**ЛАБОРАТОРНА РОБОТА № 3**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА**

**НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ**

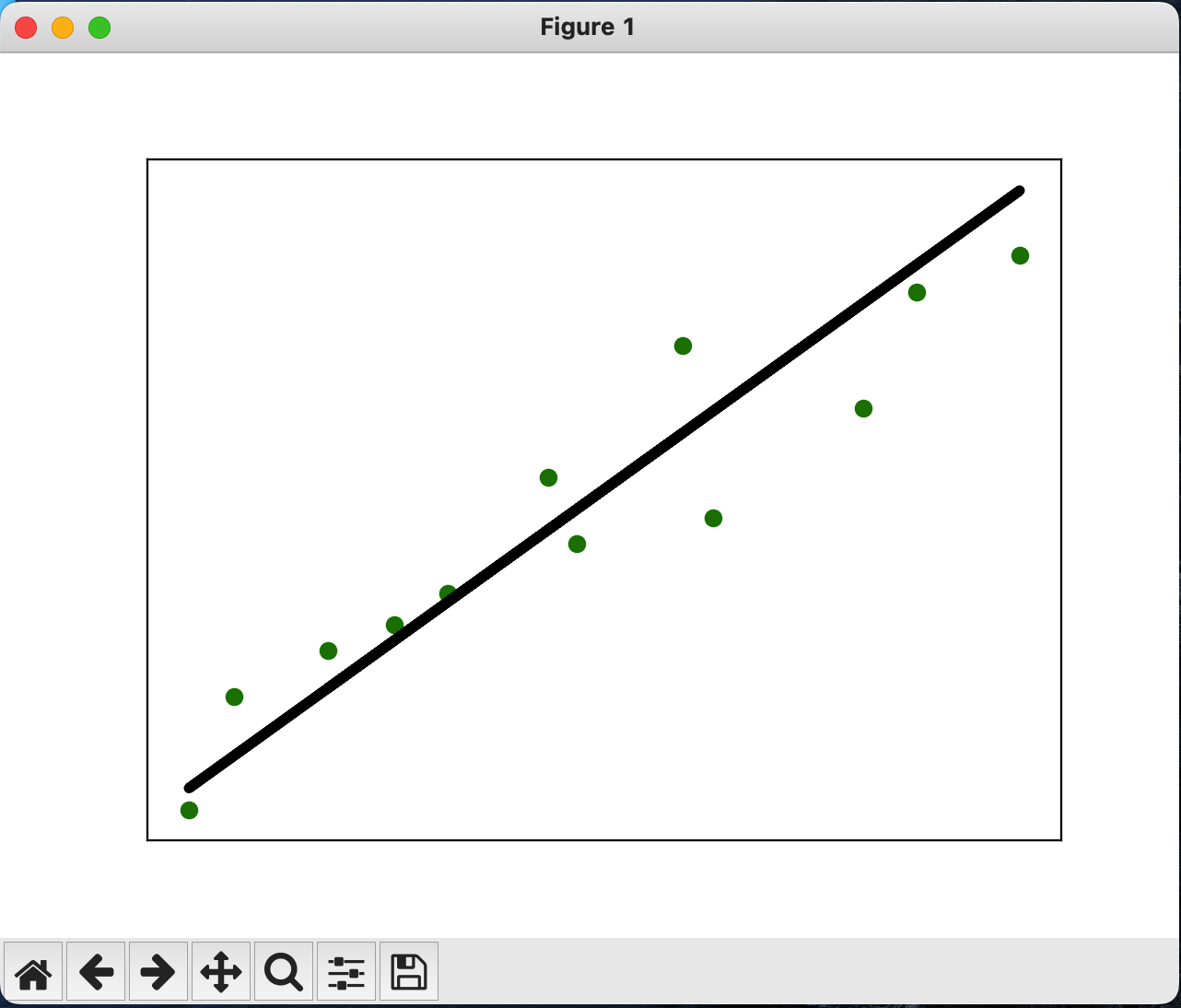
***Мета******:*** використовуючи спеціалізовані бібліотеки і мову програмування Python дослідити методи регресії та неконтрольованої класифікації даних у машинному навчанні.

**Хід роботи:**

Завдання №1: Створення регресора однієї змінної.

Лістинг програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
input\_file = 'data\_singlevar\_regr.txt'  
  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = regressor.predict(X\_test)  
  
plt.scatter(X\_test, y\_test, color='green')  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
  
print("Linear regressor performance:")  
print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  
  
output\_model\_file = 'model.pkl'  
  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))



