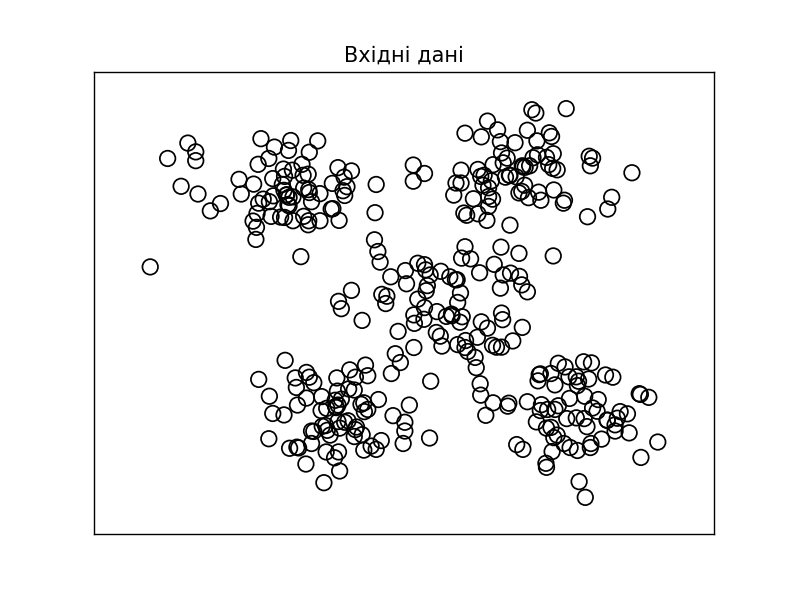
**Лабораторна робота № 7**

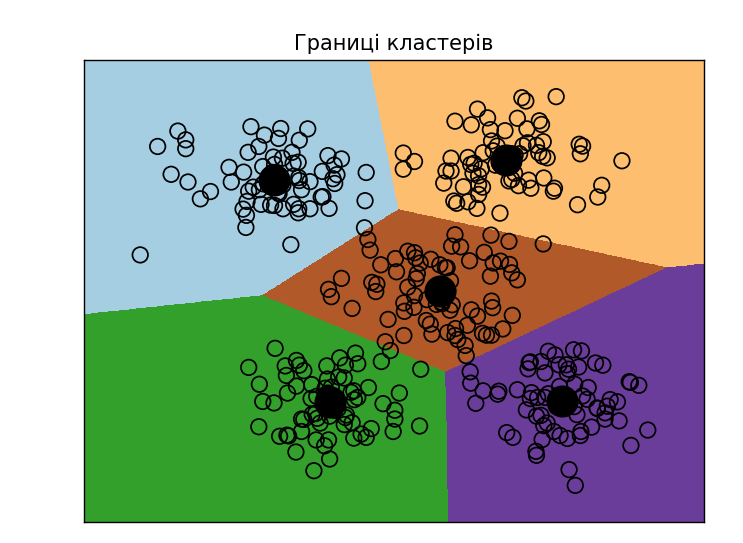
***ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ***

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

**Хід роботи:**

**Завдання 2.1. Кластеризація даних за допомогою методу k-середніх:**

****

****

Лістинг:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

X = np.loadtxt("data\_clustering.txt", *delimiter*=",")

num\_clusters = 5

*def* set\_plot\_limits(*data*, *padding*=1):

    x\_min, x\_max = *data*[:, 0].min() - *padding*, *data*[:, 0].max() + *padding*

    y\_min, y\_max = *data*[:, 1].min() - *padding*, *data*[:, 1].max() + *padding*

    return x\_min, x\_max, y\_min, y\_max

x\_min, x\_max, y\_min, y\_max = set\_plot\_limits(X)

plt.figure()

plt.scatter(X[:, 0], X[:, 1], *marker*='o', *facecolors*='none', *edgecolors*="black", *s*=80)

plt.title("Вхідні дані")

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

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

plt.xticks([])

plt.yticks([])

plt.show()

kmeans = KMeans(*init*="k-means++", *n\_clusters*=num\_clusters, *n\_init*=10, *random\_state*=42)

