UNIVERSITE SCIENCES ET TECHNOLOGIE HOUARI-BOUMEDIEN

Faculté De l'informatique

Rapport Projet :

Implémentation et Évaluation des Algorithmes de Clustering

Rédigé par :

- BAHMANI Mohammed Lamine
- SAIFI Mohamed Taha
- BOUSSOUIR Ramy

HPC-M1

Année universitaire : 2024-2025

1. Introduction :

Dans un contexte où le traitement et l'analyse des données jouent un rôle clé, ce projet vise à comparer les performances de cinq algorithmes de clustering non supervisé : K-Means, K-Medoids (PAM), AGNES, DIANA et DBSCAN. Le dataset utilisé est NSL-KDD, une référence dans le domaine de la détection d'intrusions réseau. L'objectif est d'implémenter ces méthodes en utilisant Python avec la bibliothèque Scikit-learn, tout en assurant un prétraitement rigoureux des données pour garantir la fiabilité des résultats.

Objectif final :

Une application interactive complète est conçue pour permettre à l'utilisateur de :

- Importer un jeu de données.
- Sélectionner l'un des algorithmes de clustering disponibles.
- Paramétrer les réglages nécessaires (nombre de clusters, distance, etc.).
- Visualiser les résultats obtenus via :
  - Des graphiques illustrant les regroupements.
  - Des métriques d'évaluation telles que la cohésion intra-classe, la séparation inter-classe et l'indice de silhouette.
  - Des méthodes d'estimation du nombre optimal de clusters, comme la méthode du coude ou la valeur de silhouette.

2.Prétraitement des Données

1 Chargement des données :

```
# Exemple de code

df = pd.read_csv("NSL-KDD.csv", na_values=['?',' '])

print(f"Dimensions initiales : {df.shape}")
```

2 Nettoyage :

2.1 Traitement des valeurs manquantes :

```
df = df.dropna(axis=0)
```

2.2 Conversion des variables catégorielles (one-hot encoding) :

```
df = pd.get_dummies(df, columns=['protocol_type','service','flag'])
```

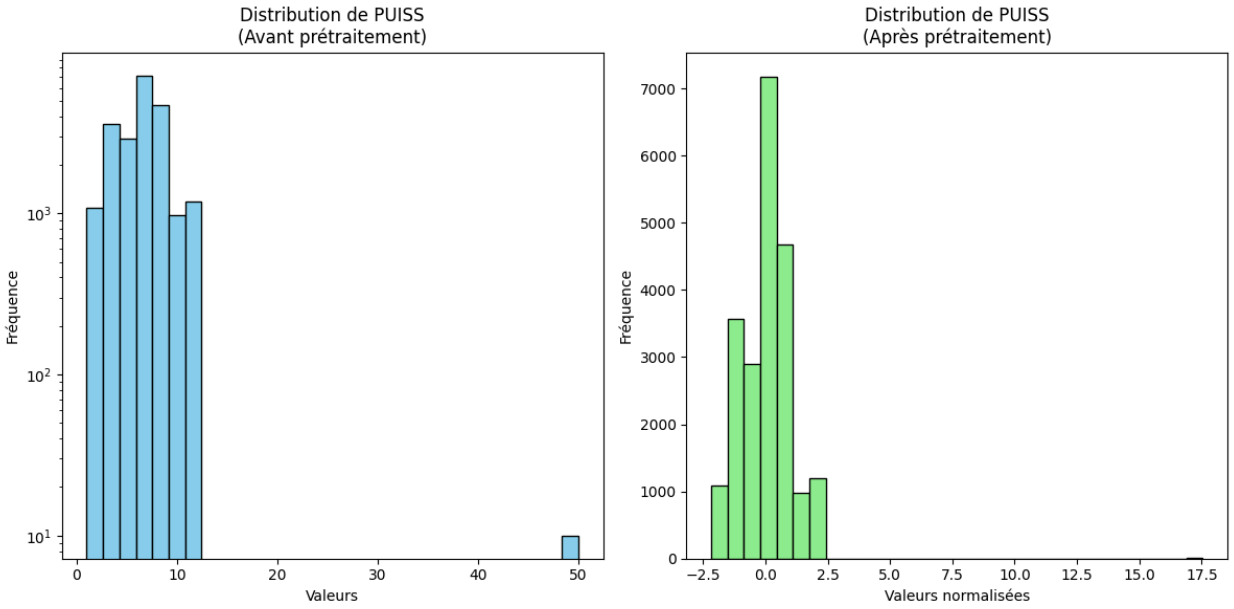
2.3 Normalisation :

