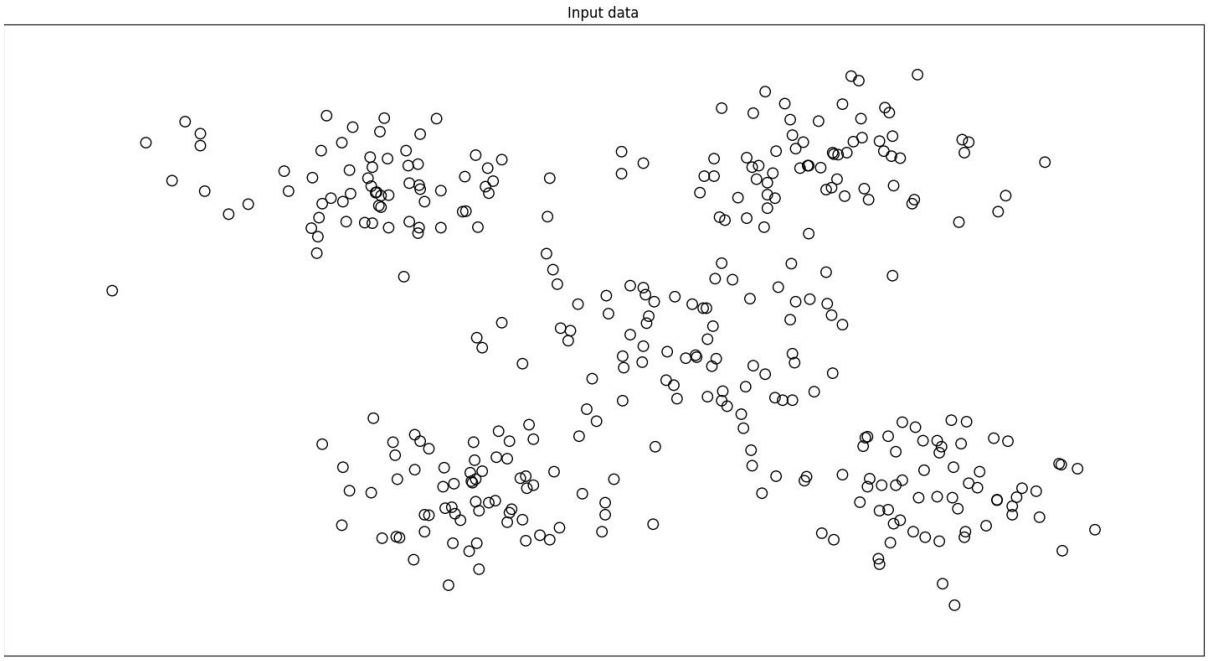
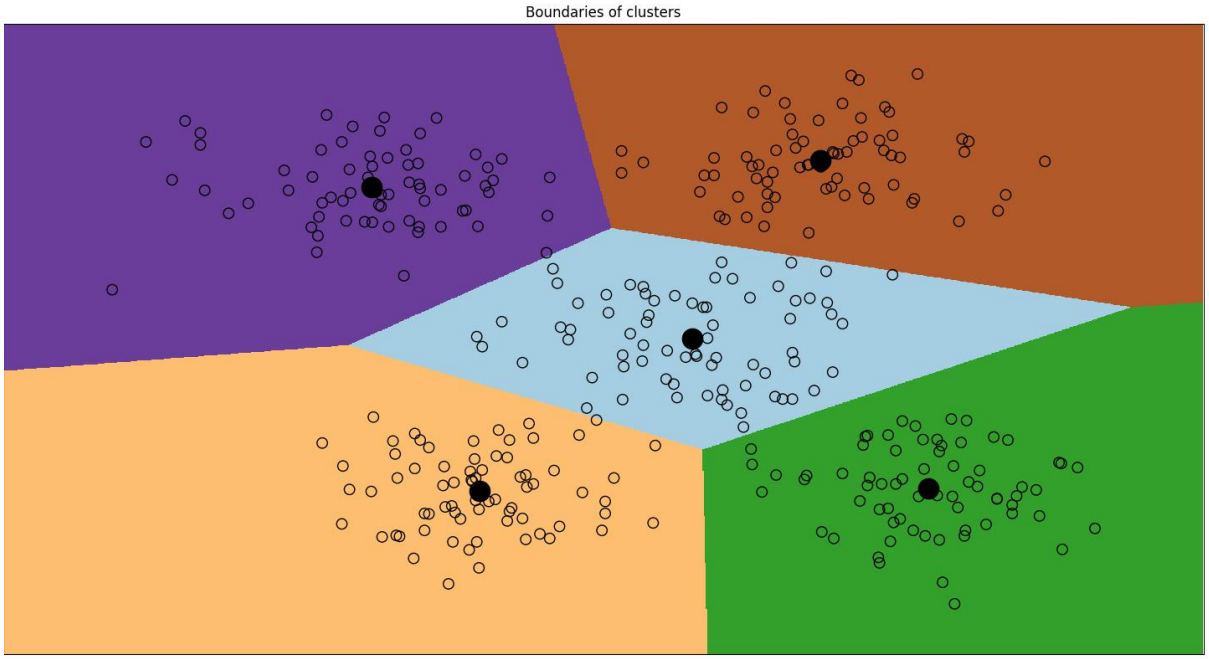
# ЛАБОРАТОРНА РОБОТА № 4
# ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

