ЛАБОРАТОРНА РОБОТА № 7

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Завдання 2.1. Кластеризація даних за допомогою методу k-середніх

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

# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')
num_clusters = 5

# Створення об'єкту Kmeans
kmeans= KMeans(init='k-means++', n_clusters=num_clusters, n_init=10)

#Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Визначення кроку сітки
step_size = 0.01

#Відображення точок сітки
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))

# Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

# Включення вхідних даних до графіка
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)
```

Рис 1.1 – лістинг програми

```python
# Відображення центрів кластерів
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')
x_min, x_max = X[:, 0].min() - 1, X[:,0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
plt.title('Границі кластеров')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

plt.show()


# Оцінка якості кластеризації
silhouette_score = metrics.silhouette_score(X, kmeans.labels_)
inertia = kmeans.inertia_

# Виведення результатів
print(f"Силуетний коефіцієнт: {silhouette_score:.3f}")
print(f"Коефіцієнт інерції: {inertia:.3f}")
```

Рис 1.2 – лістинг програми

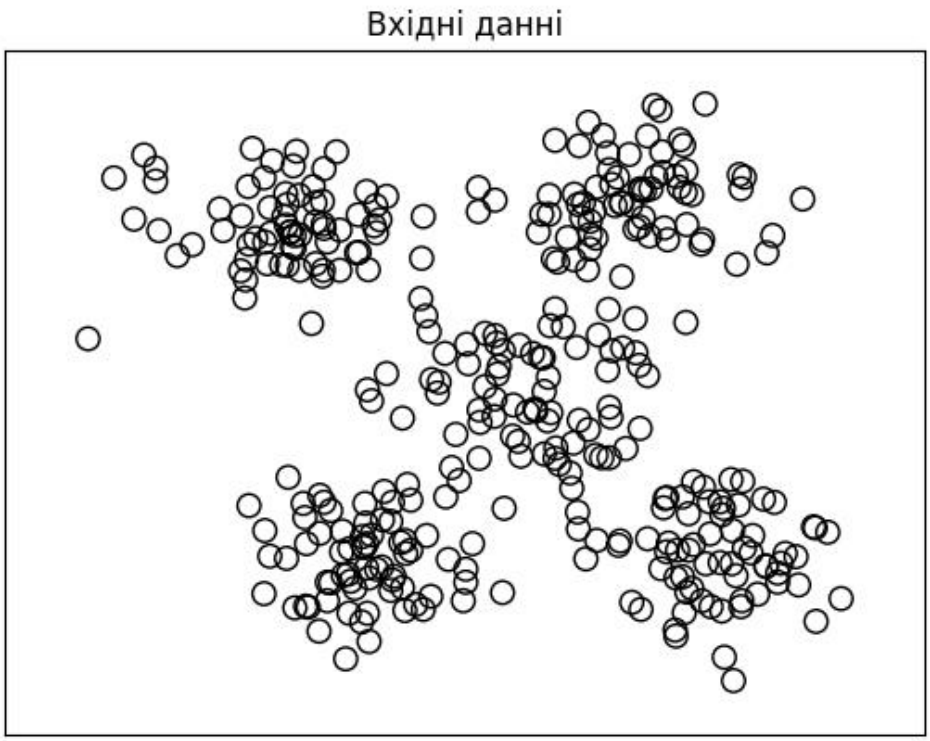

Рис 1.3 – результат виконання

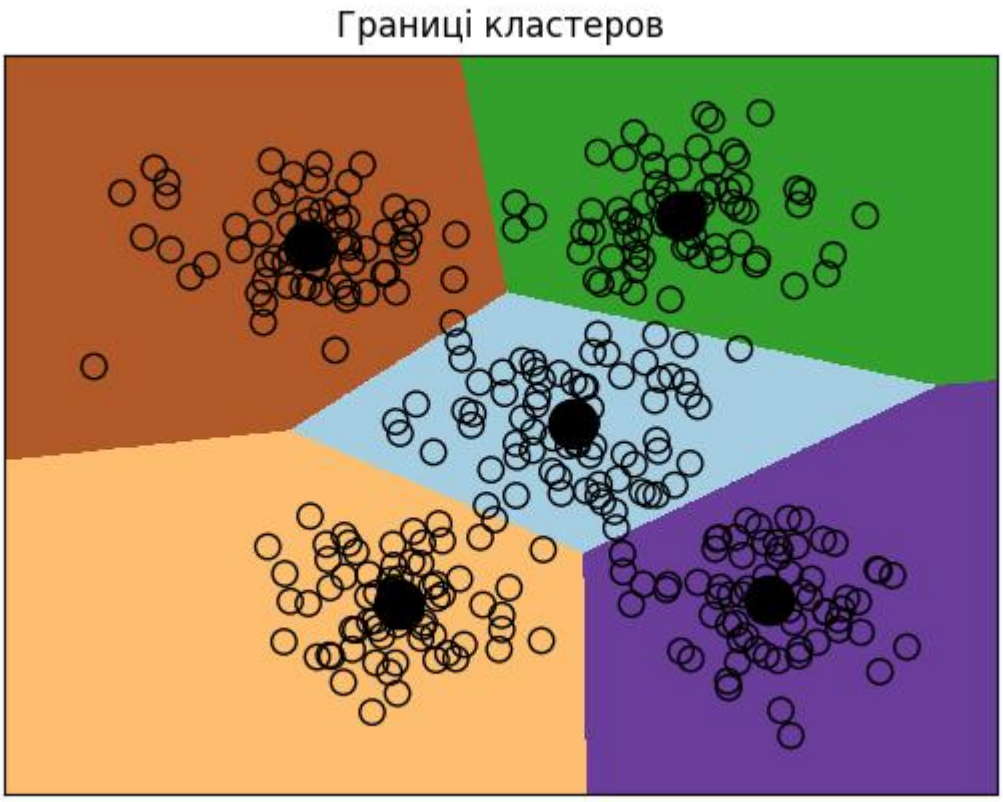

Рис 1.4 – результат виконання

