ECOLE NORMALE SUPÉRIEURE DE
L'ENSEIGNEMENT TECHNIQUE DE MOHAMMEDIA

UNIVERSITÉ HASSAN II DE CASABLANCA

ENSET

المدرسة العليا لأساتذة التعليم التقني
المحمدية
جامعة الحسن الثاني بالدار البيضاء

DEPARTEMENT MATHEMATIQUES ET INFORMATIQUE

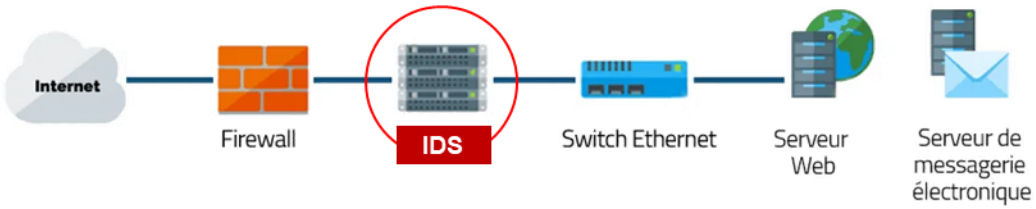
« **I**ngénierie **I**nformatique **: B**ig **D**ata et **C**loud **C**omputing »

II-BDCC - 2

Machine Learning

Rapport de projet

Système de Détection d'Intrusion Réseaux



Réalisé par :

- Soulaimane Cherkaoui
- Zaid El Mouaddibe

Encadré par :

- Pr. Soufiane Hamida

Année Universitaire : 2023 - 2024

ENSET, Avenue Hassan II - B.P. 159 - Mohammedia - Maroc
☎ 05 23 32 22 20 / 05 23 32 35 30 – **Fax :** 05 23 32 25 46 - **Site Web :** www.enset-media.ac.ma
E-Mail : contact@enset-media.ac.ma

Sommaire

Introduction

Dans un univers numérique en perpétuelle évolution, la sécurité des réseaux informatiques est devenue une pierre angulaire pour les entreprises de toutes tailles. Face à des cybermenaces toujours plus sophistiquées, le développement de systèmes de détection d'intrusion réseaux (NIDS) intégrant le Machine Learning (ML) représente une avancée majeure. Ces systèmes visent non seulement à identifier les intrusions connues à travers des signatures prédéfinies mais également à apprendre et à s'adapter à de nouvelles menaces, garantissant ainsi une sécurité dynamique et évolutive.

L'intérêt croissant pour le Machine Learning dans le domaine de la cybersécurité découle de sa capacité à traiter et à analyser des volumes massifs de données en temps réel, permettant la détection précoce d'activités suspectes et malveillantes. Contrairement aux approches traditionnelles, qui reposent sur la reconnaissance de signatures d'attaques déjà connues, les modèles de ML peuvent identifier des comportements anormaux sans connaissance préalable spécifique, offrant ainsi une protection contre les attaques zero-day et les menaces émergentes.

Notre projet, intitulé **Machine Learning : Système de Détection d'Intrusion Réseaux,** s'inscrit dans cette perspective innovante. Il ambitionne de créer un modèle de ML capable de distinguer avec précision et efficacité les trafics réseau normaux des tentatives d'intrusion. Ce modèle est ensuite rendu accessible via une application web intuitive, développée avec Streamlit, facilitant son utilisation par des professionnels de la cybersécurité ainsi que par des passionnés.

L'importance de ce projet réside dans sa contribution à la sécurisation des infrastructures critiques, en réduisant le risque de dommages potentiellement catastrophiques causés par des cyberattaques.

Partie 1 : Description du projet

Ce projet de Machine Learning (Système de Détection d'Intrusion Réseaux), représente notre exploration approfondie des capacités du Machine Learning appliqué à la cybersécurité, dans le cadre de notre formation en ingénierie informatique. Cette section présente les fondements, les objectifs pédagogiques et l'approche méthodologique de notre initiative.

Fondements du Système de Détection d'Intrusion Réseaux

Un Système de Détection d'Intrusion Réseaux (NIDS) est conçu pour surveiller le trafic sur un réseau informatique à la recherche d'activités suspectes ou manifestement malveillantes. Traditionnellement, les NIDS utilisent une combinaison de signatures connues d'attaques et d'analyses heuristiques pour identifier les menaces. Notre projet vise à explorer comment les techniques de Machine Learning peuvent enrichir cette approche, en permettant au système d'apprendre de flux de données et d'identifier des schémas indiquant une intrusion potentielle.

Objectif Éducatif

L'objectif principal de ce projet est double : d'une part, développer une compréhension pratique des applications du Machine Learning, et d'autre part, créer un modèle prototype capable de détecter les intrusions réseau.

À travers ce projet, nous souhaitons :

- **Acquérir une Connaissance Technique :** Comprendre le fonctionnement des systèmes de détection d'intrusion et comment les techniques de Machine Learning peuvent être appliquées pour améliorer leur efficacité.
- **Expérience Pratique :** Appliquer les connaissances théoriques acquises en classe à un problème réel, en passant par toutes les étapes de développement d'un modèle de Machine Learning.

Approche Méthodologique

Notre démarche s'articule autour de plusieurs phases clés :

1. **Collecte et Prétraitement des Données :** Importation de données depuis une source fiable et préparation de celles-ci pour l'entraînement, incluant le nettoyage et la normalisation.

