

Table des matières

Question 1 :	2
Origine	2
But	2
Différences vs KDD'99	2
Fichiers fournis	2
Question 2	3
Question 3	4
Question 4	5
Question 5	5
Question 6	6
Question 7	6
Question 8	7
Question 9	8
Question 10	9
Bibliographie	11

Question 1 :

Origine

C'est un fork de la base de données KDD'99.

But

Fournir un jeu de données permettant de développer, tester et comparer des systèmes de détection d'intrusion.

Différences vs KDD'99

- Le jeu de données proposé est une version améliorée de KDD'99, corrigée des redondances présentes dans les ensembles d'apprentissage et de test (htt).
- Rééquilibrage des catégories, certaines étant initialement surreprésentées (htt1).
- Réduction de la taille du jeu de données afin que les études soient comparables durant les tests (htt1).

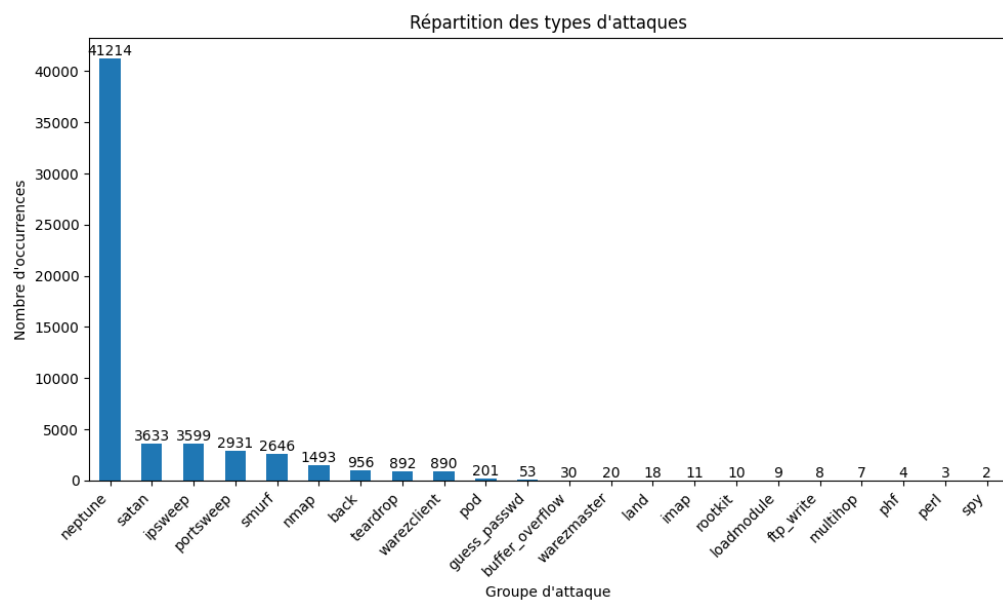
Fichiers fournis

- Deux formats disponibles (htt1) :
 - TXT (en réalité des fichiers au format CSV)
 - ARFF (extension destinée au logiciel de machine learning Weka)
- Deux jeux de données (htt1) (htt3) :
 - L'intégralité des données
 - Un sous-ensemble de 20 % des données