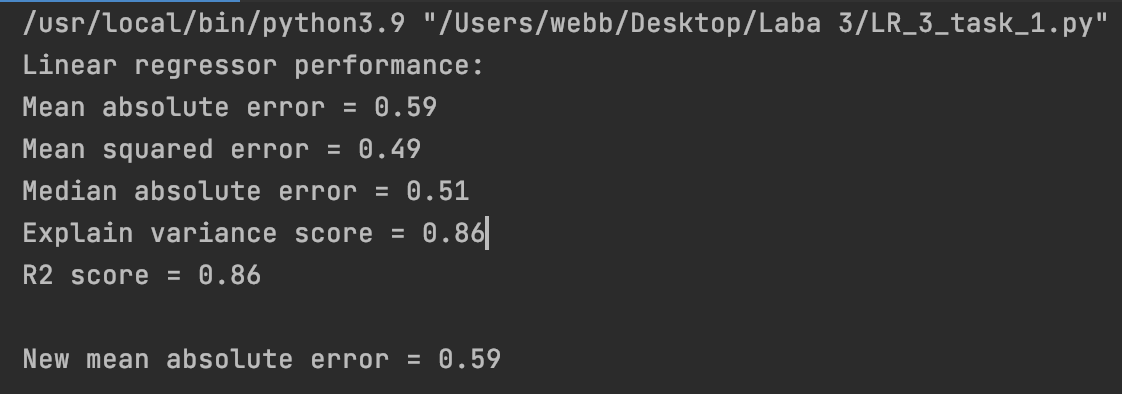


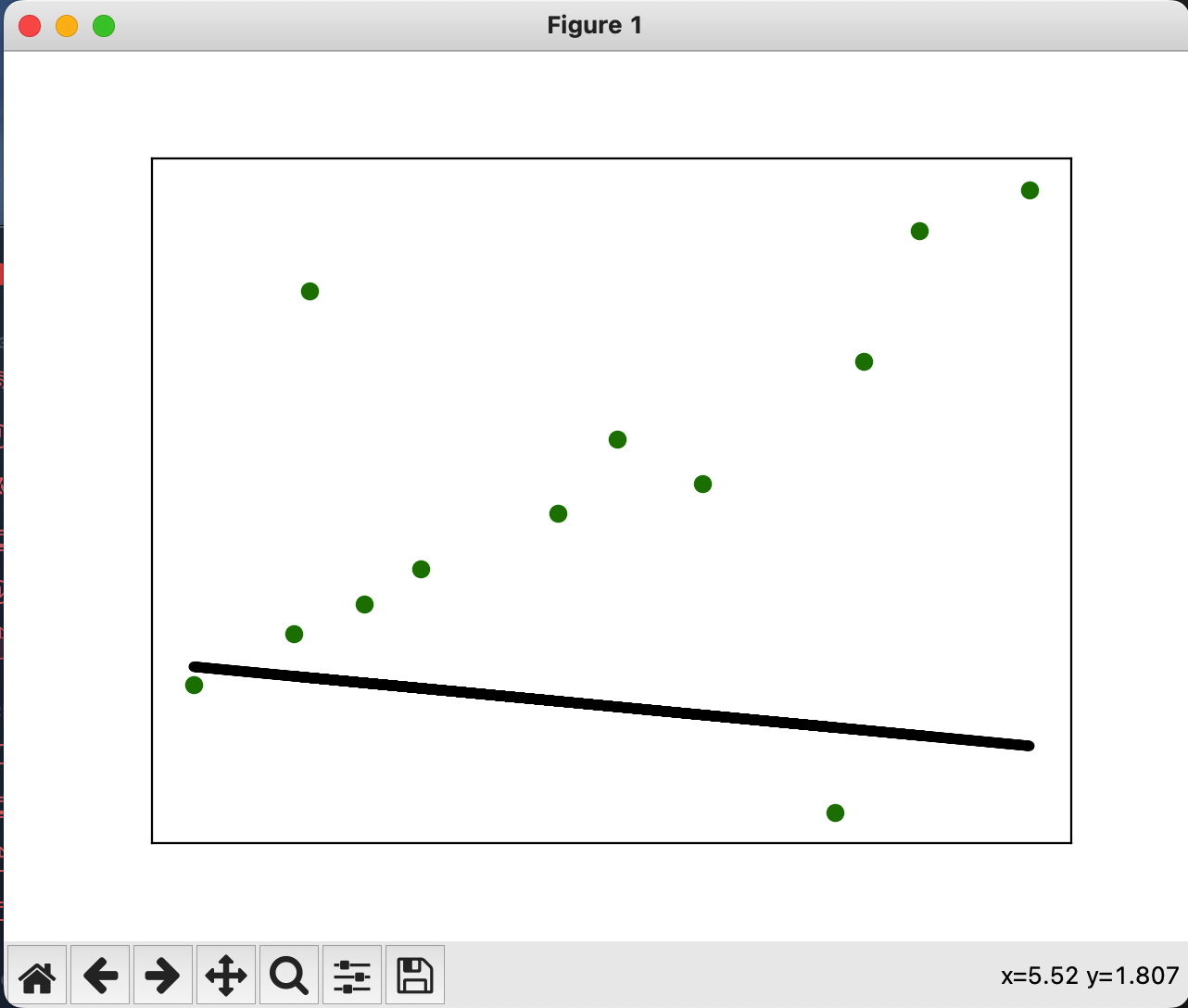
Рис. 1. Результат виконання програми

Завдання №2: Передбачення за допомогою регресії однієї змінної.

|  |  |
| --- | --- |
| № за списком | 2 |
| № варіанту | 2 |

Лістинг програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
input\_file = 'data\_regr\_2.txt'  
  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = regressor.predict(X\_test)  
  
plt.scatter(X\_test, y\_test, color='green')  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
  
print("Linear regressor performance:")  
print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  
  
output\_model\_file = 'model.pkl'  
  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))



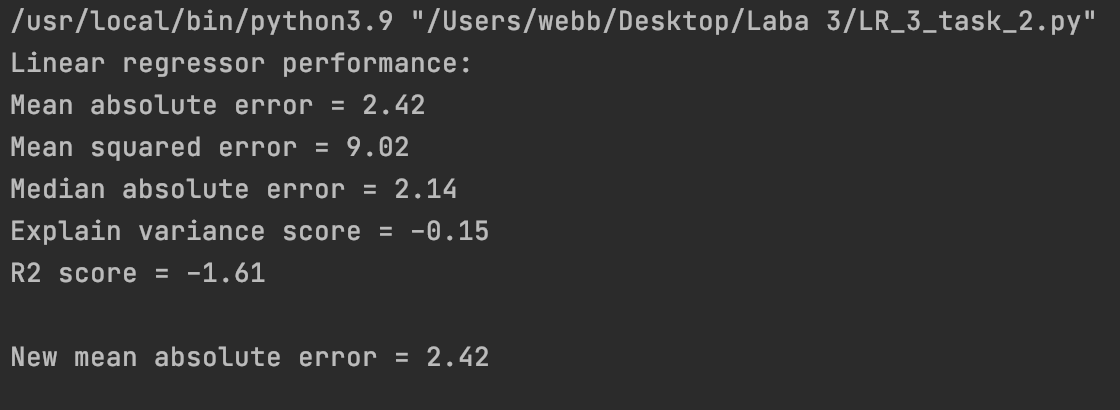


Рис. 2. Результат виконання програми

Завдання №3: Створення багатовимірного регресора.

Лістинг програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
  
input\_file = 'data\_multivar\_regr.txt'  
  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = regressor.predict(X\_test)  
  
print("Linear regressor performance:")  
print("Mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Mean squared error =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Median absolute error =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 score =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  
  
output\_model\_file = 'model.pkl'  
  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
  
with open(output\_model\_file, 'rb') as f:  
 regressor\_model = pickle.load(f)  
  
y\_test\_pred\_new = regressor\_model.predict(X\_test)  
print("\nNew mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))  
  
polynomial = PolynomialFeatures(degree=10)  
X\_train\_transformed = polynomial.fit\_transform(X\_train)  
datapoint = [[7.75, 6.35, 5.56]]  
poly\_datapoint = polynomial.fit\_transform(datapoint)  
  
poly\_linear\_model = linear\_model.LinearRegression()  
poly\_linear\_model.fit(X\_train\_transformed, y\_train)  
print("\nLinear regression:\n", regressor.predict(datapoint))  
print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint))