2. **Sélection des Caractéristiques et Modélisation :** Identification des caractéristiques les plus pertinentes pour la détection d'intrusion et choix des modèles de Machine Learning adaptés à notre problème.
3. **Évaluation et Optimisation :** Test des modèles sélectionnés pour évaluer leur précision et affiner leur performance via des techniques d'optimisation comme le Voting Classifier.
4. **Développement d'une Interface Simple :** Création d'une application simplifiée permettant de tester le modèle sur des données nouvelles, destinée à illustrer son fonctionnement plutôt qu'à fournir une solution complète pour les organisations.

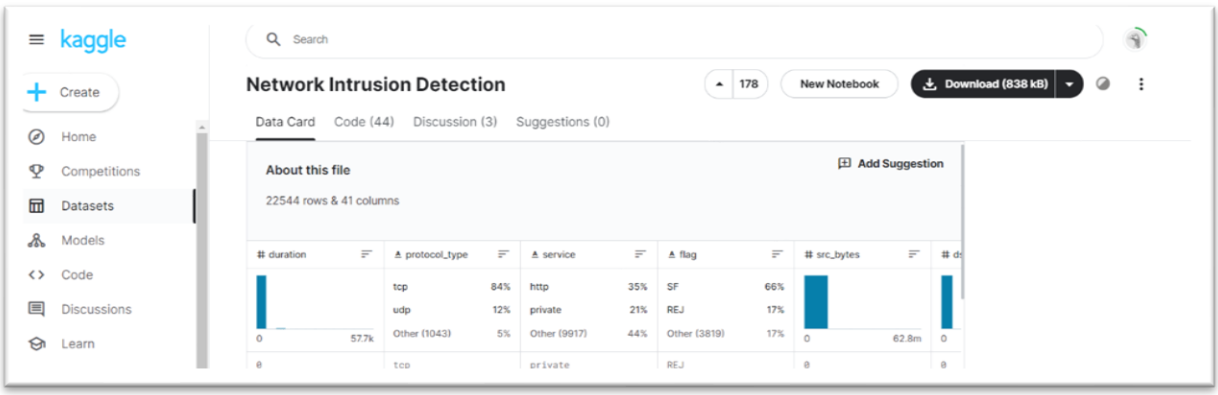
Partie 2 : Collecte des Données

La phase de collecte des données constitue la fondation sur laquelle repose notre projet de détection d'intrusion réseau via le Machine Learning. Cette étape cruciale a impliqué l'identification et l'acquisition d'un ensemble de données approprié pour entraîner et tester notre modèle. Voici comment nous avons procédé :

Source des Données

Pour obtenir un ensemble de données représentatif et riche, nous nous sommes tournés vers **Kaggle**, une plateforme en ligne qui héberge des compétitions de data science, des ensembles de données, des cours et des ressources pour les scientifiques des données et les passionnés d'apprentissage automatique.

Source : https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection



Importation des Données

Nous avons sélectionné un dataset spécifique à la détection d'intrusion, qui se distingue par sa diversité et sa pertinence pour notre projet. Ce dataset est divisé en deux parties principales :

- **Ensemble d'Entraînement :** Composé de données simulées représentant à la fois du trafic réseau normal et des activités malveillantes. Cet ensemble nous permet d'entraîner notre modèle en lui exposant à une large gamme de scénarios.
- **Ensemble de Test :** Utilisé pour évaluer la performance de notre modèle après l'entraînement. Cet ensemble contient des données non vues par le modèle durant la phase d'entraînement, ce qui nous aide à estimer son efficacité dans des conditions réalistes.

Prétraitement des Données

1. Compréhension des Données

Le prétraitement des données est une étape fondamentale dans notre projet. Elle assure que le modèle de Machine Learning reçoit des informations nettes, structurées et pertinentes pour l'entraînement. Voici comment nous avons abordé cette phase critique :

Nous avons conclu que la base de données se compose de 42 colonnes, où la colonne cible est 'attack', représentant différents types d'attaques. Si aucun type d'attaque n'est détecté, il est désigné comme "normal". Nous devons ainsi déterminer si un trafic est une attaque ou non en fonction des autres colonnes.

```
train.info()
✓ 0.0s
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 125972 entries, 0 to 125971
Data columns (total 43 columns):
 #   Column                       Non-Null Count   Dtype
---  ------                       --------------   -----
 0   duration                     125972 non-null  int64
 1   protocol_type                125972 non-null  object
 2   service                      125972 non-null  object
 3   flag                         125972 non-null  object
 4   src_bytes                    125972 non-null  int64
 5   dst_bytes                    125972 non-null  int64
 6   land                         125972 non-null  int64
 7   wrong_fragment               125972 non-null  int64
 8   urgent                       125972 non-null  int64
 9   hot                          125972 non-null  int64
 10  num_failed_logins            125972 non-null  int64
 11  logged_in                    125972 non-null  int64
 12  num_compromised              125972 non-null  int64
 13  root_shell                   125972 non-null  int64
 14  su_attempted                 125972 non-null  int64
 15  num_root                     125972 non-null  int64
 16  num_file_creations           125972 non-null  int64
 17  num_shells                   125972 non-null  int64
 18  num_access_files             125972 non-null  int64
 19  num_outbound_cmds            125972 non-null  int64
...
 41  attack                       125972 non-null  object
 42  last_flag                    125972 non-null  int64
dtypes: float64(15), int64(24), object(4)
memory usage: 41.3+ MB
```

2. Nettoyage des Données

```
Missing values in training data:
 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
...
dst_host_srv_rerror_rate      0
attack                        0
last_flag                     0
dtype: int64
```

- **Gestion des Valeurs Manquantes :** Malgré l'absence de valeurs nulles dans notre dataset initial, nous avons mis en place des mécanismes pour traiter d'éventuelles données manquantes dans de futures utilisations, assurant ainsi la robustesse de notre modèle.
- **Suppression des Caractéristiques Redondantes :** Nous avons identifié et éliminé les colonnes ne contenant que des valeurs constantes ou peu variées, qui n'apportent pas d'information significative pour la détection.

