Системи штучного інтелекту.
Лабораторна робота 6.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()
```

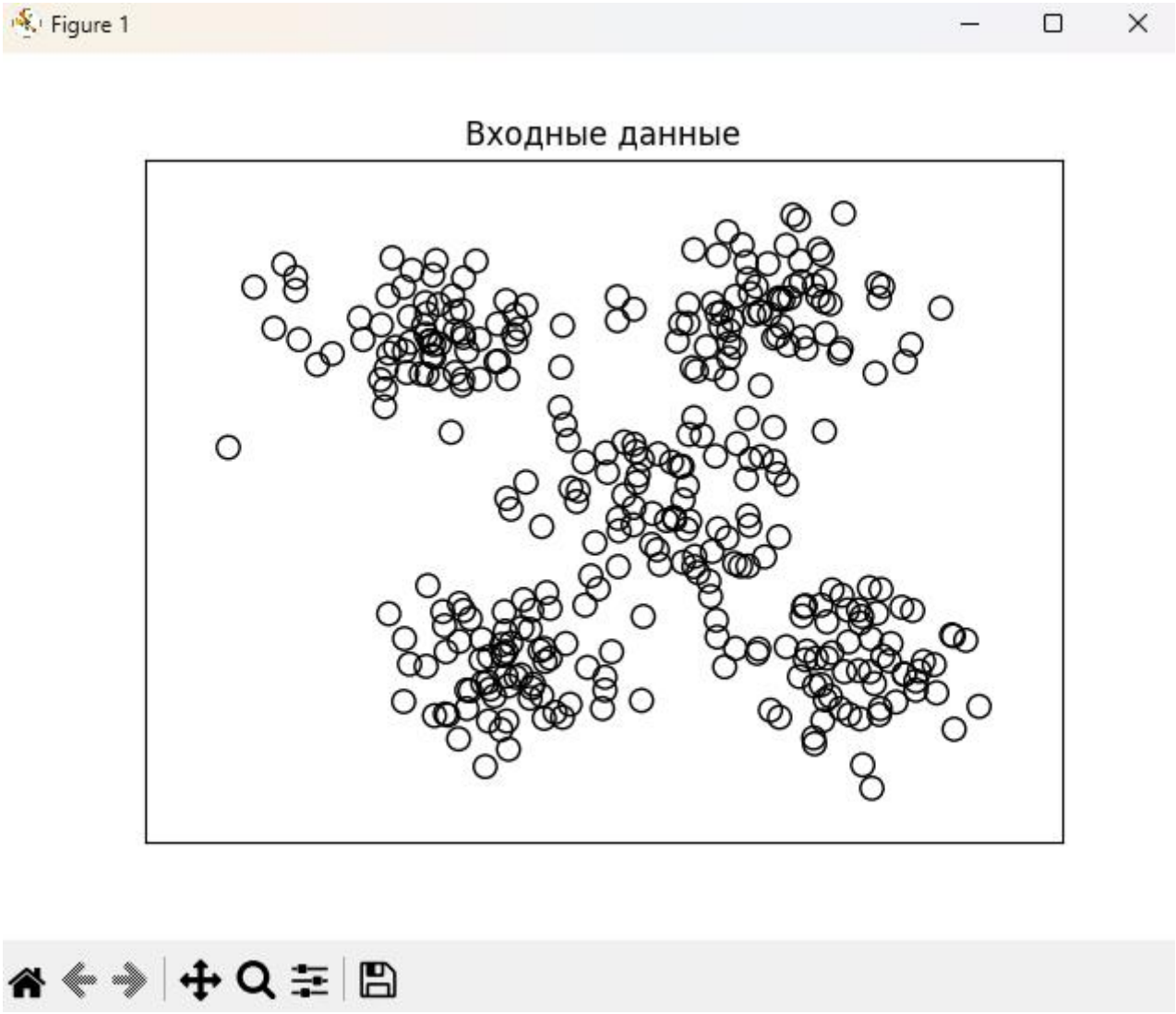
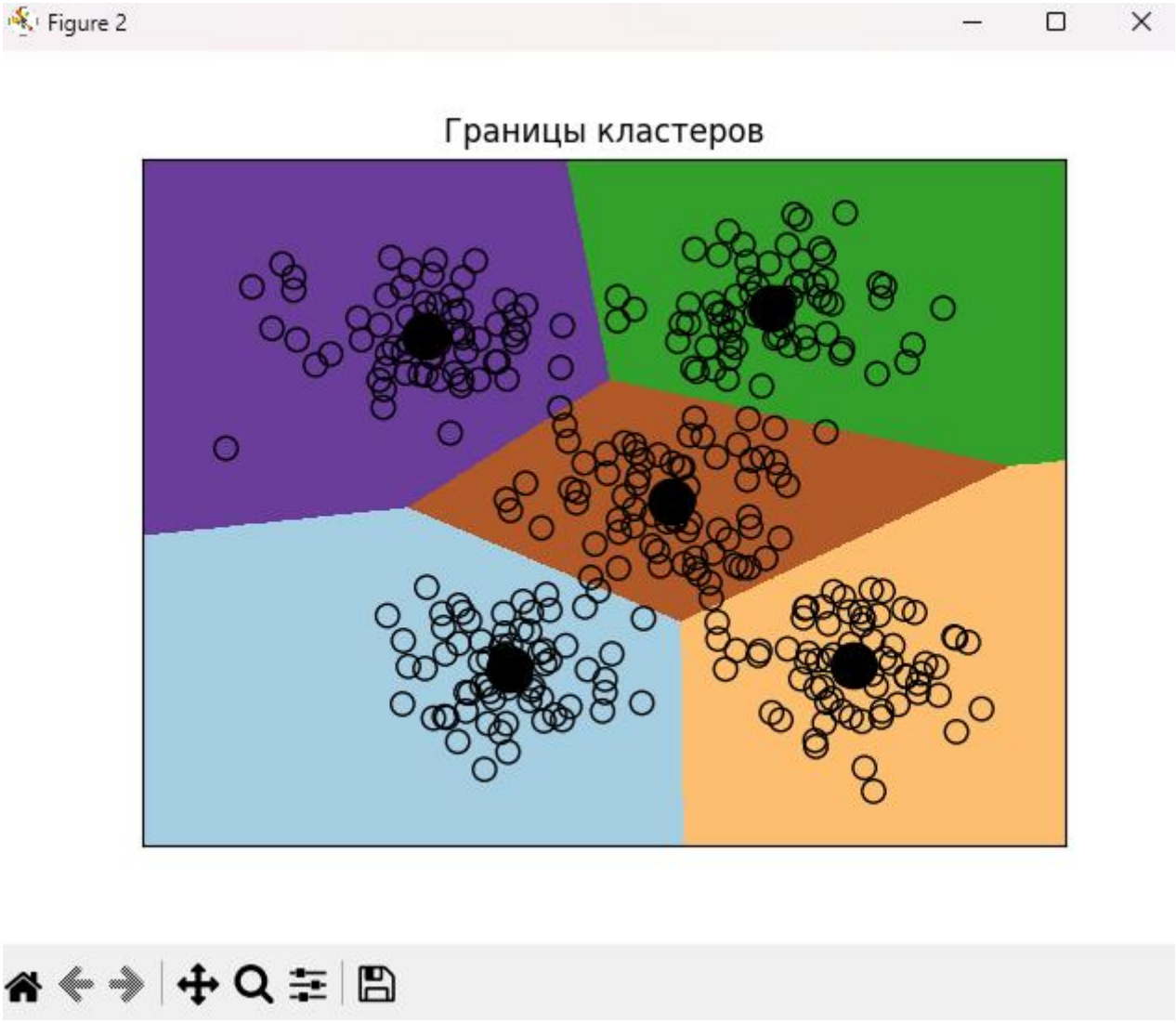


Figure 1
Входные данные



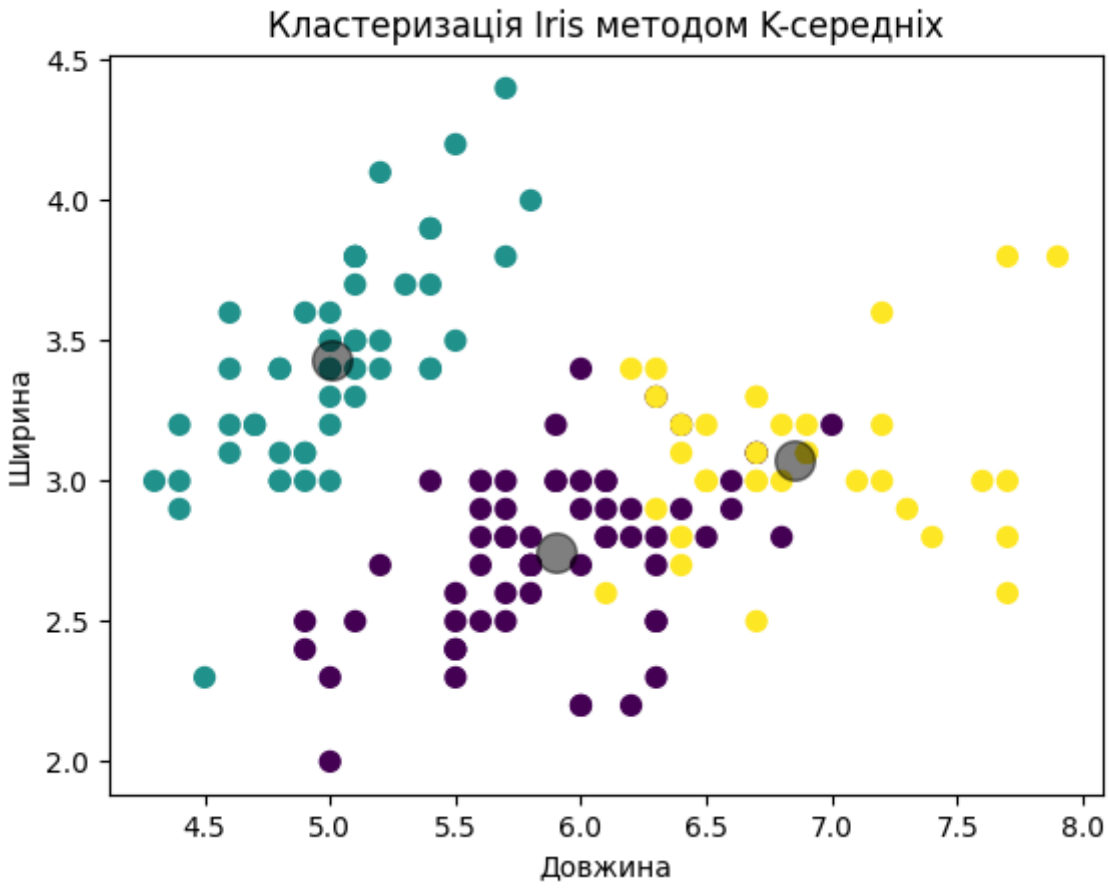
Аналіз коду підтверджує, що дані ефективно кластеризуються, і кожен кластер має чітко окреслені межі. Використання алгоритму кластеризації KMeans дозволяє виявити зони високої концентрації точок у кожному кластері. Візуалізація демонструє не лише чіткий розподіл точок між кластерами, але й правильне 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_)
```

```
Координати
[[5.9016129  2.7483871  4.39354839 1.43387097]
 [5.006      3.428      1.462      0.246     ]
 [6.85       3.07368421 5.74210526 2.07105263]]
```

