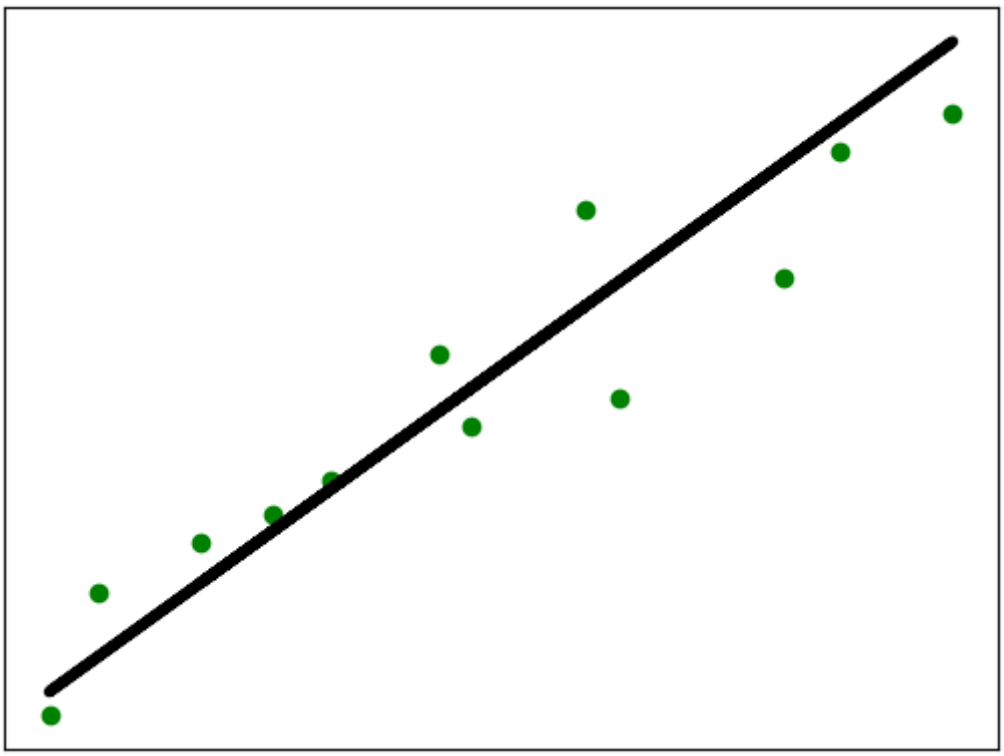
ЛАБОРАТОРНА РОБОТА № 4

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ

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

Завдання 2.1. Створення регресора однієї змінної

Вихідний графік лінійного регресора



Метричні дані регресора

```
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
```

Оцінка якості прогнозу

```
New mean absolute error = 0.59
```

Висновок: Створена модель працює досить добре, оскільки всі значення помилок (MAE, MSE, MedAE) є невеликими. Високі значення EVS (0.86) та R^2 (0.86) свідчать про те, що модель ефективно описує залежності в даних.