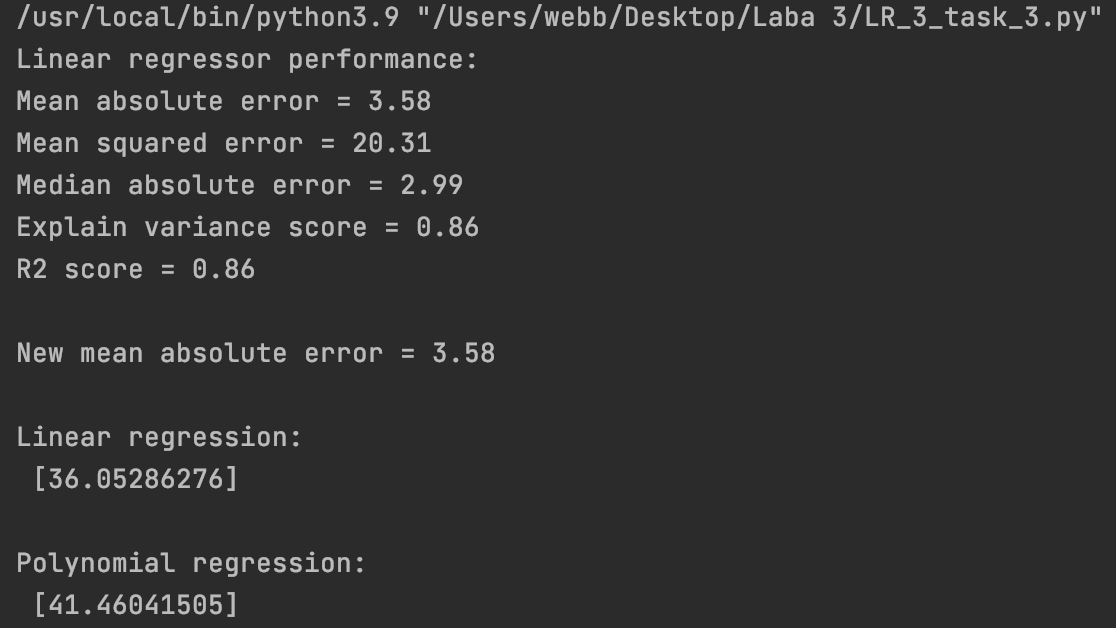
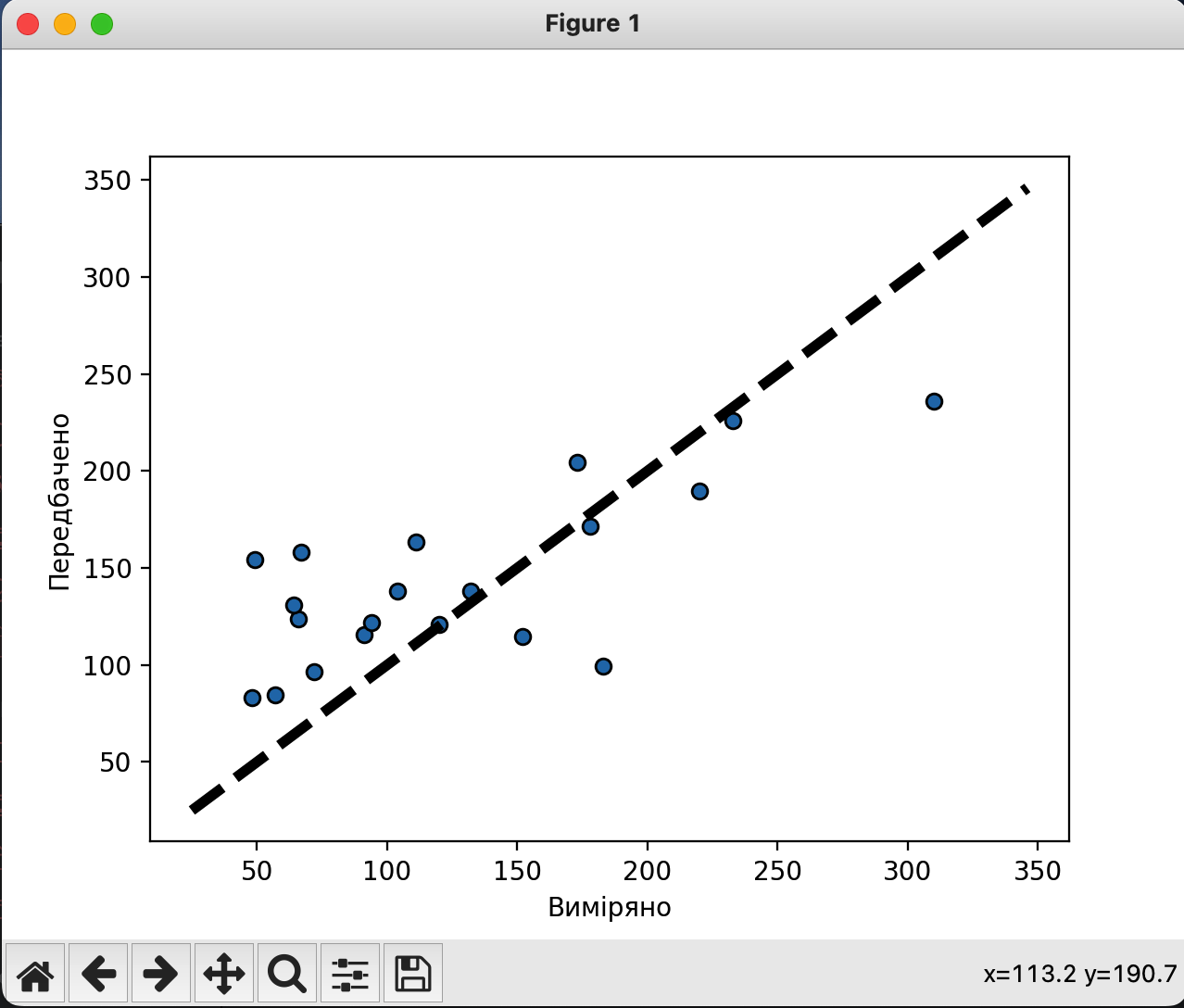


Рис. 3. Результат виконання програми

Завдання №4: Регресія багатьох змінних.

Лістинг програми:

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model  
from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error  
  
diabetes\_X, diabetes\_y = datasets.load\_diabetes(return\_X\_y=True)  
  
diabetes\_X = diabetes\_X[:, np.newaxis, 2]  
  
diabetes\_X\_train = diabetes\_X[:-20]  
diabetes\_X\_test = diabetes\_X[-20:]  
  
diabetes\_y\_train = diabetes\_y[:-20]  
diabetes\_y\_test = diabetes\_y[-20:]  
  
regr = linear\_model.LinearRegression()  
  
regr.fit(diabetes\_X\_train, diabetes\_y\_train)  
  
diabetes\_y\_pred = regr.predict(diabetes\_X\_test)  
  
# The coefficients  
print("Regression coef: \n", regr.coef\_)  
print("Regression intercept: \n", regr.intercept\_)  
# Середня абсолютна похибка  
print("Mean absolute error :",  
 round(mean\_absolute\_error(diabetes\_y\_test, diabetes\_y\_pred), 2))  
# The mean squared error  
print("Mean squared error: %.2f" % mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))  
# The coefficient of determination: 1 is perfect prediction  
print("R2 score: %.2f" % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))  
  
fig, ax = plt.subplots()  
ax.scatter(diabetes\_y\_test, diabetes\_y\_pred, edgecolors=(0, 0, 0))  
ax.plot([diabetes\_y.min(), diabetes\_y.max()], [diabetes\_y.min(), diabetes\_y.max()], 'k--', lw=4)  
ax.set\_xlabel('Виміряно')  
ax.set\_ylabel('Передбачено')  
plt.show()



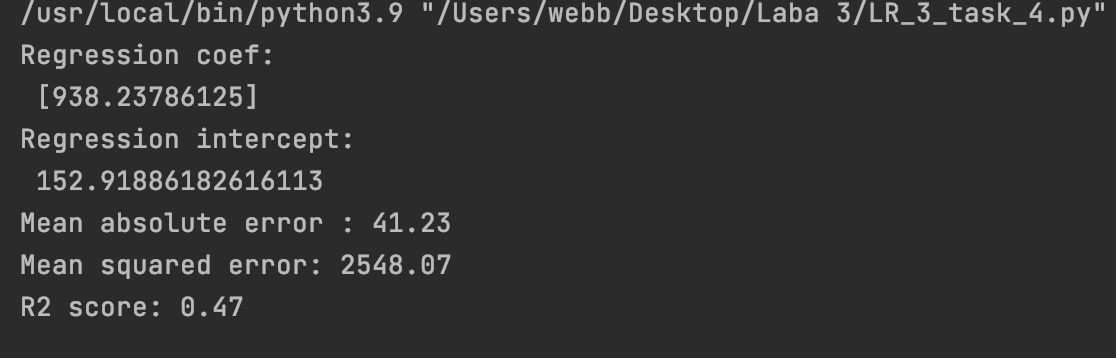
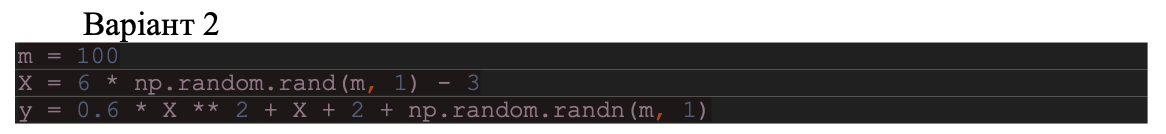


