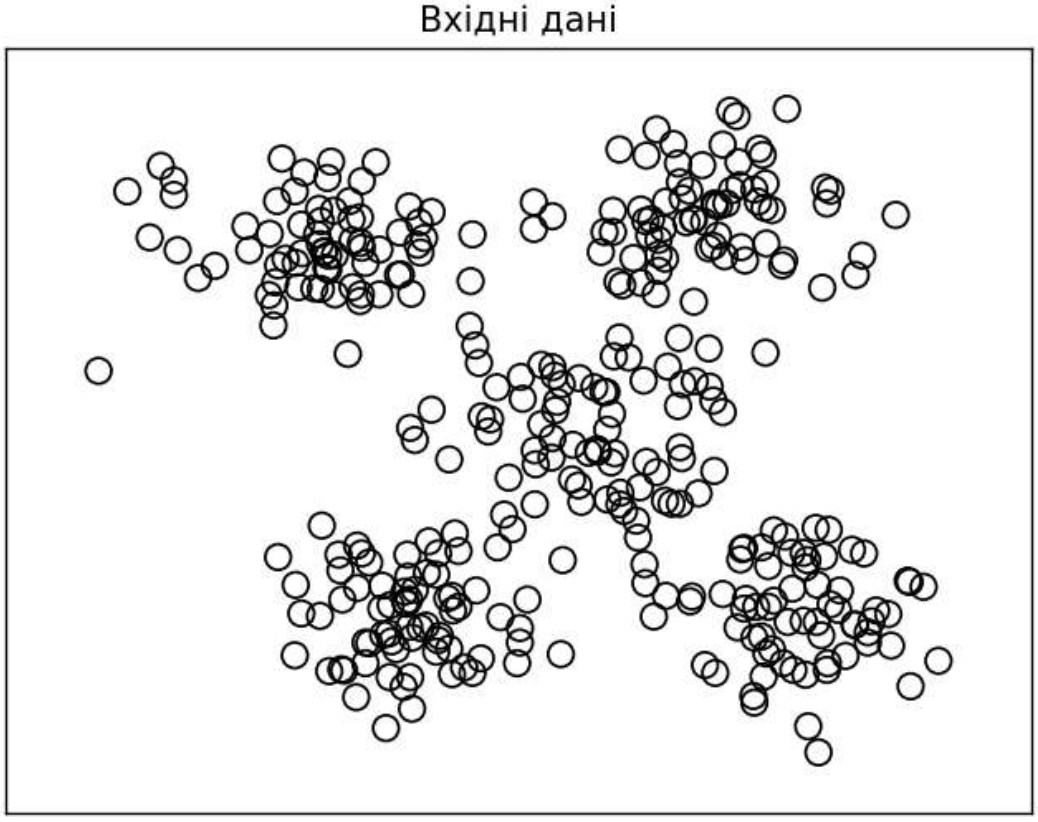
Лабораторна робота № 7

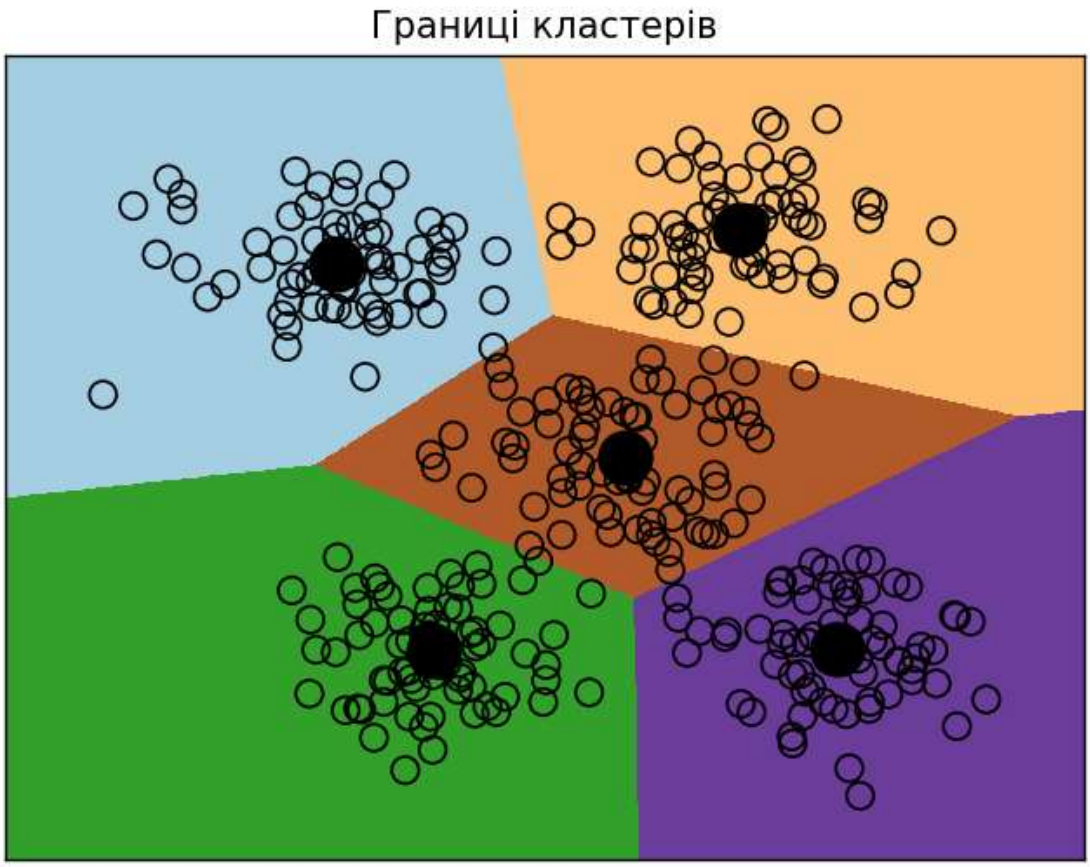
ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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:



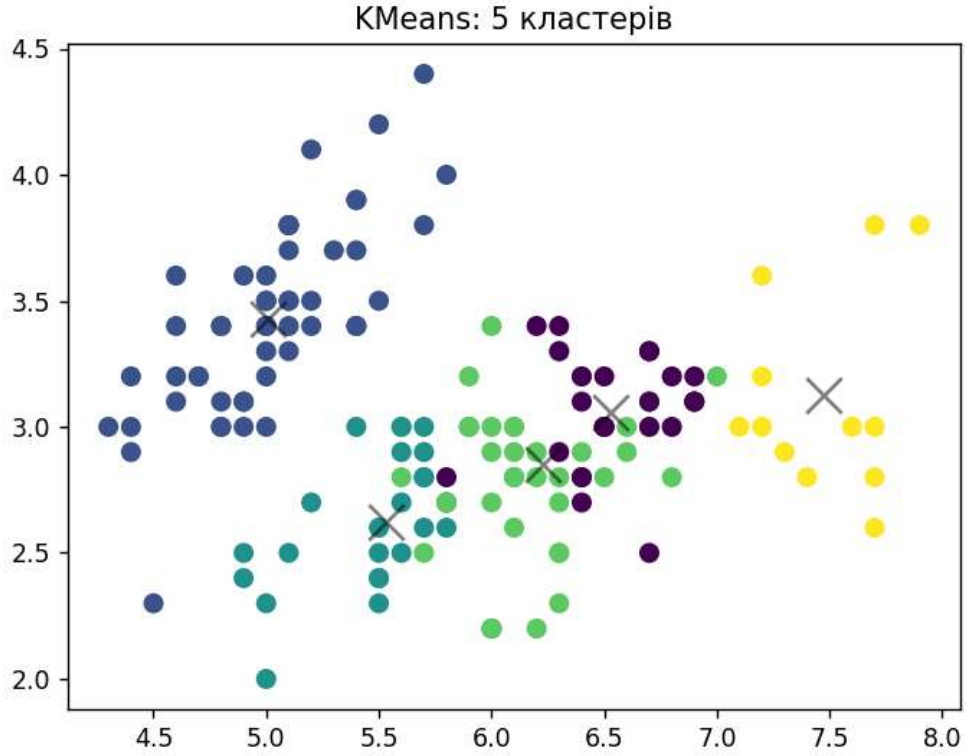


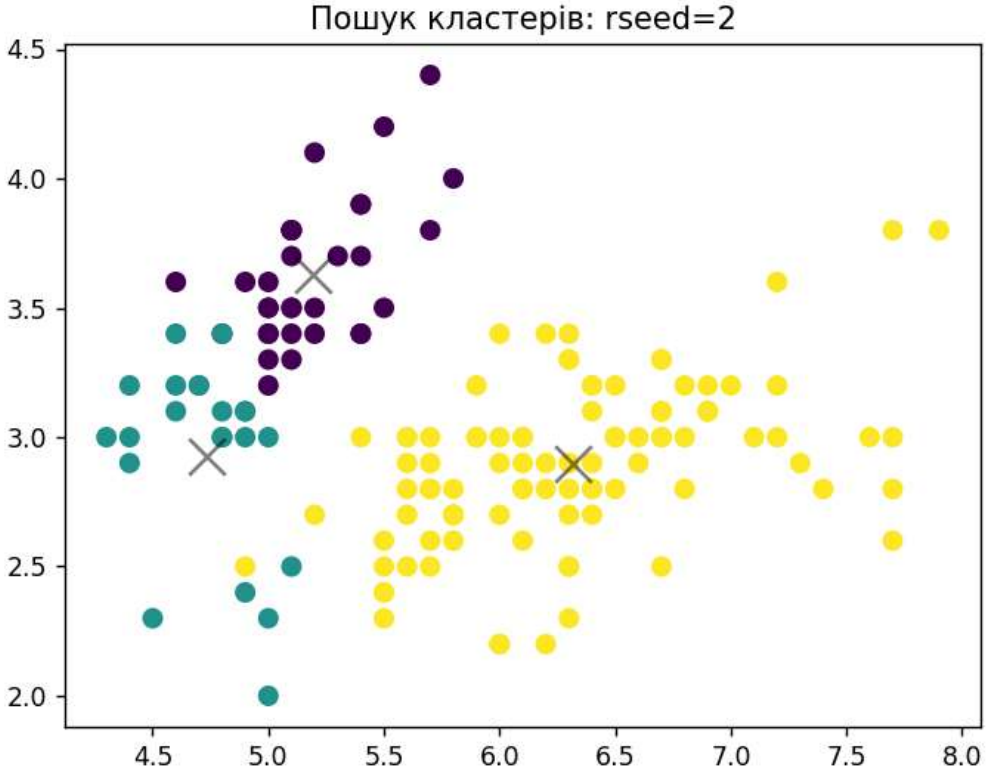
Лістинг:

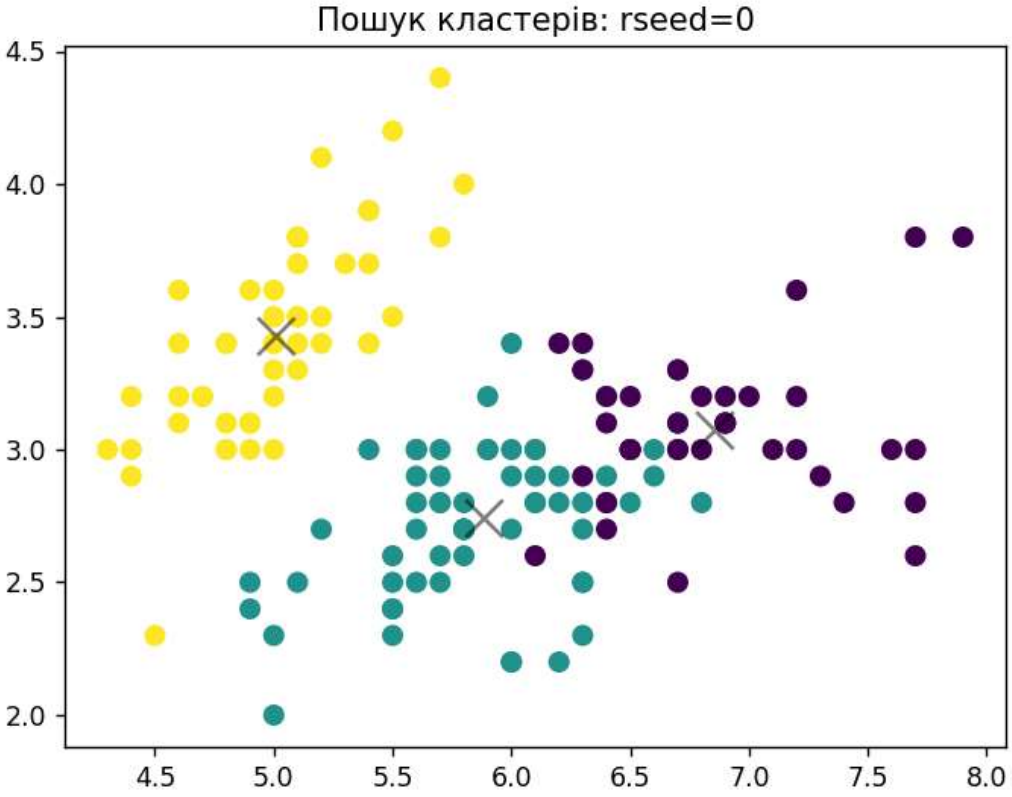
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