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))

grid\_points = np.c\_[x\_vals.ravel(), y\_vals.ravel()]

output = kmeans.predict(grid\_points).reshape(x\_vals.shape)

plt.figure()

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,

*linewidth*=4,

*color*='black',

*facecolors*='black'

)

plt.title("Границі кластерів")

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

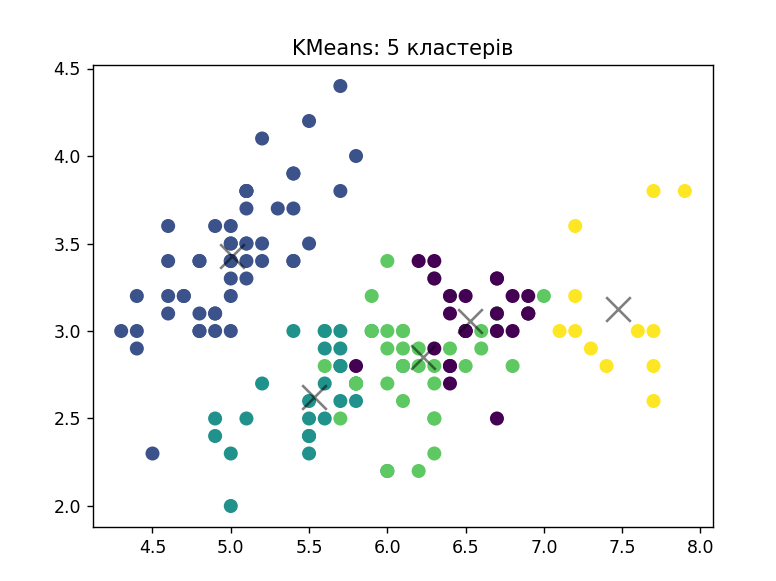
plt.ylim(y\_min, y\_max)

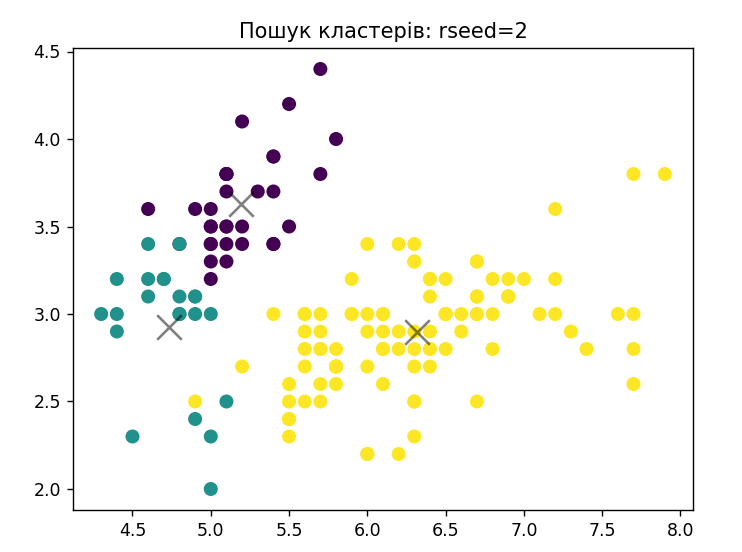
plt.xticks([])