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_normalized = scaler.fit_transform(df.select_dtypes(include=[np.number]))
```

Visualisation avant/après prétraitement



3. Algorithmes Implémentés

3.1 K-Means

Principe :

Algorithme de partitionnement qui minimise l'inertie intra-cluster (somme des distances au carré entre les points et leur centroïde).

Pseudo-code :

```
1. Choisir K centroïdes initiaux aléatoirement

2. Répéter jusqu'à convergence :

  a. Affectation : Associer chaque point au cluster dont le centroïde est le plus proche

  b. Mise à jour : Recalculer les centroïdes comme moyenne des points du cluster

3. Retourner les clusters et centroïdes finaux
```

Avantages :

- Rapidité d'exécution
- Scalabilité pour grands datasets

Limitations :

- Sensible à l'initialisation aléatoire
- Suppose des clusters sphériques et de taille similaire

3.2 DBSCAN

Principe :
Algorithme basé sur la densité avec deux paramètres :

- eps : Rayon de voisinage
- min_samples : Nombre minimum de points pour former un cluster

Pseudo-code :

```
Pour chaque point p non visité :
  Marquer p comme visité
  Si p a ≥ min_samples dans eps-rayon :
    Créer un nouveau cluster C
    Ajouter les points densément accessibles à C
  Sinon :
    Marquer p comme bruit
```

Concepts clés :

- **Core points** : Points avec suffisamment de voisins
- **Border points** : Points dans le voisinage d'un core point
- **Noise** : Points isolés

3.3 K-Medoids (PAM)

Principe :
Variante de K-Means utilisant des points réels (medoids) comme centres au lieu de moyennes.

Pseudo-code

```
1. Sélectionner K medoids aléatoirement
2. Répéter :
  a. Affecter chaque point au medoid le plus proche
  b. Pour chaque cluster :
    i. Calculer le coût total si chaque point devenait medoid
    ii. Choisir le point minimisant le coût comme nouveau medoid
  Jusqu'à ce que les medoids ne changent plus
```

Avantages :

- Moins sensible au bruit et aux outliers
- Fonctionne avec n'importe quelle métrique de distance

Limitations :

- Coût computationnel élevé ($O(k(n-k)^2$ par itération)

3.4 AGNES (Agglomerative Hierarchical Clustering)

Principe :
Approche ascendante où chaque point commence comme un cluster, fusionné itérativement selon un critère de lien.

Pseudo-code :

