Лабораторна робота №4

Лінійна регресія. Метод найменших квадратів. Інтерполяція

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

Завдання 1. Ретельно опрацювати теоретичні відомості з лекційного курсу

Завдання 2. Експериментально отримані N-значень величини Y при значеннях величини X. Відшукати параметри функції за методом найменших квадратів. Побудувати графіки, де в декартовій системі координат нанести експериментальні точки і графік апроксимуючої функції.

1

X	0	5	10	15	20	25
Y	21	39	51	63	70	90

Лістинг:

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


x = np.array([0, 5, 10, 15, 20, 25])
y = np.array([21, 39, 51, 63, 70, 90])

a = 18/7
b = 494/21

y_pred = a * x + b

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='red', label='Експериментальні точки', zorder=5)
plt.plot(x, y_pred, color='blue', label=f'Апроксимація: y = {a:.2f}x + {b:.2f}')

plt.title('Апроксимація методом найменших квадратів')
plt.xlabel('X')
plt.ylabel('Y')
plt.grid(True, linestyle='--')
plt.legend()
plt.show()
```

Результат:

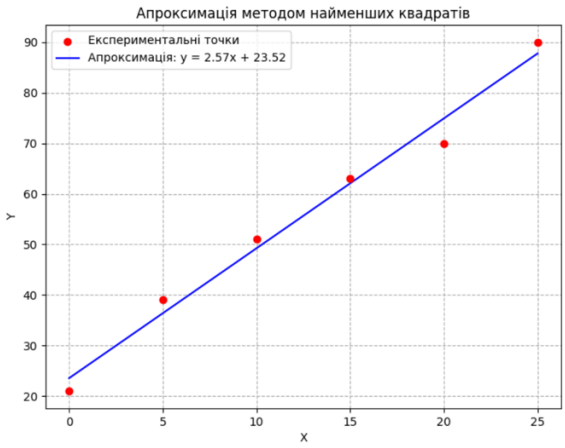

Рис.2.1. – Апроксимація методом найменших квадратів

Змн.	Арк.	№ докум.	Підпис	Дата	*ЖИТОМИРСЬКА ПОЛІТЕХНІКА.25.121.16.000 – Лр.4*		
Розроб.		*Кравченко К.М.*			*Звіт з лабораторної роботи №4*	*Літ.*	*Арк.* 1 / *Аркушів* 13
Перевір.		*Маєвський О.В.*					
Реценз.						*ФІКТ, гр. ІПЗ-22-2*	
Н. Контр.							
Зав.каф.							

Завдання № 3:
Виконати інтерполяцію функції, задану в табличній формі в п'яти точках (див. нижче). Розрахунки виконати в середовищі Python.
Вектори даних:

$$x := \begin{pmatrix} 0.1 \\ 0.3 \\ 0.4 \\ 0.6 \\ 0.7 \end{pmatrix} \qquad y := \begin{pmatrix} 3.2 \\ 3 \\ 1 \\ 1.8 \\ 1.9 \end{pmatrix}$$

Алгоритм розв'язку завдання № 3:
1. Заповнення матриці **X**;
2. Отримання коефіцієнтів інтерполяційного полінома;
3. Визначення функції полінома (прийняти поліном степеню 4);
4. Побудова графіка функції для інтерполюючого полінома;
5. Визначити значення функції в проміжних точках зі значеннями 0,2 і 0,5.

Лістинг:

```
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])

X_matrix = np.vander(x, 5)
print("Матриця X (Вандермонда):")
print(X_matrix)
print("-" * 20)

degree = 4
coefficients = np.polyfit(x, y, degree)

print("Коефіцієнти полінома (від a4 до a0):")
print(coefficients)
print("-" * 20)

poly_func = np.poly1d(coefficients)

print(f"Вигляд полінома:\n{poly_func}")
print("-" * 20)

x_targets = [0.2, 0.5]
print("Значення функції в проміжних точках:")
for val in x_targets:
    res = poly_func(val)
    print(f"x = {val} -> y = {res:.4f}")

x_smooth = np.linspace(min(x), max(x), 100)
y_smooth = poly_func(x_smooth)