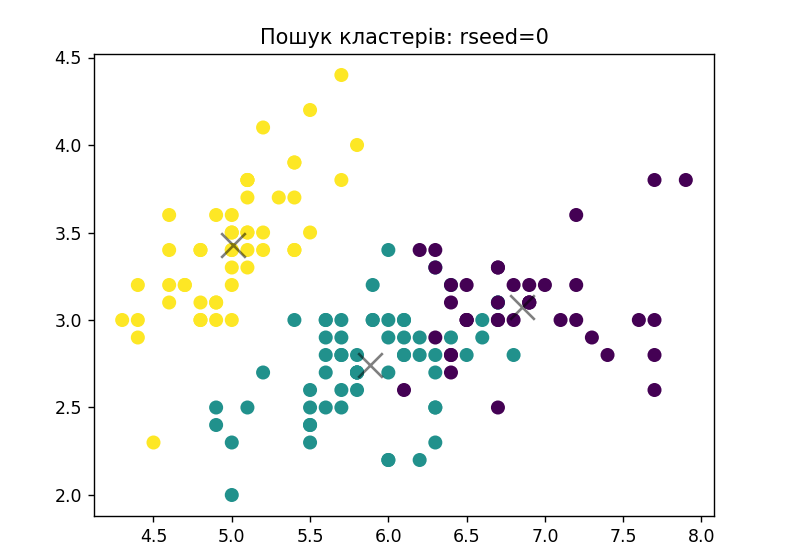
plt.yticks([])

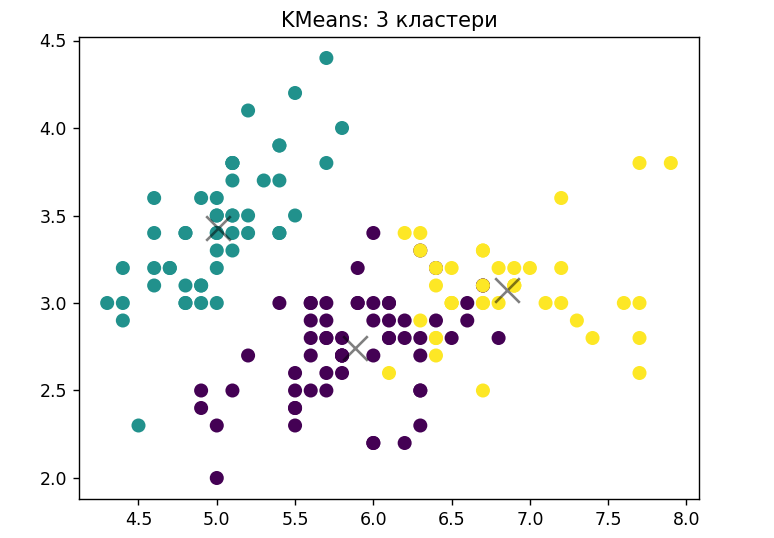
plt.show()

**Завдання 2.2. Кластеризація K-середніх для набору даних Iris:**









Лістинг:

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

iris = load\_iris()

X = iris['data']

y = iris['target']

kmeans = KMeans(*n\_clusters*=5, *init*='k-means++', *n\_init*=10, *max\_iter*=300, *random\_state*=42)

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

*def* plot\_clusters(*X*, *labels*, *centers*=None, *title*="Кластери"):

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

    if *centers* is not None:

        plt.scatter(*centers*[:, 0], *centers*[:, 1], *c*='black', *s*=200, *alpha*=0.5, *marker*='x')

    plt.title(*title*)

    plt.show()

plot\_clusters(X, y\_kmeans, *centers*=kmeans.cluster\_centers\_, *title*="KMeans: 5 кластерів")

*def* find\_clusters(*X*, *n\_clusters*, *rseed*=2):

    rng = np.random.RandomState(*rseed*)

    initial\_indices = rng.permutation(*X*.shape[0])[:*n\_clusters*]

    centers = *X*[initial\_indices]

    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

for rseed in [2, 0]:

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

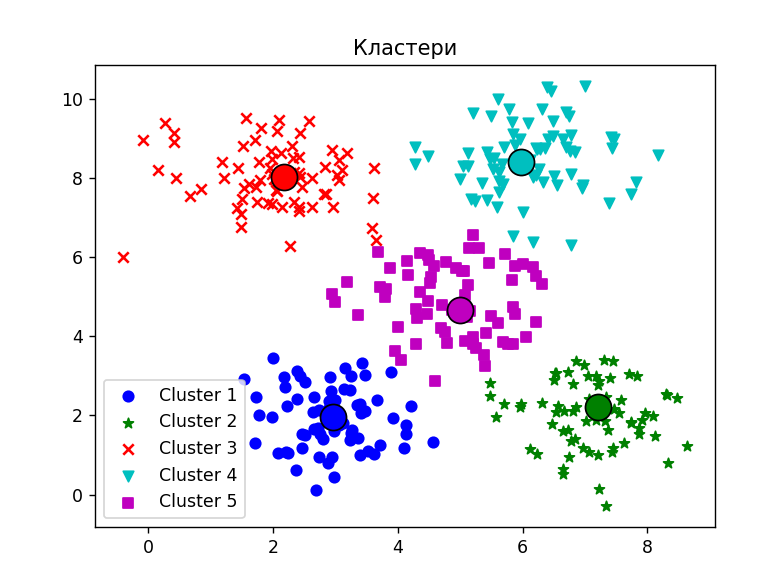
    plot\_clusters(X, labels, *centers*=centers, *title*=*f*"Пошук кластерів: rseed={rseed}")

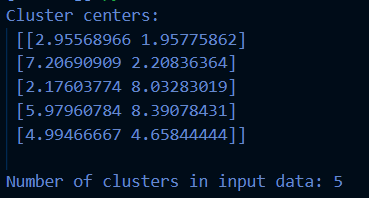
kmeans\_3 = KMeans(*n\_clusters*=3, *random\_state*=0)

labels\_3 = kmeans\_3.fit\_predict(X)

plot\_clusters(X, labels\_3, *centers*=kmeans\_3.cluster\_centers\_, *title*="KMeans: 3 кластери")

**Завдання 2.3. Оцінка кількості кластерів з використанням методу зсуву середнього:**

****

****

Лістинг:

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import MeanShift, estimate\_bandwidth

from itertools import cycle

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("Cluster centers:\n", cluster\_centers)

labels = meanshift\_model.labels\_

num\_clusters = len(np.unique(labels))

print("\nNumber of clusters in input data:", num\_clusters)

*def* plot\_meanshift\_clusters(*X*, *labels*, *cluster\_centers*, *title*="Кластери"):

    plt.figure()

    colors = cycle('bgrcmyk')

    markers = cycle('o\*xvs^')

    for cluster\_idx, color, marker in zip(range(num\_clusters), colors, markers):

        cluster\_points = *X*[*labels* == cluster\_idx]

        plt.scatter(cluster\_points[:, 0], cluster\_points[:, 1], *marker*=marker, *color*=color, *label*=*f*"Cluster {cluster\_idx + 1}")

        cluster\_center = *cluster\_centers*[cluster\_idx]

        plt.plot(cluster\_center[0], cluster\_center[1], *marker*='o', *markersize*=15,

*markerfacecolor*=color, *markeredgecolor*='black')

    plt.title(*title*)

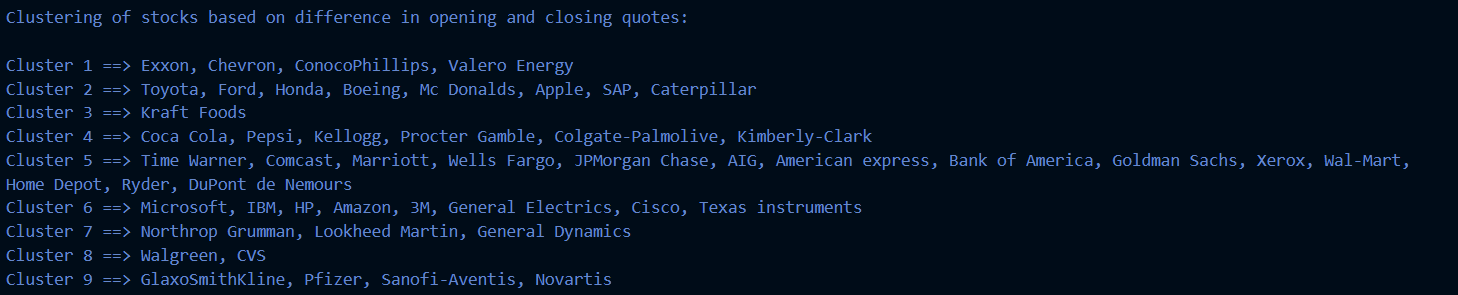
    plt.legend()

    plt.show()

plot\_meanshift\_clusters(X, labels, cluster\_centers)

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

Дані з файлу були перенесені всередину кода, також quotes\_yahoo та quotes\_historical\_yahoo\_ochl більше не існує в бібліотеці matplotlib.finance, тому було знайдено альтернативний спосіб вирішення завдання.



