ЛАБОРАТОРНА РОБОТА № 4

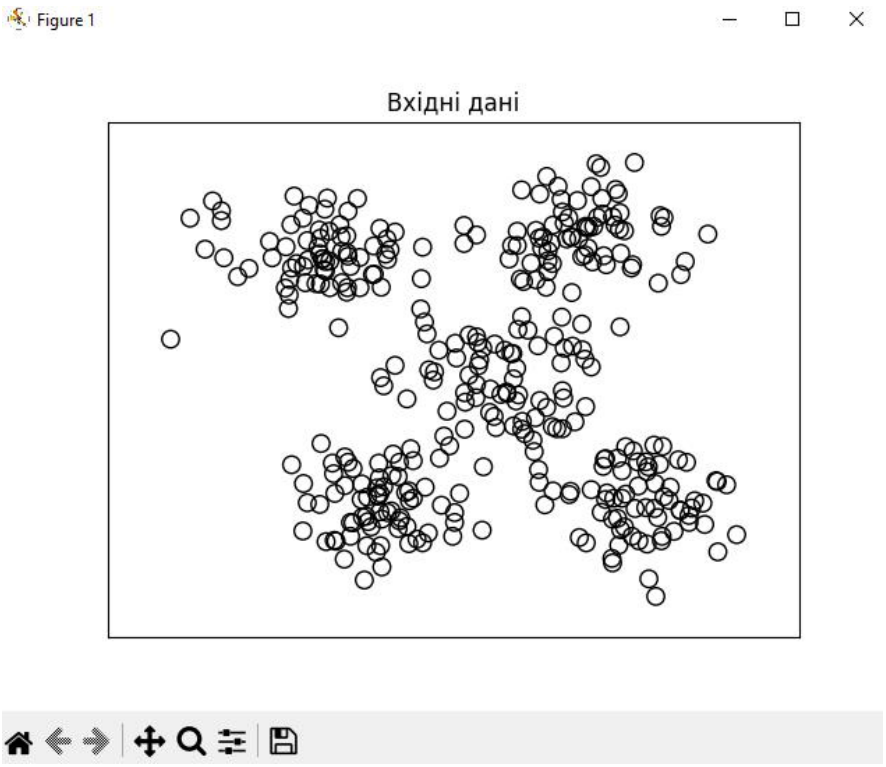
ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

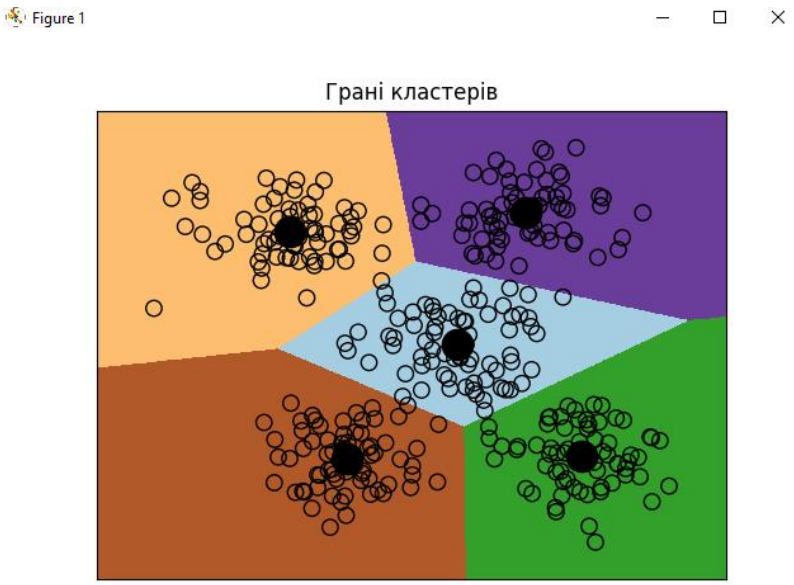
Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних у машинному навчанні.

Git: https://github.com/flekXD/SAI

Завдання 1.

Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data_clustering.txt.






```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn import metrics

# Завантаження даних із файлу
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Кількість кластерів
num_clusters = 5
kmeans = KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)
kmeans.fit(X)
step_size = 0.01

# Візуалізація вхідних даних
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_vals, y_vals = np.meshgrid(np.arange(x_min, x_max, step_size), np.arange(y_min,
y_max, step_size))
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
output = output.reshape(x_vals.shape)
plt.figure()
plt.clf()
plt.imshow(output, interpolation='nearest', extent=(x_vals.min(), x_vals.max(),
y_vals.min(), y_vals.max()),
           cmap=plt.cm.Paired,
           aspect='auto',
           origin='lower')
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black',
s=80)
cluster_centers = kmeans.cluster_centers_
plt.scatter(cluster_centers[:, 0], cluster_centers[:, 1], marker='o', s=210,
linewidths=4, color='black', zorder=12, facecolors='black')

plt.title('Грані кластерів')
```

```
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)  # Зміна для Y-вісі
plt.xticks(())
plt.yticks(())
plt.show()  # Виклик з дужками
```

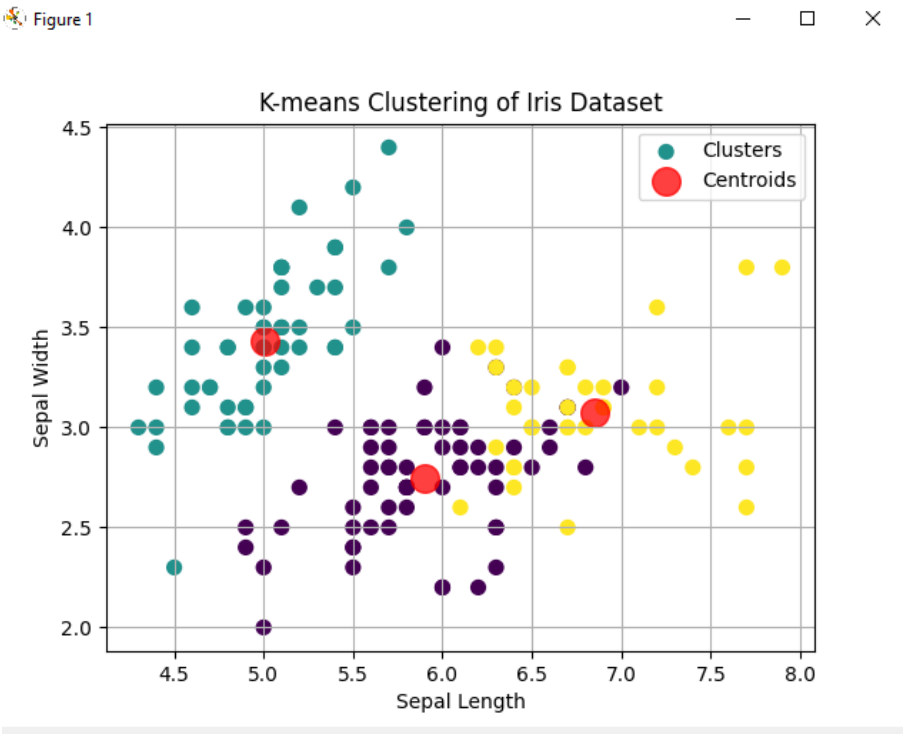
Проведена кластеризація даних методом k-середніх (k-means) дозволила організувати вхідні дані з файлу data_clustering.txt у п'ять чітко визначених груп. Застосування алгоритму із вдосконаленою ініціалізацією центроїдів (k-means++) забезпечило швидку збіжність та точне позиціонування кластерних центрів.