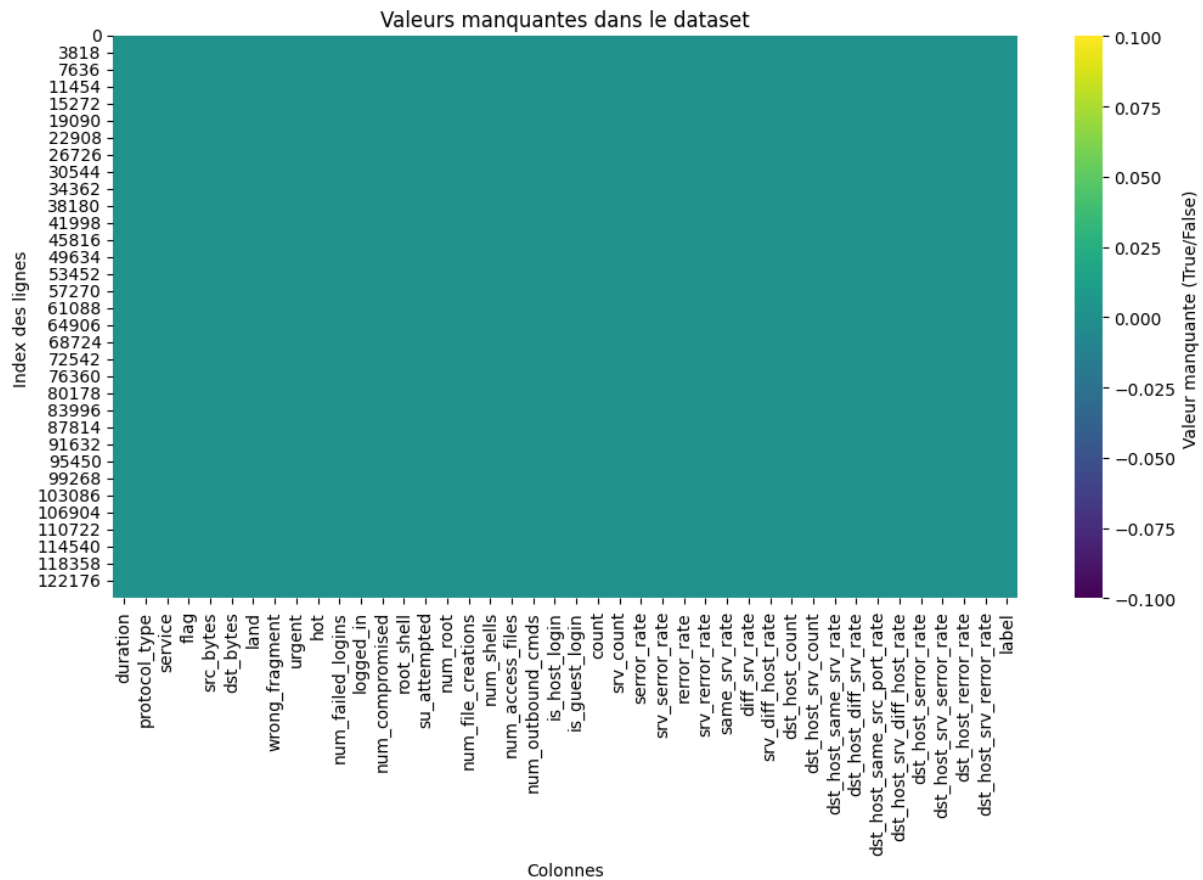
Question 2

(htt2)

Attacks in Dataset	Attack Type (37)
DOS	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
Probe	Satan, Ipsweep, Nmap, PortswEEP, Mscan, Sa int
R2L	Guess_password, Ftp_write, Imap, Phf, Multi hop, WarezmasteR, Xlock, Xsnoop, Snpmgue ss, Snpmpgetattack, Http tunnel, Sendmail, Named
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps



Question 3



Il n'y a pas de valeurs manquantes, comme le montre cette carte de chaleur. J'ai utilisé cette fonction, qui permet de vérifier la présence de valeurs manquantes dans l'ensemble du jeu de données :

```
# Calcul des valeurs manquantes dans le dataset d'entraînement  
missing_train = dataset.isnull().sum()
```

Voici une preuve que la fonction `isnull()` permet de détecter les valeurs NaN :

```
isnull().sum()  
(method) def isnull() -> DataFrame  
DataFrame.isnull is an alias for DataFrame.isna.  
Detect missing values.  
Return a boolean same-sized object indicating if the values are NA. NA values, such as None or numpy.NaN, gets mapped to True values.  
Everything else gets mapped to False values. Characters such as empty strings '' or numpy.inf are not considered NA values (unless you set  
pandas.options.mode.use_inf_as_na = True).
```

Question 4

Cela a été fait dans la question 3, car j'ai trouvé plus pertinent d'avoir les colonnes déjà correctement nommées si, plus tard, j'avais besoin d'en appeler une. De plus, il n'est pas possible d'afficher l'ensemble du jeu de données avec la fonction `display()`.

```
# Question 4 - Renommer les colonnes du dataset

## Liste des nouveaux noms de colonnes
col_names = ["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", "serror_rate", "serror": Unknown word,
             "srv_serror_rate", "rerror_rate", "srv_rerror_rate", "same_srv_rate", "serror": Unknown word,
             "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_serror_rate", "dst_host_srv_serror_rate", "serror": Unknown word,
             "dst_host_rerror_rate", "dst_host_srv_rerror_rate", "Label"] "error": Unknown word.

## Modification des noms de colonnes
dataset = pd.read_csv(dataset, header=None, names=col_names)

## Affichage du dataset d'entraînement avec les nouveaux noms de colonnes
print("Dataset avec les nouveaux noms de colonnes :")
display(dataset)
```