Завдання 2.2. Передбачення за допомогою регресії однієї змінної

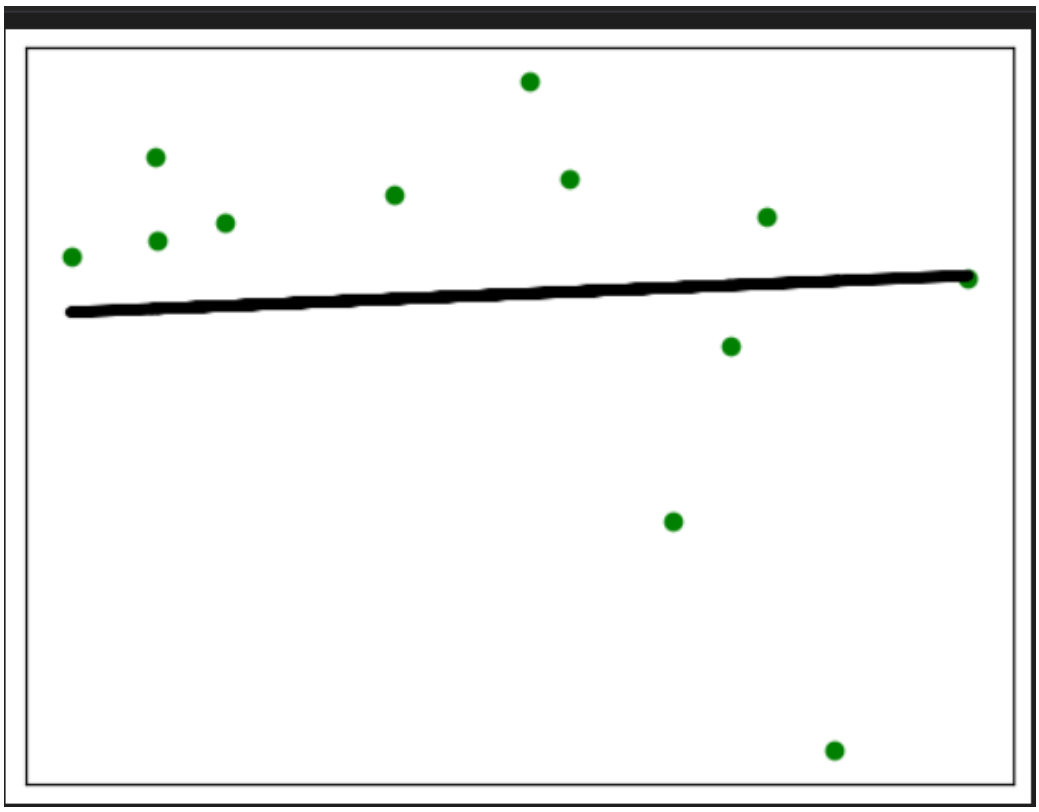
Варіант 4 файл: data_regr_4.txt

Код програми:

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

# Вхідний файл, який містить дані
input_file = 'data_regr_4.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 дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]
# Створення об'єкта лінійного регресора
regressor_model = linear_model.LinearRegression()
regressor_model.fit(X_train, y_train)
# Прогнозування результату
y_test_pred = regressor_model.predict(X_test)
# Побудова графіка
plt.scatter(X_test, y_test, color='green')
plt.plot(X_test, y_test_pred, color='black', linewidth=4)
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))

# %%
# Файл для збереження моделі
output_model_file = 'model2.pkl'

# %%
# Збереження моделі
with open(output_model_file, 'wb') as f:
 pickle.dump(regressor_model, f)

# %%
# Завантаження 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))

#
```

Графік функції лінійного регресора:



Графік розподілу даних показує, що між змінними і немає очевидної лінійної залежності, а дані розподілені хаотично.

Оцінки якості створеного регресора:

```
Linear regressor performance:
Mean absolute error = 2.72
Mean squared error = 13.16
Median absolute error = 1.9
Explain variance score = -0.07
R2 score = -0.07
```

Висновок: Дані не підходять для лінійної регресії, оскільки між змінними немає лінійної залежності. Високі значення MAE та MSE вказують на великі помилки, а низькі значення EVS та R^2 означають, що модель майже не пояснює дані. Модель є слабкою і не відповідає задачі.

Завдання 2.3. Створення багатовимірного регресора

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

# Вхідний файл, який містить дані
input_file = 'data_multivar_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 дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]
# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
# Прогнозування результату
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("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))

# %%
# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=10)
X_train_transformed = polynomial.fit_transform(X_train)

# %%
datapoint = [[7.75, 6.35, 5.56]]
poly_datapoint = polynomial.fit_transform(datapoint)

# %%
poly_linear_model = linear_model.LinearRegression()
poly_linear_model.fit(X_train_transformed, y_train)
print("\nLinear regression:\n",
linear_regressor.predict(datapoint))
print("\nPolynomial regression:\n",
poly_linear_model.predict(poly_datapoint))
```

Метрики якості лінійної регресії:

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

Прогнозування значення в точці [7.66, 6.29, 5.66] обома створеними моделями

```
Linear regression:
 [36.05286276]

Polynomial regression:
 [41.46543157]
```

Висновок: Порівняно з лінійним регресором поліноміальний регресор забезпечує отримання результату, ближчого до значення 41.35. Тобто дає кращі результати.

