ЛАБОРАТОРНА РОБОТА № 3

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ

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

Хід роботи:

Завдання 1

Лістинг програми:

```
from google.colab import files

uploaded = files.upload()

for filename in uploaded.keys():
    print(f'Завантажено файл {filename} з розміром {len(uploaded[filename])} байт')



# Вхідний файл, який містить дані
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()

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:
    loaded_regressor = pickle.load(f)

# Виконання передбачень з завантаженою моделлю
y_test_pred_new = loaded_regressor.predict(X_test)

# Розрахунок середньої абсолютної похибки
print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

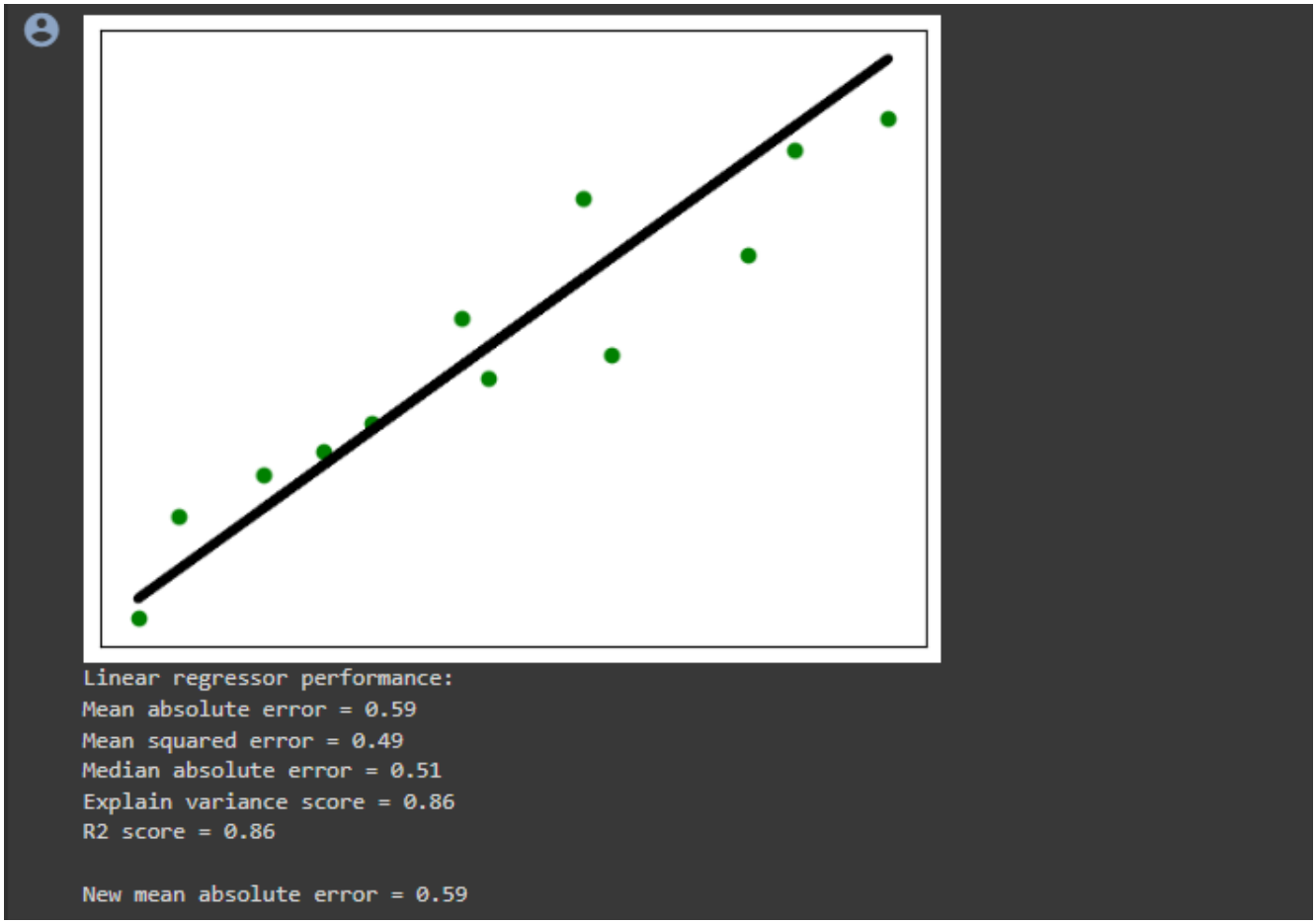


Рис.1. Графік функції. Результат прогнозування та результат оцінки якості.

Висновки: після виконання завдання ми провели прогнозування на тестових даних за допомогою лінійної регресії, а потім зберегли та завантажили отриману модель. Графік показує, що ситуація є типовою для лінійної регресії. Більша частина даних виявляє невелику варіацію і розташована близько до лінії, що свідчить про ефективність моделі лінійної регресії для апроксимації цих даних.

Щодо результатів прогнозування:

Mean Absolute Error, Mean Squared Error та Median Absolute Error, які коливаються від 0.49 до 0.59, свідчать про те, що отримані відхилення від ідеальних значень досить невеликі. Це може вважатися прийнятними результатами, але їх оцінка також залежить від конкретної задачі та сфери використання моделі.

Модель досягла значення Explained Variance Score приблизно 0.86, що свідчить про її високу здатність пояснювати зміни в даних. Іншими словами, модель успішно пояснює дисперсію в даних.

Значення R2 близьке до 0.86 означає, що приблизно 86% варіації у цільовій змінній можна пояснити за допомогою даної моделі лінійної регресії.

Після збереження та завантаження моделі ми отримали значення Mean Absolute Error приблизно 0.59, що підтверджує, що збережена модель працює на тестових даних так само, як і оригінальна модель.

Завдання 2

Лістинг програми:

```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pickle
import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
import matplotlib.pyplot as plt
from google.colab import files

# Завантаження даних з файлу
file_path = 'data_regr_5.txt'  # Вкажіть правильний шлях до вашого файлу
data = np.loadtxt(file_path, delimiter=',')
# Розділення даних на вхідну змінну (X) та відгук (y)
X = data[:, 0]  # Вкажіть правильний індекс стовпця для вашої змінної
y = data[:, 1]  # Вкажіть правильний індекс стовпця для відгуку
# Розділення даних на навчальний та тестовий набори
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Створення та навчання моделі
regressor = LinearRegression()
regressor.fit(X_train.reshape(-1, 1), y_train)

# Прогнозування на тестовому наборі
y_pred = regressor.predict(X_test.reshape(-1, 1))

# Відображення результуючої регресійної моделі
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red', linewidth=3)
plt.xlabel('Вхідна змінна')
plt.ylabel('Відгук')
plt.show()
```

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import mean_absolute_error, mean_squared_error,
median_absolute_error, explained_variance_score, r2_score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Середньоквадратична похибка (MSE): {mse:.2f}")
print(f"R-squared (R2): {r2:.2f}")
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
medae = median_absolute_error(y_test, y_pred)
evs = explained_variance_score(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Linear regressor performance:")
print(f"Mean absolute error: {mae:.2f}")
print(f"Mean squared error: {mse:.2f}")
print(f"Median absolute error: {medae:.2f}")
print(f"Explain variance score: {evs:.2f}")
print(f"R2 score: {r2:.2f}")

# Оцінка нового передбачення з використанням навченої моделі
y_test_pred_new = regressor.predict(X_test.reshape(-1, 1))
new_mae = mean_absolute_error(y_test, y_test_pred_new)

print(f"\nNew mean absolute error: {new_mae:.2f}")
```

