# ЛАБОРАТОРНА РОБОТА № 3
# ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

**Завдання 2.1** Створення регресора однієї змінної.

```python
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'

# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Тренувальні дані
X_train, y_train = X[:num_training], y[:num_training]

# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

# Створення об'єкта лінійного регресора
regressor = linear_model.LinearRegression()

regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = regressor.predict(X_test)

# Побудова графіка
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()

# Обрахування метрик
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))

# Файл для збереження моделі
output_model_file = 'model.pkl'

# Збереження моделі
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

# Завантаження моделі
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))

```

Рис 1. Код програми файлу LR_3_task_1m

```
Linear regressor performance:
Mean absolute error = 0.59
Mean squared error = 0.49
Median absolute error = 0.51
Explain variance score = 0.86
R2 score = 0.86

New mean absolute error = 0.59
```

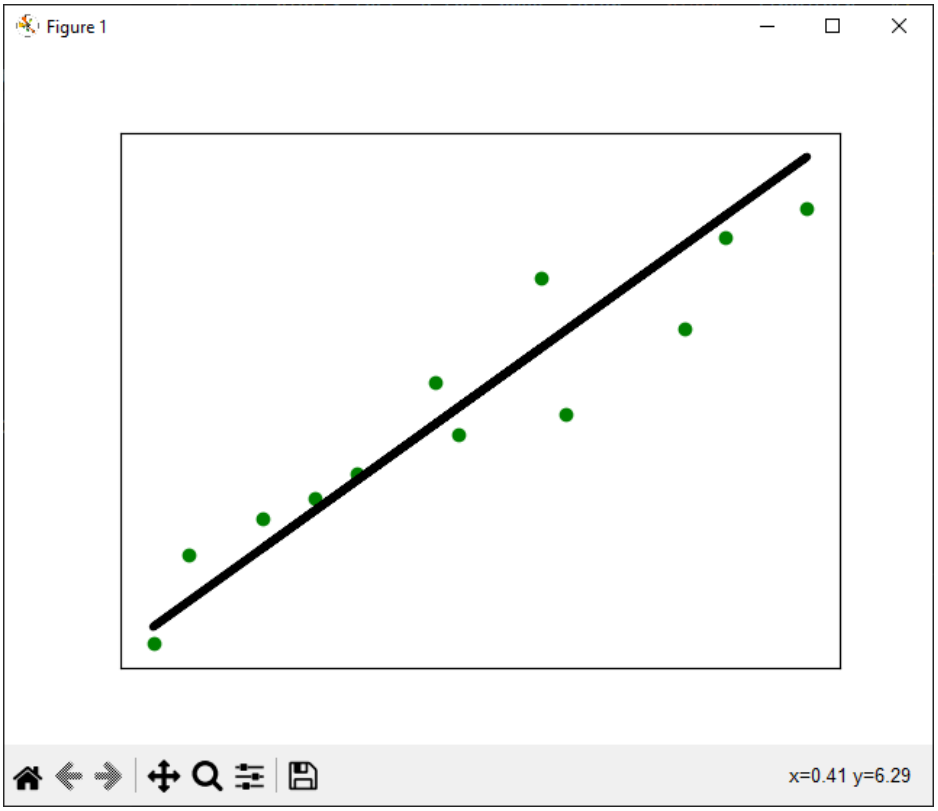



Рис 2. Результат виконання коду файлу LR_3_task_1

**Висновок:** модель для вихідних даних побудована валідно. MAE, MSE – середня якість. Показник R2 – добре.

**Завдання 2.2.** Передбачення за допомогою регресії однієї змінної.

Номер – 18

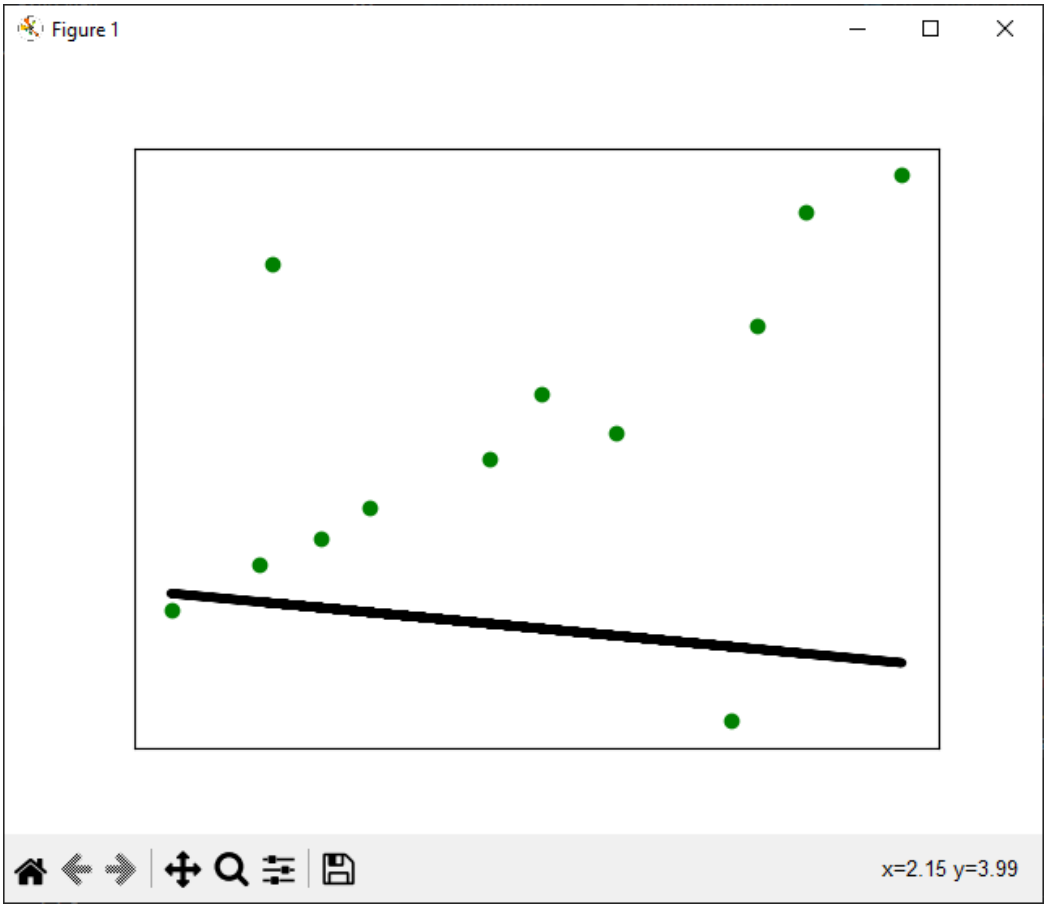
Варіант – 3

| № за списком | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| № варіанту | 1 | 2 | 3 | 4 | 5 | 1 | 2 | 3 | 4 | 5 |

