**Лабораторна робота №7**

**ІПЗ-21-5 Пархомчук Іван**

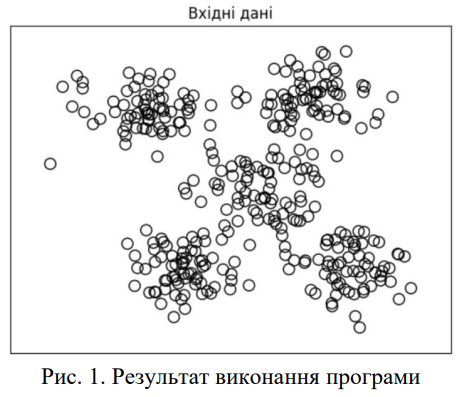
**Мета:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи неконтрольованої класифікації даних у машинному навчанні.

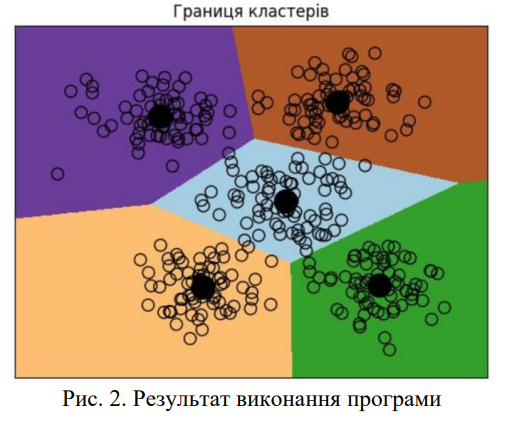
**Завдання 1.** Кластеризація даних за допомогою методу k-середніх

**Лістинг коду**

import numpy as np  
import matplotlib  
  
matplotlib.use('TkAgg')  
from matplotlib import 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  
  
# 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 КМеаns  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
  
# Навчання моделі кластеризації КМеаns  
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()])  
  
# 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('Границя кластерів')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

**Результат виконання**

****

****

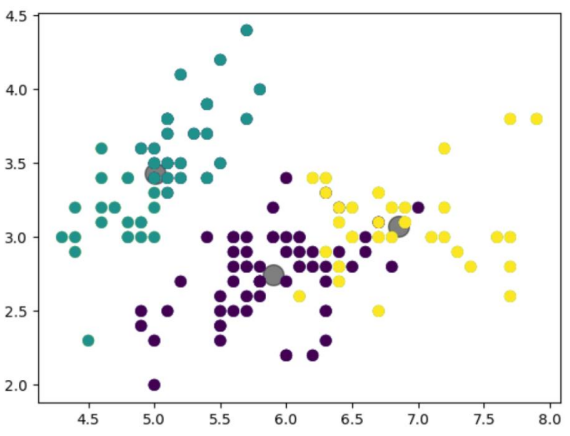
У результаті виконання програмного коду вдалося досягти доволі хороших результатів: більшість точок знаходяться повністю в заданій області, а визначені центроїди точно відображають найбільшу концентрацію точок у відповідних кластерах.

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

**Лістинг програми**

from sklearn.svm import SVC  
from sklearn.metrics import pairwise\_distances\_argmin  
import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.cluster import KMeans  
import matplotlib  
  
matplotlib.use('TkAgg')  
from matplotlib import pyplot as plt  
  
iris = load\_iris()  
X = iris['data']  
y = iris['target']  
  
# Створення об'єкту КМеаns  
kmeans = KMeans(n\_clusters=3, init='k-means++', n\_init=10)  
  
# Навчання моделі кластеризації КМеаns  
kmeans.fit(X)  
  
# Передбачення вихідних міток  
y\_kmeans = kmeans.predict(X)  
  
# Відображення вхідних точок  
plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap='viridis')  
  
# Відображення центрів кластерів  
centers = kmeans.cluster\_centers\_  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
  
  
def find\_clusters(X, n\_clusters, rseed=2):  
 # Довільне обрання кластерів  
 rng = np.random.RandomState(rseed)  
 i = rng.permutation(X.shape[0])[:n\_clusters]  
 centers = X[i]  
  
 while True:  
 # Призначення міток на основі найближчого центру  
 labels = pairwise\_distances\_argmin(X, centers)  
  
 # Знаходження нових центрів за середніми точками  
 new\_centers = np.array([X[labels == i].mean(0)  
 for i in range(n\_clusters)])  
  
 # Перевірка на збіжність  
 if np.all(centers == new\_centers):  
 break  
 centers = new\_centers  
  
 return centers, labels  
  
  
# Відображення точок  
centers, labels = find\_clusters(X, 3)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
labels = KMeans(3, random\_state=0).fit\_predict(X)  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()

**Результат виконання**

****

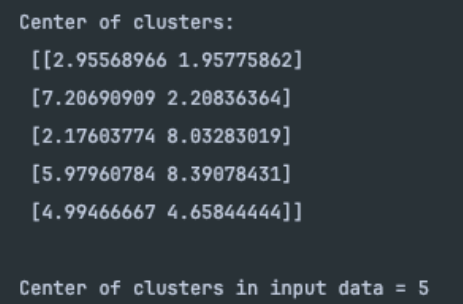
Результати виконання програмного коду виявилися середніми. Проте визначені центроїди відображають максимальну концентрацію точок у кожному кластері.

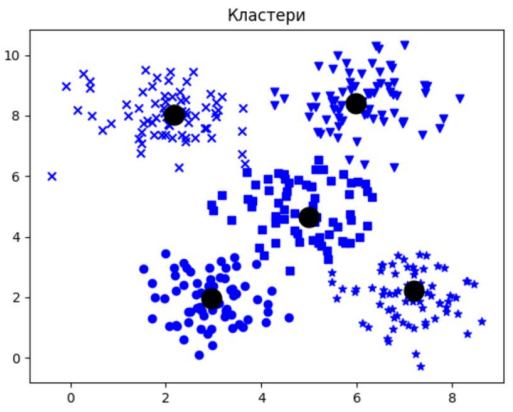
**Завдання 3.** Оцінка кількості кластерів з використанням методу зсуву середнього

**Лістинг коду**

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 Х  
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('\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()

**Результат виконання коду**





Результати підтверджують ефективність методу кластеризації за допомогою зсуву середнього. Було успішно виділено 5 кластерів, що збігається з кількістю, заданою вручну в попередніх завданнях.

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

**Лістинг коду**

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(2003, 7, 3)  
end\_date = datetime.datetime(2007, 5, 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)  
  
# 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]))

**Висновок:** використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи неконтрольованої класифікації даних у машинному навчанні.