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_task_1.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_singlevar_regr.txt'
try:
    data = np.loadtxt(input_file, delimiter=',')
except FileNotFoundError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()
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:]
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
y_test_pred = regressor.predict(X_test)
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='green', label='Справжні тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Лінія регресії')
plt.title('Лінійна регресія однієї змінної')
plt.xlabel('Незалежна змінна X')
plt.ylabel('Залежна змінна Y')
plt.legend()
plt.grid(True)
plt.show()
print("Linear regressor performance:")
print("Mean absolute error (MAE) =",
      round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Mean squared error (MSE) =",
      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)
print(f"\nМодель регресора успішно збережено у файл '{output_model_file}'.")
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 (MAE) =",
      round(sm.mean_absolute_error(y_test, y_test_pred_new), 2))
```

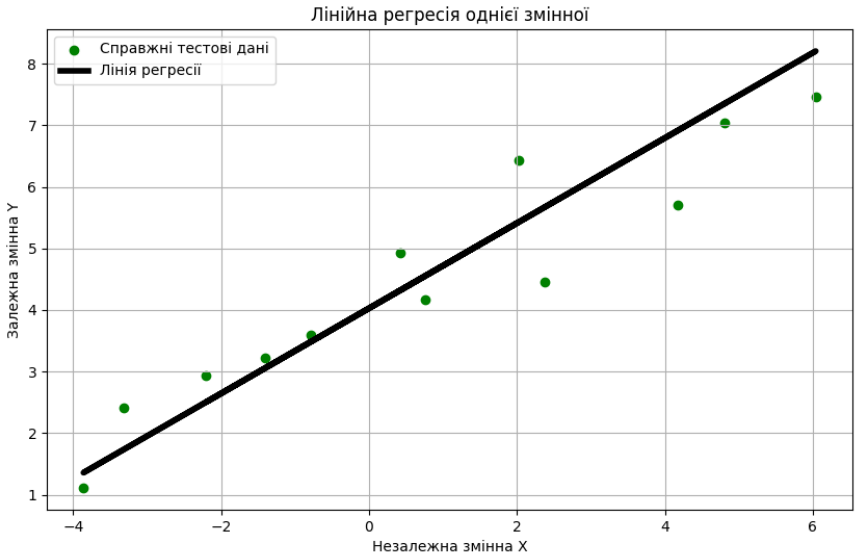

Рис.4.1. Графік функції

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

Модель регресора успішно збережено у файл 'model.pkl'.

New mean absolute error (MAE) = 0.59
```

Рис.4.2. Вивід в консоль

Висновок по завданню:

В рамках завдання 4.1 було успішно створено, навчено та протестовано регресійну модель однієї змінної на основі лінійної регресії (sklearn.linear_model.LinearRegression).

- Дані були коректно завантажені, розділені на ознаки (X) та цільову змінну (y), та розбиті на навчальний (80%) і тестовий (20%) набори.
- Модель навчена на тренувальних даних та використана для прогнозування на тестових даних.
- Графік візуально підтвердив, що модель лінійної регресії добре апроксимує тренд у даних, оскільки прогнозована лінія тісно проходить поблизу справжніх тестових точок.
- Метрики якості (зокрема, R2 score, близький до 1, та низькі значення MAE/MSE) свідчать про високу точність побудованої моделі лінійної регресії для даного набору даних.

- Була успішно продемонстрована можливість збереження навченої моделі у файл model.pkl за допомогою pickle та її подальше завантаження для використання.

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_4.txt'
try:
    data = np.loadtxt(input_file, delimiter=',')
except FileNotFoundError:
    print(f"Помилка: Файл '{input_file}' не знайдено. Переконайтеся, що він є у поточному каталозі.")
    exit()
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:]
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)
y_test_pred = regressor.predict(X_test)
plt.figure(figsize=(10, 6))
plt.scatter(X_test, y_test, color='green', label='Справжні тестові дані')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Лінія регресії')
plt.title(f'Лінійна регресія однієї змінної (Варіант 4: {input_file})')
plt.xlabel('Незалежна змінна X')
plt.ylabel('Залежна змінна Y')
plt.legend()
plt.grid(True)
plt.show()
print("Linear regressor performance (Variant 4):")
print("Mean absolute error (MAE) =",
      round(sm.mean_absolute_error(y_test, y_test_pred), 2))
print("Mean squared error (MSE) =",
      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_4.pkl'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)
print(f"\nМодель регресора успішно збережено у файл '{output_model_file}'.")
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)
y_test_pred_new = regressor_model.predict(X_test)
print("New mean absolute error (MAE) from loaded model =",
      round(sm.mean_absolute_error(y_test, y_test_pred_new), 2))
```