Варіант 3 файл: data_regr_3.txt

```python
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt

# Вхідний файл, який містить дані
input_file = 'data_regr_3.txt'

# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Тренувальні дані
X_train, y_train = X[:num_training], y[:num_training]

# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

# Створення об'єкта лінійного регресора
regressor = linear_model.LinearRegression()

# Тренування моделі
regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = regressor.predict(X_test)

# Побудова графіка
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()

# Обрахування метрик
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))

# Файл для збереження моделі
output_model_file = 'model.pkl'

# Збереження моделі
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

# Завантаження моделі
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))

```

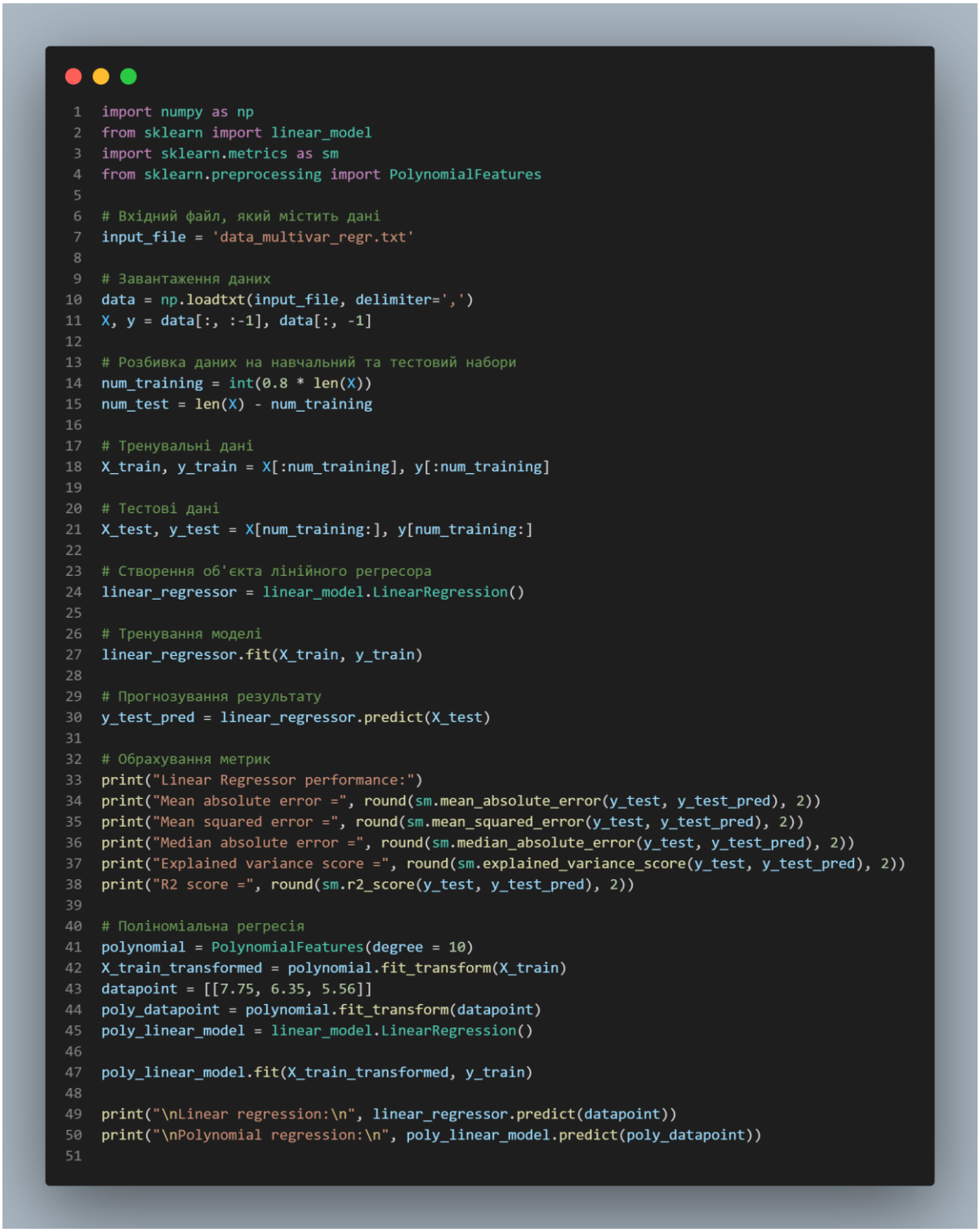
Рис 3. Код програми файлу LR_3_task_2




```
Linear regressor performance:
Mean absolute error = 2.42
Mean squared error = 9.02
Median absolute error = 2.14
Explain variance score = -0.15
R2 score = -1.61

New mean absolute error = 2.42
```

Рис 4. Результат виконання коду файлу LR_3_task_2

**Завдання 2.3.** Створення багатовимірного регресора.



```python
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures

# Вхідний файл, який містить дані
input_file = 'data_multivar_regr.txt'

# Завантаження даних
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Тренувальні дані
X_train, y_train = X[:num_training], y[:num_training]

# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]

# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()

# Тренування моделі
linear_regressor.fit(X_train, y_train)

# Прогнозування результату
y_test_pred = linear_regressor.predict(X_test)

# Обрахування метрик
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("Explained 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))

# Поліноміальна регресія
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", linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n", poly_linear_model.predict(poly_datapoint))
```

Рис 5. Код програми файлу LR_3_task_3

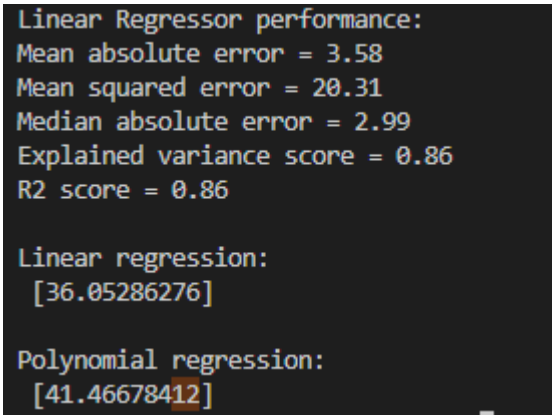



Рис 6. Результат виконання коду файлу LR_3_task_3

**Висновок:** Порівнюючи з лінійним регресором, полономіальний регресор більш кращий, тобто дозволяє показувати кращі результати.

