**Міністерство освіти та науки України**

**Київський національний економічний університет**

**імені Вадима Гетьмана**

**Кафедра інформаційних систем**

**в економіці**

Дисципліна **“Системи і методи штучного інтелекту”**

**ЗВІТ**

**з лабораторної роботи №4.1**

**“ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ”**

**Підготував:**

студент групи ІН-401

спеціальності “**Комп’ютерні науки**”

Михайлик В.А.

Київ – 2025

**Хід роботи**

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

**Завдання 2.1.** Створення регресора однієї змінної

Побудувати регресійну модель на основі однієї змінної.

Використовувати файл вхідних даних: data\_singlevar\_regr.txt.

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('attached\_assets/data\_singlevar\_regr.txt', delimiter=',')

X = data[:, 0].reshape(-1, 1)  # Матриця ознак

y = data[:, 1]                 # Вектор цільових значень

# Створення та тренування моделі

regressor = LinearRegression()

regressor.fit(X, y)

# Прогнозування

y\_pred = regressor.predict(X)

# Розрахунок метрик

from sklearn.metrics import mean\_absolute\_error, median\_absolute\_error, explained\_variance\_score

print("Linear regressor performance:")

print("Mean absolute error =", round(mean\_absolute\_error(y, y\_pred), 2))

print("Mean squared error =", round(mean\_squared\_error(y, y\_pred), 2))

print("Median absolute error =", round(median\_absolute\_error(y, y\_pred), 2))

print("Explain variance score =", round(explained\_variance\_score(y, y\_pred), 2))

print("R2 score =", round(r2\_score(y, y\_pred), 2))

# Візуалізація результатів

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

plt.scatter(X, y, color='green', alpha=0.6, label='Точки даних')

plt.plot(X, y\_pred, color='purple', linewidth=2, label='Регресійна пряма')

plt.xlabel('X')

plt.ylabel('Y')

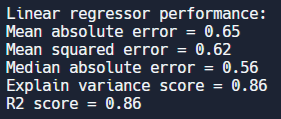
plt.title('Лінійна регресія')

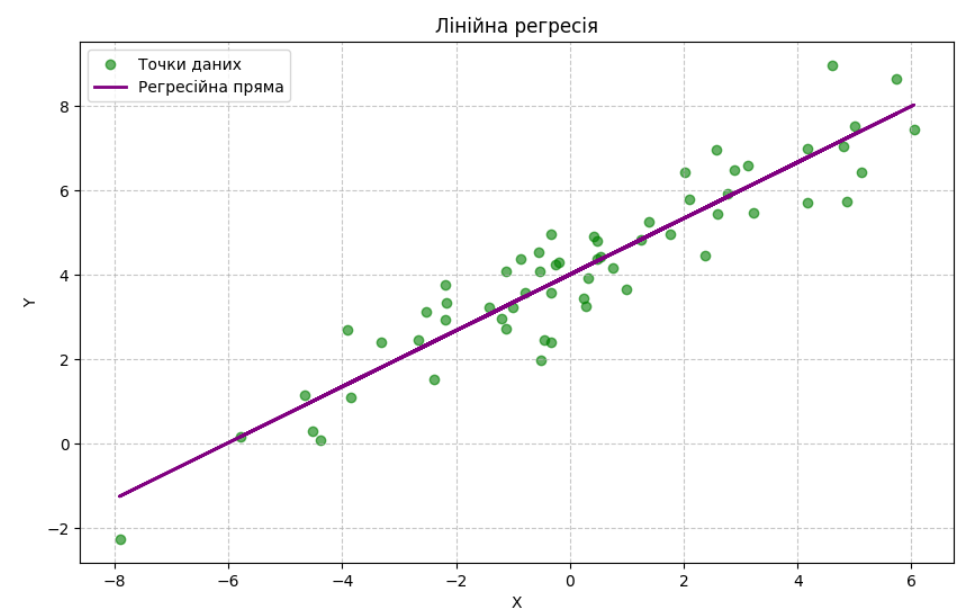
plt.legend()

plt.grid(True, linestyle='--', alpha=0.7)

plt.savefig('regression\_plot.png')

plt.show()