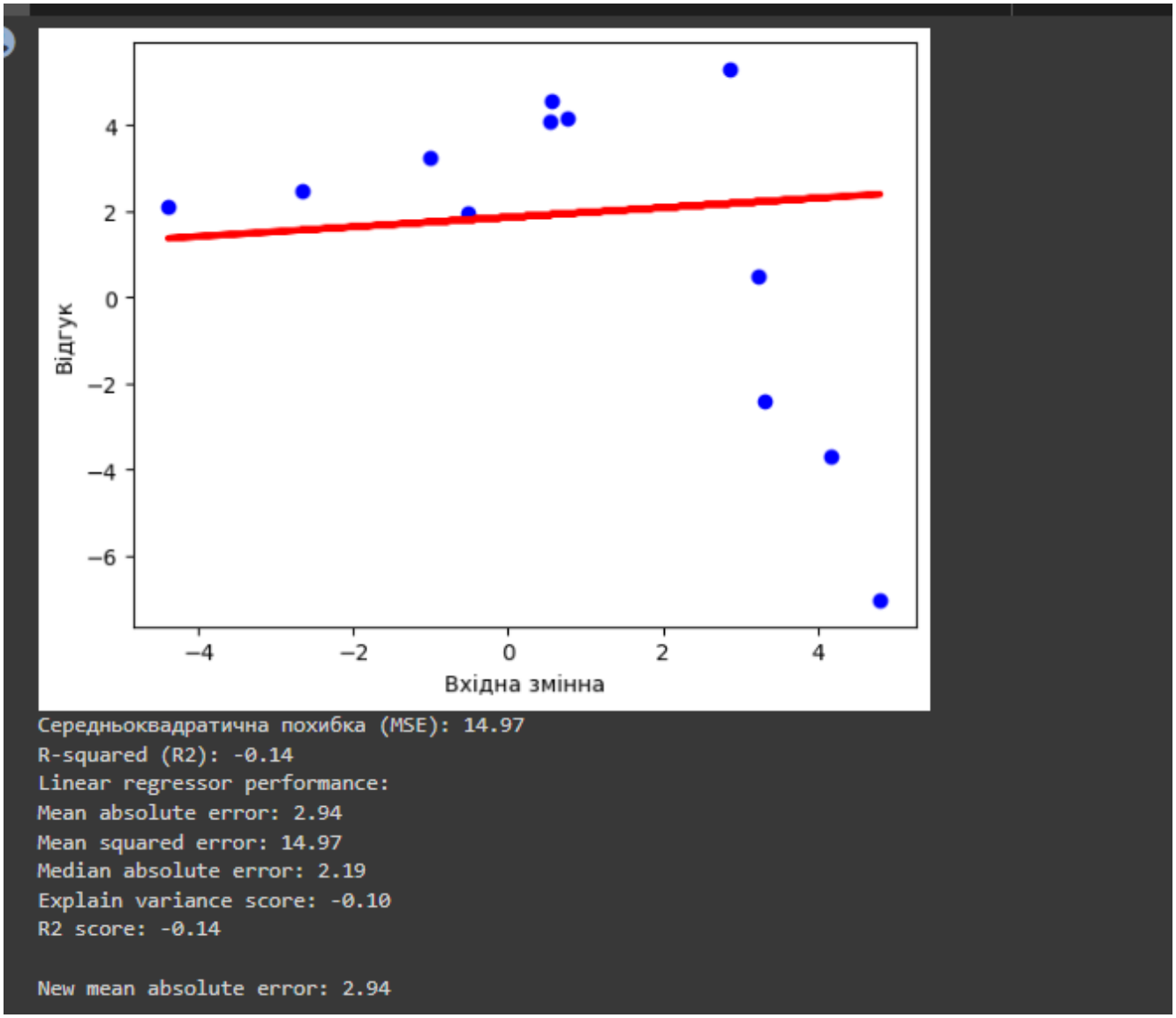


Рис.2. Графік функції. Результат прогнозування та результат оцінки якості.

Висновки: у порівнянні з завданням 1, результати роботи моделі лінійної регресії для цього конкретного набору даних виявились значно гіршими. Графік показує, що більшість даних характеризується значною варіацією і віддалено розташована від лінії, що свідчить про погану апроксимацію даних моделлю лінійної регресії.

Щодо показників:

Mean Absolute Error та Median Absolute Error зросли у 2 рази, а Mean Squared Error дорівнює 14.97, що вказує на велику середньоквадратичну похибку.

Показники Explained Variance Score і R2 Score дорівнюють -0.10 та -0.14. Це означає, що модель не в змозі пояснити варіацію у цільовій змінній, і її прогнози гірші, ніж просто середнє значення цільової змінної.

З урахуванням цього можна припустити, що лінійна регресійна модель, побудована на основі однієї змінної, не є задовільною для цих конкретних даних.

