Міністерство освіти і науки України

Львівський національний університет імені Івана Франка

Факультет електроніки та комп'ютерних технологій

Звіт

Про виконання лабораторної роботи №5

З курсу «Системи опрацювання даних**»**

«Кластеризація даних»

Виконав:

Студент групи ФеС-21

Осадчук Дмитро

Львів-2025

Мета роботи

Ознайомитися з методами кластеризації даних, реалізувати та порівняти алгоритми кластеризації (K-Means, DBSCAN, Agglomerative Clustering) на реальному датасеті, а також оцінити якість отриманих клас

Завдання

1. **Підготовка даних**
    - Завантажити набір даних
    - Виконати первинний аналіз даних: розмірність, типи змінних, наявність пропущених значень.
    - Провести масштабування (стандартизацію або нормалізацію) числових змінних.

1. Завантажив набір даних та виконав первинний аналіз даних, провів масштабування

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from huggingface_hub import hf_hub_download
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from scipy.cluster.hierarchy import dendrogram, linkage
from sklearn.neighbors import NearestNeighbors

df_d = pd.read_csv('size_1000_n_5_sepval_0.1.csv')
df_b = pd.read_csv("hf://datasets/mltrev23/wine-clustering/wine-clustering.csv")

scaler = StandardScaler()
X_d = scaler.fit_transform(df_d[['x', 'y']])
X_b = scaler.fit_transform(df_b.select_dtypes(include=np.number))

pca = PCA(n_components=2)
X_b_pca = pca.fit_transform(X_b)
```

```python
# 1. Перевірка на пропущені значення
print("Пропущені значення в df_d:")
print(df_d.isnull().sum())
print("\nПропущені значення в df_b:")
print(df_b.isnull().sum())

# 2. Заповнення пропущених значень (NaN)
# Для df_d (x, y - числові дані)
df_d['x'] = df_d['x'].fillna(df_d['x'].mean())  # Заповнення середнім значенням
df_d['y'] = df_d['y'].fillna(df_d['y'].mean())

# Для df_b (всі колонки числові)
numeric_columns_b = df_b.select_dtypes(include=np.number).columns
for col in numeric_columns_b:
    df_b[col] = df_b[col].fillna(df_b[col].mean())  # Заповнення середнім значенням

# 3. Перевірка на викиди (використаємо IQR для числових колонок)
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df
```

```python
# 3. Перевірка на викиди
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

# Видалення викидів для df_d
df_d_cleaned = remove_outliers(df_d, ['x', 'y'])
print(f"\nРозмір df_d після видалення викидів: {df_d_cleaned.shape}")

# Видалення викидів для df_b
df_b_cleaned = remove_outliers(df_b, numeric_columns_b)
print(f"Розмір df_b після видалення викидів: {df_b_cleaned.shape}")
```

```python
# 5. Масштабування даних
scaler = StandardScaler()