Завдання 2. Кластеризація K-середніх для набору даних Iris

Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів набору даних Iris.

```
# Імпортуємо необхідні бібліотеки
from sklearn.cluster import KMeans  # Для кластеризації методом k-means
from sklearn.datasets import load_iris  # Для завантаження набору даних Iris
import matplotlib.pyplot as plt  # Для візуалізації
import numpy as np  # Для роботи з масивами

# Завантажуємо набір даних Iris
iris = load_iris()
X = iris['data']  # Вибираємо дані (4 атрибути для кожного зразка)
y = iris['target']  # Вибираємо цільові значення (класи квітів)

# Створюємо об'єкт KMeans
# Вказуємо кількість кластерів (n_clusters=3, оскільки маємо три класи квітів)
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300,
random_state=42)

# Навчаємо модель на даних X
kmeans.fit(X)

# Прогнозуємо мітки кластерів для всіх точок у наборі даних
y_kmeans = kmeans.predict(X)

# Візуалізація результатів
# Відображаємо точки даних на площині за двома першими ознаками (довжина та ширина
чашолистка)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
```

```python
# Відображаємо центроїди кластерів
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75,
label='Centroids')

# Додаємо легенду та підписи до осей
plt.xlabel('Sepal Length')
plt.ylabel('Sepal Width')
plt.title('K-means Clustering of Iris Dataset')
plt.legend(['Clusters', 'Centroids'])
plt.grid(True)
plt.show()
```



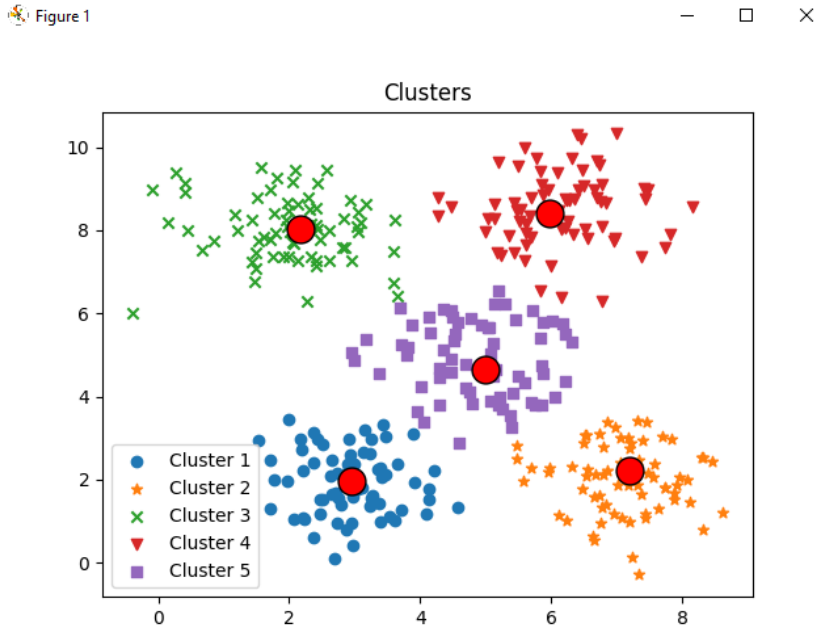
Висновок

Кластеризація набору даних Iris методом k-means успішно розподілила дані на три групи, відповідні трьом типам ірисів (Setosa, Versicolour, Virginica). Центроїди кластерів були точно визначені, а графічна візуалізація підтвердила ефективність алгоритму. Метод k-means добре підходить для аналізу подібних багатовимірних даних.

Завдання 3. Оцінка кількості кластерів з використанням методу зсуву середнього

Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle

# Завантаження даних
X = np.loadtxt('data_clustering.txt', delimiter=',')

# Оцінка ширини смуги пропускання
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# Ініціалізація та навчання моделі MeanShift
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

# Центри кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters\n', cluster_centers)

# Отримання міток кластерів та кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print('\nNumber of clusters in input data =\n', num_clusters)

# Візуалізація
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), cycle(markers)):
    # Відображення точок, що належать кластеру
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,
label=f"Cluster {i+1}")

# Відображення центрів кластерів
for center in cluster_centers:
    plt.plot(center[0], center[1], marker='o', markerfacecolor='red',
             markeredgecolor='black', markersize=15)

plt.title('Clusters')
plt.legend()
plt.show()
```



Висновок

Алгоритм зсуву середнього (MeanShift) визначив оптимальну кількість кластерів у наборі даних з файлу *data_clustering.txt*, яка становить 5. Метод автоматично виділяє кластери за щільністю даних, що робить його зручним для аналізу без попереднього знання кількості груп.

Завдання 4 Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

