**Лабораторна робота №4**

**Тема:** дослідження методів регресії

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

Варіант 2

Хід роботи:

Посилання на GitHub :

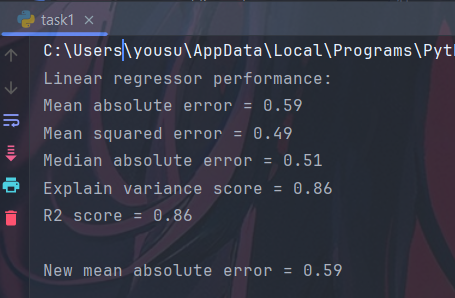
<https://github.com/Dubnitskyi/AI_all_labs/tree/master/Lab4>

**Завдання №1:** Створення регресора однієї змінної

Код програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
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)  
  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

Результат :



Изображение выглядит как снимок экрана, текст, линия, График

Автоматически созданное описание

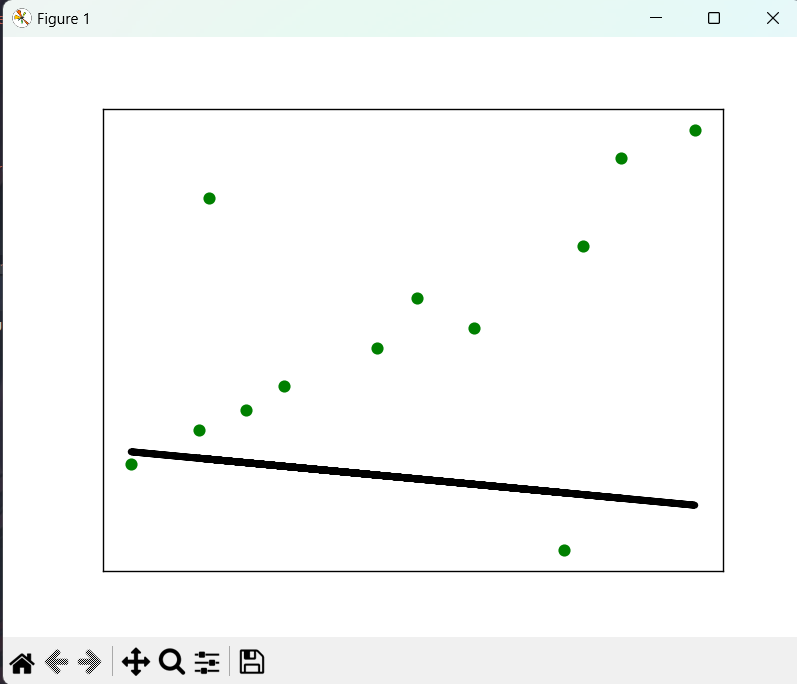
**Зробіть висновок**   
Ця регресивна модель має високу точність

**Завдання №2:** Передбачення за допомогою регресії однієї змінної

Код програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
  
input\_file = 'data\_regr\_2.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))

Результат :



Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

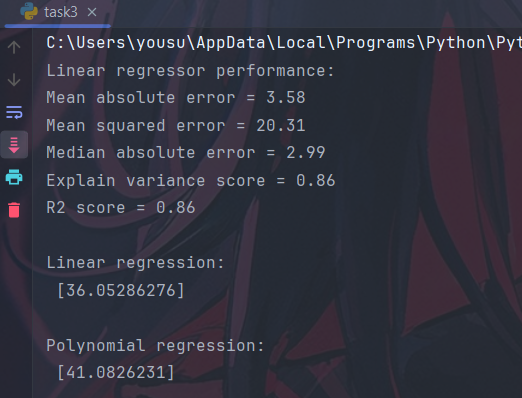
**Зробіть висновок**   
Ця модель має низьку точність.

**Завдання 3** Створення багатовимірного регресора

Код програми:

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:]  
  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
  
y\_test\_pred = 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", regressor.predict(datapoint))  
print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint))

Результат :

****

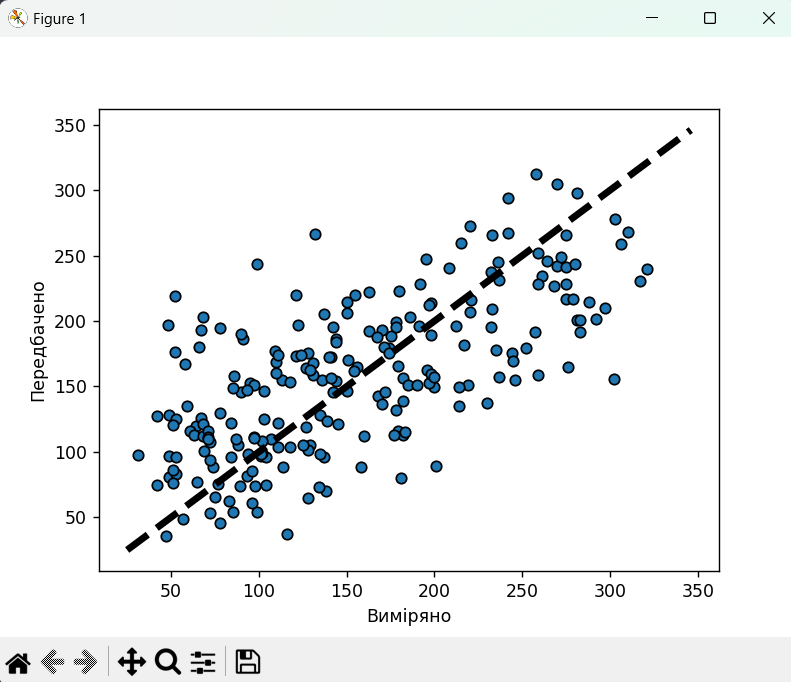
**Зробіть висновок**   
Пліноміальний регресор краще справляється за лінійний регресор