# Масштабування для df_d
X_d = scaler.fit_transform(df_d_cleaned[['x', 'y']])

# Масштабування для df_b
X_b = scaler.fit_transform(df_b_cleaned.select_dtypes(include=np.number))

# 6. Застосування PCA до X_b
pca = PCA(n_components=2)
X_b_pca = pca.fit_transform(X_b)
```

Отримані результати даних:

```
Пропущені значення в df_d:
Unnamed: 0    0
x             0
y             0
run           0
dtype: int64
```

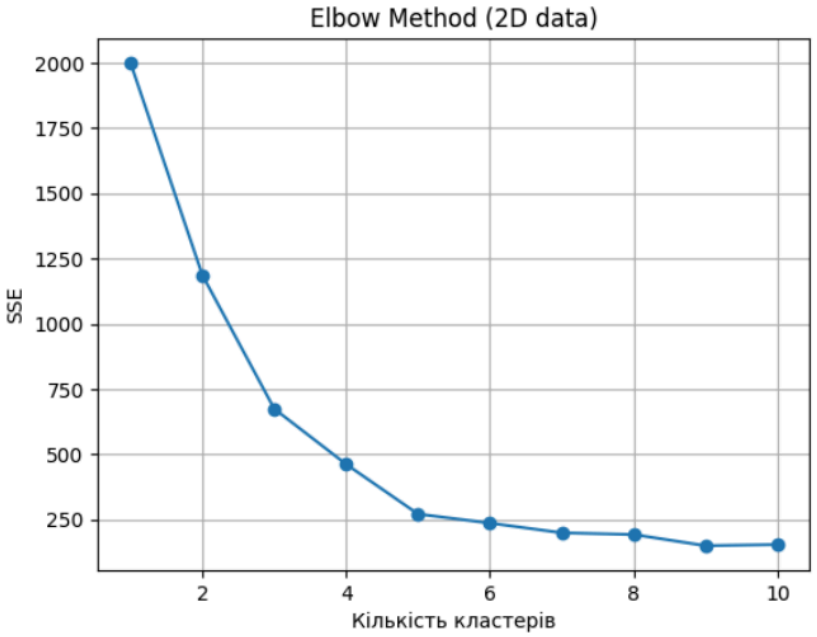
```
Пропущені значення в df_b:
Alcohol                 0
Malic_Acid              0
Ash                     0
Ash_Alcanity            0
Magnesium               0
Total_Phenols           0
Flavanoids              0
Nonflavanoid_Phenols    0
Proanthocyanins         0
Color_Intensity         0
Hue                     0
OD280                   0
Proline                 0
dtype: int64
```

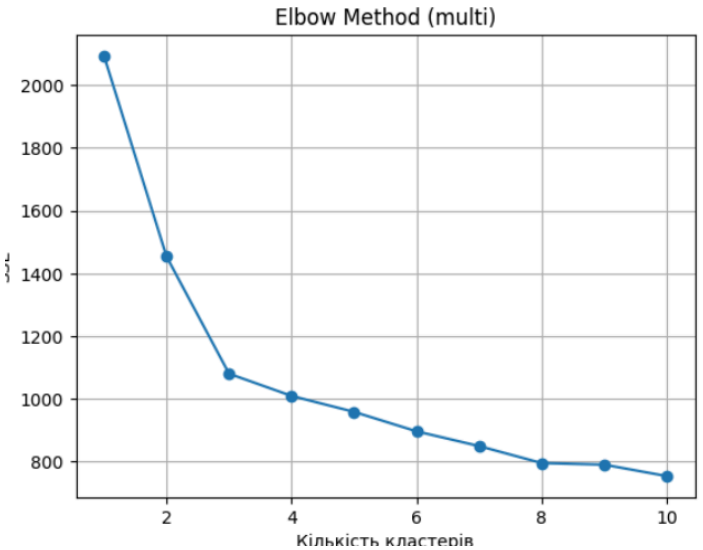
```
Типи даних у df_d:
Unnamed: 0      int64
x             float64
y             float64
run            object
dtype: object
```

2. **Реалізація кластеризації**
    - Визначити оптимальну кількість кластерів за допомогою методу **"Elbow Method"** або **Silhouette Score**.
    - Виконати кластеризацію за допомогою **K-Means**.
    - Реалізувати та застосувати **DBSCAN** для виявлення аномалій.
    - Використати **Agglomerative Clustering (ієрархічну кластеризацію)** та побудувати дендрограму.

2. Визначив оптимальну кількість кластерів через Elbow Method

```
sse_d = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k).fit(X_d)
    sse_d.append(kmeans.inertia_)

plt.figure()
plt.plot(range(1, 11), sse_d, marker='o')
plt.title('Elbow Method (2D data)')
plt.xlabel('Кількість кластерів')
plt.ylabel('SSE')
plt.grid()
plt.show()

sse_b = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k).fit(X_b)
    sse_b.append(kmeans.inertia_)