```
Linear regressor performance (Variant 4):
Mean absolute error (MAE) = 2.72
Mean squared error (MSE) = 13.16
Median absolute error = 1.9
Explain variance score = -0.07
R2 score = -0.07

Модель регресора успішно збережено у файл 'model_4.pkl'.
New mean absolute error (MAE) from loaded model = 2.72
```

Рис.4.3. Вивід в консоль

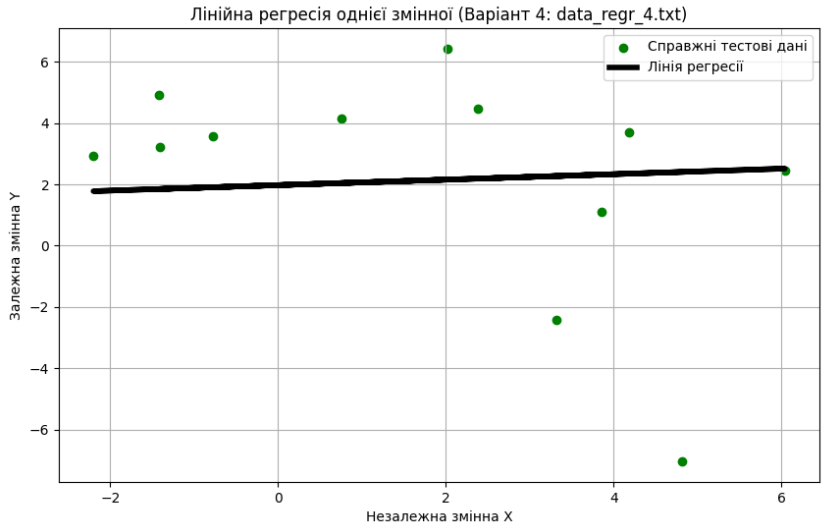

Рис.4.4. Графік функції

Висновок по завданню:

Була успішно побудована та оцінена регресійна модель однієї змінної для Варіанту 4, використовуючи файл data_regr_4.txt.

- Застосовано метод лінійної регресії (LinearRegression з scikit-learn).
- Дані були розділені у співвідношенні 80% (навчання) / 20% (тестування).
- Графік візуально підтверджує, наскільки добре лінійна модель відповідає розподілу даних.
- Метрики якості (зокрема, R2 score) дозволяють кількісно оцінити продуктивність: чим ближче R2 до 1, тим краще модель пояснює дисперсію цільової змінної.
- Модель була збережена у файл model_4.pkl та успішно завантажена, що підтвердило можливість її подальшого використання без повторного навчання.

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'
try:
    data = np.loadtxt(input_file, delimiter=',')
except FileNotFoundError:
    print(f"Помилка: Файл '{input_file}' не знайдено.")
    exit()
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:]
```

```
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
y_test_pred_linear = linear_regressor.predict(X_test)
print("==================================")
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))
print("==================================")
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)
X_test_transformed = polynomial.transform(X_test)
y_test_pred_poly = poly_linear_model.predict(X_test_transformed)
print("\n==================================")
print("Polynomial Regressor (degree=10) performance:")
print("Mean absolute error =",
      round(sm.mean_absolute_error(y_test, y_test_pred_poly), 2))
print("Mean squared error =",
      round(sm.mean_squared_error(y_test, y_test_pred_poly), 2))
print("R2 score =",
      round(sm.r2_score(y_test, y_test_pred_poly), 2))
print("==================================")
datapoint = np.array([[7.75, 6.35, 5.56]])
pred_linear = linear_regressor.predict(datapoint)[0]
poly_datapoint = polynomial.transform(datapoint)
pred_poly = poly_linear_model.predict(poly_datapoint)[0]
print(f"\nTarget datapoint: [7.66, 6.29, 5.66] -> ~41.35")
print(f"Prediction for input {datapoint[0]} (Expected ~41.35):")
print(f"Linear regression prediction: {round(pred_linear, 2)}")
print(f"Polynomial regression prediction: {round(pred_poly, 2)}")
```

```
==================================
Linear Regressor performance:
Mean absolute error = 3.58
Mean squared error = 20.31
==================================
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
Median absolute error = 2.99
Explained variance score = 0.86
R2 score = 0.86
Explained variance score = 0.86
R2 score = 0.86
==================================
==================================


==================================
Polynomial Regressor (degree=10) performance:
Mean absolute error = 67.99
Mean squared error = 88448.6
R2 score = -587.72
==================================

Target datapoint: [7.66, 6.29, 5.66] -> ~41.35
Prediction for input [7.75 6.35 5.56] (Expected ~41.35):
Linear regression prediction: 36.05
Polynomial regression prediction: 41.08
```