```
import json
import numpy as np
import pandas as pd
from sklearn import covariance, cluster
import yfinance as yf

# Function to download stock quotes from Yahoo Finance
def quotes_yahoo(ticker, start, end):
    df = yf.download(ticker, start=start, end=end)
    df['Date'] = df.index
    return df

# Load company symbols mapping
input_file = 'company_symbol_mapping.json'
with open(input_file, 'r') as f:
    company_symbols_map = json.load(f)

symbols, names = np.array(list(company_symbols_map.items())).T

# Parameters for analysis
start_date = "2003-07-03"
end_date = "2007-05-04"
```

```
# Collect data for all companies
all_quotes = {}
common_dates = None

for symbol in symbols:
    print(f"Downloading data for {symbol}...")
    try:
        quotes = quotes_yahoo(symbol, start_date, end_date)
        if quotes.empty:
            print(f"No data found for {symbol}")
            continue  # Skip this symbol if no data is returned
        quotes['variation'] = quotes['Close'] - quotes['Open']
        all_quotes[symbol] = quotes.set_index('Date')['variation']

        # Determine the common date range
        if common_dates is None:
            common_dates = quotes.index
        else:
            common_dates = common_dates.intersection(quotes.index)
    except Exception as e:
        print(f"Error downloading data for {symbol}: {e}")

# Align all data to the common date range
aligned_quotes = []
valid_symbols = []

for symbol, series in all_quotes.items():
    # Reindex to align with common dates
    series = series.reindex(common_dates)
    if not series.isna().all():  # Exclude stocks with completely missing data
        aligned_quotes.append(series.fillna(0).values)  # Replace NaN with 0
        valid_symbols.append(symbol)

# Convert to NumPy array
if aligned_quotes:
    X = np.array(aligned_quotes).T  # Transpose to match sklearn's requirements
else:
    print("No valid data available for analysis.")
    X = None

# Ensure X is valid for modeling
if X is None or X.shape[1] < 2:
    raise ValueError("Insufficient data for covariance modeling.")

# Normalize data
X /= X.std(axis=0, where=~np.isnan(X))  # Handle possible NaN during
standardization

# Covariance model and clustering
edge_model = covariance.GraphicalLassoCV()
with np.errstate(invalid='ignore'):
```

```
    edge_model.fit(X)

_, labels = cluster.affinity_propagation(edge_model.covariance_)

# Display clustering results
num_labels = labels.max()
print("\nClustering results:")
for i in range(num_labels + 1):
    cluster_companies = [valid_symbols[j] for j in range(len(valid_symbols)) if
labels[j] == i]
    print(f"Cluster {i + 1}: {', '.join(cluster_companies)}")
```

```
Clustering results:
Cluster 1: XOM, CVX, COP, VLO
Cluster 2: TM, F, HMC, BA, MCD, AAPL, SAP, CAT
Cluster 3: MDLZ
Cluster 4: KO, PEP, K, PG, CL, KMB
Cluster 5: TWX, CMCSA, MAR, WFC, JPM, AIG, AXP, BAC, GS, XRX, WMT, HD, R, DD
Cluster 6: MSFT, IBM, HPQ, AMZN, MMM, GE, CSCO, TXN
Cluster 7: NOC, LMT, GD
Cluster 8: WBA, CVS
Cluster 9: GSK, PFE, SNY, NVS
PS D:\123>
```

Висновок:

У результаті дослідження методів неконтрольованої класифікації даних за допомогою Python було виявлено, що найбільш ефективними є алгоритми, як-от K-середні для кластеризації та PCA для зменшення розмірності. Використання бібліотек Python, таких як Scikit-learn та TensorFlow, дозволяє ефективно обробляти неструктуровані дані та отримувати корисні патерни без необхідності в маркуванні даних. Це підкреслює важливість вибору правильних методів залежно від типу даних.