Лабораторна робота 4

Лінійна регресія. Метод найменших квадратів. Інтерполяція

Мета роботи: Опрацювати поняття «лінійна регресія», дослідити метод найменших квадратів та набути навички роботи в середовищі Python.

Хід роботи

Лінійна регресія (Завдання 2)

Постановка задачі: Для експериментально отриманих значень X та Y знайти параметри функції $y = \beta_0 + \beta_1 x$ методом найменших квадратів (МНК) та побудувати графік апроксимуючої функції.

Мій варіант (Варіант 8):

8	X	6	7	8	9	10	12
	Y	2	3	3	4	6	5

1. Ручний розрахунок за методом найменших квадратів (МНК)

Необхідно знайти коефіцієнти β_1 (нахил) та β_0 (зсув).

Формули МНК:

$$\beta_1 = \frac{n\sum(x_i y_i) - \sum x_i \sum y_i}{n\sum x_i^2 - (\sum x_i)^2}$$

$$\beta_0 = \frac{\sum y_i - \beta_1 \sum x_i}{n}$$

Проміжні суми (n=6):

1. $\sum x_i = 6 + 7 + 8 + 9 + 10 + 12 = 52$
2. $\sum y_i = 2 + 3 + 3 + 4 + 6 + 5 = 23$
3. $\sum x_i^2 = 36 + 49 + 64 + 81 + 100 + 144 = 474$
4. $\sum x_i y_i = (6 \cdot 2) + (7 \cdot 3) + (8 \cdot 3) + (9 \cdot 4) + (10 \cdot 6) + (12 \cdot 5) = 12 + 21 + 24 + 36 + 60 + 60 = 213$

Розрахунок коефіцієнтів:

$$\beta_1 = \frac{6 \cdot 213 - 52 \cdot 23}{6 \cdot 474 - 52^2} = \frac{1278 - 1196}{2844 - 2704} = \frac{82}{140} \approx 0.5857$$

$$\beta_0 = \frac{23 - 0.5857 \cdot 52}{6} = \frac{23 - 30.4564}{6} = \frac{-7.4564}{6} \approx -1.2427$$

Отримане рівняння регресії: y = -1.24 + 0.59x

2. Побудова моделі у середовищі Python

Код для автоматичного розрахунку та візуалізації:

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

X_data = np.array([[6], [7], [8], [9], [10], [12]])
Y_data = np.array([2, 3, 3, 4, 6, 5])

# Створення та навчання моделі
model = LinearRegression()
model.fit(X_data, Y_data)

# Отримання коефіцієнтів
beta_0 = model.intercept_
beta_1 = model.coef_[0]

print(f"Коефіцієнт нахилу (beta_1): {beta_1}")
print(f"Вільний член (beta_0): {beta_0}")

X_fit = np.linspace(X_data.min(), X_data.max(), 100).reshape(-1, 1)
Y_pred = model.predict(X_fit)

plt.figure(figsize=(8, 6))
plt.scatter(X_data, Y_data, color='red', s=100, label='Експериментальні точки
(Вар. 8)')
plt.plot(X_fit, Y_pred, color='blue', linewidth=2, label=f'Апроксимація: y =
{beta_0:.2f} + {beta_1:.2f}x')
plt.title('Лінійна регресія для Варіанту 8')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```

