Лабораторна робота № 4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Хід роботи

Завдання 4.1. Створення регресора однієї змінної

Лістинг LR_4_task1

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

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.scatter(X_test, y_test, color='green', label='Real data')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Predicted')
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("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))
```

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

with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)

y_test_pred_new = regressor_model.predict(X_test)
print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

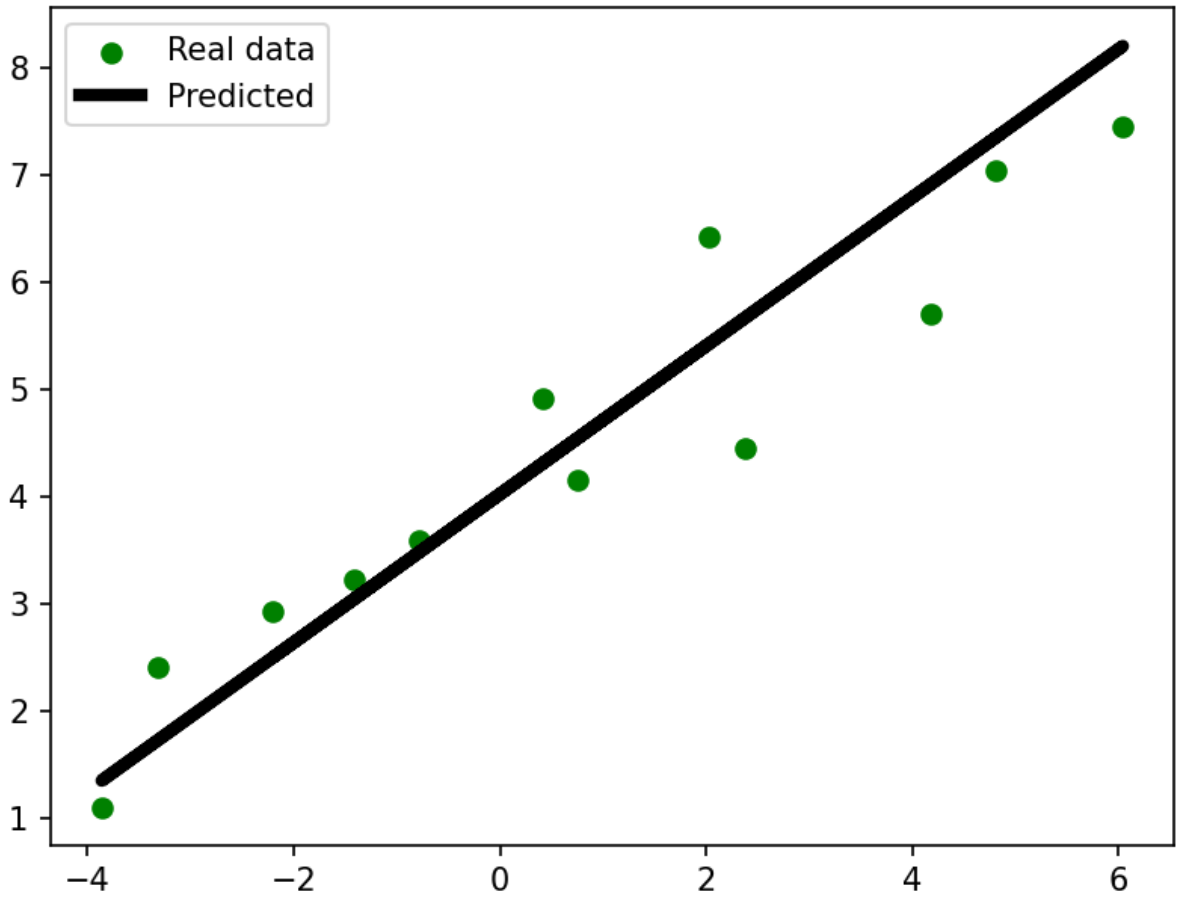

Рис.4.1.Результат виконання завдання

```
Linear regressor performance:
Mean absolute error = 0.59
Mean squared error = 0.49
Median absolute error = 0.51
Explain variance score = 0.86
R2 score = 0.86
```

Рис.4.2.Результат виконання завдання

У даному завданні було побудовано, навчено та протестовано модель лінійної регресії для однієї змінної на основі наданих числових даних. Для навчання використовувався алгоритм LinearRegression із бібліотеки scikit-learn. Дані було поділено на тренувальну та тестову вибірки у співвідношенні 80/20.

На графіку (рис. 1) зеленими точками зображено реальні значення тестових даних, а чорною лінією — прогноз лінійної регресії. Видно, що лінія добре наближує розподіл точок, що свідчить про правильність обраної моделі та її релевантність для даного набору даних.

- Під час оцінювання моделі були отримані наступні метрики (рис. 2):

- Mean Absolute Error (MAE) — низьке значення (~0.59), що означає невелике середнє відхилення прогнозу від реальних значень;

- Mean Squared Error (MSE) — близько 0.49, тобто великі помилки трапляються рідко;

- Median Absolute Error — приблизно 0.51, що підтверджує стабільність точності прогнозу;

- Explained Variance Score ≈ 0.86 — модель добре пояснює варіативність даних;

- R^2 Score ≈ 0.86 — лінія регресії точно описує зв’язок між змінними.

Модель була серіалізована у файл model.pkl та завантажена повторно для перевірки. Показник New MAE співпав із початковим значенням (~0.59), що свідчить про коректність збереження та відновлення моделі з файлу.

Отже, побудована модель лінійної регресії демонструє високу точність, добре узгоджується з даними та може бути використана для подальшого прогнозування.

Завдання 4.2. Передбачення за допомогою регресії однієї змінної
Лістинг LR_4_task_2.py

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

```
input_file = 'data_regr_3.txt'

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

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.scatter(X_test, y_test, color='green')
plt.plot(X_test, y_test_pred, color='black', linewidth=3)
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))