```
C:\Users\Администратор\PycharmProject
Силуетний коефіцієнт: 0.591
Коефіцієнт інерції: 433.803
```

Рис 1.5 – результат виконання

На основі результатів кластеризації, відображених на графіку, можна зробити висновок, що дані успішно розподілені на п'ять окремих кластерів, кожен з яких має чіткі межі. Центри кластерів (чорні точки) точно відповідають середині своїх відповідних груп, що підтверджує коректність кластеризації. Візуалізація також демонструє, що кластери добре відокремлені один від одного, що свідчить про високу якість класифікації за допомогою методу KMeans. Силуетний коефіцієнт має значення в діапазоні від -1 до 1, де значення близьке до 1 свідчить про гарну кластеризацію, а значення близьке до -1 — про погану.

Завдання 2.2 Кластеризація K-середніх для набору даних Iris

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import load_iris
from sklearn.metrics import pairwise_distances_argmin

# Завантажуємо набір даних Iris
iris = load_iris()
X = iris['data']
y = iris['target']

# Ініціалізація KMeans з 5 кластерами
kmeans = KMeans(n_clusters=5, init='k-means++', n_init=10, max_iter=300, tol=0.0001, random_state=None)
kmeans.fit(X)

# Передбачення міток для кожної точки
y_kmeans = kmeans.predict(X)

# Візуалізація результатів кластеризації
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)
plt.title("KMeans кластеризація з 5 кластерами")
plt.show()


# Функція для пошуку кластерів без використання KMeans
def find_clusters(X, n_clusters, rseed=2):  2 usages  new *
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]  # Вибираємо випадкові точки як початкові центри
    centers = X[i]

    while True:
        labels = pairwise_distances_argmin(X, centers)
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clusters)])  # Обчислюємо нові центри клас
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels


# Використовуємо функцію find_clusters з 3 кластерами
centers, labels = find_clusters(X, n_clusters: 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title("Кластери за допомогою функції find_clusters (3 кластери)")
plt.show()
```

Рис 1.6 – лістинг програми

```python
# Використовуємо функцію find_clusters з 3 кластерами
centers, labels = find_clusters(X, n_clusters: 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title("Кластери за допомогою функції find_clusters (3 кластери)")
plt.show()

# Використовуємо функцію find_clusters з іншими випадковими центрами
centers, labels = find_clusters(X, n_clusters: 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title("Кластери за допомогою find_clusters з іншими центрами")
plt.show()

# Кластери за допомогою KMeans з 3 кластерами
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.title("KMeans кластеризація з 3 кластерами")
plt.show()
```