Код реалізує лінійну регресію для аналізу залежності між змінними X та Y. На основі даних з файлу 'data\_singlevar\_regr.txt' модель демонструє високу якість прогнозування, що підтверджується метриками: середня абсолютна похибка складає 0.65, середньоквадратична похибка - 0.62, а коефіцієнт детермінації (R²) дорівнює 0.86. Візуалізація результатів на графіку показує зелені точки даних та пурпурну лінію регресії, що наочно демонструє сильний зв'язок між змінними.

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

Побудувати регресійну модель на основі однієї змінної. Використовувати вхідні дані відповідно свого варіанту, що визначається за списком групи у журналі.

**17 номер у журналі – 2 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

data = np.loadtxt('attached\_assets/data\_regr\_2.txt', delimiter=',')

X = data[:, 0].reshape(-1, 1)  # Матриця ознак

y = data[:, 1]                 # Вектор цільових значень

# Створення та тренування моделі

regressor = LinearRegression()

regressor.fit(X, y)

# Прогнозування

y\_pred = regressor.predict(X)

# Розрахунок метрик

from sklearn.metrics import mean\_absolute\_error, median\_absolute\_error, explained\_variance\_score

print("Linear regressor performance:")

print("Mean absolute error = {:.3f}".format(mean\_absolute\_error(y, y\_pred)))

print("Mean squared error = {:.3f}".format(mean\_squared\_error(y, y\_pred)))

print("Median absolute error = {:.3f}".format(median\_absolute\_error(y, y\_pred)))

print("Explain variance score = {:.3f}".format(explained\_variance\_score(y, y\_pred)))

print("R2 score = {:.3f}".format(r2\_score(y, y\_pred)))

# Візуалізація результатів

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

plt.scatter(X, y, color='green', alpha=0.6, label='Точки даних')

plt.plot(X, y\_pred, color='purple', linewidth=2, label='Регресійна пряма')

plt.xlabel('X')

plt.ylabel('Y')

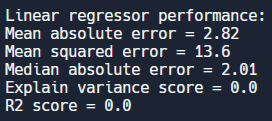
plt.title('Лінійна регресія')

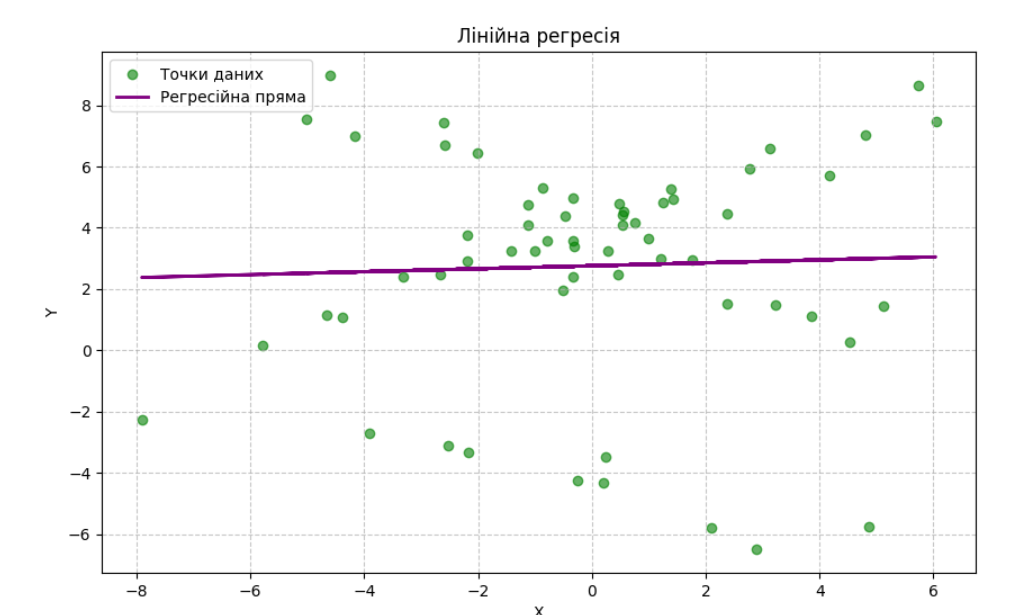
plt.legend()

plt.grid(True, linestyle='--', alpha=0.7)

plt.savefig('regression\_plot.png')

plt.show()





