_Data["attack"])
Tested_Data['attack'] = attack_LE.fit_transform(Tested_Data["attack"])
```

Python

`.get_dummies` is one-hot encoding .`LabelEncoder` is incremental encoding, such as 0,1,2,3,4,... and her just we use both method , but we can apply `get_dummies` for both , also for `labelEncoder`

Les lignes suivantes enlèvent les colonnes ' attack ', ' level ' et ' attack_state ' de Trained_Data

```
# These lines remove the columns 'attack', 'level', and 'attack_state' from the Trained_Data

# features of traindata
X_train = Trained_Data.drop('attack', axis = 1)
X_train = Trained_Data.drop('level', axis = 1)
X_train = Trained_Data.drop('attack_state', axis = 1)

# features of testdata
X_test = Tested_Data.drop('attack', axis = 1)
X_test = Tested_Data.drop('level', axis = 1)
X_test = Tested_Data.drop('attack_state', axis = 1)

# set target of traindata/setdata
Y_train = Trained_Data['attack_state']
Y_test = Tested_Data['attack_state']
```

Python

Remarque:

X_train et X_test contiennent les caractéristiques des colonnes (input data for the model), à l'exclusion de la colonne cible (attack_state).

Y_train et Y_test contiennent la colonne cible(attack_state), c'est la colonne que le modèle essaiera de prédire.

PARTIE 2 :**1-Conception de notre modèle de machine Learning (Decision Tree):**

La partie suivante utilise RobustScaler de sklearn.preprocessing pour normaliser les données d'entraînement et de test de manière robuste aux valeurs aberrantes(les données qui diffèrent fortement des autres points)

```

Ro_scaler = RobustScaler()
X_train_train = Ro_scaler.fit_transform(X_train_train)
X_test_train= Ro_scaler.transform(X_test_train)
X_train_test = Ro_scaler.fit_transform(X_train_test)
X_test_test= Ro_scaler.transform(X_test_test)

X_train_train

array([[ 0.00000000e+00,  5.68840580e-01,  2.38301887e+00, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00,  5.28985507e-01,  0.00000000e+00, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00, -1.30434783e-01,  0.00000000e+00, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       ...,
       [ 0.00000000e+00,  1.07608696e+00,  5.30188679e-01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00,  7.24637681e-03,  2.11320755e-01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 0.00000000e+00,  7.46376812e-01,  2.94018868e+01, ...,
         0.00000000e+00,  0.00000000e+00,  0.00000000e+00]],
      shape=(94479, 124))

```

Python

Maintenant on vas voir le DecisionTreeClassifier :

```

DT =DecisionTreeClassifier(max_features=6, max_depth=4)
DT.fit(X_train_train, Y_train_train)

```

Python

DecisionTreeClassifier

DecisionTreeClassifier(max_depth=4, max_features=6)

D'après les résultats on a obtenue les résultats suivants:

max_depth = 4 (la hauteur maximale de l'arbre)

max_features = 6 (Le nombre de fonctionnalités à prendre en compte lors de la recherche de la meilleure répartition)

Remarque : la recherche d'une division ne s'arrête pas tant qu'au moins une partition valide des échantillons de nœuds n'est pas trouvée, même si elle nécessite d'inspecter efficacement plus de max_features.

Maintenant on va définir une fonction qui va nous donner le taux de precision de notre modèle Décision Tree, et une graph qui se nomme Confusion Matrix, et finalement une courbe ROC qui représente visuellement les performance du modèle :

Le code qui permet de faire cela est le suivant :

```
def Evaluate(Model_Name, Model_Abb, X_test, Y_test):
    # Predictions and metrics
    Pred_Value = Model_Abb.predict(X_test)
    Accuracy = metrics.accuracy_score(Y_test, Pred_Value)
    Sensitivity = metrics.recall_score(Y_test, Pred_Value)
    Precision = metrics.precision_score(Y_test, Pred_Value)
    F1_score = metrics.f1_score(Y_test, Pred_Value)
    Recall = metrics.recall_score(Y_test, Pred_Value)