**Завдання 2.4.** Регресія багатьох змінних.

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split

diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size = 0.5, random_state = 0)
regr = linear_model.LinearRegression()

regr.fit(Xtrain, ytrain)

ypred = regr.predict(Xtest)

# Обрахування метрик
print("regr.coef =", np.round(regr.coef_, 2))
print("regr.intercept =", round(regr.intercept_,  2))
print("R2 score =", round(r2_score(ytest, ypred), 2))
print("Mean absolute error =", round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =", round(mean_squared_error(ytest, ypred), 2))

fig, ax = plt.subplots()

ax.scatter(ytest, ypred, edgecolors = (0, 0, 0))
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw = 4)
ax.set_xlabel('Виміряно')
ax.set_ylabel('Передбачено')
plt.show()

```

Рис 7. Код програми файлу LR_3_task_4

```
PS C:\ztu\штучний інтелект\lab3> python LR_3_task_4.py
regr.coef = [ -20.4  -265.89  564.65  325.56 -692.16  395.56   23.5   116.36  843.95
   12.72]
regr.intercept = 154.36
R2 score = 0.44
Mean absolute error = 44.8
Mean squared error = 3075.33
PS C:\ztu\штучний інтелект\lab3>
```

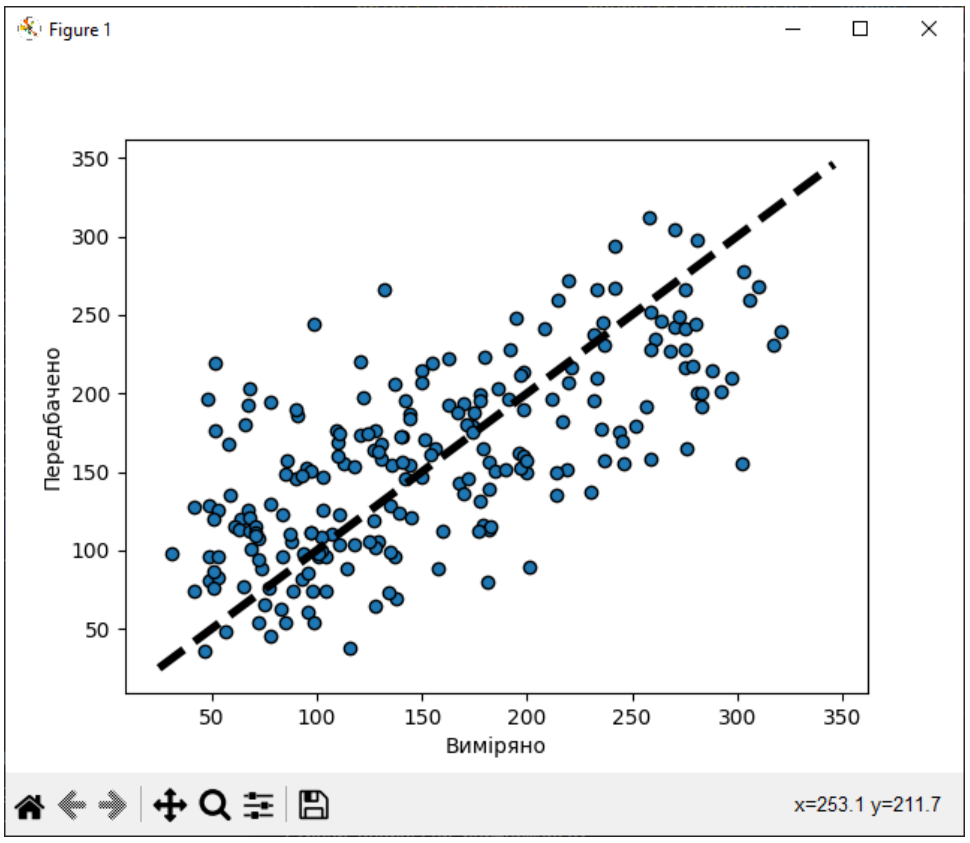
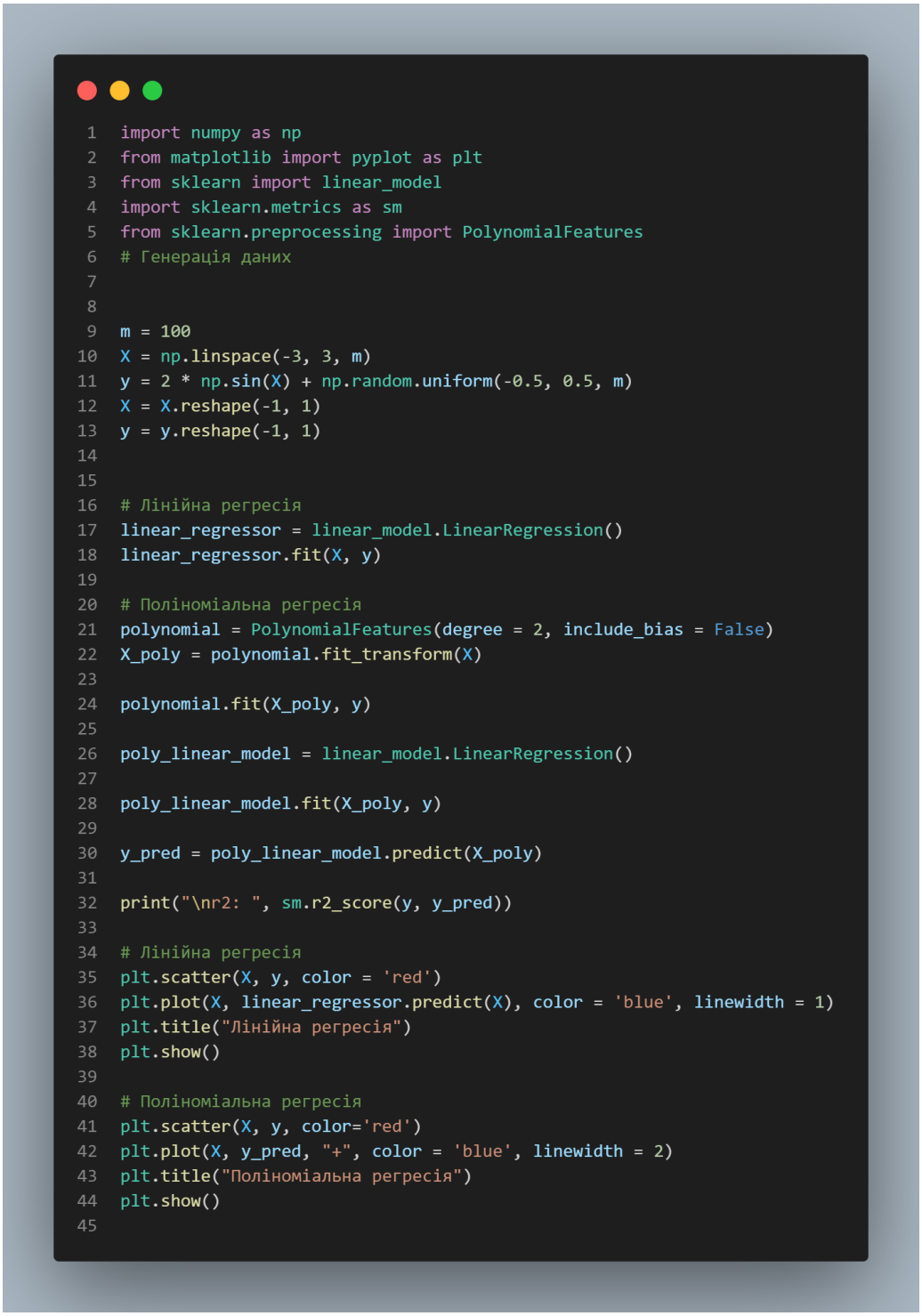



Рис 8. Результат виконання коду файлу LR_3_task_4

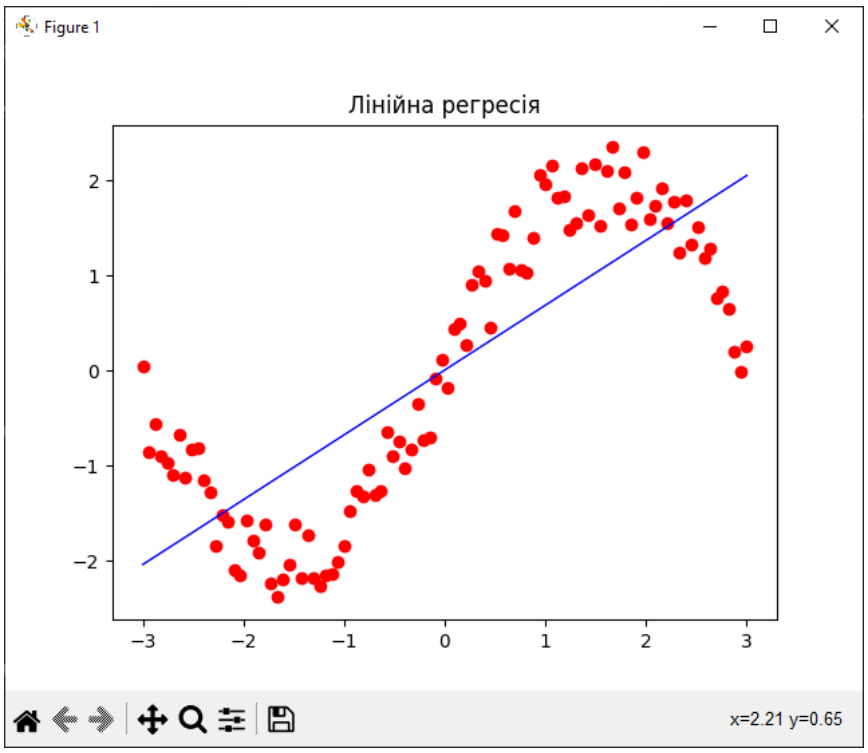
**Завдання 2.5.** Самостійна побудова регресії.

| № за списком | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| № варіанту | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |



