# Projet 1 : Pre-traitement des donnees

**Base de donnee: NSLKDD :** ***https://www.kaggle.com/datasets/hassan06/nslkdd***

**Barème total: 30 points**

***Questions & points (à traiter dans l'ordre, avec captures & code)***

1. **Qu'est-ce que la base NSL-KDD ?** (origine, but, différences vs KDD'99, fichiers fournis) **(2 pts)**
2. **Quelles sont les attaques** présentes dans NSL-KDD ? Dresser la **liste** des types d'attaques et un **graphe** de distribution (barres) **(4 pts)**
3. **Valeurs manquantes :** NSL-KDD contient-il des **valeurs nulles** ? **Justifiez** par une **représentation graphique** (ex. carte de chaleur des NaN, histogramme des valeurs manquantes par colonne). **(4 pts)**
4. **Nommer les colonnes** exactement comme suit (**vérifier l'ordre !**) **(2 pts)**

par :

```
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",
    "srv_serror_rate","rerror_rate","srv_rerror_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_serror_rate","dst_host_srv_serror_rate",
    "dst_host_rerror_rate","dst_host_srv_rerror_rate","label"]
```

5. **Doublons : Y a-t-il des lignes dupliquées ? Combien ? (montrer la méthode) (3 pts)**

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

6. **Suppression des doublons :** retirer proprement les duplicates et **vérifier** la nouvelle taille du dataset **(2 pts)**
7. **Encodage des données :** encoder les variables **catégorielles** (justifier le **choix d'algorithme** : LabelEncoder vs One-Hot, ColumnTransformer, etc.) **(4 pts)**
8. **Normalisation / Mise à l'échelle :** appliquer une stratégie adaptée (StandardScaler / MinMaxScaler / RobustScaler) et **motiver le choix (3 pts)**
9. **Découpage en ensembles :** produire un **train / test** (stratifié sur *label*), fixer un **random_state**, afficher les **tailles** et la **répartition des classes (3 pts)**
10. **Regroupement par familles d'attaques** : créer une colonne **attack_family** (DoS, Probe, R2L, U2R, Normal), puis **rejouer** les étapes 7–9 **pour chaque famille** (encodage & split pertinents). **(3 pts)**

***Livrables (Type de rendu)***

1. **Rapport (PDF, 6–10 pages)**
   - Répondre à chaque question dans l'ordre, avec figures (graphiques), tableaux et explications.
   - Inclure le code Python dans le rapport (extraits essentiels) ET en annexe (fichiers .py ou .ipynb).
   - Commenter le code : chaque étape expliquée (pourquoi / comment).
2. **Dossier de code joint (zip)**
   - Notebook .ipynb ou script .py proprement structurés.
   - Un README expliquant comment exécuter (venv/conda, pip install -r requirements.txt, chemins des CSV).
3. **Reproductibilité**
   - Fixer les seeds (random_state).
   - Indiquer les versions (Python, pandas, scikit-learn, matplotlib…).

**Nommage des fichiers**

- Rapport : Projet1_Pretraitement_NomPrenom.pdf
- Code : Projet1_Pretraitement_NomPrenom.ipynb (ou .py)
- Archive finale : Projet1_Pretraitement_NomPrenom.zip

**Grille d'évaluation (synthèse)**

- Fond (exactitude, rigueur, justifications) …… 18 pts
- Qualité du code (lisible, commenté, structuré, propre) …… 8 pts
- Présentation (figures, clarté, mise en page) …… 4 pts
  Total : 30 pts