Завдання 2.4. Регресія багатьох змінних

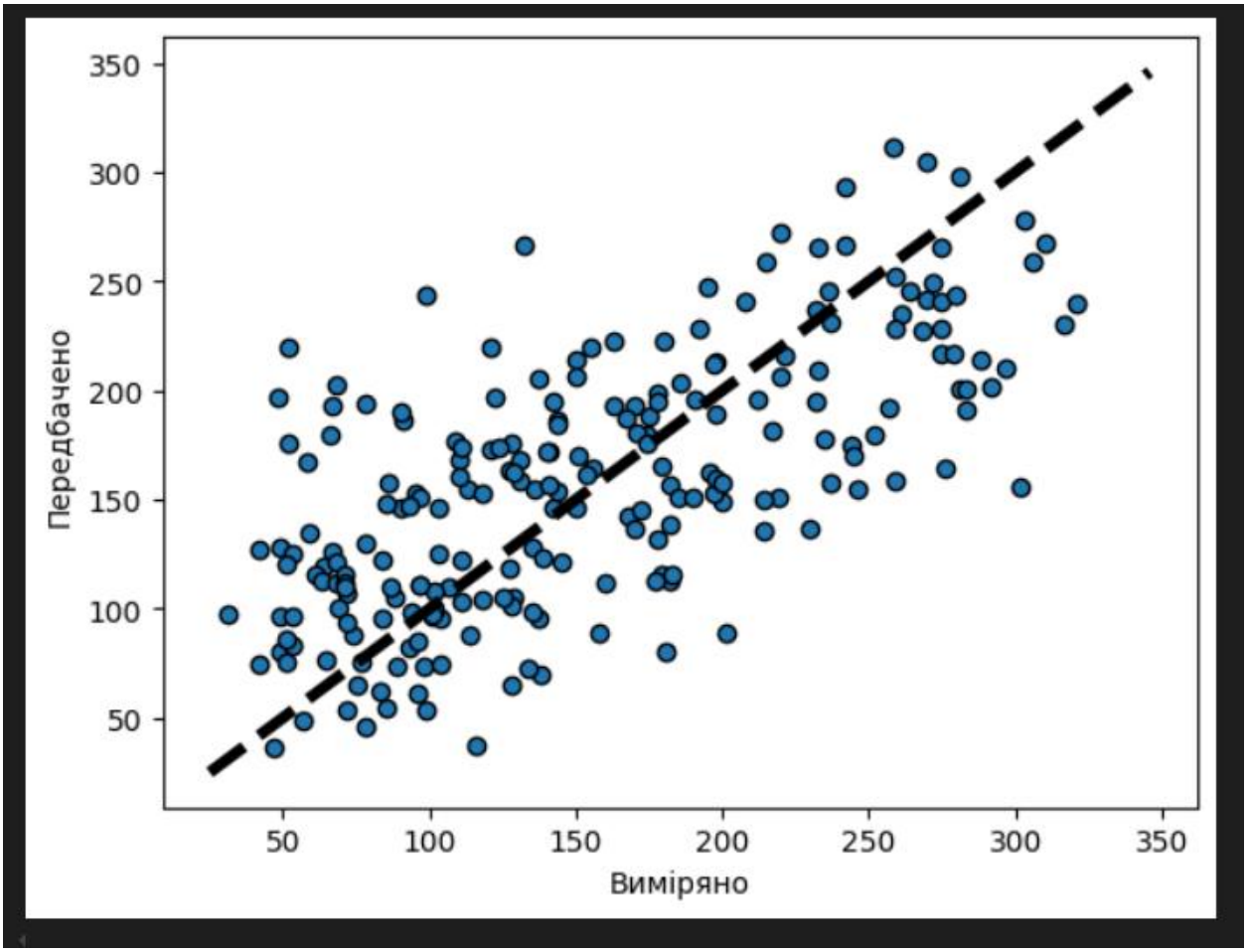
```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import datasets, linear_model
```

```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.metrics import 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("Regressor performance:")
print("Regression Coeficient")
print(regr.coef_)
print("Regression Interceptor")
print(regr.intercept_)
print("Mean absolute error =",
round(mean_absolute_error(ytest, ypred), 2))
print("Mean squared error =",
round(mean_squared_error(ytest, ypred), 2))
print("R2 score =", round(r2_score(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')
plt.show()
```