3. Encodage de données catégorielles en numérique

Entrée [12]:
```
categorical_cols = ['protocol_type', 'service', 'flag']
label_encoder = LabelEncoder()
for col in categorical_cols:
    train[col] = label_encoder.fit_transform(train[col])
    test[col] = label_encoder.transform(test[col])

# Handle the 'attack' feature by encoding it into numerical format
train['attack'] = np.where(train['attack'] == 'normal', 0, 1)
test['attack'] = np.where(test['attack'] == 'normal', 0, 1)

train.head()
```

Out[12]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_ho
0	0	2	44	9	146	0	0	0	0	0	...	0.00	0.60	
1	0	1	49	5	0	0	0	0	0	0	...	0.10	0.05	
2	0	1	24	9	232	8153	0	0	0	0	...	1.00	0.00	
3	0	1	24	9	199	420	0	0	0	0	...	1.00	0.00	
4	0	1	49	1	0	0	0	0	0	0	...	0.07	0.07	

5 rows × 43 columns

Nous avons converti les données catégoriques en données numériques pour faciliter la manipulation, puis nous avons classifié les attaques en deux catégories : 1 pour tout type d'attaque et 0 s'il n'y a pas d'attaque.

4. Standardisation des Données

Nous avons utilisé **StandardScaler** pour normaliser nos caractéristiques, garantissant que chaque attribut contribue équitablement à l'analyse sans être biaisé par son échelle. Cela est crucial pour des algorithmes comme KNN et SVC, qui sont sensibles aux variations d'échelle entre les caractéristiques.

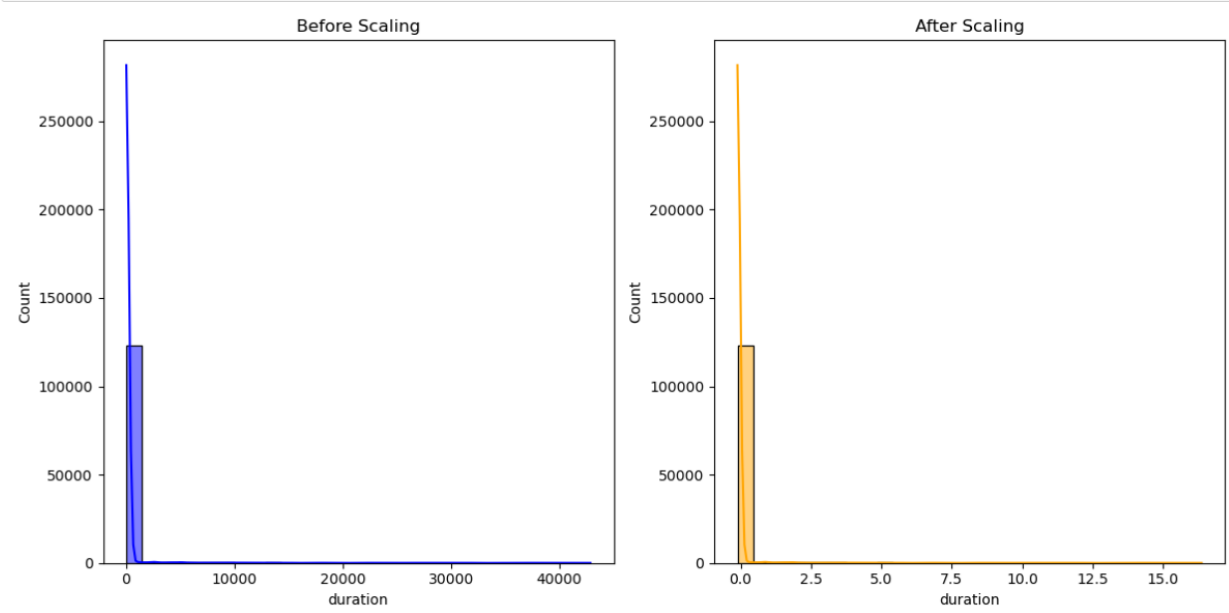
Entrée [13]:
```
# Copy the original dataframe to keep the original values for comparison
train_original = train.copy()

# Identifying Numerical Columns
numerical_cols = [col for col in train.columns if col not in ['attack'] + categorical_cols]
scaler = StandardScaler().fit(train[numerical_cols])  # Fit only on train data

# Calculating means and standard deviations for scaling
means = train[numerical_cols].mean()
stds = train[numerical_cols].std()

# Manual Scaling
train[numerical_cols] = (train[numerical_cols] - means) / stds
test[numerical_cols] = (test[numerical_cols] - means) / stds
train.head()
```

Out[13]:

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_same_srv_rate	dst_host_diff_srv_
0	-0.110249	2	44	9	-0.007737	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-1.161030	2.736
1	-0.110249	1	49	5	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-0.938288	-0.174
2	-0.110249	1	24	9	-0.007723	-0.002891	-0.014089	-0.089486	-0.007736	-0.095076	...	1.066389	-0.439
3	-0.110249	1	24	9	-0.007728	-0.004814	-0.014089	-0.089486	-0.007736	-0.095076	...	1.066389	-0.439
4	-0.110249	1	49	1	-0.007762	-0.004919	-0.014089	-0.089486	-0.007736	-0.095076	...	-1.005111	-0.068

