ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

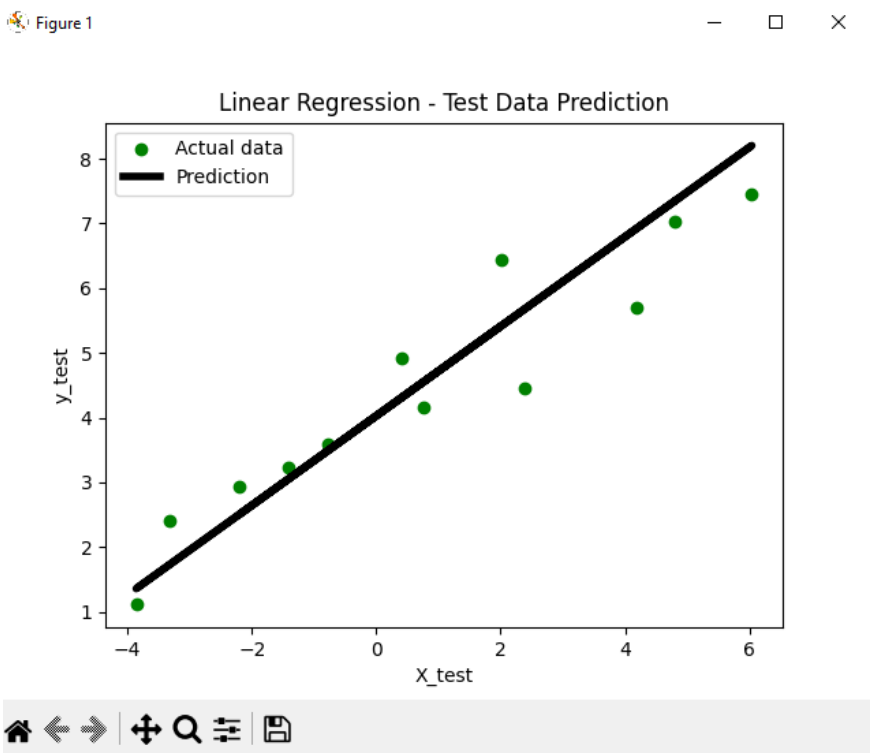
Мета: використовуючи спеціалізовані бібліотеки та мову програмування Python дослідити методи регресії даних у машинному навчанні.

Git: https://github.com/flekXD/SAI

Завдання 1.

```python
# Файл: LR_3_task_1.py

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

# Вхідний файл з даними
input_file = 'data_singlevar_regr.txt'

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

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Тренувальні та тестові дані
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# Створення та навчання моделі лінійної регресії
regressor = linear_model.LinearRegression()
regressor.fit(X_train, y_train)

# Прогнозування результату для тестових даних
y_test_pred = regressor.predict(X_test)

# Побудова графіка
plt.scatter(X_test, y_test, color='green', label='Actual data')
plt.plot(X_test, y_test_pred, color='black', linewidth=4, label='Prediction')
plt.legend()
plt.xlabel("X_test")
plt.ylabel("y_test")
plt.title("Linear Regression - Test Data Prediction")
plt.show()

# Обчислення метрик якості регресії
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))

# Збереження моделі у файл
output_model_file = 'model.pkl'
with open(output_model_file, 'wb') as f:
    pickle.dump(regressor, f)

# Завантаження моделі з файлу
with open(output_model_file, 'rb') as f:
    regressor_model = pickle.load(f)

# Прогнозування результатів з використанням завантаженої моделі
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))
```



```
PS D:\123> python lab4_task_1.py
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

New mean absolute error = 0.59
```

Модель лінійної регресії показала хороші результати:

MAE: 0.59, MSE: 0.49, R^2: 0.86, що свідчить про точність і стабільність.

Після завантаження моделі її продуктивність не змінилася (New MAE: 0.59).

Модель підходить для прогнозування подібних даних.

