**ЛАБОРАТОРНА РОБОТА № 7**

ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

**Хід роботи:**

Посилання на GitHub:

**Завдання 1**

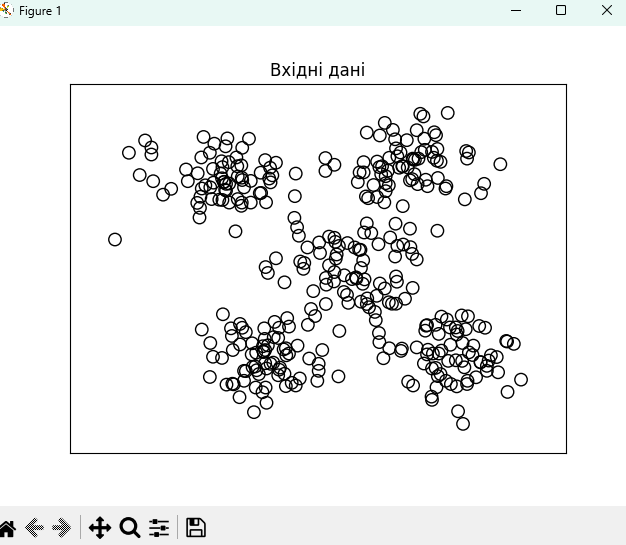
Кластеризація даних за допомогою методу k-середніх.

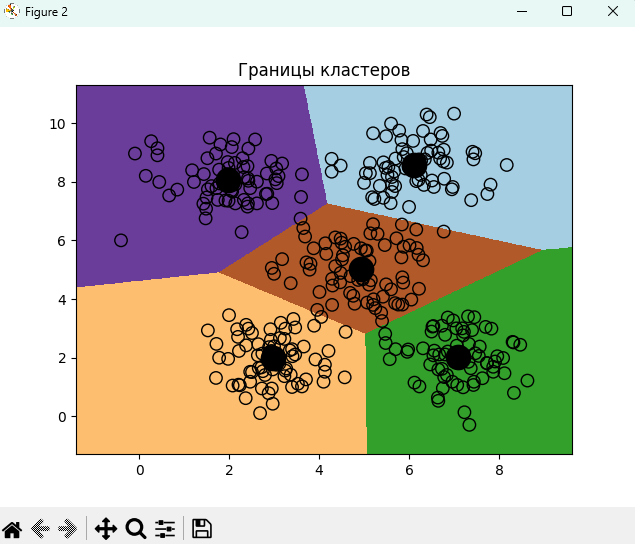
**Програмний код:**

import numpy as np  
from sklearn.cluster import KMeans  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
num\_clusters = 5  
  
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(())  
  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
kmeans.fit(X)  
  
step\_size = 0.01  
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('Границы кластеров')  
plt.show()

Результат виконання:



****

**Висновок:**

Після виконання коду ми можемо зрозуміти, що наші дані складаються з п'яти груп.