plt.figure(figsize=(8, 6))
plt.scatter(x, y, color='red', s=80, label='Задані точки', zorder=5)
plt.plot(x_smooth, y_smooth, color='blue', label='Інтерполяційний поліном (ст.
4)')
plt.scatter(x_targets, poly_func(x_targets), color='green', marker='x', s=100,
label='Шукані точки (0.2, 0.5)', zorder=6)
```

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

Результат:

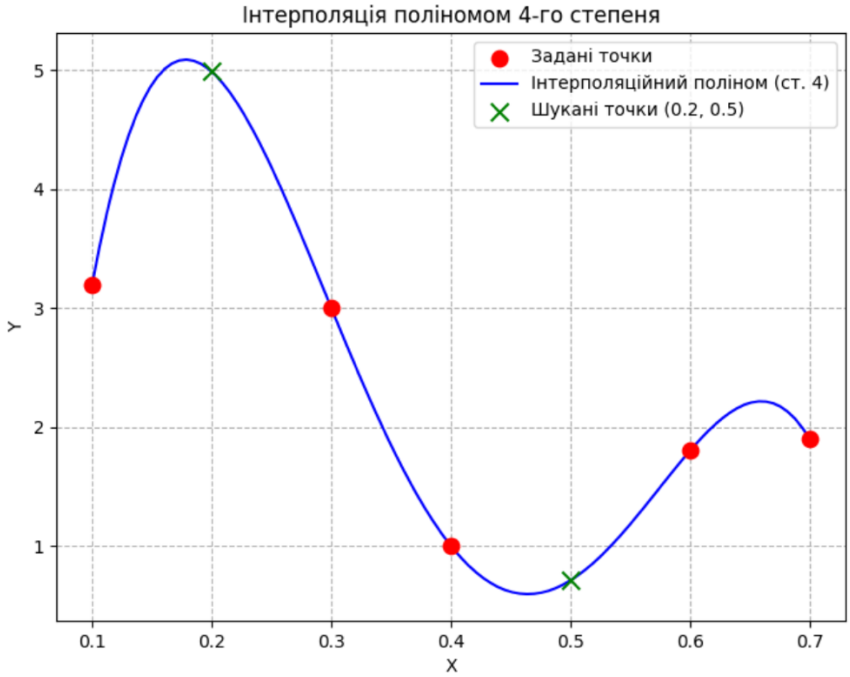

Рис.2.2. - Інтерполяція поліномом 4-го степеня

Завдання 2.1. Створення регресора однієї змінної

Побудувати регресійну модель на основі однієї змінної. Використовувати файл вхідних даних: data_singlevar_regr.txt.

Лістинг:

```
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'

try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл '{input_file}' не знайдено. Перевірте, чи він у тій
самій папці.")
    exit()

num_training = int(0.8 * len(X))
num_test = len(X) - num_training

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

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
```

```
y_test_pred = regressor.predict(X_test)

plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='green', label='Тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Лінія регресії')
plt.title('Лінійна регресія (одна змінна)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
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("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)
    print(f"\nМодель успішно збережено у файл: {output_model_file}")
```

Результат:

```
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

Модель успішно збережено у файл: model.pkl
```

Рис.2.3. – Результат регресора однієї змінної

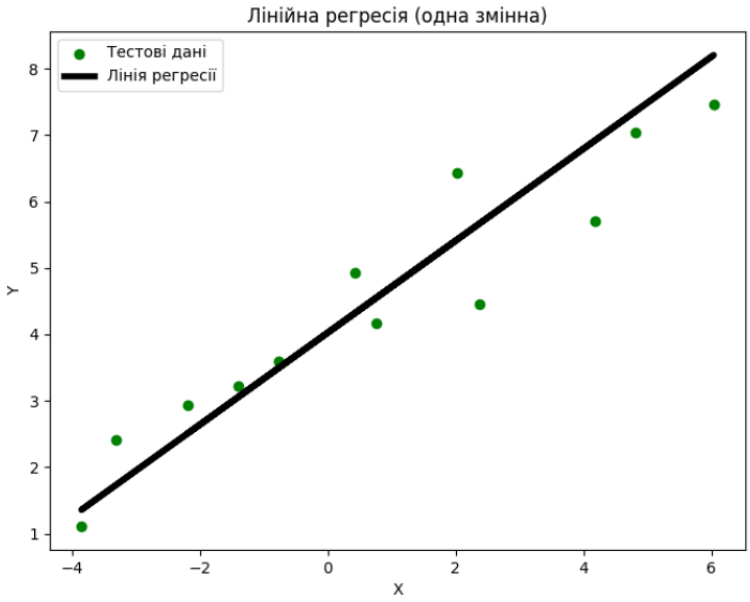

Рис.2.4. – Графік регресії однієї змінної

У роботі реалізовано модель лінійної регресії засобами Python. Графік показує, що побудована пряма добре проходить через експериментальні точки. Високий коефіцієнт детермінації (R2≈0.86) підтверджує точність моделі та її придатність для прогнозів.

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 (таблиця 2.1).

16
1

Варіант 1 файл: data_regr_1.txt

Лістинг:

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

input_file = 'data_regr_1.txt'

try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()

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:]

regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

y_test_pred = regressor.predict(X_test)

print("Linear regressor performance (Task 2.2):")
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))

plt.figure(figsize=(8, 6))
plt.scatter(X_test, y_test, color='green', label='Тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Лінія регресії')
plt.title('Лінійна регресія (одна змінна)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

Результат:

```
Linear regressor performance (Task 2.2):
Mean absolute error = 0.59
Mean squared error = 0.49
Median absolute error = 0.51
Explained variance score = 0.86
R2 score = 0.86
```

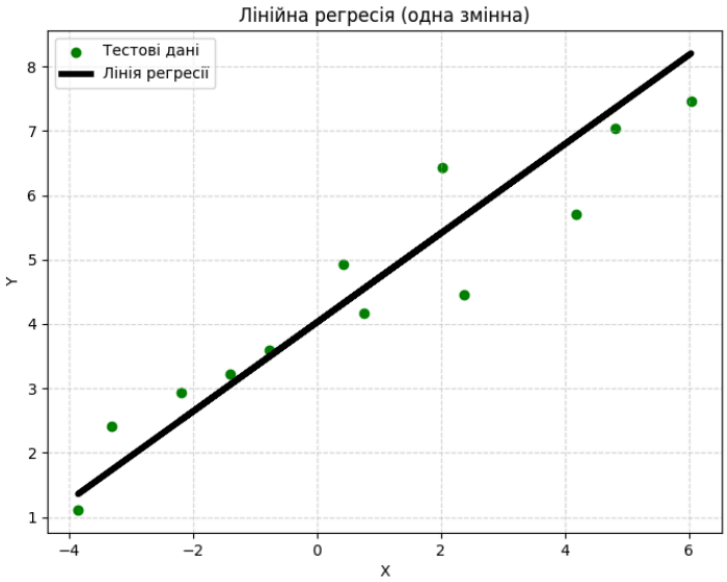

Рис.2.5.- 2.6. – Результат виконання завдання

У ході роботи побудовано та протестовано регресійну модель для прогнозування однієї змінної. Отримано високий коефіцієнт детермінації (R2≈0.86), що свідчить про якісну апроксимацію даних.

Варто зазначити, що результати (метрики похибок та графік) повністю співпадають із результатами попереднього завдання. Це пояснюється тим, що вхідний файл data_regr_1.txt містить ідентичний набір даних, що й файл із попереднього пункту, тому модель навчилася та спрацювала аналогічно.

Завдання 2.3. Створення багатовимірного регресора

Використовувати файл вхідних даних: data_multivar_regr.txt, побудувати регресійну модель на основі багатьох змінних.

Лістинг:

```
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'

try:
    data = np.loadtxt(input_file, delimiter=',')
    X, y = data[:, :-1], data[:, -1]
except OSError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()

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:]
```

```
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)

y_test_pred = linear_regressor.predict(X_test)

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))