Рис. 4. Результат виконання програми

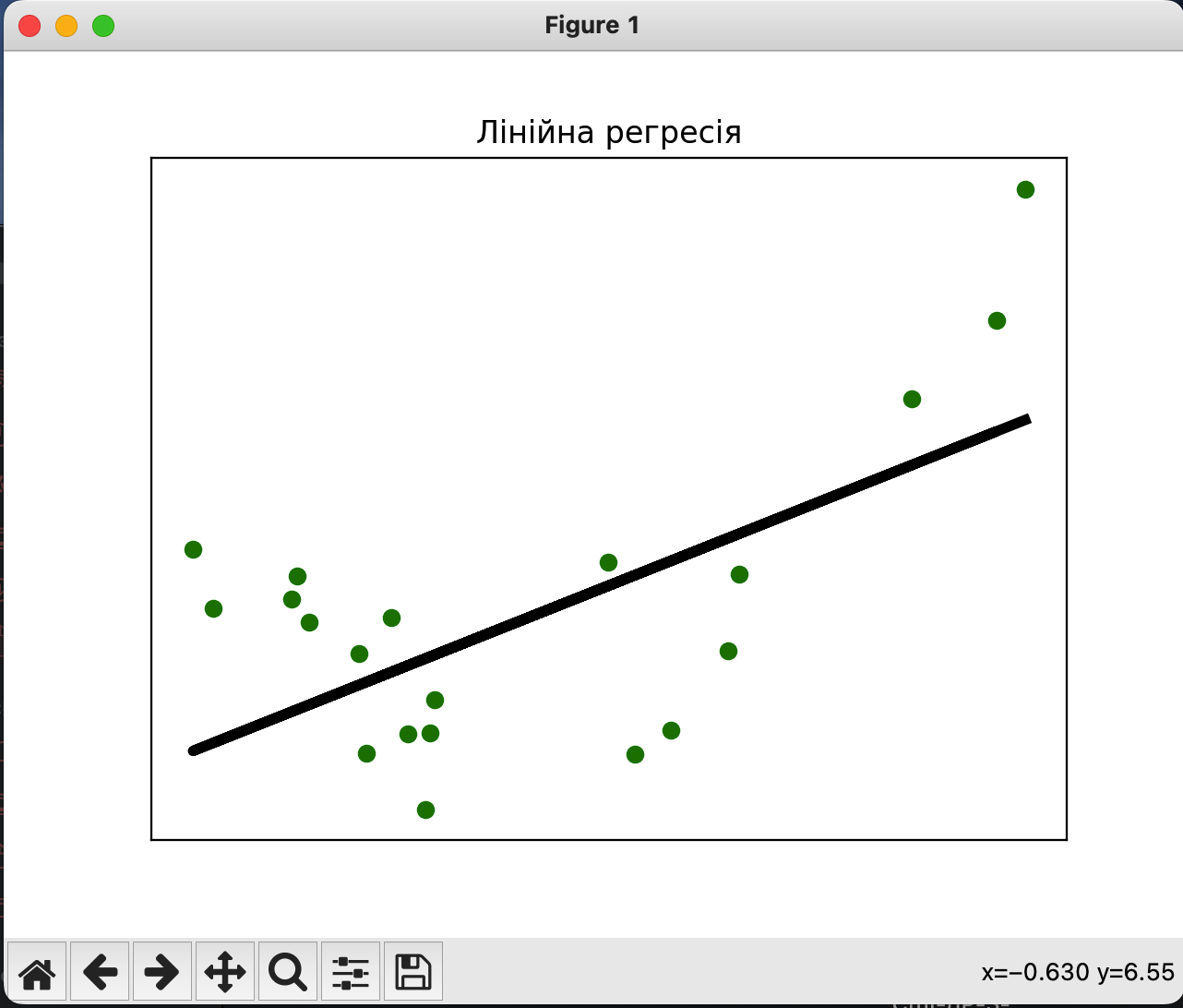
Завдання №5: Самостійна побудова регресії.

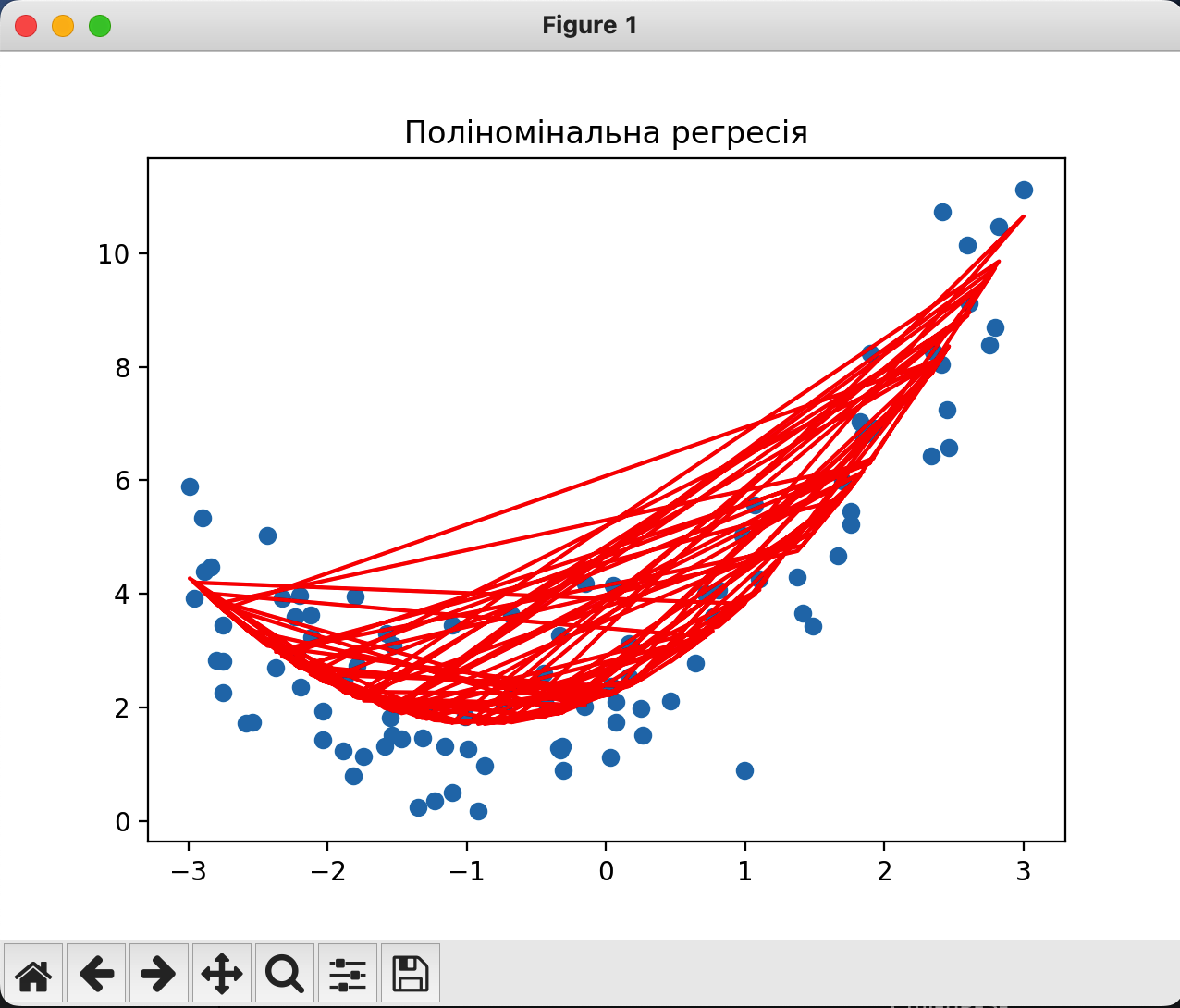
|  |  |
| --- | --- |
| № за списком | 2 |
| № варіанту | 2 |