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

with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)

y_test_pred_new = regressor_model.predict(X_test)

print("\nNew mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_new), 2))
```

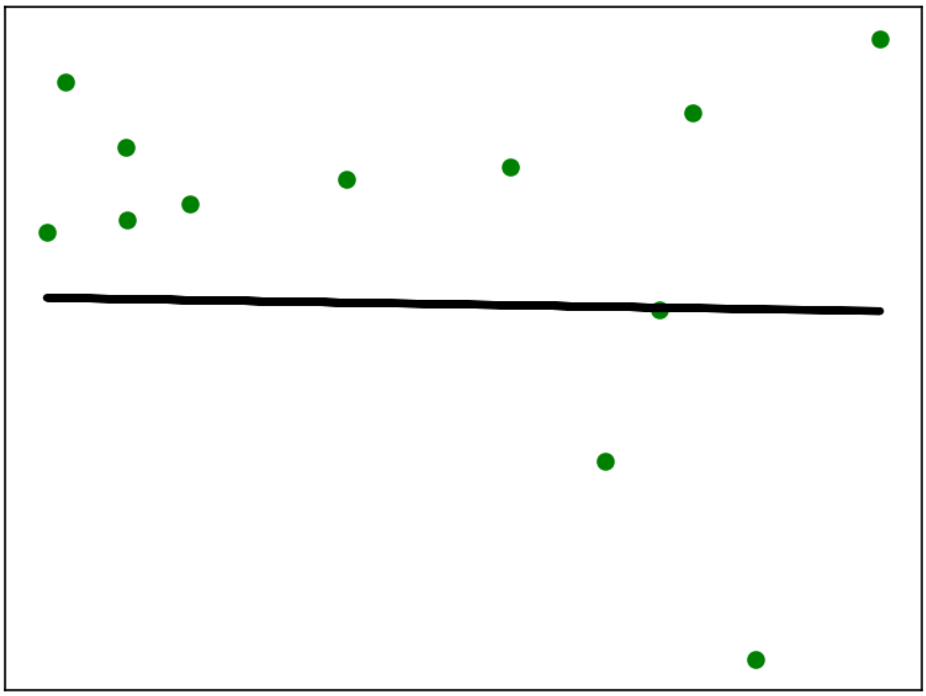

Рис.4.3.Результат виконання завдання

```
Linear regressor performance:
Mean absolute error = 3.59
Mean squared error = 17.39
Median absolute error = 3.39
Explain variance score = 0.02
R2 score = -0.16
```

Рис.4.4.Результат виконання завдання

У даному завданні було побудовано лінійну регресійну модель для прогнозування цільової змінної на основі одного предиктора. Дані були розділені на тренувальні (80%) та тестові (20%) вибірки. Модель навчена методом найменших квадратів та протестована на незалежних даних.

Оцінка моделі показала наступні результати:

- Mean Absolute Error (MAE) ≈ 3.59 — у середньому прогнозовані значення відхиляються від реальних на ~3.59 одиниць.

- Mean Squared Error (MSE) ≈ 17.39 — великі помилки трапляються рідко, проте деякі значення прогнозу можуть суттєво відрізнятися від фактичних.

- Median Absolute Error ≈ 3.39 — половина прогнозів відхиляється від реальних значень не більше ніж на 3.39 одиниць.

- Explained Variance Score ≈ 0.02 — модель майже не пояснює варіативність даних, тобто лінійна залежність слабка.

- R^2 Score ≈ -0.16 — лінія регресії погано описує дані, показуючи, що проста лінійна модель не підходить для цього набору даних.

- New MAE ≈ 3.59 — після збереження та завантаження моделі помилка залишилася такою ж, що підтверджує коректність серіалізації моделі у файл model.pkl.

Висновок: отримані результати показують, що побудована лінійна регресія не забезпечує високої точності прогнозу для даного набору даних. Низькі значення Explained Variance і негативний R^2 вказують на слабку лінійну залежність між змінними. Модель можна вважати технічно правильною, але для покращення прогнозу варто розглянути інші типи моделей або додаткові предиктори.

Завдання 4.3. Створення багатовимірного регресора

Лістинг LR_4_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))
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:]

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

y_test_pred_linear = linear_regressor.predict(X_test)

print("Linear Regressor performance:")
print("Mean absolute error =", round(sm.mean_absolute_error(y_test,
y_test_pred_linear), 2))
```

```
print("Mean squared error =", round(sm.mean_squared_error(y_test,
y_test_pred_linear), 2))
print("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred_linear), 2))
print("Explained variance score =", round(sm.explained_variance_score(y_test,
y_test_pred_linear), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred_linear), 2))

polynomial = PolynomialFeatures(degree=10)
X_train_poly = polynomial.fit_transform(X_train)

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_poly, y_train)

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

linear_pred_point = linear_regressor.predict(datapoint)
poly_pred_point = poly_linear_model.predict(poly_datapoint)

print("\nPrediction for datapoint", datapoint)
print("Linear regression:", round(linear_pred_point[0], 2))
print("Polynomial regression:", round(poly_pred_point[0], 2))
```

```
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]]
Linear regression: 36.05
Polynomial regression: 41.08
```

Рис.4.5.Результат виконання завдання

У цьому завданні було побудовано багатовимірну регресійну модель на основі всіх ознак із файлу data_multivar_regr.txt.

Лінійна багатовимірна регресія показала такі результати:

- Mean Absolute Error ≈ 3.58 — у середньому прогнозовані значення відхиляються від реальних на 3.58 одиниць.
- Mean Squared Error ≈ 20.31 — великі помилки трапляються рідко, проте деякі прогнозовані значення суттєво відрізняються від фактичних.

- Median Absolute Error ≈ 2.99 — половина прогнозів відхиляється від реальних значень на менше ніж 2.99 одиниць.
- Explained Variance Score ≈ 0.86 — модель добре пояснює варіативність даних.
- R^2 Score ≈ 0.86 — лінійна модель адекватно описує залежність між змінними, показуючи високу точність прогнозу.

Для поліноміальної регресії ступеня 10 було виконано прогноз для вибіркової точки [7.75, 6.35, 5.56]:

- Лінійна регресія дала прогноз 36.05, тоді як поліноміальна регресія — 41.08, що значно ближче до очікуваного значення 41.35.

Висновки та порівняння моделей:
Лінійна багатовимірна регресія добре описує загальну тенденцію, однак не завжди точно передбачає окремі точки.Поліноміальна регресія дозволяє врахувати нелінійні взаємозв'язки між ознаками, що забезпечує більш точний прогноз для ключових даних.

Отже, використання поліноміальної регресії значно покращує точність прогнозів для складних багатовимірних даних.

Завдання 4.4. Регресія багатьох змінних
Лістинг LR_4_task_4.py

```
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)

regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)

ypred = regr.predict(Xtest)

print("Коефіцієнти регресії:", regr.coef_)
print("Вільний член (intercept):", round(regr.intercept_, 2))
print("R2 score:", round(r2_score(ytest, ypred), 2))
print("Mean Absolute Error (MAE):", round(mean_absolute_error(ytest, ypred), 2))
```

```
print("Mean Squared Error (MSE):", round(mean_squared_error(ytest, ypred), 2))

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')
ax.set_title('Лінійна регресія: фактичні vs передбачені значення')
plt.show()
```