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)

datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)

print("\n--- Prediction for datapoint [7.75, 6.35, 5.56] ---")
print("Expected value (approx): 41.35")

linear_prediction = linear_regressor.predict(datapoint)[0]
poly_prediction = poly_linear_model.predict(poly_datapoint)[0]

print(f"Linear regression prediction: {linear_prediction:.2f}")
print(f"Polynomial regression prediction: {poly_prediction:.2f}")

print("\n--- Висновок ---")
if abs(41.35 - poly_prediction) < abs(41.35 - linear_prediction):
    print("Поліноміальна регресія дала точніший результат.")
else:
    print("Лінійна регресія дала точніший результат.")
```

Результат:

```
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

--- Prediction for datapoint [7.75, 6.35, 5.56] ---
Expected value (approx): 41.35
Linear regression prediction: 36.05
Polynomial regression prediction: 41.08

--- Висновок ---
Поліноміальна регресія дала точніший результат.
```

Рис.2.7. – Результат виконання завдання

Побудовано модель багатовимірної регресії. Лінійна модель показала результат R2=0.86, проте при перевірці на конкретній вибірковій точці вона дала значну похибку (прогноз 36.05 проти очікуваних 41.35).

Застосування **поліноміальної регресії** дозволило отримати значно точніший прогноз (**41.08**), який майже співпадає з реальним значенням. Це підтверджує, що для даного набору даних поліноміальна модель є ефективнішою, ніж проста лінійна.

Завдання 2.4. Регресія багатьох змінних

Розробіть лінійний регресор, використовуючи набір даних по діабету, який існує в sklearn.datasets. Набір даних містить 10 вихідних змінних — вік, стать, індекс маси тіла, середній артеріальний тиск і шість вимірювань сироватки крові, отриманих у 442 пацієнтів із цукровим діабетом, а також реакцію, що цікавить, — кількісний показник прогресування захворювання через 1 рік після вихідного рівня. Отже, існує 442 екземпляри з 10 атрибутами. Колонка 11 є кількісною мірою прогресування захворювання через 1 рік після вихідного рівня. Кожен з цих 10 атрибутів був відцентрований по середньому та масштабований за часом стандартного відхилення n_samples (тобто сума квадратів кожного стовпця складає 1). Оригінальні дані можна завантажити з: https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html'.
Використайте всі функції набору даних про діабет, щоб побудувати двовимірний графік лінійної регресії. Побудуйте графік залежності між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням (крапками) та пряму лінію, по цьому графіку, що покаже, як лінійна регресія намагається провести пряму лінію, яка мінімізує залишкову суму квадратів між спостережуваними відповідями в наборі даних і відповідями, передбаченими лінійним наближенням. Також розрахуйте коефіцієнт кореляції R2, середню абсолютну помилку (MAE) і середньоквадратичну помилку (MSE).

Лістинг:

```
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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5,
random_state=0)

regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)

y_pred = regr.predict(X_test)

print("Коефіцієнти регресії (Coefficients): \n", regr.coef_)
print("Вільний член (Intercept): \n", regr.intercept_)

r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)

print("-" * 30)
print(f"R2 score (Коефіцієнт детермінації): {r2:.2f}")
print(f"Mean Absolute Error (MAE): {mae:.2f}")
```

```
print(f"Mean Squared Error (MSE): {mse:.2f}")
print("-" * 30)

fig, ax = plt.subplots(figsize=(8, 6))

ax.scatter(y_test, y_pred, edgecolors=(0, 0, 0), alpha=0.7, label='Дані')

ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=4, label='Ідеальна
відповідність')

ax.set_xlabel('Виміряно (Реальні дані)')
ax.set_ylabel('Передбачено (Прогноз моделі)')
ax.set_title('Графік: Виміряні vs Передбачені значення')
ax.legend()
plt.grid(True, linestyle='--', alpha=0.5)
plt.show()
```

Результат:

```
Коефіцієнти регресії (Coefficients):
 [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Вільний член (Intercept):
 154.35892852801342
------------------------------
R2 score (Коефіцієнт детермінації): 0.44
Mean Absolute Error (MAE): 44.80
Mean Squared Error (MSE): 3075.33
------------------------------
```

Рис.2.8. – Результати виконання завдання

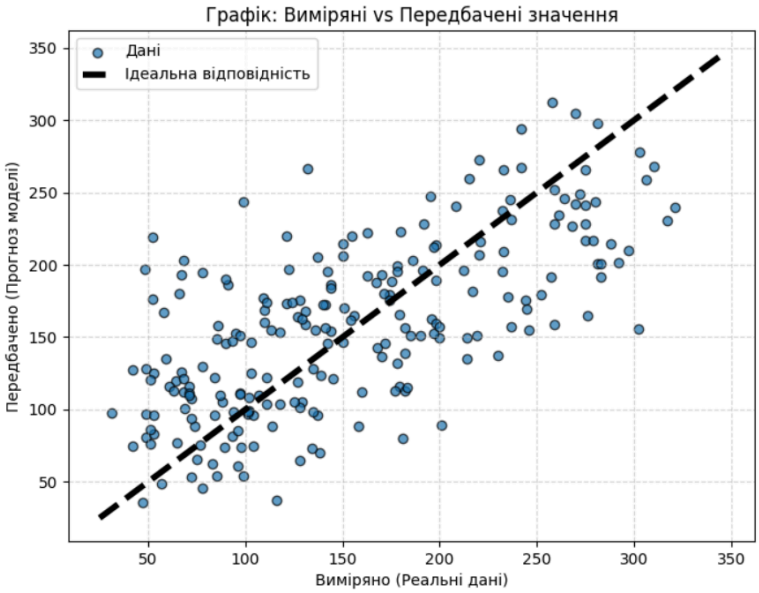

Рис.2.9. – Графік до завдання 2.4.

У роботі реалізовано лінійний регресор для прогнозування прогресування діабету на основі 10 фізіологічних показників (вік, стать, ІМТ, тиск, аналізи крові). Використано стандартний датасет sklearn.datasets.diabetes.

За результатами тестування на вибірці (50% від загального обсягу) отримано такі показники точності:

- **Коефіцієнт детермінації (R2): 0.44**. Це означає, що модель пояснює 44% варіації даних, що є прийнятним результатом для медичного прогнозування.

- **Середня абсолютна похибка (MAE): 44.80**. У середньому прогноз моделі відхиляється від реального показника на 44.8 одиниці.
- **Середньоквадратична похибка (MSE): 3075.33**.

Графічна візуалізація та розраховані метрики показують, що між вхідними даними та прогресуванням хвороби існує лінійна кореляція, проте вона не є строгою, що призводить до помітного розкиду значень на графіку.

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. Побудуйте по них модель лінійної регресії, виведіть на графік. Побудуйте по них модель поліноміальної регресії, виведіть на графік. Оцініть її якість.

16
6

Варіант 6

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

Лістинг:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import r2_score

np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, (m, 1))

lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin_pred = lin_reg.predict(X)

poly_features = PolynomialFeatures(degree=3, include_bias=False)
X_poly = poly_features.fit_transform(X)

poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)

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

print("--- Лінійна регресія ---")
print(f"R2 Score: {r2_score(y, y_lin_pred):.4f}")
print(f"Рівняння: y = {lin_reg.coef_[0][0]:.2f}x + ({lin_reg.intercept_[0]:.2f})")

print("\n--- Поліноміальна регресія (degree=3) ---")
print(f"R2 Score: {r2_score(y, poly_reg.predict(X_poly)):.4f}")
coefs = poly_reg.coef_[0]
intercept = poly_reg.intercept_[0]
print(f"Коефіцієнти: {coefs}")
print(f"Рівняння: y = {coefs[2]:.2f}x^3 + {coefs[1]:.2f}x^2 + {coefs[0]:.2f}x +
({intercept:.2f})")

plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', alpha=0.5, label='Дані (2*sin(X) + шум)')
plt.plot(X, y_lin_pred, color='red', linestyle='--', linewidth=2, label='Лінійна
```

```
регресія')
plt.plot(X_new, y_poly_pred, color='green', linewidth=3, label='Поліноміальна
регресія (deg=3)')

plt.title('Апроксимація функції y = 2sin(x)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True, linestyle='--', alpha=0.6)
plt.show()
```

