Practical Work 3: Clustering, Unsupervised Analysis, and MLOps

Louis Fippo Fitime, Claude Tinku, Kerolle Sonfack
Computer Engineering Department, ENSPY

October 11, 2025

TP Objective

- Implement and compare the three major families of Clustering algorithms: **K-Means**, **GMM/EM**, and **DBSCAN**.
- Master unsupervised evaluation metrics (**Inertia**, **Silhouette Coefficient**).
- Study the role of **Clustering** in **Customer Knowledge (Sales)** and the **Gaming Industry**.
- Consolidate the **MLOps** phase by tracking clustering configurations (number of clusters, hyperparameters) via **MLflow**.

---

1 Phase I: Representative-Based Clustering (K-Means)

1.1 Use Case 1: Customer Knowledge and Segmentation (Sales)

We will segment customers based on their annual spending and spending score. The objective is to find homogeneous groups for targeted marketing campaigns.

- **Task 1.1: Data Preparation:** Load the customer dataset (simulation), reduce the dimension to two variables for visualization (`Annual`$_Incomeand$

```
Code Python 1.1: Data Preparation (Simulation)
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt

Simulated data for customer segmentation (2D for visualization)
```

```
columns: CustomerID, Genre, Age, Annual_Income (k$), Spending_Score (1-100)
X_sim = np.array([
[15, 39], [17, 35], [18, 92], [20, 95], [23, 75], [78, 7], [75, 12],
[77, 20], [79, 3], [40, 40], [42, 52], [45, 35], [55, 60], [58, 58]
])

Data Standardization
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_sim)

print("Standardized data (excerpt):")
print(X_scaled[:5])
```

1.2 K-Means Optimization and Evaluation

- **Task 1.2: Choosing K (Elbow Method):** Apply the K-Means algorithm for K ranging from 1 to 10. Calculate and visualize the inertia (sum of intra-cluster squares) as a function of K. Deduce the optimal number of clusters.
- **Task 1.3: Silhouette Evaluation:** Calculate the **Silhouette Coefficient** for the chosen optimal K. Briefly recall the meaning of this metric.

```
Code Python 1.2 & 1.3: K Optimization
from sklearn.metrics import silhouette_score

inertia = []
K_range = range(1, 11)

for k in K_range:

TO BE COMPLETED: Train KMeans and store the inertia in the 'inertia' list
Elbow Method Visualization
plt.figure(figsize=(8, 5))
plt.plot(K_range, inertia, 'bo-')
plt.title("Elbow Method for Inertia")
plt.xlabel("Number of Clusters (K)")
plt.ylabel("Inertia")
plt.show()

TO BE COMPLETED: Choose K_optimal and calculate the silhouette score
K_optimal = ...
kmeans_final = KMeans(n_clusters=K_optimal, random_state=42)
labels = kmeans_final.fit_predict(X_scaled)
silhouette_avg = silhouette_score(X_scaled, labels)

print(f"\nSilhouette Coefficient for K={K_optimal}: {silhouette_avg:.4f}")
```

2 Phase II: Advanced Clustering and Critical Comparison

2.1 Use Case 2: User Behavior Segmentation (Gaming)

Real video game user data is often noisy and non-globular in shape. We will use synthetic data to compare models.

- **Task 2.1: Probabilistic Clustering (GMM/EM):** Explain the role of the **Expectation-Maximization (EM) Algorithm** in fitting a **Mixture of Gaussian Models (GMM)**. Train a GMM and visually compare the results with K-Means on non-globular data (e.g., two moons).

```
Code Python 2.1: GMM and Comparison
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_moons # Non-globular data

Generating 'moons' data
X_moons, y_moons = make_moons(n_samples=200, noise=0.05, random_state=42)
X_moons_scaled = StandardScaler().fit_transform(X_moons)

K-Means on Moons
kmeans_moons = KMeans(n_clusters=2, random_state=42)
labels_kmeans = kmeans_moons.fit_predict(X_moons_scaled)

GMM on Moons
gmm = GaussianMixture(n_components=2, random_state=42)
labels_gmm = gmm.fit_predict(X_moons_scaled)

TO BE COMPLETED: Visualize the results of the two algorithms
Use plt.scatter(X_moons_scaled[:, 0], X_moons_scaled[:, 1], c=labels_kmeans/gmm, cmap=
```