Рис 1.7 – лістинг програми

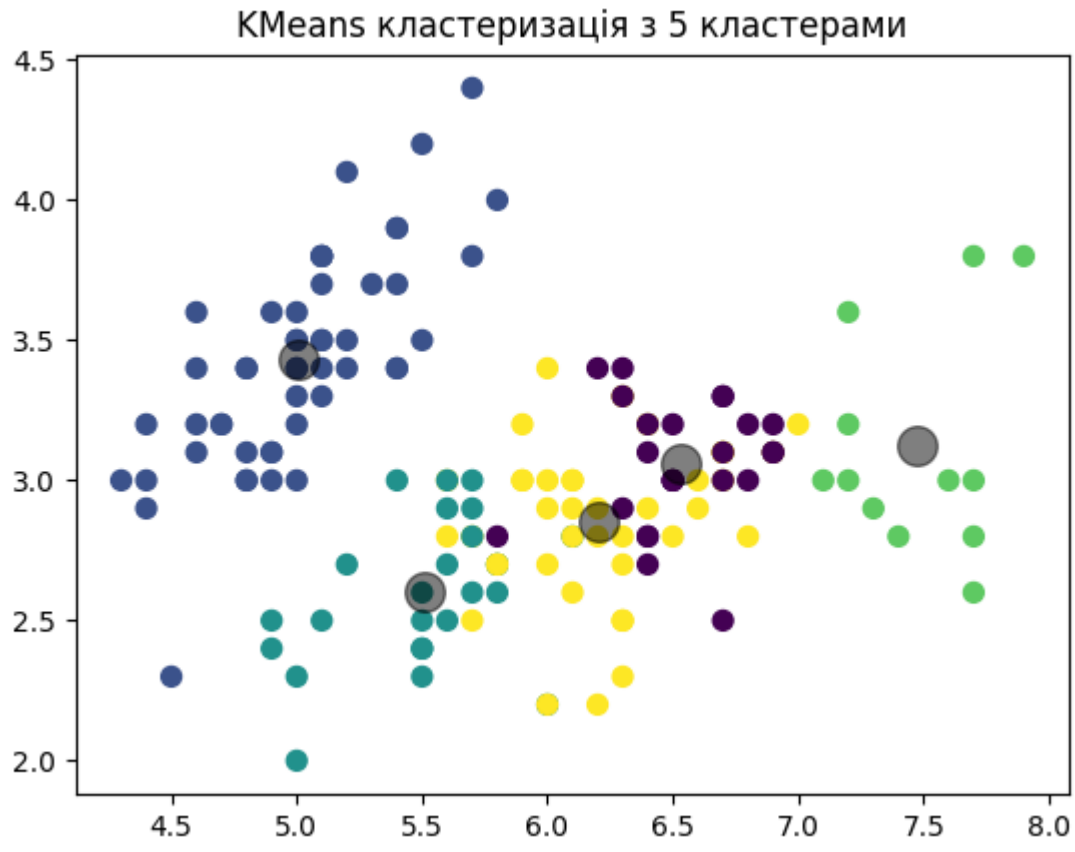

Рис 1.8 – результат виконання

Кластеризація з 5 кластерами показує, що алгоритм створює більше груп, ніж існує реальних класів у наборі даних, що може призвести до деяких помилок у класифікації.