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






LR_7_task_1.py

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

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('Input data')
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)
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()))
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', zor
plt.title('Boundaries of clusters')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()

```

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

LR_7_task_2.py

```python
from sklearn.svm import SVC
from sklearn.metrics import pairwise_distances_argmin
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
import numpy as np
import matplotlib.pyplot as plt

# Load the iris dataset
iris = load_iris()
X = iris['data']
y = iris['target']

# Initialize KMeans with corrected parameters
kmeans = KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_stat

# Fit the model
kmeans.fit(X)

# Predict the clusters
y_kmeans = kmeans.predict(X)

# Plot the clusters
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.show()

# Define a function to find clusters
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

# Find clusters using the custom function
centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()

# Use KMeans to predict clusters
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')
plt.show()
```

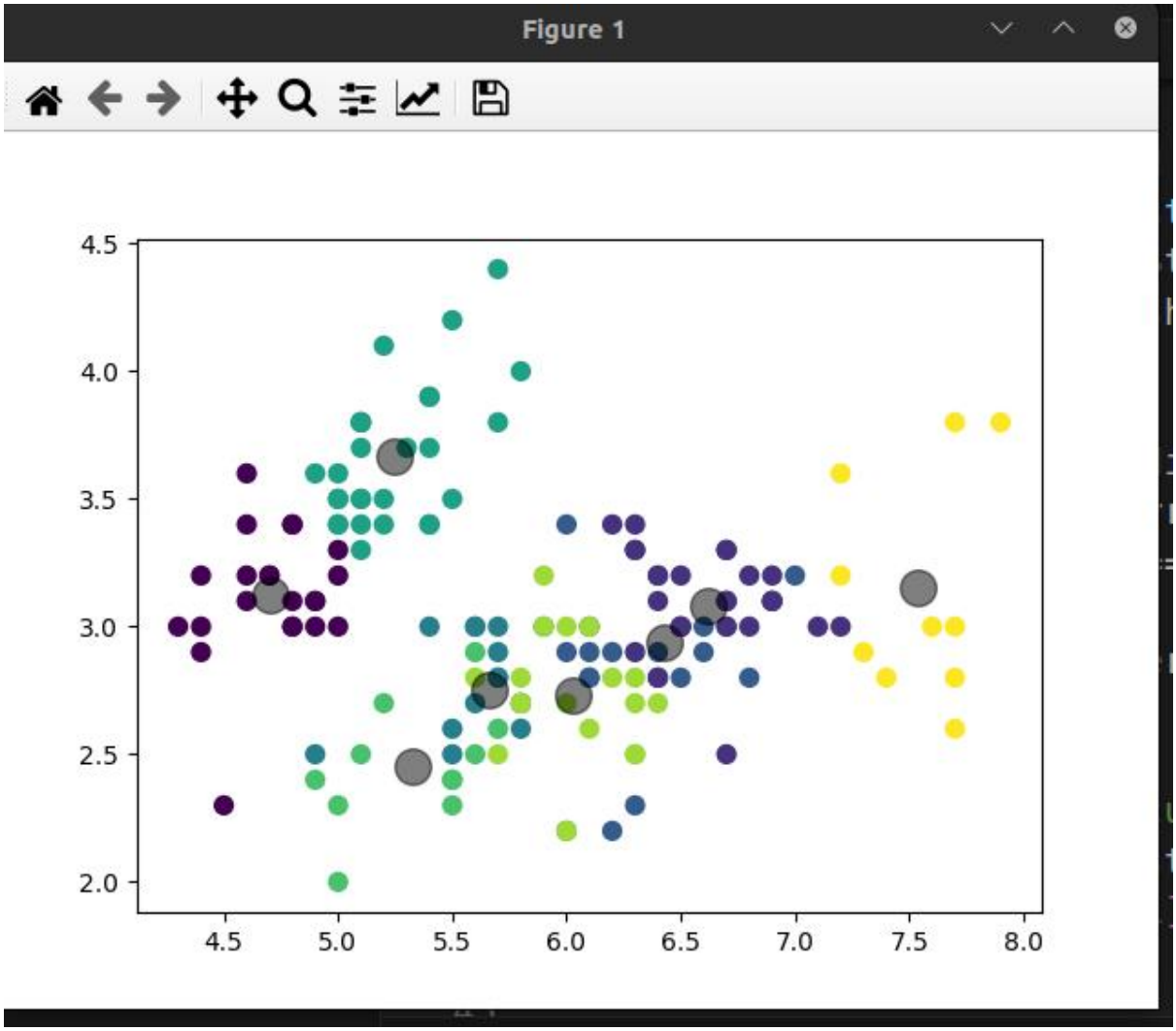


Figure 1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

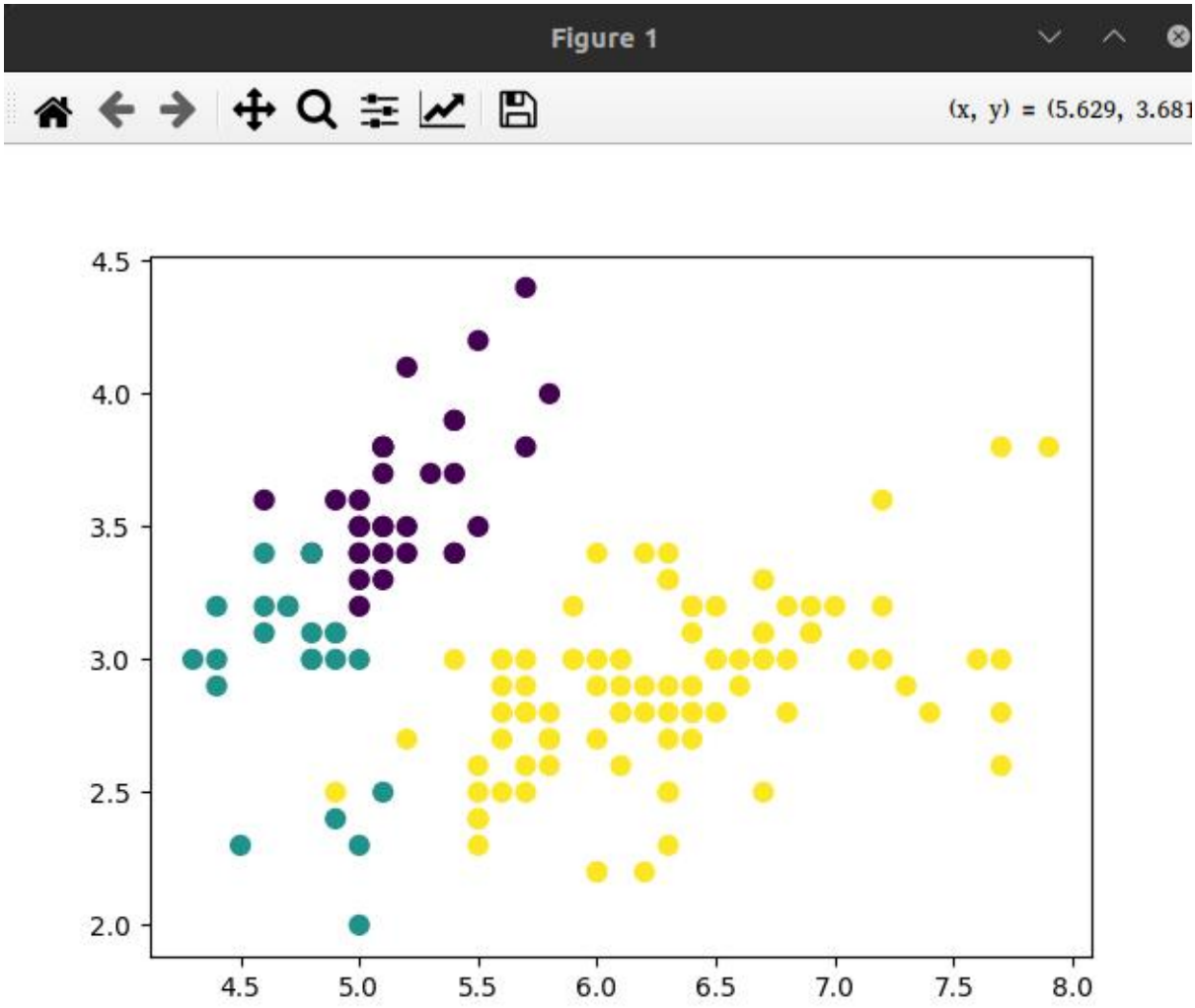


Figure 1
(x, y) = (5.629, 3.681
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

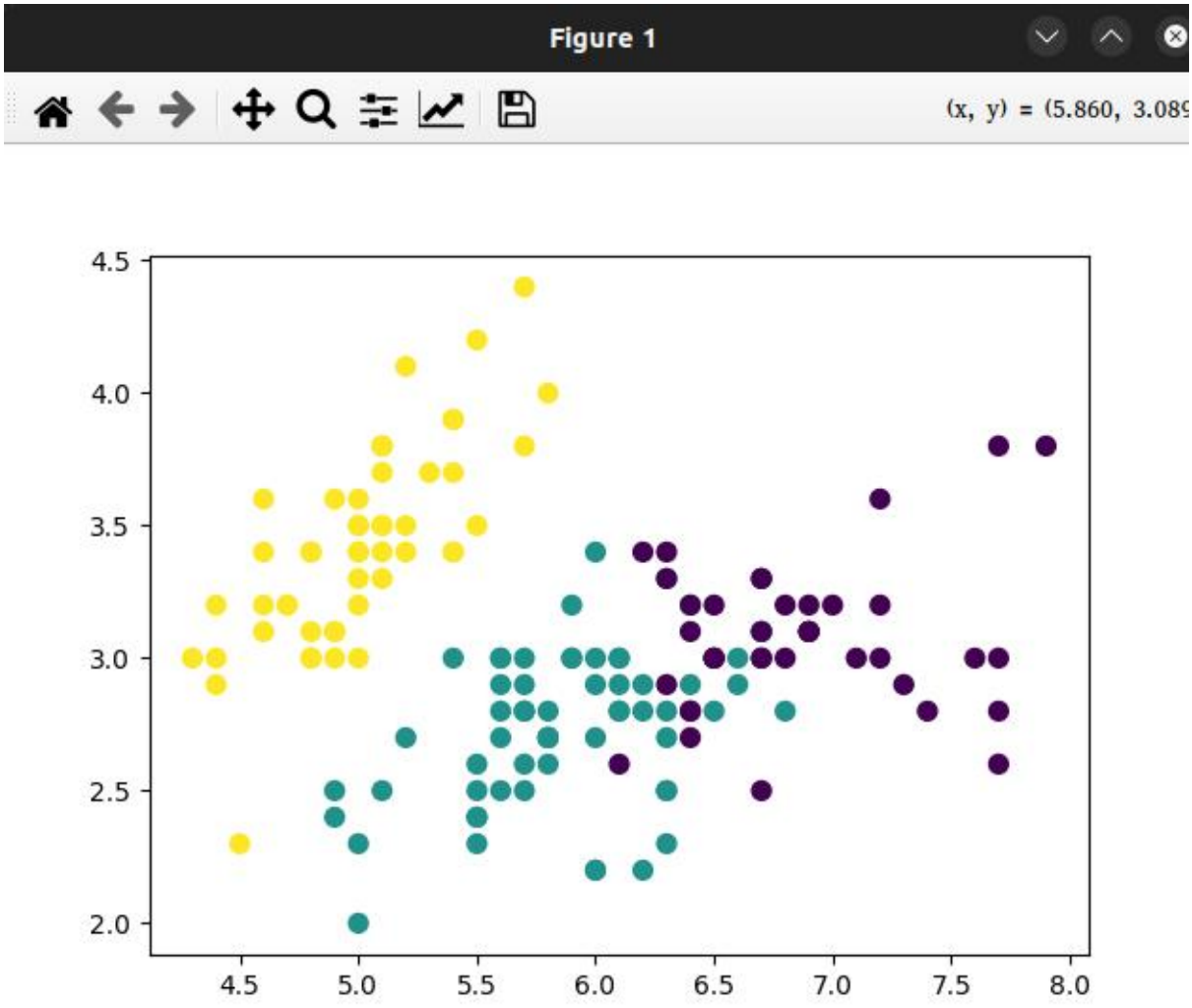
