ЛАБОРАТОРНА РОБОТА № 4

Дослідження методів неконтрольованого навчання

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

Хід роботи:

Завдання 2.1. Кластеризація даних за допомогою k-середніх

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

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

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

# Включення вхідних даних до графіка
plt.figure()
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', s=80)
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('Вхідні дані')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

# 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)
```

Рис. 1. Код програми

```
)

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

# Графічне відображення областей та виділення їх кольором
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')

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', s=80)

# 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('Межі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

Рис. 2. Код програми

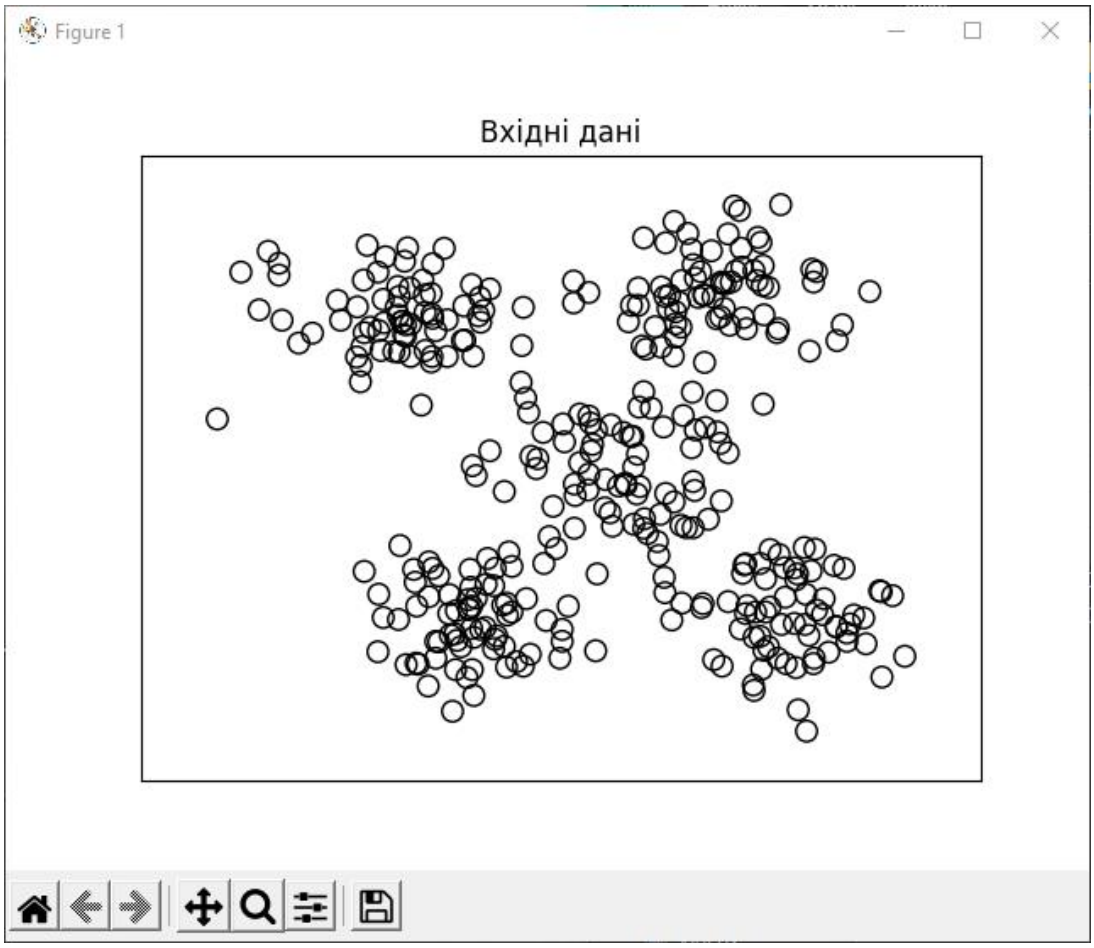

Рис. 3. Графік результату пошуку кластерів

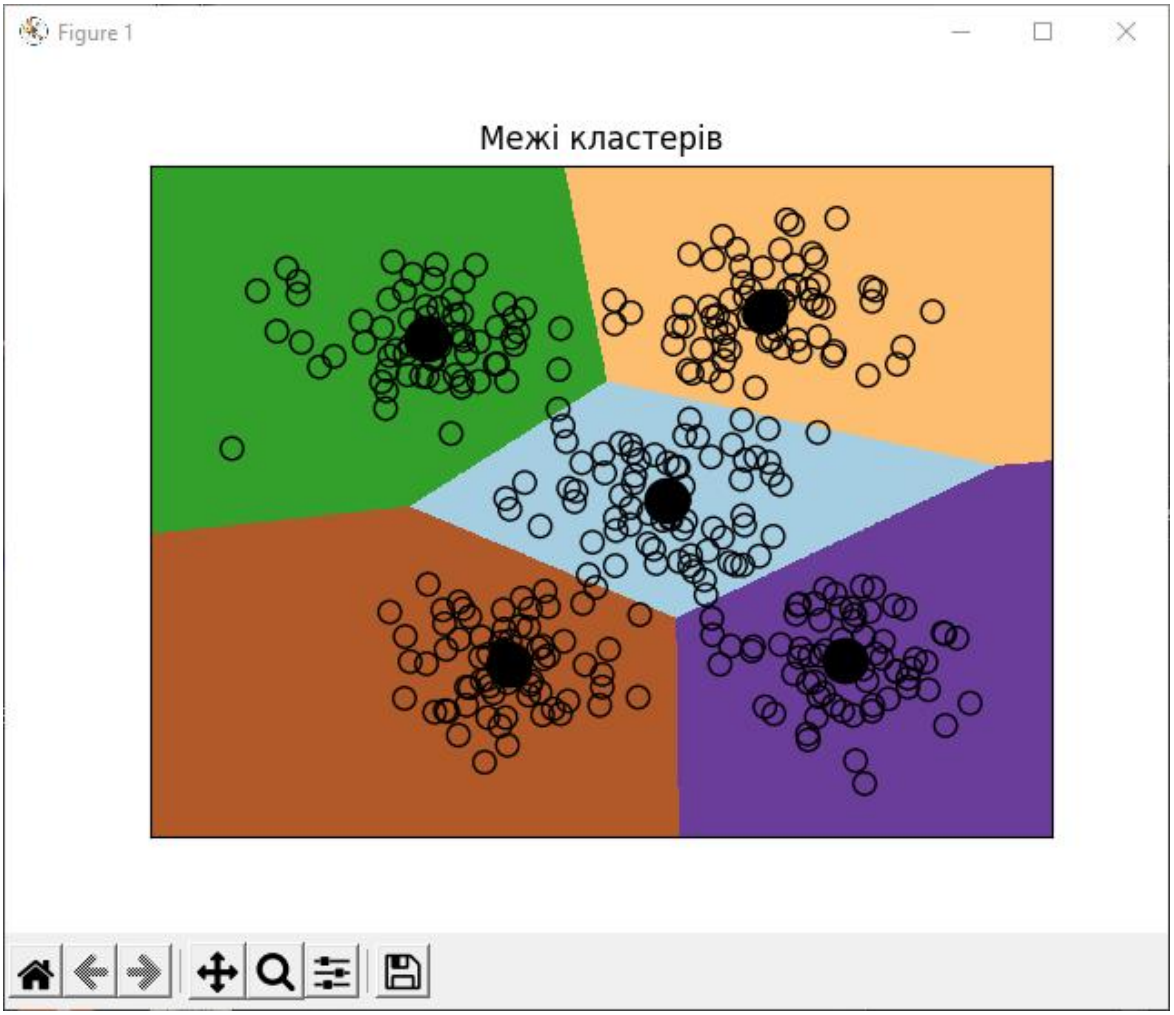

Рис. 4. Графік меж кластерів

Завдання 2.2. 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.

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

# Завантажуємо вхідні дані про ірис
iris = load_iris()
X = iris['data']
y = iris['target']

# Створюємо об'єкт KMeans з параметрами: кількість кластерів - 8, метод ініціалізації - k-means++ (для покращеного
# вибору центроїдів), максимальна кількість ітерацій - 300, значення відносної терпимості - 0.0001, вимкнений режим
# багатослівності, рандом використовується з пакету numpy, вихідні дані не змінюються, автоматичний вибір алгоритму
#
kmeans = KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None,
                copy_x=True, algorithm='auto')

# Створюємо об'єкт KMeans з параметрами: кількість кластерів - 5
# kmeans = KMeans(n_clusters=5)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Передбачення вихідних міток
y_kmeans = kmeans.predict(X)

# Графічне відображення точок та центрів кластерів
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)
```

Рис. 5. Код програми

```python
# Метод для пошуку кластерів, параметр rseed використовується для ініціалізації рандому
def find_clusters(X, n_clusters, rseed=2):
    # Ініціалізуємо рандом, перемішуємо отримані значення та отримуємо значення центрів
    rnd = np.random.RandomState(rseed)
    i = rnd.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # Знаходимо мінімальні відстані між точками та центрами
        labels = pairwise_distances_argmin(X, centers)