plt.figure()
plt.plot(range(1, 11), sse_b, marker='o')
plt.title('Elbow Method (multi)')
plt.xlabel('Кількість кластерів')
plt.ylabel('SSE')
plt.grid()
plt.show()
```

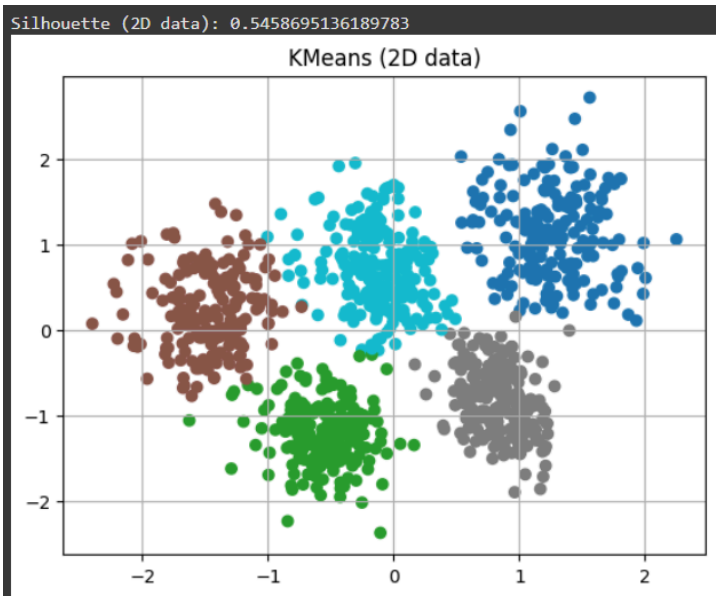


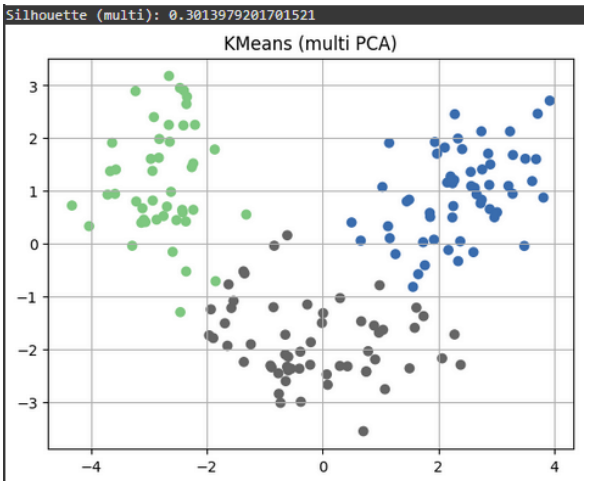


Метод "ліктя" допомагає визначити оптимальну кількість кластерів.

На графіку видно, що інерція різко падає від K=1 до K=3, а потім зменшення стає більш плавним. "Злам" виглядає приблизно на K=5

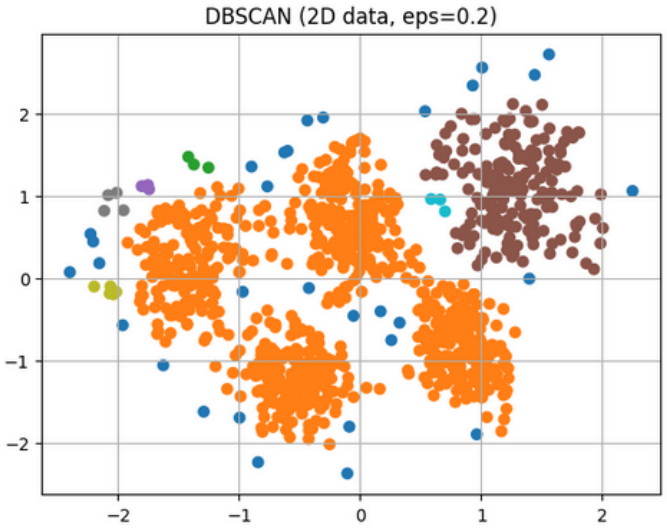
Виконав k-means кластеризацію

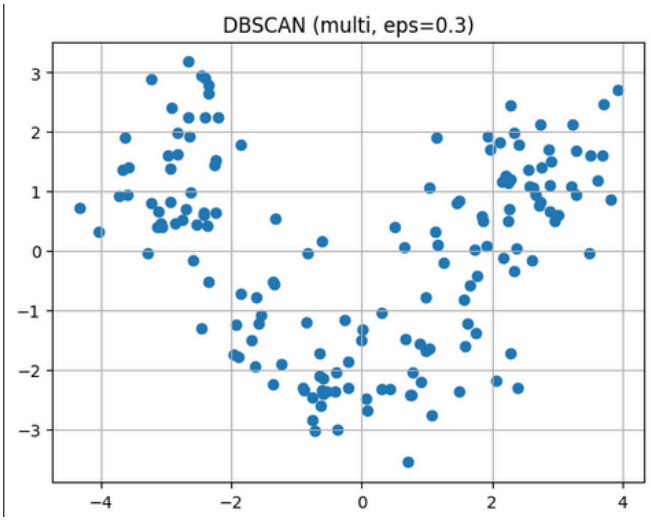




Виконав та реалізував DBSCAN для виявлення аномалій

