**ЛАБОРАТОРНА РОБОТА № 3**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ**

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

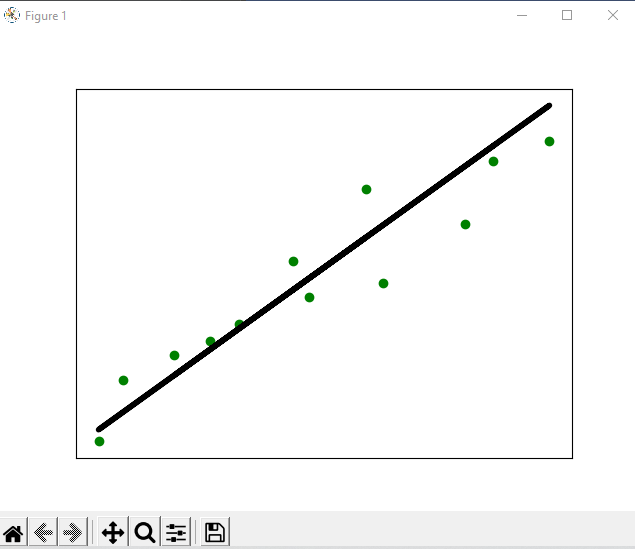
**Хід роботи**

Завдання 2.1. Створення регресора однієї змінної

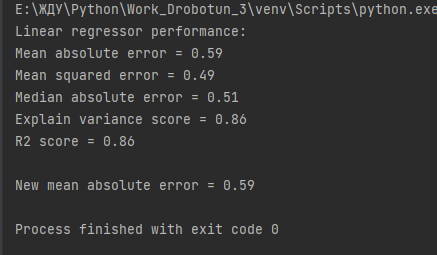
import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_singlevar\_regr.txt'  
  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = 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 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 = linear\_model.LinearRegression()  
  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.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))  
  
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 Код програми**



**Рис. 1.2 Результат виконання програми**



**Рис. 1.3 Результат виконання програми**

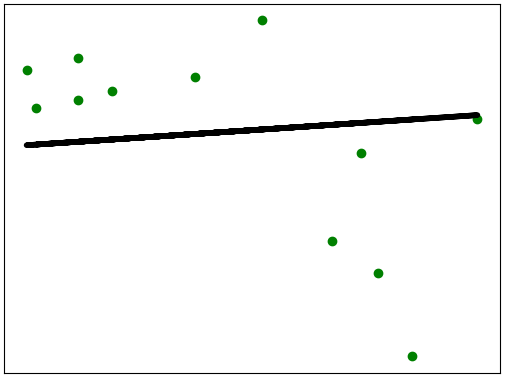
Завдання 2.2. Передбачення за допомогою регресії однієї змінної

За списком у журналі я **5** варіант

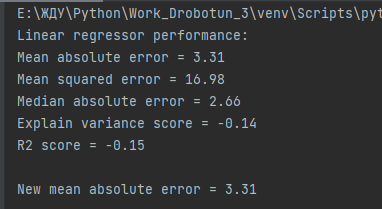
import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
# Вхідний файл, який містить дані  
input\_file = 'data\_regr\_5.txt'  
  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-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 = linear\_model.LinearRegression()  
  
regressor.fit(X\_train, y\_train)  
  
# Прогнозування результату  
y\_test\_pred = regressor.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))  
  
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. 1.4 Код програми**



**Рис. 1.5 Результат виконання програми**



**Рис. 1.6 Результат виконання програми**

Завдання 2.3. Створення багатовимірного регресора

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("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 = 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. 1.7 Код програми**

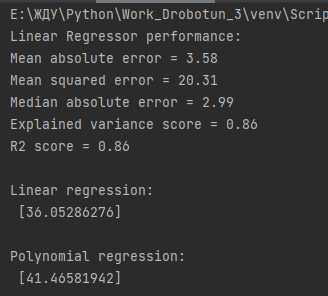


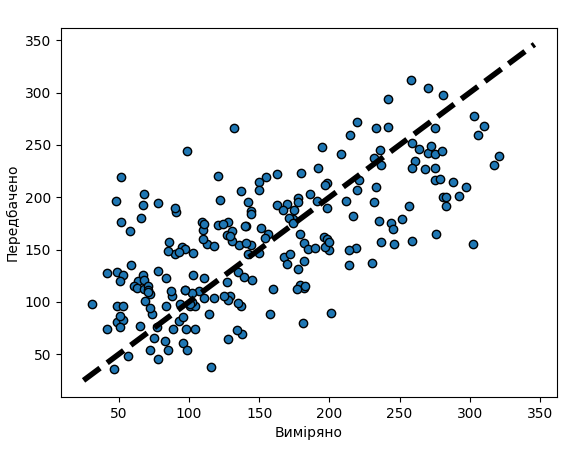
Рис. 1.8 Результат виконання програми

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)