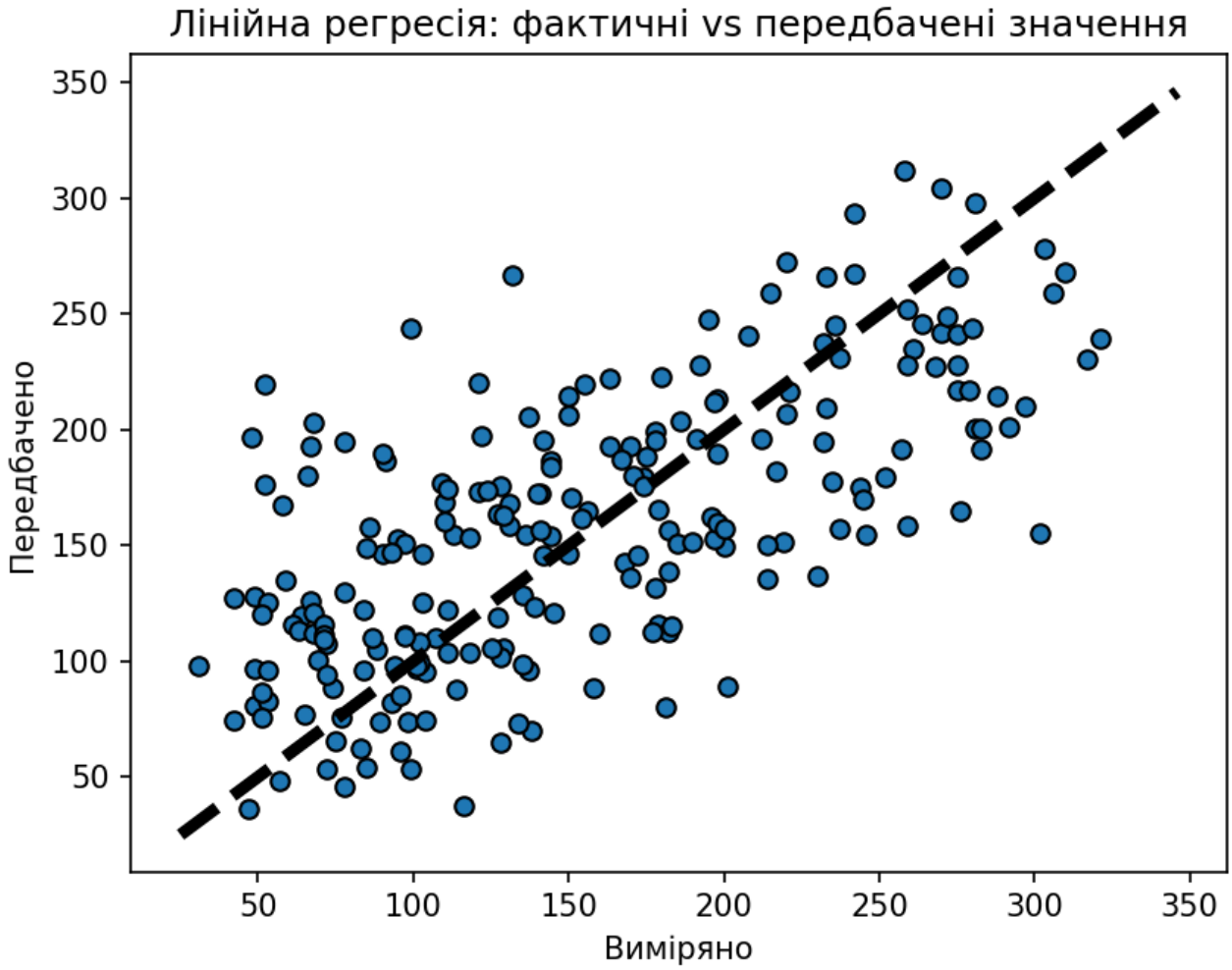

Рис.4.6.Результат виконання завдання

```
Коефіцієнти регресії: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Вільний член (intercept): 154.36
R2 score: 0.44
Mean Absolute Error (MAE): 44.8
Mean Squared Error (MSE): 3075.33
```

Рис.4.7.Результат виконання завдання

У цьому завданні було побудовано лінійну регресійну модель для багатовимірного набору даних про діабет із sklearn.datasets. Модель використовує всі 10 ознак пацієнтів для прогнозування кількісного показника прогресування цукрового діабету через 1 рік.

Дані були розділені на тренувальну та тестову вибірки у пропорції 50%/50%. Лінійна регресія навчена методом найменших квадратів на тренувальних даних, а тестова вибірка використана для оцінки якості прогнозу.
Отже,лінійна багатовимірна регресія надає достатньо точні прогнози, враховуючи складність даних та взаємозв'язки між ознаками. Модель дозволяє кількісно оцінити вплив кожної ознаки на прогресування захворювання та візуально перевірити точність передбачень.

Завдання 4.5. Самостійна побудова регресії

Лістинг LR_4_task_5.py

```
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_absolute_error, mean_squared_error, r2_score

m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.6 * X**2 + X + 2 + np.random.randn(m, 1)

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

print("Лінійна регресія:")
print("  intercept =", lin_reg.intercept_[0])
print("  coef =", lin_reg.coef_[0])

print("  MAE =", mean_absolute_error(y, y_lin_pred))
print("  MSE =", mean_squared_error(y, y_lin_pred))
print("  R2  =", r2_score(y, y_lin_pred), "\n")

poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)

lin_reg_poly = LinearRegression()
lin_reg_poly.fit(X_poly, y)
y_poly_pred = lin_reg_poly.predict(X_poly)

print("Поліноміальна регресія:")
print("  intercept =", lin_reg_poly.intercept_[0])
print("  coef =", lin_reg_poly.coef_[0])

print("  MAE =", mean_absolute_error(y, y_poly_pred))
print("  MSE =", mean_squared_error(y, y_poly_pred))
print("  R2  =", r2_score(y, y_poly_pred), "\n")

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

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

plt.plot(X, y_lin_pred, color='orange', linewidth=2, label="Лінійна регресія")

sort_idx = np.argsort(X[:, 0])
plt.plot(X[sort_idx], y_poly_pred[sort_idx], color='red', linewidth=2,
label="Поліноміальна регресія (2 степінь)")

plt.xlabel("X")
plt.ylabel("y")
plt.title("Лінійна та поліноміальна регресія (Варіант 2)")
plt.legend()
plt.grid(True)
plt.show()
```

