**ЛАБОРАТОРНА РОБОТА № 3**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ**

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

**Хід роботи:**

**Завдання 1:**  Створення регресора однієї змінної

**Результат:**

Код:

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')

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))

output\_model\_file = 'model.pkl'

with open(output\_model\_file, 'wb') as f:

    pickle.dump(regressor, f)

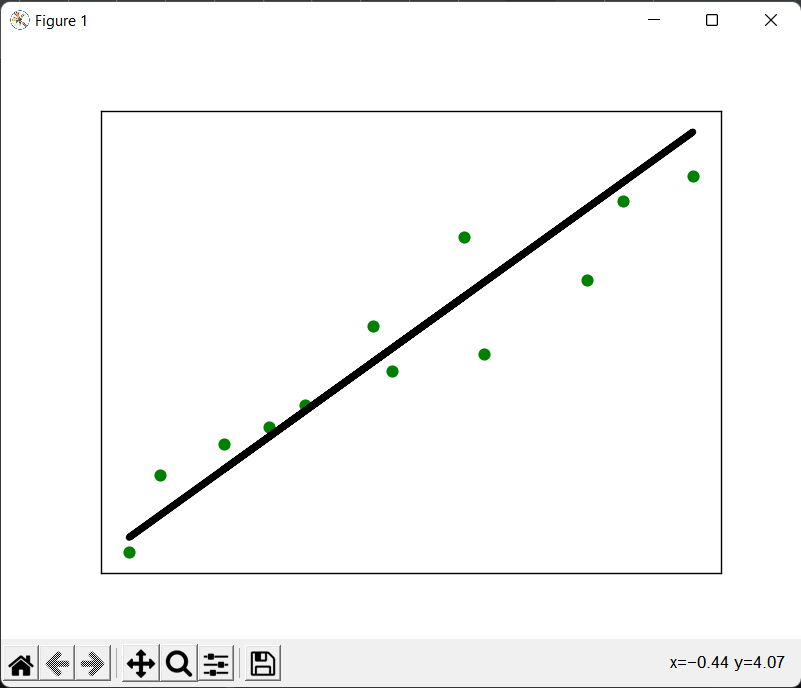
with open(output\_model\_file, 'rb') as f:

    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))



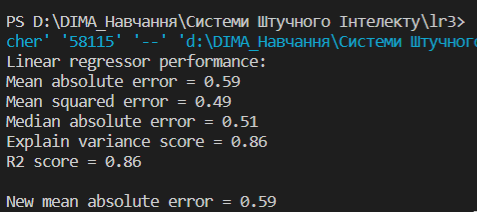


Рис. 1.1. Результат виконання

Модель добре справляється з поставленим завданням.

**Хід роботи:**

**Завдання 2:**  Передбачення за допомогою регресії однієї змінної



Варіант 3 файл: data\_regr\_3.txt

**Результат:**

Код:

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=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))

output\_model\_file = 'model.pkl'

with open(output\_model\_file, 'wb') as f:

    pickle.dump(regressor, f)

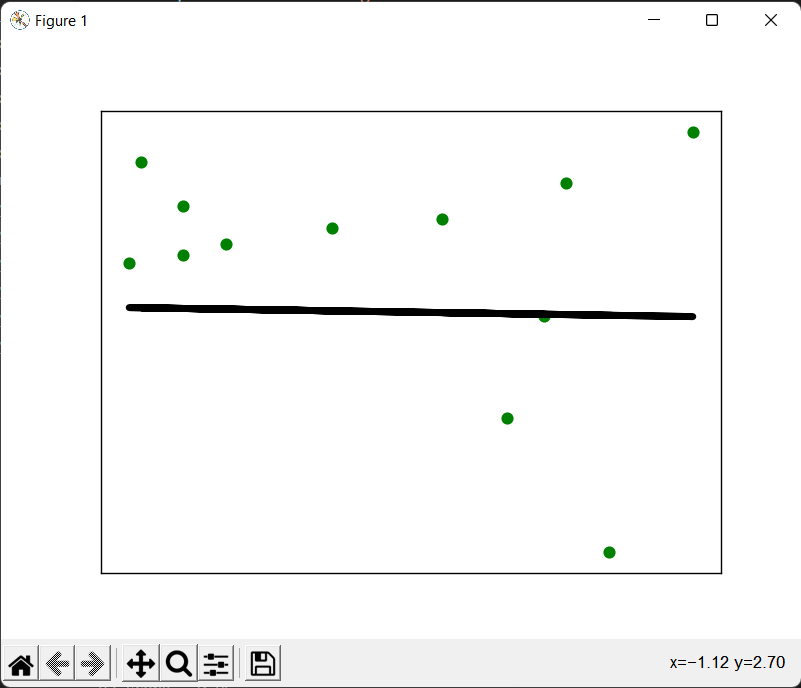
with open(output\_model\_file, 'rb') as f:

    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))



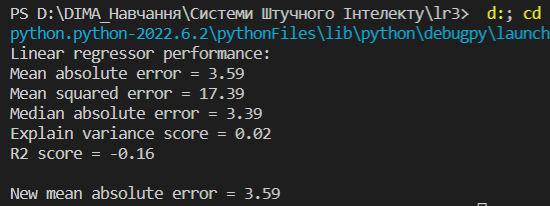


Рис. 2.1. Результат виконання

Цього разу в датасеті є аномальні дані MSE vs MAE, до яких модель не пристосована.

**Хід роботи:**

**Завдання 3:**  Створення багатовимірного регресора

**Результат:**

Код:

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\_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))

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))

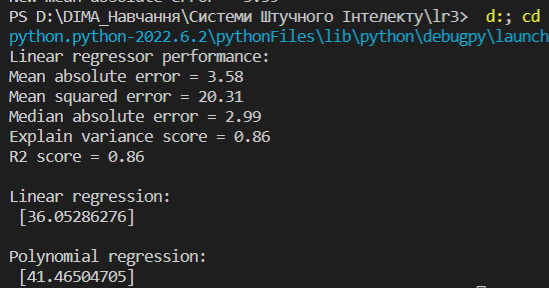


Рис. 3.1. Виведення характеристик та порівняння моделей

Поліноміальний регресор справляється краще за лінійний при регресії з декількома характеристиками.

**Хід роботи:**

**Завдання 4:**  Регресія багатьох змінних

**Результат:**

Код:

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("Linear regressor performance:")

print("regr.coef\_ =", regr.coef\_)

print("regr.intercept\_ =", regr.intercept\_)

print("r2\_score =", round(r2\_score(y\_test, y\_pred), 2))

print("mean\_absolute\_error =", round(mean\_absolute\_error(y\_test, y\_pred), 2))

print("mean\_squared\_error =", round(mean\_squared\_error(y\_test, y\_pred), 2))

fig, ax = plt.subplots()

ax.scatter(y\_test, y\_pred, 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()

