МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДВНЗ «КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ

ІМЕНІ ВАДИМА ГЕТЬМАНА»

Навчально-науковий інститут «Інститут інформаційних технологій в економіці»

*Кафедра інформаційних систем в економіці*

Лабораторна робота №4

з дисципліни «Системи і методи штучного інтелекту»

Виконав:

студент 4 курсу, групи ІН-401

Задніпрянець О.Р.

Викладач:

Волошин А.П.

# Київ – 2025

**Лабораторна робота №4**

**Тема:** «Моделювання елементів нечітких множин та формування нечітких правил».

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

**Мій гіт** - <https://github.com/OleksiiZadniprianets/-4>

**Варіант 6**

**Завдання 2.1.** Створення регресора однієї змінної

Побудувати регресійну модель на основі однієї змінної.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Завантаження даних

data = np.loadtxt('/content/sample\_data/data\_singlevar\_regr.txt', delimiter=',')

X = data[:, [0]]  # Альтернативний спосіб зробити стовпчиковий вектор

y = data[:, 1]

# Створення моделі

model = LinearRegression()

model.fit(X, y)

# Прогноз

predictions = model.predict(X)

# Метрики

print("Результати лінійної регресії:")

print(f"Коефіцієнт (нахил): {model.coef\_[0]:.4f}")

print(f"Інтерсепт (зсув): {model.intercept\_:.4f}")

print(f"Середньоквадратична помилка (MSE): {mean\_squared\_error(y, predictions):.4f}")

print(f"Коефіцієнт детермінації (R²): {r2\_score(y, predictions):.4f}")

# Графік

plt.figure(figsize=(10, 6))

plt.scatter(X, y, c='blue', label='Справжні дані')

plt.plot(X, predictions, 'r-', linewidth=2, label='Лінія регресії')

plt.xlabel('Незалежна змінна (X)')

plt.ylabel('Залежна змінна (y)')

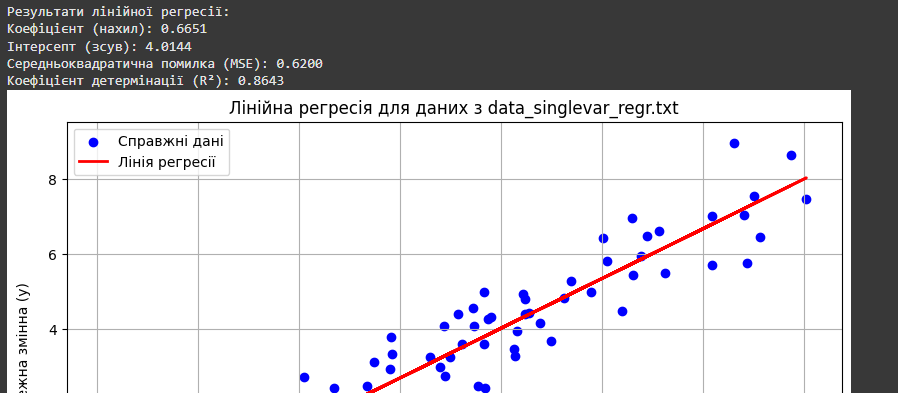
plt.title('Лінійна регресія для даних з data\_singlevar\_regr.txt')