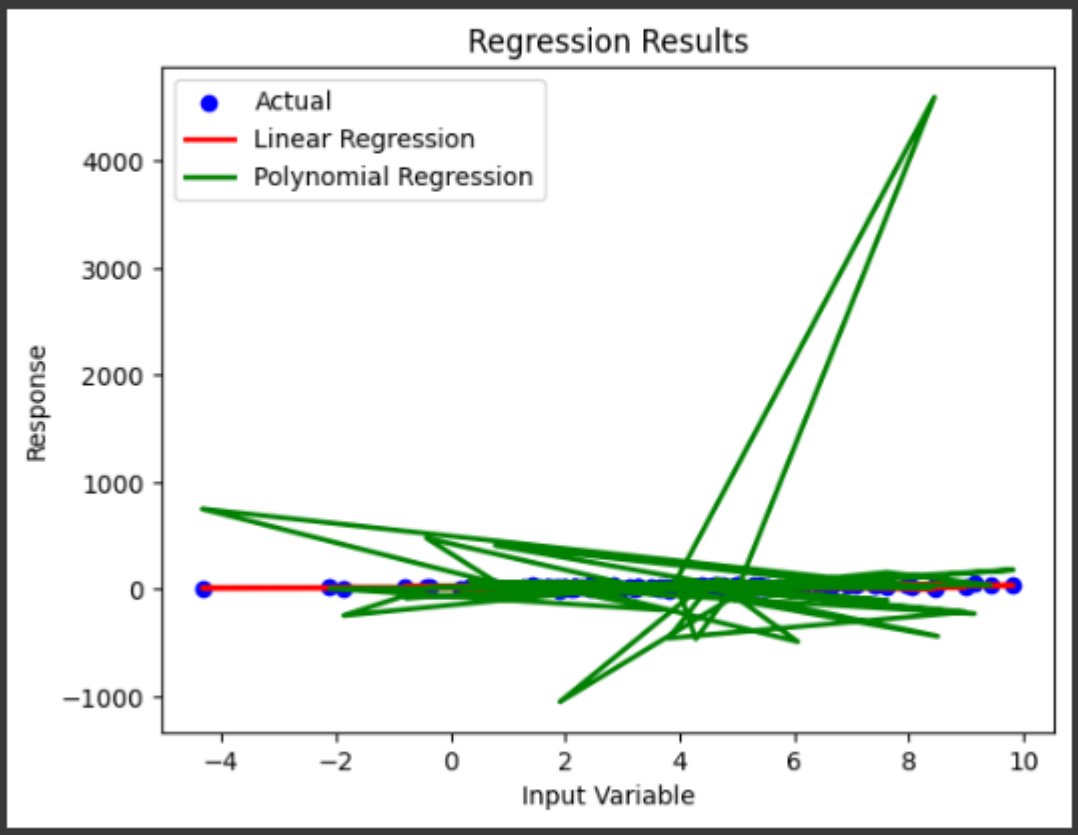
Завдання 3

Лістинг програми:

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

# Завантаження даних з файлу
data = np.loadtxt('data_multivar_regr.txt', delimiter=',')

# Розбивка даних на навчальний та тестовий набори
X, y = data[:, :-1], data[:, -1]
num_training = int(0.8 * len(X))
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# Створення та навчання моделі лінійної регресії
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогноз результату для тестового набору даних
y_test_pred = regressor.predict(X_test)

# Виведення метрик якості лінійної регресії
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))
```

```
# Створення поліноміального регресора ступеня 10 та навчання його на тре-
нувальних даних
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)

# Вибір вибіркової точки даних і створення поліноміального представлення
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)

# Створення об'єкта лінійного регресора та підгонка до полінома
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)

# Прогноз з використанням лінійного регресора та поліноміального регресора
linear_prediction = regressor.predict(datapoint)
poly_prediction = poly_linear_model.predict(poly_datapoint)

# Візуалізація результатів
plt.scatter(X_test[:, 0], y_test, color='blue', label='Actual')
plt.plot(X_test[:, 0], y_test_pred, color='red', linewidth=2, label='Linear
Regression')
plt.plot(X_test[:, 0],
poly_linear_model.predict(polynomial.transform(X_test)), color='green',
linewidth=2, label='Polynomial Regression')
plt.xlabel('Input Variable')
plt.ylabel('Response')
plt.legend()
plt.title('Regression Results')
plt.show()

# Порівняння результатів
print("\nComparison of predictions:")
print("Linear regression prediction is", linear_prediction - 41.35, "away from
the actual value.")
print("Polynomial regression prediction is", poly_prediction - 41.35, "away
from the actual value.")
```

```
Linear Regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
Median absolute error = 2.99
Explained variance score = 0.86
R2 score = 0.86
```




```
Comparison of predictions:
Linear regression prediction is [-5.29713724] away from the actual value.
Polynomial regression prediction is [0.11319764] away from the actual value.
```

Рис.3. Результат оцінки якості лінійної регресії та різниця у прогнозах.

Висновки: порівнюючи отримані прогнози, можемо визначити, що прогноз, отриманий за допомогою лінійної регресії, відхиляється на -5.30 одиниць. У той час як прогноз, отриманий за допомогою поліноміальної регресії, має відхилення лише 0.11 одиниць і, відповідно, набагато ближчий до фактичного значення. Це свідчить про вищу точність в порівнянні з лінійною регресією.

Завдання 4

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
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
# Завантаження даних
diabetes = datasets.load_diabetes()
X = diabetes.data
y = diabetes.target

# Поділіть дані на навчальний та тестовий набори
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.5,
random_state=0)

# Створення моделі лінійної регресії та навчання
regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)

# Зробіть прогноз по тестовій вибірці
ypred = regr.predict(Xtest)

# Виведіть коефіцієнти регресії та показники
print("Коефіцієнти регресії:")
print(regr.coef_)
print("Перехоплення:")
print(regr.intercept_)
print("R-squared (R2) Score:", r2_score(ytest, ypred))
print("Mean Absolute Error (MAE):", mean_absolute_error(ytest, ypred))
print("Mean Squared Error (MSE):", mean_squared_error(ytest, ypred))

# Побудова графіку
fig, ax = plt.subplots()
# Задайте стилі для точок та лінії
plt.scatter(ytest, ypred, edgecolors=(0, 0, 0), c='blue', label='Дані')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4, )

# Додайте підписи до осей та легенду
plt.xlabel('Виміряно')
plt.ylabel('Передбачено')
plt.legend(loc='best')

# Додайте заголовок
plt.title('Порівняння виміряних та передбачених даних')

# Відобразіть сітку на графіку
plt.grid(True)

# Збільште розмір шрифту на графіку
plt.rcParams.update({'font.size': 14})
```

```
# Відобразіть графік
plt.show()
```

```
Коефіцієнти регресії:
[ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Перехоплення:
154.35892852801342
R-squared (R2) Score: 0.4377497118254099
Mean Absolute Error (MAE): 44.800645233553276
Mean Squared Error (MSE): 3075.3306886803252
```