    # Display Metrics
    print('-----\n')
    print('The {} Model Accuracy   = {}'.format(Model_Name, np.round(Accuracy, 3)))
    print('The {} Model Sensitivity = {}'.format(Model_Name, np.round(Sensitivity, 3)))
    print('The {} Model Precision   = {}'.format(Model_Name, np.round(Precision, 3)))
    print('The {} Model F1 Score    = {}'.format(Model_Name, np.round(F1_score, 3)))
    print('The {} Model Recall     = {}'.format(Model_Name, np.round(Recall, 3)))
    print('-----\n')

    # Confusion Matrix
    Confusion_Matrix = metrics.confusion_matrix(Y_test, Pred_Value)
    fig, ax = plot_confusion_matrix(conf_mat=Confusion_Matrix,
                                    class_names=['Normal', 'Attack'],
                                    colorbar=True)
    plt.show()

    # ROC Curve
    RocCurveDisplay.from_estimator(Model_Abb, X_test, Y_test)
    plt.show()
```

Python

Voici les résultats obtenus :

```
Evaluate('Decision Tree Classifier', DT, X_test_train, Y_test_train)
```

Python

```
-----
The Decision Tree Classifier Model Accuracy   = 0.864
The Decision Tree Classifier Model Sensitivity = 0.76
The Decision Tree Classifier Model Precision  = 0.939
The Decision Tree Classifier Model F1 Score   = 0.84
The Decision Tree Classifier Model Recall    = 0.76
```

Le score obtenu de notre modèle est : **0.864**

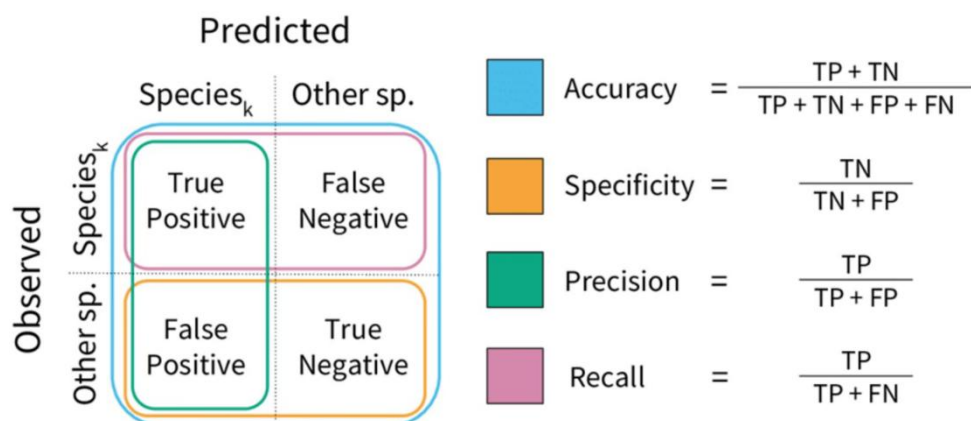
Remarque :

Figure 14 Model Métriques

Et le F1-score est une métrique de classification qui mesure la capacité d'un modèle à bien prédire les individus positifs, tant en termes de précision (taux de prédictions positives correctes) qu'en termes de recall (taux de positifs correctement prédits)

$$F1\ score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Voici un tableau qui donne l'interprétation de F1 score :

F1 score	Interpretation
> 0.9	Very good
0.8 - 0.9	Good
0.5 - 0.8	OK
< 0.5	Not good

Figure 15 F1 score interprétation