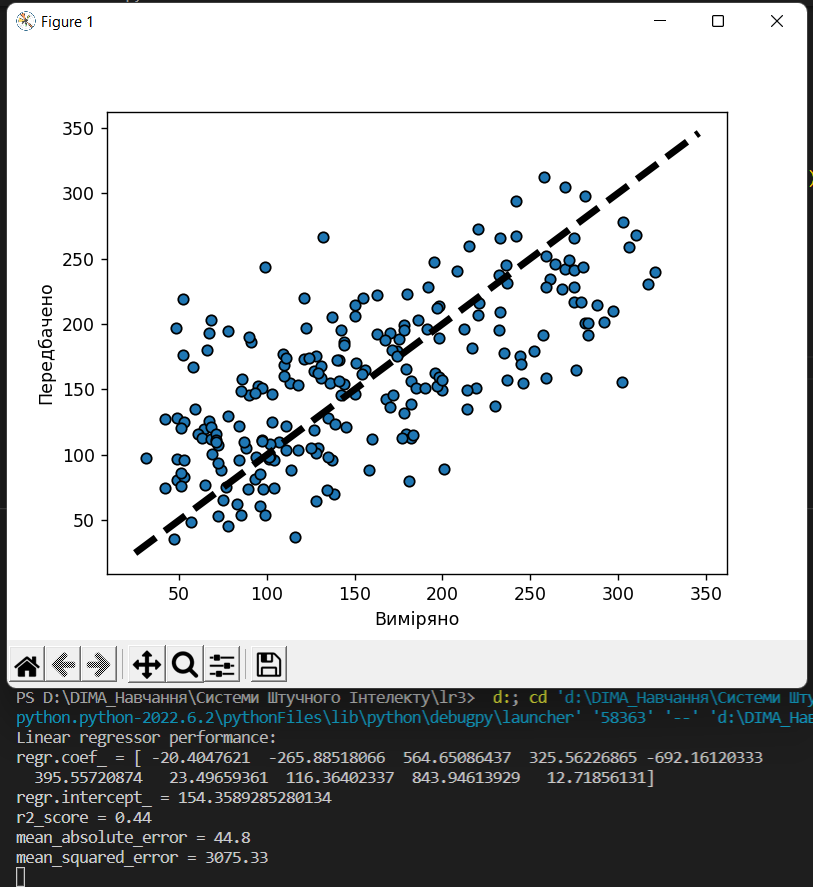
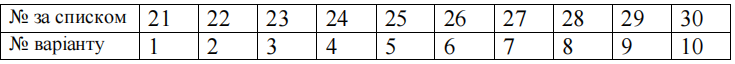


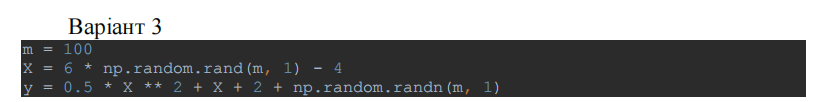
Рис. 4.1. Результат

Похибка є великою, але обрана регресія працює краще ніж звичайна регресія з використанням середніх значень.

**Хід роботи:**

**Завдання 5:**  Самостійна побудова регресії





**Результат:**

Код:

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

from sklearn.preprocessing import PolynomialFeatures

m = 100

X = 6 \* np.random.rand(m, 1) - 4

y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)

fig, ax = plt.subplots()

ax.scatter(X, y, edgecolors = (0, 0, 0))

plt.show()

print(X[1], y[1])

poly\_features = PolynomialFeatures(degree=2, include\_bias=False)

X\_poly = poly\_features.fit\_transform(np.array(X).reshape(-1, 1))

linear\_regression = linear\_model.LinearRegression()

linear\_regression.fit(X\_poly, y)

print(linear\_regression.intercept\_, linear\_regression.coef\_)

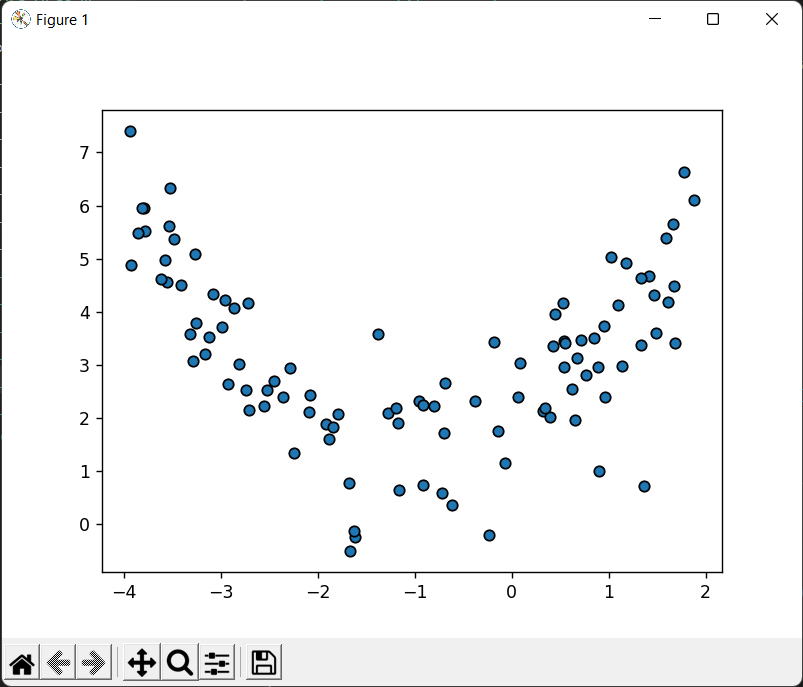
y\_pred = linear\_regression.predict(X\_poly)