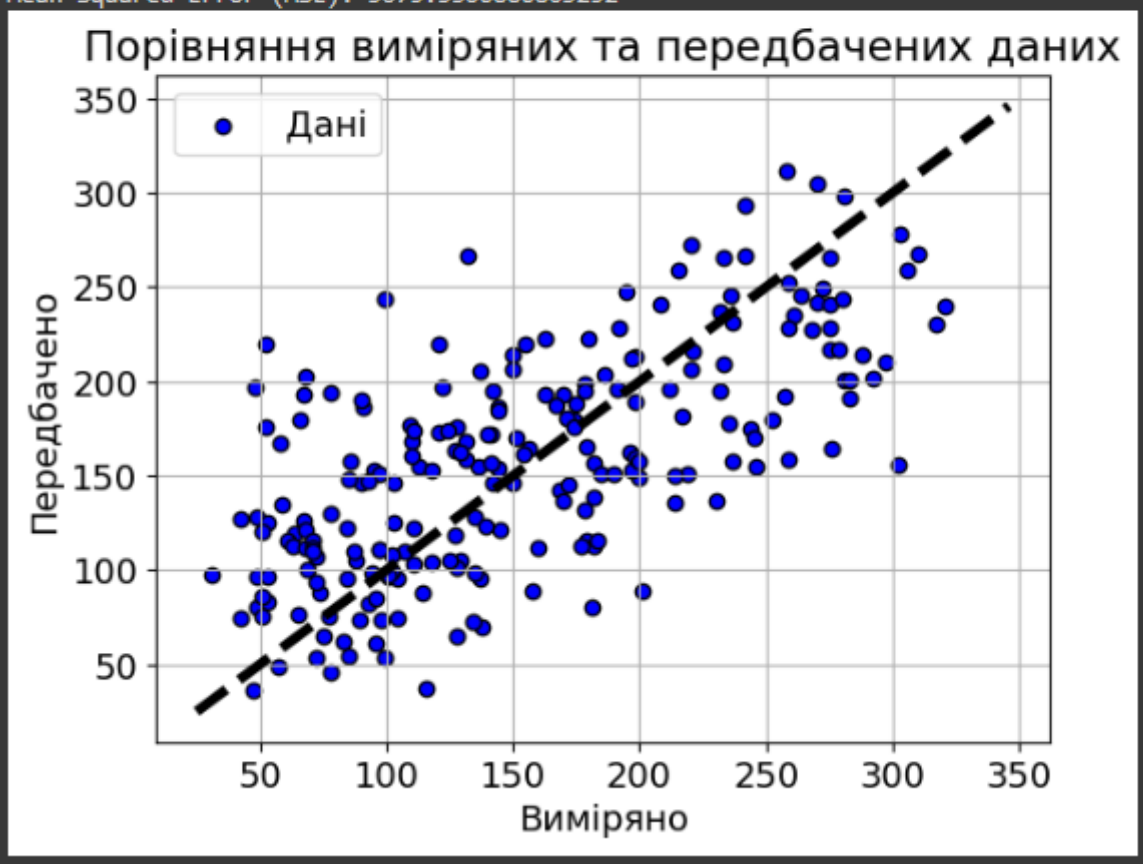


Рис.7. Результат оцінки якості лінійної регресії та графік.

Висновки: Модель має десять коефіцієнтів регресії, які вказують на вплив кожного вхідного параметра на цільову змінну. Наприклад, деякі параметри можуть мати позитивний вплив, в той час як інші впливають негативно. Перехоплення становить приблизно 154.36. Це означає, що при усіх вхідних параметрах (коефіцієнтах), рівних нулю, очікуване значення цільової змінної дорівнює 154.36. Значення R2 Score становить близько 0.44. Іншими словами, модель пояснює лише 44% варіації у цільовій змінній, що не є дуже задовільним

для передбачення даних. Середня абсолютна похибка становить приблизно 44.80, що свідчить про значну похибку у прогнозах моделі.

Отриманий графік також ілюструє розкид даних. З цього можна зробити висновок, що лінійна регресійна модель, побудована на даних

Завдання 5

Лістинг програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

# Генеруємо випадкові дані на освові варіанту
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)

# Побудова моделі лінійної регресії
lin_reg = LinearRegression()
lin_reg.fit(X, y)

# Побудова моделі поліноміальної регресії
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)

# Виведення коефіцієнтів лінійної регресії
print("Лінійна регресія:")
print("Перехоплення:", lin_reg.intercept_)
print("Коефіцієнт регресії:", lin_reg.coef_)

# Виведення коефіцієнтів поліноміальної регресії
print("Поліноміальна регресія:")
print("Перехоплення:", poly_reg.intercept_)
print("Коефіцієнти регресії:", poly_reg.coef_)

# Виведення даних на графіку
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.scatter(X, y, c='b', label='Дані', s=20)  # Змінено розмір точок
plt.plot(X, lin_reg.predict(X), 'r-', linewidth=2, label='Лінійна регресія')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='upper left')
plt.title('Лінійна регресія')
```

```
plt.subplot(1, 2, 2)
X_new = np.linspace(-5, 1, 100).reshape(100, 1)
X_new_poly = poly_features.transform(X_new)
y_new = poly_reg.predict(X_new_poly)
plt.scatter(X, y, c='b', label='Дані', s=20)  # Змінено розмір точок
plt.plot(X_new, y_new, 'r-', linewidth=2, label='Поліноміальна регресія')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc='upper left')
plt.title('Поліноміальна регресія')

plt.tight_layout()  # Забезпечує належне розташування підписів на графіку
plt.show()
```

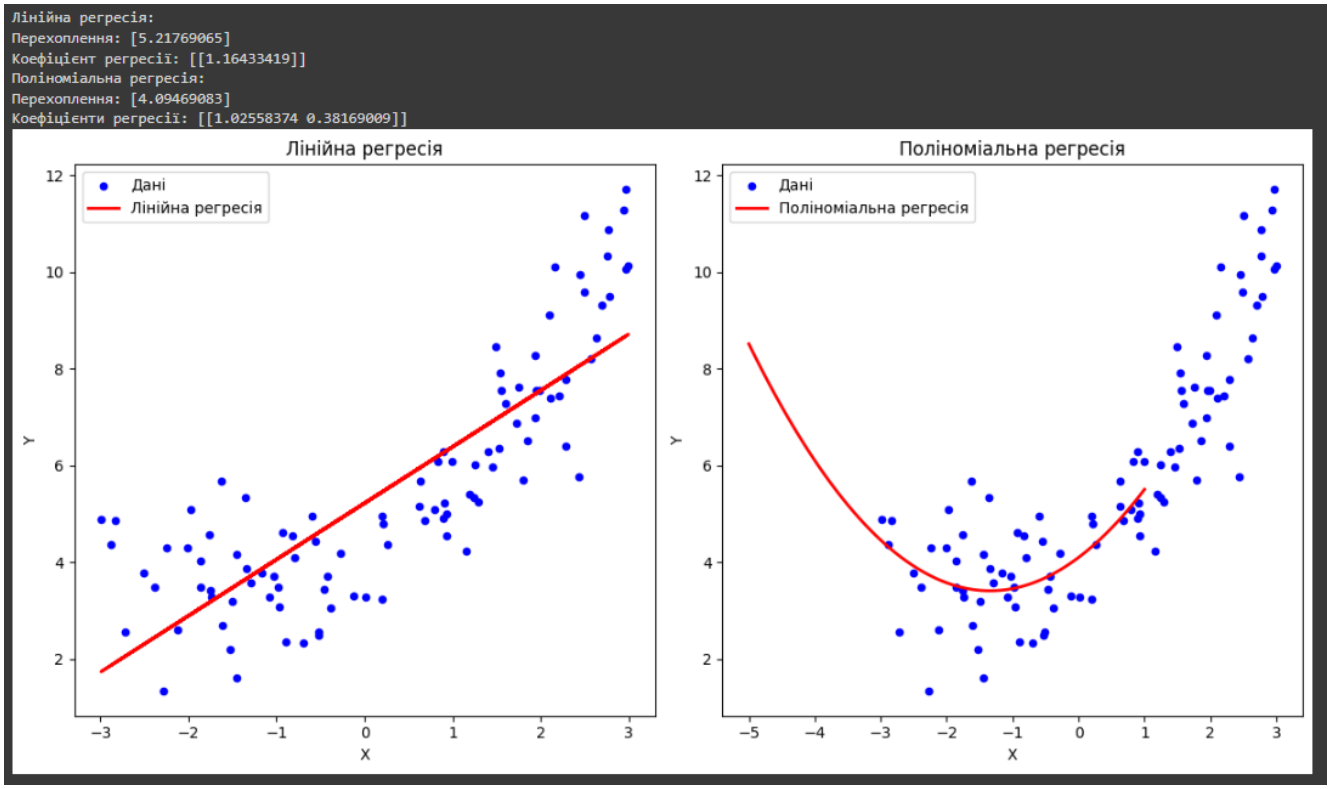


Рис.9. Графіки та перехоплення та коефіцієнти регресії.

Модель 5 варіанта у вигляді математичного рівняння:

$y=0.4x^2+1x+4+гауссовий\ шум.$

Отримана модель регресії з передбаченими коефіцієнтами:

$y=1x^2+0.38x+4$

Висновки: Для лінійної моделі, коефіцієнт перед x (близько 1.16) дуже близький до 1. Це свідчить про те, що лінійна модель практично не враховує основну залежність між змінною x та змінною y. З іншого боку, поліноміальна модель має коефіцієнти перед x^2 (близько 1.03) та x (близько 0.38), які вказують на наявність квадратичного зв'язку між змінною x і змінною y. Це означає, що поліноміальна модель краще враховує і описує вихідні дані.

Отже, на підставі цих аналізів, поліноміальна регресія більше підходить для апроксимації цього набору даних, оскільки вона краще відображає складну залежність між змінними, в той час як лінійна модель практично не враховує цю залежність.

Завдання 6

Лістинг програми:

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

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.xlabel("Training set size")
    plt.ylabel("RMSE")
    plt.legend()
    plt.show()
```

```
# Генеруємо випадкові дані, як у варіанті
m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.4 * X ** 2 + X + 4 + np.random.randn(m, 1)

# Створення та побудова кривих навчання для лінійної регресії
lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)

# Створення та побудова кривих навчання для поліноміальної регресії 10-го
ступеня
polynomial_regression = Pipeline([
    ("poly features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression())
])

plot_learning_curves(polynomial_regression, X, y)
```