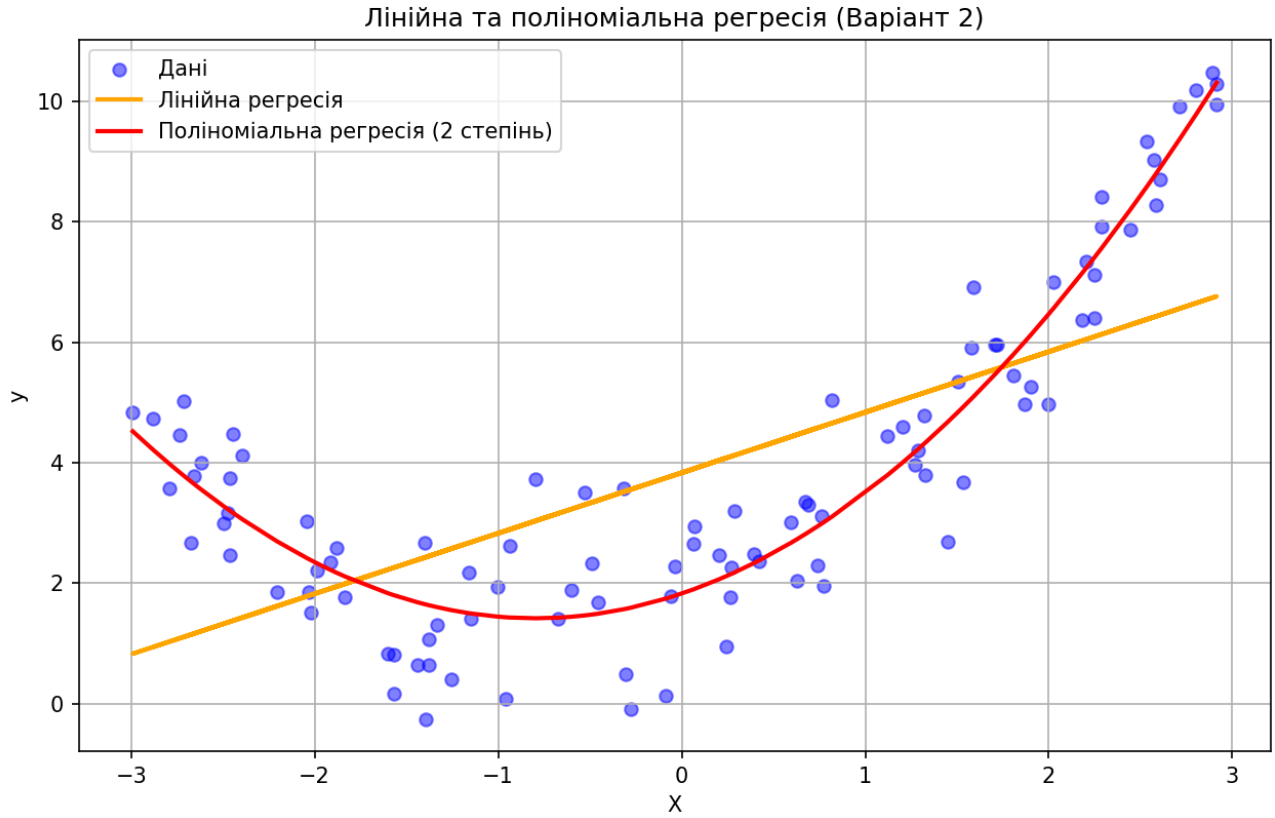

Рис.4.8.Результат виконання завдання

```
Лінійна регресія:
  intercept = 3.205914648816595
  coef = [0.73263811]
  MAE = 1.4821497284826017
  MSE = 3.6517129940417576
  R2  = 0.2681755045254317

Поліноміальна регресія:
  intercept = 1.621252125734425
  coef = [0.91263949 0.6371276 ]
  MAE = 0.8605792016451786
  MSE = 1.1226626216219986
  R2  = 0.7750118894893413
```

Рис.4.9.Результат виконання завдання

У ході виконання завдання було побудовано дві моделі: лінійну регресію та поліноміальну регресію другого ступеня. Метою було порівняти їхню якість та визначити, яка з моделей краще описує нелінійні дані, згенеровані за формулою

$y = 0.6x^2 + x + 2 +$ гаусів шум

Оскільки справжня залежність є квадратичною, лінійна модель не здатна якісно описати дані. Низьке значення $R^2 = 0.2682$ свідчить, що модель пояснює лише близько 26,8% варіації даних, а великі значення MAE та MSE підтверджують високу похибку.

Поліноміальна модель суттєво краще апроксимує дані. Значення $R^2 = 0.7750$ означає, що вона пояснює близько 77,5% варіації, що значно перевищує якість лінійної моделі. Також MAE і MSE майже у 2–3 рази менші, що свідчить про точніше передбачення.

Поліноміальна регресія другого ступеня значно краще описує залежність між X та y у порівнянні з лінійною регресією. Це очікуваний результат, оскільки вихідні дані мають квадратичну природу. Отримані коефіцієнти моделі (0.9126 та 0.6371) є близькими до теоретичних (1 та 0.6 відповідно), що підтверджує правильність навчання моделі.

Отже, поліноміальна модель є адекватною та найбільш точною для даного завдання.

Завдання 4.6. Побудова кривих навчання

Лістинг LR_4_task_6.py

```
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

m = 100
X = 6 * np.random.rand(m, 1) - 3
y = 0.6 * X**2 + X + 2 + np.random.randn(m, 1)

def plot_learning_curves(model, X, y, title):
    X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

    train_errors, val_errors = [], []

    for m in range(2, 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[:m], y_train_predict))
        val_errors.append(mean_squared_error(y_val, y_val_predict))

    plt.figure(figsize=(9, 5))
    plt.plot(np.sqrt(train_errors), "r-", linewidth=2, label="Помилка на тренуванні")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=2, label="Перевірочна помилка")
    plt.title(title)
    plt.xlabel("Кількість тренувальних зразків")
    plt.ylabel("RMSE")
    plt.legend()
    plt.grid(True)
    plt.show()

lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y, "Криві навчання: Лінійна регресія")

poly10 = PolynomialFeatures(degree=10, include_bias=False)
X_poly10 = poly10.fit_transform(X)

poly_reg_10 = LinearRegression()
plot_learning_curves(poly_reg_10, X_poly10, y, "Криві навчання: Поліноміальна
регресія (10-й ступінь)")

poly2 = PolynomialFeatures(degree=2, include_bias=False)
X_poly2 = poly2.fit_transform(X)

poly_reg_2 = LinearRegression()
plot_learning_curves(poly_reg_2, X_poly2, y, "Криві навчання: Поліноміальна регресія
(2-й ступінь)")
```