```python
import numpy as np
from matplotlib import pyplot as plt
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures
# Генерація даних


m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.5, 0.5, m)
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)


# Лінійна регресія
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X, y)

# Поліноміальна регресія
polynomial = PolynomialFeatures(degree = 2, include_bias = False)
X_poly = polynomial.fit_transform(X)

polynomial.fit(X_poly, y)

poly_linear_model = linear_model.LinearRegression()

poly_linear_model.fit(X_poly, y)

y_pred = poly_linear_model.predict(X_poly)

print("\nr2: ", sm.r2_score(y, y_pred))

# Лінійна регресія
plt.scatter(X, y, color = 'red')
plt.plot(X, linear_regressor.predict(X), color = 'blue', linewidth = 1)
plt.title("Лінійна регресія")
plt.show()

# Поліноміальна регресія
plt.scatter(X, y, color='red')
plt.plot(X, y_pred, "+", color = 'blue', linewidth = 2)
plt.title("Поліноміальна регресія")
plt.show()

```

Рис 9. Код програми файлу LR_3_task_5



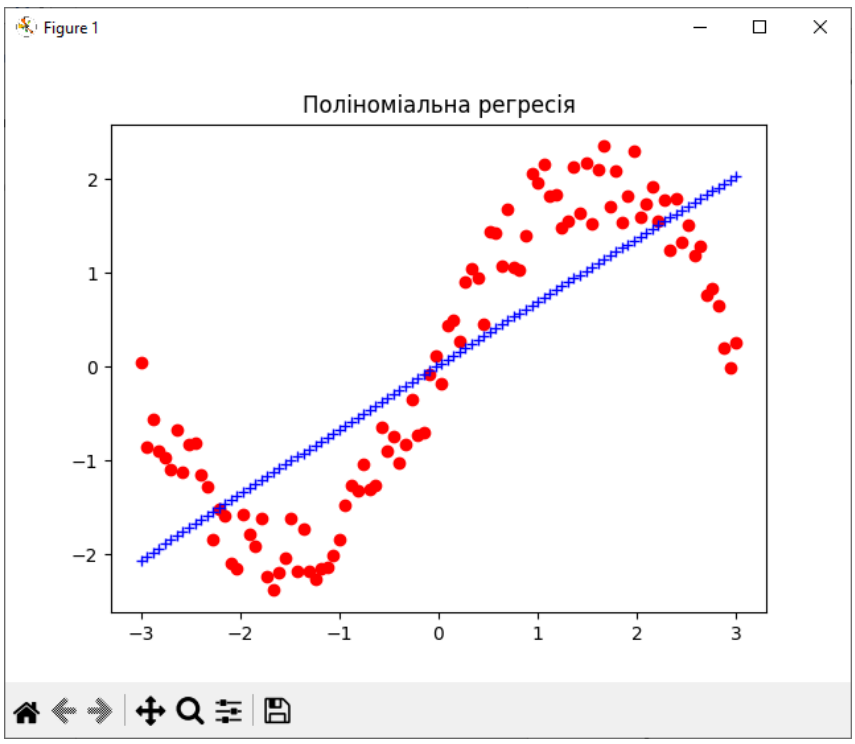



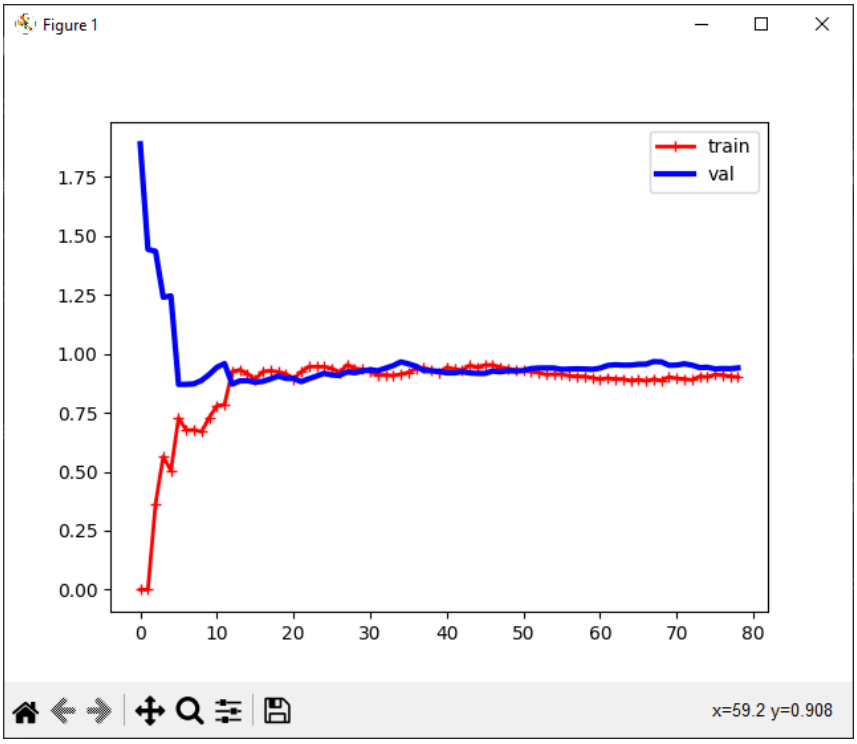

Рис 10. Результат виконання коду файлу LR_3_task_5

**Завдання 2.6.** Побудова кривих навчання.

| № за списком | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|
| № варіанту | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