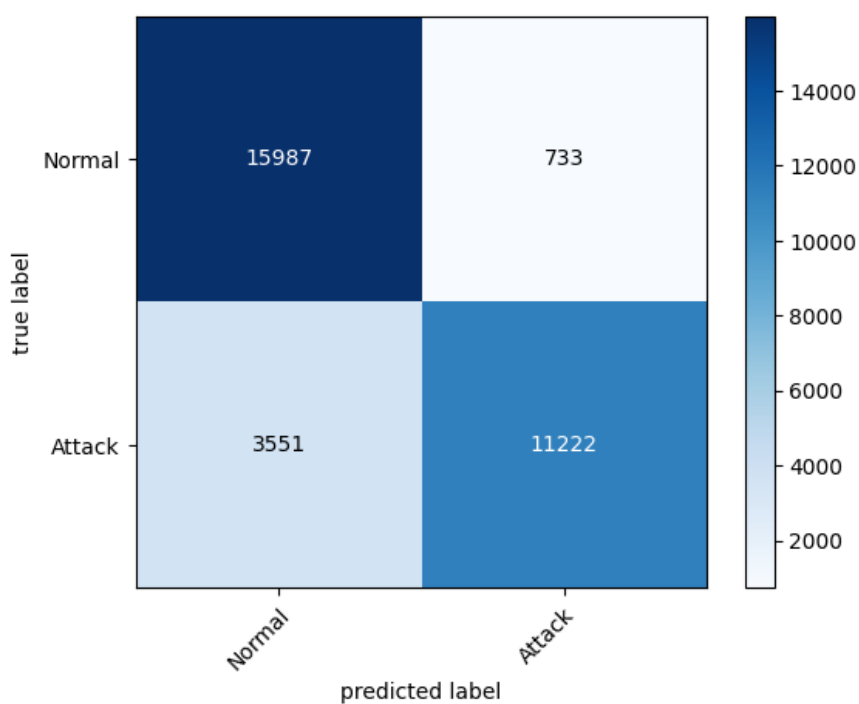
Pour le graph de Confusion Matrix :

la matrice de confusion est une matrice qui mesure la qualité de notre système de classification:

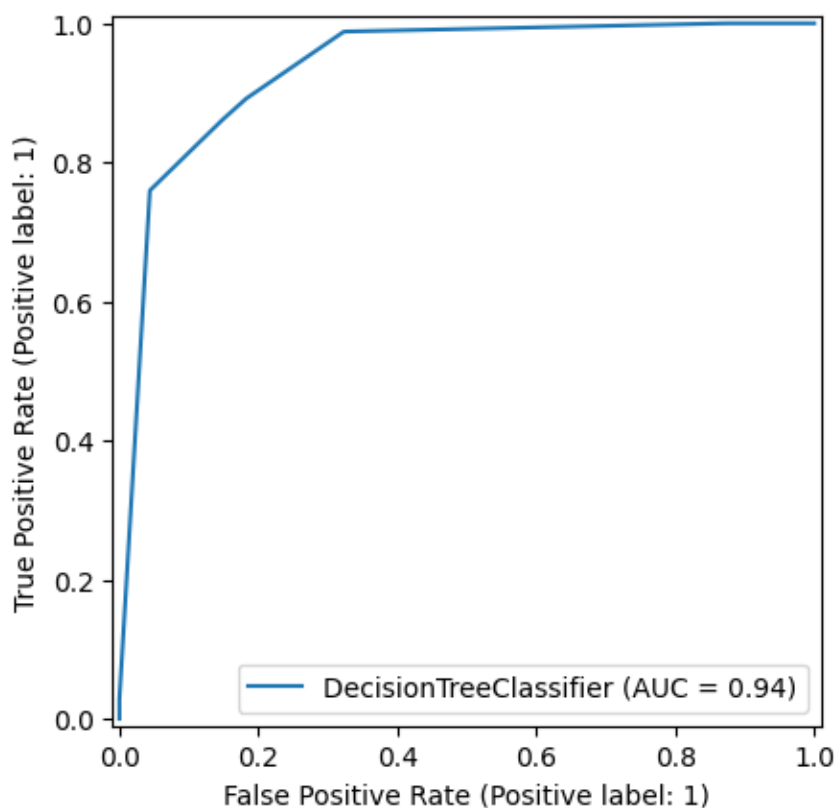
		Predicted class	
		P	N
Actual Class	P	True Positives (TP)	False Negatives (FN)
	N	False Positives (FP)	True Negatives (TN)

Figure 16 Confusion Matrix

Voici le résultat obtenu:



Voici la courbe ROC qui représente visuellement les performance du modèle Decision Tree:



Remarque :

La courbe ROC est dessinée en calculant le taux de vrais positifs (TPR). et de faux positifs pour chaque seuil possible (en pratique, des intervalles sélectionnés), puis le TPR et le FPR.

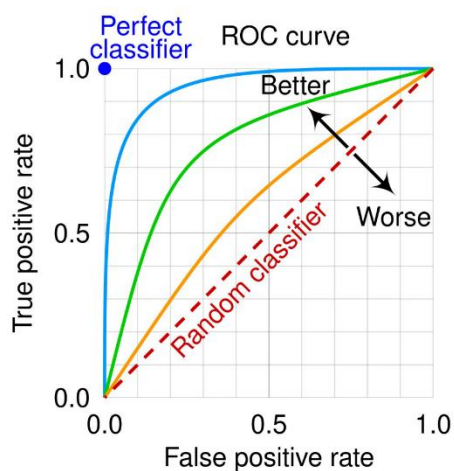


Figure 17 ROC curve

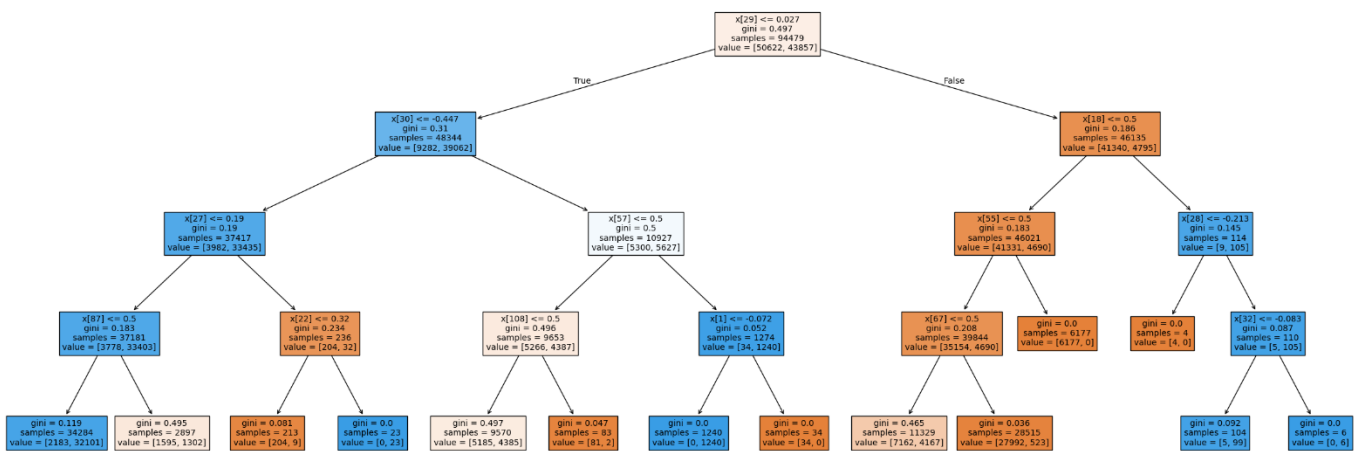
Maintenant on vas visualiser notre arbre de décision grace à la commande suivante:

visualize the structure of the trained decision tree.

