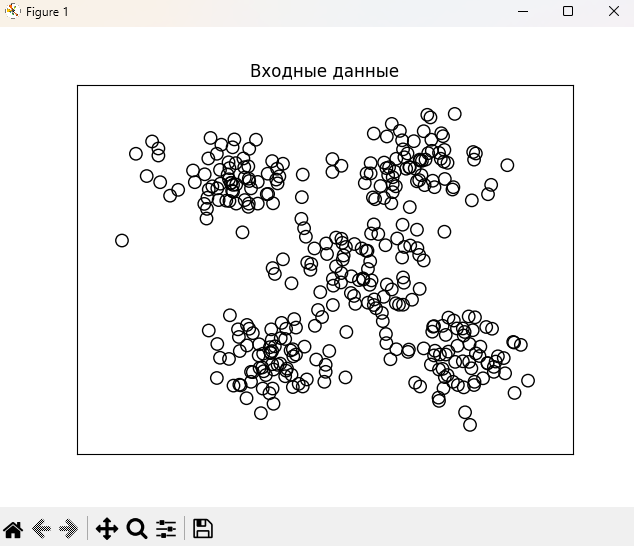
Системи штучного інтелекту.

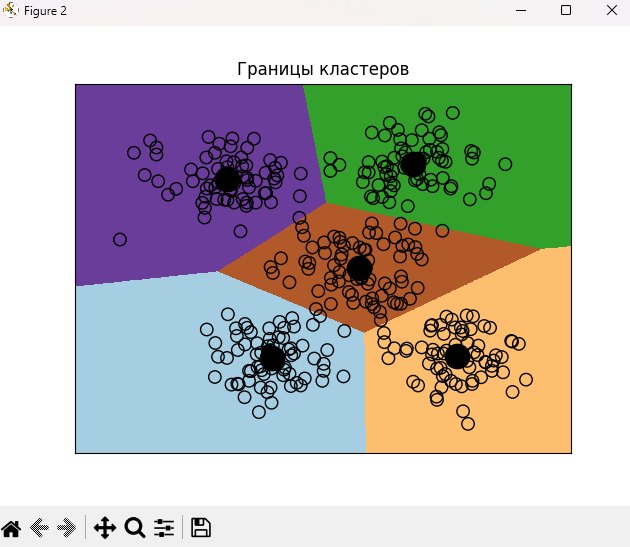
Лабораторна робота 7.Федорович Дмитро ІПЗ-21-3

<https://github.com/Dmitrij3/lab7AI>

**Завдання №1.** Кластеризація даних за допомогою методу k-середніх

import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib  
from sklearn.cluster import KMeans  
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('Входные данные')  
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\_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')  
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('Границы кластеров')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

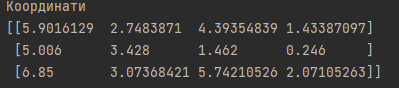
****

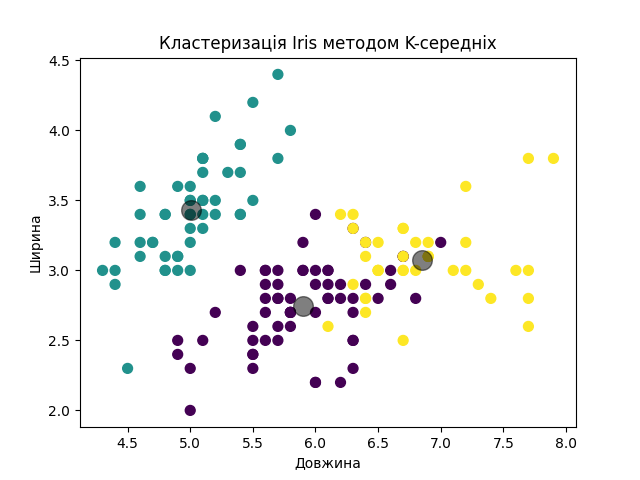
****

Аналіз коду підтверджує, що дані ефективно кластеризуються, і кожен кластер має чітко окреслені межі. Використання алгоритму кластеризації KMeans дозволяє виявити зони високої концентрації точок у кожному кластері. Візуалізація демонструє не лише чіткий розподіл точок між кластерами, але й правильне визначення центрів кластерів, що відповідають зонам найбільшої щільності даних. Це свідчить про високу якість кластеризації та відповідність моделі вихідним даним.