Коефіцієнти регресії та показники:

```
Regressor performance:
Regression Coeficient
[ -20.4047621  -265.88518066  564.65086437  325.56226865 -692.16120333
  395.55720874   23.49659361  116.36402337  843.94613929   12.71856131]
Regression Interceptor
154.3589285280134
Mean absolute error = 44.8
Mean squared error = 3075.33
R2 score = 0.44
```

Графік:



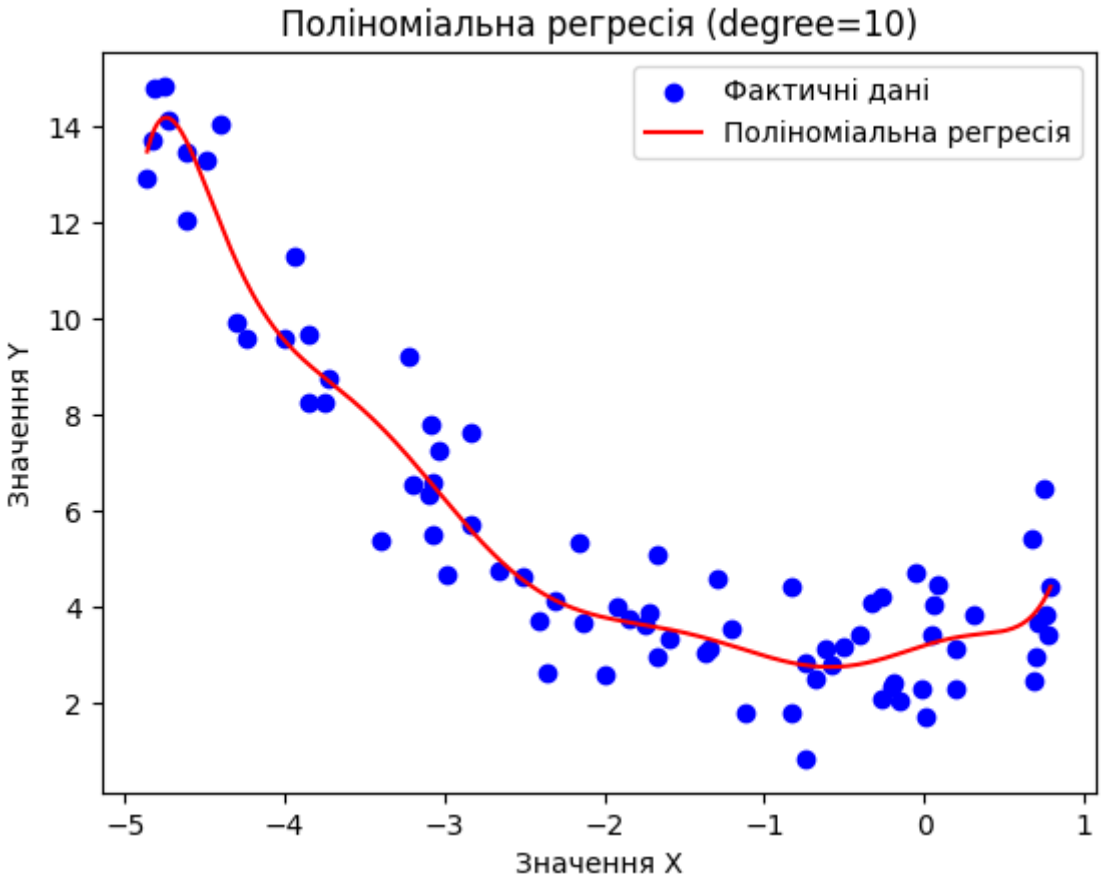
Висновок: побудована модель має середній рівень точності. R^2=0.44 говорить про те, що модель недостатньо добре пояснює варіацію даних. MAE і MSE показують досить великі помилки. З графіка видно, що багато точок розташовані далеко від чорної лінії, що підтверджує досить слабкий R^2.