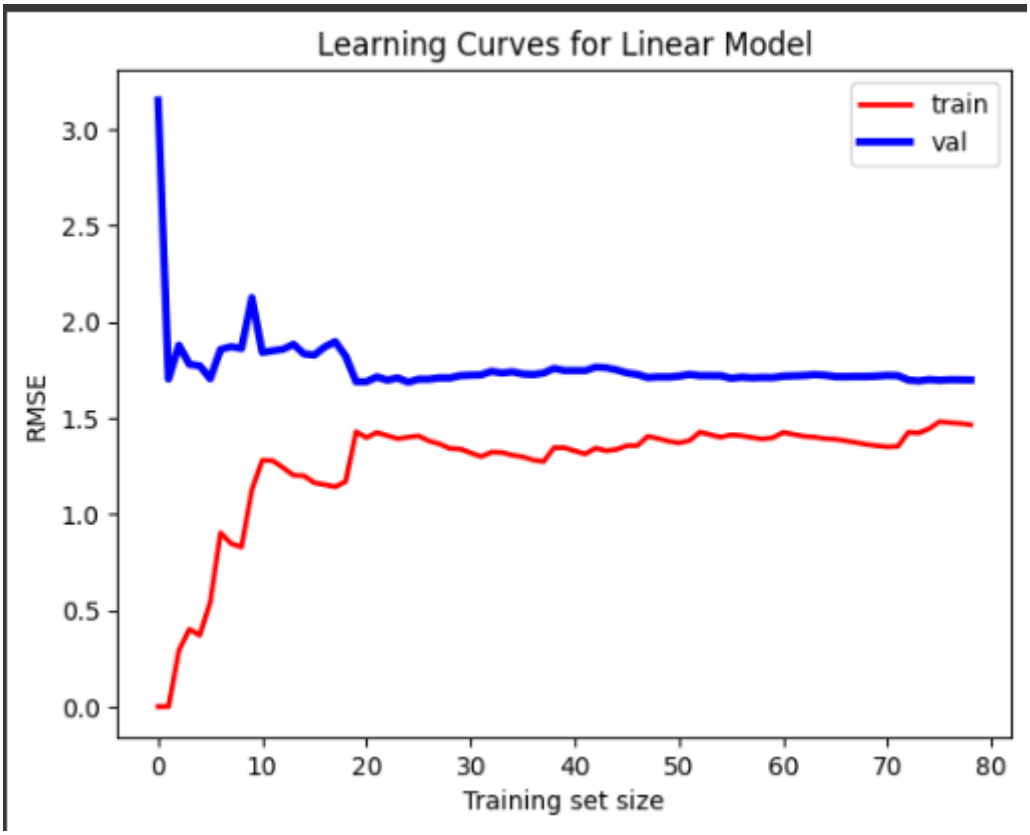


Рис.11.Криві навчання для лінійної моделі.

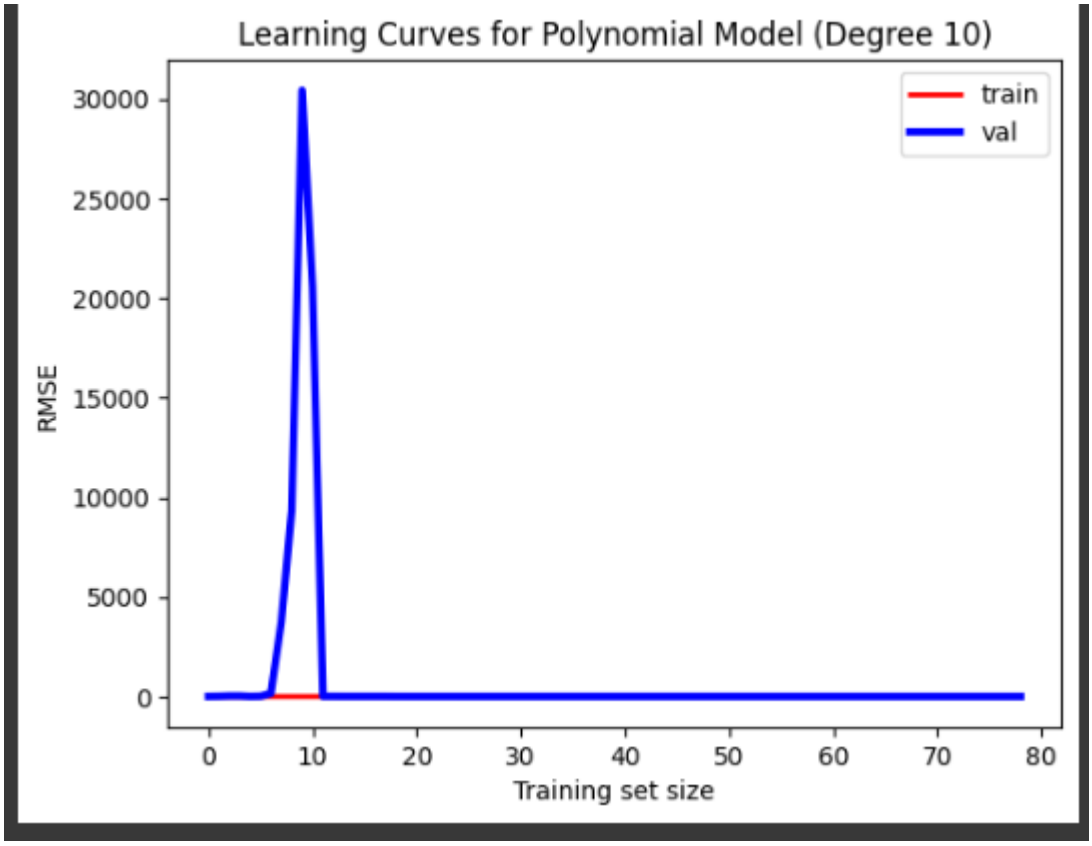


Рис.12. Криві навчання для поліноміальної моделі.