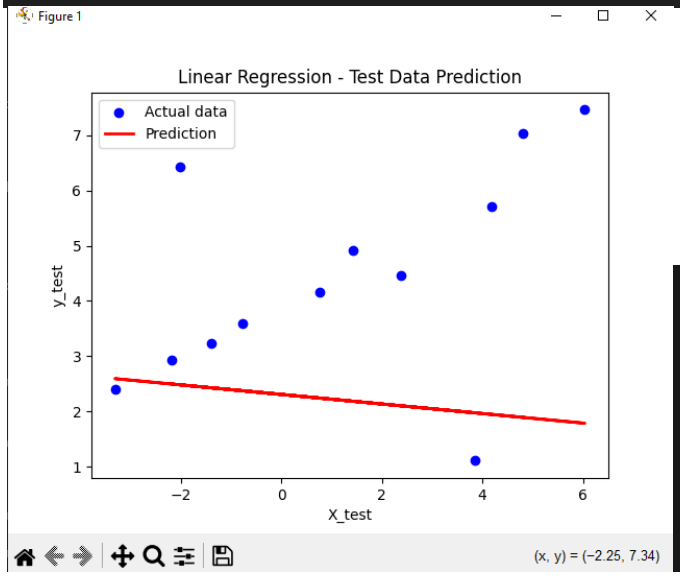
Завдання 2:

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

Варіант 2 файл: data_regr_2.txt

```
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_2.txt'

try:
    data = np.loadtxt(input_file, delimiter=',')
except OSError:
    raise FileNotFoundError(f"Файл {input_file} не знайдено. Переконайтесь, що файл
існує.")

X, y = data[:, :-1], data[:, -1]

# Розбивка даних на навчальний та тестовий набори
num_training = int(0.8 * len(X))
num_test = len(X) - num_training

# Тренувальні та тестові дані
X_train, y_train = X[:num_training], y[:num_training]
X_test, y_test = X[num_training:], y[num_training:]

# Створення та навчання моделі лінійної регресії
regressor = linear_model.LinearRegression()
```

```
regressor.fit(X_train, y_train)

# Прогнозування результату для тестових даних
y_test_pred = regressor.predict(X_test)

# Побудова графіка
plt.scatter(X_test, y_test, color='blue', label='Actual data')
plt.plot(X_test, y_test_pred, color='red', linewidth=2, label='Prediction')
plt.legend()
plt.xlabel("X_test")
plt.ylabel("y_test")
plt.title("Linear Regression - Test Data Prediction")
plt.show()

# Обчислення метрик якості регресії
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_regr_2.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)

# Прогнозування результатів з використанням завантаженої моделі
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))
```




```
PS D:\123> python lab4_task_2.py
Linear regressor performance:
Mean absolute error = 2.42
Mean squared error = 9.02
Median absolute error = 2.14
Explained variance score = -0.15
R2 score = -1.61

New mean absolute error = 2.42
```

Модель показала низьку якість: MAE: 2.42, MSE: 9.02, R^2: -1.61. Це свідчить про погану відповідність даним та низьку якість прогнозування.

Завдання 3:

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

```
# Файл: LR_3_task_3.py

import numpy as np
from sklearn import linear_model
import sklearn.metrics as sm
from sklearn.preprocessing import PolynomialFeatures

# Вхідний файл з даними
input_file = 'data_multivar_regr.txt'

# Завантаження даних
try:
    data = np.loadtxt(input_file, delimiter=',')
except OSError:
    raise FileNotFoundError(f"Файл {input_file} не знайдено. Переконайтесь, що файл
існує.")

X, y = data[:, :-1], data[:, -1]

# Розбивка даних на навчальний та тестовий набори
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:]

# Лінійна регресія
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
y_test_pred = linear_regressor.predict(X_test)

# Метрики для лінійного регресора
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))
```

```python
# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)
X_test_transformed = polynomial.transform(X_test)

poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
y_test_pred_poly = poly_linear_model.predict(X_test_transformed)

# Прогноз для вибіркової точки
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.transform(datapoint)

print("\nLinear regression prediction for datapoint:\n",
linear_regressor.predict(datapoint))
print("\nPolynomial regression prediction for datapoint:\n",
poly_linear_model.predict(poly_datapoint))

# Порівняння метрик для поліноміальної регресії
print("\nPolynomial Regressor 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("Median absolute error =", round(sm.median_absolute_error(y_test,
y_test_pred_poly), 2))
print("Explained variance score =", round(sm.explained_variance_score(y_test,
y_test_pred_poly), 2))
print("R2 score =", round(sm.r2_score(y_test, y_test_pred_poly), 2))
```

```
PS D:\123> python lab4_task_3.py
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

Linear regression prediction for datapoint:
 [36.05286276]

Polynomial regression prediction for datapoint:
 [41.46543157]

Polynomial Regressor performance:
Mean absolute error = 84.26
Mean squared error = 173728.48
Median absolute error = 6.13
Explained variance score = -1153.1
R2 score = -1155.36
```