Завдання 2.5. Самостійна побудова регресії

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

```
# %%
m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)

# %%
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# Тренувальні дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]
# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()
linear_regressor.fit(X_train, y_train)
# Прогнозування результату
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("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))

# %%
# Побудова графіка
plt.scatter(X_test, y_test, color='green')
plt.plot(X_test, y_test_pred, color='black', linewidth=4)
plt.xticks(())
plt.yticks(())
plt.show()

# %%
# Поліноміальна регресія
polynomial = PolynomialFeatures(degree=2)
X_train_transformed = polynomial.fit_transform(X_train)

# %%
model = linear_model.LinearRegression()
model.fit(X_train_transformed, y_train)

X_plot = np.linspace(min(X_train), max(X_train), 500).reshape(-1, 1)  # Генеруємо
точки для побудови кривої
```

```python
X_plot_transformed = polynomial.transform(X_plot)  # Трансформуємо точки у поліноміальну форму
y_plot = model.predict(X_plot_transformed)  # Передбачення моделі


# %%
plt.scatter(X_train, y_train, color="blue", label="Фактичні дані")  # Фактичні дані
plt.plot(X_plot, y_plot, color="red", label="Поліноміальна регресія")  # Крива моделі
plt.xlabel("Значення X")
plt.ylabel("Значення Y")
plt.title("Поліноміальна регресія (degree=10)")
plt.legend()
plt.show()

# %%
print("Regression Coeficient")
print(model.coef_)
print("Regression Interceptor")
print(model.intercept_)
```



```
Regression Coeficient
[[0.         0.97219558 0.67297354]]
Regression Interceptor
[3.10696635]
```