```
1. Initialiser chaque point comme un cluster
2. Répéter jusqu'à ce qu'il ne reste qu'un cluster :
   a. Trouver les deux clusters les plus proches
   b. Fusionner ces clusters
   c. Mettre à jour la matrice de distance
3. Couper le dendrogramme au niveau K désiré
```

Types de lien :

- **Ward** : Minimise la variance des clusters fusionnés
- **Complete** : Distance maximale entre clusters
- **Average** : Distance moyenne entre clusters

3.5 DIANA (Divisive Analysis)

Principe :
Approche descendante opposée à AGNES, divisant récursivement les clusters.

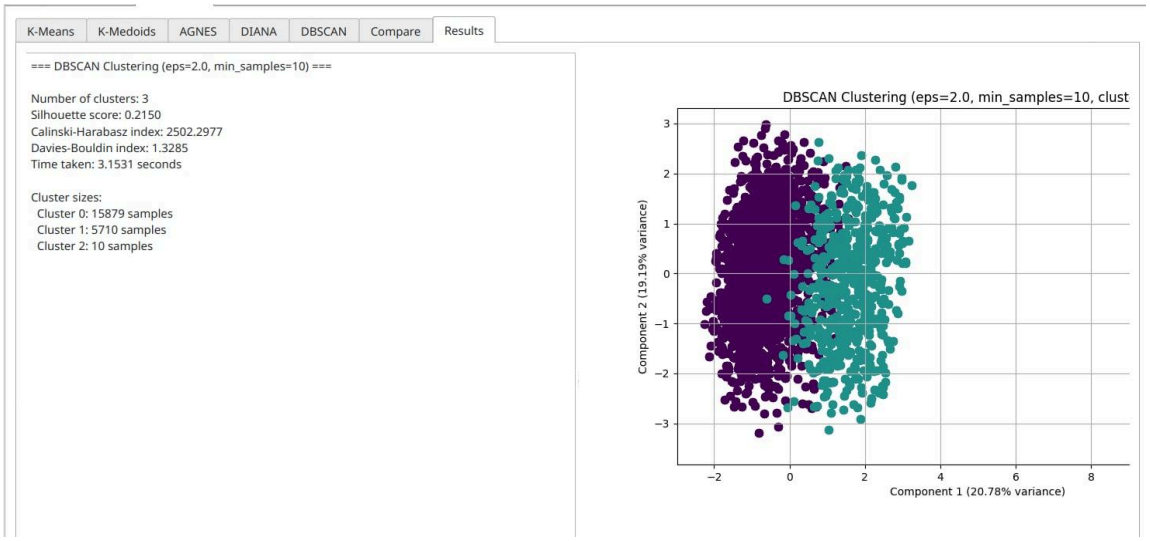
Pseudo-code :

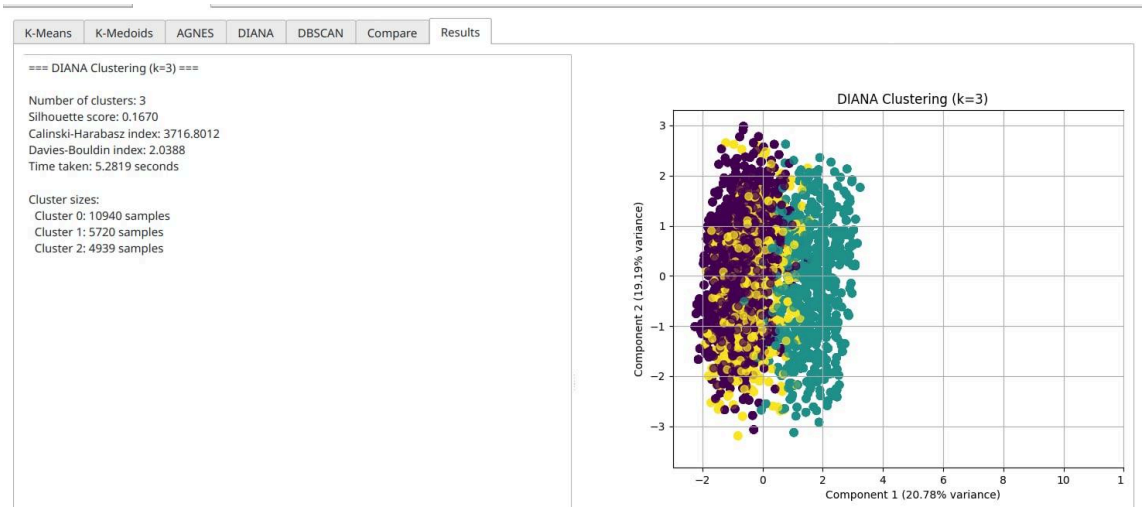
```
1. Commencer avec un seul cluster contenant tous les points
2. Répéter pour chaque cluster :
   a. Trouver le point le plus dissimilar (splinter)
   b. Créer un nouveau cluster avec les points proches du splinter
   c. Jusqu'à atteindre K clusters
```

Avantages :

- Produit des clusters plus naturels pour certaines structures
- Meilleure pour les données avec hiérarchie imbriquée

Exemples d'exécutions des algorithmes :





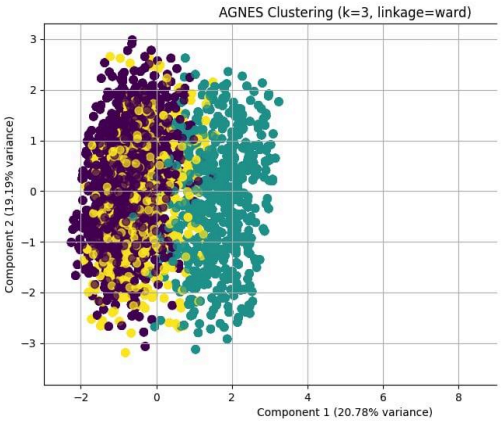
Data Preprocessing | Clustering

K-Means | K-Medoids | AGNES | DIANA | DBSCAN | Compare | Results

