**Лабораторна робота №4**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ**

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

**Хід роботи**

**Завдання №1:**

Напишемо код для регресора однієї змінної:

import pickle

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

import matplotlib.pyplot as plt

# Input file containing data

input\_file = 'data\_singlevar\_regr.txt'

# Read data

data = np.loadtxt(input\_file, delimiter=',')

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

# Train and test split

num\_training = int(0.8 \* len(X))

num\_test = len(X) - num\_training

# Training data

X\_train, y\_train = X[:num\_training], y[:num\_training]

# Test data

X\_test, y\_test = X[num\_training:], y[num\_training:]

# Create linear regressor object

regressor = linear\_model.LinearRegression()

# Train the model using the training sets

regressor.fit(X\_train, y\_train)

# Predict the output

y\_test\_pred = regressor.predict(X\_test)

# Plot outputs

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()

# Compute performance metrics

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))

# Model persistence

output\_model\_file = 'model.pkl'

# Save the model

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

    pickle.dump(regressor, f)

# Load the model

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

    regressor\_model = pickle.load(f)

# Perform prediction on test data

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))

Результати:

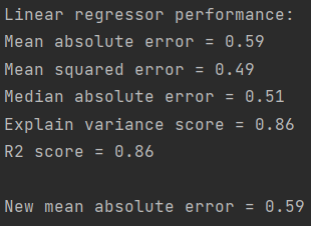


Рис. 1. Результати оцінки якості регресора однієї змінної

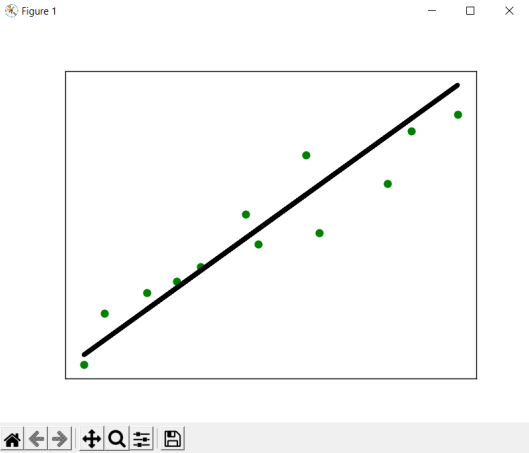


Рис. 2. Результати графіка регресора однієї змінної

Так, з урахуванням оцінок якості та аналізу графіка, можна зробити висновок, що суттєвих відхилень у результатах не спостерігається.

**Завдання №2:**

Код побудованої регресії за своїм варіантом:

import pickle

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

import matplotlib.pyplot as plt

# Input file containing data

input\_file = 'data\_regr\_14.txt'

# Read data

data = np.loadtxt(input\_file, delimiter=',')

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

# Train and test split

num\_training = int(0.8 \* len(X))

num\_test = len(X) - num\_training

# Training data

X\_train, y\_train = X[:num\_training], y[:num\_training]

# Test data

X\_test, y\_test = X[num\_training:], y[num\_training:]

# Create linear regressor object

regressor = linear\_model.LinearRegression()

# Train the model using the training sets

regressor.fit(X\_train, y\_train)

# Predict the output

y\_test\_pred = regressor.predict(X\_test)

# Plot outputs

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()

# Compute performance metrics

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))

# Model persistence

output\_model\_file = 'model.pkl'

# Save the model

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

    pickle.dump(regressor, f)

# Load the model

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

    regressor\_model = pickle.load(f)

# Perform prediction on test data

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))

Результати:

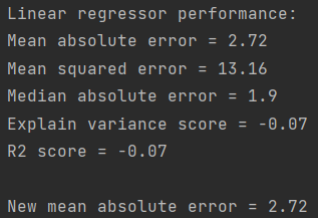


Рис. 3. Результати оцінки якості регресора однієї змінної за своїм варіантом

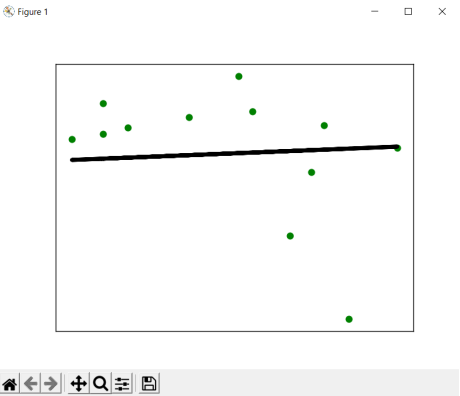


Рис. 4. Результати графіка регресора однієї змінної за своїм варіантом

Отже, судячи з оцінок якості та графіка, можемо спостерігати значні відхилення.