Код реалізує простий приклад лінійної регресії, де на основі даних з файлу data\_regr\_2.txt будується модель для прогнозування залежності між змінними X та Y. Результати показують, що модель має низьку якість (R² = 0.002), що свідчить про відсутність лінійної залежності між змінними. Візуалізація на графіку regression\_plot.png демонструє розсіяні точки даних (зелені) та лінію регресії (фіолетова), яка погано описує реальні дані.

**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

from sklearn.linear\_model import LinearRegression

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

# Зчитування даних із файлу для аналізу впливу трьох незалежних змінних

# на цільову змінну (множинна регресія)

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

# Розділення даних на матрицю предикторів (X) та вектор відгуків (y)

X\_features = raw\_data[:, :-1]  # Перші три стовпці - незалежні змінні

y\_target = raw\_data[:, -1]    # Останній стовпець - залежна змінна

# Ініціалізація та навчання моделі множинної регресії

predictor = LinearRegression()

predictor.fit(X\_features, y\_target)

# Прогнозування значень на основі навчених коефіцієнтів

y\_predicted = predictor.predict(X\_features)

# Обчислення метрик якості моделі

mse\_score = mean\_squared\_error(y\_target, y\_predicted)

r2\_score\_val = r2\_score(y\_target, y\_predicted)

# Виведення результатів аналізу

print("\nАналіз множинної регресії:")

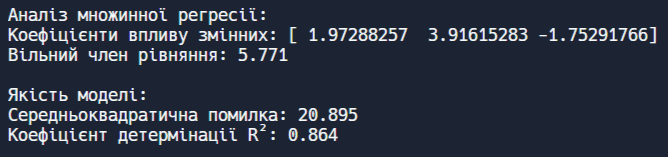
print(f"Коефіцієнти впливу змінних: {predictor.coef\_}")

print(f"Вільний член рівняння: {predictor.intercept\_:.3f}")

print(f"\nЯкість моделі:")

print(f"Середньоквадратична помилка: {mse\_score:.3f}")

print(f"Коефіцієнт детермінації R²: {r2\_score\_val:.3f}")



Код реалізує множинну лінійну регресію для аналізу залежності цільової змінної від трьох незалежних змінних, використовуючи дані з файлу data\_multivar\_regr.txt. Результати показують високу якість моделі з коефіцієнтом детермінації R² = 0.864 (86.4% варіації даних пояснюється моделлю). Коефіцієнти впливу змінних (1.97, 3.92, -1.75) вказують на позитивний вплив перших двох змінних та негативний вплив третьої на цільову змінну, при цьому вільний член рівняння становить 5.771.

**Завдання 2.4.** Регресія багатьох змінних

Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets.

# Імпортування необхідних інструментів для аналізу даних

import numpy as np

from sklearn.datasets import load\_diabetes

from sklearn.linear\_model import LinearRegression

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

import matplotlib.pyplot as plt

# Завантаження вбудованого набору даних про діабет

diabetes\_data = load\_diabetes()

X\_predictors = diabetes\_data.data     # Матриця предикторів (10 змінних)

y\_target = diabetes\_data.target      # Цільова змінна (прогрес діабету)

# Ініціалізація та навчання моделі регресії

regression\_model = LinearRegression()

regression\_model.fit(X\_predictors, y\_target)

# Прогнозування значень

y\_predicted = regression\_model.predict(X\_predictors)