```python
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import Pipeline

# Генерація даних
m = 100
X = np.linspace(-3, 3, m)
y = 2 * np.sin(X) + np.random.uniform(-0.5, 0.5, m)
X = X.reshape(-1, 1)
y = y.reshape(-1, 1)

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()
    plt.show()

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 11. Код програми файлу LR_3_task_6



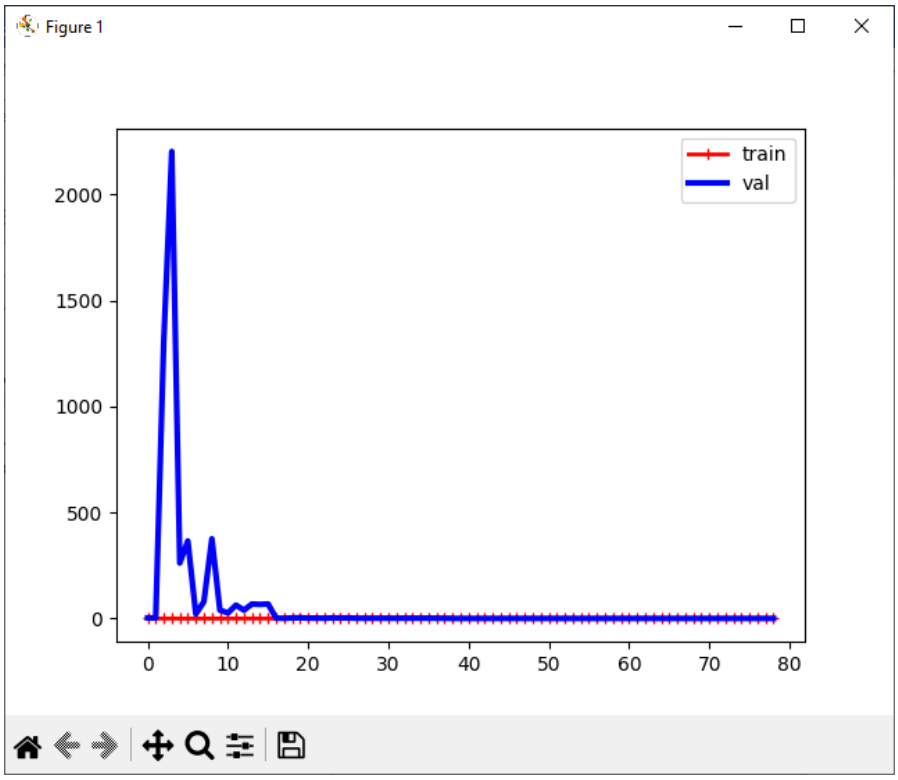




Рис 12. Результат виконання коду файлу LR_3_task_6

**Висновок:** для з'ясування ступеня складності необхідної моделі ми можемо використати криві навчання. Для досягнення успіху необхідно досягти компромісу

між зміщенням та дисперсією. В нашому випадку найкращий результат показала модель 2 ступеня.

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

# Завантаження вхідних даних
X = np.loadtxt('data_clustering.txt', delimiter=',')
num_clusters = 5

# Включення вхідних даних до графіка
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('Вхідні дані')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

# Створення об'єкту KMeans
kmeans = KMeans(init='k-means++', n_clusters = num_clusters, n_init = 10)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Визначення кроку сітки
step_size = 0.01

# Відображення точок сітки
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))

#  Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

# Графічне відображення областей та виділення їх кольором
output = output.reshape(x_vals.shape)

plt.figure()
plt.clf()
```

```python
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'

)

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], marker = 'o', facecolors = 'none', edgecolors = 'black', s = 80)