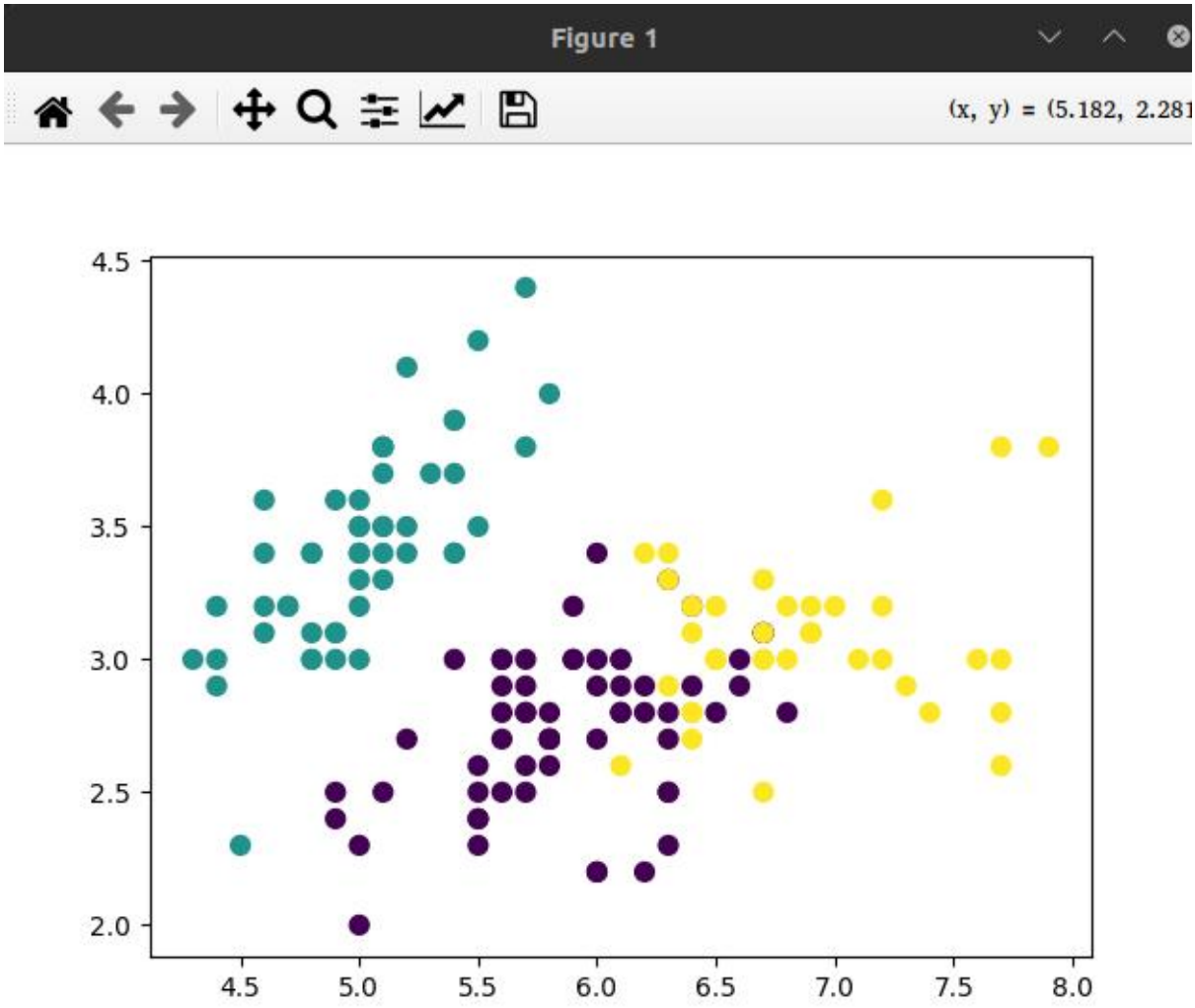


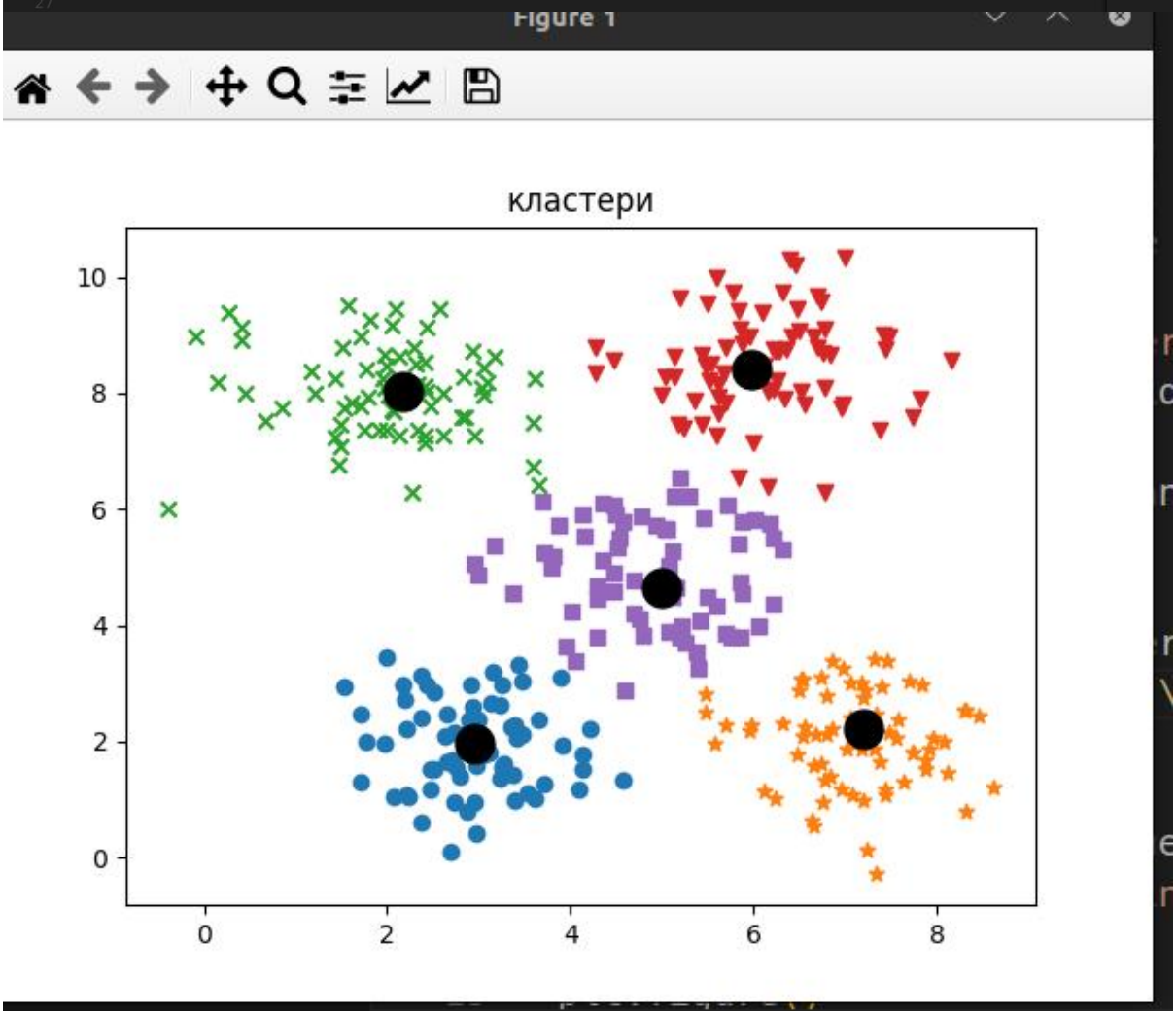
Figure 1
(x, y) = (5.860, 3.089
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

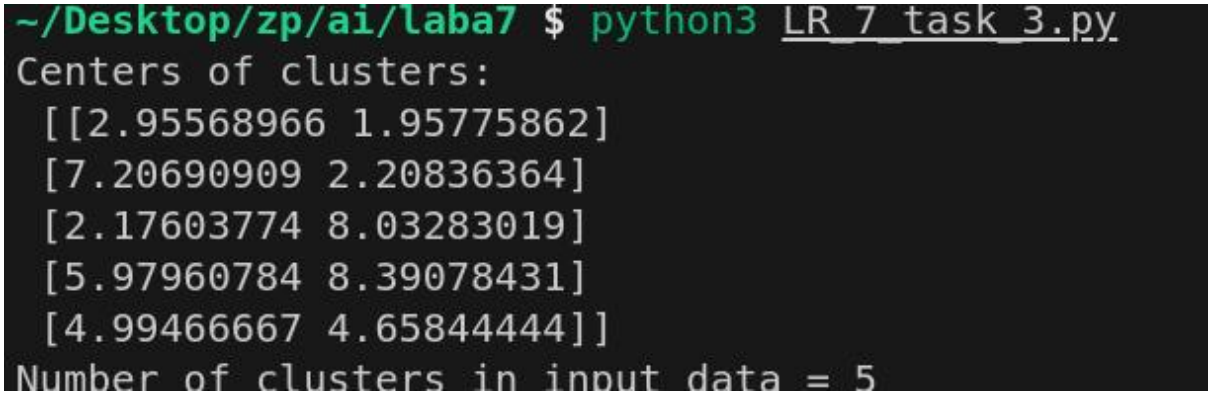



**Завдання 3.** Оцінка кількості кластерів з використанням методу зсуву середнього

LR_7_task_3.py