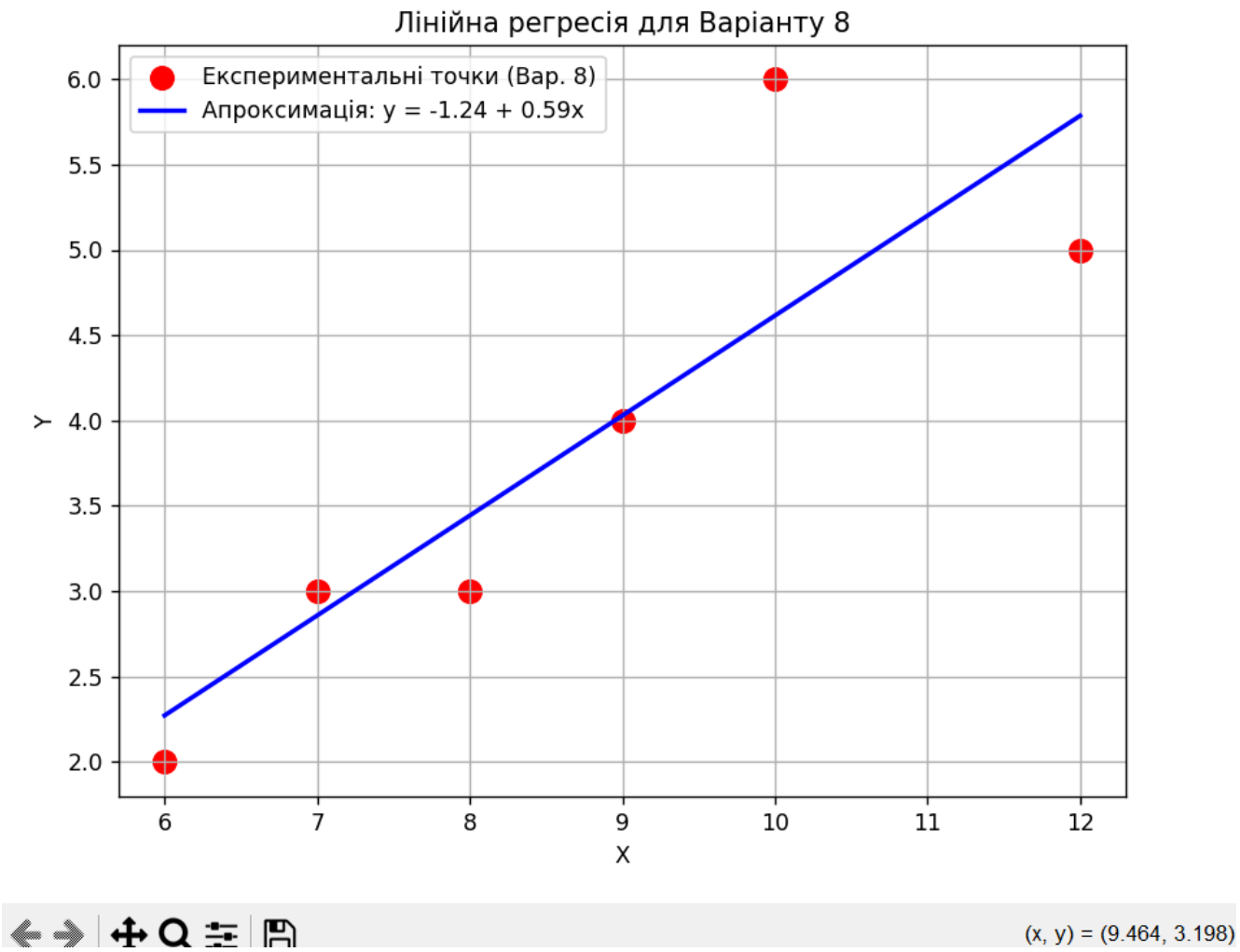


Рис. 1. Графік лінійної регресії

Для даних було знайдено лінійну залежність y = 0.59x - 1.24. Коефіцієнт 0.59 показує, що зі збільшенням X на одиницю, Y зростає приблизно на 0.6. Лінія проходить близько до експериментальних точок, що свідчить про адекватність моделі.

Інтерполяція (Завдання 3)

Постановка задачі: Виконати інтерполяцію функції, заданої таблично (5 точок), поліномом 4-го ступеня. Знайти значення функції у проміжних точках x=0.2 та x=0.5.

Вхідні дані:

X = [0.1, 0.3, 0.4, 0.6, 0.7]

Y = [3.2, 3.0, 1.0, 1.8, 1.9]

Алгоритм:

1. Сформувати систему лінійних рівнянь на основі матриці Вандермонда.
2. Знайти коефіцієнти полінома $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$.
3. Обчислити значення в точках 0.2 та 0.5.

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([0.1, 0.3, 0.4, 0.6, 0.7])
Y = np.array([3.2, 3.0, 1.0, 1.8, 1.9])

# 1. Знаходження коефіцієнтів через матрицю Вандермонда
# Система рівнянь X_mat * A = Y
X_mat = np.vander(X, increasing=True)
A = np.linalg.solve(X_mat, Y)

# Створення функції полінома
P = np.poly1d(A[::-1]) # A[::-1] бо poly1d очікує від старшого степеня до
молодшого

print("Коефіцієнти полінома:", A)
print(f"Значення P(0.2) = {P(0.2):.4f}")
print(f"Значення P(0.5) = {P(0.5):.4f}")

X_new = np.linspace(0.1, 0.7, 100)
Y_new = P(X_new)

plt.figure(figsize=(8, 6))
plt.plot(X, Y, 'ro', label='Задані точки')
plt.plot(X_new, Y_new, 'b-', label='Інтерполяційний поліном')
plt.plot(0.2, P(0.2), 'go', label=f'P(0.2)={P(0.2):.2f}')
plt.plot(0.5, P(0.5), 'go', label=f'P(0.5)={P(0.5):.2f}')

plt.title('Інтерполяція поліномом 4-го ступеня')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```