# Відображення центрів кластерів
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('Межі кластерів')
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())
plt.show()
```

**Завдання 2.7.** Кластеризація даних за допомогою методу k-середніх.

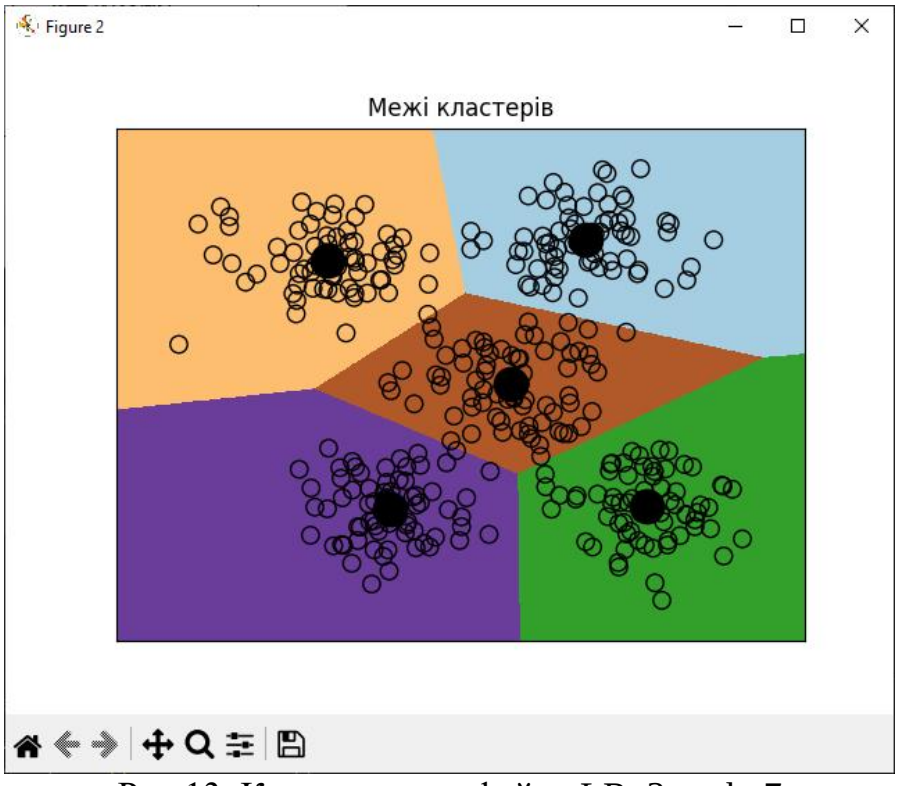



Рис 13. Код програми файлу LR_3_task_7

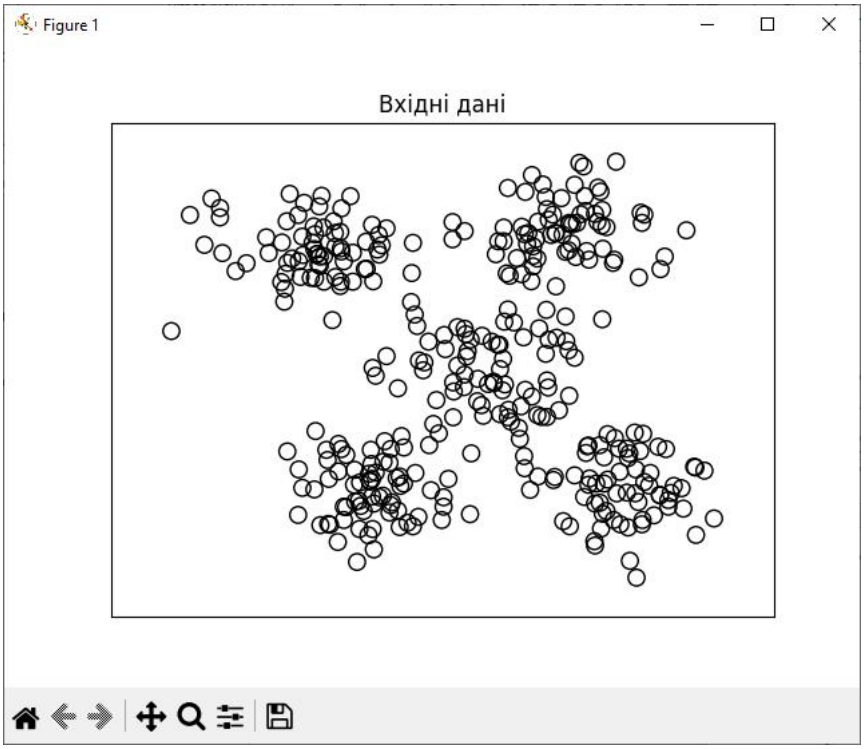



Рис 14. Результат виконання коду файлу LR_3_task_7

**Висновок:** метод k-середніх валідно працює, але за умови, відомої кількісті кластерів.

**Завдання 2.8.** Кластеризація K-середніх для набору даних Iris.

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

# Отримуємо дані
iris = datasets.load_iris()
X = iris.data[:, :2]
Y = iris.target

# Визначаємо початкові кластери
kmeans = KMeans(
    n_clusters = Y.max() + 1,
    init = 'k-means++',
    n_init = 10,
    max_iter = 300,
    tol = 0.0001,
    verbose = 0,
    random_state = None,
    copy_x = True
)

kmeans.fit(X)

y_pred = kmeans.predict(X)

print("n_clusters: 3, n_init: 10, max_iter: 300, tol: 0.0001, verbose: 0, ran-dom_state: None, copy_x: True")
print(y_pred)

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

centers = kmeans.cluster_centers_

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

def find_clusters(X, n_clusters, rseed = 2):
    # Випадково обираємо кластери
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # Оголошуємо label базуючись на найближчому центрі
        labels = pairwise_distances_argmin(X, centers)

        # Знаходимо нові центри з середини точок
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clusters)])