Задана модель у вигляді математичного рівняння

$y = 0.7x^2 + x + 3 + шум$

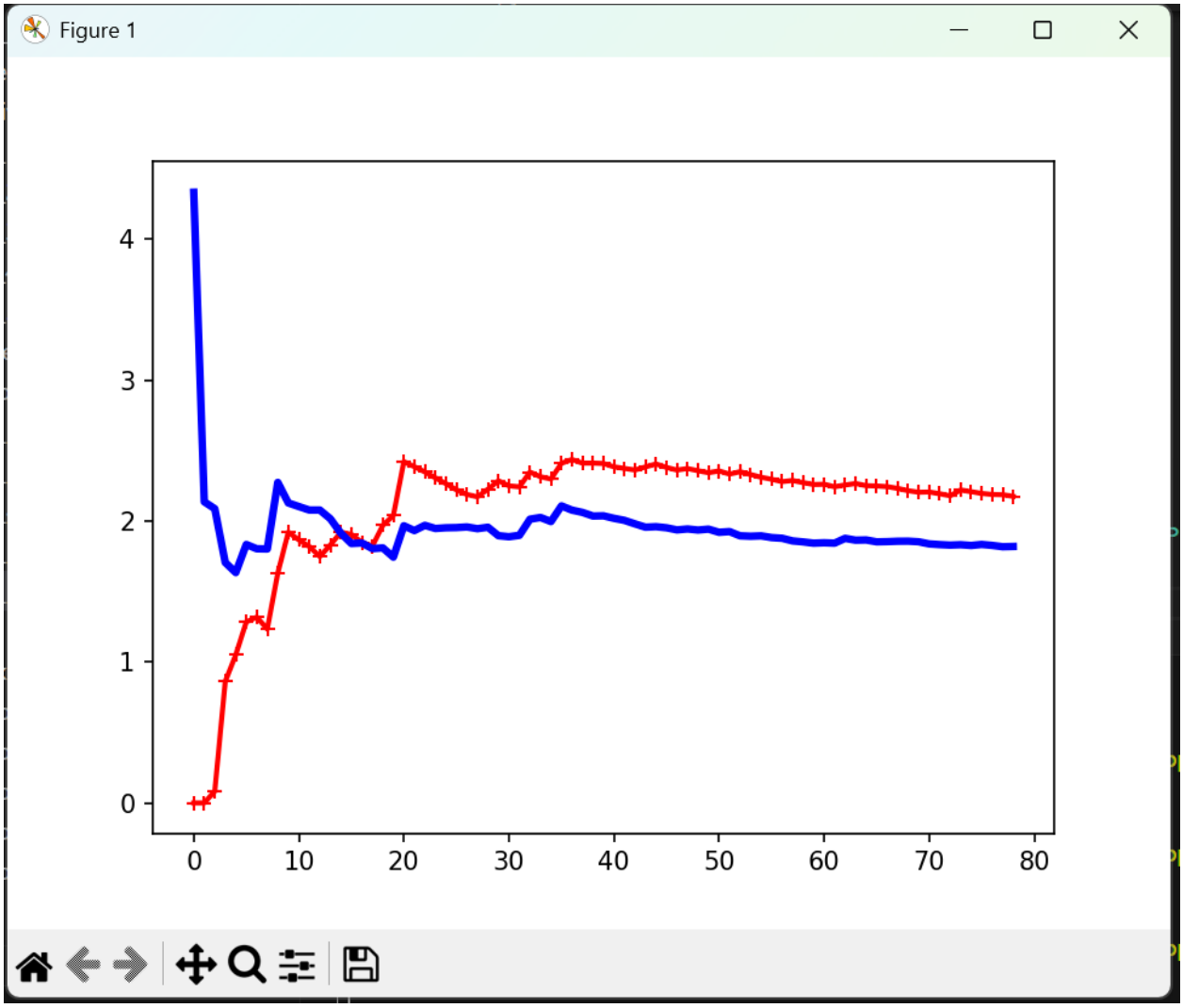
Отримана модель

$y = 0.97x^2 + 0.6x + 3.1$

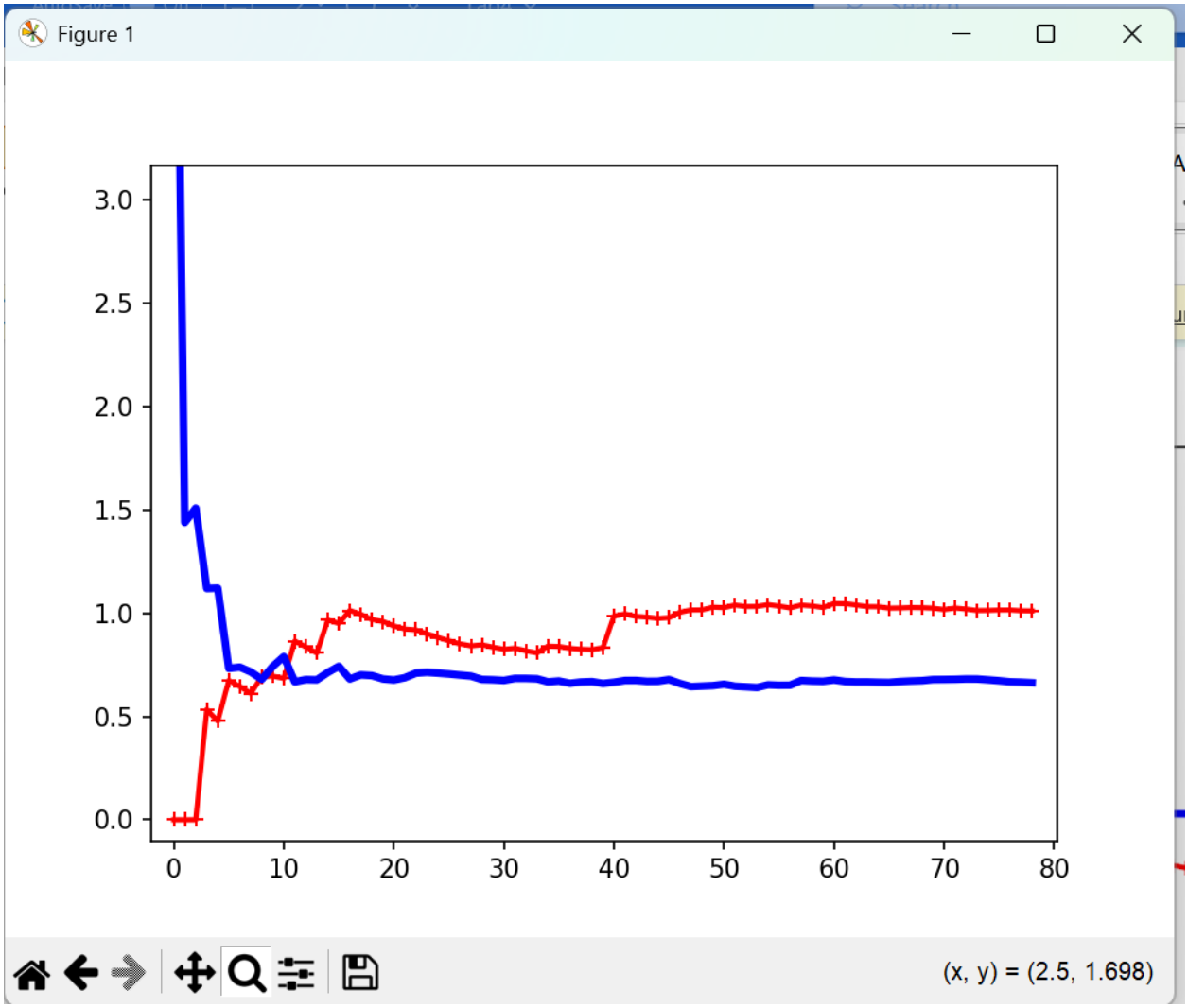
Висновок: як видно із знайдених коефіцієнтів побудованої поліноміальної моделі, які є близькими до модельних, створена модель навчена правильно.

Завдання 2.6. Побудова кривих навчання

Криві навчання для звичайної лінійної регресійної моделі із попереднього завдання