Лістинг програми:

import pickle  
import sklearn.metrics as sm  
import numpy as np  
from sklearn import linear\_model  
from sklearn.preprocessing import PolynomialFeatures  
import matplotlib.pyplot as plt  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.6 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
num\_training = int(0.8 \* len(X))  
num\_test = len(X) - num\_training  
  
X\_train, y\_train = X[:num\_training], y[:num\_training]  
  
X\_test, y\_test = X[num\_training:], y[num\_training:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = regressor.predict(X\_test)  
  
plt.scatter(X\_test, y\_test, color='green')  
plt.title("Лінійна регресія")  
plt.plot(X\_test, y\_test\_pred, color='black', linewidth=4)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
  
poly = PolynomialFeatures(degree=2, include\_bias=False)  
poly\_features = poly.fit\_transform(X.reshape(-1, 1))  
poly\_reg\_model = linear\_model.LinearRegression()  
poly\_reg\_model.fit(poly\_features, y)  
y\_predicted = poly\_reg\_model.predict(poly\_features)  
plt.title("Поліномінальна регресія")  
plt.scatter(X, y)  
plt.plot(X, y\_predicted, c="red")  
plt.show()  
print("Intercept = ", poly\_reg\_model.intercept\_)  
print("Coef = ", poly\_reg\_model.coef\_)





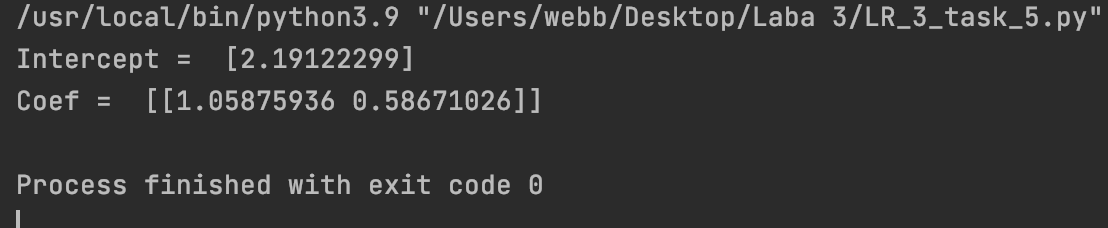


Рис. 5. Результат виконання програми

Завдання №6: Побудова кривих навчання.

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn import linear\_model  
  
def plot\_learning\_curves(model, X, y):  
 X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2)  
 train\_errors, val\_errors = [], []  
 for m in range(1, len(X\_train)):  
 model.fit(X\_train[:m], y\_train[:m])  
 y\_train\_predict = model.predict(X\_train[:m])  
 y\_val\_predict = model.predict(X\_val)  
 train\_errors.append(mean\_squared\_error(y\_train\_predict, y\_train[:m]))  
 val\_errors.append(mean\_squared\_error(y\_val\_predict, y\_val))  
 plt.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label="train")  
 plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label="val")  
 plt.legend(['Навчальний набір', 'Перевіряючий набір'])  
 plt.show()  
  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 5  
y = 0.7 \* X \*\* 2 + X + 3 + np.random.randn(m, 1)  
  
  
lin\_reg = linear\_model.LinearRegression()  
plot\_learning\_curves(lin\_reg, X, y)  
  
polynomial\_regression = Pipeline([  
 ("poly\_features",  
 PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", linear\_model.LinearRegression())  
])  
plot\_learning\_curves(polynomial\_regression, X, y)