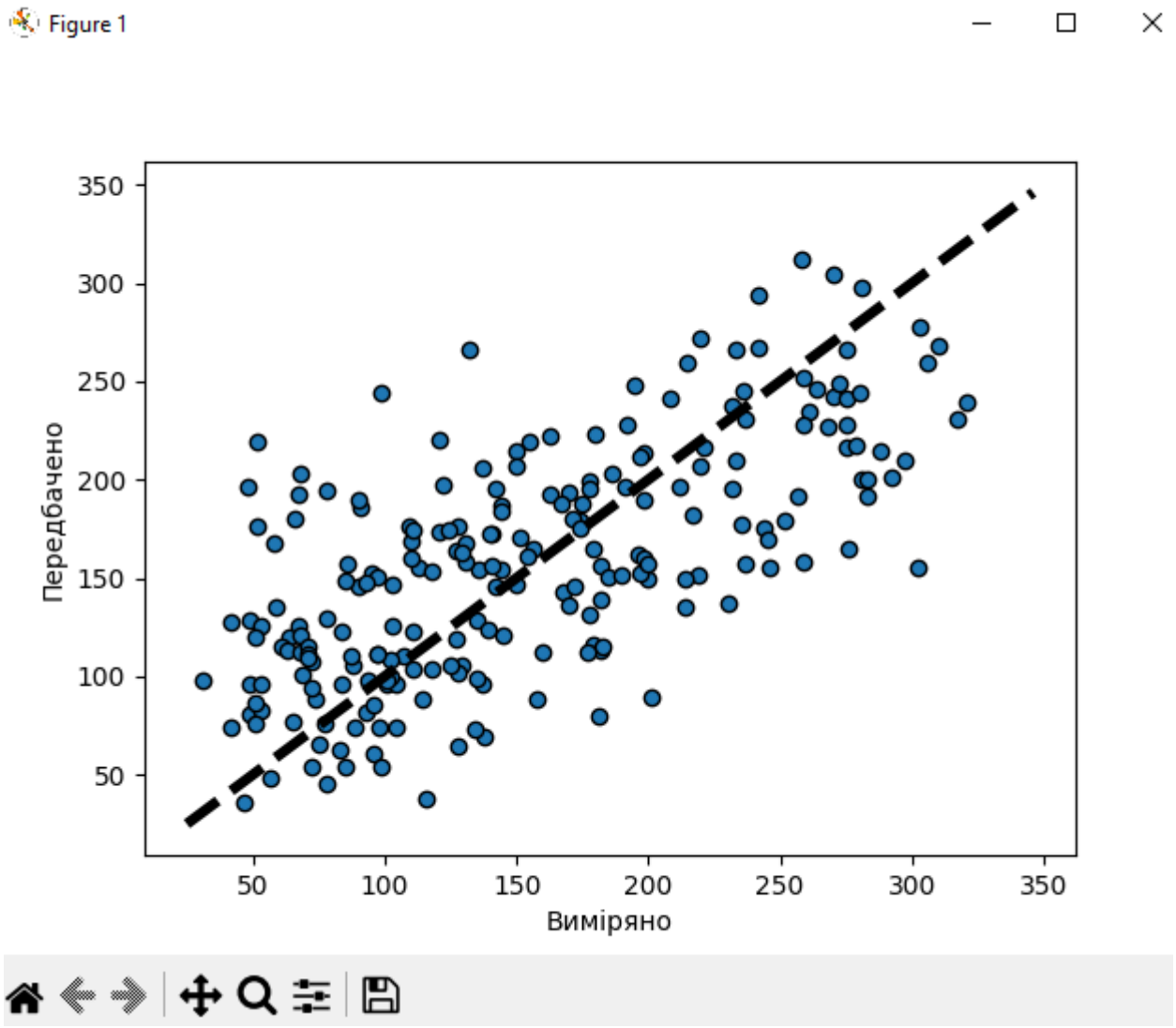
Лінійна регресія показала хороші результати (R^2: 0.86), але прогноз 36.05 трохи відрізняється від очікуваного 41.35. Поліноміальна регресія (R^2: -1155.36) перенавчилася, хоча її прогноз ближчий до 41.35.