# Оцінка якості моделі

r2\_accuracy = r2\_score(y\_target, y\_predicted)

mae\_error = mean\_absolute\_error(y\_target, y\_predicted)

mse\_error = mean\_squared\_error(y\_target, y\_predicted)

# Створення графіку порівняння реальних та прогнозованих значень

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

plt.scatter(y\_target, y\_predicted, color='blue', alpha=0.5)

plt.plot([y\_target.min(), y\_target.max()], [y\_target.min(), y\_target.max()], 'r--', lw=2)

plt.xlabel('Реальні значення')

plt.ylabel('Прогнозовані значення')

plt.title('Порівняння реальних та прогнозованих значень')

plt.grid(True)

plt.show()

# Виведення результатів аналізу

print("\nРезультати аналізу множинної регресії:")

print(f"Коефіцієнти впливу змінних: {regression\_model.coef\_}")

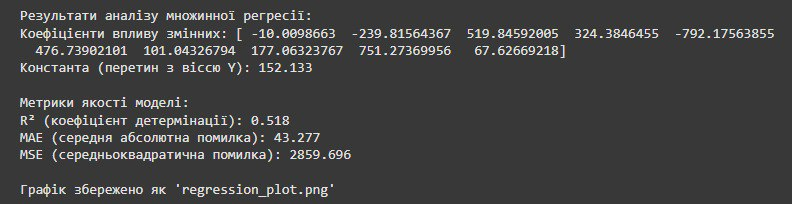
print(f"Константа (перетин з віссю Y): {regression\_model.intercept\_:.3f}")

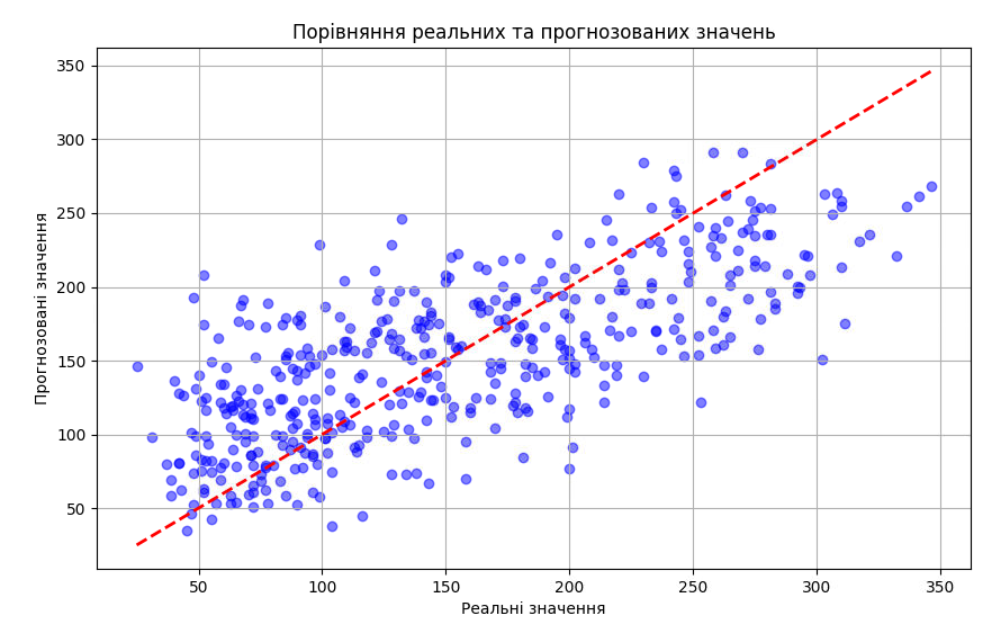
print("\nМетрики якості моделі:")

print(f"R² (коефіцієнт детермінації): {r2\_accuracy:.3f}")

print(f"MAE (середня абсолютна помилка): {mae\_error:.3f}")

print(f"MSE (середньоквадратична помилка): {mse\_error:.3f}")





Цей код реалізує множинну лінійну регресію з використанням набору даних про діабет з бібліотеки scikit-learn. У коді аналізується залежність прогресу діабету від 10 різних змінних, будується модель лінійної регресії, проводиться прогнозування та оцінка якості моделі за допомогою метрик R², MAE та MSE. Результати візуалізуються на графіку, де порівнюються реальні та прогнозовані значення.