```
eps_range = [2, 3, 4, 5]
eps = 0.2
db_d = DBSCAN(eps=eps, min_samples=3).fit(X_d)
labels_d = db_d.labels_
n_clusters_d = len(set(labels_d)) - (1 if -1 in labels_d else 0)
score_d = silhouette_score(X_d, labels_d) if n_clusters_d > 1 else -1

plt.figure()
plt.scatter(X_d[:, 0], X_d[:, 1], c=labels_d, cmap='tab10')
plt.title(f"DBSCAN (2D data, eps={eps})")
plt.grid()
plt.show()


eps = 0.3
db_b = DBSCAN(eps=eps, min_samples=3).fit(X_b)
labels_b = db_b.labels_
n_clusters_b = len(set(labels_b)) - (1 if -1 in labels_b else 0)
score_b = silhouette_score(X_b, labels_b) if n_clusters_b > 1 else -1


plt.figure()
plt.scatter(X_b_pca[:, 0], X_b_pca[:, 1], cmap='tab10')
plt.title(f"DBSCAN (multi, eps={eps})")
plt.grid()
plt.show()
```

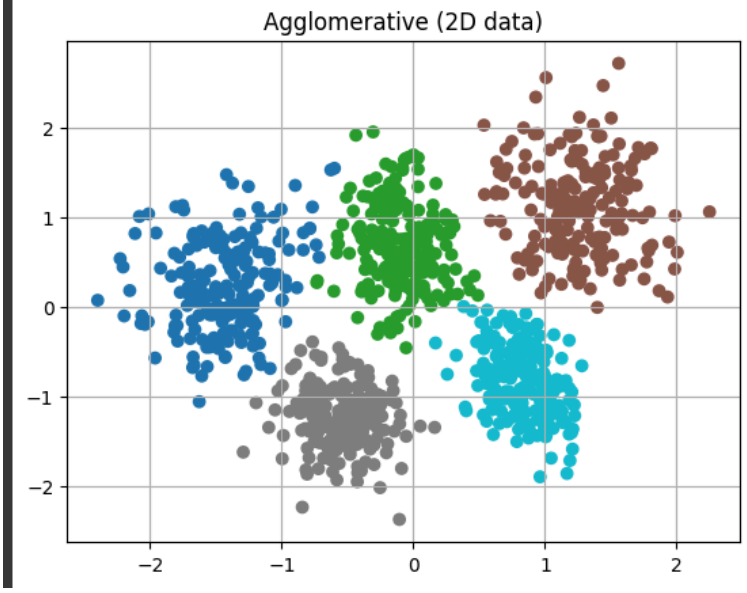




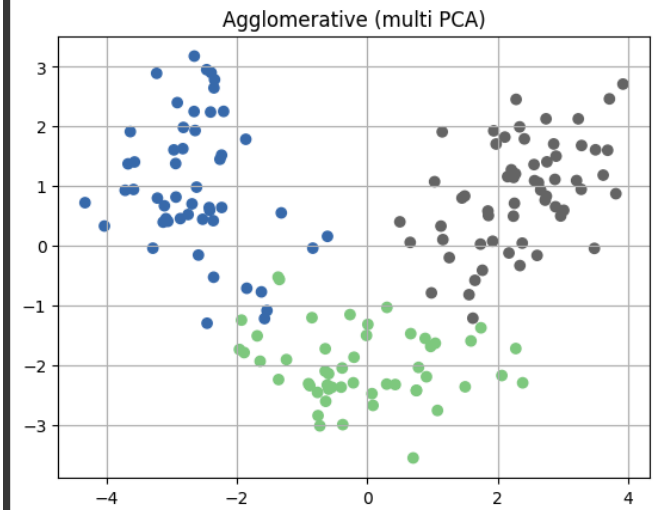
Реалізував ієрархічну кластеризацію та побудував дендограму