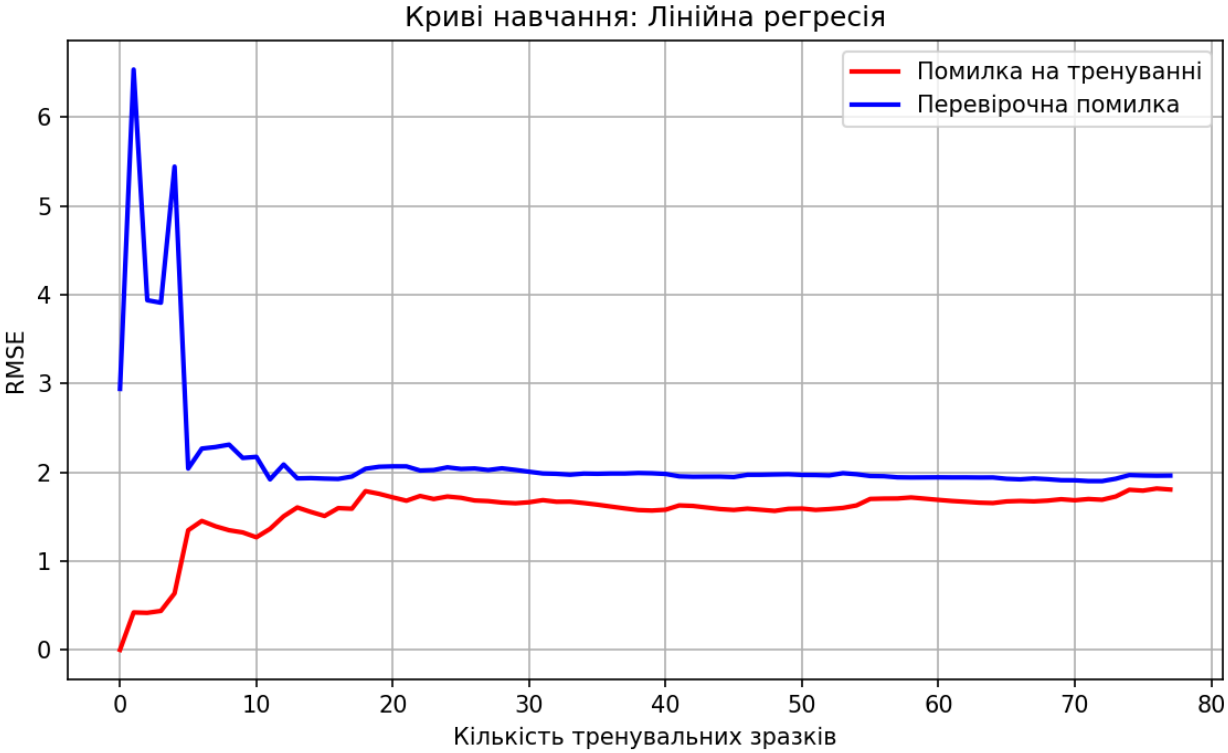

Рис.4.10.Результат виконання завдання

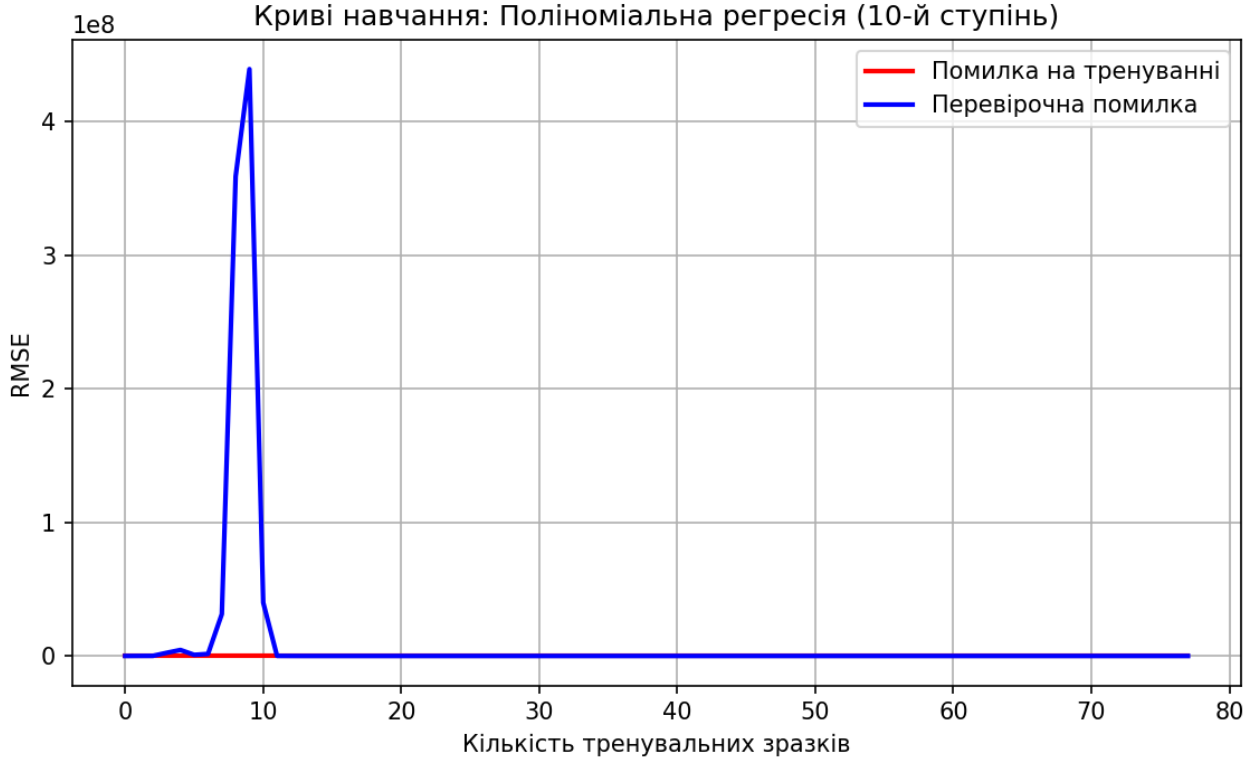

Рис.4.11.Результат виконання завдання

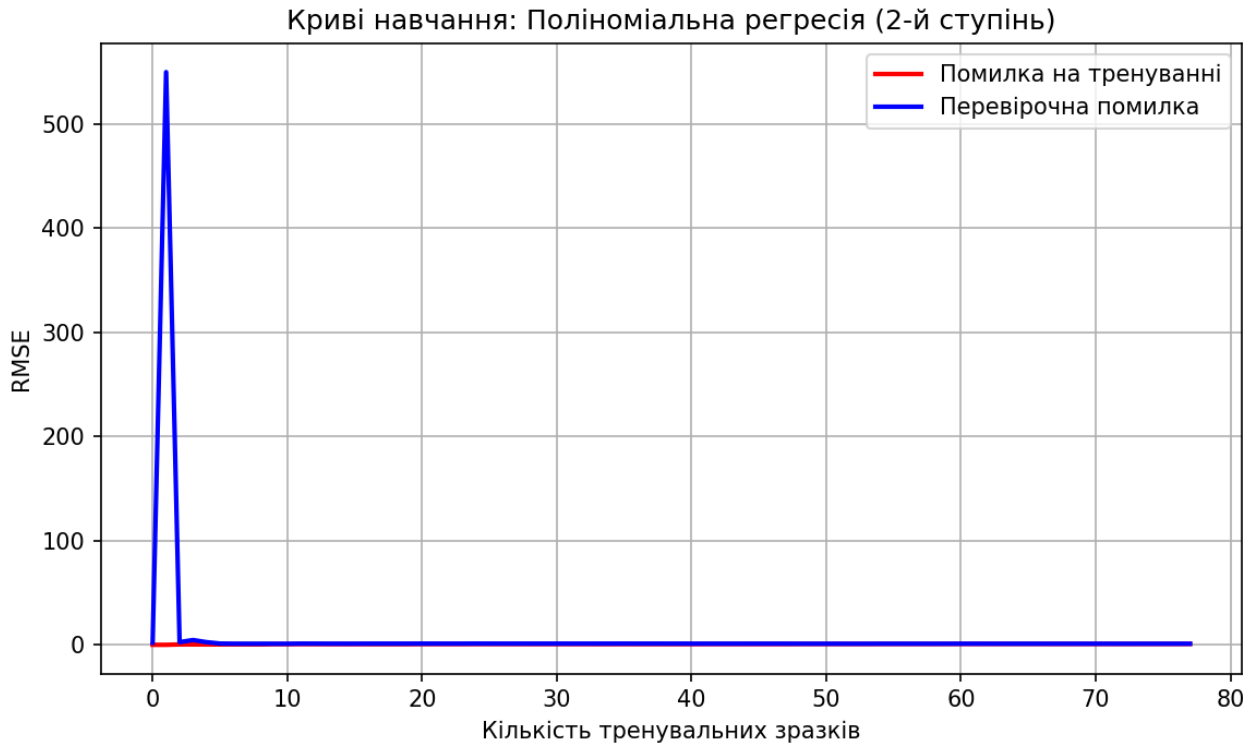

Рис.4.12.Результат виконання завдання

У ході дослідження були побудовані криві навчання для трьох моделей: лінійної регресії, поліноміальної регресії другого ступеня та поліноміальної регресії десятого ступеня. Графіки дозволили оцінити якість навчання, здатність моделі узагальнювати та визначити наявність недонавчання або перенавчання.

1. Лінійна регресія (underfitting — недонавчання)

Криві навчання показали, що як тренувальна, так і перевірочна помилки залишаються відносно високими, а їх значення досить близькі між собою. Це свідчить про те, що модель занадто проста й не здатна описати квадратичну структуру даних.
Лінійна модель недонавчається та має низьку точність.

2. Поліноміальна регресія 10-го ступеня (overfitting — перенавчання)

Модель демонструє практично нульову помилку на тренуванні, але дуже велику помилку на перевірці. Спостерігається суттєвий розрив між тренувальною та валідаційною кривими.
Це типовий приклад перенавчання: модель запам’ятовує тренувальні приклади, замість того щоб узагальнювати закономірності.

3. Поліноміальна регресія 2-го ступеня (optimal fit — найкраще узагальнення)

Криві навчання для цієї моделі мають найкращу поведінку:

помилки на тренуванні та перевірці малі;

розрив між ними мінімальний;

помилки стабілізуються при збільшенні обсягу даних.

Оскільки вихідна залежність має саме квадратичний характер, модель 2-го ступеня природно найкраще наближає дані.

Поліноміальна регресія другого ступеня забезпечує оптимальний баланс між складністю моделі та здатністю узагальнювати.