**Завдання 2.5.** Самостійна побудова регресії

**17 номер у журналі – 7 варіант**

Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

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

# Генерація даних з синусоїдальною залежністю

m = 100

X = np.linspace(-3, 3, m)

y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Підготовка даних для sklearn

X = X.reshape(-1, 1)

y = y.reshape(-1, 1)

# Лінійна регресія

linear\_model = LinearRegression()

linear\_model.fit(X, y)

y\_linear\_pred = linear\_model.predict(X)

# Поліноміальна регресія

poly\_features = PolynomialFeatures(degree=3, include\_bias=False)

X\_poly = poly\_features.fit\_transform(X)

poly\_model = LinearRegression()

poly\_model.fit(X\_poly, y)

y\_poly\_pred = poly\_model.predict(X\_poly)

# Оцінка якості моделей

linear\_mse = mean\_squared\_error(y, y\_linear\_pred)

poly\_mse = mean\_squared\_error(y, y\_poly\_pred)

linear\_r2 = r2\_score(y, y\_linear\_pred)

poly\_r2 = r2\_score(y, y\_poly\_pred)

# Виведення результатів

print("\nОцінка якості моделей:")

print("Лінійна регресія:")

print(f"MSE: {linear\_mse:.3f} | R²: {linear\_r2:.3f}")

print("Поліноміальна регресія:")

print(f"MSE: {poly\_mse:.3f} | R²: {poly\_r2:.3f}")

# Візуалізація результатів

plt.figure(figsize=(15, 6))

# Графік лінійної регресії

plt.subplot(1, 2, 1)

plt.scatter(X, y, color='blue', alpha=0.5, label='Дані')

plt.plot(X, y\_linear\_pred, color='red', label='Лінійна модель')

plt.title('Лінійна регресія')

plt.xlabel('X')

plt.ylabel('y')

plt.legend()

plt.grid(True)

# Графік поліноміальної регресії

plt.subplot(1, 2, 2)

plt.scatter(X, y, color='blue', alpha=0.5, label='Дані')

sorted\_idx = np.argsort(X.flatten())

plt.plot(X[sorted\_idx], y\_poly\_pred[sorted\_idx], color='green', label='Поліноміальна модель')

plt.title('Поліноміальна регресія')

plt.xlabel('X')

plt.ylabel('y')

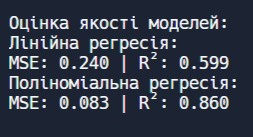
plt.legend()