Завдання 4:

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

```
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

# Поділяємо дані на навчальну та тестову вибірки
Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.5,
random_state=0)

# Створюємо модель лінійної регресії та натренуємо її
regr = linear_model.LinearRegression()
regr.fit(Xtrain, ytrain)

# Зробимо прогноз по тестовій вибірці
ypred = regr.predict(Xtest)

# Розрахунок коефіцієнтів регресії та різних показників якості
print('Коефіцієнти регресії:', regr.coef_)
print('Перехоплення (intercept):', regr.intercept_)
print('R^2 (коефіцієнт детермінації):', r2_score(ytest, ypred))
print('Середня абсолютна помилка (MAE):', mean_absolute_error(ytest, ypred))
print('Середньоквадратична помилка (MSE):', mean_squared_error(ytest, ypred))

# Побудова графіка
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')
```

```
plt.show()
```




```
PS D:\123> python lab4_task_4.py
Коефіцієнти регресії: [ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Перехоплення (intercept): 154.3589285280134
R^2 (коефіцієнт детермінації): 0.43774971182541
Середня абсолютна помилка (MAE): 44.800645233553276
Середньоквадратична помилка (MSE): 3075.330688680324
```

Модель лінійної регресії дає обмежені результати ($R^2 = 0.44$), що вказує на те, що існують значні невраховані фактори або що лінійна модель не є найкращим підходом для цієї задачі.

Завдання 5:

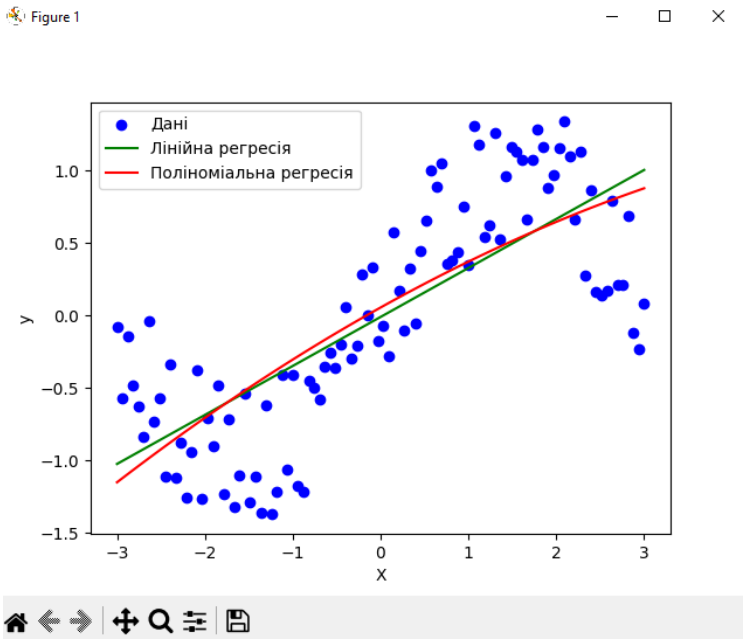
Згенеруйте свої випадкові дані обравши за списком відповідно свій варіант (згідно табл. 2.2) та виведіть їх на графік. Побудуйте по них модель лінійної

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

Варіант 7

```
m = 100
X = np.linspace(-3, 3, m)
y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)
```