```
fig = plt.figure(figsize=(30,11))
tree.plot_tree(OT, filled=True)
```

Python

Voici le résultat :



2-Conception de notre Modele de Machine Learning (Random Forest):

Pour le modèle de Random Forest on va suivre le meme principe de Décision Tree :

Voici la commande qui permet de préciser la hauteur de chaque sous-arbre.

```
max_depth = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30]
```

```
Parameters = {'max_depth': max_depth}
```

```
Parameters
```

Python

```
{'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 20, 25, 30]}
```

Ici on a défini une fonction qui se nomme GridSearch qui permet d'optimiser les hyperparamètres d'un modèle en testant toutes les combinaisons possibles d'un espace de paramètres défini, afin de trouver la configuration qui maximise la performance du modèle.

```
def GridSearch(Model_Abb, Parameters, X_train, Y_train):  
    Grid = GridSearchCV(estimator=Model_Abb, param_grid= Parameters, cv = 3, n_jobs=-1) # Create a GridSearchCV object with  
                                                #cross-validation and parallelization  
    ## Fit the model using the training data and parameter grid  
    Grid_Result = Grid.fit(X_train, Y_train)  
    # Get the best model (with the optimal hyperparameters)  
    Model_Name = Grid_Result.best_estimator_  
  
    return (Model_Name)
```

Python

Dans le code suivant on fait appelle a la fonction GridSearch pour trouver la meilleure configuration d'hyperparamètre pour le modèle Random Forest :

```
RF= RandomForestClassifier()  
GridSearch(RF, Parameters, X_train_train, Y_train_train)
```

[47]

Python

```
▼ RandomForestClassifier ⓘ ?  
RandomForestClassifier(max_depth=13)
```

Le résultat affiché montre la meilleure configuration d'hyperparamètres pour le modèle, ici on a trouvé que max_depth=13. Cela signifie que l'hyperparamètre max_depth (profondeur maximale des arbres dans la forêt) a été optimisé, et la valeur optimale trouvée est 13.

Et maintenant on entraine le modèle Random Forest

```
RF.fit(X_train_train, Y_train_train)
```

Python

```
▼ RandomForestClassifier ⓘ ?  
RandomForestClassifier()
```

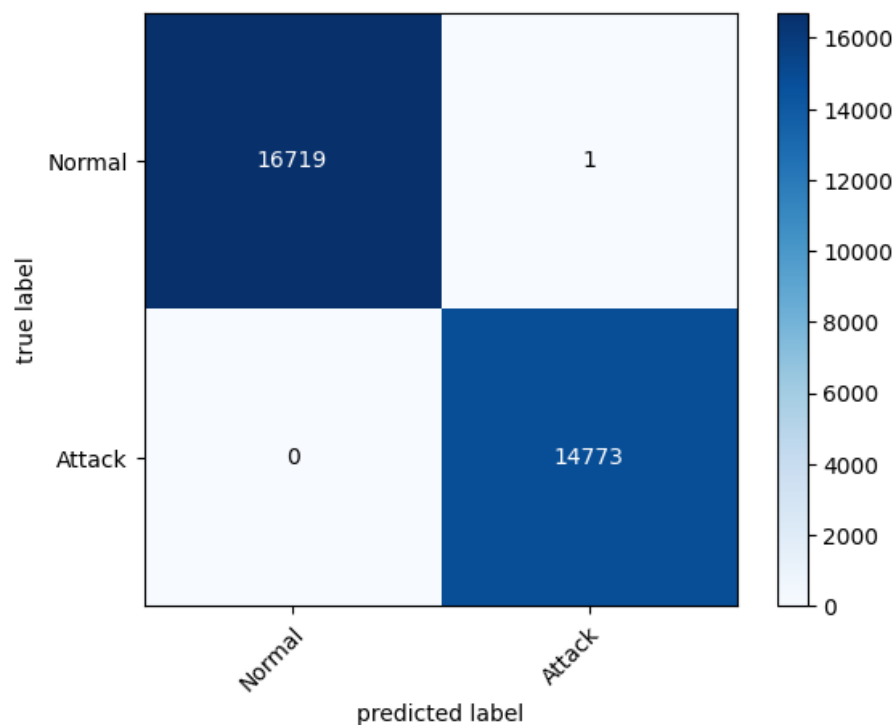
Voici le résultat quand on appelle la fonction evaluate:

```
Evaluate('Random Forest Classifier', RF, X_test_train, Y_test_train)
```

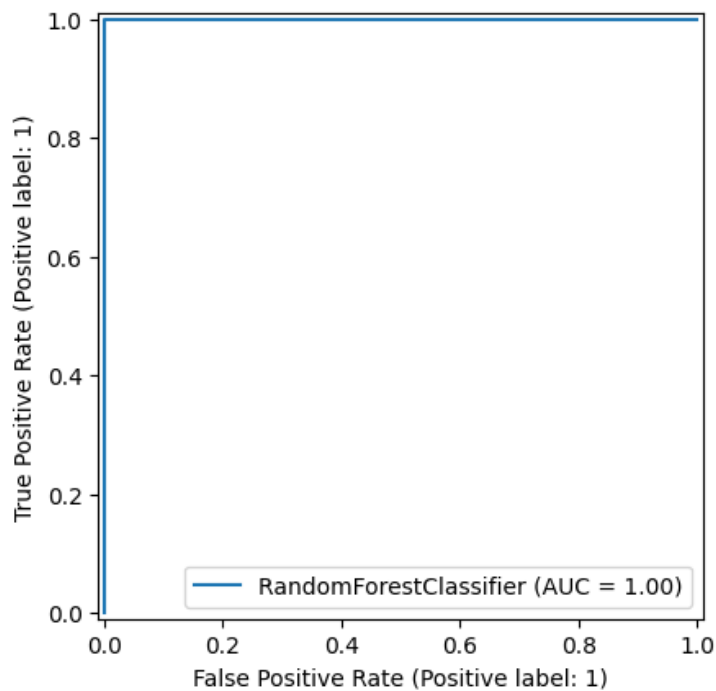
```
-----  
The Random Forest Classifier Model Accuracy  = 1.0  
The Random Forest Classifier Model Sensitivity = 1.0  
The Random Forest Classifier Model Precision  = 1.0  
The Random Forest Classifier Model F1 Score  = 1.0  
The Random Forest Classifier Model Recall    = 1.0  
-----
```

Le score obtenu de notre modèle (Random Forest) est : 1.0

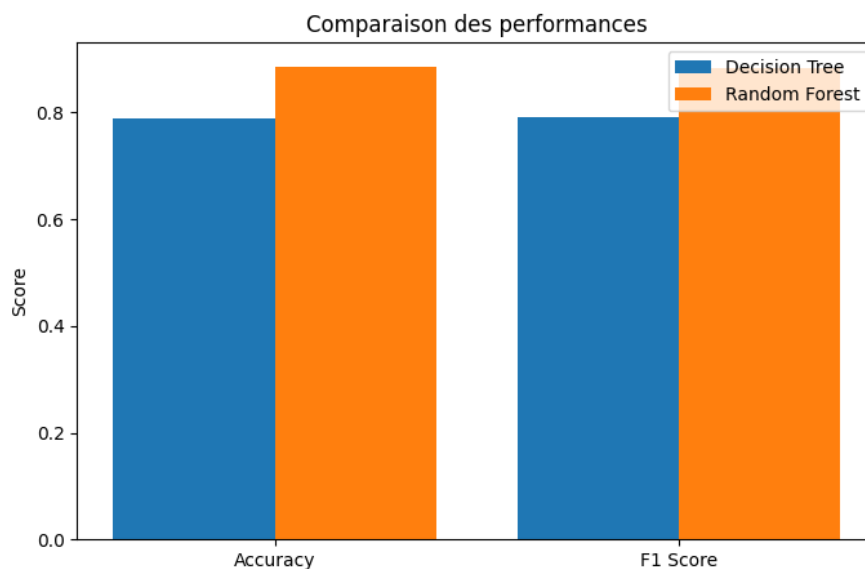
Voici le graph de Confusion Matrix :



Voici la courbe ROC qui représente visuellement les performances du modèle Random Forest :



Comparaison entre les deux modèles (Decision Tree et Random Forest) :



Le graphe au-dessus nous donne une comparaison entre le modèle Decision Tree et Random Forest pour les deux métriques Accuracy et F1 score.

On remarque que l'Accuracy du modèle Random Forest est meilleure que celle de Decision Tree ce qui est normale car le modèle Random Forest utilise la prédiction d'une combinaison de plusieurs arbres de décision. On remarque aussi que pour le F1 score du modèle Random Forest est meilleure que celui du Decision Tree ce qui montre que dans le modèle Random Forest on a une meilleure détection des TRUE POSITIVES.

Conclusion et critiques :

Voici les résultats obtenus pour les 2 models Decision Tree et Random Forest :

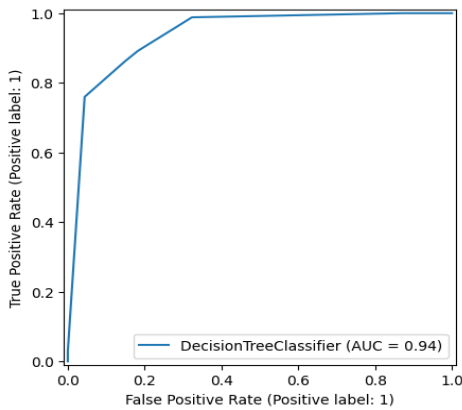
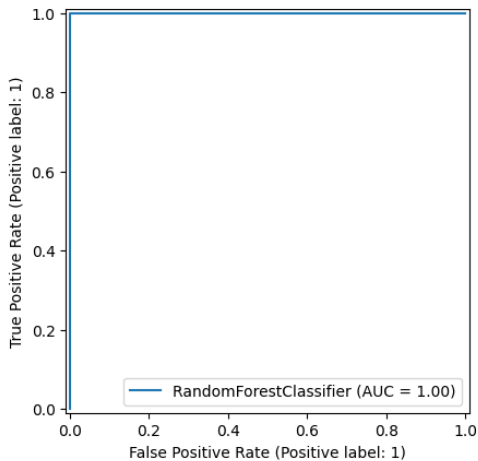
	Decision Tree	Random Forest
Model Accuracy	0.864	1.0
Model Sensitivité	0.76	1.0
Model Précision	0.939	1.0
Model F1 Score	0.84	1.0
Model Recall	0.76	1.0
ROC CURVE	 <p>ROC Curve for DecisionTreeClassifier (AUC = 0.94)</p>	 <p>ROC Curve for RandomForestClassifier (AUC = 1.00)</p>

Tableau 3 Conclusion des résultats

Le modèle Random Forest a atteint des résultats idéaux avec des notes de 1.0 sur toutes les mesures (exactitude, précision, rappel, F1-score, et sensibilité). Cela indique que le modèle identifie toutes les menaces sans produire de faux positifs ni de faux négatifs.

En revanche, l'arbre de décision offre des résultats corrects, bien que moins efficaces. Par exemple, sa précision est de 0.864 et son Reccal de 0.76, ce qui indique qu'il y a des menaces potentielles qui sont négligées.

Interprétation :

Bien que les résultats du Random Forest soient encourageants, un score idéal peut signaler un problème d'overfitting, où le modèle "retient" les données d'entraînement sans acquérir des généralisations pertinentes pour de nouvelles données. Les arbres de décision, bien qu'ils ne soient pas aussi efficaces, paraissent être moins sensibles au surapprentissage, ce qui pourrait les rendre plus résistants dans certains cas.

Critique des Résultats :

Overfitting Possible pour le Random Forest