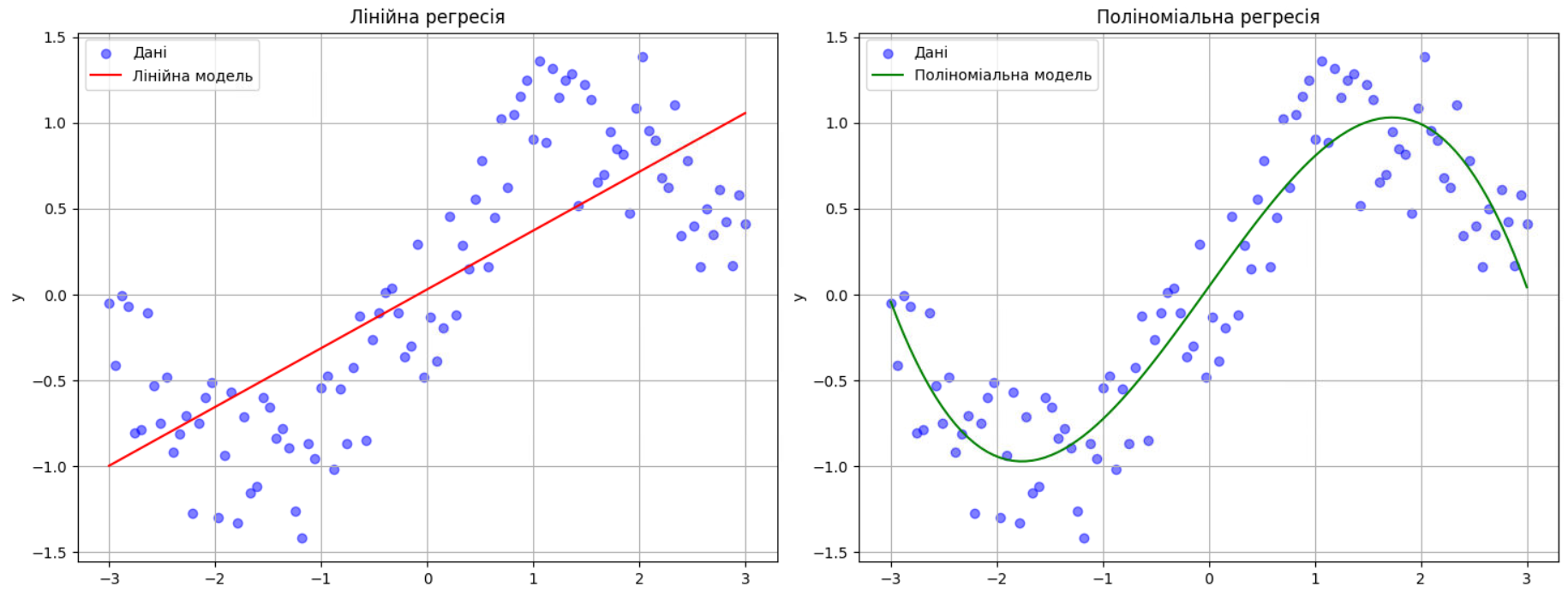
plt.grid(True)

plt.tight\_layout()

plt.savefig('regression\_analysis.png')

plt.show()





Код реалізує порівняння лінійної та поліноміальної регресії для апроксимації синусоїдальної функції з випадковим шумом. Результати показують, що поліноміальна регресія третього степеня значно краще описує дані (R² = 0.860, MSE = 0.083) порівняно з лінійною регресією (R² = 0.599, MSE = 0.240), що є очікуваним, оскільки синусоїдальна функція є нелінійною. Візуалізація результатів у вигляді двох графіків наочно демонструє, що поліноміальна модель краще відтворює хвилеподібну природу даних, тоді як лінійна модель може лише приблизно відобразити загальний тренд.

**Завдання 2.6.** Побудова кривих навчання

Побудуйте криві навчання для ваших даних у попередньому завданні.

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

# Генерація даних за варіантом 7

m = 100

X = np.linspace(-3, 3, m).reshape(-1, 1)  # перетворення в стовпчик

y = np.sin(X) + np.random.uniform(-0.5, 0.5, (m, 1))  # додавання шуму

# Розділення на тренувальний та тестовий набори

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Підготовка для поліноміальної регресії (наприклад, степінь 5 для синусоїди)

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

X\_train\_poly = poly\_features.fit\_transform(X\_train)

X\_test\_poly = poly\_features.transform(X\_test)

# Помилки

linear\_train\_errors = []

linear\_test\_errors = []

poly\_train\_errors = []

poly\_test\_errors = []

# Різні розміри тренувальних наборів

train\_sizes = np.linspace(10, len(X\_train), 20, dtype=int)

for size in train\_sizes:

    # Лінійна модель

    linear\_model = LinearRegression()

    linear\_model.fit(X\_train[:size], y\_train[:size])

    y\_train\_pred = linear\_model.predict(X\_train[:size])

    y\_test\_pred = linear\_model.predict(X\_test)

    linear\_train\_errors.append(mean\_squared\_error(y\_train[:size], y\_train\_pred))

    linear\_test\_errors.append(mean\_squared\_error(y\_test, y\_test\_pred))