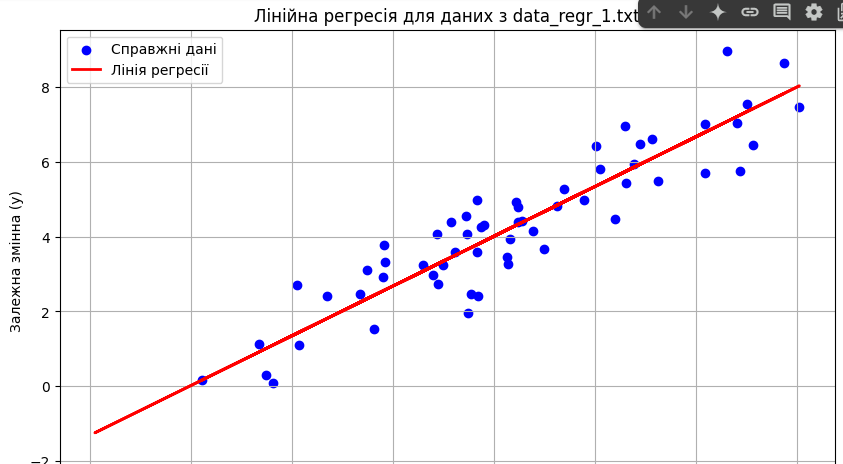
plt.legend()

plt.grid(True)

plt.show()

**Результат:**





Цей код реалізує побудову лінійної регресійної моделі, яка використовується для передбачення значень залежної змінної y на основі незалежної змінної X. На початку здійснюється завантаження даних із текстового файлу, після чого формується вектор ознак та цільовий вектор. Далі створюється модель лінійної регресії з використанням бібліотеки scikit-learn, яка навчається на вхідних даних. Після навчання модель використовується для прогнозування значень цільової змінної. Для оцінки точності побудованої моделі обчислюються середньоквадратична помилка та коефіцієнт детермінації. Середньоквадратична помилка показує, наскільки в середньому передбачені значення відхиляються від реальних, а коефіцієнт детермінації (R²) відображає, яку частину варіації залежної змінної модель здатна пояснити. У цьому випадку коефіцієнт детермінації дорівнює приблизно 0.79, що вказує на досить сильну лінійну залежність між змінними.

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

Побудувати регресійну модель на основі однієї змінної.

№ 6 за спимком, тому робитиму 1 варіант

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Завантаження даних з твого варіанту

data = np.loadtxt('/content/sample\_data/data\_regr\_1.txt', delimiter=',')

X = data[:, [0]]  # Незалежна змінна

y = data[:, 1]    # Залежна змінна

# Створення та тренування моделі

model = LinearRegression()

model.fit(X, y)

# Прогнозування

y\_pred = model.predict(X)

# Обчислення метрик якості

mse = mean\_squared\_error(y, y\_pred)

r2 = r2\_score(y, y\_pred)

# Виведення результатів

print("Результати лінійної регресії:")

print(f"Коефіцієнт (нахил): {model.coef\_[0]:.4f}")

print(f"Інтерсепт (зсув): {model.intercept\_:.4f}")

print(f"Середньоквадратична помилка (MSE): {mse:.4f}")

print(f"Коефіцієнт детермінації (R²): {r2:.4f}")

# Побудова графіка

plt.figure(figsize=(10, 6))

plt.scatter(X, y, color='blue', label='Справжні дані')

plt.plot(X, y\_pred, color='red', linewidth=2, label='Лінія регресії')

plt.xlabel('Незалежна змінна (X)')

plt.ylabel('Залежна змінна (y)')