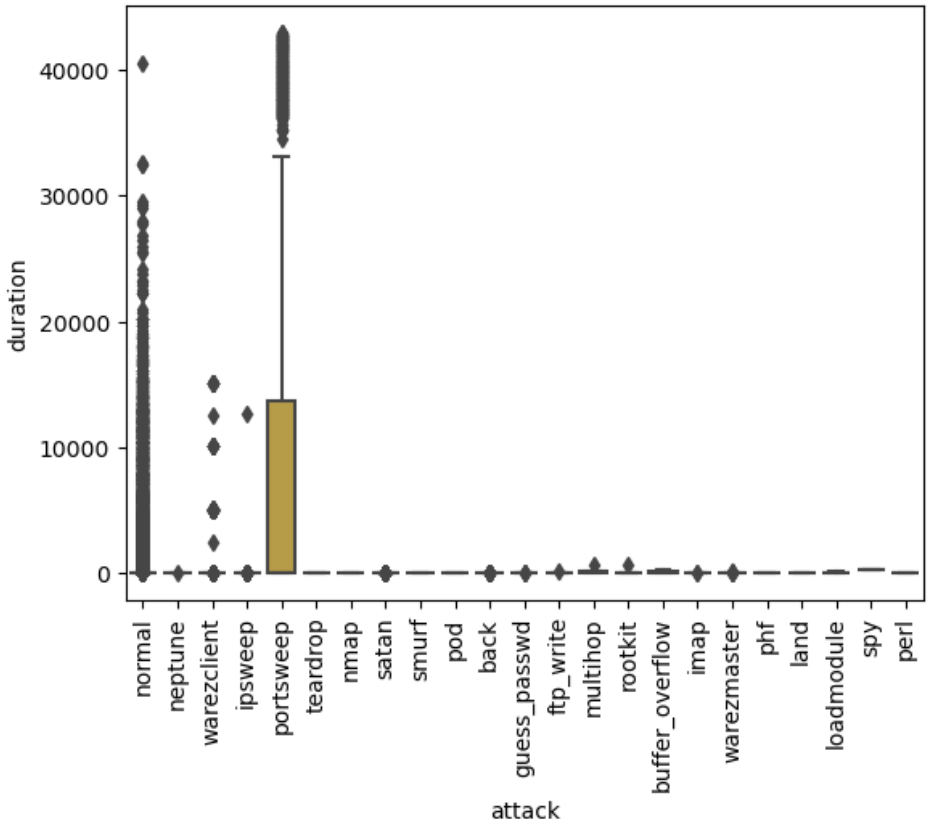
5 rows × 43 columns

Nous avons utilisé la fonction de mise à l'échelle (Scaler) pour mettre à l'échelle notre ensemble de données d'entraînement, où nous avons extrait la moyenne et l'écart type pour les réutiliser afin de mettre à l'échelle l'ensemble de données de test. Ensuite, nous avons appliqué cette transformation à l'ensemble des données.