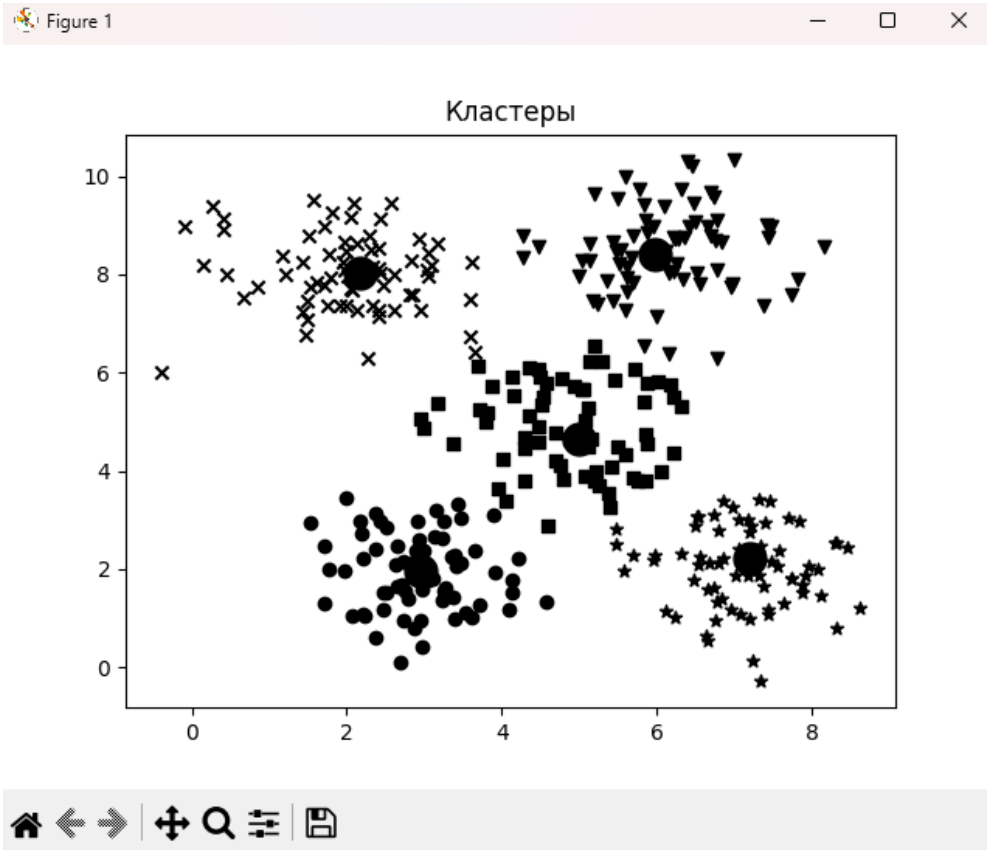


Завдання №3. Оцінка кількості кластерів з використанням методу зсуву середнього

```
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()
```

```
C:\Users\dimad\AppData\Local\Programs\Python\Python39\python.exe "F:/4 курс/СШI/lab7/LR_7_task_3.py"

Centers of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]
```



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]))
```

```
C:\Users\dimad\AppData\Local\Programs\Python\Python39\python.exe "F:/4 курс/СШІ/lab7/LR_7_task_4.py"
[**********100%**********]  1 of 1 completed

1 Failed download:
['TOT']: YFTzMissingError('$%ticker%: possibly delisted; no timezone found')
Skipping symbol TOT due to missing data.
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
[**********100%**********]  1 of 1 completed
```