```python
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 = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))

ms = MeanShift(bandwidth=bandwidth, bin_seeding=True)
ms.fit(X)

cluster_centers = ms.cluster_centers_
print('Centers of clusters:\n', cluster_centers)

labels = ms.labels_
num_clusters = len(np.unique(labels))
print("Number 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)
    cluster_centers = ms.cluster_centers_[i]
    plt.plot(cluster_centers[0], cluster_centers[1], marker='o', markerfacecolor='black', markeredgecolor='
    plt.title('кластери')
plt.show()
```






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

LR_7_task_4.py

```python
import datetime
import json
import numpy as np
from sklearn import covariance, cluster
import pandas as pd
import yfinance as yf
from sklearn.impute import SimpleImputer

# Load company symbols
input_file = 'company_symbol_mapping.json'
with open(input_file, 'r') as f:
    company_symbols_map = json.loads(f.read())
symbols = list(company_symbols_map.keys())

# Set date range
start_date = "2003-07-03"
end_date = "2007-05-04"

# Load stock data
quotes = {}
for symbol in symbols:
    try:
        print(f"Loading {symbol} ({company_symbols_map[symbol]})...", end='')
        q = yf.download(symbol, start=start_date, end=end_date)
        if not q.empty:
            quotes[symbol] = q
            print("done.")
        else:
            print("no data.")
    except Exception as e:
        print(f"error: {e}")
```

LR_7_task_4.py

```python
# Align all data to the same index (date range)
aligned_quotes = pd.concat(quotes.values(), keys=quotes.keys(), names=["Symbol", "Date"])
aligned_quotes = aligned_quotes.unstack(level=0)  # Organize by symbol

# Extract opening and closing quotes
opening_quotes = aligned_quotes['Open'].values  # Shape: (time, stocks)
closing_quotes = aligned_quotes['Close'].values  # Shape: (time, stocks)

# Compute differences
quotes_diff = closing_quotes - opening_quotes

# Handle missing values
imputer = SimpleImputer(strategy='mean')  # Replace NaN with column mean
quotes_diff_imputed = imputer.fit_transform(quotes_diff)

# Normalize data
X = quotes_diff_imputed.copy()
X /= X.std(axis=0)  # Normalize each stock's data

# Graphical Lasso for covariance estimation
edge_model = covariance.GraphicalLassoCV(cv=3)

# Train model
with np.errstate(invalid='ignore'):
    edge_model.fit(X)

# Perform clustering
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()

# Output clustering results
```

```
58
59  # Perform clustering
60  _, labels = cluster.affinity_propagation(edge_model.covariance_)
61  num_labels = labels.max()
62
63  # Output clustering results
64  names = np.array([company_symbols_map[symbol] for symbol in quotes.keys()])
65  for i in range(num_labels + 1):
66      print("Cluster", i+1, "==>", ', '.join(names[labels == i]))
67
```

```
Cluster 1 ==> Exxon, Chevron, ConocoPhillips, Valero Energy
Cluster 2 ==> Toyota, Ford, Honda, Boeing, Mc Donalds, Apple, SAP, Caterpillar
Cluster 3 ==> Kraft Foods
Cluster 4 ==> Coca Cola, Pepsi, Kellogg, Procter Gamble, Colgate-Palmolive, Kimberly-Clark
Cluster 5 ==> Time Warner, Comcast, Marriott, Wells Fargo, JPMorgan Chase, AIG, American express, Bank of America, Goldman
 Sachs, Xerox, Wal-Mart, Home Depot, Ryder, DuPont de Nemours
Cluster 6 ==> Microsoft, IBM, HP, Amazon, 3M, General Electrics, Cisco, Texas instruments
Cluster 7 ==> Northrop Grumman, Lookheed Martin, General Dynamics
Cluster 8 ==> Walgreen, CVS
Cluster 9 ==> GlaxoSmithKline, Pfizer, Sanofi-Aventis, Novartis
```

Посилання на GitHub: [https://github.com/missShevel/SHI_Shevel_Olha_IPZ-21-1/tree/master/Lab7](https://github.com/missShevel/SHI_Shevel_Olha_IPZ-21-1/tree/master/Lab7)