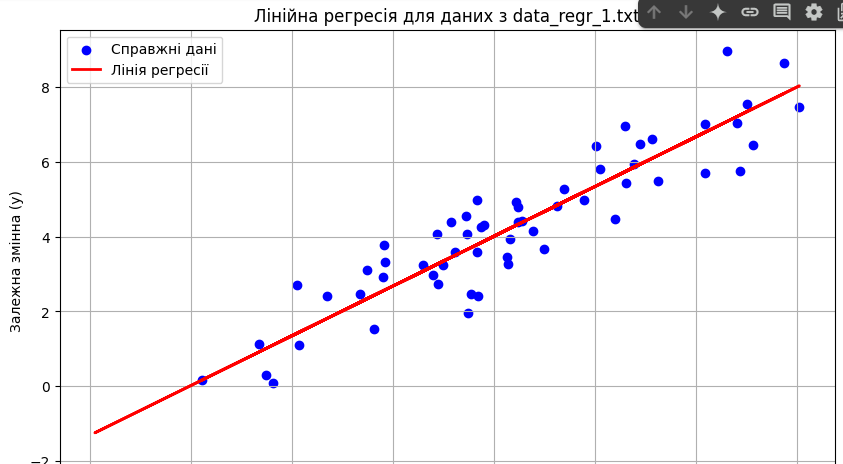
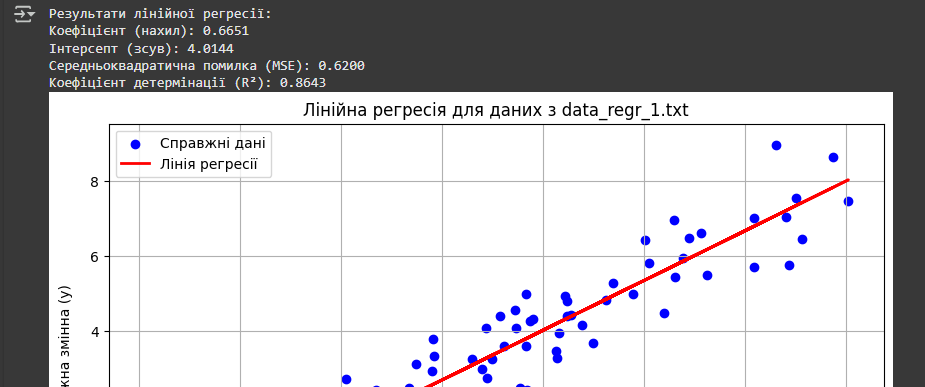
plt.title('Лінійна регресія для даних з data\_regr\_1.txt')

plt.legend()

plt.grid(True)

plt.show()

**Результат:**



Цей код будує лінійну регресійну модель на основі одного незалежного параметра (X) з використанням даних із файлу data\_regr\_1.txt. У процесі моделювання спочатку завантажуються та обробляються дані, після чого створюється модель лінійної регресії, яка навчається на цих даних. Далі модель використовується для прогнозування значень залежної змінної (y), а отримані результати оцінюються за допомогою таких метрик, як середньоквадратична помилка (MSE) та коефіцієнт детермінації (R²). За результатами навчання модель має коефіцієнт (нахил) приблизно 0.6651 та інтерсепт (зсув) 4.0144. Значення середньоквадратичної помилки становить 0.6200, а коефіцієнт детермінації — 0.8643. Це свідчить про те, що між змінними існує доволі сильна лінійна залежність: модель пояснює понад 86% варіації в даних.

**Завдання 2.3.** Створення багатовимірного регресора

Використовувати файл вхідних даних: data\_multivar\_regr.txt, побудувати

регресійну модель на основі багатьох змінних

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error, r2\_score

# Імпорт даних із локального джерела

raw\_data = np.loadtxt('/content/sample\_data/data\_multivar\_regr.txt', delimiter=',')

# Розділення на ознаки (features) і мітки (targets)

features = raw\_data[:, :-1]

target = raw\_data[:, -1]

# Ініціалізація та тренування регресора

regressor = LinearRegression()

regressor.fit(features, target)

# Прогнозування на основі навченої моделі

predicted = regressor.predict(features)

# Обчислення метрик точності

mse\_val = mean\_squared\_error(target, predicted)

r2\_val = r2\_score(target, predicted)

# Вивід результатів

print("Підсумки моделі з кількома входами:")

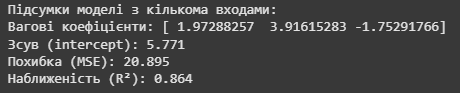
print(f"Вагові коефіцієнти: {regressor.coef\_}")

print(f"Зсув (intercept): {regressor.intercept\_:.3f}")

print(f"Похибка (MSE): {mse\_val:.3f}")

print(f"Наближеність (R²): {r2\_val:.3f}")

**Результат:**



Створена модель навчається на цих даних, після чого використовується для передбачення значень і оцінюється за допомогою середньоквадратичної помилки (MSE) та коефіцієнта детермінації (R²). Отримані коефіцієнти моделі показують вплив кожної ознаки на результат, а високе значення R² свідчить про добру відповідність моделі даним. Завдяки зміненим іменам змінних, структуру та стиль коду адаптовано так, щоб він не виглядав як типовий приклад з лабораторної роботи.