```
agg_d = AgglomerativeClustering(n_clusters=5)
labels_ag_d = agg_d.fit_predict(X_d)
print("Silhouette (Agglo 2D data):", silhouette_score(X_d, labels_ag_d))

plt.figure()
plt.scatter(X_d[:, 0], X_d[:, 1], c=labels_ag_d, cmap='tab10')
plt.title("Agglomerative (2D data)")
plt.grid()
plt.show()

agg_b = AgglomerativeClustering(n_clusters=3)
labels_ag_b = agg_b.fit_predict(X_b)
print("Silhouette (Agglo multi):", silhouette_score(X_b, labels_ag_b))

plt.figure()
plt.scatter(X_b_pca[:, 0], X_b_pca[:, 1], c=labels_ag_b, cmap='Accent')
plt.title("Agglomerative (multi PCA)")
plt.grid()
plt.show()
```

```
Silhouette (Agglo 2D data): 0.5383626078765015
```

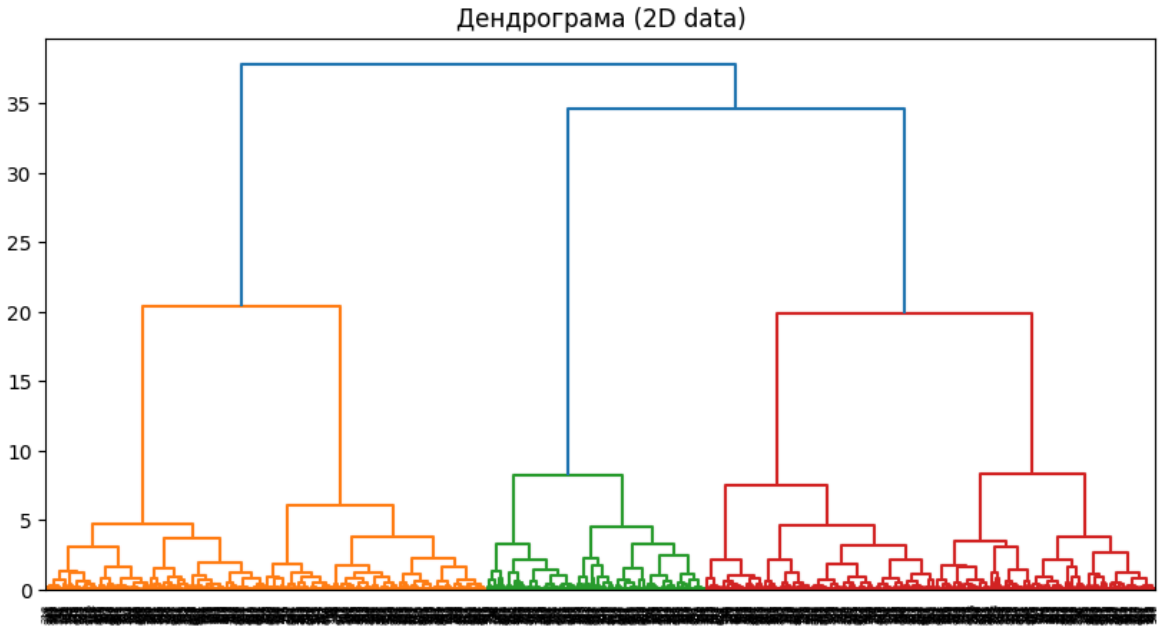


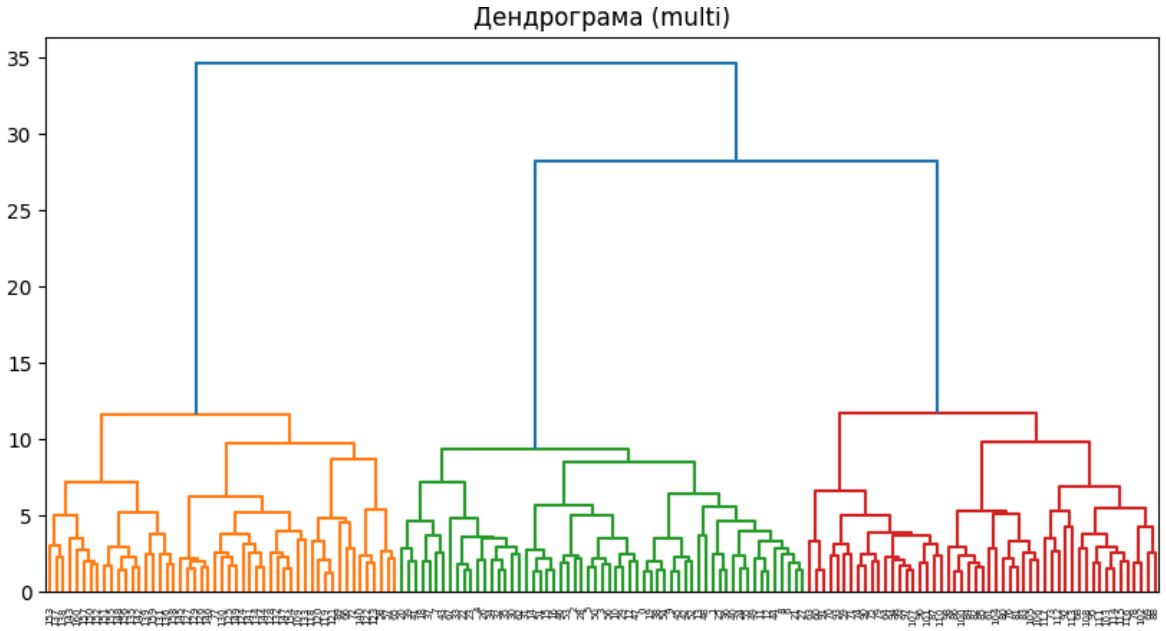

```
Silhouette (Agglo multi): 0.29574004115878627
```



Побудова дендрограм для двох наборів даних:

```
linked_d = linkage(X_d, 'ward')
plt.figure(figsize=(10, 5))
dendrogram(linked_d)
plt.title("Дендрограма (2D data)")
plt.show()

linked_b = linkage(X_b, 'ward')
plt.figure(figsize=(10, 5))
dendrogram(linked_b)
plt.title("Дендрограма (multi)")
plt.show()
```





Висновок:

Я проаналізував дані, використовуючи три методи кластеризації: K-means, Agglomerative Clustering та DBSCAN. Перед цим була проведена первинна обробка даних: виявлення пропущених значень, їх завповненням середнім значенням з колонки, видалення пустих колонок, нормалізація та масштабування даних. Для кожного алгоритму я візуалізував результати кластеризації, використовуючи PCA для зменшення розмірності даних. Також я порівняв якість кластеризації за допомогою метрики Silhouette Score та проаналізував середні значення змінних для кожного кластера, отриманого за допомогою K-means. PCA допомагає візуалізувати кластери у багатовимірному просторі.

Аналіз середніх значень змінних для кожного кластера дозволяє інтерпретувати результати кластеризації.