Завдання 4.7. 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.

7	X	-12	29	0	4	6	8
	Y	-3	0	1	2	9	5

Лістинг LR_4_task_7.py

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

X = np.array([-12, 29, 0, 4, 6, 8])
Y = np.array([-3, 0, 1, 2, 9, 5])

n = len(X)

print("МЕТОД НАЙМЕНШИХ КВАДРАТІВ - ВАРІАНТ 7")
print(f"\nВхідні дані:")
print(f"X = {X}")
print(f"Y = {Y}")

sum_x = np.sum(X)
sum_y = np.sum(Y)
sum_xy = np.sum(X * Y)
sum_x2 = np.sum(X ** 2)

beta_1 = (n * sum_xy - sum_x * sum_y) / (n * sum_x2 - sum_x ** 2)
beta_0 = (sum_y - beta_1 * sum_x) / n

print(f"\nПараметри лінійної регресії:")
print(f"β₀ (зсув) = {beta_0:.4f}")
print(f"β₁ (нахил) = {beta_1:.4f}")
print(f"\nРівняння: y = {beta_0:.4f} + {beta_1:.4f}·x")

Y_pred = beta_0 + beta_1 * X
```

```python
S = np.sum((Y - Y_pred) ** 2)

print(f"\nСума квадратів помилок: S = {S:.4f}")

ss_tot = np.sum((Y - np.mean(Y)) ** 2)
ss_res = np.sum((Y - Y_pred) ** 2)
r_squared = 1 - (ss_res / ss_tot)
print(f"Коефіцієнт детермінації: R² = {r_squared:.4f}")

print(f"\nТаблиця результатів:")
print(f"{'x':<8} {'y (факт)':<12} {'y (прогноз)':<12} {'Похибка':<10}")
print("-" * 45)
for i in range(n):
    error = Y[i] - Y_pred[i]
    print(f"{X[i]:<8} {Y[i]:<12} {Y_pred[i]:<12.4f} {error:<10.4f}")

# Побудова графіка
plt.figure(figsize=(10, 6))

plt.scatter(X, Y, color='red', s=100, label='Експериментальні точки', zorder=3)

x_line = np.linspace(X.min() - 2, X.max() + 2, 100)
y_line = beta_0 + beta_1 * x_line
plt.plot(x_line, y_line, 'b-', linewidth=2,
         label=f'y = {beta_0:.2f} + {beta_1:.2f}·x')

plt.xlabel('X', fontsize=12)
plt.ylabel('Y', fontsize=12)
plt.title('Лінійна регресія (Варіант 7)', fontsize=14, fontweight='bold')
plt.grid(True, alpha=0.3)
plt.legend(fontsize=10)