Результат:

```
--- Лінійна регресія ---
R2 Score: 0.6206
Рівняння: y = 0.68x + (-0.04)

--- Поліноміальна регресія (degree=3) ---
R2 Score: 0.9438
Коефіцієнти: [ 1.79840077 -0.00220173 -0.20347892]
Рівняння: y = -0.20x^3 + -0.00x^2 + 1.80x + (-0.03)
```

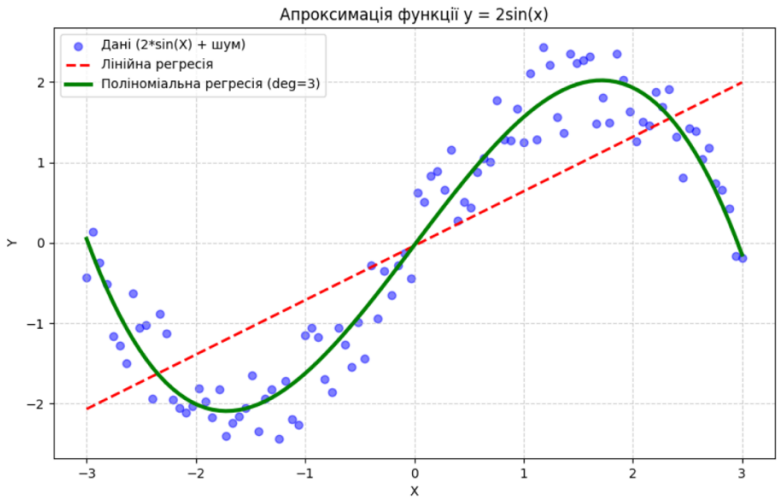

Рис.2.10. - 2.11 – Результати та графік виконання завдання

У ході виконання завдання було досліджено ефективність застосування регресійних моделей для апроксимації нелінійних даних. Для експерименту було згенеровано набір даних за законом y=2sin(x) з додаванням випадкового шуму. За результатами моделювання встановлено:

1. **Лінійна регресія** виявилася неефективною для опису хвилеподібної залежності. Коефіцієнт детермінації склав **R2≈0.62**, що вказує на низьку якість моделі.
2. **Поліноміальна регресія (ступеня 3)** показала високу точність апроксимації з коефіцієнтом детермінації **R2≈0.94**. Крива регресії вдало відтворила форму синусоїди.

Отримані коефіцієнти полінома (y≈−0.20x3+1.80x) узгоджуються з теоретичним розкладом функції 2sin(x) у ряд Тейлора (2x−x3/3), що підтверджує правильність навчання моделі та її здатність виявляти приховані математичні закономірності у зашумлених даних.

Завдання 2.6. Побудова кривих навчання

Побудуйте криві навчання для ваших даних у попередньому завданні.

Лістинг:

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

np.random.seed(42)
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = 2 * np.sin(X) + np.random.uniform(-0.6, 0.6, (m, 1))


def plot_learning_curves(model, X, y, title, ax, ylim=None):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=10)
    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)

        # Зберігаємо RMSE
        train_errors.append(np.sqrt(mean_squared_error(y_train[:m],
y_train_predict)))
        val_errors.append(np.sqrt(mean_squared_error(y_val, y_val_predict)))

    ax.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="Навчальний набір")
    ax.plot(np.sqrt(val_errors), "b-", linewidth=3, label="Перевірочний набір")
    ax.set_title(title)
    ax.set_xlabel("Розмір навчального набору")
    ax.set_ylabel("RMSE")
    ax.legend()
    ax.grid(True)
    if ylim:
        ax.set_ylim(ylim)


fig, axes = plt.subplots(1, 3, figsize=(18, 5))

# A) Лінійна регресія
linear_reg = LinearRegression()
plot_learning_curves(linear_reg, X, y, "Лінійна регресія (Underfitting)", axes[0],
ylim=[0, 3])