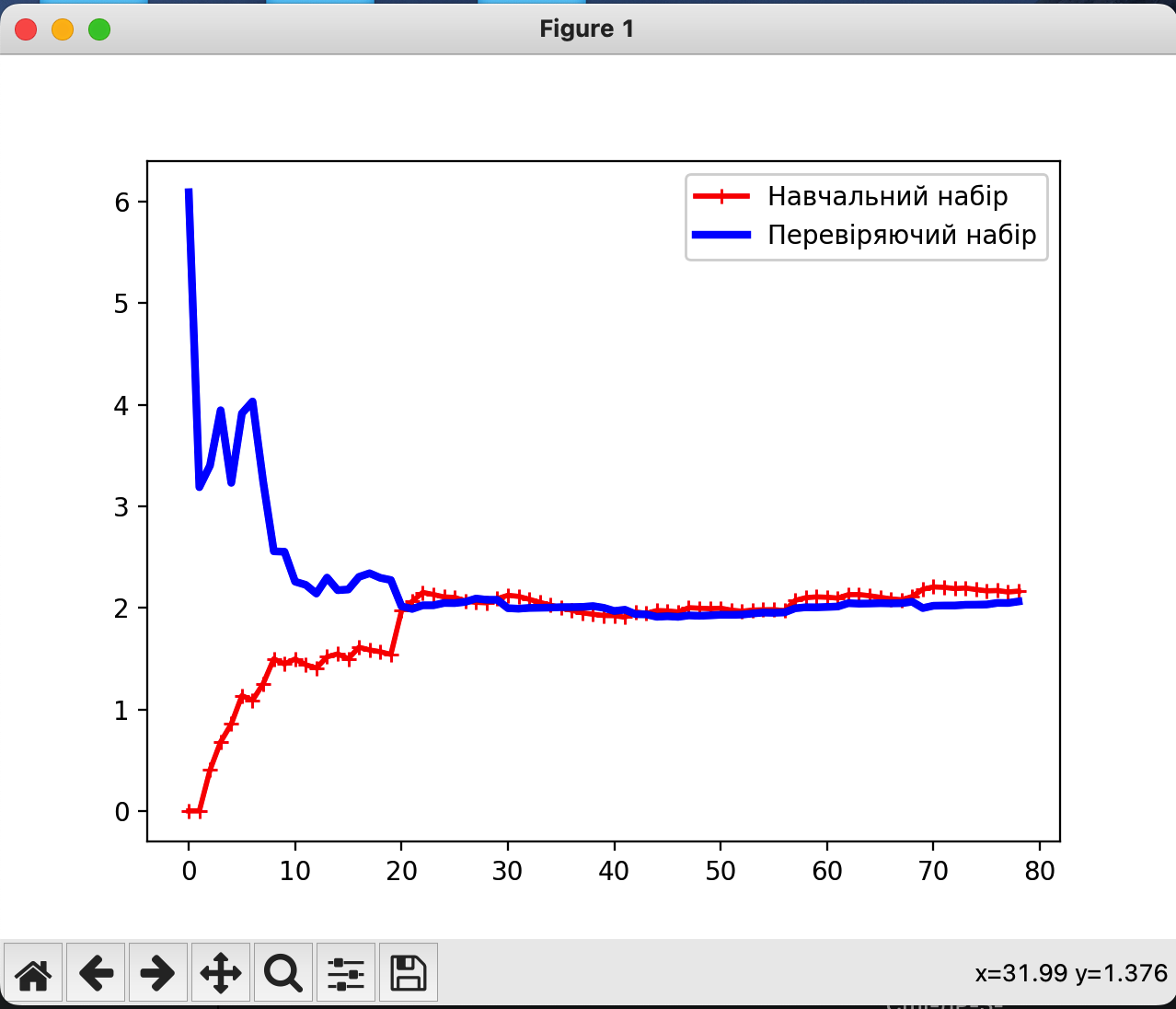


Рис. 6. Результат виконання програми

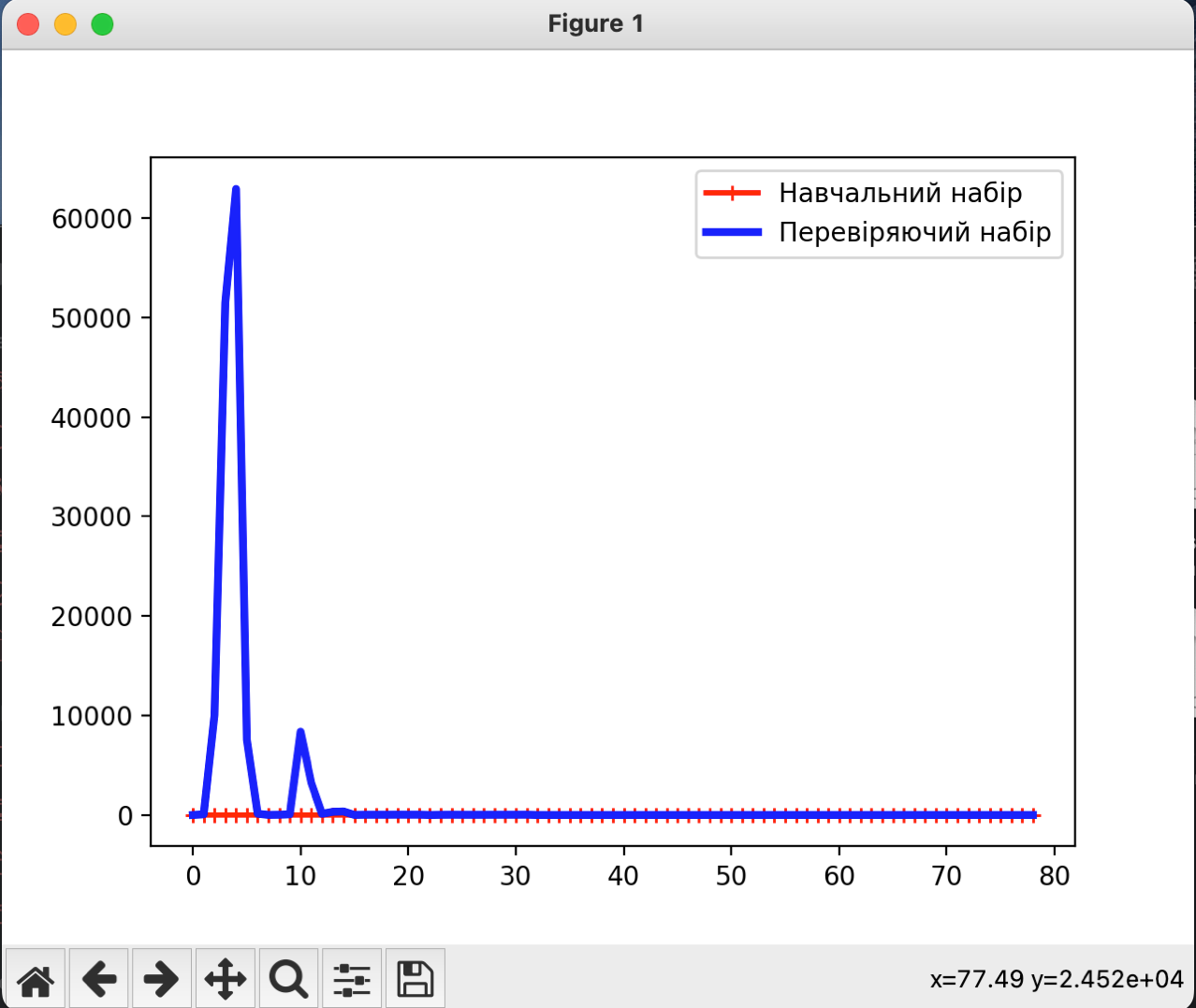


Рис. 7. Результат виконання програми

Завдання №7: Кластеризація даних за допомогою методу k-середніх.

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
  
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(())  
  
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("Claster Edges")  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
  
plt.xticks(())  
plt.yticks(())  
plt.show()