✓ 0.1s Python

Dataset avec les nouveaux noms de colonnes :

	duration	protocol_type	service	flag	src_bytes	dst_bytes	land	wrong_fragment	urgent	hot	...	dst_host_srv_count	dst_host_same_srv_rate	dst_host_diff_srv_rate	dst_host
0	0	tcp	ftp_data	SF	491	0	0	0	0	0	...	25	0.17	0.03	
1	0	udp	other	SF	146	0	0	0	0	0	...	1	0.00	0.60	
2	0	tcp	private	S0	0	0	0	0	0	0	...	26	0.10	0.05	
3	0	tcp	http	SF	232	8153	0	0	0	0	...	255	1.00	0.00	
4	0	tcp	http	SF	199	420	0	0	0	0	...	255	1.00	0.00	
...
125968	0	tcp	private	S0	0	0	0	0	0	0	...	25	0.10	0.06	
125969	8	udp	private	SF	105	145	0	0	0	0	...	244	0.96	0.01	
125970	0	tcp	smtp	SF	2231	384	0	0	0	0	...	30	0.12	0.06	
125971	0	tcp	klogin	S0	0	0	0	0	0	0	...	8	0.03	0.05	
125972	0	tcp	ftp_data	SF	151	0	0	0	0	0	...	77	0.30	0.03	

125973 rows x 42 columns

Question 5

```
# Question 5 - Vérifier le nombre de lignes dupliquées dans le dataset d'entraînement

print("Nombre total de lignes dupliquées dans le dataset d'entraînement :", dataset.duplicated().sum())
```

✓ 0.2s

Nombre total de lignes dupliquées dans le dataset d'entraînement : 0

Il n'y a pas de doublons, ce qui est logique, car nous travaillons sur un jeu de données qui a été retravaillé pour les éliminer.

Question 6

```
# Question 6 - Suppression des doublons

print("Avant suppression, nombre de lignes dans le dataset d'entraînement :", len(dataset))
dataset.drop_duplicates(inplace=True) # Supprimer les doublons
print("Après suppression, nombre de lignes dans le dataset d'entraînement :", len(dataset))

✓ 0.0s
```

```
Avant suppression, nombre de lignes dans le dataset d'entraînement : 125973
Après suppression, nombre de lignes dans le dataset d'entraînement : 125973
```

Il n'y a pas de doublons. J'ai ajouté l'option *inplace=True* pour que la modification s'applique directement au jeu de données.

Question 7

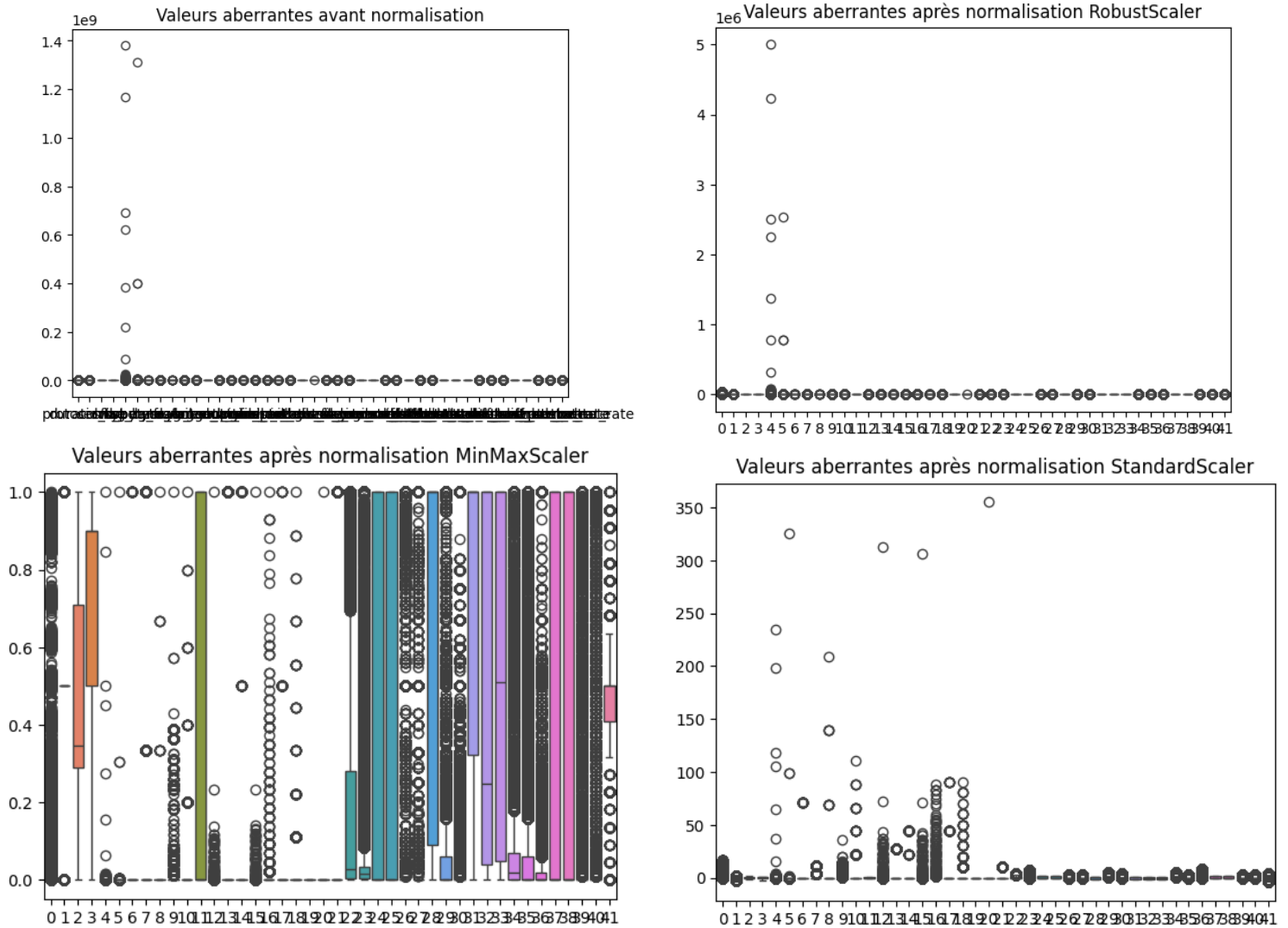
J'ai utilisé `LabelEncoder` plutôt que `OneHotEncoder` parce que c'est plus simple à implémenter. De plus, certaines catégories, comme `service`, contiennent de nombreuses valeurs différentes, ce qui pourrait entraîner du surapprentissage et ralentir le modèle.