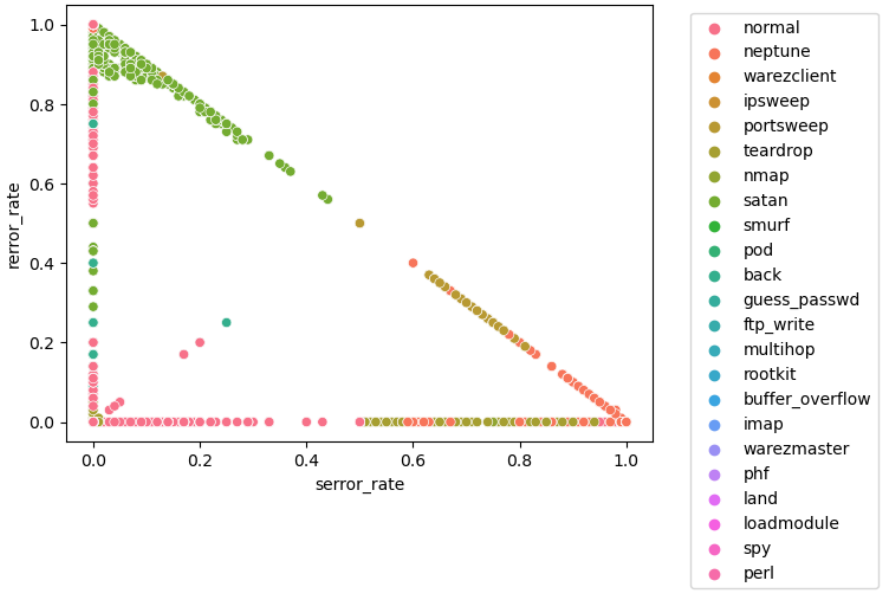


5. Visualisation des Données

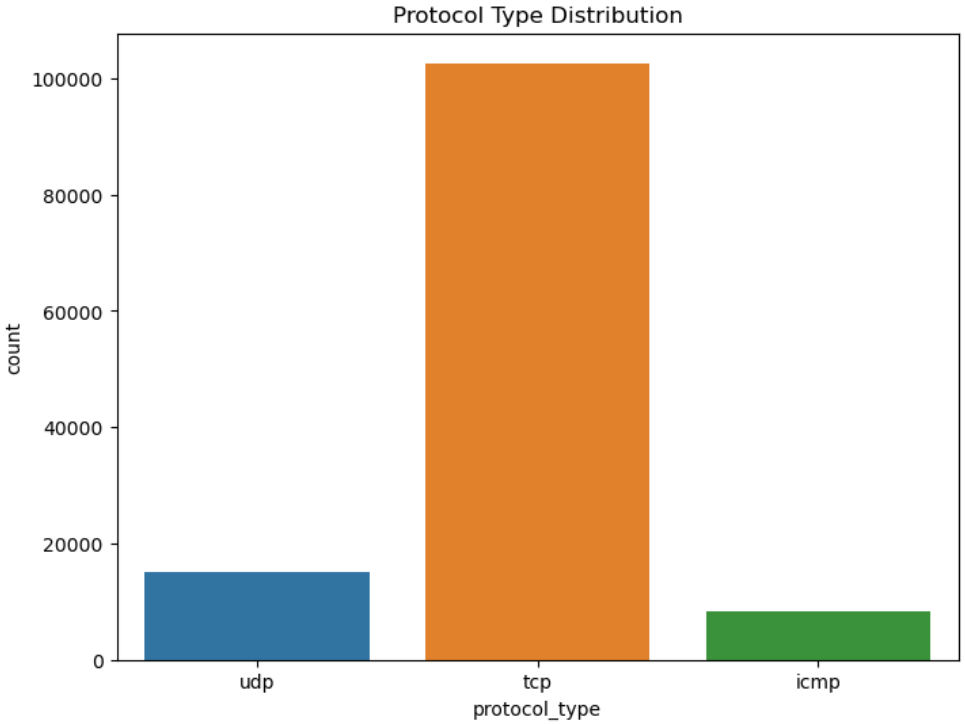
Des visualisations ont été générées pour mieux comprendre les relations entre les caractéristiques et la variable cible. Ces insights visuels nous ont aidés à identifier les schémas, les anomalies et les tendances dans les données, facilitant ainsi la sélection des caractéristiques les plus pertinentes.



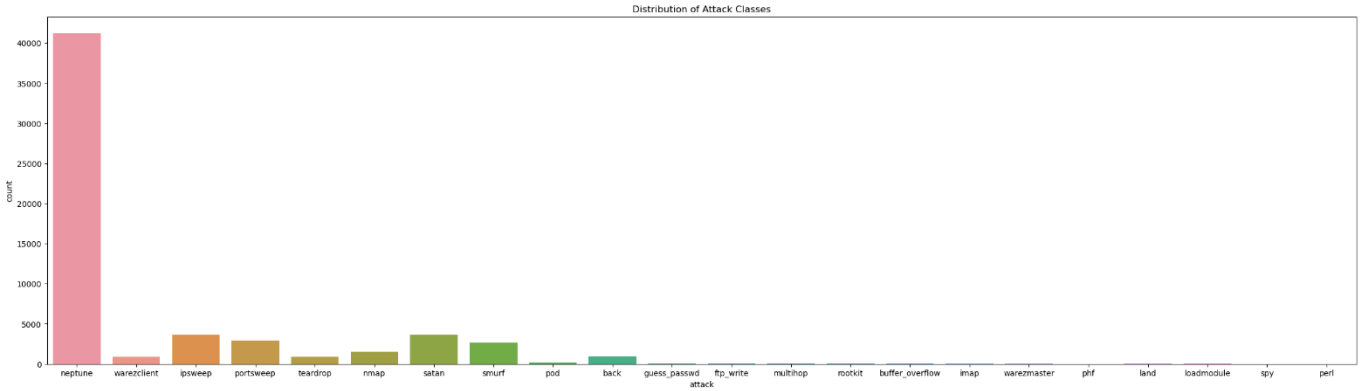
Graphique de la durée des connexions : Ce diagramme montre combien de temps durent les connexions pour les différents types d'attaques. Il semble que la plupart des activités normales ont des durées variées, alors que les attaques ont tendance à être plus courtes.



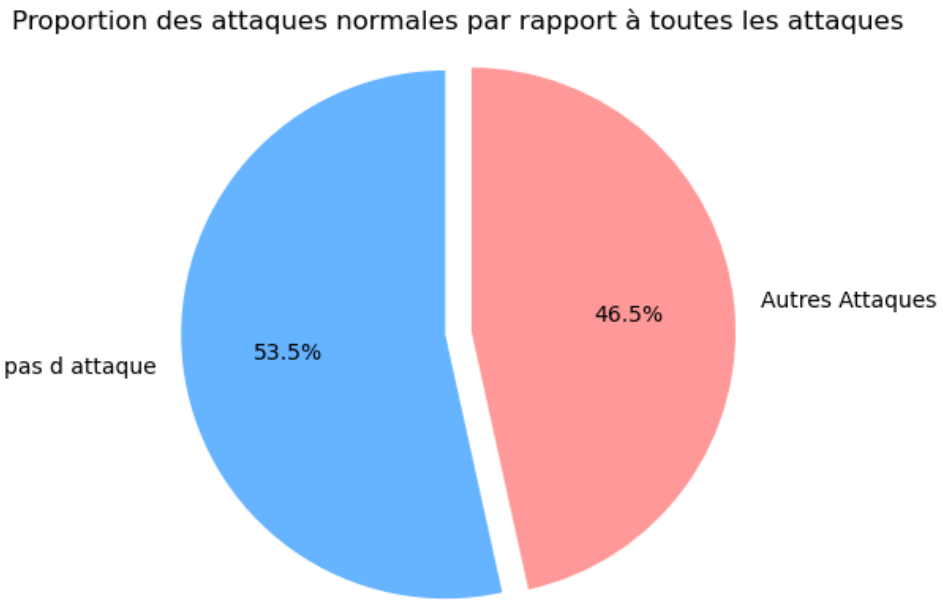
Graphique des taux d'erreurs : Ce nuage de points compare deux types d'erreurs dans le réseau.