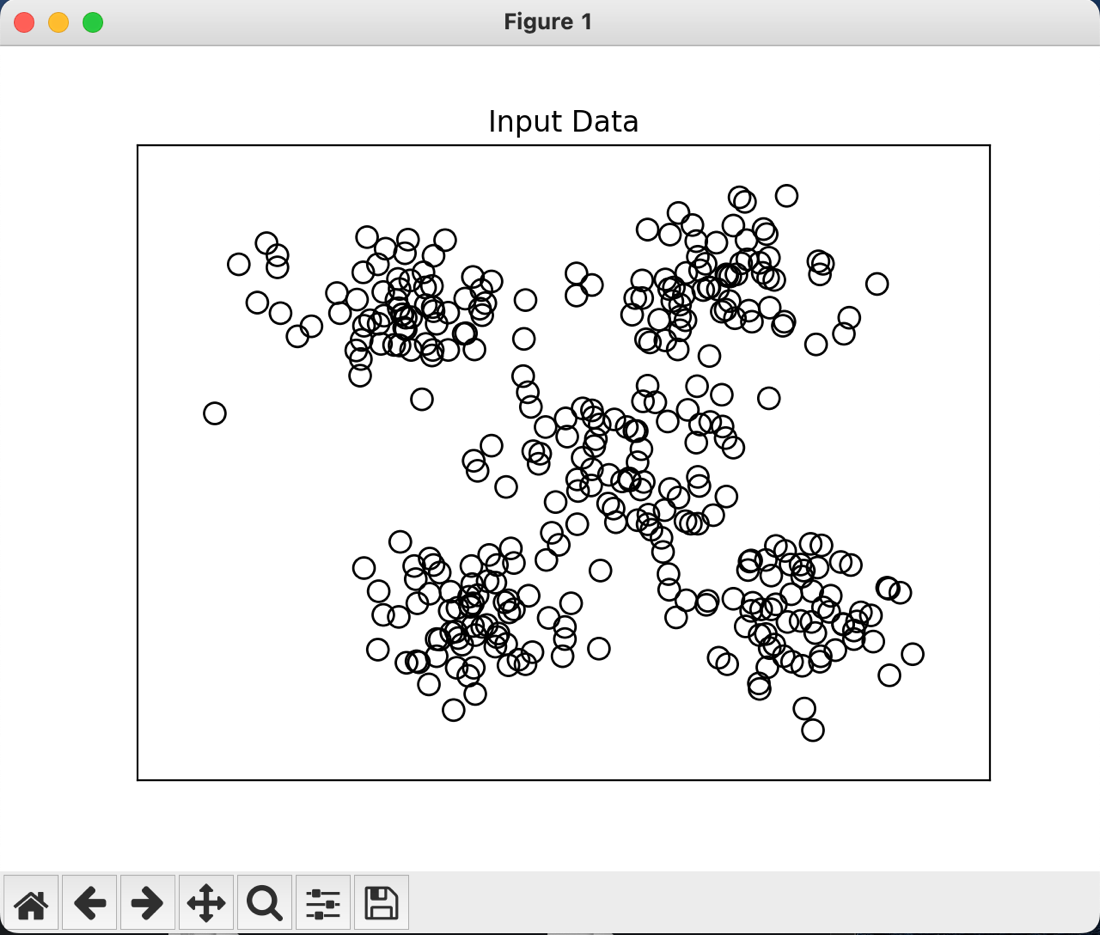


Рис. 8. Результат виконання програми

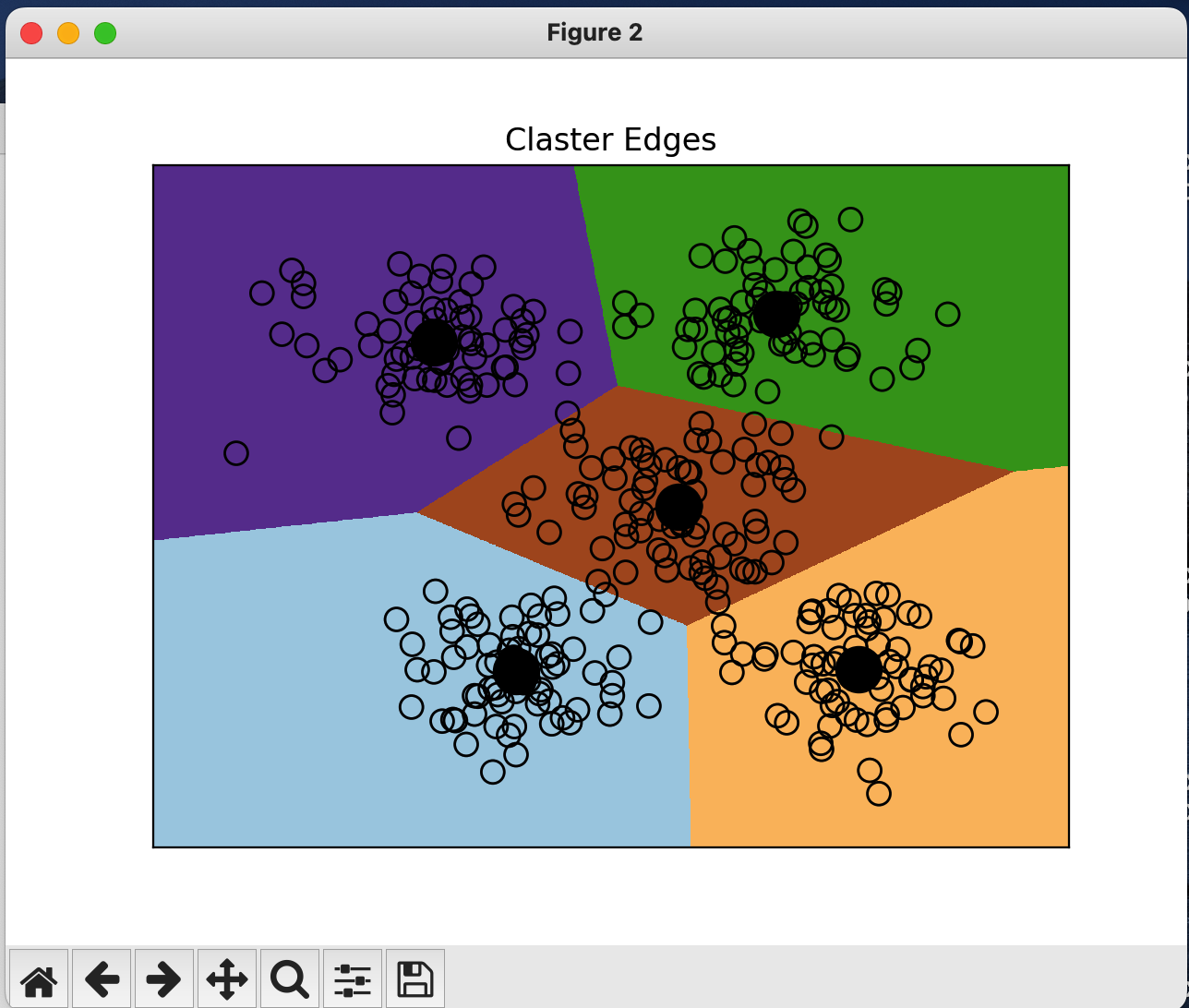


Рис. 9. Результат виконання програми

Завдання №8: Кластеризація K-середніх для набору даних Iris.

Лістинг програми:

from sklearn.metrics import pairwise\_distances\_argmin  
import numpy as np  
from sklearn.datasets import load\_iris  
from sklearn.cluster import KMeans  
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)  
  
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)  
  
  
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  
  
  
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()

