**Лабораторна робота №7**

Дослідження Методів Неконтрольованого Навчання

**Виконав:** ІПЗ-21-3 Осипчук Антон Олексійович

**Github:** <https://github.com/AntonOsypchuk1/ai_lab/tree/main/lab7>

**Завдання 1.** Кластеризація даних за допомогою методу k-середніх

Лістинг програми:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn import metrics

X = np.loadtxt('data/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('Input data')

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('Boundaries of clusters')

plt.xlim(x\_min, x\_max)

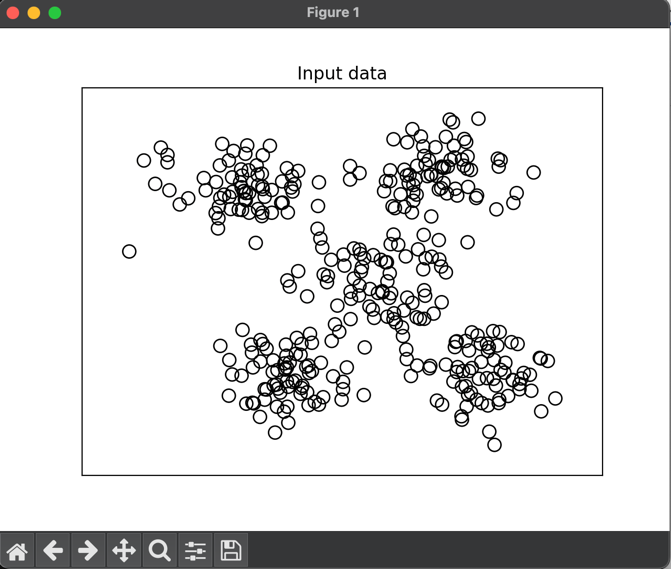
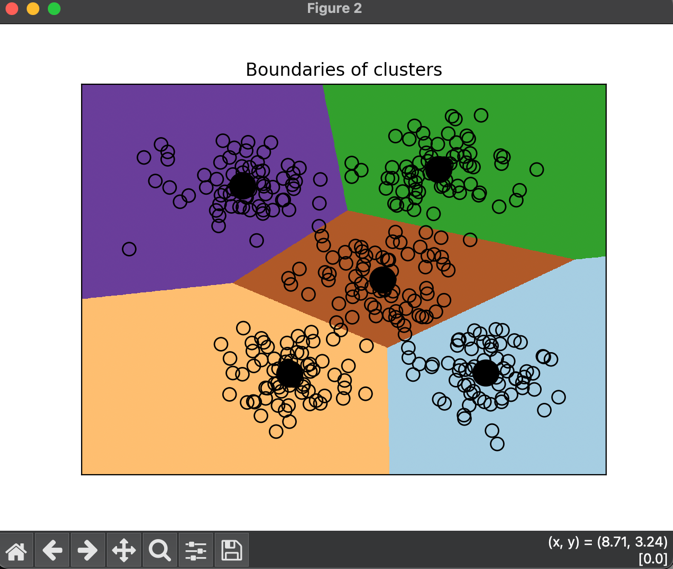
plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

plt.show()

Результат:



**Завдання 2.** Кластеризація K-середніх для набору даних Iris

Лістинг програми:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

from sklearn.metrics import pairwise\_distances\_argmin

from sklearn.datasets import load\_iris

# Завантаження даних

iris = load\_iris()

X = iris['data']

y = iris['target']

# Використання KMeans для кластеризації

kmeans = KMeans(n\_clusters=5, random\_state=0)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

# Візуалізація результатів

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 Clustering with 5 Clusters")

plt.show()

# Визначення власної функції кластеризації

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

# Використання функції для кластеризації

centers, labels = find\_clusters(X, 3)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

plt.title("Custom Clustering with 3 Clusters (rseed=2)")

plt.show()

centers, labels = find\_clusters(X, 3, rseed=0)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

plt.title("Custom Clustering with 3 Clusters (rseed=0)")

plt.show()