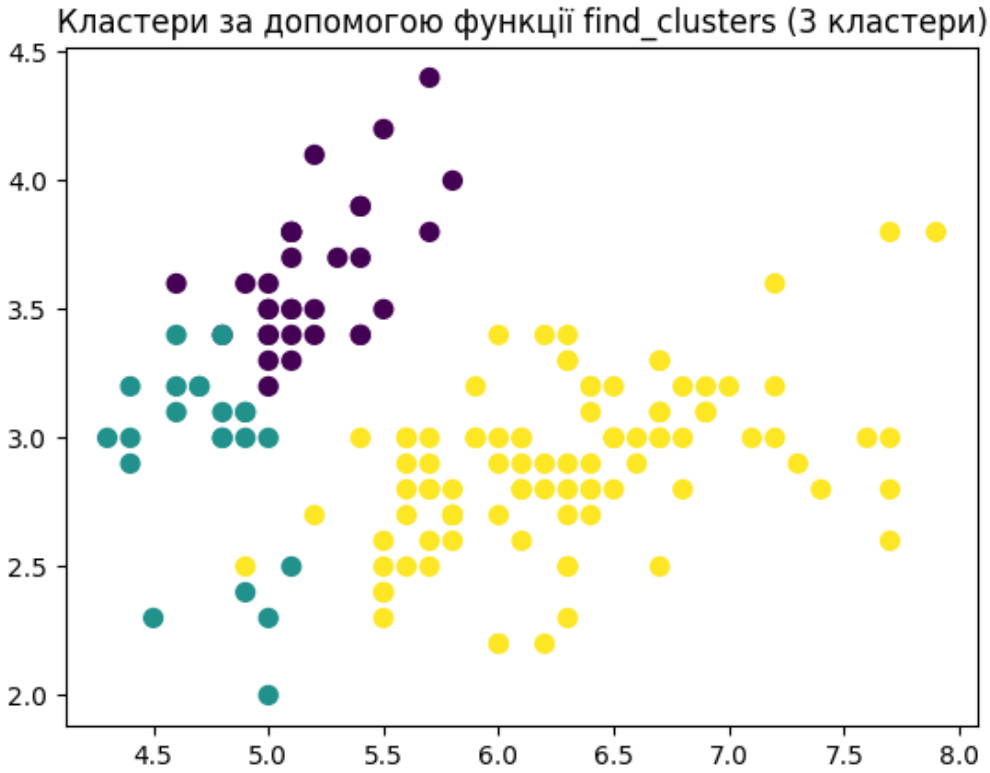

Рис 1.9 – результат виконання

Ручна кластеризація за допомогою функції find_clusters з 3 кластерами точніше відображає структуру даних, оскільки кількість кластерів відповідає реальним класам квітів.

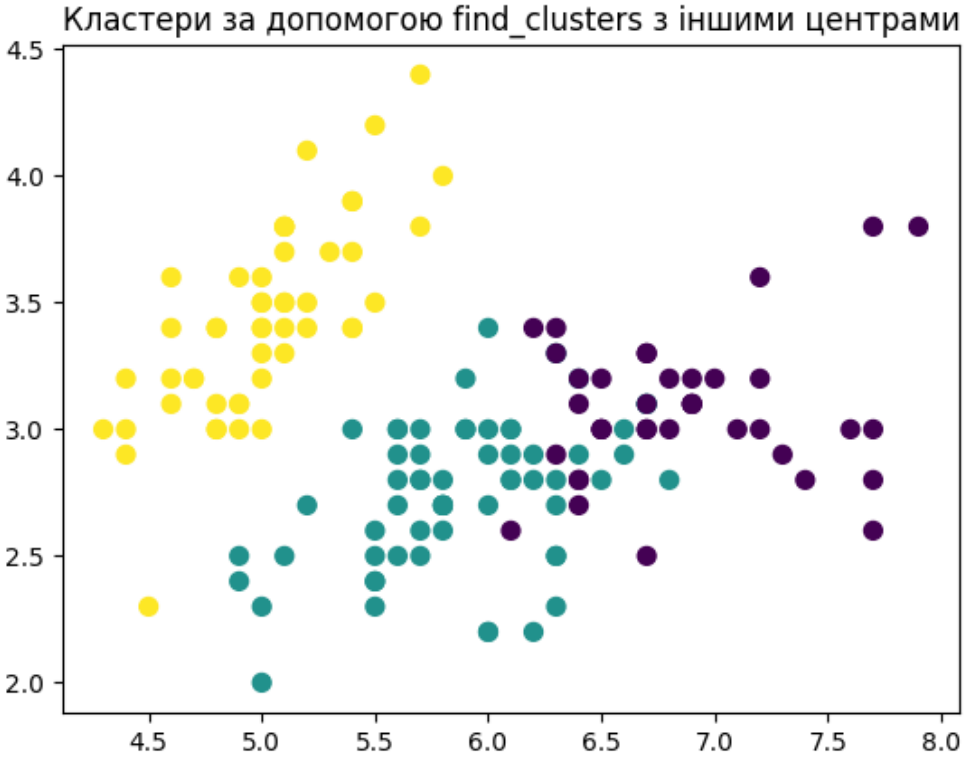

Рис 1.10 – результат виконання

Кластеризація з іншими випадковими початковими центрами показує варіативність результатів, що залежить від вибору початкових точок, підтверджуючи чутливість алгоритмів до їх ініціалізації.