**Завдання 4** Регресія багатьох змінних

Код програми:

import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
import numpy as np  
import sklearn.metrics as sm  
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)  
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()  
  
print("Linear regressor performance:")  
print("Mean absolute error =", round(sm.mean\_absolute\_error(ytest, ypred), 2))  
print("Mean squared error =", round(sm.mean\_squared\_error(ytest, ypred), 2))  
print("Median absolute error =", round(sm.median\_absolute\_error(ytest, ypred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(ytest, ypred), 2))  
print("R2 score =", round(sm.r2\_score(ytest, ypred), 2))

Результат :

****

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Зробіть висновок**   
Дані занадто сильно розкидані, тому моделі складно точно обчислити велику кількість даних.

**Завдання 5** Самостійна побудова регресії

Код програми:

import numpy as np  
from sklearn import linear\_model  
from sklearn.preprocessing import PolynomialFeatures  
  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg') \  
  
m = 100  
X = np.linspace(-3, 3, m).reshape(-1, 1)  
y = np.sin(X).flatten() + np.random.uniform(-0.5, 0.5, m)  
  
poly = PolynomialFeatures(degree=2, include\_bias=False)   
X\_poly = poly.fit\_transform(X)  
  
lin\_reg = linear\_model.LinearRegression()  
lin\_reg.fit(X\_poly, y)  
  
print("Intercept:", lin\_reg.intercept\_)  
print("Coefficients:", lin\_reg.coef\_)  
  
y\_pred = lin\_reg.predict(X\_poly)  
  
plt.scatter(X, y, color='green', label='Дані з шумом')  
plt.plot(X, y\_pred, color='red', linewidth=1, label='Прогноз')  
plt.xlabel("X")  
plt.ylabel("y")  
plt.legend()  
plt.show()

Результат :

Изображение выглядит как текст, снимок экрана, диаграмма, График

Автоматически созданное описание

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Зробіть висновок**

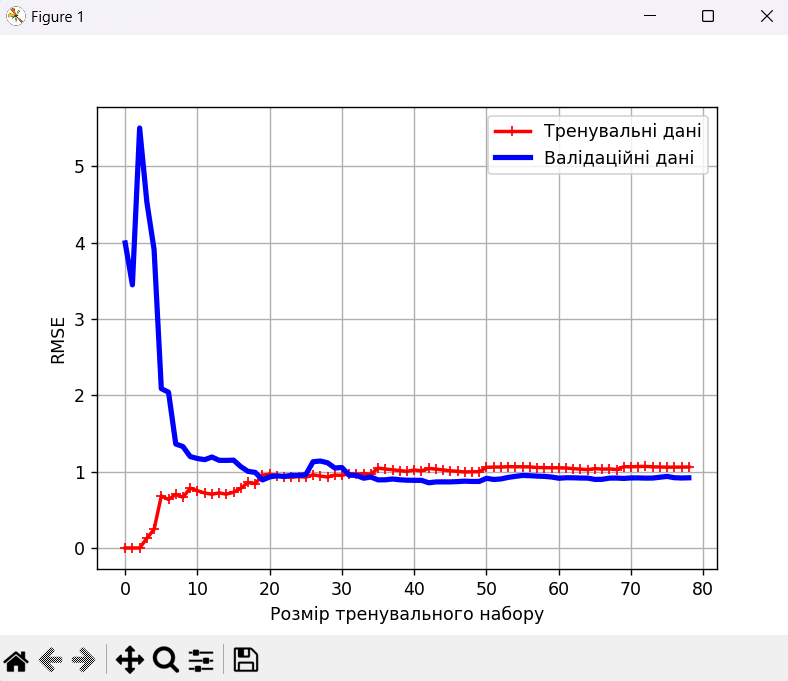
Поліномна регресія дає змогу аналізувати не лінійні дані

**Завдання 6** Побудова кривих навчання

Код програми:

from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
import numpy as np  
from sklearn import linear\_model  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.pipeline import Pipeline  
  
import matplotlib  
import matplotlib.pyplot as plt  
matplotlib.use('TkAgg')  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.4 \* X \*\* 2 + X + 4 + 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[: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="Тренувальні дані")  
 plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label="Валідаційні дані")  
 plt.xlabel("Розмір тренувального набору")  
 plt.ylabel("RMSE")  
 plt.legend()  
 plt.grid()  
 plt.show()  
  
  
polynomial\_regression = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=2, include\_bias=False)),  
 ("lin\_reg", linear\_model.LinearRegression()),  
])  
  
plot\_learning\_curves(polynomial\_regression, X, y)

Результат :

****

**Висновок:** Під час лабораторної роботи я використав спеціалізовані бібліотеки та мову програмування Python та дослідив методи регресії даних у машинному навчанні.