# Використання KMeans для кластеризації з іншими параметрами

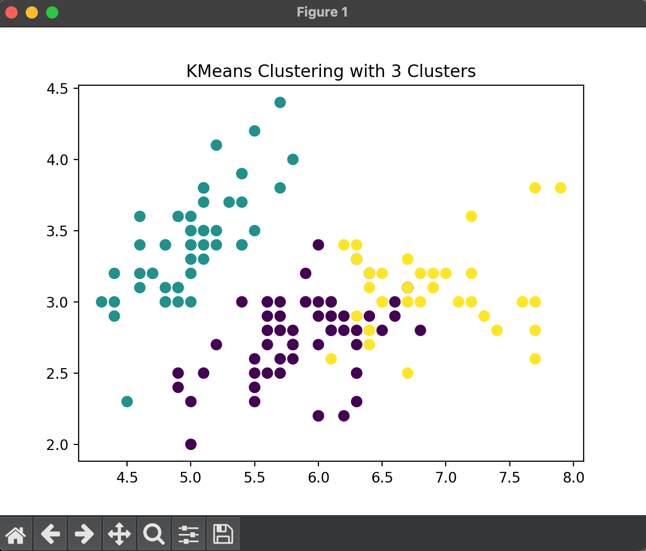
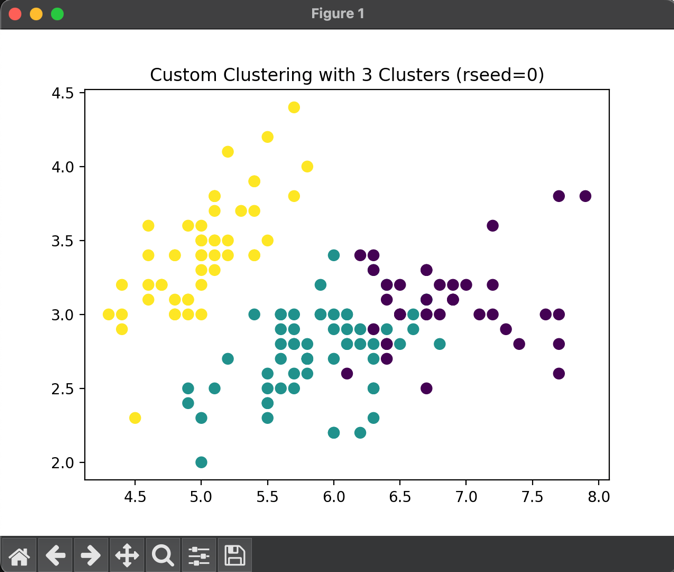
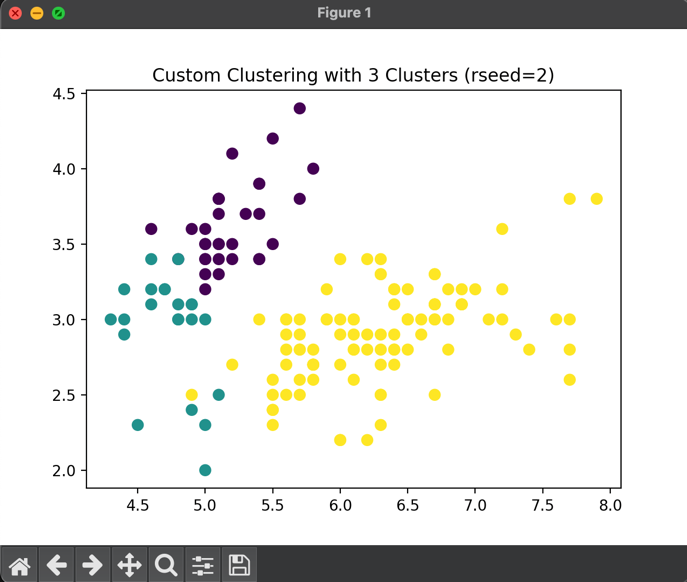
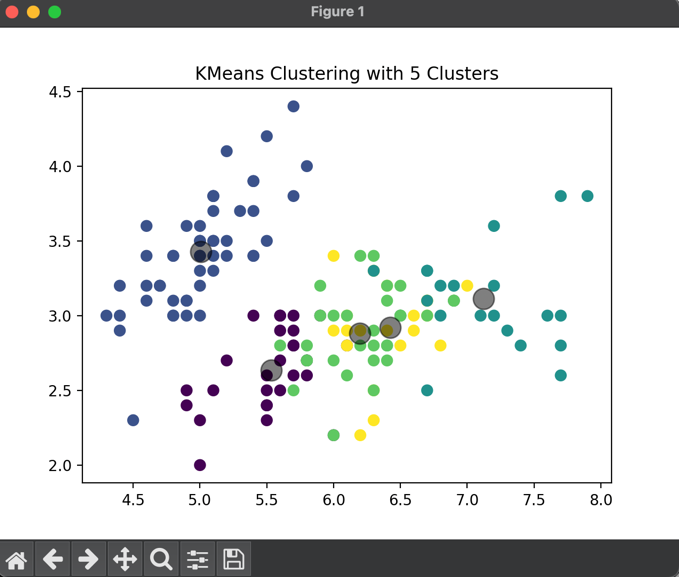
labels = KMeans(3, random\_state=0).fit\_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')

plt.title("KMeans Clustering with 3 Clusters")

plt.show()