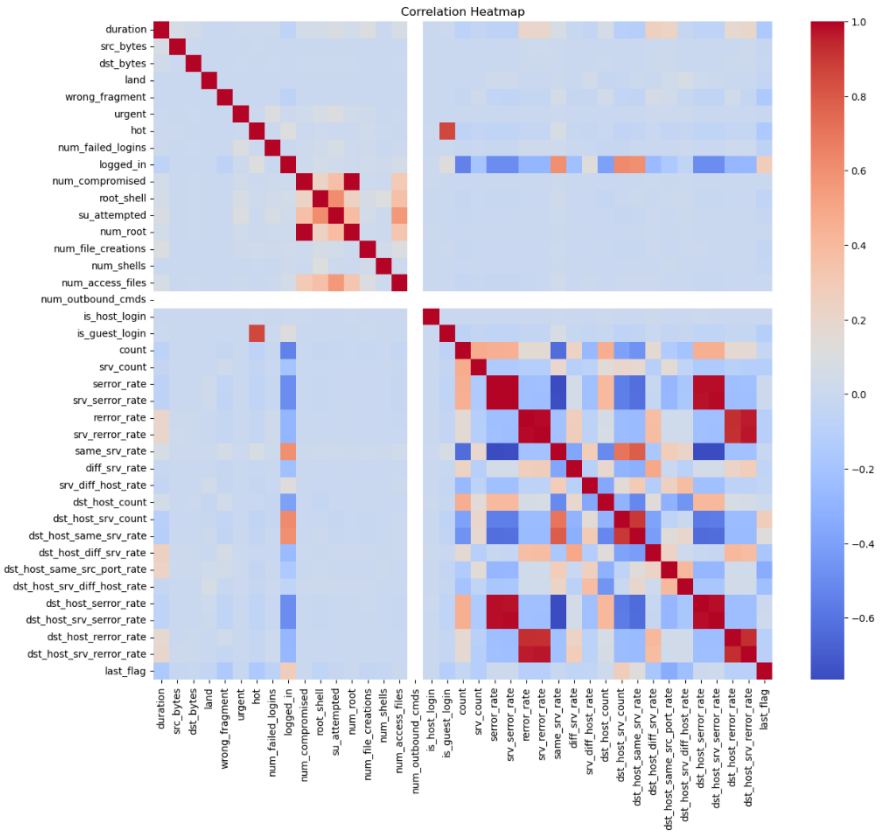
Répartition des protocoles réseau : On observe ici le nombre de fois que chaque type de protocole réseau apparaît. Le prrotocole 'tcp' est clairement le plus fréquent.



Distribution des types d'attaques : Ce graphique à barres nous indique combien chaque type d'attaque est commun. 'Neptune' ressort comme l'attaque la plus commune, avec beaucoup de trafic normal également présent.



Proportion des attaques normales : La moitié de la tarte montre le trafic normal, et l'autre moitié représente toutes les attaques détectées.



Heatmap de corrélation : Ce tableau coloré nous montre comment différentes informations réseau sont liées entre elles. Les couleurs plus chaudes indiquent une forte relation, tandis que les couleurs plus froides signifient peu ou pas de lien.

6. Sélection des Caractéristiques

La première étape que nous avons effectuée a été d'éliminer la colonne ne contenant que des zéros. Ensuite, nous avons utilisé différentes approches pour trouver les meilleures caractéristiques (features) pour l'entraînement des modèles.

```
Entrée [13]: train = train.drop(columns=['num_outbound_cmds'])
             test = test.drop(columns=['num_outbound_cmds'])
             train.info()
             train.head()
```

La sélection des meilleures caractéristiques dans ce cas semble être basée sur l'importance des caractéristiques mesurée par le modèle **ExtraTreesClassifier**. Voici comment cela fonctionne :

- **Entraînement du modèle :** ExtraTreesClassifier est instancié avec 100 arbres de décision (n_estimators = 100). Ce modèle est entraîné sur les données d'entraînement après avoir exclu la colonne 'attack', car nous cherchons à identifier les caractéristiques les plus importantes pour prédire la variable cible 'attack'.

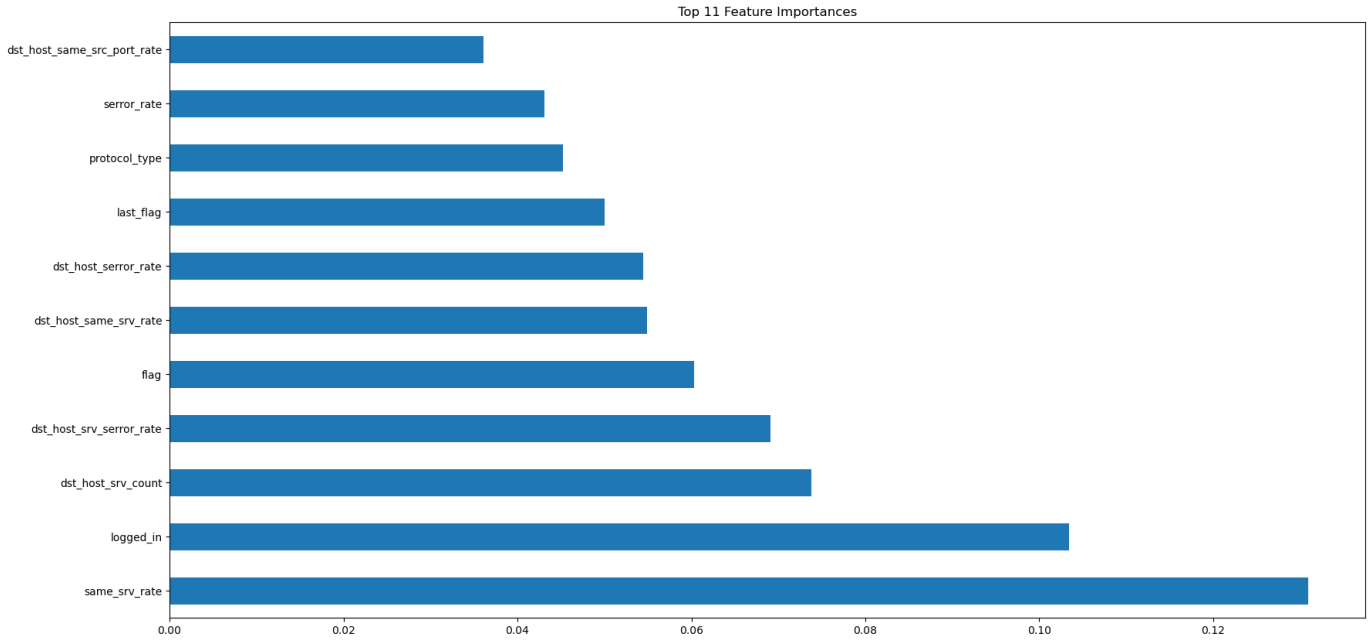
- **Mesure de l'importance des caractéristiques :** Une fois le modèle entraîné, l'importance de chaque caractéristique est calculée à l'aide de la méthode feature_importances_ du modèl . Cette mesure quantifie l'importance de chaque caractéristique dans la prédiction de la variable cible.