        # Перевірка збіжності
        if np.all(centers == new_centers):
            break

        centers = new_centers

    return centers, labels

print("using find_clusters():")

centers, labels = find_clusters(X, 3)

print("n_clusters: 3, rseed: 2")

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

centers, labels = find_clusters(X, 3, rseed = 0)

print("n_clusters: 3, rseed: 0")

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

labels = KMeans(3, random_state = 0).fit_predict(X)

print("n_clusters: 3, rseed: 0")

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

Рис 15. Код програми файлу LR_3_task_8

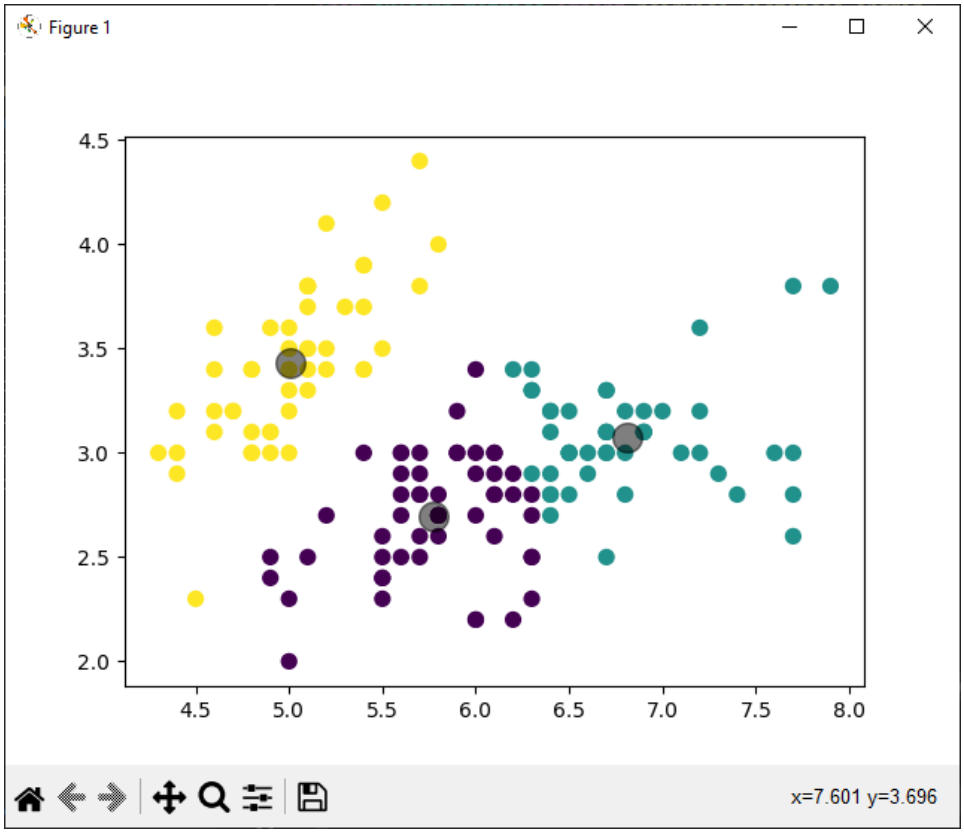


Figure 1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
x=7.601 y=3.696

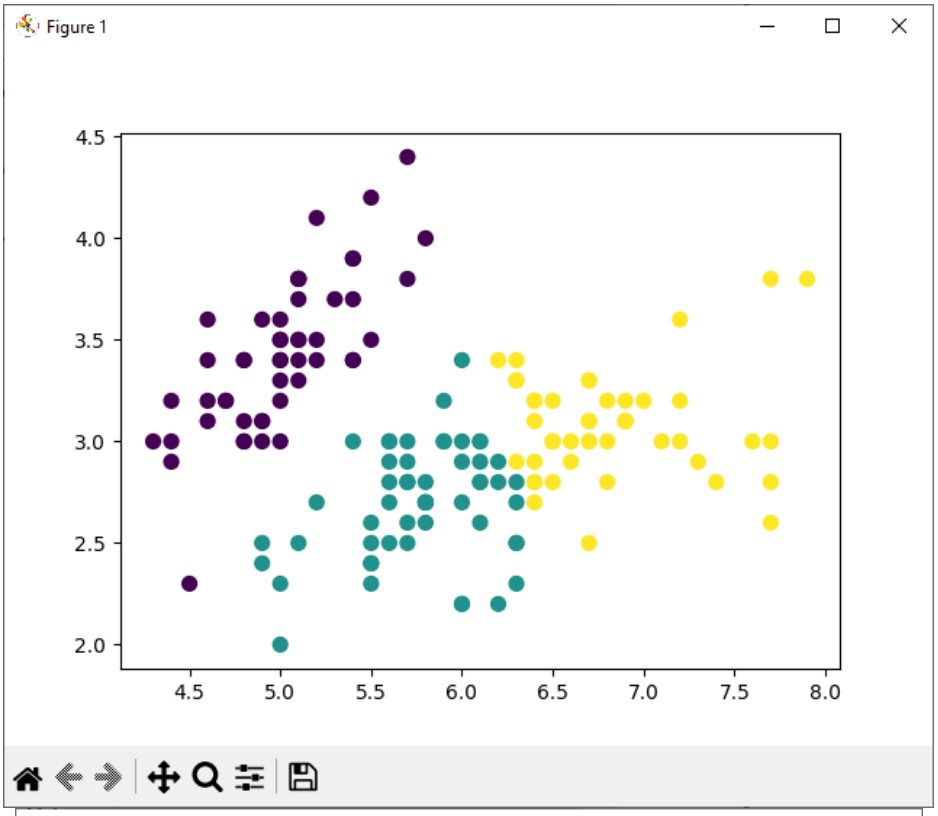


Figure 1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0

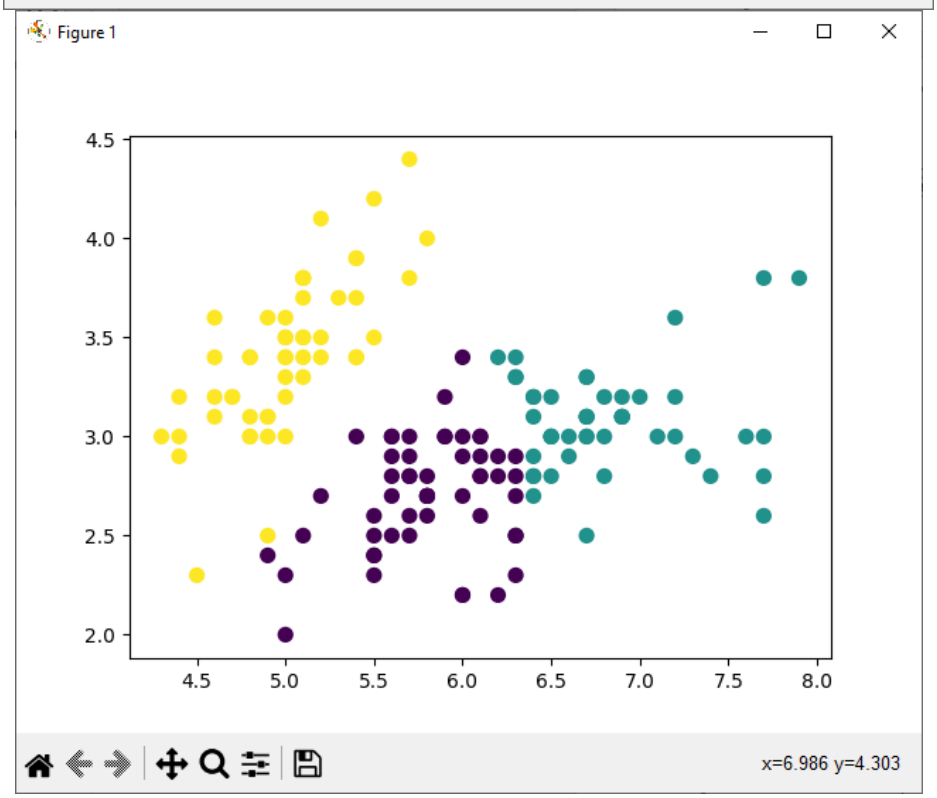


Figure 1
4.5
4.0
3.5
3.0
2.5
2.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
x=6.986 y=4.303

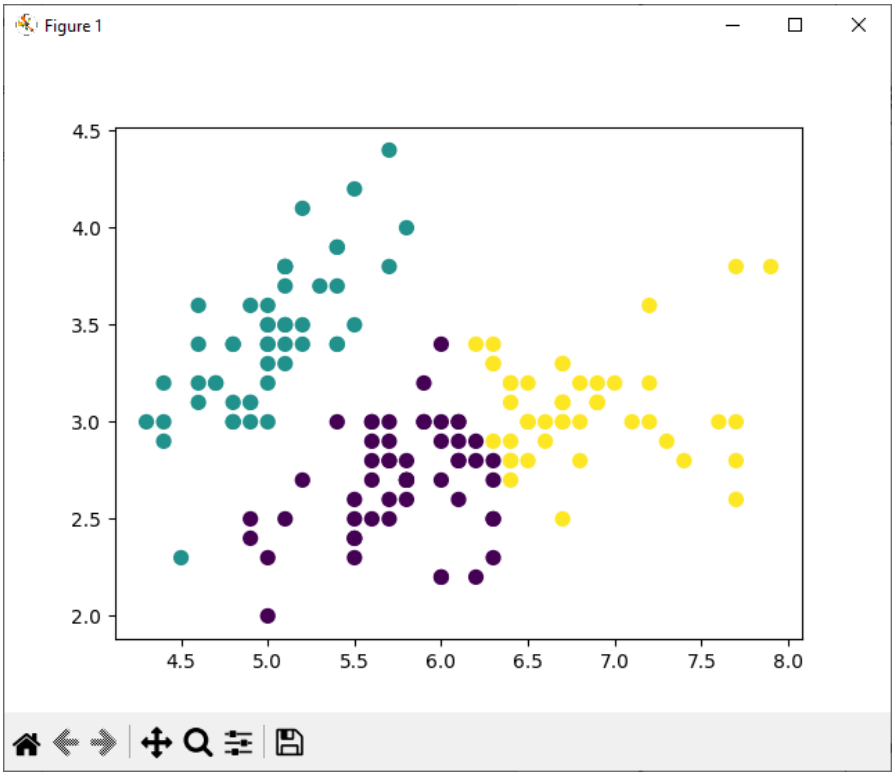
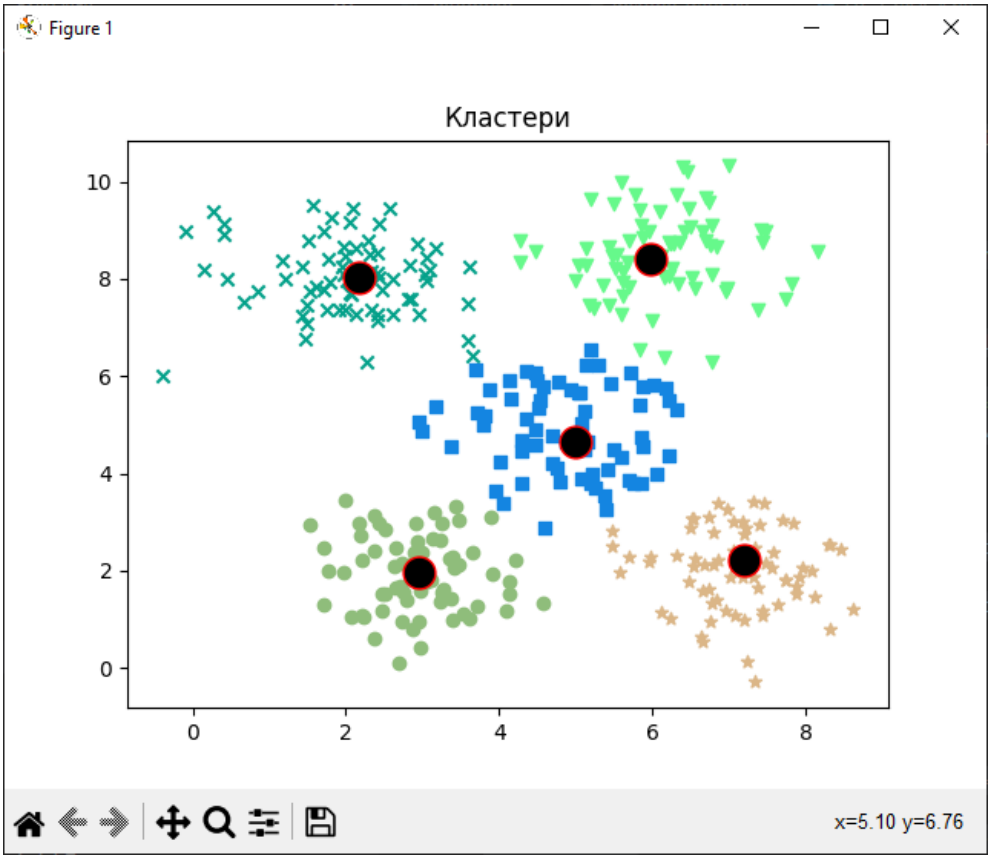



Рис 16. Результат виконання коду файлу LR_3_task_8

**Завдання 2.9.** Оцінка кількості кластерів з використанням методу зсуву середнього.

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle

# Завантаження даних
X = np.loadtxt('data_clustering.txt', delimiter = ',')

# Оцінка ширини вікна для X
bandwidth_X = estimate_bandwidth(X, quantile = 0.1, n_samples = len(X))

# Кластеризація даних методом зсуву середнього
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding = True)
meanshift_model.fit(X)