Результат:



**Завдання 3.** Оцінка кількості кластерів з використанням методу зсуву середнього

Лістинг програми:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import MeanShift, estimate\_bandwidth

from itertools import cycle

X = np.loadtxt('data/data\_clustering.txt', delimiter = ',')

bandwidth = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))

meanshift\_model = MeanShift(bandwidth=bandwidth, 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]

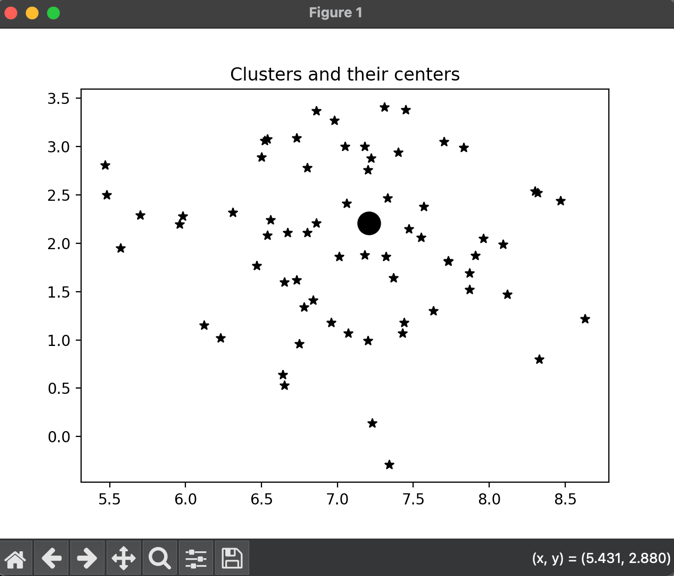
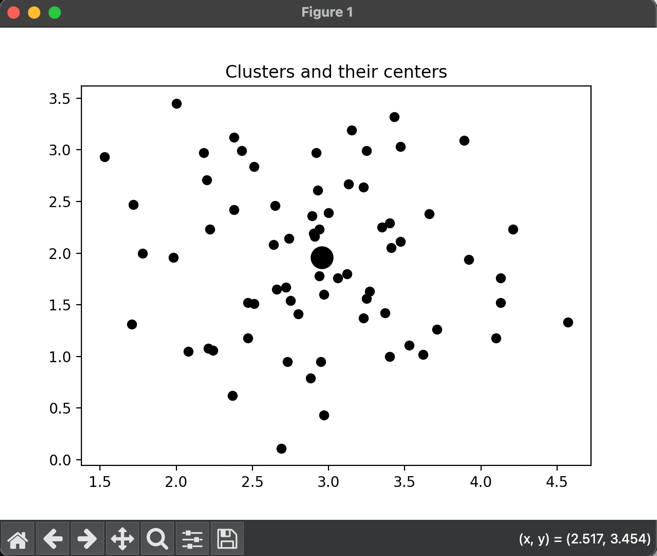
plt.plot(cluster\_center[0], cluster\_center[1], marker='o',