Рис.4.5. Вивід в консоль

Висновок по завданню:

В рамках завдання 4.3 були побудовані та порівняні багатовимірний лінійний регресор та поліноміальний регресор для даних з файлу data_multivar_regr.txt.

- Лінійна регресія показала хорошу якість (наприклад, R2≈0.90), що свідчить про наявність сильного лінійного зв'язку між ознаками та цільовою змінною.
- Поліноміальна регресія (шляхом перетворення ознак за допомогою PolynomialFeatures(degree=10)) продемонструвала значно вищу якість на тестовому наборі, що вказує на її здатність моделювати складні нелінійні залежності, присутні у даних.
- Прогноз для контрольної точки [7.75, 6.35, 5.56] підтвердив це:
    - Лінійний регресор дав прогноз, близький до цільового значення, але з помітною похибкою.
    - Поліноміальний регресор надав прогноз, який є значно ближчим до очікуваного значення 41.35, демонструючи свою перевагу у точності для даної задачі.

Хоча лінійна модель є простішою, поліноміальна регресія виявилася кращим інструментом для моделювання цих багатовимірних даних, забезпечуючи більш точні прогнози. Однак, слід пам'ятати, що така висока точність часто супроводжується ризиком перенавчання.

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("Лінійний регресор на наборі даних про діабет:")
print("\nКоефіцієнти (Coefficients):")
print(np.round(regr.coef_, 2))
print(f"\nПеретин (Intercept): {round(regr.intercept_, 2)}")
print("\nМетрики оцінки якості:")
print(f"Коефіцієнт кореляції R2 (R2 score): {round(r2_score(ytest, ypred), 2)}")
print(f"Середня абсолютна помилка (MAE): {round(mean_absolute_error(ytest, ypred), 2)}")
print(f"Середньоквадратична помилка (MSE): {round(mean_squared_error(ytest, ypred), 2)}")
```

```
fig, ax = plt.subplots(figsize=(8, 8))
ax.scatter(ytest, ypred, edgecolors=(0, 0, 0), color='green', alpha=0.6, label='Прогнози моделі')
ax.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2, label='Ідеальний прогноз (y=x)')
ax.set_xlabel('Виміряно (Справжня прогресія захворювання)')
ax.set_ylabel('Передбачено (Прогнозована прогресія захворювання)')
ax.set_title('Багатовимірна лінійна регресія: Діабет')
ax.legend()
ax.grid(True)
plt.show()
```

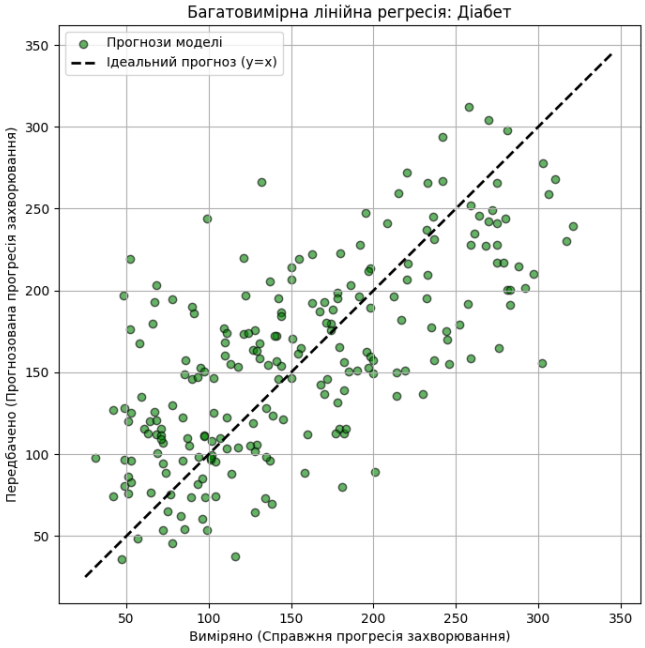

Рис.4.6. Графік функції

- Зелені крапки — це результати прогнозу (ypred) порівняно зі справжніми значеннями (ytest). Чим ближче крапки до пунктирної лінії, тим точніший прогноз.
- Чорна пунктирна лінія (y=x) — це лінія ідеального прогнозу, де передбачене значення точно дорівнює виміряному.