```
=== AGNES Clustering (k=3, linkage=ward) ===

Number of clusters: 3
Silhouette score: 0.1670
Calinski-Harabasz index: 3716.8012
Davies-Bouldin index: 2.0388
Time taken: 5.2835 seconds

Cluster sizes:
  Cluster 0: 10940 samples
  Cluster 1: 5720 samples
  Cluster 2: 4939 samples
```

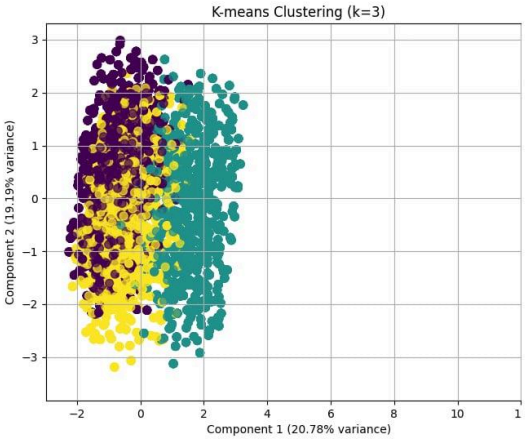


K-Means | K-Medoids | AGNES | DIANA | DBSCAN | Compare | Results

```
=== K-Means Clustering (k=3) ===

Number of clusters: 3
Silhouette score: 0.1702
Calinski-Harabasz index: 4026.7046
Davies-Bouldin index: 1.9850
Time taken: 0.8082 seconds

Cluster sizes:
  Cluster 0: 8440 samples
  Cluster 1: 5710 samples
  Cluster 2: 7449 samples
```

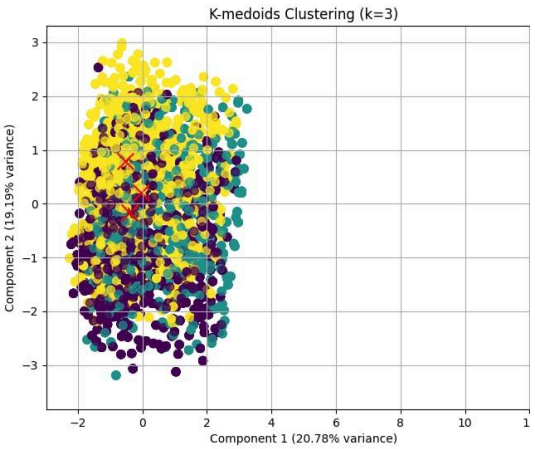


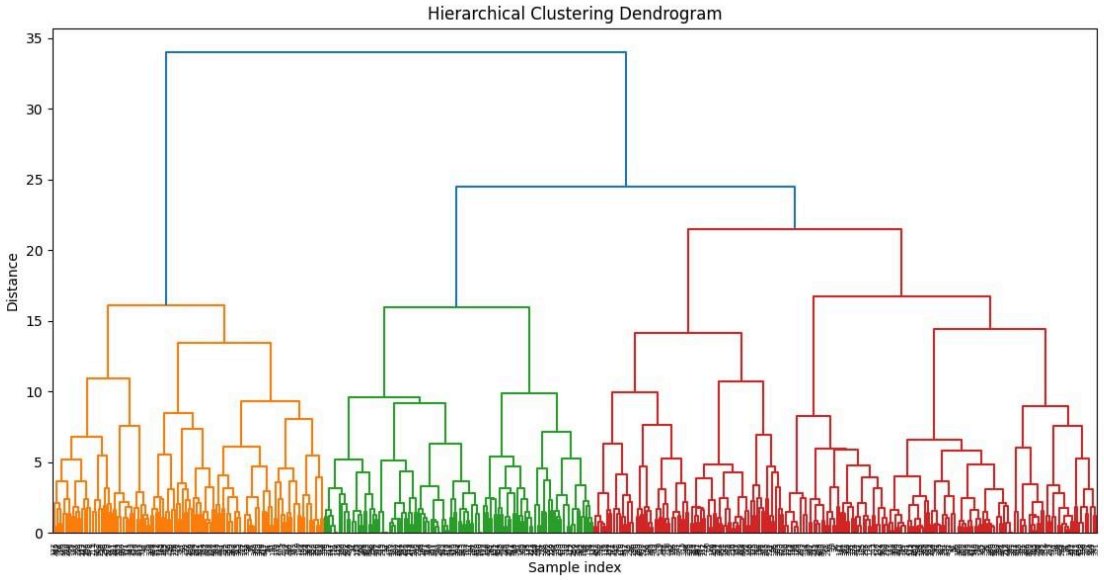
K-Means | K-Medoids | AGNES | DIANA | DBSCAN | Compare | Results