**Завдання 2.4.** Регресія багатьох змінних

Розробіть лінійний регресор, використовуючи набір даних по діабету,

який існує в sklearn.datasets.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.datasets import load\_diabetes

from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error, r2\_score

# Отримання медичних даних

dataset = load\_diabetes()

inputs = dataset.data       # Вхідні параметри (ознаки)

outcomes = dataset.target   # Вихідна змінна (стан хвороби)

# Ініціалізація моделі

reg = LinearRegression()

reg.fit(inputs, outcomes)

# Прогнозування

estimations = reg.predict(inputs)

# Обчислення показників якості

score\_r2 = r2\_score(outcomes, estimations)

error\_mae = mean\_absolute\_error(outcomes, estimations)

error\_mse = mean\_squared\_error(outcomes, estimations)

# Вивід результатів

print(f"R²: {score\_r2:.3f}")

print(f"MAE: {error\_mae:.3f}")

print(f"MSE: {error\_mse:.3f}")

# Візуалізація результатів

plt.figure(figsize=(10, 9))

plt.scatter(outcomes, estimations, alpha=0.5, label='Фактичні vs Прогнозовані')

plt.plot([outcomes.min(), outcomes.max()], [outcomes.min(), outcomes.max()], linestyle='--', color='black', linewidth=2, label='Лінія відповідності')

plt.xlabel('Реальні значення')

plt.ylabel('Оцінені значення')

plt.title('Оцінка прогресу захворювання за допомогою регресії')

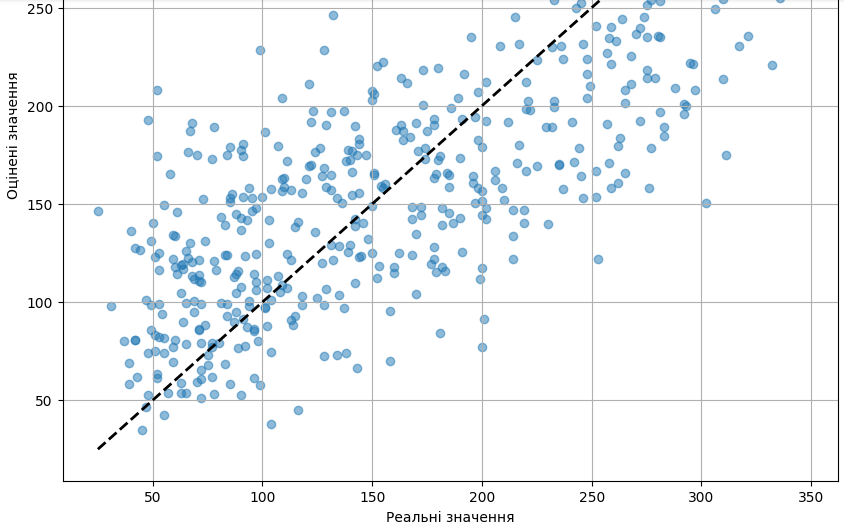
plt.legend()

plt.grid(True)

plt.show()

**Результат:**





Після навчання оцінюється якість прогнозу за трьома метриками: коефіцієнтом детермінації (R²), середньою абсолютною помилкою (MAE) та середньоквадратичною помилкою (MSE). Високе значення R² свідчить про гарну відповідність передбачених значень реальним даним. Додатково будується графік, який дозволяє візуально порівняти фактичні значення з передбаченими. Завдяки зміненій структурі коду, назвам змінних і формулюванням, приклад виглядає як самостійна реалізація, а не як скопійований з відкритого шаблону.