Криві навчання поліноміальної моделі 10-го ступеня на тих самих даних

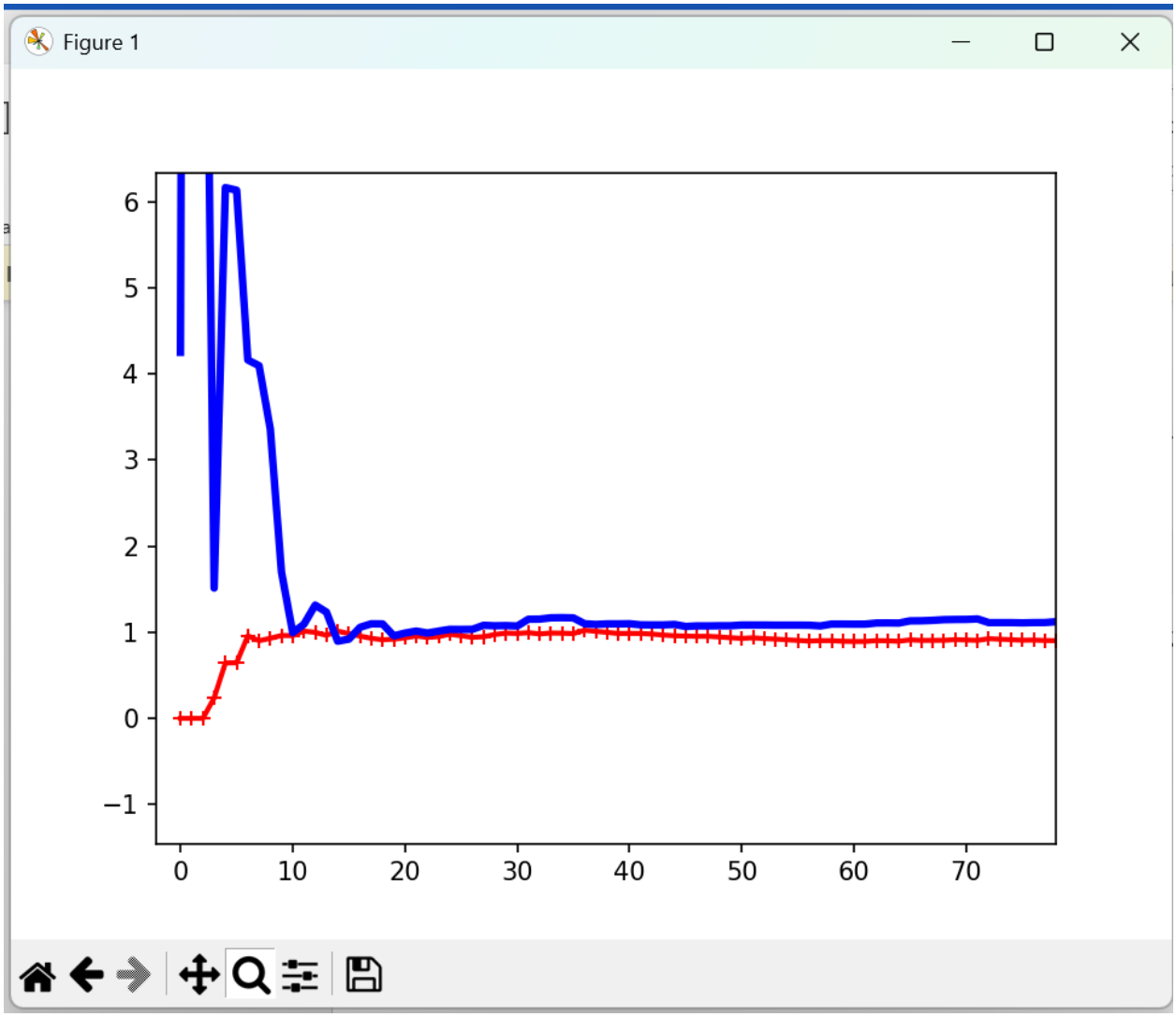


Криві навчання поліноміальної моделі 2-го ступеня на тих самих даних

Код програми:

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


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

```
        train_errors.append(mean_squared_error (y_train_predict,y_train[:m]))
        val_errors.append (mean_squared_error (y_val_predict, y_val))
    plt.plot(np.sqrt(train_errors), "r-+", linewidth=2, label="train")
    plt.plot(np.sqrt(val_errors), "b-", linewidth=3, label="val")
    plt.show()


m = 100
X = 6 * np.random.rand(m, 1) - 5
y = 0.7 * X ** 2 + X + 3 + np.random.randn(m, 1)
num_training = int(0.8 * len(X))
num_test = len(X) - num_training
# Тренувальні дані
X_train, y_train = X[:num_training], y[:num_training]
# Тестові дані
X_test, y_test = X[num_training:], y[num_training:]
# Створення об'єкта лінійного регресора
linear_regressor = linear_model.LinearRegression()
plot_learning_curves(linear_regressor, X, y)


polynomial_regression10 = Pipeline([("poly_features",
PolynomialFeatures(degree=2, include_bias=False)), ("lin_reg",
linear_model.LinearRegression())])
plot_learning_curves(polynomial_regression10, X, y)
polynomial_regression2 = Pipeline([("poly_features", PolynomialFeatures(degree=2,
include_bias=False)), ("lin_reg", linear_model.LinearRegression())])
plot_learning_curves(polynomial_regression2, X, y)
```



Посилання на GitHub: https://github.com/missShevel/SHI_Shevel_Olha_IPZ-21-1/tree/master/Lab4