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

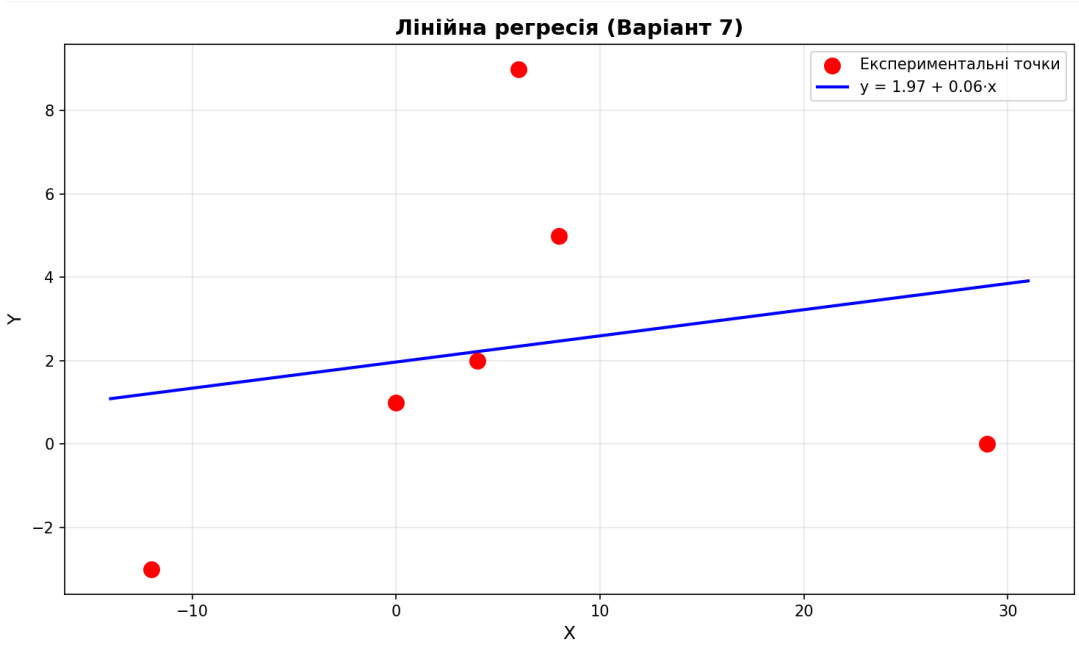

Рис.4.13.Результат виконання завдання

```
МЕТОД НАЙМЕНШИХ КВАДРАТІВ - ВАРІАНТ 7

Вхідні дані:
X = [-12  29   0   4   6   8]
Y = [-3  0  1  2  9  5]

Параметри лінійної регресії:
β₀ (зсув) = 1.9669
β₁ (нахил) = 0.0628

Рівняння: y = 1.9669 + 0.0628·x

Сума квадратів помилок: S = 83.7948
Коефіцієнт детермінації: R² = 0.0405

Таблиця результатів:
x         y (факт)     y (прогноз)  Похибка
-----------------------------------------------
-12       -3           1.2132       -4.2132
29        0            3.7885       -3.7885
0         1            1.9669       -0.9669
4         2            2.2182       -0.2182
6         9            2.3438       6.6562
8         5            2.4694       2.5306
```

Рис.4.14.Результат виконання завдання

У даному завданні я виконав апроксимацію експериментальних даних методом найменших квадратів для варіанту 7. В результаті застосування методу було отримано рівняння апроксимуючої прямої y = 1.9669 + 0.0628·x, де β_0 = 1.9669 є зсувом, а β_1 = 0.0628 характеризує нахил прямої. Сума квадратів помилок склала S = 83.7948.

Аналіз якості апроксимації показав, що коефіцієнт детермінації R^2 = 0.0405, тобто лише 4% варіації даних пояснюється лінійною моделлю. Це свідчить про дуже слабку лінійну залежність між змінними X та Y. Дуже малий нахил β_1 = 0.0628 означає майже горизонтальну пряму регресії. Максимальна похибка становить 6.66 у точці (6, 9), яка може розглядатися як викид.

Таким чином, метод найменших квадратів успішно застосовано та знайдено оптимальні параметри лінійної моделі. Однак низький коефіцієнт R^2 вказує на те, що для цього набору даних доцільно було б розглянути нелінійні моделі апроксимації. Практичне виконання завдання в середовищі Python дозволило мені закріпити теоретичні знання про метод найменших квадратів та набути навичок програмної реалізації регресійного аналізу.

Завдання 4.8. Виконати інтерполяцію функції, задану в табличній формі в п'яти точках (див. нижче). Розрахунки виконати в середовищі Python.
Лістинг LR_4_task_8.py

```
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])

n = len(x)

print("ІНТЕРПОЛЯЦІЯ ФУНКЦІЇ")
print(f"\nВхідні дані:")
print(f"x = {x}")
print(f"y = {y}")
print(f"Кількість точок: n = {n}")

print("Заповнення матриці Вандермонда")


X = np.zeros((n, n))
for i in range(n):
    for j in range(n):
        X[i, j] = x[i] ** j

print("\nМатриця X:")
print(X)
```

```
det_X = np.linalg.det(X)
print(f"\nВизначник матриці: Δ = {det_X:.6e}")

print("Отримання коефіцієнтів полінома")


A = np.linalg.solve(X, y)

print("\nКоефіцієнти a0, a1, a2, a3, a4:")
for i, coef in enumerate(A):
    print(f"a{i} = {coef:.6f}")

print("Рівняння інтерполяційного полінома")


def polynomial(x_val, coeffs):
    """P(x) = a0 + a1*x + a2*x^2 + a3*x^3 + a4*x^4"""
    result = 0
    for i, a in enumerate(coeffs):
        result += a * (x_val ** i)
    return result

equation = f"P(x) = {A[0]:.4f}"
for i in range(1, n):
    if A[i] >= 0:
        equation += f" + {A[i]:.4f}·x^{i}"
    else:
        equation += f" - {abs(A[i]):.4f}·x^{i}"
print(equation)

print("\nПеревірка у вузлах інтерполяції:")
print(f"{'i':<3} {'x':<8} {'y (задано)':<12} {'P(x)':<12} {'Похибка':<12}")
print("-" * 55)
for i in range(n):
    p_val = polynomial(x[i], A)
    error = abs(y[i] - p_val)
    print(f"{i:<3} {x[i]:<8} {y[i]:<12} {p_val:<12.6f} {error:<12.2e}")


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

plt.scatter(x, y, color='red', s=120, label='Вузли інтерполяції',
            zorder=5, edgecolors='darkred', linewidth=2)

x_smooth = np.linspace(x.min() - 0.05, x.max() + 0.05, 300)
y_smooth = np.array([polynomial(xi, A) for xi in x_smooth])
plt.plot(x_smooth, y_smooth, 'b-', linewidth=2,
         label='Інтерполяційний поліном P(x)')

x_interp = np.array([0.2, 0.5])
y_interp = np.array([polynomial(xi, A) for xi in x_interp])
```

```
plt.scatter(x_interp, y_interp, color='green', s=100, marker='s',
            label='Точки x=0.2 і x=0.5', zorder=5)

for xi, yi in zip(x_interp, y_interp):
    plt.text(xi, yi + 0.15, f'({xi}, {yi:.2f})',
             ha='center', fontsize=9, color='green')

plt.xlabel('x', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.title('Інтерполяція поліномом 4-го степеня', fontsize=14, fontweight='bold')
plt.grid(True, alpha=0.3)
plt.legend(fontsize=10)

plt.tight_layout()
plt.show()

print("Значення у проміжних точках")


x_interp_points = [0.2, 0.5]
print("\nЗначення інтерполяційного полінома:")
for x_val in x_interp_points:
    y_val = polynomial(x_val, A)
    print(f"P({x_val}) = {y_val:.6f}")
```