**Завдання №3:**

Код поліноміального регресора:

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

from sklearn.preprocessing import PolynomialFeatures

# Input file containing data

input\_file = 'data\_multivar\_regr.txt'

# Load the data from the input file

data = np.loadtxt(input\_file, delimiter=',')

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

# Split data into training and testing

num\_training = int(0.8 \* len(X))

num\_test = len(X) - num\_training

# Training data

X\_train, y\_train = X[:num\_training], y[:num\_training]

# Test data

X\_test, y\_test = X[num\_training:], y[num\_training:]

# Create the linear regressor model

linear\_regressor = linear\_model.LinearRegression()

# Train the model using the training sets

linear\_regressor.fit(X\_train, y\_train)

# Predict the output

y\_test\_pred = linear\_regressor.predict(X\_test)

# Measure performance

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))

# Polynomial regression

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))

Результат:

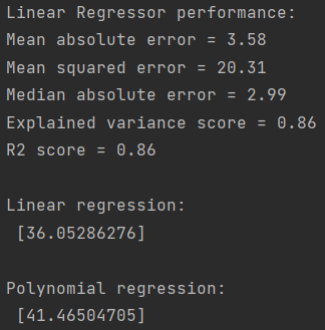


Рис. 5. Результати оцінки якості поліноміального регресора

**Завдання №4:**

Напишемо код для лінійного регресора, використвуючи набір даних по діабету:

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("Linear Regressor performance:")

print("Mean absolute error =", np.round(mean\_absolute\_error(ytest, ypred), 2))

print("Mean squared error =", np.round(mean\_squared\_error(ytest, ypred), 2))

print("R2 score =", np.round(r2\_score(ytest, ypred), 2))

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

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

ig, 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()

Результат:

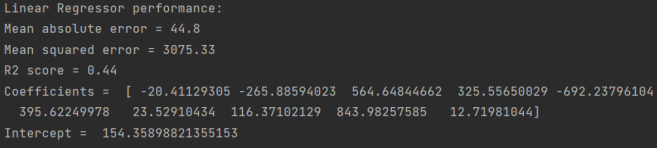


Рис. 6. Результати лінійного регресора, використовуючи дані по діабету

**Завдання №5:**

Напишемо код для побудови регресій, використовуючи дані за своїм варіантом:

import matplotlib.pyplot as plt

import numpy as np

from sklearn import linear\_model

from sklearn.preprocessing import PolynomialFeatures

np.random.seed(42)

m = 100

X = 6 \* np.random.rand(m, 1) - 5

y = 0.7 \* X \*\* 2 + X + 3 + np.random.randn(m, 1)

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

X\_poly = poly\_features.fit\_transform(X)

print("X[0] = ", X[0])

print("X\_poly = ", X\_poly)

lin\_reg = linear\_model.LinearRegression()

lin\_reg.fit(X\_poly, y)

print("Coefficients = ", lin\_reg.coef\_)

print("Intercept = ", lin\_reg.intercept\_)

X\_new = np.linspace(-3, 3, 100).reshape(100, 1)

X\_new\_poly = poly\_features.transform(X\_new)

y\_new = lin\_reg.predict(X\_new\_poly)

plt.plot(X, y, "b.")

plt.plot(X\_new, y\_new, "r-", linewidth=2, label="Predictions")

plt.xlabel("$x\_1$", fontsize=18)

plt.ylabel("$y$", rotation=0, fontsize=18)

plt.legend(loc="upper left", fontsize=14)

plt.axis([-3, 3, 0, 10])

plt.show()

Результати:



Рис. 7. Коефіцієнти за створеною моделлю

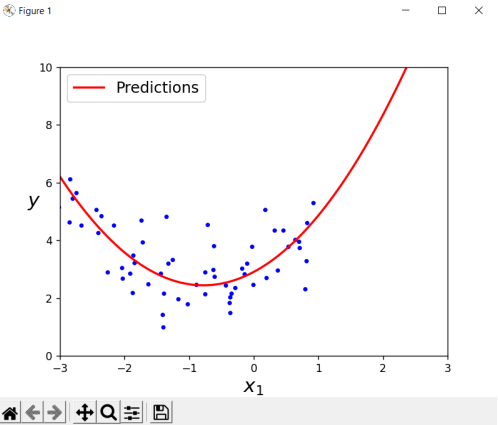


Рис. 8. Отриманий графік, використовуючи створену модель та дані за варіантом

Отже, запишемо модель у вигляді математичного рівняння: 𝑦 = 0.76𝑥1 2 + 1.19𝑥1 + 2.9.

**Завдання №6:**

Код створення кривих навчання моделі для встановлених навчальних даних:

import matplotlib.pyplot as plt

import numpy as np

from sklearn import linear\_model

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

def plot\_learning\_curves(model, X, y):

X\_train, X\_val, y\_train, y\_val = train\_test\_split(

    X, y, test\_size=0.2, random\_state=10)

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[:m], y\_train\_predict))

    val\_errors.append(mean\_squared\_error(y\_val, y\_val\_predict))

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

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

    plt.legend(loc="upper right", fontsize=14)

    plt.xlabel("Training set size", fontsize=14)

    plt.ylabel("RMSE", fontsize=14)

    plt.axis([0, 80, 0, 3])

    plt.show()

np.random.seed(42)

m = 100

X = 6 \* np.random.rand(m, 1) - 5

y = 0.7 \* X \*\* 2 + X + 3 + np.random.randn(m, 1)

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

X\_poly = poly\_features.fit\_transform(X)

lin\_reg = linear\_model.LinearRegression()

lin\_reg.fit(X\_poly, y)

plot\_learning\_curves(lin\_reg, X, y)

polynomial\_regression = Pipeline([

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

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

])

plot\_learning\_curves(polynomial\_regression, X, y)

Результати:

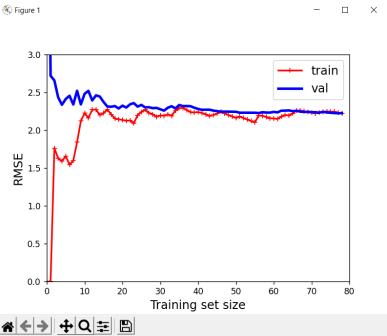


Рис. 9. Криві навчання для лінійної моделі

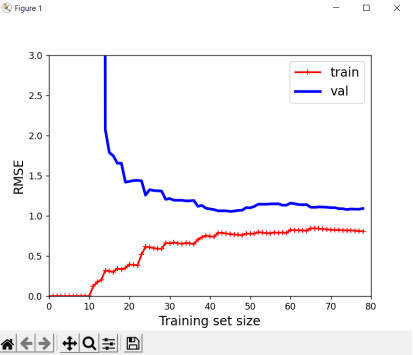


Рис. 10. Криві навчання для поліноміальної моделі 10-ступеня

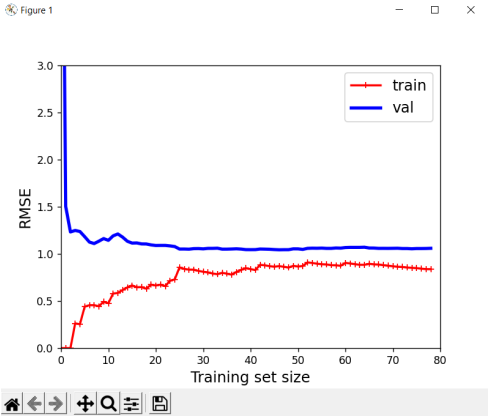


Рис. 11. Криві навчання для поліноміальної моделі 2-ступеня

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