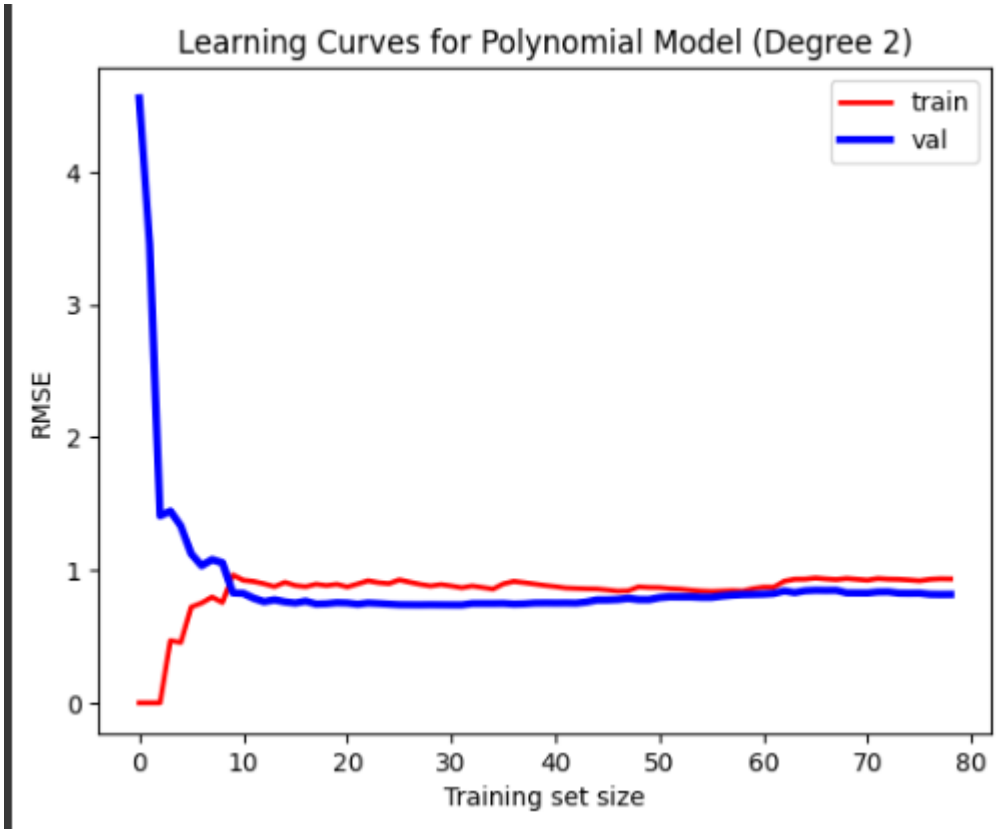


Рис.13. Криві навчання поліноміальної моделі 2-го ступеня.

Завдання 7

Лістинг програми:

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

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

# Кількість кластерів
num_clusters = 5

# Визуалізація вхідних даних
plt.figure(figsize=(8, 6))
plt.scatter(X[:, 0], X[:, 1], marker='o', edgecolors='black', s=80, c='blue',
label='Дані точки')
plt.title('Розподіл даних')
plt.xlabel('Ось X')
plt.ylabel('Ось Y')

# Створення та навчання моделі k-середніх
kmeans = KMeans(n_clusters=num_clusters, init='k-means++', n_init=10,
random_state=0)
kmeans.fit(X)

# Побудова центроїд кластерів
centroids = kmeans.cluster_centers_
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, marker='x', c='red',
label='Центроїди')

# Вивід результату на графіку
plt.legend()

# Визначення кроку сітки
step_size = 0.01

# Визначення меж 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()])

# Відображення областей та виділення їх кольором
output = output.reshape(x_vals.shape)
```

```
plt.figure(figsize=(8, 6))
plt.imshow(output, interpolation='nearest', extent=(x_vals.min(),
x_vals.max(), y_vals.min(), y_vals.max()), cmap=plt.cm.Paired, origin="lower")

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], marker='o', edgecolors="black", s=80,
c=kmeans.labels_, cmap=plt.cm.Paired)

# 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', label='Центроїди')

plt.title('Кластеризація даних')
plt.xlabel('Ось X')
plt.ylabel('Ось Y')
plt.legend()
plt.show()