# Б) Поліноміальна регресія 10-го степеня
polynomial_regression_10 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression_10, X, y, "Поліном 10-го ступеня
(Overfitting)", axes[1], ylim=[0, 3])

# В) Поліноміальна регресія 2-го степеня
polynomial_regression_2 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(polynomial_regression_2, X, y, "Поліном 2-го ступеня",
```

```
axes[2], ylim=[0, 3])

plt.tight_layout()
plt.show()
```

Результат:

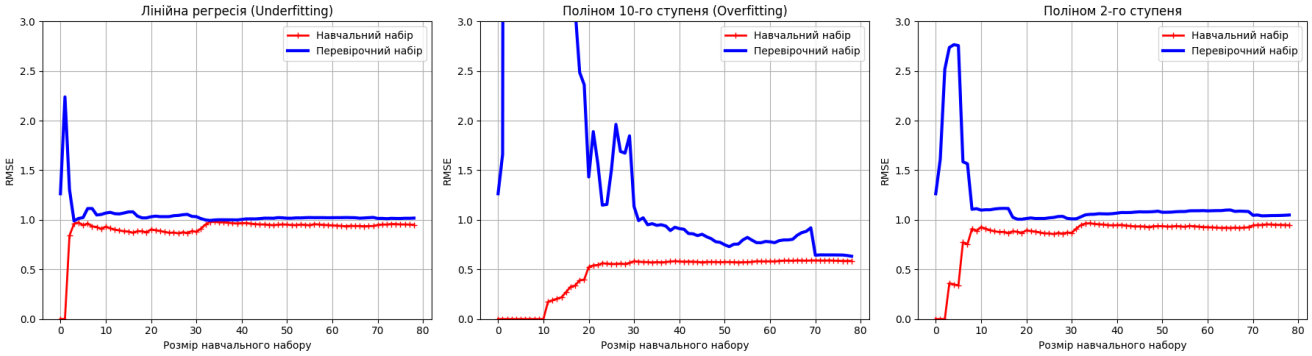

Рис.2.12. - 2.14 – Графіки до завдання 2.6

У ході роботи було побудовано криві навчання (Learning Curves) для діагностики поведінки трьох різних моделей регресії. Аналіз графіків дозволив оцінити баланс між зміщенням (Bias) та дисперсією (Variance):

1. **Лінійна регресія (Графік 1):** Демонструє типову картину **недонавчання (Underfitting)**. Криві навчання та перевірки швидко сходяться, але стабілізуються на досить високому рівні помилки (RMSE ≈ 1.0). Це свідчить про те, що модель занадто проста ("високе зміщення") і не може описати складну структуру синусоїдальних даних, навіть при збільшенні обсягу вибірки.
2. **Поліном 10-го ступеня (Графік 2):** Демонструє ознаки **перенавчання (Overfitting)**. Спостерігається суттєвий розрив між кривою навчання (червона лінія, помилка низька ≈ 0.6) та кривою перевірки (синя лінія, помилка вища). Це вказує на "високу дисперсію": модель запам'ятовує шум у навчальних даних і погано узагальнює результат на нових даних.
3. **Поліном 2-го ступеня (Графік 3):** Показує проміжний результат. Криві сходяться краще, ніж у моделі 10-го ступеня, і знаходяться трохи нижче, ніж у лінійної моделі, що свідчить про кращу узагальнюючу здатність.

Побудова кривих навчання довела, що проста лінійна модель є недостатньою для даних типу y=2sin(x), а поліном 10-го ступеня є надлишковим. Оптимальна модель має знаходитися посередині (наприклад, поліном 3-го або 4-го ступеня) для досягнення балансу між точністю на тренувальних даних та якістю прогнозу.

Висновок: У ході роботи ми опанували метод найменших квадратів та реалізували моделі лінійної і поліноміальної регресії в Python. Експериментально доведено, що для нелінійних даних поліноміальна регресія є значно точнішою. Також за допомогою метрик (R2, MSE) та кривих навчання ми навчилися оцінювати якість моделей і виявляти проблеми перенавчання чи недонавчання.