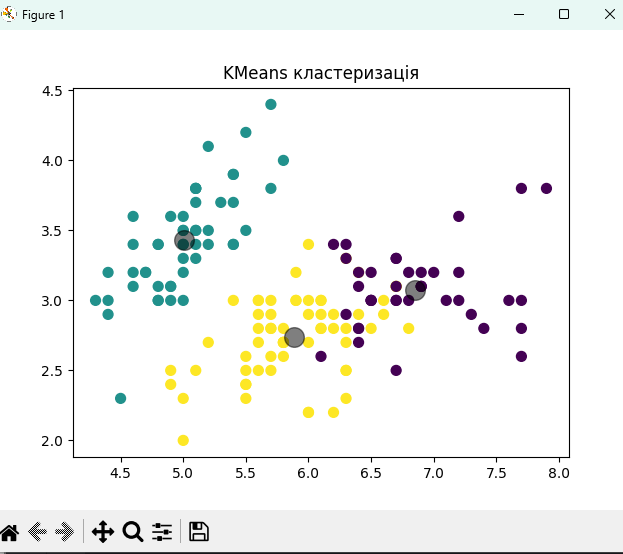
**Завдання 2.**

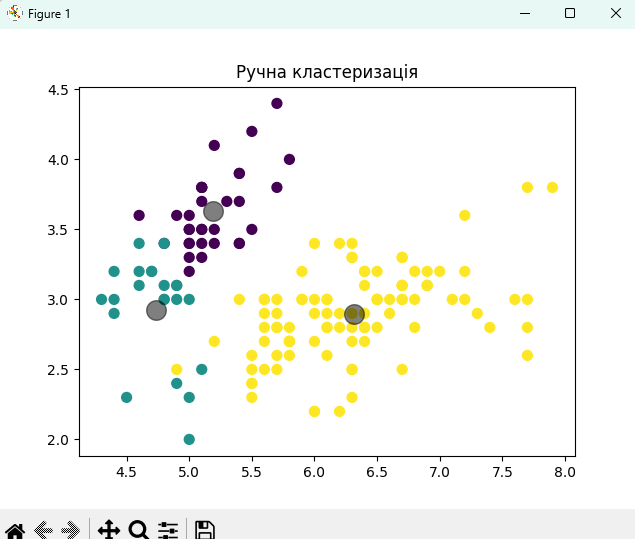
Кластеризація K-середніх для набору даних Iris

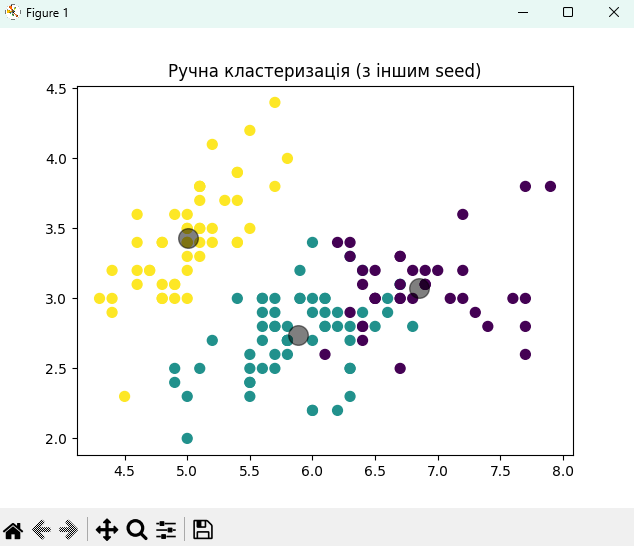
**Програмний код:**

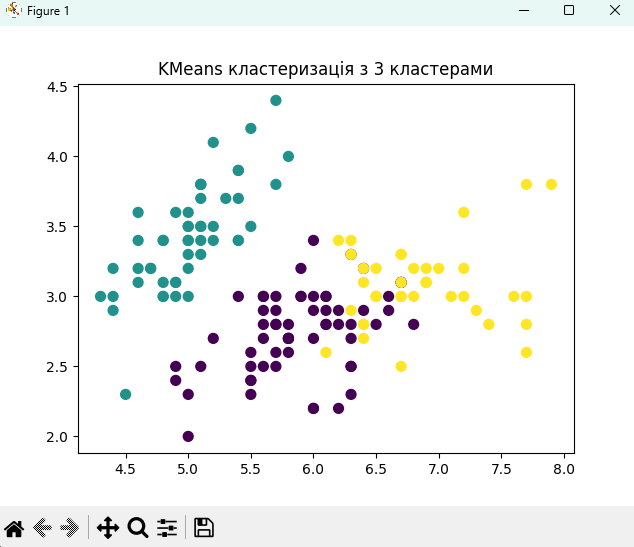
from sklearn.svm import SVC  
from sklearn.metrics import pairwise\_distances\_argmin  
from sklearn.datasets import load\_iris  
from sklearn.cluster import KMeans  
import numpy as np  
  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
iris = load\_iris()  
X = iris['data']  
y = iris['target']  
  
kmeans = KMeans(n\_clusters=3, random\_state=42)  
kmeans.fit(X)  
y\_kmeans = kmeans.predict(X)  
  
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")  
plt.show()  
  
def find\_clusters(X, n\_clusters, rseed=2):  
 rng = np.random.RandomState(rseed)  
 i = rng.permutation(X.shape[0])[:n\_clusters]  
 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)])  
 if np.all(centers == new\_centers):  
 break  
 centers = new\_centers  
 return centers, labels  
  
centers, labels = find\_clusters(X, 3)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
plt.title("Ручна кластеризація")  
plt.show()  
  
centers, labels = find\_clusters(X, 3, rseed=0)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
plt.title("Ручна кластеризація (з іншим seed)")  
plt.show()  
  
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()

Результат виконання:





****

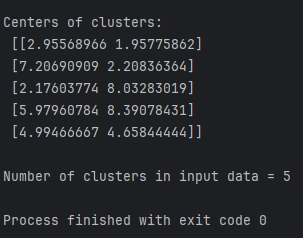
****

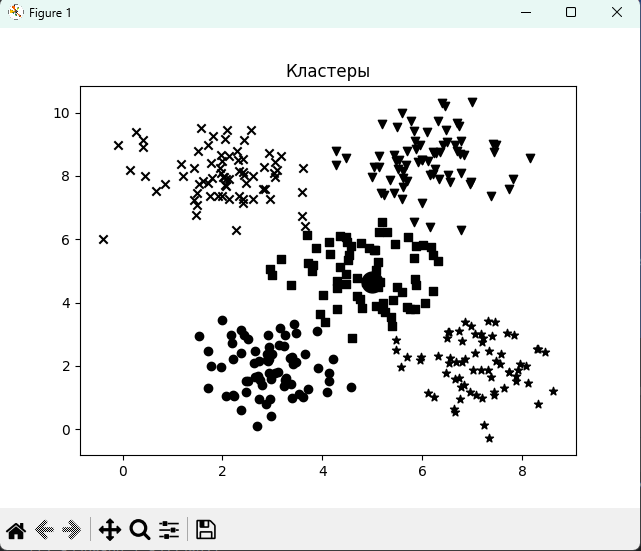
**Завдання 3.** Оцінка кількості кластерів з використанням методузсуву середнього

**Програмний код:**

import numpy as np  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from itertools import cycle  
  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
bandwidth\_X = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))  
meanshift\_model = MeanShift(bandwidth=bandwidth\_X, bin\_seeding=True)  
meanshift\_model.fit(X)  
cluster\_centers = meanshift\_model.cluster\_centers\_  
print('\nCenters of clusters: \n', cluster\_centers)  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print("\nNumber of clusters in input data =", num\_clusters)  
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='black')  
cluster\_center = cluster\_centers[i]  
plt.plot(cluster\_center[0], cluster\_center[1], marker='o',  
 markerfacecolor='black', markeredgecolor='black', markersize=15)  
plt.title('Кластеры')  
plt.show()

Результат виконання:

****

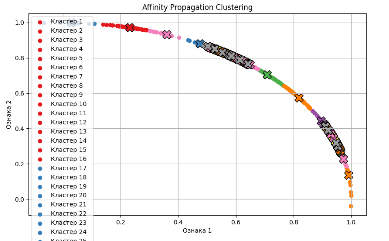
****

**Висновок:**

Програма дійсно змогла обчислити кількість кластерів, як і зображено на графіку

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

import numpy as np  
import pandas as pd  
from sklearn.cluster import AffinityPropagation  
from sklearn.preprocessing import normalize  
import matplotlib.pyplot as plt  
  
# Завантаження даних із файлу data\_clustering.txt  
data = np.loadtxt('data\_clustering.txt', delimiter=',') # Передбачається, що числа розділені комами  
  
# Обчислення різниць між координатами (у даному разі залежить від формату даних)  
# Якщо це двовимірні дані, їх можна кластеризувати без обчислення різниць  
normalized\_data = normalize(data)  
  
# Створення моделі кластеризації  
affinity\_model = AffinityPropagation(random\_state=0)  
affinity\_model.fit(normalized\_data)  
  
# Отримання результатів  
cluster\_centers\_indices = affinity\_model.cluster\_centers\_indices\_  
labels = affinity\_model.labels\_  
  
# Виведення результатів  
n\_clusters = len(cluster\_centers\_indices)  
print(f"Кількість кластерів: {n\_clusters}")  
print("Кластери та їх центри:")  
for i, center in enumerate(cluster\_centers\_indices):  
 print(f"Кластер {i+1}: Центр - {normalized\_data[center]}")  
  
# Візуалізація результатів  
plt.figure(figsize=(10, 6))  
colors = plt.cm.Set1(np.linspace(0, 1, n\_clusters))  
  
for k, col in zip(range(n\_clusters), colors):  
 class\_members = labels == k  
 cluster\_center = normalized\_data[cluster\_centers\_indices[k]]  
 plt.scatter(normalized\_data[class\_members, 0], normalized\_data[class\_members, 1], color=col, label=f'Кластер {k+1}')  
 plt.scatter(cluster\_center[0], cluster\_center[1], s=200, color=col, edgecolors='k', marker='X')  
  
plt.title('Affinity Propagation Clustering')  
plt.xlabel('Ознака 1')  
plt.ylabel('Ознака 2')  
plt.legend()  
plt.grid(True)  
plt.show()

****

**Висновок:** Під час лабораторної роботи я навчилася використовувати спеціалізовані бібліотеки та мови програмування Python та дослідила методи неконтрольованої класифікації даних у машинному навчанні.