Лістинг:

import datetime

import numpy as np

import matplotlib.pyplot as plt

import yfinance as yf

from sklearn import covariance, cluster

company\_symbols\_map = {

    "TOT": "Total", "XOM": "Exxon", "CVX": "Chevron", "COP": "ConocoPhillips",

    "VLO": "Valero Energy", "MSFT": "Microsoft", "IBM": "IBM", "TWX": "Time Warner",

    "CMCSA": "Comcast", "CVC": "Cablevision", "YHOO": "Yahoo", "DELL": "Dell",

    "HPQ": "HP", "AMZN": "Amazon", "TM": "Toyota", "CAJ": "Canon",

    "MTU": "Mitsubishi", "SNE": "Sony", "F": "Ford", "HMC": "Honda",

    "NAV": "Navistar", "NOC": "Northrop Grumman", "BA": "Boeing",

    "KO": "Coca Cola", "MMM": "3M", "MCD": "Mc Donalds", "PEP": "Pepsi",

    "MDLZ": "Kraft Foods", "K": "Kellogg", "UN": "Unilever", "MAR": "Marriott",

    "PG": "Procter Gamble", "CL": "Colgate-Palmolive", "GE": "General Electrics",

    "WFC": "Wells Fargo", "JPM": "JPMorgan Chase", "AIG": "AIG",

    "AXP": "American express", "BAC": "Bank of America", "GS": "Goldman Sachs",

    "AAPL": "Apple", "SAP": "SAP", "CSCO": "Cisco", "TXN": "Texas instruments",

    "XRX": "Xerox", "LMT": "Lookheed Martin", "WMT": "Wal-Mart",

    "WBA": "Walgreen", "HD": "Home Depot", "GSK": "GlaxoSmithKline",

    "PFE": "Pfizer", "SNY": "Sanofi-Aventis", "NVS": "Novartis",

    "KMB": "Kimberly-Clark", "R": "Ryder", "GD": "General Dynamics",

    "RTN": "Raytheon", "CVS": "CVS", "CAT": "Caterpillar", "DD": "DuPont de Nemours"

}

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 = []

valid\_symbols = []

for symbol in symbols:

    try:

        stock\_data = yf.Ticker(symbol).history(*start*=start\_date, *end*=end\_date)

        if not stock\_data.empty:

            quotes.append(stock\_data)

            valid\_symbols.append(symbol)

        else:

            print(*f*"{symbol}: No data found")

    except Exception as e:

        print(*f*"{symbol}: Error fetching data - {e}")

if not quotes:

    print("No valid data retrieved.")

    exit()

min\_length = min(len(quote) for quote in quotes)

quotes = [quote.iloc[:min\_length] for quote in quotes]

opening\_quotes = np.array([quote['Open'].values for quote in quotes], *dtype*=np.float64)

closing\_quotes = np.array([quote['Close'].values for quote in quotes], *dtype*=np.float64)

quotes\_diff = closing\_quotes - opening\_quotes

X = quotes\_diff.copy().T

X /= X.std(*axis*=0)

edge\_model = covariance.GraphicalLassoCV()

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

    edge\_model.fit(X)

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

num\_labels = labels.max()

print('\nClustering of stocks based on difference in opening and closing quotes:\n')

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

    cluster\_symbols = np.array(valid\_symbols)[labels == i]

    cluster\_names = np.array(names)[np.isin(symbols, cluster\_symbols)]

    print(*f*"Cluster {i + 1} ==> {', '.join(cluster\_names)}")

Git: <https://github.com/PavlenkoOks/AI>