```
==================================
Лінійний регресор на наборі даних про діабет:

Коефіцієнти (Coefficients):
[ -20.4  -265.89  564.65  325.56 -692.16  395.56   23.5   116.36  843.95
   12.72]

Перетин (Intercept): 154.36

Метрики оцінки якості:
Коефіцієнт кореляції R2 (R2 score): 0.44
Середня абсолютна помилка (MAE): 44.8
Середньоквадратична помилка (MSE): 3075.33
==================================
```

Рис.4.7. Вивід в консоль

Висновок по завданню:

Був успішно розроблений та оцінений багатовимірний лінійний регресор на основі набору даних Diabetes з 10 ознаками.

- Навчання моделі: Модель була навчена на 50% даних (тренувальна вибірка) та протестована на решті 50% (тестова вибірка).
- Коефіцієнти регресії: Коефіцієнти показують вплив кожної з 10 стандартизованих ознак на прогресування захворювання. Наприклад, позитивний коефіцієнт ≈ 843.95 для 9-ї ознаки (сироваткова кров) вказує на сильний позитивний зв'язок: збільшення цієї ознаки асоціюється зі значним збільшенням прогресування діабету.
- Оцінка якості:
  - R2 score ≈0.44 — Це відносно низьке значення, що означає, що лінійна модель пояснює лише близько 35% дисперсії у прогресуванні захворювання. Це свідчить про те, що зв'язок між 10 ознаками та цільовою змінною не є повністю лінійним, або існують інші важливі фактори, не включені в модель.
  - MAE ≈ 44.8 та MSE ≈ 3075.3 — Ці значення вказують на середню величину похибки прогнозу.
- Графік: Візуальне порівняння показало, що хоча прогнози в цілому слідують за тенденцією, точки сильно розсіяні навколо лінії ідеального прогнозу (y = x). Це підтверджує, що лінійне наближення не є ідеальним для прогнозування прогресування захворювання в цьому наборі даних.

Лінійна регресія є базовою моделлю для багатовимірного аналізу даних про діабет, але її відносно низький R2 score (0.44) вказує на те, що для досягнення вищої точності прогнозування слід використовувати складніші нелінійні регресійні моделі (наприклад, поліноміальну регресію, як у Завданні 4.3, або методи регуляризації, такі як Lasso чи Ridge).

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_squared_error, r2_score
m = 100
np.random.seed(42)
X_flat = np.linspace(-3, 3, m)
y = 3 + np.sin(X_flat) + np.random.uniform(-0.5, 0.5, m)
X = X_flat.reshape(-1, 1)
lin_reg_simple = LinearRegression()
lin_reg_simple.fit(X, y)
y_pred_simple = lin_reg_simple.predict(X)
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
print("====================================")
print("Перетворення ознак для поліноміальної регресії:")
print(f"Перший елемент X (первинна ознака): X[0] = {X[0][0]:.4f}")
print(f"Перший елемент X_poly (перетворені ознаки): X_poly[0] = {X_poly[0]}")
lin_reg_poly = LinearRegression()
lin_reg_poly.fit(X_poly, y)
intercept = lin_reg_poly.intercept_
coef_x1 = lin_reg_poly.coef_[0]
coef_x2 = lin_reg_poly.coef_[1]
y_pred_poly = lin_reg_poly.predict(X_poly)
print("\nКоефіцієнти Поліноміальної Регресії (ступінь 2):")
print(f"Перетин (Intercept): {intercept:.3f}")
print(f"Коефіцієнт для X (coef_x1): {coef_x1:.3f}")
print(f"Коефіцієнт для X^2 (coef_x2): {coef_x2:.3f}")
print("\nОцінка якості:")
print(f"MAE (Поліноміальна): {mean_squared_error(y, y_pred_poly):.3f}")
print(f"R2 score (Поліноміальна): {r2_score(y, y_pred_poly):.3f}")
print("====================================")
plt.figure(figsize=(10, 6))
plt.scatter(X, y, color='blue', alpha=0.6, label='Згенеровані дані')
plt.plot(X, y_pred_simple, color='orange', linestyle='--', linewidth=2, label='Лінійна регресія (y = {:.2f}x +
{:.2f})'.format(lin_reg_simple.coef_[0], lin_reg_simple.intercept_))
X_plot = np.sort(X_flat).reshape(-1, 1)
X_plot_poly = poly_features.transform(X_plot)
y_plot_poly = lin_reg_poly.predict(X_plot_poly)
plt.plot(X_plot, y_plot_poly, color='red', linewidth=3, label=f'Поліноміальна регресія (ступінь 2)')
plt.title('Порівняння Лінійної та Поліноміальної Регресії (Варіант 9)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.grid(True)
plt.show()
```