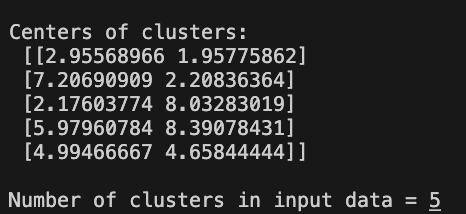
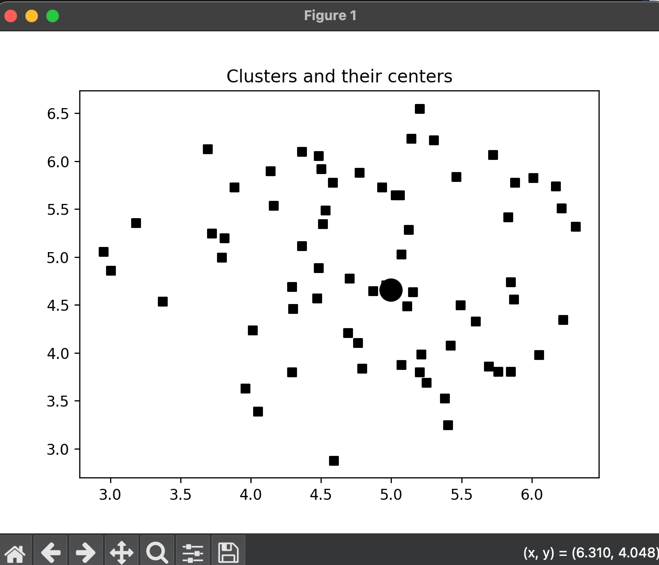
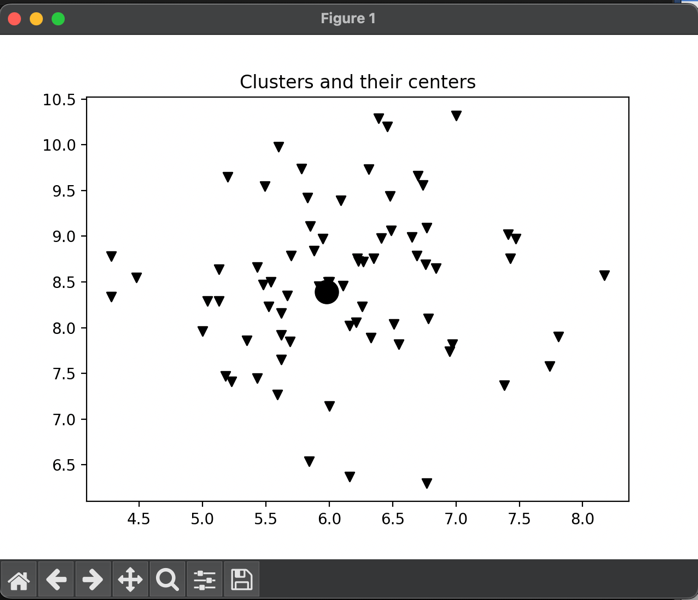
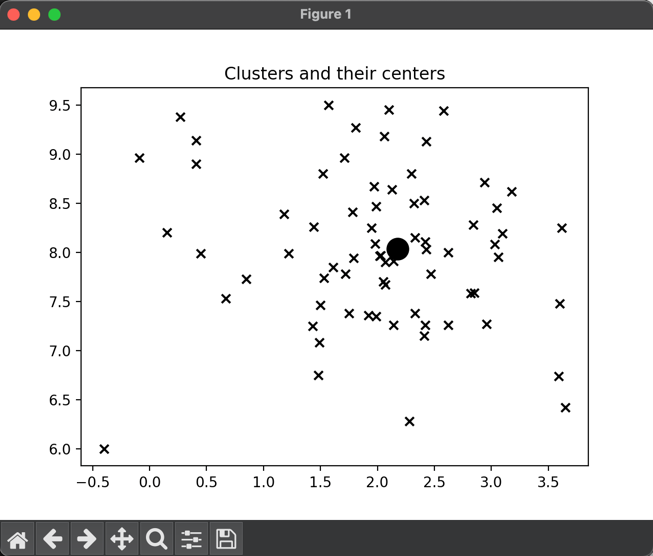
markerfacecolor='black', markeredgecolor='black', markersize=15)

plt.title('Clusters and their centers')

plt.show()

Результат:

****

****

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

Лістинг програми:

import datetime

import json

import numpy as np

import matplotlib.pyplot as plt

from sklearn import covariance, cluster

from matplotlib.finance import quotes\_historical\_yahoo\_ochl as quotes\_yahoo

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 = datetime.datetime(2003, 7, 3)

end\_date = datetime.datetime(2007, 5, 4)

quotes = [quotes\_yahoo(symbol, start\_date, end\_date,

asobject=True) for symbol in symbols]

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)

quotes\_diff = closing\_quotes - opening\_quotes

X = quotes\_diff.copy().T

X /= X.std(axis=0)

edge\_model = covariance.GraphLassoCV()

with np.errstate(invalid='ignore'):

edge\_model.fit(X)

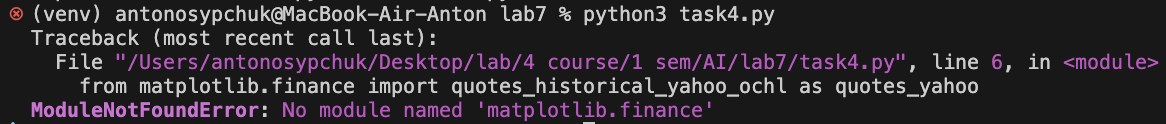
\_, labels = cluster.affinity\_propagation(edge\_model.covariance\_)

num\_labels = labels.max()

for i in range(num\_labels + 1):

print("Cluster", i+1, "==>", ', '.join(names[labels == i]))

Результат:

****