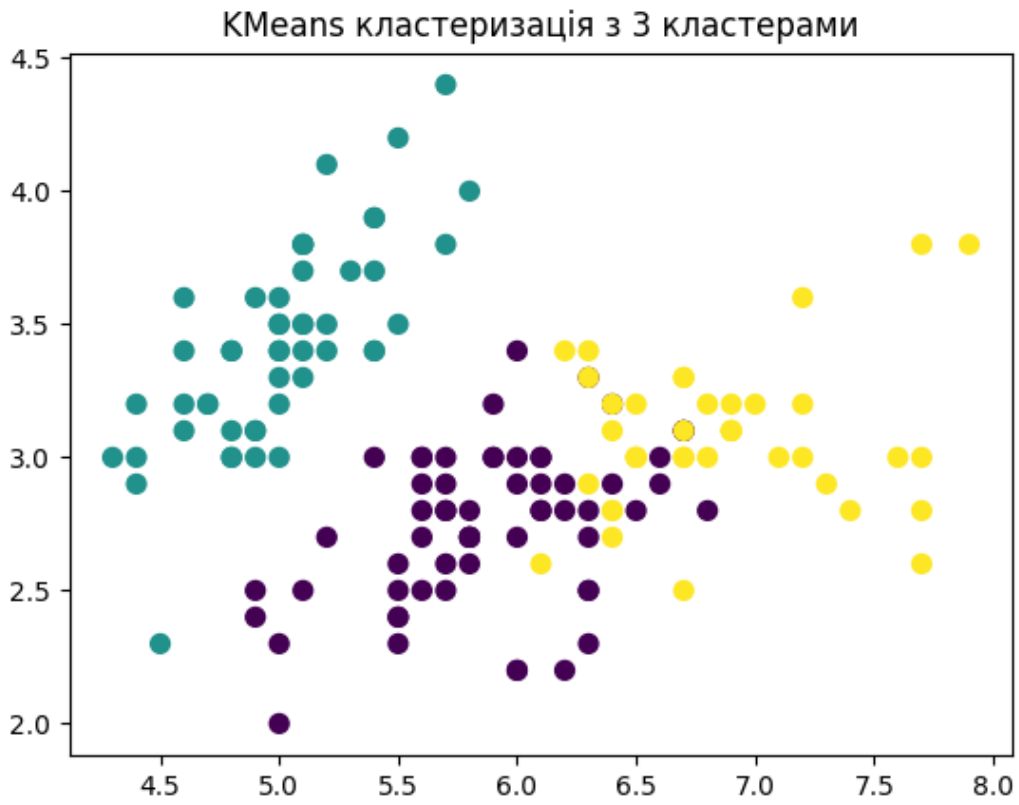

Рис 1.11 – результат виконання

Кластеризація KMeans з 3 кластерами дає найкращий результат, оскільки кількість кластерів відповідає реальним класам ірисів, що забезпечує точний поділ даних.

Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього

```python
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
from matplotlib import 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 вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))
```

Рис 1.12 – лістинг програми

```python
# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)

# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_
print('\nCenter of clusters:\n', cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))

print('\nCenter of clusters in input data =', num_clusters)
# Відображення на графіку точок та центрів кластерів
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color='blue')
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o', markerfacecolor='black', markeredgecolor='black',
             markersize=15)
plt.title('Кластери')
plt.show()
```