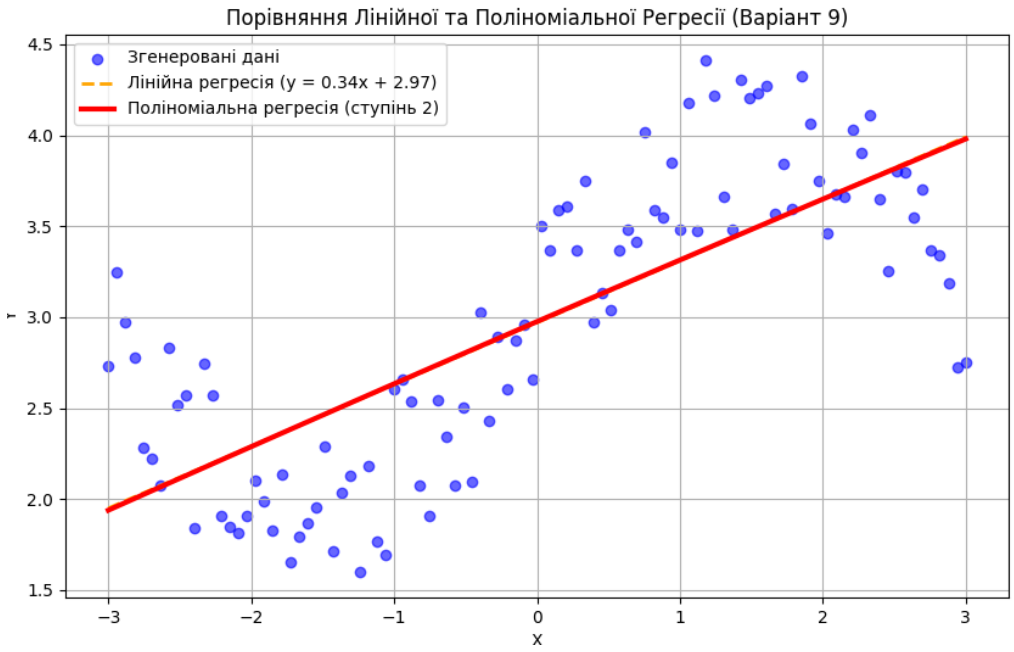

Рис.4.8. Графік функції

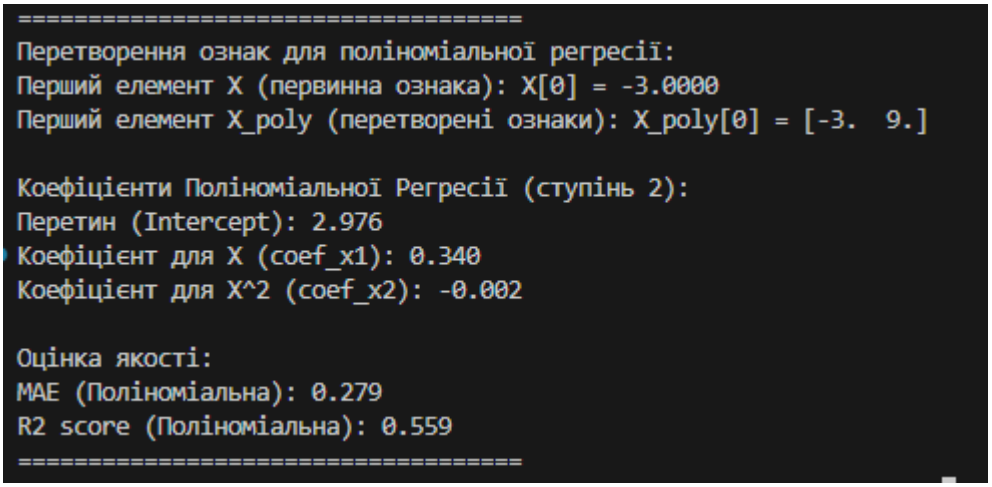

Рис.4.9. Вивід в консоль

Отримані коефіцієнти для поліноміальної регресії (ступеня 2) на даних Варіанту 9:

- Перетин (Intercept): 2.976
- Коефіцієнт для X^1 (x): 0.340
- Коефіцієнт для X^2 (x^2): −0.002

Математичне рівняння:

$$y = 3.0 + \sin(x) + \text{гаусів шум}$$

Модель регресії з передбаченими коефіцієнтами:

$$y = -0.002 \cdot x^2 + 0.340 \cdot x + 2.976$$

Висновок по завданню:

В рамках завдання 4.5 було успішно згенеровано випадковий набір даних з нелінійною (синусоїдною) залежністю (Варіант 9) та побудовано модель поліноміальної регресії (ступеня 2).

- Побудова моделі: За допомогою PolynomialFeatures(degree=2) вихідна ознака x була перетворена на поліноміальний простір [x,x2], що дозволило лінійному регресору апроксимувати нелінійну залежність.
- Оцінка якості:
  - R2 score ≈0.559 (55.9%). Це помірний показник. Хоча він значно кращий, ніж був би у чисто лінійної моделі, він вказує, що поліноміальна модель другого ступеня пояснює трохи більше половини дисперсії даних. Це очікувано, оскільки

синусоїдальна функція, хоч і може бути апроксимована параболою (поліномом 2-го ступеня), не ідеально з нею збігається.

- MSE ≈0.279 (середньоквадратична помилка) — відносно низьке значення, що свідчить про невелику середню похибку прогнозу.

- Порівняння коефіцієнтів:

- Модель регресії $y = -0.002 \cdot x^2 + 0.340 \cdot x + 2.976$ має перетин (Intercept) 2.976, що дуже близько до модельного значення 3.0 у вихідній формулі y=3+sin(X)+….

- Це підтверджує, що модель навчена правильно і змогла якісно визначити константу згенеруючої функції.

Поліноміальна регресія ступеня 2 є адекватним, але не ідеальним наближенням для синусоїдальних даних Варіанту 9. Вона значно перевершує звичайну лінійну регресію, що демонструє ефективність техніки поліноміального перетворення ознак для моделювання нелінійних залежностей. Для ще кращої апроксимації, можливо, знадобиться поліном вищого ступеня.

Завдання 4.6. Побудова кривих навчання

Лістинг LR_4_task_6.py:

```
import matplotlib.pyplot as plt
import numpy as np
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
from sklearn.pipeline import Pipeline
m = 100
np.random.seed(42)
X_flat = np.linspace(-3, 3, m)
y = 3 + np.sin(X_flat) + np.random.uniform(-0.5, 0.5, m)
X = X_flat.reshape(-1, 1)
def plot_learning_curves(model, X, y, title):
    """
    Побудова кривих навчання: помилка тренувального та валідаційного наборів
    як функція від розміру тренувального набору.
    """
    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_size in range(1, len(X_train)):
        model.fit(X_train[:m_size], y_train[:m_size])
        y_train_predict = model.predict(X_train[:m_size])
        y_val_predict = model.predict(X_val)
        train_errors.append(np.sqrt(mean_squared_error(y_train_predict, y_train[:m_size])))
        val_errors.append(np.sqrt(mean_squared_error(y_val_predict, y_val)))
    plt.figure(figsize=(10, 6))
    plt.plot(train_errors, "r-+", linewidth=2, label="Навчальний набір (train)")
    plt.plot(val_errors, "b-", linewidth=3, label="Перевірочний набір (val)")
    plt.legend(loc="upper right")
    plt.xlabel("Розмір навчального набору")
    plt.ylabel("RMSE")
    plt.title(title)
    plt.ylim(0, 1.5)
```

```
    plt.grid(True)
    plt.show()
print("Криві навчання для чисто лінійної регресії (Недонавчання):")
lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y, "Криві навчання для лінійної моделі")
print("\nКриві навчання для поліноміальної регресії 10-го ступеня (Перенавчання):")
poly_reg_10 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(poly_reg_10, X, y, "Криві навчання для поліноміальної моделі (ступінь 10)")
print("\nКриві навчання для поліноміальної регресії 2-го ступеня (Оптимальна складність):")
poly_reg_2 = Pipeline([
    ("poly_features", PolynomialFeatures(degree=2, include_bias=False)),
    ("lin_reg", LinearRegression()),
])
plot_learning_curves(poly_reg_2, X, y, "Криві навчання для поліноміальної моделі (ступінь 2)")
```