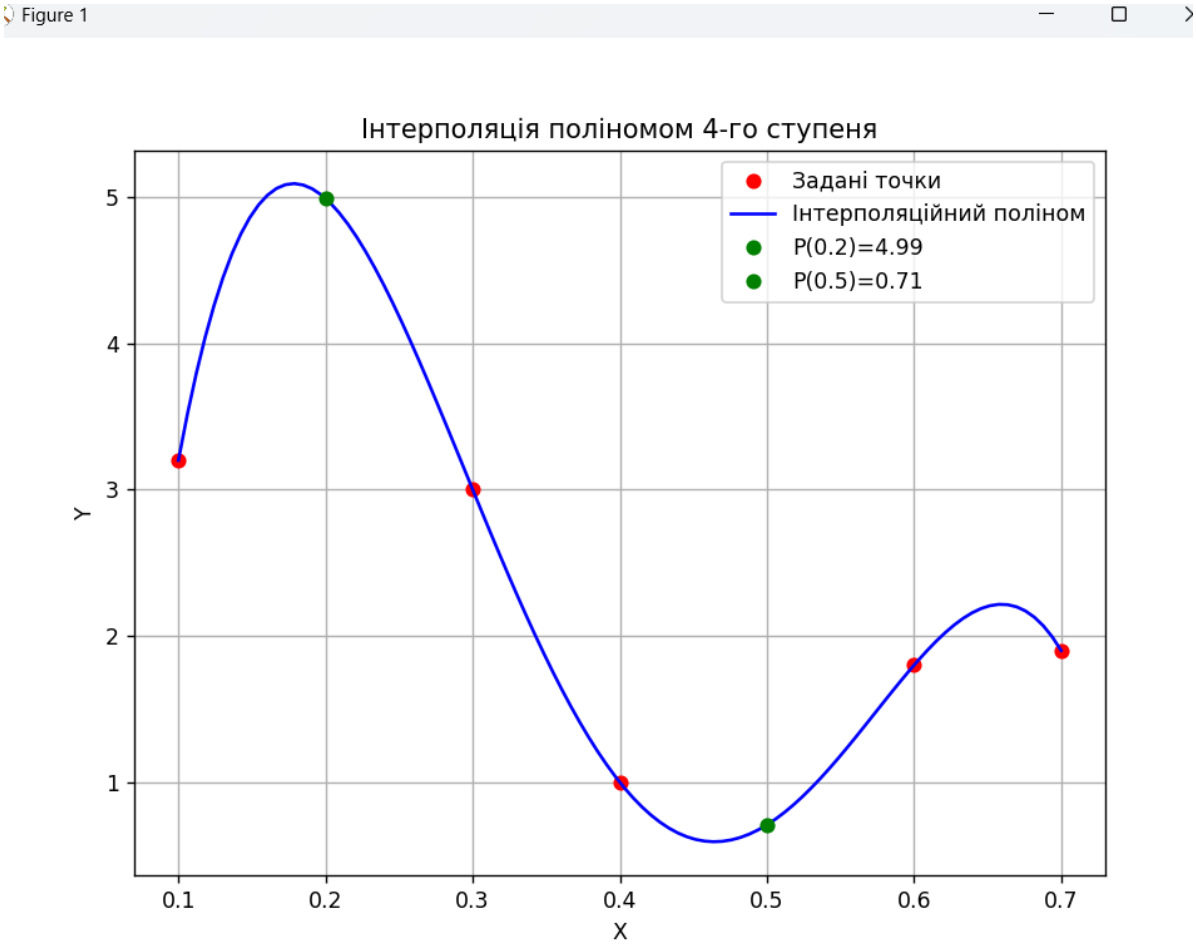


Рис. 2. Графік інтерполяційного полінома

Результати розрахунків:

Програма вивела наступні значення у шуканих точках:

При x = 0.2, y ≈4.9889

При x = 0.5, y ≈ 0.7089

За допомогою інтерполяційного полінома Лагранжа (через розв'язок системи Вандермонда) вдалося побудувати функцію, яка точно проходить через усі задані вузли. Це дозволило відновити значення функції у проміжних точках, де вимірювання не проводилися

Лабораторна роботи №4

на тему: "Дослідження методів регресії"

Мета роботи: використовуючи спеціалізовані бібліотеки та мову програмування Python, дослідити методи регресії даних у машинному навчанні.

Хід роботи

Завдання 2.1. Створення регресора однієї змінної

Побудувати регресійну модель на основі однієї змінної, використовуючи файл вхідних даних data_singlevar_regr.txt.

Виконання:

Створив файл LR_3_task_1.py.

Завантажив дані з файлу, розділив їх на навчальну (80%) та тестову (20%) вибірки.

Створив та навчив модель лінійної регресії (LinearRegression).

Виконав прогнозування та побудував графік.

Оцінив якість моделі за метриками (MAE, MSE, Median AE, R2).

Зберіг модель у файл model.pkl та перевірив її повторне завантаження.

код програми LR_3_task_1.py:

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

input_file = 'data_singlevar_regr.txt'

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

    # Розбивка даних на навчальний та тестовий набори (80% на 20%)
    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 об'єкта лінійного регресора
    regressor = linear_model.LinearRegression()
    regressor.fit(X_train, y_train)

    # Прогнозування результату
    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.title('Linear Regression (Single Variable)')
    plt.show()

    # Обчислення метричних параметрів
    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))

    output_model_file = 'model.pkl'
    with open(output_model_file, 'wb') as f:
        pickle.dump(regressor, f)