Рис 1.13 – лістинг програми

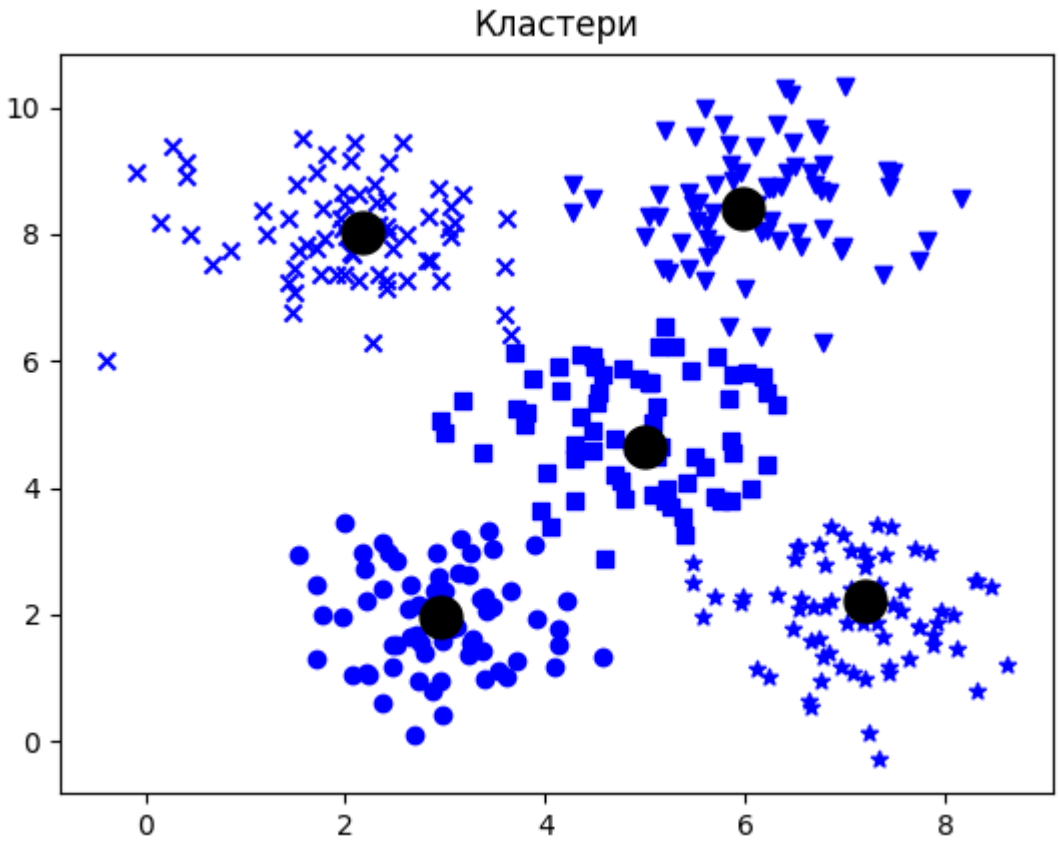

Рис 1.14 – результат виконання

```
C:\Users\Администратор\PycharmProjects\.venv\Sc

Center of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Center of clusters in input data = 5
```

Рис 1.15 – результат виконання

Графік показує кластеризацію даних методом зсуву середнього (MeanShift), де кожен кластер позначений різними маркерами. Центри кластерів вказані чорними точками, що демонструє, як алгоритм розділяє дані на окремі групи на основі оцінки ширини вікна.

Завдання 2.4. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

```python
import datetime
import json
import numpy as np
from sklearn import covariance, cluster
import matplotlib
matplotlib.use('TkAgg')
from matplotlib import pyplot as plt
from matplotlib.finance import quotes_historical_yahoo_ochl as quotes_yahoo

# Вхідний файл із символічними позначеннями компаній
input_file = 'company_symbol_mapping.json'

# Завантаження прив'язок символів компаній до їх повних назв
with open(input_file, 'r') as f:
 company_symbols_map = json.loads(f.read())
symbols, names = np.array(list(company_symbols_map.items())).T

# Завантаження архівних даних котирувань
start_date = datetime.datetime( year: 2003,  month: 7,  day: 3)
end_date = datetime.datetime( year: 2007,  month: 5,  day: 4)
quotes = [quotes_yahoo(symbol, start_date, end_date, asobject=True) for symbol in symbols]

# Вилучення котирувань, що відповідають відкриттю та закриттю біржі
opening_quotes = np.array([quote.open for quote in quotes]).astype(np.float)
closing_quotes = np.array([quote.close for quote in quotes]).astype(np.float)

# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
X = quotes_diff.copy().T
X /= X.std(axis=0)
```

Рис 1.16 – лістинг програми

```python
# Обчислення різниці між двома видами котирувань
quotes_diff = closing_quotes - opening_quotes
X = quotes_diff.copy().T
X /= X.std(axis=0)

# Створення моделі графа
edge_model = covariance.GraphicalLassoCV()

# Навчання моделі
with np.errstate(invalid='ignore'):
    edge_model.fit(X)

# Створення моделі кластеризації на основі поширення подібності
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()
for i in range(num_labels + 1):
    print("Cluster", i + 1, "==>", ','.join(names[labels == i]))
```

Рис 1.17 – літсинг програми

```
C:\Users\Администратор\PycharmProjects\.venv\Scripts\python.exe C:\Users\Администратор\PycharmProje
Traceback (most recent call last):
  File "C:\Users\Администратор\PycharmProjects\lab7\LR_7_task_4.py", line 8, in <module>
    from matplotlib.finance import quotes_historical_yahoo_ochl as quotes_yahoo
ModuleNotFoundError: No module named 'matplotlib.finance'
```

Рис 1.18 – результат виконання

На жаль, завдання не могло виконатись, бо модуль matplotlib.finance, який я намагаюсь імпортувати був видалений з нових версій бібліотеки matplotlib.

Висновок: На лабораторній роботі мага змогу використовувати спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних

Посилання на репозиторій:
https://github.com/ipz211/shi_prokopchuk_oleksandra_ipz-21-1_