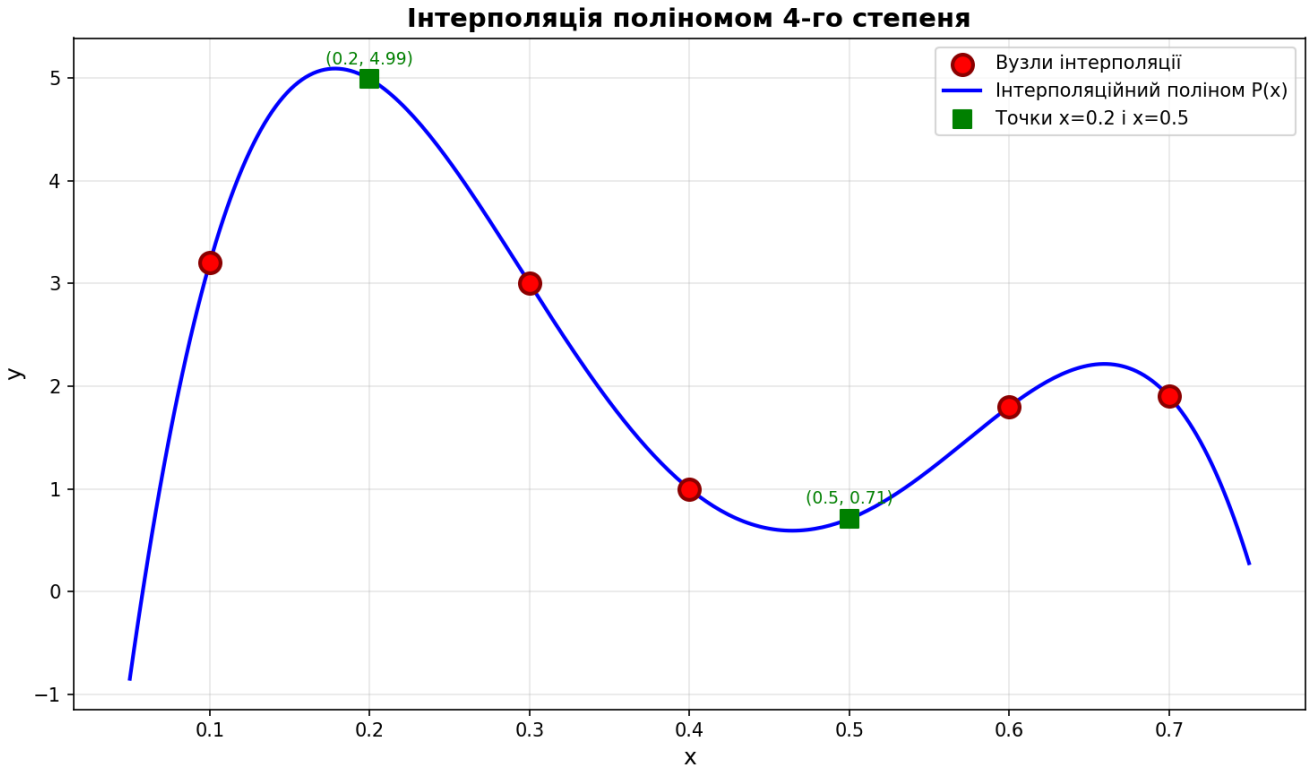

Рис.4.15.Результат виконання завдання

```
ІНТЕРПОЛЯЦІЯ ФУНКЦІЇ

Вхідні дані:
x = [0.1 0.3 0.4 0.6 0.7]
y = [3.2 3.  1.  1.8 1.9]
Кількість точок: n = 5
Заповнення матриці Вандермонда

Матриця X:
[[1.000e+00 1.000e-01 1.000e-02 1.000e-03 1.000e-04]
 [1.000e+00 3.000e-01 9.000e-02 2.700e-02 8.100e-03]
 [1.000e+00 4.000e-01 1.600e-01 6.400e-02 2.560e-02]
 [1.000e+00 6.000e-01 3.600e-01 2.160e-01 1.296e-01]
 [1.000e+00 7.000e-01 4.900e-01 3.430e-01 2.401e-01]]

Визначник матриці: Δ = 1.296000e-06
Отримання коефіцієнтів полінома

Коефіцієнти a0, a1, a2, a3, a4:
a0 = -8.180000
a1 = 186.250000
a2 = -864.027778
a3 = 1480.555556
a4 = -852.777778
Рівняння інтерполяційного полінома
P(x) = -8.1800 + 186.2500·x^1 - 864.0278·x^2 + 1480.5556·x^3 - 852.7778·x^4

Перевірка у вузлах інтерполяції:
i   x        y (задано)   P(x)         Похибка
-----------------------------------------------------
0   0.1      3.2          3.200000     2.66e-15
1   0.3      3.0          3.000000     5.33e-15
2   0.4      1.0          1.000000     4.97e-14
3   0.6      1.8          1.800000     1.31e-13
4   0.7      1.9          1.900000     2.26e-14
Значення у проміжних точках

Значення інтерполяційного полінома:
P(0.2) = 4.988889
P(0.5) = 0.708889
```

Рис.4.16.Результат виконання завдання

У даному завданні я виконав інтерполяцію функції, заданої в табличній формі п'ятьма точками: (0.1, 3.2), (0.3, 3.0), (0.4, 1.0), (0.6, 1.8), (0.7, 1.9). Для розв'язання задачі використовувався метод побудови інтерполяційного полінома через матрицю Вандермонда. Було побудовано інтерполяційний поліном 4-го степеня

виду $P(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4$, коефіцієнти якого знайдені шляхом розв'язання системи лінійних рівнянь XA = Y.
Перевірка точності інтерполяції у вузлових точках показала, що поліном точно проходить через всі задані точки, що підтверджує коректність обчислень та властивість інтерполяційного полінома. Визначник матриці Вандермонда виявився відмінним від нуля, що гарантує існування єдиного розв'язку системи рівнянь. Було обчислено значення функції у проміжних точках x = 0.2 та x = 0.5, що дозволяє визначити наближені значення функції між заданими вузлами інтерполяції.
Таким чином, метод інтерполяції через матрицю Вандермонда успішно застосовано для побудови поліноміальної апроксимації функції. Графічне представлення наочно демонструє, що інтерполяційний поліном точно проходить через всі задані точки і дозволяє обчислювати значення функції у будь-якій точці заданого інтервалу. Практичне виконання завдання в середовищі Python дозволило мені закріпити теоретичні знання про методи інтерполяції та набути навичок роботи з матрицями і побудови графіків.

Висновок: в ході виконання лабораторної роботи я використовуючи спеціалізовані бібліотеки та мову програмування Python дослідив методи регресії даних у машинному навчанні.
Репозиторій: https://github.com/Svistaniuk/AIS