Cependant, comme ce sont des variables nominales, il serait plus approprié d'utiliser `OneHotEncoder`.

Question 8

J'ai testé trois méthodes : RobustScaler, MinMaxScaler et StandardScaler.

Pour chacune de ces méthodes, j'ai réalisé un nuage de points avant et après leur application :



Au final, j'ai choisi le StandardScaler, car c'est le modèle le mieux adapté pour gérer les valeurs extrêmes. Comme le montre le graphique, les valeurs sont mieux contenues.

Question 9

Au départ, j'ai mélangé les données en utilisant un *random_state* de 42, puis j'ai calculé la taille totale du jeu de données afin d'en extraire 80 %. Après cette opération, j'ai obtenu 80 % des valeurs pour l'ensemble principal et les 20 % restants pour l'autre portion du jeu de données.

J'ai ensuite enregistré le jeu de données au format CSV, afin qu'il puisse être téléchargé et réutilisé par d'autres pour l'entraînement de modèles. Pour moi, le CSV est le format le plus approprié, car il est largement répandu, même en dehors du contexte du machine learning, et n'est pas spécifique à un logiciel particulier.

```
# Question 9 - Découpage en ensembles (train / test)

dataset = dataset.sample(frac=1, random_state=42) # Mélanger les données (utile si les données avaient un c

dataset_size = len(dataset) # Taille totale du dataset
train_size = int(0.8 * dataset_size) # 80% pour l'entraînement, 20% pour le test

train_data_final = dataset[:train_size] # Jeu d'entraînement final (80%)
test_data_final = dataset[train_size:] # Jeu de test final (20%)

## Affichage du nombre de lignes dans chaque jeu
print("Nombre de lignes dans le jeu d'entraînement final :", len(train_data_final))
print("Nombre de lignes dans le jeu de test final :", len(test_data_final))

## Affichage des jeux finaux
print("Jeu d'entraînement final :")
display(train_data_final)
print("Jeu de test final :")
display(test_data_final)

## Sauvegarde des jeux finaux dans des fichiers CSV (optionnel)
train_data_final.to_csv("data/Dataset_pre_traiter/NSL_KDD_train_final.csv", index=False)
test_data_final.to_csv("data/Dataset_pre_traiter/NSL_KDD_test_final.csv", index=False)
```

✓ 3.2s

```
Nombre de lignes dans le jeu d'entraînement final : 100778
Nombre de lignes dans le jeu de test final : 25195
Jeu d'entraînement final :
```


Question 10

Pour cette question, j'ai d'abord repris le jeu de données que j'avais copié auparavant, afin de ne pas être affecté par les modifications précédentes qui m'auraient empêché de refaire un prétraitement.