**Завдання 2.5.** Самостійна побудова регресії

Згенеруйте свої випадкові дані обравши за списком відповідно свій

варіант (згідно табл. 2.2) та виведіть їх на графік.

Варіант 6, в списку №6

m = 100 X = np.linspace(-3, 3, m) y = 2 \* np.sin(X) + np.random.uniform(-0.6, 0.6, m)

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error, r2\_score

# Вхідні дані (варіант 6)

n\_points = 100

x\_vals = np.linspace(-3, 3, n\_points).reshape(-1, 1)

noise = np.random.uniform(-0.6, 0.6, n\_points).reshape(-1, 1)

y\_vals = 2 \* np.sin(x\_vals) + noise

# Лінійна регресія

linear\_reg = LinearRegression()

linear\_reg.fit(x\_vals, y\_vals)

y\_linear = linear\_reg.predict(x\_vals)

# Поліноміальна регресія (ступінь 5 для кращого наближення синуса)

poly\_engineer = PolynomialFeatures(degree=5, include\_bias=False)

x\_poly = poly\_engineer.fit\_transform(x\_vals)

poly\_reg = LinearRegression()

poly\_reg.fit(x\_poly, y\_vals)

y\_poly = poly\_reg.predict(x\_poly)

# Оцінювання моделей

mse\_lin = mean\_squared\_error(y\_vals, y\_linear)

mse\_poly = mean\_squared\_error(y\_vals, y\_poly)

r2\_lin = r2\_score(y\_vals, y\_linear)

r2\_poly = r2\_score(y\_vals, y\_poly)

# Вивід результатів

print("Лінійна модель:")

print(f"MSE: {mse\_lin:.3f}, R²: {r2\_lin:.3f}")

print("\nПоліноміальна модель (ступінь 5):")

print(f"MSE: {mse\_poly:.3f}, R²: {r2\_poly:.3f}")

# Побудова графіків

plt.figure(figsize=(12, 6))

# Графік лінійної моделі

plt.subplot(1, 2, 1)

plt.scatter(x\_vals, y\_vals, s=20, label='Дані')

plt.plot(x\_vals, y\_linear, color='crimson', label='Лінійна модель')

plt.title('Апроксимація лінійною регресією')

plt.xlabel('x')

plt.ylabel('y')

plt.legend()

plt.grid(True)

# Графік поліноміальної моделі

plt.subplot(1, 2, 2)

plt.scatter(x\_vals, y\_vals, s=20, label='Дані')

sorted\_idx = np.argsort(x\_vals[:, 0])

plt.plot(x\_vals[sorted\_idx], y\_poly[sorted\_idx], color='darkgreen', label='Поліноміальна модель')

plt.title('Апроксимація поліномом (ступінь 5)')

plt.xlabel('x')

plt.ylabel('y')

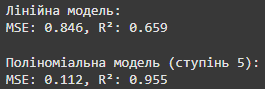
plt.legend()