# Витягування центрів кластерів
cluster_centers = meanshift_model.cluster_centers_

print('\nCenters of clusters:\n', cluster_centers)

# Оцінка кількості кластерів
labels = meanshift_model.labels_
num_clusters = len(np.unique(labels))

print("\nNumber of clusters in input data =", num_clusters)

# Відображення на графіку точок та центрів кластерів
plt.figure()

markers = 'o*xvs'

for i, marker in zip(range(num_clusters), markers):
    # Відображення на графіку точок, що належать поточному кластеру
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker = marker,
                color = np.random.rand(3,))

    # Відображення на графіку центру кластера
    cluster_center = cluster_centers[i]

    plt.plot(
        cluster_center[0],
        cluster_center[1],
        marker = 'o',
        markerfacecolor = 'black',
        markeredgecolor = 'red',
        markersize = 15
    )

plt.title('Кластери')
plt.show()

```

Рис 17. Код програми файлу LR_3_task_9




```
PS C:\ztu\штучний інтелект\lab3> python LR_3_task_9.py

Centers of clusters:
 [[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
```

Рис 18. Результат виконання коду файлу LR_3_task_9

Метод зсуву середнього – доволі валідний алгоритм, головною перевагою якого є непотрібність жодних припущень щодо базового розподілу даних, має змогу обробляти довільні простори функцій, проте важливу роль відіграє обрана ширина вікна (bandwidth).

**Висновок:** під час виконання лабораторної роботи я навчився: використовувати спеціалізовані бібліотеки і мову програмування Python, досліджувати методи регресії та неконтрольованої класифікації даних у машинному навчанні.