**Завдання №2.** Кластеризація K-середніх для набору даних Iris

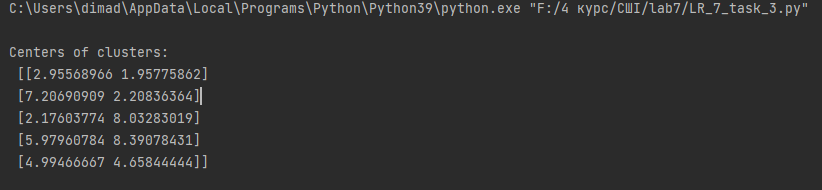
from sklearn.cluster import KMeans  
from sklearn.datasets import load\_iris  
import matplotlib.pyplot as plt  
  
iris = load\_iris()  
X = iris['data']  
y = iris['target']  
  
  
kmeans = KMeans(  
 n\_clusters=3,  
 init='k-means++',  
 n\_init=10,  
 max\_iter=300,  
 random\_state=42  
)  
kmeans.fit(X)  
  
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.xlabel('Довжина')  
plt.ylabel('Ширина')  
plt.title('Кластеризація Iris методом K-середніх')  
plt.show()  
print("Координати")  
print(kmeans.cluster\_centers\_)

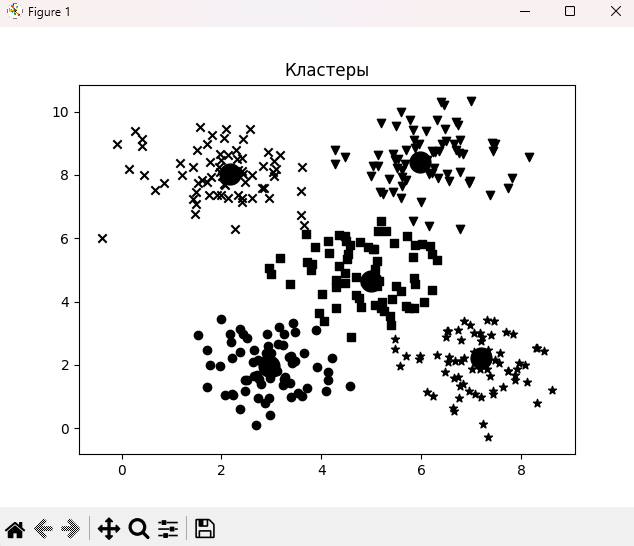




**Завдання №3.** Оцінка кількості кластерів з використанням методу зсуву середнього

import numpy as np  
import matplotlib.pyplot as plt  
import matplotlib  
matplotlib.use('TkAgg')  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
  
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("\nNumberof 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()





**Завдання №4.** Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності

import json  
import numpy as np  
from sklearn.covariance import GraphicalLassoCV  
from sklearn.cluster import AffinityPropagation  
import yfinance as yf   
  
input\_file = 'company\_symbol\_mapping.json'  
  
with open(input\_file, 'r') as f:  
 company\_symbols\_map = json.loads(f.read())  
  
symbols, names = np.array(list(company\_symbols\_map.items())).T  
  
start\_date = '2003-07-03'  
end\_date = '2007-05-04'  
  
quotes = []  
valid\_symbols = []  
for symbol in symbols:  
 try:  
 data = yf.download(symbol, start=start\_date, end=end\_date)  
 if not data.empty and 'Open' in data.columns and 'Close' in data.columns:  
 quotes.append(data)  
 valid\_symbols.append(symbol)  
 else:  
 print(f"Skipping symbol {symbol} due to missing data.")  
 except Exception as e:  
 print(f"Error fetching data for symbol {symbol}: {e}")  
  
symbols = np.array(valid\_symbols)  
names = np.array([company\_symbols\_map[symbol] for symbol in symbols])  
  
opening\_quotes = np.array([quote['Open'].values for quote in quotes]).astype(np.float64)  
closing\_quotes = np.array([quote['Close'].values for quote in quotes]).astype(np.float64)  
  
quotes\_diff = closing\_quotes - opening\_quotes  
  
X = quotes\_diff.copy().T  
X /= X.std(axis=0)  
  
edge\_model = GraphicalLassoCV()  
with np.errstate(invalid='ignore'):  
 edge\_model.fit(X)  
  
affinity\_model = AffinityPropagation()  
affinity\_model.fit(edge\_model.covariance\_)  
  
labels = affinity\_model.labels\_  
num\_labels = labels.max()  
  
for i in range(num\_labels + 1):  
 print("Cluster", i + 1, "==>", ', '.join(names[labels == i]))