# Виведення інформації про кластеризацію
labels = kmeans.labels_
silhouette_score = metrics.silhouette_score(X, labels, metric='euclidean')
print(f"Силуетний коефіцієнт: {silhouette_score}")
```

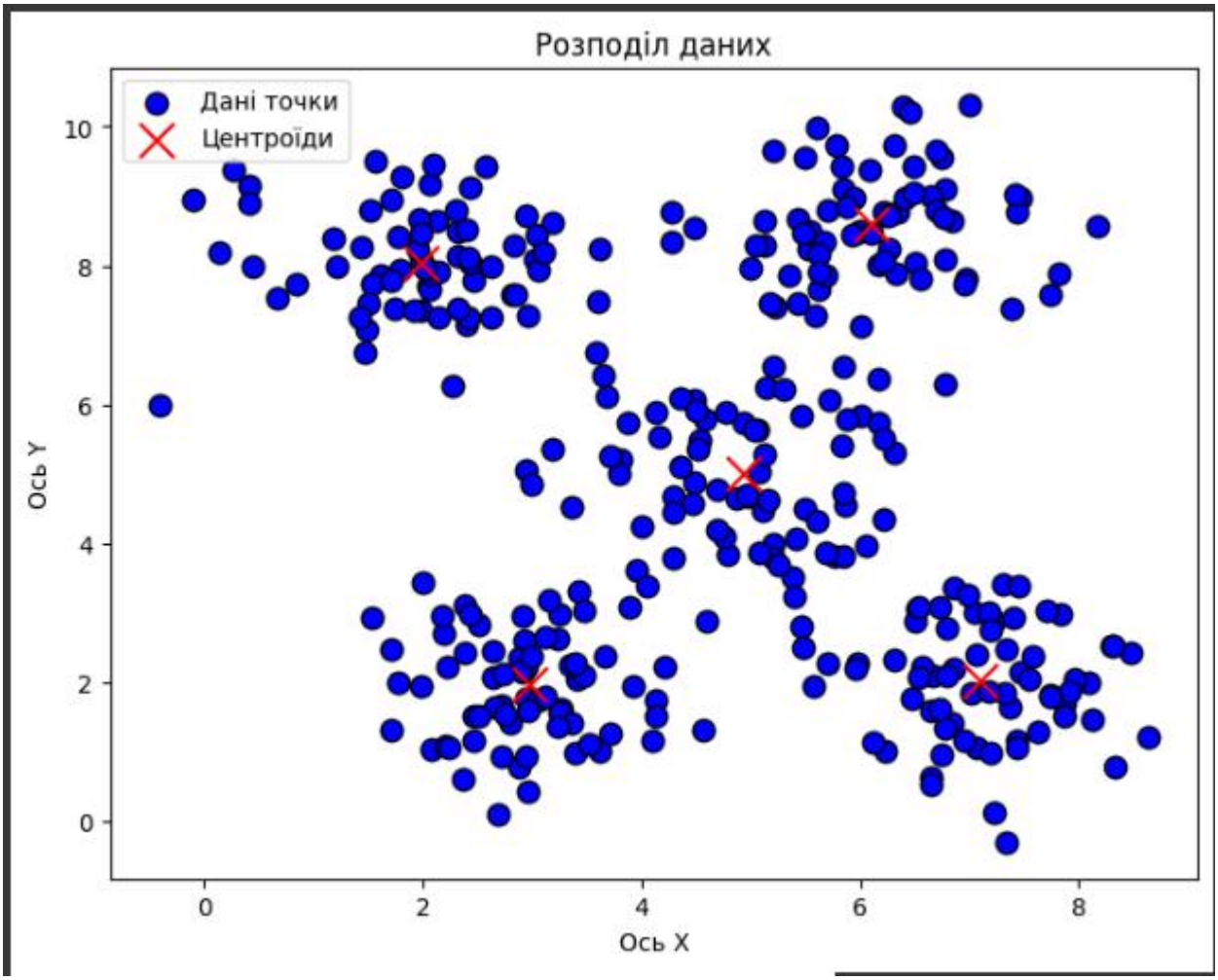
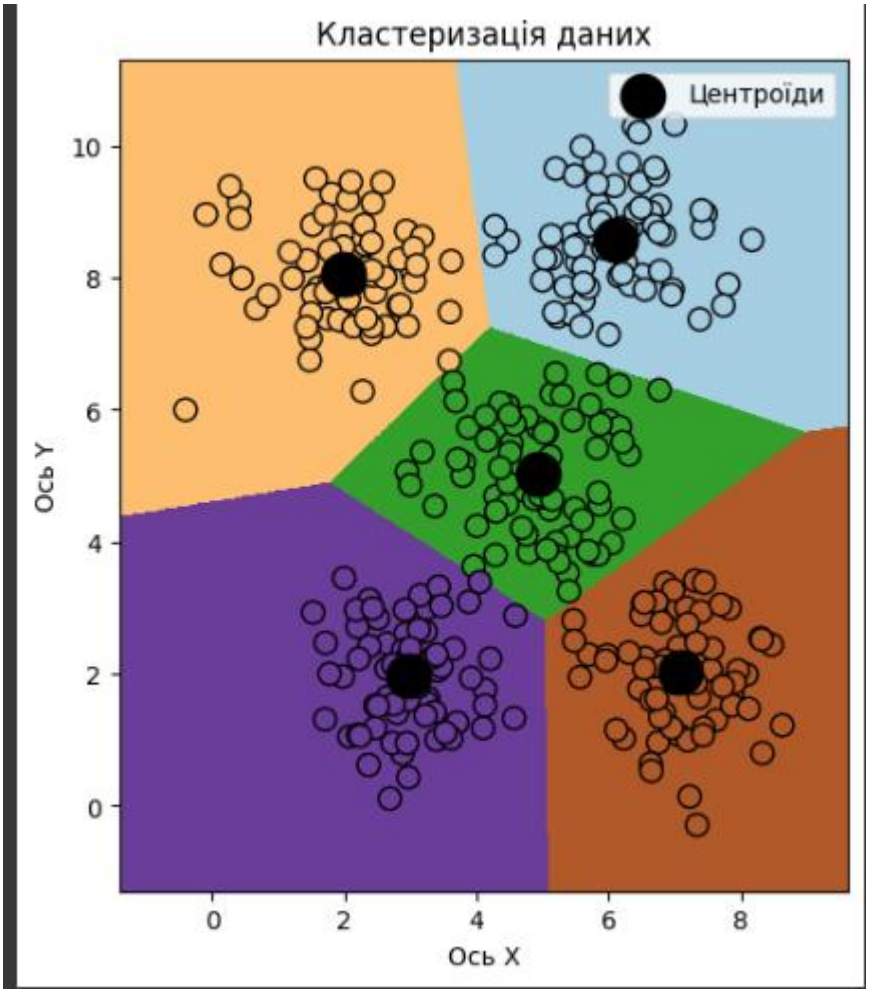


Рис.14. Графік результату і силуетний коефіцієнт.




```
Силуетний коефіцієнт: 0.5906519837897815
```

Рис.15. Графік з результатом кластеризації даних.

Висновки: Силуетний коефіцієнт, приблизно 0.59, свідчить про високу якість кластеризації, використовуючи метод k-середніх. Об'єкти в межах кожного кластера розташовані близько один до одного, і більшість з них віддалені від об'єктів інших кластерів (хоча не всі). Графік показує наявність 5 кластерів, і алгоритм k-середніх успішно визначив ці кластери та їхні центроїди.

Завдання 8

Лістинг програми:

```
# Імпортуємо необхідні бібліотеки
from sklearn.cluster import KMeans
import numpy as np
from sklearn.metrics import pairwise_distances_argmin
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris

# Завантажуємо датасет iris
```

```
    iris = load_iris()
    X = iris.data  # Ознаки
    y = iris.target  # Мітки класів

    # Ініціалізуємо модель k-середніх з 5 кластерами
    kmeans = KMeans(n_clusters=5)
    kmeans.fit(X)  # Навчання моделі

    # Передбачаємо кластери для вхідних даних
    y_kmeans = kmeans.predict(X)

    # Візуалізація результатів
    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()

    # Функція для знаходження кластерів, використовуючи pair-
wise_distances_argmin
    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