except OSError:
    print(f"Помилка: Файл {input_file} не знайдено.")
```

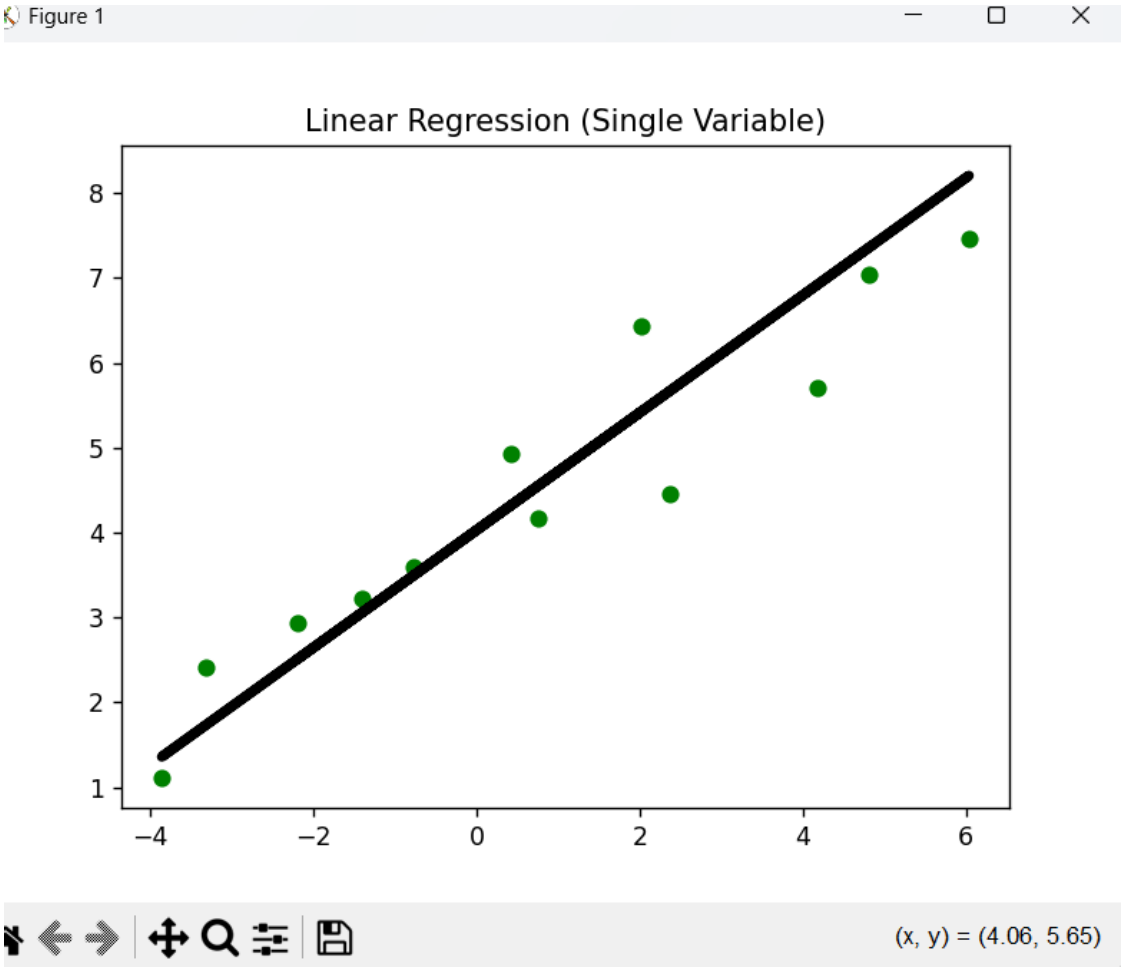


Рис. 1. Графік точки даних та лінія регресії

Вивід з консолі з метриками:

```
"E:\University\4_course\Artificial intelligenc
Linear regressor performance:
Mean absolute error = 0.59
Mean squared error = 0.49
Median absolute error = 0.51
Explained variance score = 0.86
R2 score = 0.86
```

Завдання 2.2. Передбачення за допомогою регресії однієї змінної

Виконання:

Створив файл LR_3_task_2.py.

Використав дані з файлу мого варіанту.

Побудував модель аналогічно до попереднього завдання.

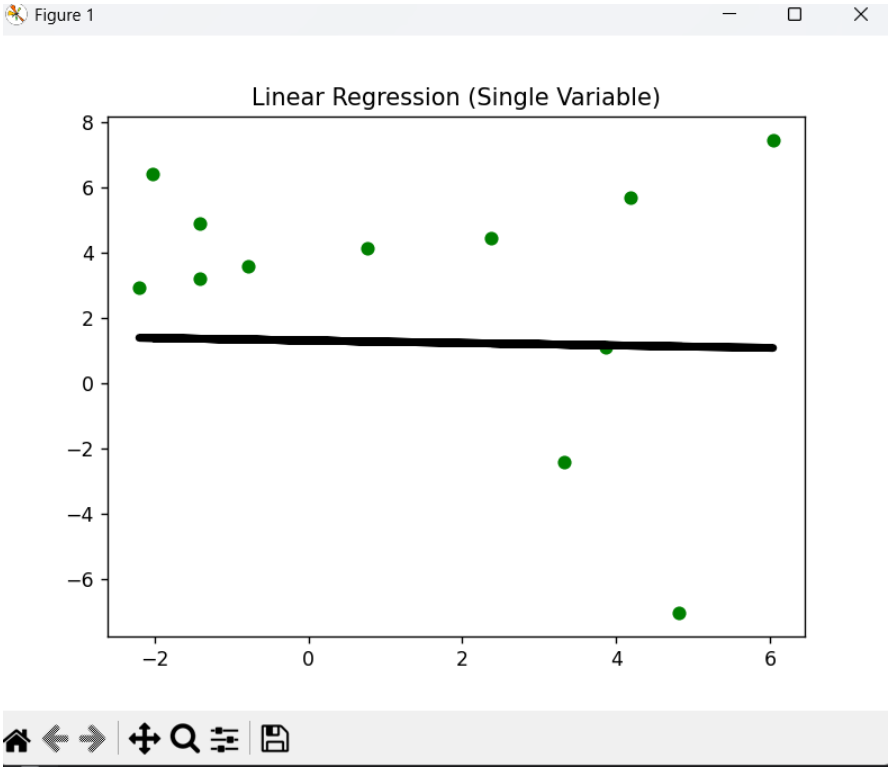


Рис. 2. Побудований регресійної моделі

Завдання 2.3. Створення багатовимірного регресора

Побудувати регресійну модель на основі багатьох змінних, використовуючи файл data_multivar_regr.txt. Порівняти лінійну та поліноміальну регресії.

Виконання:

Створив файл LR_3_task_3.py.

Побудував лінійний регресор для багатовимірних даних та оцінив його.

Створив поліноміальний регресор (ступінь 10) за допомогою PolynomialFeatures.

код програми LR_3_task_3.py:

```
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))
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# 1. Лінійна регресія
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

# 2. Поліноміальна регресія (ступінь 10)
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)

# Тестова точка з завдання
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.fit_transform(datapoint)

print("\nLinear regression prediction:", linear_regressor.predict(datapoint))
print("Polynomial regression prediction:",
poly_linear_model.predict(poly_datapoint))

# Метрики для лінійної моделі
y_test_pred = linear_regressor.predict(X_test)
print("\nLinear Regressor performance:")
print("R2 score =", round(sm.r2_score(y_test, y_test_pred), 2))
```

Вивід консолі: метрики лінійної регресії та порівняння прогнозів Linear vs Polynomial

```
E:\University\4_course\Artificial Intelligence Systems

Linear regression prediction: [36.05286276]
Polynomial regression prediction: [41.08282745]

Linear Regressor performance:
R2 score = 0.86
```

Завдання 2.4. Регресія багатьох змінних (Dataset Diabetes)

Розробити лінійний регресор, використовуючи вбудований набір даних про діабет (sklearn.datasets.load_diabetes).

Виконання:

Створив файл LR_3_task_4.py.

Завантажив датасет, розділив на навчальну та тестову вибірки (50/50).

Навчив модель, розрахував метрики (коефіцієнти, R2, MSE, MAE).

Побудував графік залежності виміряних та передбачених значень.

код вашої програми LR_3_task_4.py

```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, 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)

# Навчання моделі
regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)

# Прогноз
ypred = regr.predict(Xtest)

# Вивід параметрів
print("Coefficients:", regr.coef_)
print("Intercept:", regr.intercept_)
print("R2 score =", r2_score(ytest, ypred))
print("Mean Absolute Error =", mean_absolute_error(ytest, ypred))
print("Mean Squared Error =", mean_squared_error(ytest, ypred))

# Побудова графіка
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()
```

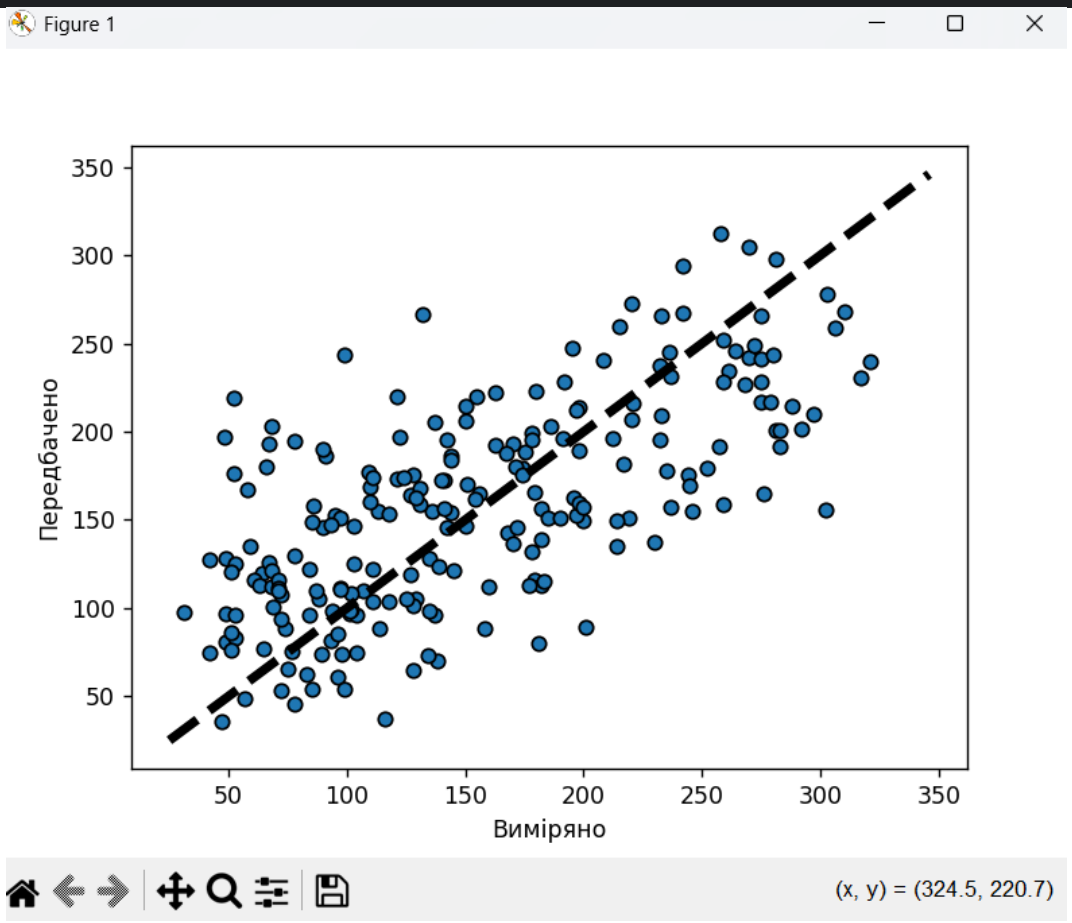


Рис. 3. Побудований графік "Виміряно vs Передбачено"

Вивід консолі з коефіцієнтами та помилками:

```
"E:\University\4_course\Artificial intelligence systems\lab4\.venv\Scripts\python.exe" "E:
Coefficients: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Intercept: 154.3589285280134
R2 score = 0.43774971182541
Mean Absolute Error = 44.800645233553276
Mean Squared Error = 3075.330688680324
```

Завдання 2.5. Самостійна побудова регресії

Згенерувати випадкові дані згідно з варіантом, побудувати лінійну та поліноміальну (ступінь 2) моделі.

Виконання:

Створив файл LR_3_task_5.py.

Побудував графік накладання лінії прогнозу на точки даних.

код програми LR_3_task_5.py:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = 2 * np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Поліноміальні ознаки (ступінь 2 - спробуйте також 3 або 5 для синусоїди)
poly_features = PolynomialFeatures(degree=3, include_bias=False)
X_poly = poly_features.fit_transform(X)

# Навчання лінійної регресії на поліноміальних даних
lin_reg = LinearRegression()
lin_reg.fit(X_poly, y)

# Вивід коефіцієнтів
print("Intercept (вільний член):", lin_reg.intercept_)
print("Coefficients (коефіцієнти при ступенях):", lin_reg.coef_)

X_new = np.linspace(-3, 3, 100).reshape(100, 1)
X_new_poly = poly_features.transform(X_new)
y_new = lin_reg.predict(X_new_poly)

plt.plot(X, y, "b.", label="Дані (Варіант 8)")
plt.plot(X_new, y_new, "r-", linewidth=2, label="Прогноз")
plt.xlabel("$x_1$", fontsize=18)
plt.ylabel("$y$", rotation=0, fontsize=18)
plt.legend(loc="upper left", fontsize=14)
plt.title("Поліноміальна регресія для y = 2*sin(X)")
plt.show()
```

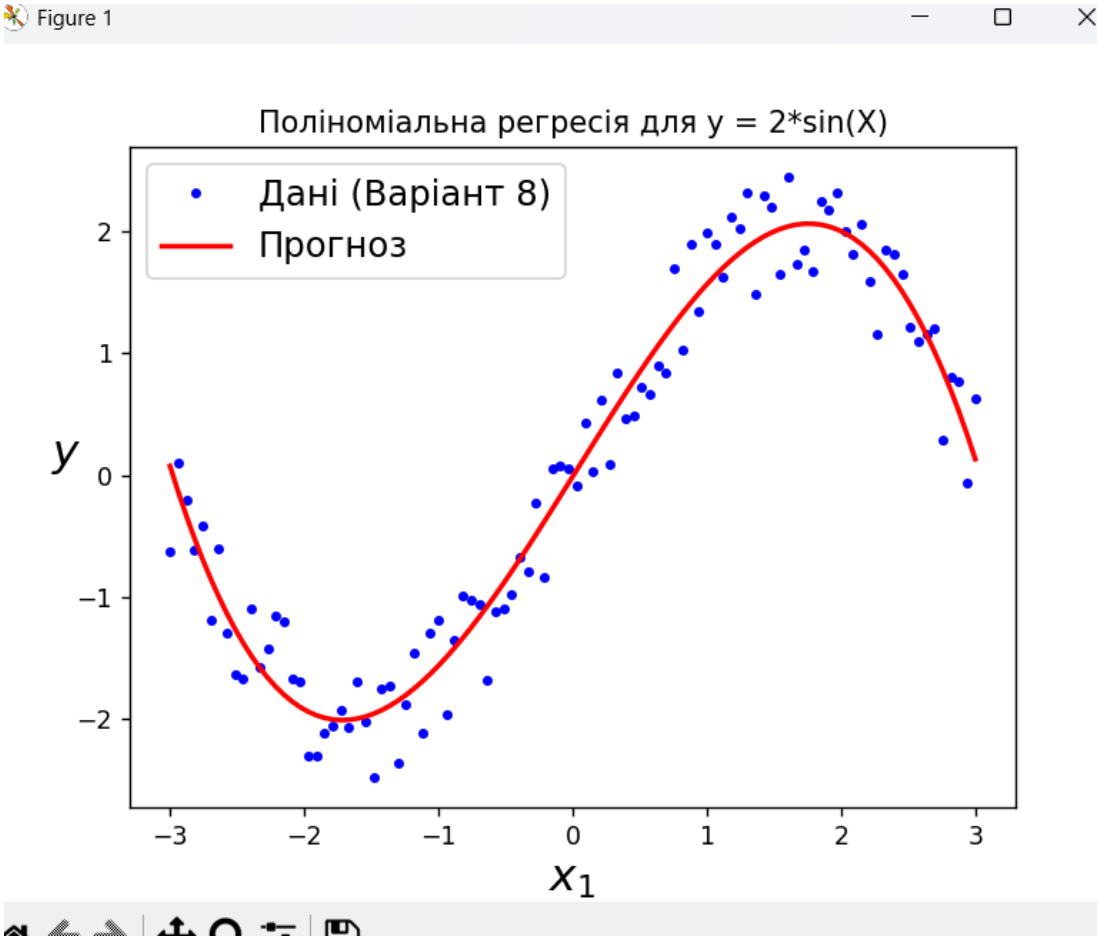


Рис. 4. Побудований графік з точками та кривою прогнозу

Висновок: Поліноміальна регресія успішно відновила форму залежності, знайшовши коефіцієнти, які наближено відповідають розкладанню функції синуса в ряд. Відхилення пояснюються наявністю шуму у вхідних даних.