- **Sélection des meilleures caractéristiques** : Les caractéristiques sont ensuite triées par ordre d'importance, et les 11 caractéristiques ayant les plus grandes importances sont sélectionnées. Ces caractéristiques sont considérées comme les plus importantes pour prédire la variable cible 'attack'.

- **Visualisation des caractéristiques importantes** : Les 11 caractéristiques les plus importantes sont ensuite visualisées sous forme de diagramme à barres horizontal, permettant une compréhension visuelle de leur importance relative dans la prédiction de 'attack'.

```
# Using ExtraTreesClassifier for Feature Selection
model = ExtraTreesClassifier(n_estimators=100)
X_train = train.drop('attack', axis=1)  # Dropping the 'attack' column for feature selection
y_train = train['attack']
model.fit(X_train, y_train)

# Plotting Feature Importances
feat_importances = pd.Series(model.feature_importances_, index=X_train.columns)
top_features = feat_importances.nlargest(11).index
print(top_features)
plt.figure(figsize=(20, 10))
feat_importances.nlargest(11).plot(kind='barh')
plt.title('Top 15 Feature Importances')
plt.show()
print(y_train.unique())
```

Résultat :



Ensuite, nous allons utiliser uniquement le DataFrame contenant les meilleures caractéristiques que nous avons sélectionnées.

```
# Keeping top 15 features plus the target variable for both train and test datasets
train = train[top_features.tolist() + ['attack']]
test = test[top_features.tolist() + ['attack']]
train.head()
```

	logged_in	dst_host_srv_count	same_srv_rate	serror_rate	dst_host_serror_rate	protocol_type	srv_serror_rate	last_flag	dst_host_same_srv_rate	dst_host_
0	-0.809264	-1.035689	-1.321415	-0.637210	-0.639533	2	-0.631930	-1.965539	-1.161030	
1	-0.809264	-0.809859	-1.389655	1.602649	1.608744	1	1.605089	-0.219967	-0.938288	
2	1.235681	1.258741	0.771285	-0.189238	-0.572085	1	-0.184526	0.652820	1.066389	
3	1.235681	1.258741	0.771285	-0.637210	-0.639533	1	-0.631930	0.652820	1.066389	
4	-0.809264	-0.873091	-1.139441	-0.637210	-0.639533	1	-0.631930	0.652820	-1.005111	

Impact du Prétraitement : transformé les données brutes en un ensemble structuré et optimisé pour l'entraînement de modèles de Machine Learning. En nettoyant, normalisant et sélectionnant stratégiquement les caractéristiques les plus significatives.

Partie 3 : Choix des Modèles et Évaluation

Après avoir préparé nos données, nous avons procédé à la sélection des modèles de Machine Learning adaptés à la tâche de détection d'intrusion réseau. Cette sélection s'est basée sur des critères de performance, de pertinence pour le problème et de viabilité compte tenu de notre niveau de compétences en tant qu'étudiants ingénieurs.

1. Modèles Envisagés

Voici les modèles que nous avons considérés pour notre projet :

1. **Régression Logistique :** Choisie pour sa simplicité et son efficacité en classification, elle sert de point de départ pour évaluer la linéarité des données.

2. **Random Forest :** Préférée pour sa robustesse et sa capacité à traiter des données de grande dimensionnalité sans prétraitement extensif.

3. **Naive Bayes :** basé sur le théorème de Bayes avec l'hypothèse d'indépendance conditionnelle entre les caractéristiques. Sélectionné pour sa rapidité et son efficacité, surtout dans les cas où l'indépendance des caractéristiques est une hypothèse raisonnable.

4. **KNN (K-Nearest Neighbors) :** Intégré pour sa capacité à classifier en se basant sur la proximité des données dans l'espace des caractéristiques.

5. **SVC (Support Vector Classifier) :** un algorithme de classification qui cherche à trouver l'hyperplan optimal pour séparer les classes dans l'espace des caractéristiques. Utilisé pour sa performance dans les espaces de grande dimension et sa capacité à trouver la meilleure frontière de décision.