Ensuite, j'ai regroupé les sous-catégories en grandes catégories sous forme de listes et construit des dictionnaires correspondants.

```
# Listes fournies stockées dans des variables
dos = [
    "back", "land", "neptune", "pod", "smurf", "teardrop", "mailbomb",
    "processtable", "udpstorm", "apache2", "worm"    "processtable": Unknown word.
]

probe = [
    "satan", "ipsweep", "nmap", "portsweep", "mscan", "saint"    "satan": Unknown word.
]

r2l = [
    "guess_passwd", "ftp_write", "imap", "phf", "multihop", "warezmaster",    "multihop": Unknown word.
    "xlock", "xsnoop", "snmpguess", "snmpgetattack", "httptunnel", "sendmail", "named"    "xsnoop": Unknown word.
]

u2r = [
    "buffer_overflow", "loadmodule", "rootkit", "perl", "sqlattack", "xterm", "ps"    "loadmodule": Unknown word.
]

normal = ["normal"]

# Construire un dictionnaire de correspondance (label -> grand groupe)
mapping = {}
for name in dos:
    mapping[name] = 'DOS'
for name in probe:
    mapping[name] = 'Probe'
for name in r2l:
    mapping[name] = 'R2L'
for name in u2r:
    mapping[name] = 'U2R'
for name in normal:
    mapping[name] = 'Normal'
```

Une fois cela fait, j'ai copié la colonne qui m'intéressait et appliqué les regroupements par grande catégorie. J'ai prévu une gestion d'erreurs pour les éléments qui ne figuraient pas dans mes listes, en les plaçant dans la catégorie « Other ».

Enfin, j'ai vérifié la sortie pour m'assurer qu'elle était correcte et qu'aucune valeur n'était restée hors des catégories définies.

En faisant cela, j'ai remarqué que « warezclient » n'était finalement pas dans la liste, ce que j'ai corrigé. En revanche, « spy » n'y figurait pas non plus ; je l'ai laissé tel quel, car je n'ai pas trouvé de catégorie appropriée et peu de lignes étaient concernées (2 lignes).

Par la suite, j'ai ajouté la colonne au jeu de données, puis j'ai effectué la séparation en plusieurs jeux de données en fonction des valeurs uniques présentes dans cette colonne.

```
## Lecture / extraction de la colonne 'label'
labels_raw = train_data_q10['label'].astype(str) # S'assurer que la colonne est du texte

## Mettre 'Other' pour les labels inconnus (non présents dans les listes fournies)
labels_grouped = labels_raw.apply(lambda x: mapping.get(x, 'Other'))

## Afficher les comptes par grand groupe
print("Comptes par grand groupe :")
print(labels_grouped.value_counts())
print("\n")

## Ajouter la nouvelle colonne au jeu d'entraînement et afficher le résultat
train_data_q10['attack_family'] = labels_grouped
print("Jeu d'entraînement avec la nouvelle colonne 'attack_family' :")
display(train_data_q10)
print("\n")
print("Voir les lignes où il y a la valeur 'Other' dans la colonne 'attack_family' :")
display(train_data_q10[train_data_q10['attack_family'] == 'Other'])
print("\n")

## Découpage par famille d'attaque en jeux distincts

dataset_DOS = train_data_q10[train_data_q10['attack_family'] == 'DOS']
dataset_Probe = train_data_q10[train_data_q10['attack_family'] == 'Probe']
dataset_R2L = train_data_q10[train_data_q10['attack_family'] == 'R2L']
dataset_U2R = train_data_q10[train_data_q10['attack_family'] == 'U2R']
dataset_Normal = train_data_q10[train_data_q10['attack_family'] == 'Normal']

## Liste de tous les jeux pour des boucles ultérieures
list_datasets = [dataset_DOS, dataset_Probe, dataset_R2L, dataset_U2R, dataset_Normal]
list_names = ['DOS', 'Probe', 'R2L', 'U2R', 'Normal']

## Vérification des jeux créés (boucle où "i" est l'index et "dataset" est la variable concrète)
for i, dataset in enumerate(list_datasets):
    print(f"Affichage du dataset {list_names[i]} : ")
    display(dataset)
    print("\n")
```

Pour finir, il ne me restait plus qu'à faire des boucles for sur la liste de jeux de données que j'avais créée, en réutilisant le code initial avec quelques ajustements.

Bibliographie

(s.d.). Récupéré sur

<https://www.sciencedirect.com/science/article/pii/S1877050920308334/pdf>

(s.d.). Récupéré sur <https://www.unb.ca/cic/datasets/nsl.html>

(s.d.). Récupéré sur <https://www.ijert.org/a-detailed-analysis-on-nsl-kdd-dataset-using-various-machine-learning-techniques-for-intrusion-detection-2>

(s.d.). Récupéré sur <https://www.kaggle.com/datasets/hassan06/nslkdd>