Завдання 2.6. Побудова кривих навчання

Побудувати криві навчання (Learning Curves) для лінійної моделі та поліноміальних моделей різного ступеня (2 та 10) на даних з попереднього завдання.

Виконання:

Створив файл LR_3_task_6.py.

Реалізував функцію plot_learning_curves, яка будує графік RMSE від розміру навчального набору.

Побудував графіки для: Лінійної регресії.

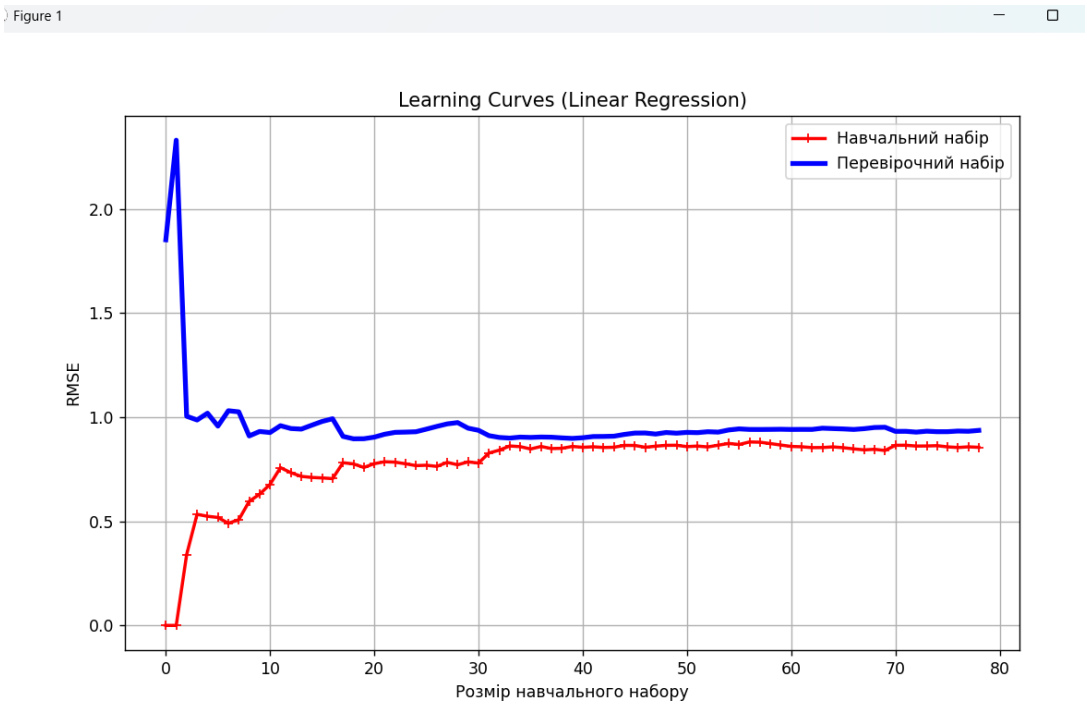
Поліноміальної регресії (ступінь 10).

Поліноміальної регресії (ступінь 2).

код програми LR_3_task_6.py:

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

m = 100
X = np.linspace(-3, 3, m)
noise = np.random.uniform(-0.5, 0.5, m)

y = 2 * np.sin(X) + noise

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,
random_state=10)
    train_errors, val_errors = [], []

    # Тренуємо модель на зростаючому наборі даних
    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="Навчальний набір")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Перевірочний набір")
    plt.ylabel("RMSE")
    plt.xlabel("Розмір навчального набору")
    plt.legend()
    plt.grid(True)

plt.figure(figsize=(10, 6))
linear_reg = LinearRegression()
plot_learning_curves(linear_reg, X, y)
plt.title("Learning Curves (Linear Regression)")
plt.show()

polynomial_regression_10 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])

plt.figure(figsize=(10, 6))
plot_learning_curves(polynomial_regression_10, X, y)
plt.axis([0, 80, 0, 1.5])
plt.title("Learning Curves (Polynomial Degree 10)")
plt.show()

polynomial_regression_opt = Pipeline([
    ("poly_features", PolynomialFeatures(degree=3, include_bias=False)),
    ("lin_reg", LinearRegression()),
```