2. Méthodologie d'Évaluation

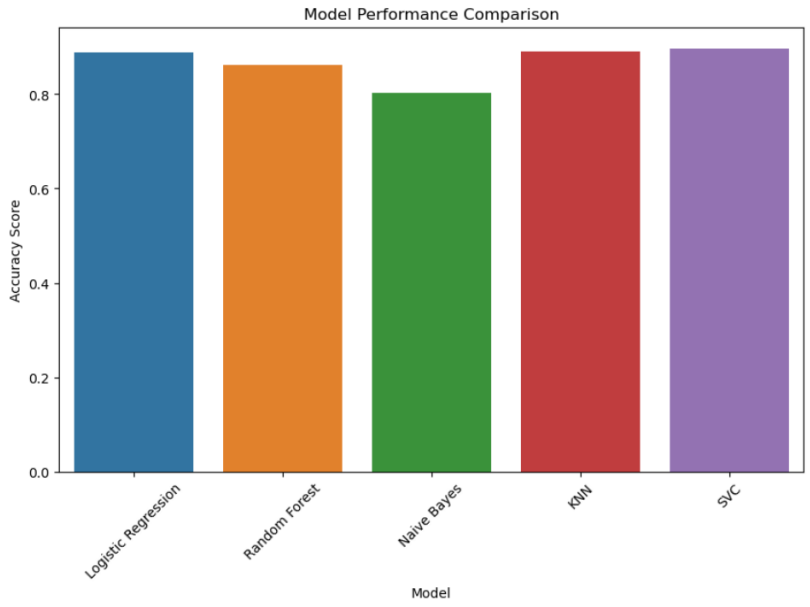
Chaque modèle a été entraîné sur notre ensemble de données normalisé et évalué sur l'ensemble de test. Nous avons utilisé l'**accuracy** comme métrique principale pour juger de la performance des modèles, ce qui nous permet d'avoir une estimation directe de la proportion de prédictions correctes.

3. Résultats d'Évaluation

Voici un aperçu des accuracy obtenues pour chaque modèle :

Modèles	Accuracy
Régression Logistique	0.88804
Random Forest	0.86377
Naive Bayes	0.80260
KNN	0.88994
SVC	**0.89735**

Ces résultats ont mis en évidence les forces et faiblesses de chaque modèle, nous permettant d'affiner notre choix en fonction de l'objectif de précision et de performance.



Optimisation du Modèle avec Voting Classifier

L'optimisation de notre modèle est une étape cruciale pour améliorer les performances globales de détection. Pour cela, nous avons utilisé une technique avancée de Machine Learning appelée **Voting Classifier**. Cette méthode combine les prédictions de différents modèles pour obtenir une décision finale qui est souvent plus fiable que celle de chaque modèle individuel.

Pour améliorer notre modèle de détection d'intrusion, nous avons mis en œuvre une stratégie d'optimisation en deux phases :

- **Première Phase :** Cinq Algorithmes

  Nous avons commencé par définir cinq modèles différents : Régression logistique, Forêt aléatoire, Naive Bayes, K plus proches voisins (KNN), et Machine à vecteurs de support (SVC).

  Chaque modèle a été ajusté avec des hyperparamètres spécifiques et le Voting Classifier a été entraîné et testé pour sa performance. Cependant, **l'accuracy obtenue de 0.88285**, bien que respectable, a été jugée perfectible.

- **Deuxième Phase :** Amélioration
  - **Élimination des Modèles Sous-Performants :** Les modèles Naive Bayes et Random Forest ont été retirés du Voting Classifier.
  - **Concentration sur les Modèles Performants :** L'accent a été mis sur les trois modèles les plus prometteurs – Régression Logistique, KNN et SVC.
  - **Réentrainement et Réévaluation :** Le Voting Classifier révisé a été réentrainé et les prédictions ont été recalculées, ce qui a abouti à **une meilleure accuracy de 0.89851**

Déploiement du Modèle

Le déploiement du modèle finalisé est la dernière étape de notre projet, qui rend notre système de détection d'intrusion accessible et utilisable.

- **Création d'une Application Web**

Nous avons développé une application web avec Streamlit. Cette application permet aux utilisateurs d'interagir avec notre modèle de détection d'intrusion de manière intuitive. Ils peuvent saisir manuellement les valeurs des 11 caractéristiques sélectionnées ou télécharger un fichier CSV pour que le modèle fasse des prédictions.

- **Accessibilité et Utilisation**

L'application Streamlit est hébergée en ligne, permettant à quiconque avec une connexion Internet d'utiliser notre modèle de détection d'intrusion réseau. Cela démontre non seulement l'efficacité de notre modèle mais aussi sa facilité d'intégration dans des environnements réels.

Conclusion

À travers la réalisation de ce projet, nous avons plongé dans les profondeurs du Machine Learning appliqué à la détection d'intrusion réseau, un domaine à la fois fascinant et complexe. Nous avons navigué à travers les étapes critiques du processus d'apprentissage automatique, depuis la collecte des données jusqu'à l'optimisation des algorithmes. Notre voyage a été ponctué par la découverte et l'application de techniques de prétraitement, l'évaluation de modèles de classification variés, et finalement, l'implémentation d'une stratégie d'optimisation qui a prouvé son efficacité.

Le projet s'est concrétisé par le déploiement d'une application web intuitive, rendant notre modèle de détection d'intrusion accessible et facile à tester. Cette plateforme, bien qu'éducative, ouvre des perspectives sur comment les modèles de Machine Learning peuvent être intégrés dans des environnements réels pour renforcer la sécurité informatique.

Cette expérience a été une opportunité enrichissante de mettre en pratique nos connaissances théoriques. Nous avons été confrontés à des défis réalistes et avons appris à les surmonter avec persévérance et créativité. Plus qu'un projet académique, c'était une simulation de ce que pourrait être notre future implication dans l'industrie de la cybersécurité.

En somme, notre projet " **Système de Détection d'Intrusion Réseaux** " a été une pierre angulaire dans notre parcours éducatif, nous dotant d'une base solide pour notre future carrière professionnelle et nos recherches académiques.