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()

**Рис. 1.9 Код програми**



**Рис. 2.1 Результат виконання програми**

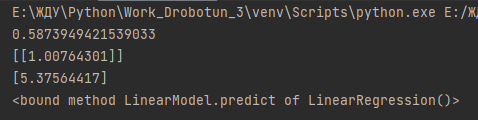
Завдання 2.5. Самостійна побудова регресії

За списком у журналі я **5** варіант

import numpy as np  
from sklearn.linear\_model import LinearRegression  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.4 \* X \*\* 2 + X + 4 + np.random.randn(m, 1)  
  
reg = LinearRegression().fit(X, y)  
print(reg.score(X, y))  
  
print(reg.coef\_)  
  
print(reg.intercept\_)

print(reg.predict)

**Рис. 2.2 Код програми**

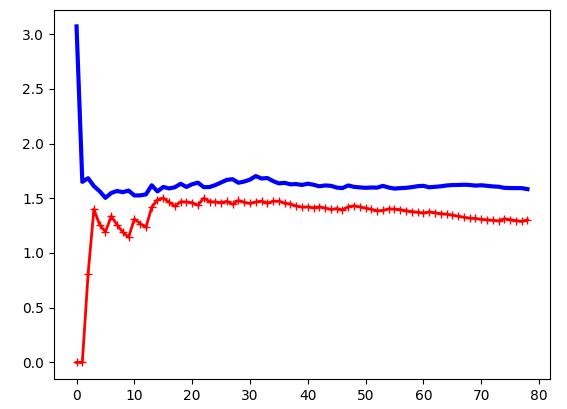


**Рис. 2.3 Результат виконання програми**

Завдання 2.6. Побудова кривих навчання

from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.preprocessing import PolynomialFeatures  
  
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\_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()  
lin\_reg=LinearRegression()  
plot\_learning\_curves(lin\_reg,X,y)

**Рис. 2.4 Код програми**



**Рис. 2.5 Результат виконання програми**

from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.preprocessing import PolynomialFeatures  
  
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\_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()  
lin\_reg=LinearRegression()  
plot\_learning\_curves(lin\_reg,X,y)  
  
from sklearn.pipeline import Pipeline

polynomial\_regression = Pipeline([("poly features",PolynomialFeatures(degree=10, include\_bias=False)),("lin\_reg", LinearRegression()),])  
  
plot\_learning\_curves(polynomial\_regression,X,y)

**Рис. 2.6 Код програми**

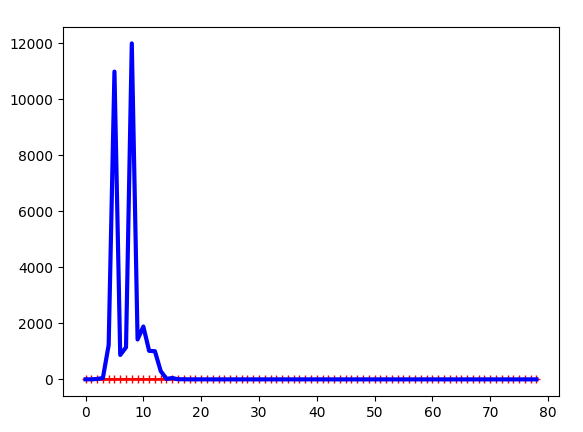


Рис. 2.7 Результат виконання програми

**Висновок:** я використовав спеціалізовані бібліотеки та мову програмування Python дослідивши методи регресії даних у машинному навчанні.

Простий спосіб передбачає додавання ступенів кожної ознаки у вигляді нових ознак і наступне навчання лінійної моделі на такому розширеному наборі ознак. Цей прийом називається поліноміальною регресією (polynomial regression).Якщо вхідні данні розподілені нелінійно, то, безумовно, пряму лінію ніколи не буде підігнано під такі дані належним чином. Тому скориставшись класом PolynomialFeatures з Scikit-Learn, щоб перетворити наші навчальні дані, додавши як нові ознаки квадрат (поліном 2-го ступеня) кожної ознаки.