```
])

plt.figure(figsize=(10, 6))
plot_learning_curves(polynomial_regression_opt, X, y)
plt.axis([0, 80, 0, 1.5])
plt.title("Learning Curves (Polynomial Degree 3)")
plt.show()
```



Поліноміальна модель (ступінь 10):

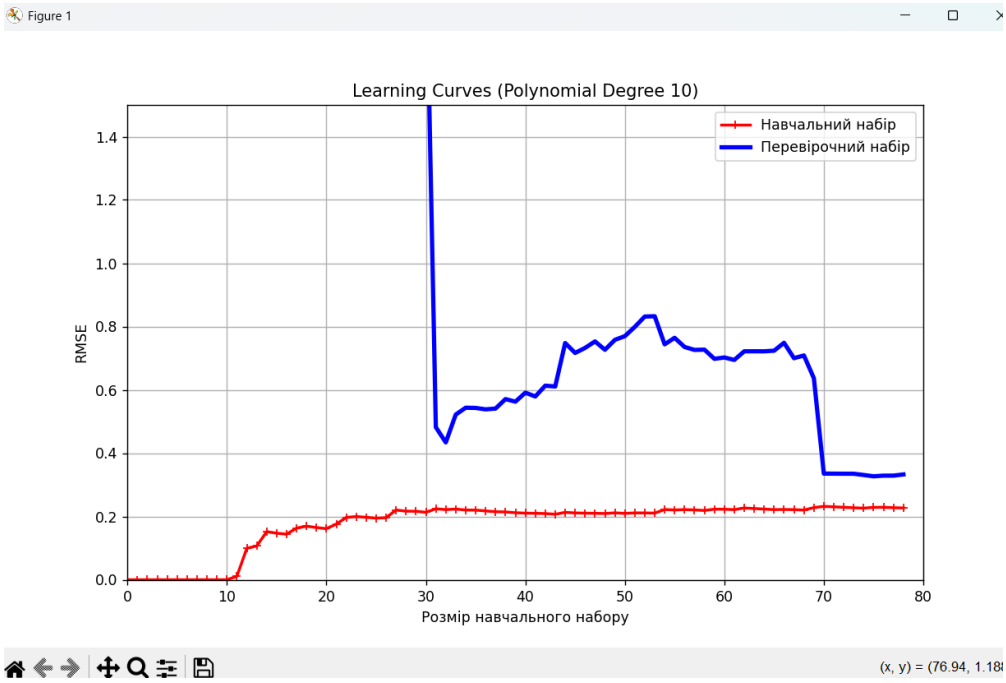


Рис. 5. Побудований графік кривих навчання для ступеня 10

Поліноміальна модель (ступінь 3):

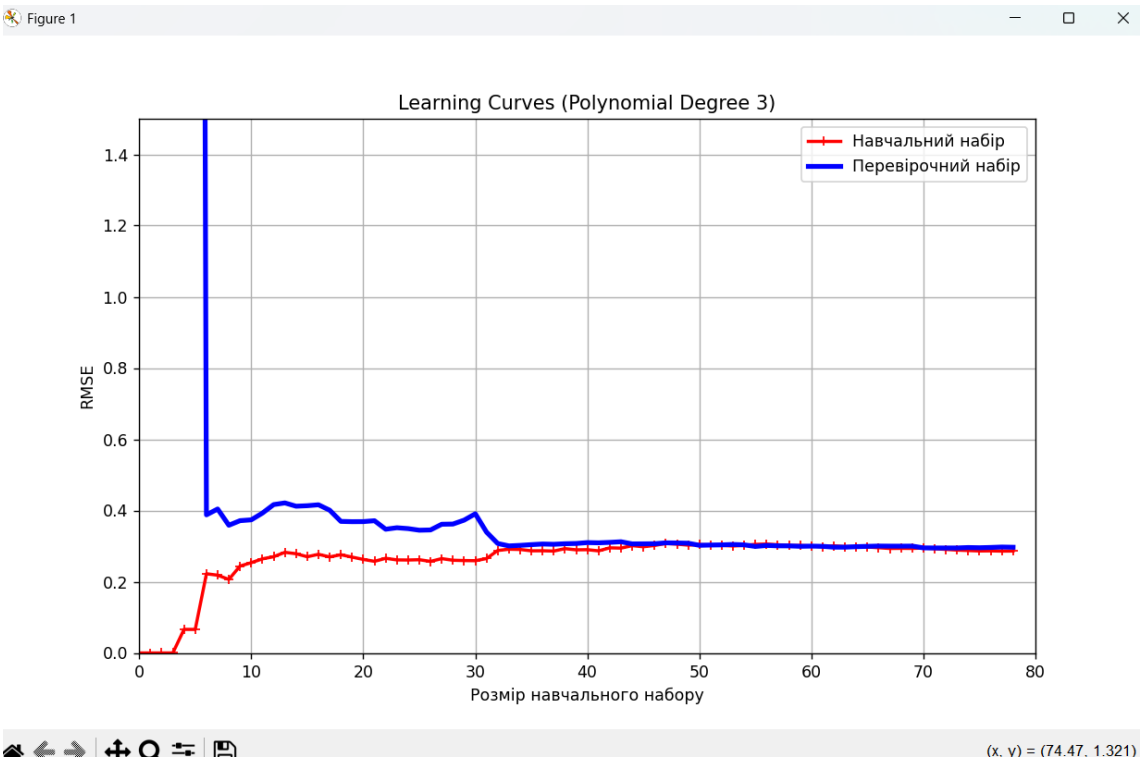


Рис. 6. Побудований графік кривих навчання для ступеня 3

Загальний висновок:

У ході лабораторної роботи я дослідив різні методи регресії: просту лінійну, багатовимірну та поліноміальну. Навчився оцінювати якість моделей за допомогою метрик (MAE, MSE, R2) та візуалізувати результати. Також дослідив поняття недонавчання та перенавчання за допомогою кривих навчання.