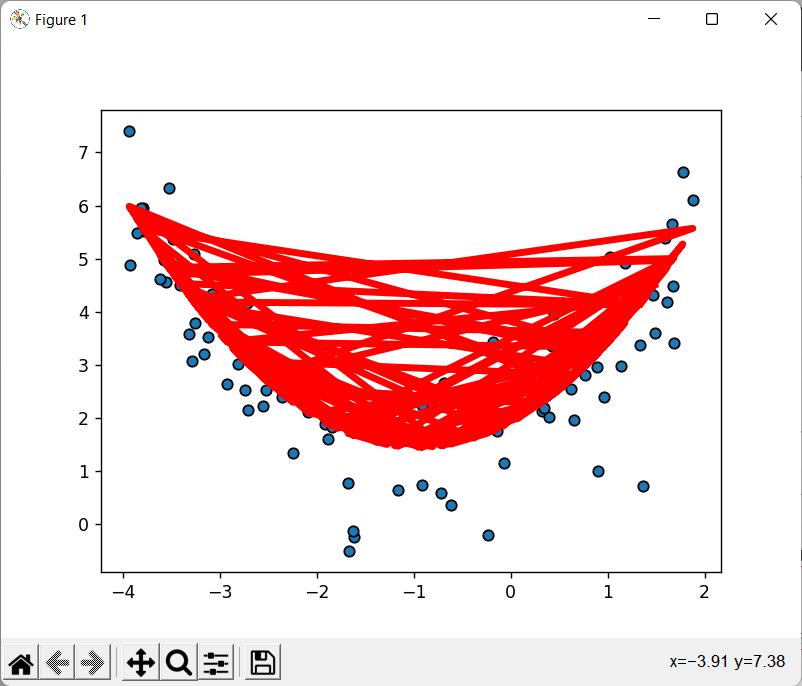
fig, ax = plt.subplots()

ax.scatter(X, y, edgecolors = (0, 0, 0))

plt.plot(X, y\_pred, color='red', linewidth=4)

plt.show()





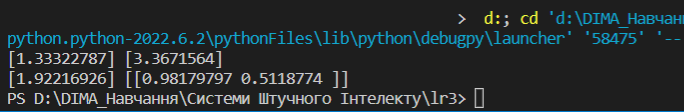


Рис. 5.1. Результат

Модель рівняння:

y = 0.5x2 + 1x + 2 + гаусовий шум

Отримана модель регресії з передбаченими коефіцієнтами:

y = 0.53x2 + 1.1x + 2

**Хід роботи:**

**Завдання 6:**  Побудова кривих навчання

**Результат:**

Код:

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

from sklearn.preprocessing import PolynomialFeatures

from sklearn.pipeline import Pipeline

m = 100

X = 6 \* np.random.rand(m, 1) - 4

y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)

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))

    fig, ax = plt.subplots()

    plt.ylim(0, 2)

    ax.plot(np.sqrt(train\_errors), "r-+", linewidth = 2, label = 'train')

    ax.plot(np.sqrt(val\_errors), "b-", linewidth = 3, label = 'val')

    plt.show()

linear\_regression = linear\_model.LinearRegression()

plot\_learning\_curves(linear\_regression, np.array(X).reshape(-1, 1), y)

polynomial\_regression = Pipeline([

    ('poly\_features', PolynomialFeatures(degree=2, include\_bias=False)),

    ('lin\_reg', linear\_model.LinearRegression()),

])

plot\_learning\_curves(polynomial\_regression, np.array(X).reshape(-1, 1), y)