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

# Генерація випадкових даних
m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)  # Розтягування вектору X в матрицю
y = np.sin(X).ravel() + np.random.uniform(-0.5, 0.5, m)  # Генерація y

# Побудова графіка даних
plt.scatter(X, y, color='blue', label='Дані')

# Лінійна регресія
lin_reg = LinearRegression()
lin_reg.fit(X, y)
y_lin_pred = lin_reg.predict(X)
plt.plot(X, y_lin_pred, color='green', label='Лінійна регресія')

# Поліноміальна регресія (степінь 2)
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)
poly_reg = LinearRegression()
poly_reg.fit(X_poly, y)
y_poly_pred = poly_reg.predict(X_poly)

# Побудова графіка для поліноміальної регресії
plt.plot(X, y_poly_pred, color='red', label='Поліноміальна регресія')

# Підпис графіків
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()

# Виведення коефіцієнтів
print("Коефіцієнти лінійної регресії:", lin_reg.coef_)
print("Перехоплення лінійної регресії:", lin_reg.intercept_)
print("Коефіцієнти поліноміальної регресії:", poly_reg.coef_)
```




```
PS D:\123> python lab4_task_5.py
Коефіцієнти лінійної регресії: [0.33820117]
Перехоплення лінійної регресії: -0.009869285363685714
Коефіцієнти поліноміальної регресії: [ 0.33820117 -0.02130286]
```

Лінійна регресія: Лінійна модель має вигляд:

Y = 0.3382x-0.0099

де:

0.3382 — коефіцієнт при X (нахил),

-0.0099 — перехоплення (свобода).

Поліноміальна регресія : Поліноміальна регресія має вигляд:

$Y = 0.3382x-0.0213x^2$

де:

0.3382 — коефіцієнт при x

-0.0213 — коефіцієнт при x^2 (для поліноміальної складової).

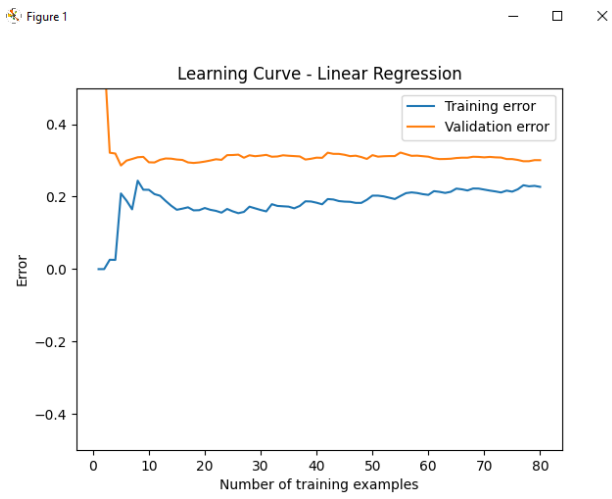
Робота показала важливість вибору правильної моделі для даних з нелінійними залежностями, де лінійна регресія не завжди є найкращим варіантом. Поліноміальна регресія дозволяє отримати більш точне наближення до нелінійних даних.

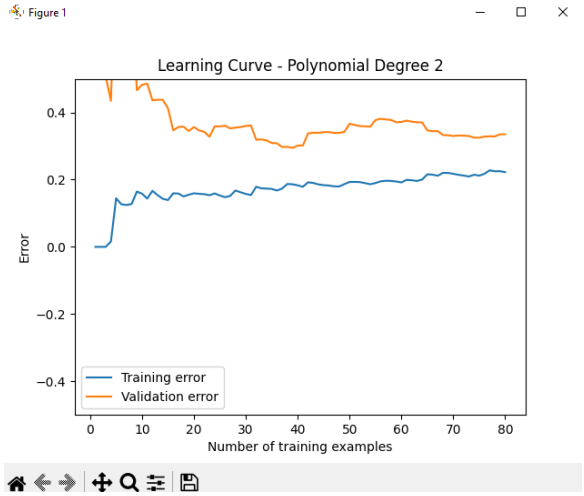
Завдання 6:

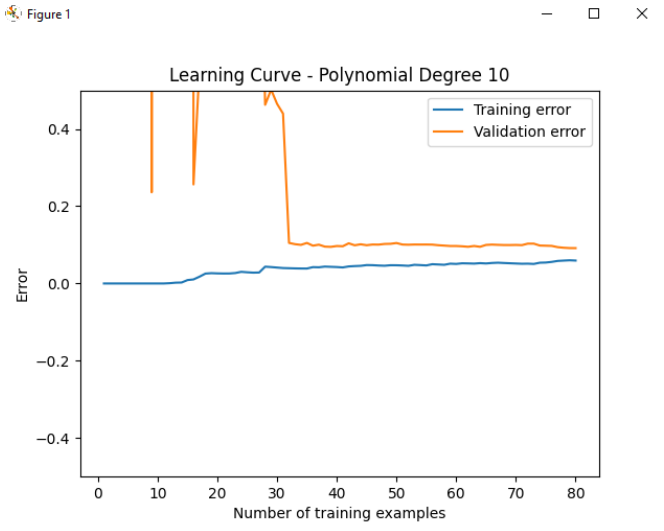
Побудуйте криві навчання для ваших даних у попередньому завданні.

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


m = 100
X = np.linspace(-3, 3, m).reshape(-1, 1)
y = np.sin(X).flatten() + np.random.uniform(-0.5, 0.5, m)

# Розбиття на навчальні та перевірочні дані
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2,
random_state=42)

def plot_learning_curve_pipeline(model, X_train, y_train, X_val, y_val, degree=2,
title="Learning Curve"):
    train_errors, val_errors = [], []