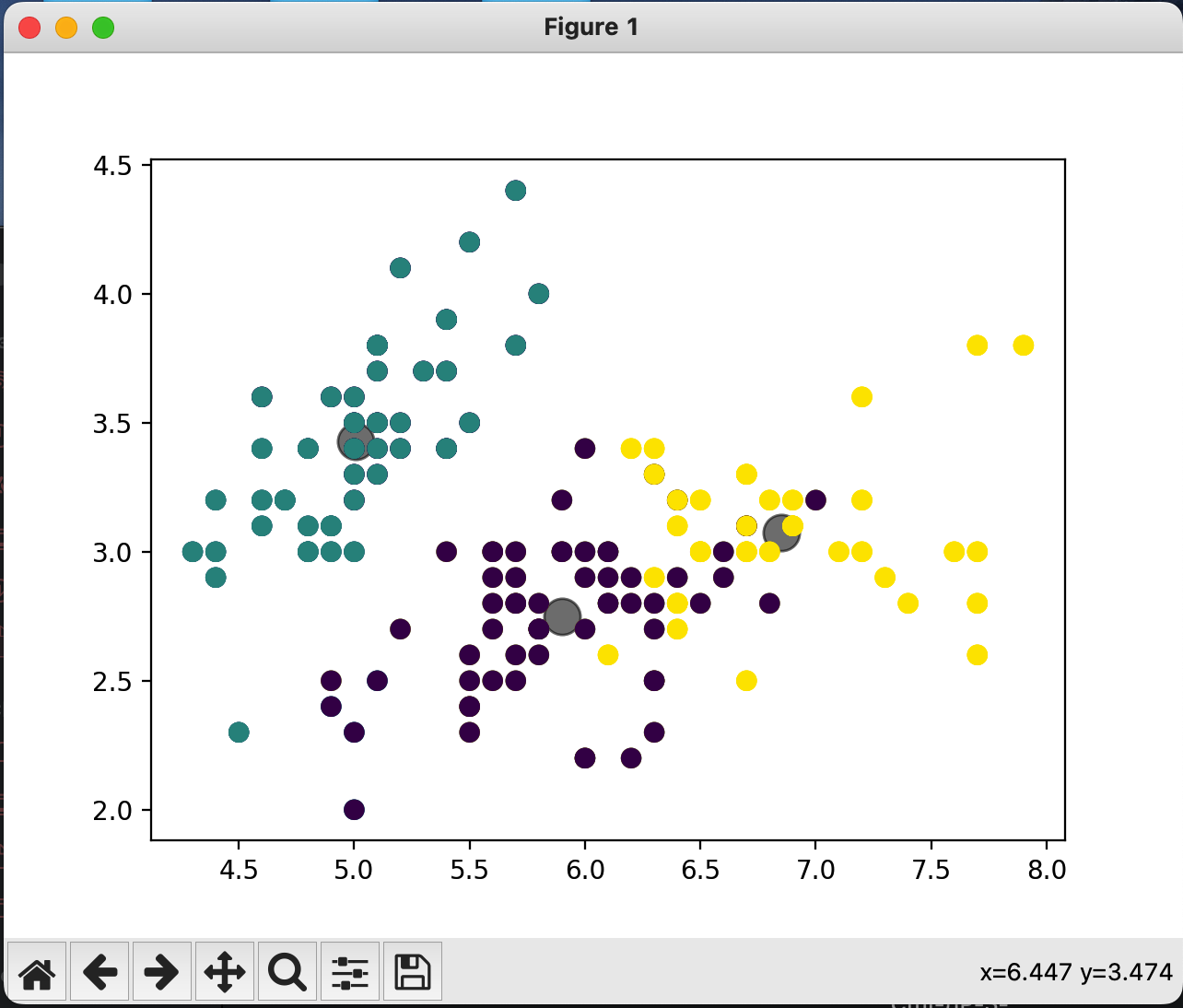
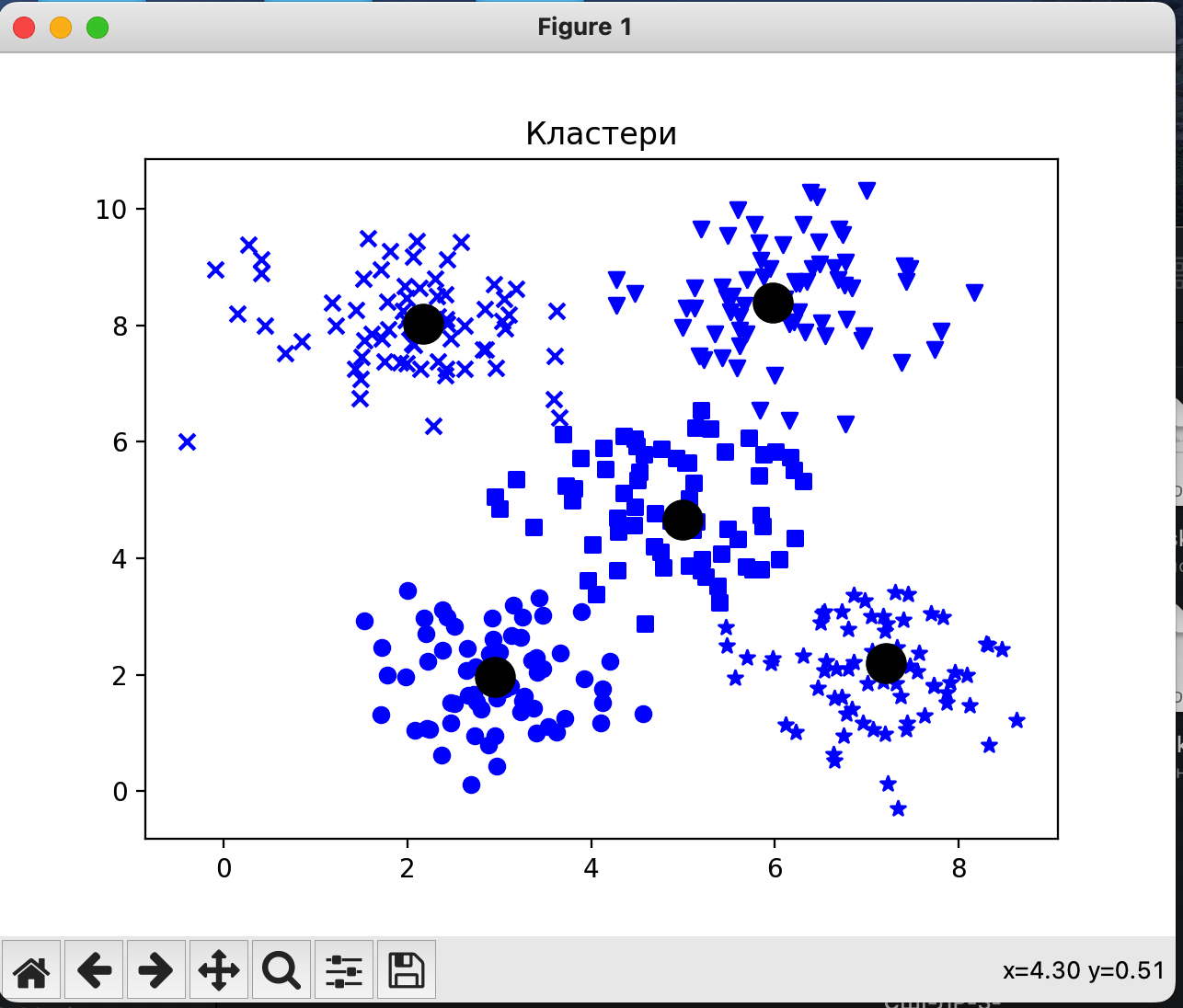


Рис. 10. Результат виконання програми

Завдання №9: Оцінка кількості кластерів з використанням методу зсуву середнього.

Лістинг програми:

import numpy as np  
import matplotlib.pyplot as plt  
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('\nCenter of clusters:\n', cluster\_centers)  
  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print('\nCenter 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, 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()



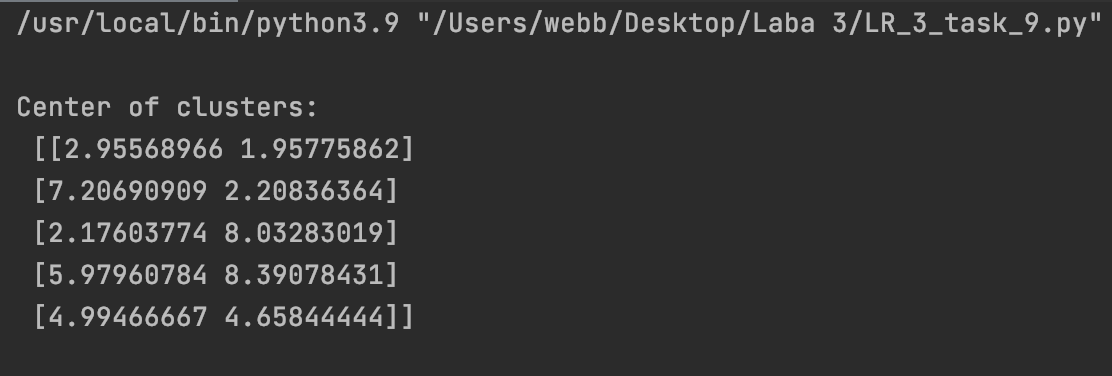


Рис. 11. Результат виконання програми

Завдання №10: Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.

Лістинг програми:

import datetime  
import json  
import numpy as np  
from sklearn import covariance, cluster  
import matplotlib  
  
matplotlib.use('TkAgg')  
from matplotlib.finance import quotes\_historical\_yahoo\_ochl as quotes\_yahoo  
  
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 = 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]  
  
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)  
  
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()  
  
for i in range(num\_labels + 1):  
 print("Cluster", i + 1, "==>", ','.join(names[labels == i]))

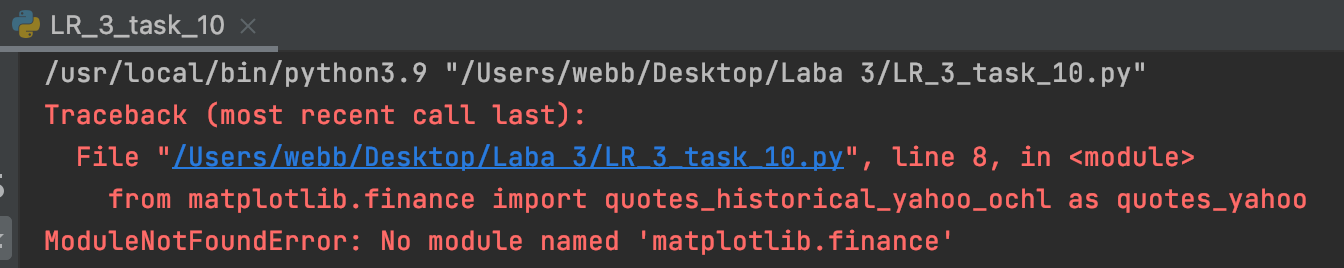


Рис. 12. Результат виконання програми

Виконати поставлене завдання неможливо, бо пакет, що використовується для отримання початкових даних, вже застарілий і отримати неможливо.

**Висновки:** було досліджено методи регресії та неконтрольованої класифікації даних у машинному навчанні, використовуючи спеціалізовані бібліотеки і мову програмування Python.