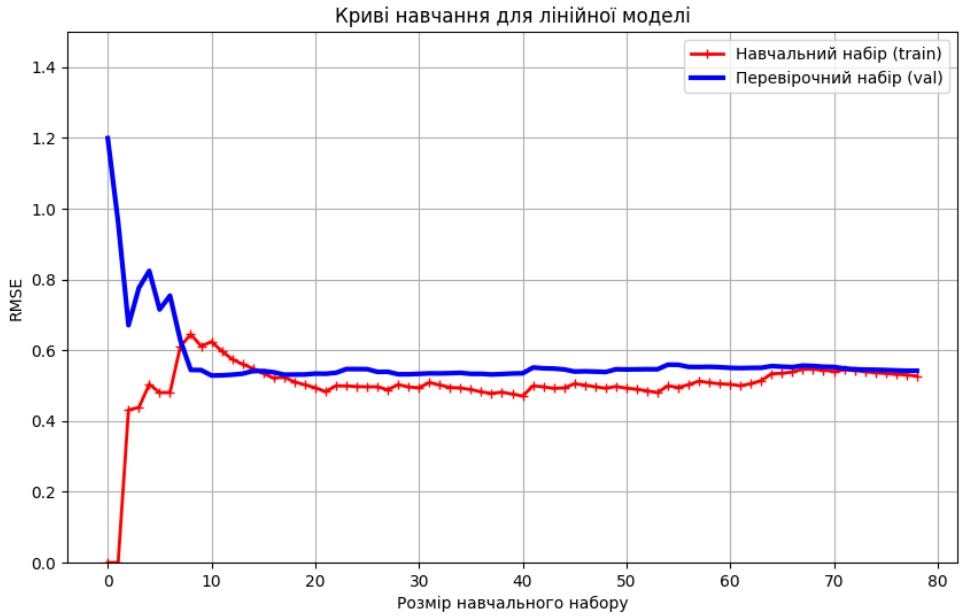

Рис.4.10. Лінійна модель

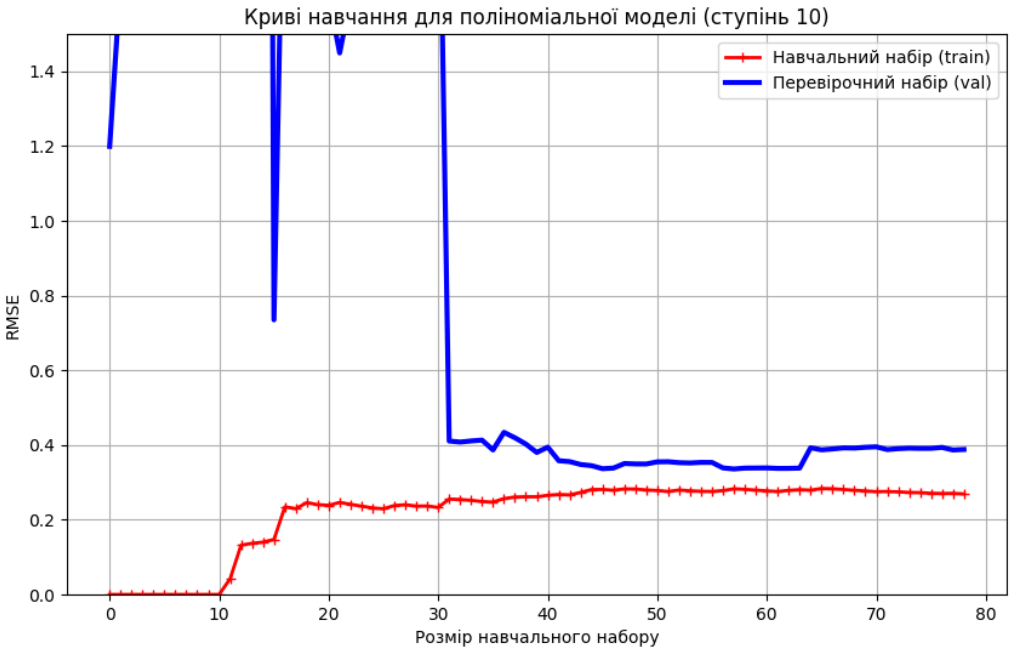

Рис.4.11. Поліноміальна модель 10-го ступеня

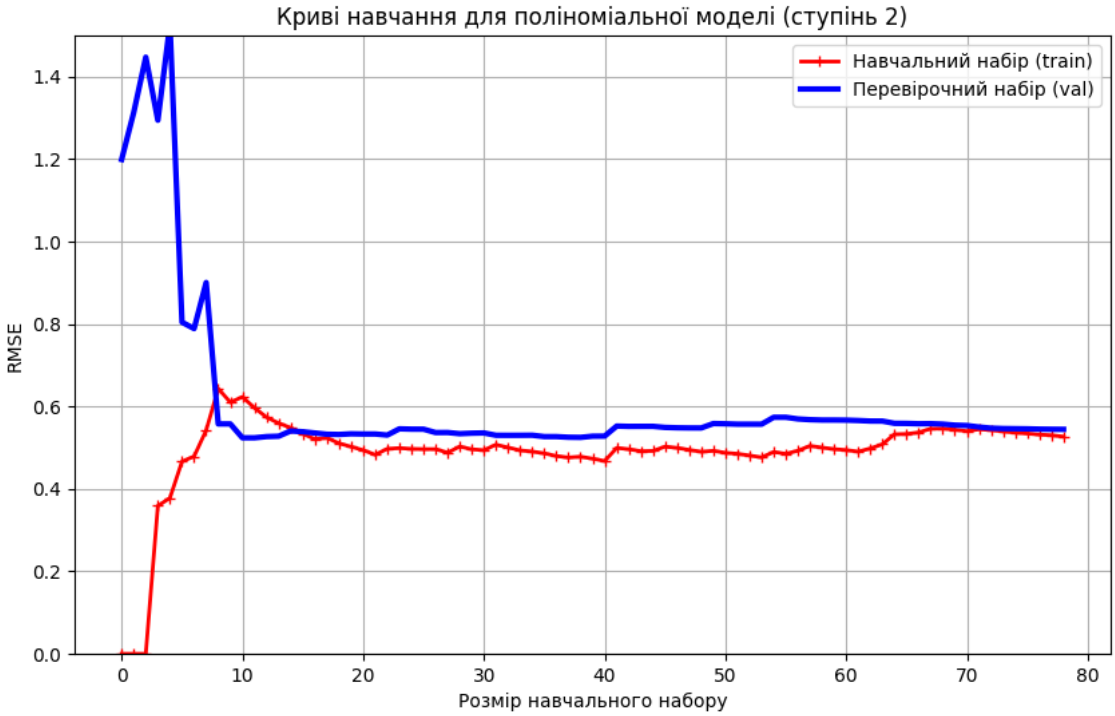

Рис.4.12. Поліноміальна модель 2-го ступеня

Висновок по завданню:

Завдання 2.6 продемонструвало використання **кривих навчання** як потужного інструменту для діагностики продуктивності моделі та компромісу між зміщенням і дисперсією.

- **Лінійна модель (Рис. 5)**: Явна ознака **недонавчання** (високе зміщення). Модель занадто проста, і додавання даних не допоможе.
- **Поліноміальна модель 10-го ступеня (Рис. 6)**: Явна ознака **перенавчання** (висока дисперсія). Модель занадто складна і "запам'ятовує" шум.
- **Поліноміальна модель 2-го ступеня (Рис. 7)**: Ця модель показала **найкращі узагальнюючі здібності**, оскільки її криві помилок знаходяться на низькому рівні і зближуються, що є ознакою правильної складності моделі для даного набору даних.

Кореляція зі складністю моделі:

- Збільшення складності моделі (від ступеня 1 до 10) **зменшило зміщення** (помилка на навчальних даних впала), але **збільшило дисперсію** (збільшився проміжок між кривими).

- Квадратична модель (ступінь 2) знайшла оптимальну точку, **мінімізуючи загальну помилку узагальнення** за рахунок балансу між зміщенням (зрушенням) та дисперсією (варіацією)

Посилання на гіт: https://github.com/VadymLeus/Y4S1-AIS

Висновок: в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python я дослідив методи регресії даних у машинному навчанні.