    for m in range(1, len(X_train) + 1):
        X_train_subset, y_train_subset = X_train[:m], y_train[:m]

        # Створюємо Pipeline з перетворенням даних та лінійною регресією
        pipeline = Pipeline([
            ("poly_features", PolynomialFeatures(degree=degree)),
            ("lin_reg", LinearRegression())
        ])

        # Навчаємо модель
        pipeline.fit(X_train_subset, y_train_subset)

        # Оцінка помилок
        train_predictions = pipeline.predict(X_train_subset)
        val_predictions = pipeline.predict(X_val)

        # Обчислення помилок
        train_errors.append(mean_squared_error(y_train_subset, train_predictions))
        val_errors.append(mean_squared_error(y_val, val_predictions))

    # Побудова графіка
    plt.plot(range(1, len(X_train) + 1), train_errors, label="Training error")
    plt.plot(range(1, len(X_train) + 1), val_errors, label="Validation error")
    plt.title(title)
    plt.xlabel("Number of training examples")
    plt.ylabel("Error")
    plt.ylim(-0.5, 0.5)  # Обмеження по осі Y
    plt.legend()
```

```
    plt.show()

# Побудова кривих навчання для лінійної регресії
print("Learning Curve for Linear Regression")
plot_learning_curve_pipeline(LinearRegression(), X_train, y_train, X_val, y_val,
degree=1, title="Learning Curve - Linear Regression")

# Побудова кривих навчання для поліноміальної регресії 2-го ступеня
print("Learning Curve for Polynomial Regression (degree=2)")
plot_learning_curve_pipeline(LinearRegression(), X_train, y_train, X_val, y_val,
degree=2, title="Learning Curve - Polynomial Degree 2")

# Побудова кривих навчання для поліноміальної регресії 10-го ступеня
print("Learning Curve for Polynomial Regression (degree=10)")
plot_learning_curve_pipeline(LinearRegression(), X_train, y_train, X_val, y_val,
degree=10, title="Learning Curve - Polynomial Degree 10")
```







Висновок:

У ході дослідження було вивчено методи регресії, включаючи лінійну та поліноміальну регресію. Лінійна регресія показала недонавчання через свою простоту, тоді як поліноміальна регресія вищих ступенів схильна до перенавчання. Криві навчання продемонстрували компроміс між зміщенням і дисперсією: прості моделі мають високе зміщення, а складні — високу дисперсію. Використання бібліотек Python, зокрема scikit-learn, дозволило ефективно виконати аналіз і візуалізацію. Оптимальний вибір моделі залежить від характеру даних і потреб у точності та узагальненні.