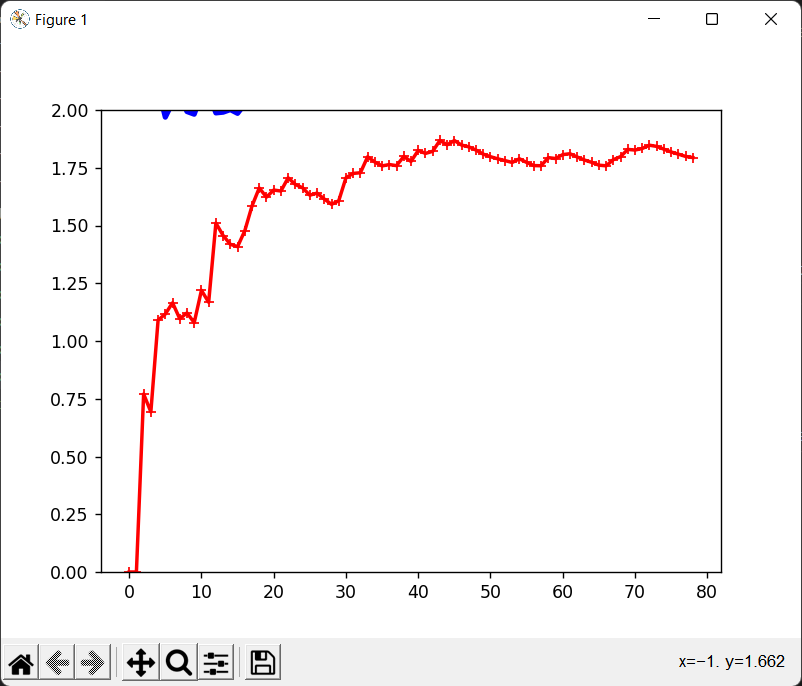


Рис. 6.1. Криві навчання для лінійної моделі

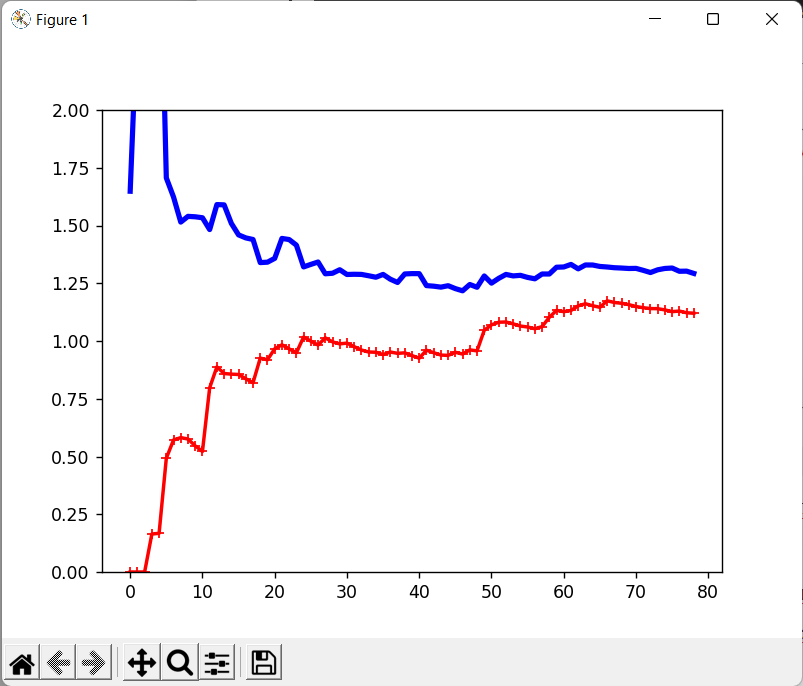


Рис. 6.2. Криві навчання для поліноміальної моделі

**Висновок:** на цій лабораторній роботі ми використовуючи спеціалізовані бібліотеки та мову програмування Python дослідили методи регресії даних у машинному навчанні.