        # отримуємо середні значення масиву
        new_centers = np.array([X[np.array_equal(labels, i)]]).mean(0)

        for i in range(n_clusters):
            # якщо попередньо знайдені центри відповідать поточним - завершуємо цикл
            if np.array_equal(centers, new_centers):
                break
            centers = new_centers

        return centers, labels


# зображаємо нові знайдені точки кластерів при "зерні" рандому 2
centers, labels = find_clusters(X, 2)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

# зображаємо нові знайдені точки кластерів при "зерні" рандому 0
centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

# знаходимо точки кластерів імпортованим методом та показуємо їх
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```

Рис. 6. Код програми

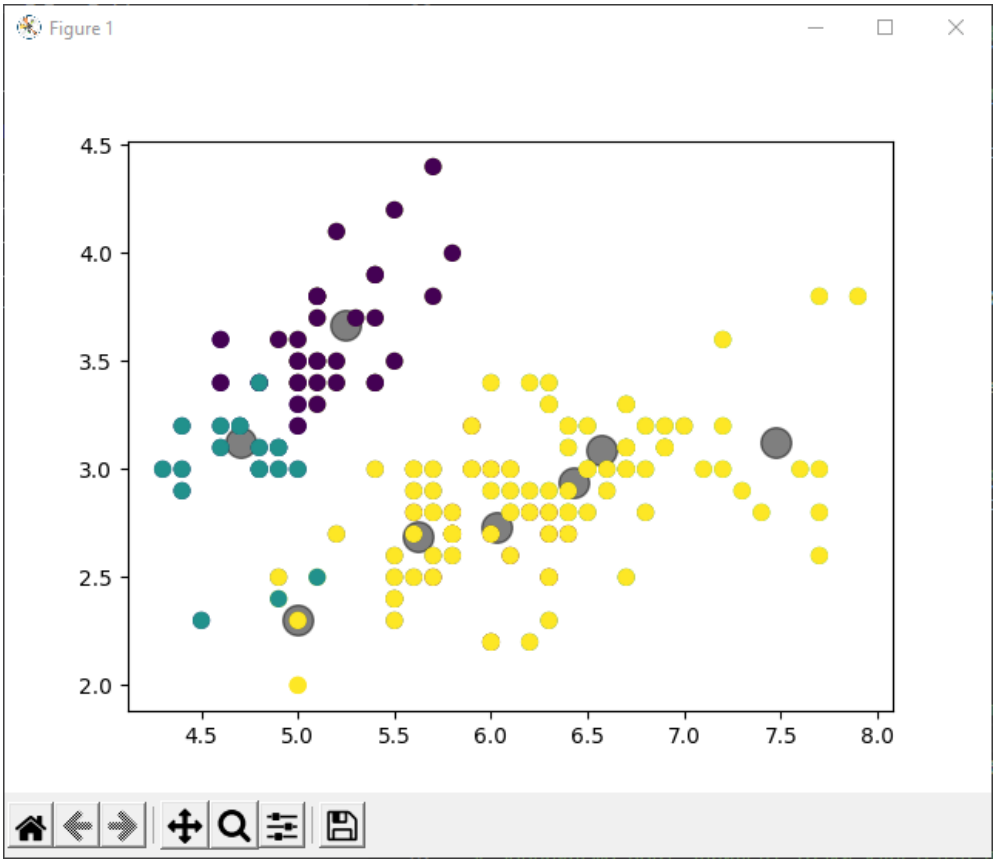

Рис. 7. Графік кластеризації при seed = 2

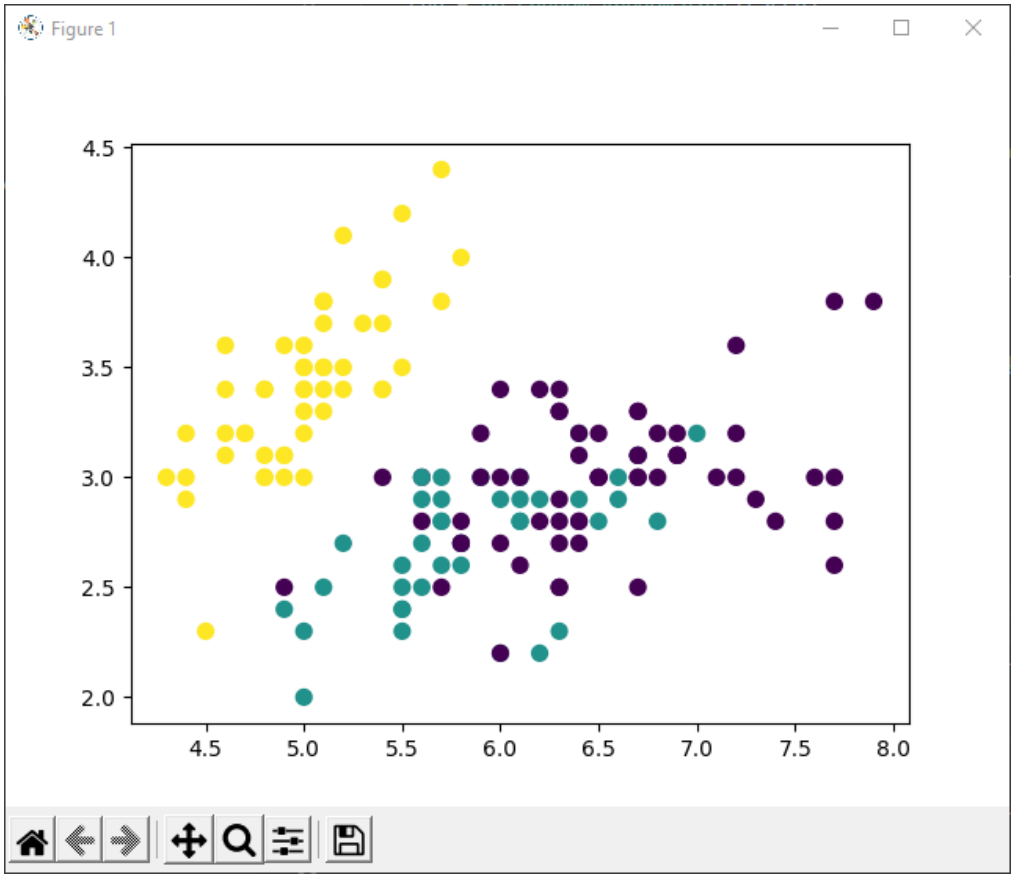

Рис. 8. Графік кластеризації при seed = 0

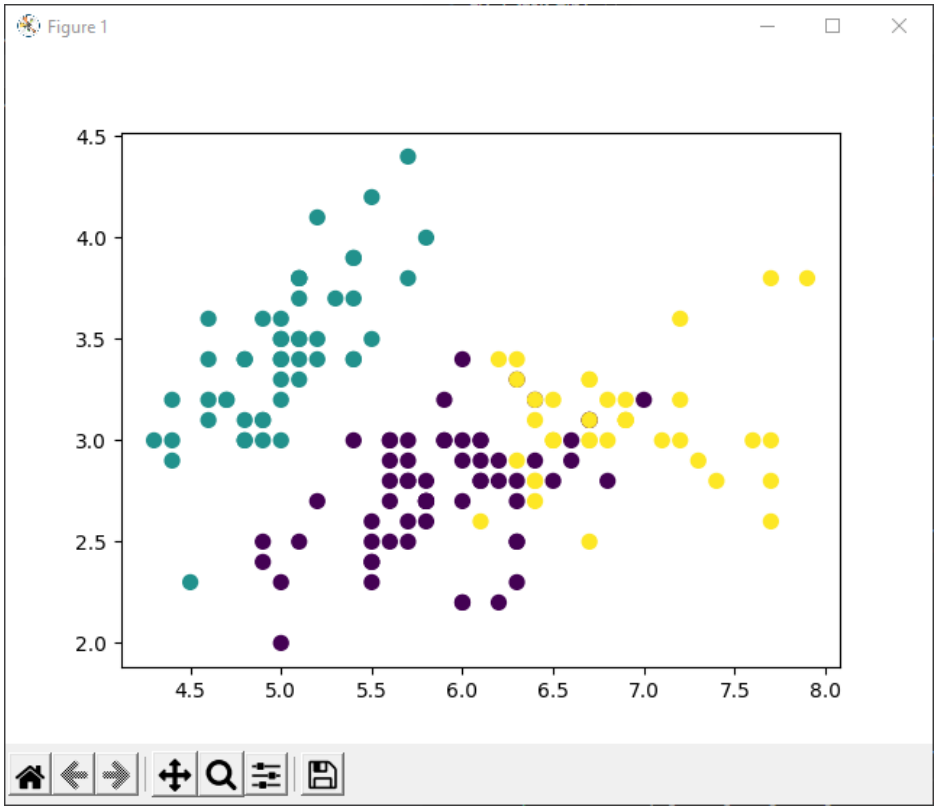

Рис. 9. Графік кластеризації вбудованого методу

З отриманих вище результатів можна зробити висновок, що найбільш подібні результати дає метод з seed = 0 та вбудований.