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)
```

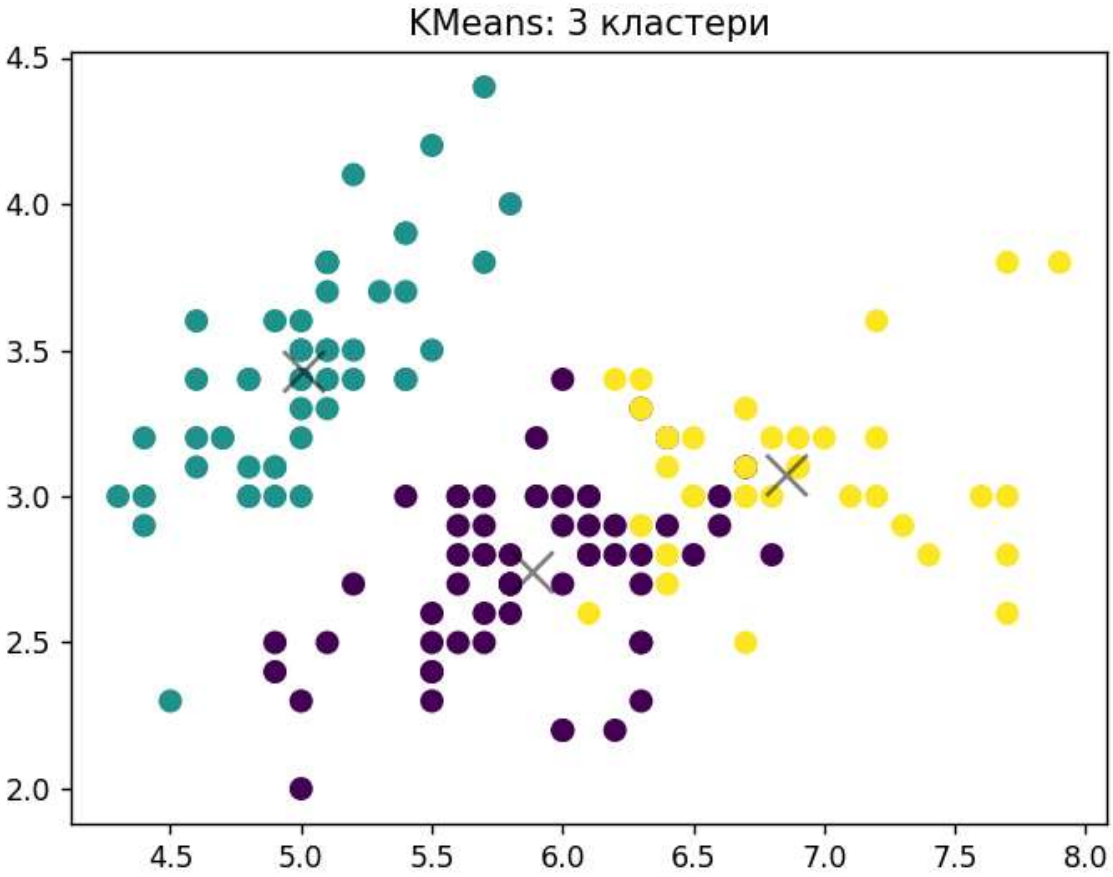
```
plt.xticks([])
plt.yticks([])
plt.show()

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:






Пошук кластерів: rseed=0
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

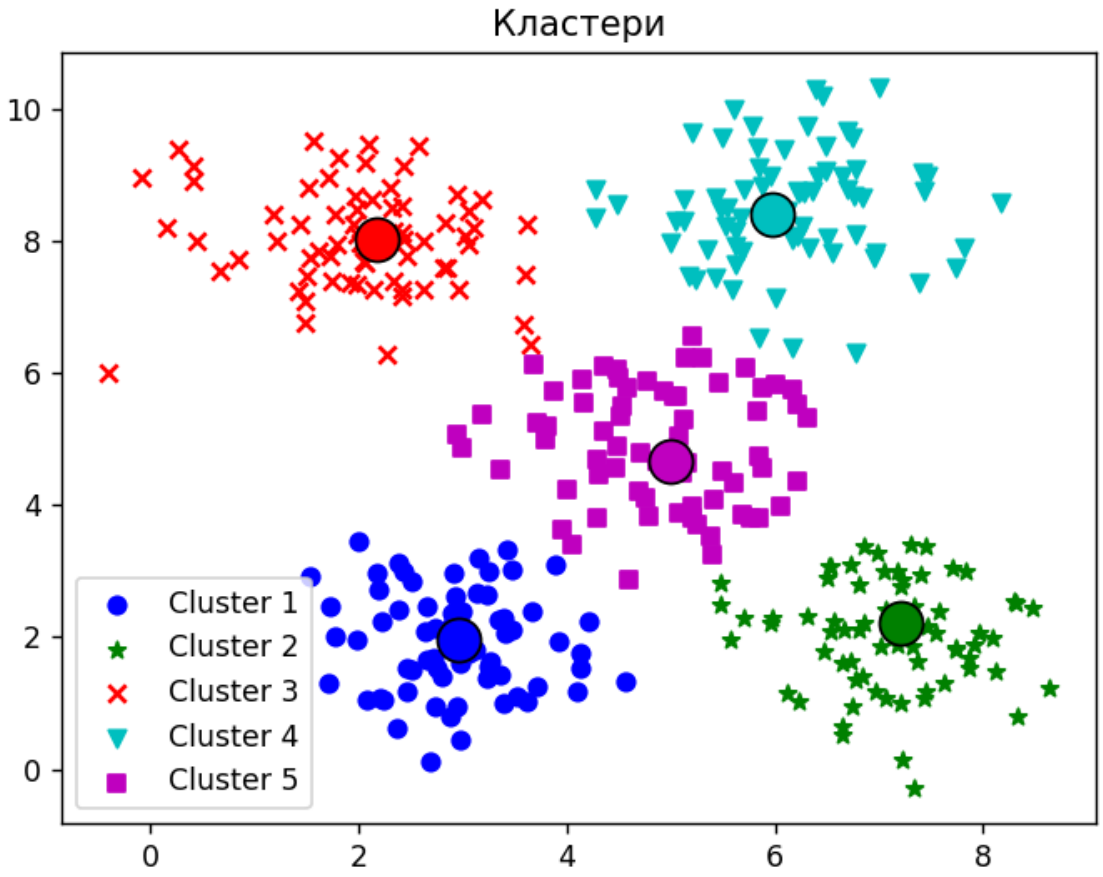

KMeans: 3 кластери
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

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:



```
Cluster centers:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Number of clusters in input data: 5
```

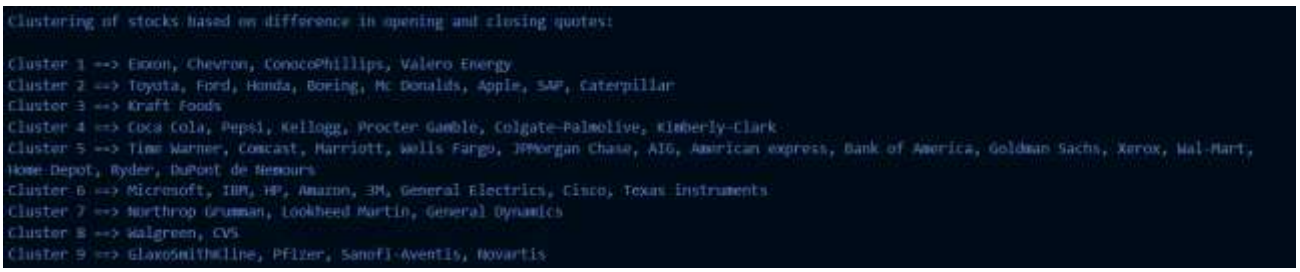
Лістинг:

```
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