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

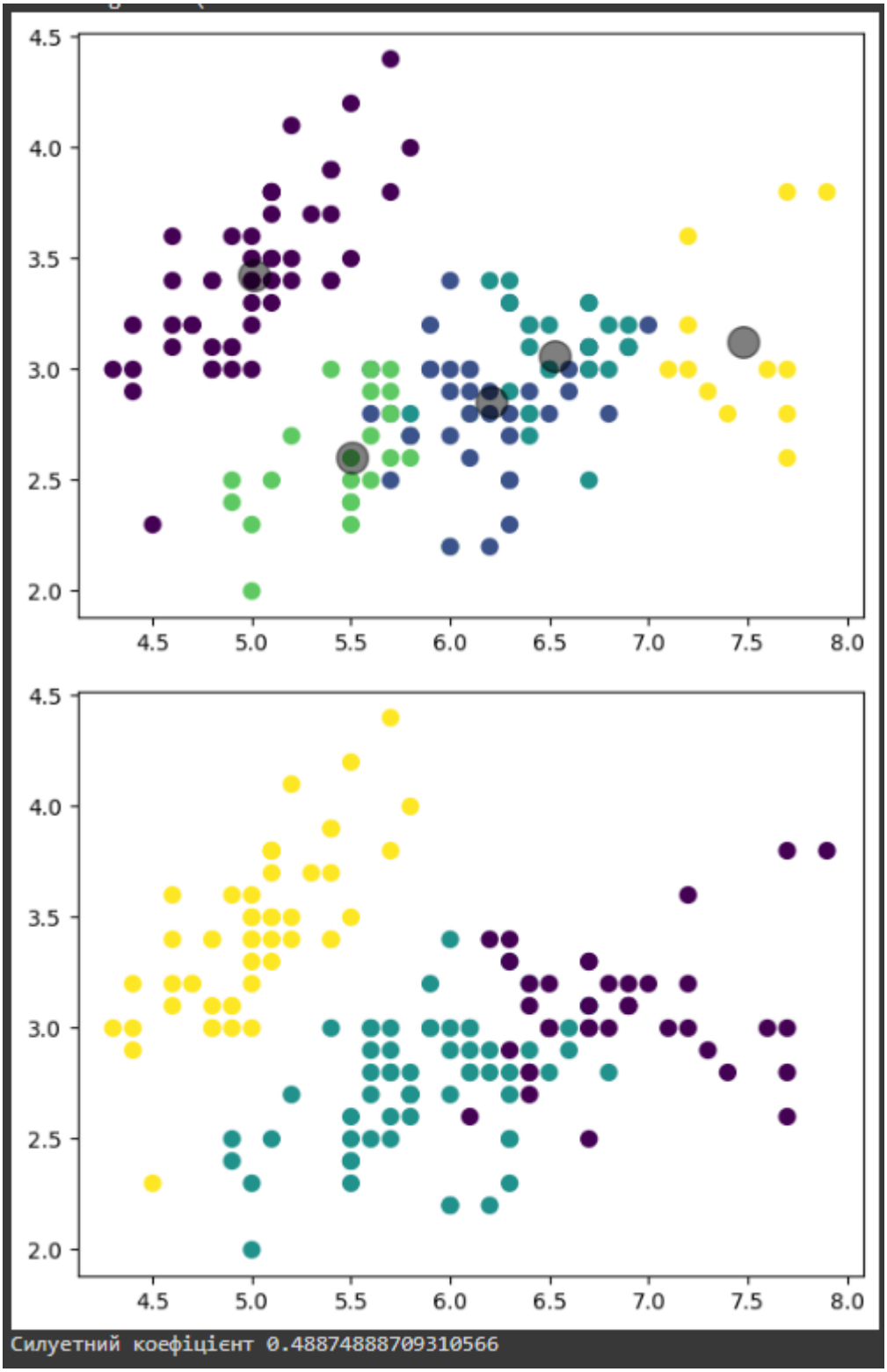


Рис.16. Результат кластеризації.

Висновки: Перший графік відображає кластеризацію набору даних iris за допомогою методу k-середніх на 5 кластерів. Кожен кластер має власну окрему

кольорову мітку, і центри кластерів позначені чорним кольором. На другому графіку використовується функція find_clusters для розділення даних на 3 кластери. Іншими словами, структура даних залишилася незмінною, але 4 менших кластера були об'єднані в два більших кластера. Силуетний коефіцієнт становить приблизно 0.488, що вказує на те, що якість кластеризації за допомогою методу k-середніх є помірною.

Завдання 9

Лістинг програми:

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

# Завантаження вхідних 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):
    plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color="black")

    # Відображення на графіку центру кластера
    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()
```

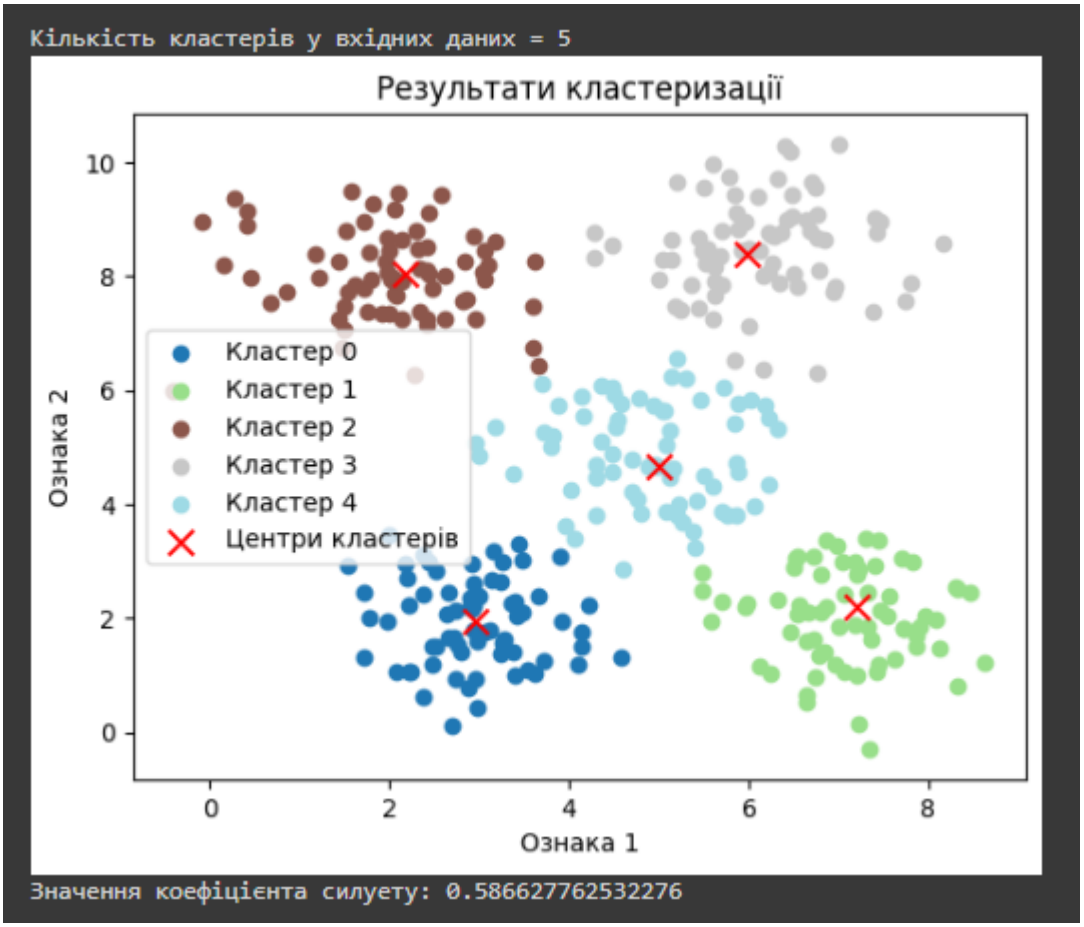


Рис.17. Результат кластеризації.

Висновки: Порівнавши результати кластеризації методом MeanShift із методом k-середніх у виконанні 7 завдання, видно, що вони дуже схожі візуально. Обидва методи створили по 5 кластерів з чітко визначеними центрами. Значення силуетного коефіцієнта становить 0.587, що свідчить про високу ефективність кластеризації за допомогою алгоритму Mean Shift.