Завдання 2.3. 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.

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

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

# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

# 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("Centers", cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))
print("Number of clusters", num_clusters)

# Відображення на графіку точок та центрів кластерів
plt.figure()
markers = 'o*xvs'
colors = ['red', 'green', 'blue', 'orange', 'purple', 'yellow']
for i, marker in zip(range(num_clusters), markers):
    # Відображення на графіку точок, які належать поточному кластеру
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color=colors[i])

    # Відображення на графіку центру кластера
    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. 10. Код програми

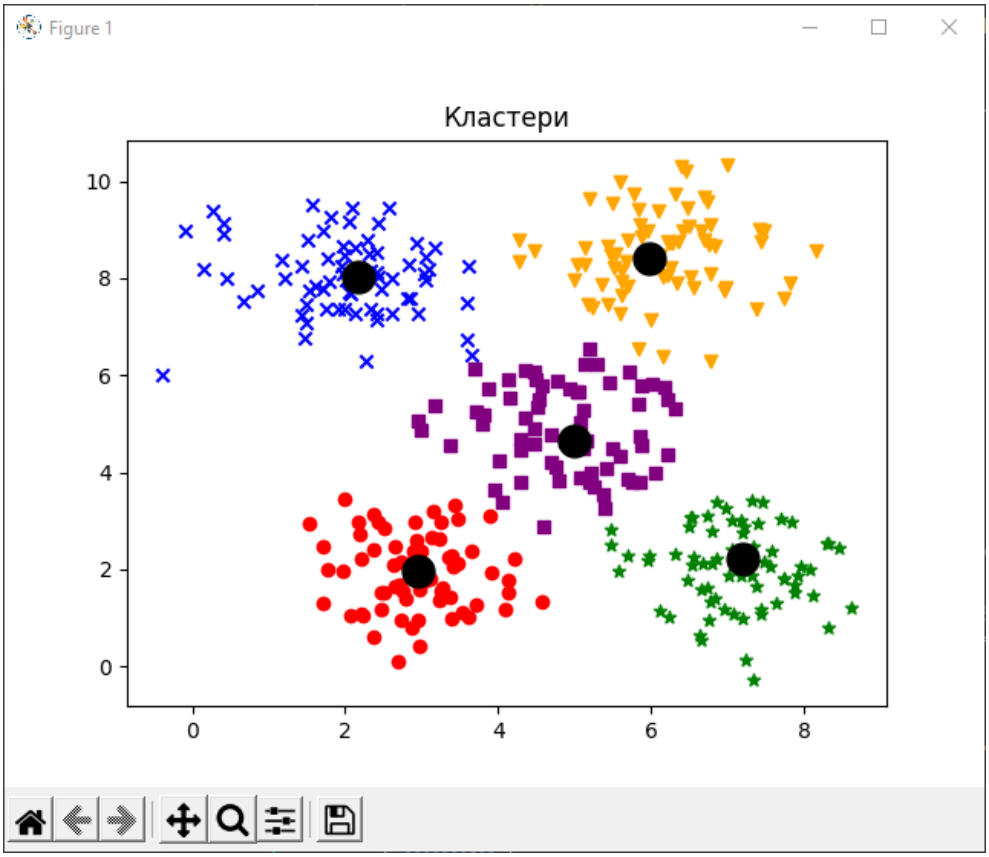

Рис. 11. Результат виконання програми

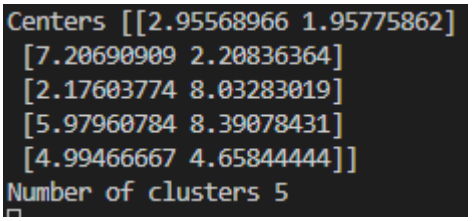

Рис. 12. Результат виконання програми

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

Використовуючи модель поширення подібності, знайти підгрупи серед учасників фондового ринку. У якості керуючих ознак будемо використовувати варіацію котирувань між відкриттям і закриттям біржі. Використовувати файл вхідних даних фондового ринку, що доступний в бібліотеці matplotlib. Прив'язки символічних позначень компаній до повних назв містяться у файлі company_symbol_mapping.json.

Пакет matplotlib.finance зараз є deprecated.

Висновок: в ході виконання лабораторної роботи було досліджено методи неконтрольованої класифікації даних у машинному навчанні за допомогою мови програмування Python.

GitHub