Les résultats obtenus montrent des scores parfaits pour le modèle Random Forest sur toutes les métriques (Accuracy, précision, recall, F1-score, sensibilité). Cependant, ces performances idéales peuvent indiquer un phénomène de surapprentissage (overfitting).

- Origine du problème : notre modèle semble avoir "mémorisé" les données d'entraînement plutôt que d'apprendre des schémas généralisables applicables à de nouvelles données.
- Conséquence : Cela pourrait limiter son efficacité lorsqu'il est exposé à des données réelles ou non vues, où il est probable qu'il soit incapable de détecter les menaces ou de maintenir des performances similaires.

Manque de Validation sur Données Réelles

Les performances actuelles semblent être évaluées sur des données d'entraînement ou un ensemble de validation interne. Cette méthode est insuffisante pour garantir que le modèle est efficace dans des situations réelles.

Recommandations :

Il est également possible de tester des algorithmes plus avancés pour enrichir l'efficacité du système de détection d'anomalies dans les environnements cloud. XGBoost, par exemple, représente une excellente option pour les scénarios où les données se révèlent volumineuses et déséquilibrées. Sa rapidité et sa capacité à gérer les interactions complexes entre les variables en font un choix optimal pour détecter des menaces sophistiquées. Cependant, l'exploration de méthodes de machine learning non supervisé peut offrir une approche innovante pour détecter des anomalies inédites ou imprévues. Des algorithmes tels que les Autoencoders, le clustering avec k-means, ou les modèles de Gaussian Mixture peuvent être utilisés pour identifier des écarts significatifs dans les comportements des données sans avoir besoin de labels préalables. Cela est particulièrement utile dans des environnements dynamiques, où de nouvelles menaces émergent régulièrement.

Exemple d'Utilisation :

L'entreprise XYZ, ayant un serveur avec des ressources modestes en RAM et CPU, envisage de mettre en place un système de détection des menaces s'appuyant sur le Machine Learning pour examiner les journaux produits chaque jour par son infrastructure cloud.

En raison de la limitation des ressources, l'entreprise opte pour des modèles de machine Learning plus légers, tels que Random Forest ou Decision Tree, qui consomment peu de mémoire et de puissance de calcul. Ces modèles facilitent l'identification d'anomalies et de comportements douteux dans les journaux en temps réel, tout en réduisant l'effet sur la performance du serveur. Afin d'optimiser d'avantage les ressources, l'entreprise effectue un prétraitement des données des logs en ne conservant que les informations essentielles et en employant des échantillons de données au lieu de traiter l'ensemble des logs. De cette manière, l'entreprise est en mesure d'assurer un suivi efficace de sa sécurité tout en tenant compte de ses limitations matérielles.