2.2 Density-Based Clustering (DBSCAN)

- **Task 2.2: DBSCAN:** Explain the functioning of the **DBSCAN** algorithm and the interpretation of its hyperparameters (, `minPts`). Apply DBSCAN to the `X`$_moons_scaleddataandcommentonitsc$

```
Code Python 2.2: DBSCAN
from sklearn.cluster import DBSCAN
import numpy as np

TO BE COMPLETED: Choose epsilon and min_samples hyperparameters
dbscan = DBSCAN(eps=..., min_samples=...)
labels_dbscan = dbscan.fit_predict(X_moons_scaled)

Display the number of clusters found (excluding noise points, label -1)
num_clusters = len(set(labels_dbscan)) - (1 if -1 in labels_dbscan else 0)
print(f"\nDBSCAN: Number of clusters found: {num_clusters}")
print(f"Number of noise points (label -1): {np.sum(labels_dbscan == -1)}")
```


Louis Fippo Fitime: louis.fippo@univ-yaounde1.cm

2.3 Neighbor and Similarity Concepts

- **Task 2.3: k-NN and Metrics:** Describe the difference between the objectives of **Supervised Learning** (k-NN in classification/regression) and **Unsupervised Learning** (Clustering). Cite at least three relevant similarity metrics for textual or high-dimensional data, beyond Euclidean distance (e.g., Cosine, Jaccard).
- **Task 2.4 (Bonus): Search Optimization:** Explain how **KD-Trees** and **Locality-Sensitive Hashing (LSH)** structures are used to reduce the time complexity of nearest neighbor search in large databases (e.g., a document-based recommendation system).

---

3 Phase III: MLOps and Clustering Model Management

As seen in the previous TP, traceability is essential. Choosing the optimal number of clusters (K) is a crucial hyperparameter.

3.1 Segmentation Model Tracking with MLflow

- **Task 3.1: Experiment Structuring:** Using the **MLflow** approach, structure the code to log the results of the K-Means experiment (Phase I). The objective is to later compare the impact of the choice of K (hyperparameter) on Inertia and the Silhouette Score (metrics).

```
Code Python 3.1: Clustering Configuration Tracking
import mlflow
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, calinski_harabasz_score
import joblib

TO BE COMPLETED: Ensure MLflow is configured (mlflow.set_experiment)
def track_clustering_run(X_data, k, run_name):
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
labels = kmeans.fit_predict(X_data)

Calculate metrics
score_silhouette = silhouette_score(X_data, labels)
inertia_value = kmeans.inertia_

with mlflow.start_run(run_name=run_name):
    # 1. Logging Hyperparameters
    mlflow.log_param("K", k)
    mlflow.log_param("Algorithm", "KMeans")

    # 2. Logging Metrics
    mlflow.log_metric("Inertia", inertia_value)
    # TO BE COMPLETED: Log the Silhouette Coefficient
```


Louis Fippo Fitime: louis.fippo@univ-yaounde1.cm

```
    # 3. Logging Artifact (Centroids)
    # joblib.dump(kmeans, "kmeans_model.pkl")
    # mlflow.log_artifact("kmeans_model.pkl")

Example tracing for K=3 and K=4
Assuming X_scaled is available from Phase I
track_clustering_run(X_scaled, 3, "KMeans_K3_Initial")
track_clustering_run(X_scaled, 4, "KMeans_K4_Alternative")
print("\nMLflow tracking of K-Means experiments finished.")
```

3.2 Segmentation Model Deployment

- **Task 3.2: Packaging (Recap):** Once the optimal K is chosen, explain how the selected clustering model (K-Means) and the associated `StandardScaler` would be **packaged** (e.g., via `joblib`) for integration into a Dockerized web API (recap from previous TP). What is the role of this package in production for the marketing department?
- **Task 3.3: Post-Production (Monitoring):** If this customer segmentation model is deployed, cite and briefly describe two specific indicators of **Data Drift** or **Model Decay** that you should monitor (e.g., centroid drift, cluster size distribution).

---

4 Advanced Theoretical Concepts (EM, LDA)

- **Task 4.1: Initialization Comparison:** Discuss the problems of the non-convex optimization objective in Clustering (especially K-Means and GMM). Cite and compare two initialization techniques (e.g., K-Means++ vs. Random Initialization) to better converge toward a global optimum.
- **Task 4.2: Mixed Membership Modeling (LDA):** Describe the concept of **Latent Dirichlet Allocation (LDA)** for Topic Modeling in documents. How does LDA differ from K-Means clustering applied directly to word vectors (TF-IDF), particularly through the idea of *mixed membership*?

Louis Fippo Fitime: louis.fippo@univ-yaounde1.cm