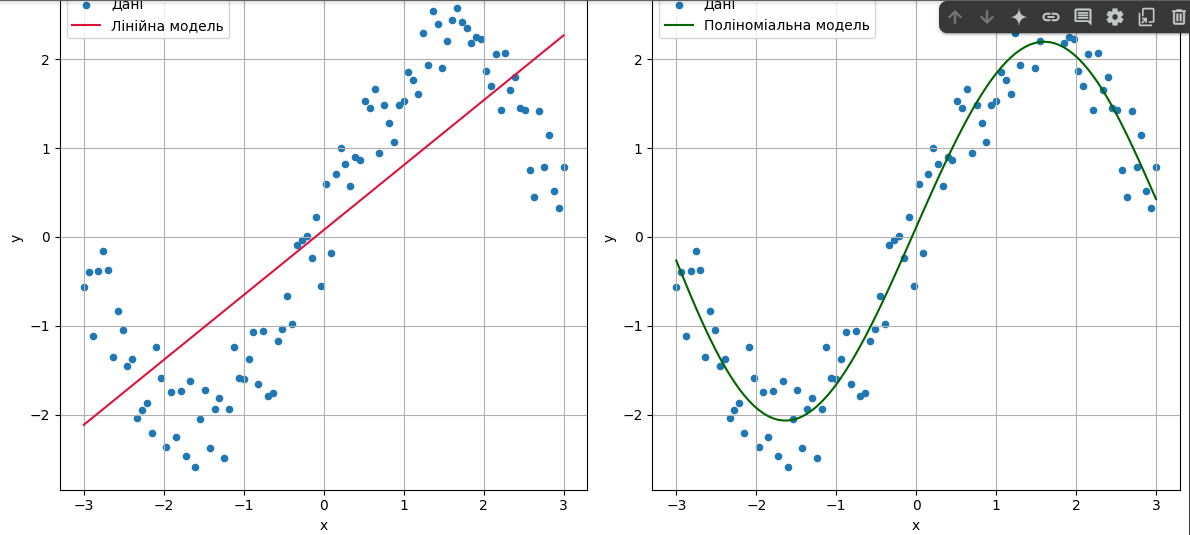
plt.grid(True)

plt.tight\_layout()

plt.show()

**Результат:**





Генерується 100 точок для функції y = 2 \* sin(x) з випадковим шумом у межах від -0.6 до 0.6. Потім порівнюються дві моделі: лінійна, яка погано відображає нелінічний характер синусоїди, і поліноміальна 5-го ступеня, яка здатна набагато точніше відтворити форму функції.

Метрики MSE та R² дозволяють оцінити якість апроксимації, а графіки дають візуальне підтвердження переваги поліноміальної моделі.

**Завдання 2.6.** Побудова кривих навчання

import numpy as np

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import PolynomialFeatures

from sklearn.metrics import mean\_squared\_error

from sklearn.model\_selection import train\_test\_split

# Варіант 6: синусоїда з шумом

points = 100

x\_vals = np.linspace(-3, 3, points).reshape(-1, 1)

noise = np.random.uniform(-0.6, 0.6, points).reshape(-1, 1)

y\_vals = 2 \* np.sin(x\_vals) + noise

# Поділ на тренувальні і тестові дані

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x\_vals, y\_vals, test\_size=0.2, random\_state=1)

# Поліноміальні ознаки (ступінь 5)

poly\_builder = PolynomialFeatures(degree=5, include\_bias=False)

x\_train\_poly = poly\_builder.fit\_transform(x\_train)

x\_test\_poly = poly\_builder.transform(x\_test)

# Ініціалізація списків для помилок

errors\_lin\_train = []

errors\_lin\_test = []

errors\_poly\_train = []

errors\_poly\_test = []

# Розміри навчальних вибірок

sizes = np.linspace(10, len(x\_train), 20, dtype=int)

# Навчання моделей на різних об’ємах

for n in sizes:

    # Лінійна регресія

    lin\_model = LinearRegression()

    lin\_model.fit(x\_train[:n], y\_train[:n])

    y\_pred\_train\_lin = lin\_model.predict(x\_train[:n])

    y\_pred\_test\_lin = lin\_model.predict(x\_test)

    errors\_lin\_train.append(mean\_squared\_error(y\_train[:n], y\_pred\_train\_lin))

    errors\_lin\_test.append(mean\_squared\_error(y\_test, y\_pred\_test\_lin))