Conclusion Générale

Pour conclure ce mini-projet, on peut dire que la sécurité cloud est un domaine très importants de nos jours, c'est pour cela dans le cadre de ce mini-projet on a créé deux modèles de Machine Learning (Decision Tree et Random Forest) qui permettent de détecter les intrusion dans un environnement cloud, et on a trouvé des résultats très intéressant (0.864 pour l'Accuracy de Decision Tree et 1.0 pour l'Accuracy de Random Forest), mais même avec ces résultats on a plusieurs remarques, tels que notre solution à besoin de validation sur des données réelles pour une meilleur prédiction d'anomalie toute en minimisant les faux positifs

Durant ce mini-projet, nous avons rencontré plusieurs difficultés tels que le choix de la DataSet appropriée aussi le modèle du Machine Learning, de même comment adopter la modélisation de la recherche opérationnelle dans notre mini-projet.

D'autre part le monde du Cloud Computing est dans une révolution constante, et notre mini-projet traite un aspect très critique qui concerne la sécurité des environnements cloud, et les grandes entreprises du Cloud Computing investissent de grande somme d'argents pour développer des solutions de sécurité. Et pour donner une continuité future à ce mini-projet on peut dire qu'on peut encore mieux le développer pour pouvoir gérer des vrai environnements cloud.

Finalement, dans ce mini-projet on a appris comment mieux travailler en équipe, et augmenter l'esprit de collaboration ce qui nous a permis de bien réaliser ce mini-projet et bien comprendre nôtre sujet.

Bibliographie

- [1] <https://www.iguazio.com/glossary/false-positive-rate/#:~:text=What%20is%20the%20False%20Positive,to%20measure%20binary%20context%20problems>
- [2] <https://developers.google.com/machine-learning/crash-course/classification/thresholding>
- [3] Article Cloud-Based Intrusion Detection Approach Using Machine Learning Techniques
<https://ieeexplore.ieee.org/abstract/document/10097662>
- [4] <https://www.theinsightpartners.com/fr/reports/cloud-security-market>
- [5] <https://www.mordorintelligence.com/fr/industry-reports/global-cloud-network-security-software-market-industry>
- [6] <https://secureframe.com/fr-fr/blog/cloud-security-statistics>
- [7] <https://www.oracle.com/a/ocom/docs/top-five-cloud-security-trends-fr.pdf>
- [8] <https://aws.amazon.com/fr/what-is-cloud-computing/>
- [9] <https://www.ibm.com/think/topics/cloud-computing>
- [10] <https://www.ibm.com/fr-fr/topics/cloud-security>
- [11] <https://www.geeksforgeeks.org/decision-tree-introduction-example/>
- [12] <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>