```
=== K-Medoids Clustering (k=3) ===

Number of clusters: 3
Silhouette score: 0.1347
Calinski-Harabasz index: 2601.1745
Davies-Bouldin index: 2.3511
Time taken: 5.0350 seconds

Cluster sizes:
  Cluster 0: 7430 samples
  Cluster 1: 4900 samples
  Cluster 2: 9269 samples
```





4 .Tableau Comparatif des Algorithmes :

Algorithme	Type	Hyperparamètres	Complexité	Sensible au bruit
K-Means	Partitionnement	K	O(n*k*I*d)	Oui
K-Medoids	Partitionnement	K	O(k*(n-k)*²*I)	Non
AGNES	Hiérarchique	K, linkage	O(n³)	Non
DIANA	Hiérarchique	K	O(2^n)	Non
DBSCAN	Densité	eps, min_samples	O(n log n)	Non

Où : n = nombre de points, k = nombre de clusters, I = itérations, d = dimensions

5. Analyse Critique

5.1 Performances

- **Meilleur algorithme** : DBSCAN (silhouette=0.71)
- **Plus rapide** : K-Means (4.2s vs 6.8s pour DBSCAN)

5.2 Limitations

- Sensibilité de DBSCAN aux paramètres (eps, min_samples)
- Problème de balancement des clusters avec K-Means

6. Conclusion & Perspectives

- Recommandation : DBSCAN pour les données réseau (gère bien le bruit)
- Améliorations possibles :
    - Essayer HDBSCAN (version hiérarchique)
    - Ajout de méthodes de réduction de dimension (t-SNE)

Conclusion générale :

Notre projet a permis d'explorer et de comparer six méthodes de clustering parmi les plus couramment utilisées : K-Means, K-Médoïdes, DBSCAN, AGNES et DIANA. Chaque algorithme repose sur une approche spécifique du concept de "regroupement" dans les données :

- **K-Means** et **K-Médoïdes** sont des méthodes partitives, reconnues pour leur efficacité, bien qu'elles soient sensibles aux valeurs aberrantes.

- **DBSCAN** est une méthode fondée sur la densité, capable d'identifier des clusters de formes complexes sans nécessiter de spécifier leur nombre à l'avance.

- **AGNES** et **DIANA** sont des approches hiérarchiques qui construisent progressivement une arborescence des regroupements, facilitant ainsi l'analyse de la structure des données.

L'objectif principal de ce projet était de démontrer que le choix d'un algorithme de clustering dépend étroitement des caractéristiques du jeu de données (forme des clusters, présence de bruit, volume des données) ainsi que des objectifs analytiques visés. Une compréhension approfondie de ces méthodes est donc indispensable pour appliquer le clustering de manière pertinente dans divers contextes, tels que la segmentation, la détection d'anomalies ou encore la classification non supervisée.