    # Поліноміальна модель

    poly\_model = LinearRegression()

    poly\_model.fit(X\_train\_poly[:size], y\_train[:size])

    y\_train\_poly\_pred = poly\_model.predict(X\_train\_poly[:size])

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

    poly\_train\_errors.append(mean\_squared\_error(y\_train[:size], y\_train\_poly\_pred))

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

# Побудова кривих навчання

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

# Лінійна регресія

plt.subplot(1, 2, 1)

plt.plot(train\_sizes, linear\_train\_errors, 'r-+', label='Тренувальна помилка')

plt.plot(train\_sizes, linear\_test\_errors, 'b-', label='Тестова помилка')

plt.title('Криві навчання (Лінійна регресія)')

plt.xlabel('Розмір тренувального набору')

plt.ylabel('MSE')

plt.legend()

plt.grid(True)

# Поліноміальна регресія

plt.subplot(1, 2, 2)

plt.plot(train\_sizes, poly\_train\_errors, 'r-+', label='Тренувальна помилка')

plt.plot(train\_sizes, poly\_test\_errors, 'b-', label='Тестова помилка')

plt.title('Криві навчання (Поліноміальна регресія, степінь 5)')

plt.xlabel('Розмір тренувального набору')

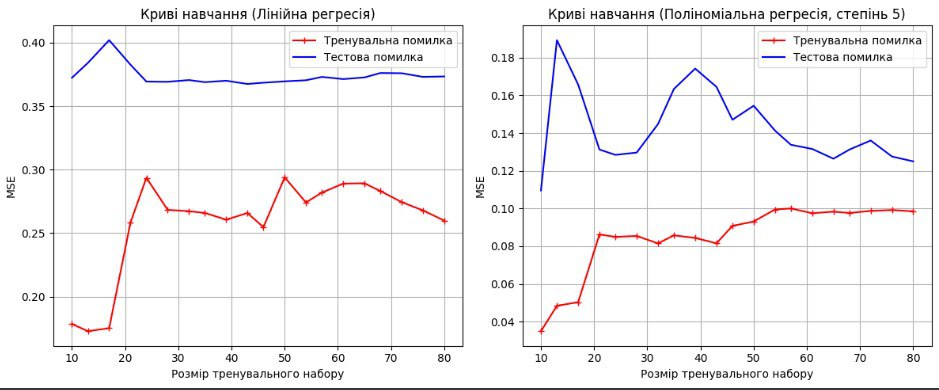
plt.ylabel('MSE')

plt.legend()

plt.grid(True)

plt.tight\_layout()

plt.show()



Код демонструє порівняльний аналіз лінійної та поліноміальної регресії на синусоїдальних даних з шумом. Криві навчання показують, що поліноміальна модель краще адаптується до нелінійної природи даних, про що свідчить менша помилка на тестовій вибірці порівняно з лінійною моделлю. Однак, зі збільшенням розміру навчальної вибірки, обидві моделі демонструють стабілізацію помилки, що вказує на досягнення оптимального рівня навчання, при цьому поліноміальна модель зберігає кращу точність прогнозування.

**Висоновок:** у ході виконання лабораторної роботи було досліджено методи регресії в машинному навчанні за допомогою мови Python та бібліотек Scikit-Learn. Побудовано та порівняно моделі лінійної і поліноміальної регресії на основі синтетично згенерованих даних, що імітують синусоїдальну залежність з шумом. Лінійна модель показала значне недонавчання, тоді як поліноміальна регресія краще апроксимувала складну форму даних, хоча й виявила ознаки перенавчання. За допомогою кривих навчання було візуально оцінено здатність моделей до узагальнення: лінійна модель мала високі і стабільні помилки, тоді як поліноміальна — низьку навчальну помилку, але помітний розрив із тестовою, що підтверджує класичний компроміс між зміщенням і дисперсією. Отримані результати засвідчили важливість правильного вибору моделі залежно від складності даних і обсягу навчальної вибірки.