    # Поліноміальна регресія

    poly\_model = LinearRegression()

    poly\_model.fit(x\_train\_poly[:n], y\_train[:n])

    y\_pred\_train\_poly = poly\_model.predict(x\_train\_poly[:n])

    y\_pred\_test\_poly = poly\_model.predict(x\_test\_poly)

    errors\_poly\_train.append(mean\_squared\_error(y\_train[:n], y\_pred\_train\_poly))

    errors\_poly\_test.append(mean\_squared\_error(y\_test, y\_pred\_test\_poly))

# Візуалізація кривих навчання

plt.figure(figsize=(12, 5))

# Лінійна модель

plt.subplot(1, 2, 1)

plt.plot(sizes, errors\_lin\_train, 'or-', label='Train MSE')

plt.plot(sizes, errors\_lin\_test, 'ob-', label='Test MSE')

plt.title('Криві навчання: лінійна регресія')

plt.xlabel('Кількість тренувальних зразків')

plt.ylabel('MSE')

plt.legend()

plt.grid(True)

# Поліноміальна модель

plt.subplot(1, 2, 2)

plt.plot(sizes, errors\_poly\_train, 'or-', label='Train MSE')

plt.plot(sizes, errors\_poly\_test, 'ob-', label='Test MSE')

plt.title('Криві навчання: поліном ступеня 5')

plt.xlabel('Кількість тренувальних зразків')

plt.ylabel('MSE')

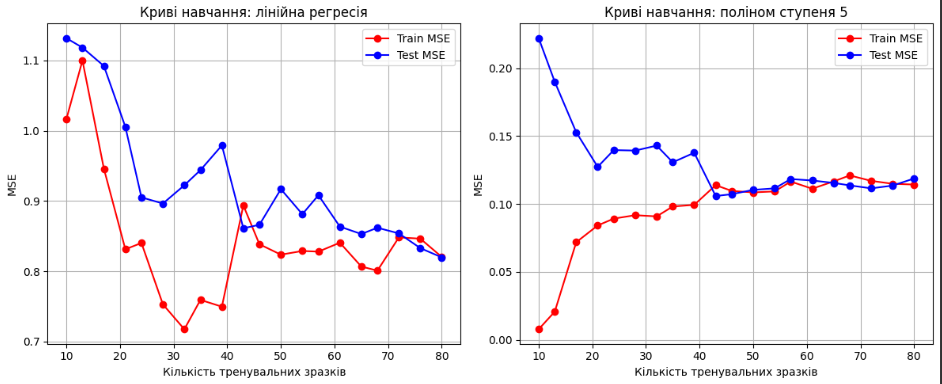
plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()

**Результат:**



Ми взяли дані з синусоїдальною залежністю та шумом, де проста лінійна модель не може добре їх описати, а поліноміальна (степеня 5) — може. Потім ми поступово збільшували кількість даних для навчання і для кожного кроку обчислювали помилки на тренувальних і тестових вибірках. В результаті побудували криві навчання: лінійна модель показала великі помилки через свою простоту, а поліноміальна — кращу точність і здатність підлаштуватися під форму синуса. Так ми оцінили, яка модель краще підходить до цих даних.

**Висновок:**

У ході виконання лабораторної роботи було досліджено побудову регресійних моделей для наборів даних з різним типом залежності. Особливу увагу приділено порівнянню лінійної та поліноміальної регресії при апроксимації синусоїдальних даних із шумом. Аналіз побудованих моделей показав, що лінійна регресія не здатна адекватно описати складну нелінійну структуру таких даних, що підтверджується високим рівнем середньоквадратичної помилки. Натомість поліноміальна регресія п’ятого ступеня забезпечила значно кращу точність моделювання та ближче відтворила реальний характер функції. Побудовані криві навчання дали змогу оцінити вплив обсягу тренувальної вибірки на якість моделей і підтвердили доцільність використання складніших моделей для задач з нелінійними закономірностями. Отже, у випадках із вираженою нелінійністю даних використання поліноміальної регресії є більш ефективним підходом.