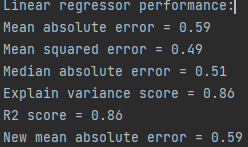
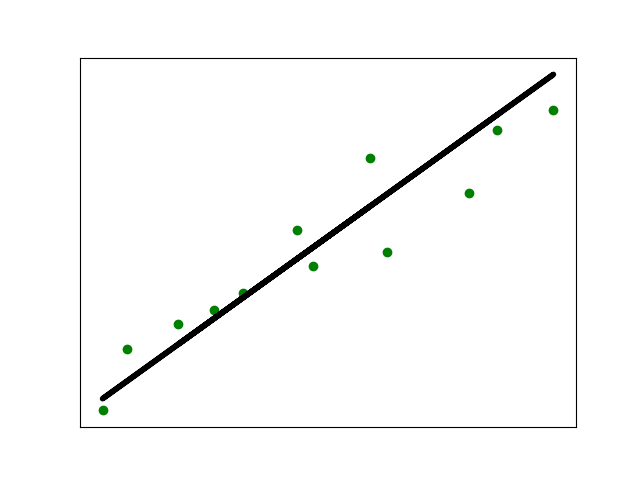
Системи штучного інтелекту.

Лабораторна робота 4.Федорович Дмитро ІПЗ-21-3

**Завдання №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))  
# Файл для збереження моделі  
output\_model\_file = 'model.pkl'  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
# Завантаження моделі  
with open(output\_model\_file, 'rb') as f:  
 regressor\_model = pickle.load(f)  
y\_test\_pred\_new = regressor\_model.predict(X\_test)  
print("New mean absolute error =", round(sm.mean\_absolute\_error(y\_test,  
y\_test\_pred\_new), 2))





Графік зображує незалежні змінні x та залежну змінну y. Лінійна модель регресії показує нормальні показники, проте є відхилення.

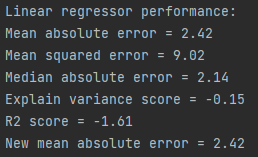
**Завдання №2.** Передбачення за допомогою регресії однієї змінної

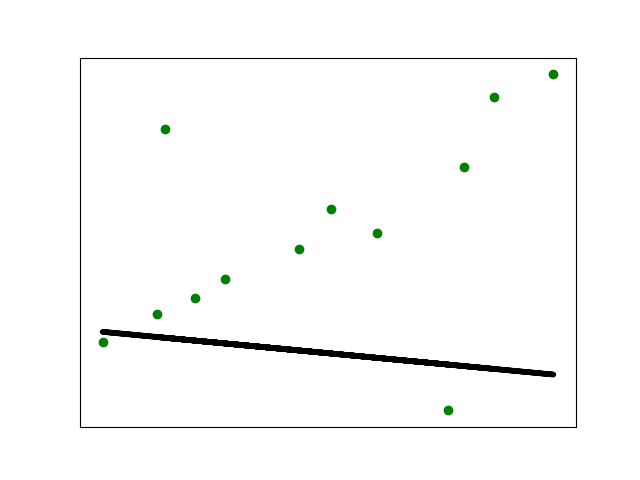
№ за списком – 22

№ варіанту – 2

Варіант 2 файл: data\_regr\_2.txt

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\_2.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))  
# Файл для збереження моделі  
output\_model\_file = 'model\_task2.pkl'  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
# Завантаження моделі  
with open(output\_model\_file, 'rb') as f:  
 regressor\_model = pickle.load(f)  
y\_test\_pred\_new = regressor\_model.predict(X\_test)  
print("New mean absolute error =", round(sm.mean\_absolute\_error(y\_test,  
y\_test\_pred\_new), 2))

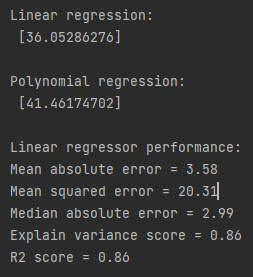




Лінійна регресія показує непогані показники - середнє абсолютне відхилення, тобто середня різниця між прогнозованими й фактичними даними є 2.42. А середня квадратична різниця - 9.09. Оцінка r2 -1.61 свідчить що модель не може пояснити відхилення між частиною показників.

**Завдання №3.** Створення багатовимірного регресора

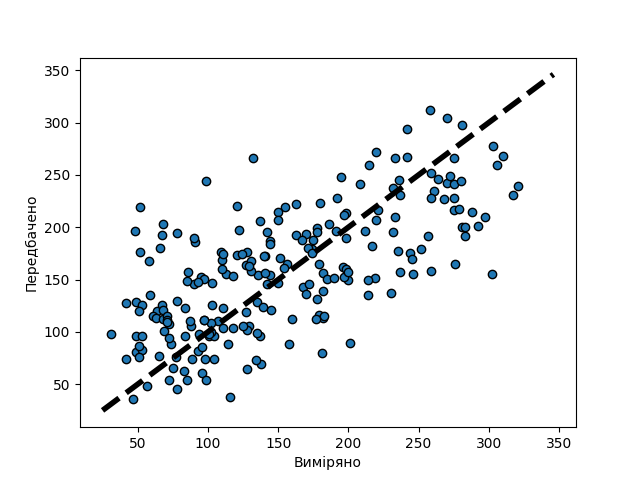
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
# Вхідний файл, який містить дані  
input\_file = 'data\_multivar\_regr.txt'  
# Завантаження даних  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-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:]  
# 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)  
# Створення об'єкта полілінійного регресора  
poly\_linear\_model = linear\_model.LinearRegression()  
poly\_linear\_model.fit(X\_train\_transformed, y\_train)  
# Створення об'єкта лінійного регресора  
linear\_regressor = linear\_model.LinearRegression()  
linear\_regressor.fit(X\_train, y\_train)  
print("\nLinear regression:\n",  
linear\_regressor.predict(datapoint))  
print("\nPolynomial regression:\n",  
poly\_linear\_model.predict(poly\_datapoint))  
# Прогнозування результату  
y\_test\_pred = linear\_regressor.predict(X\_test)  
print("\nLinear 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))

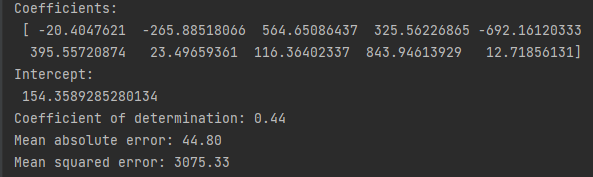


В даному прикладі поліноміальна регресія дає ближчий до реальності результат ніж лінійна регресія (41.46 - 41.35), що показує її вищу ефективність в складних залежностях.

**Завдання №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('Coefficients: \n', regr.coef\_)  
print('Intercept: \n', regr.intercept\_)  
print('Coefficient of determination: %.2f' % r2\_score(ytest, ypred))  
print('Mean absolute error: %.2f' % mean\_absolute\_error(ytest, ypred))  
print('Mean squared error: %.2f' % mean\_squared\_error(ytest, ypred))  
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()





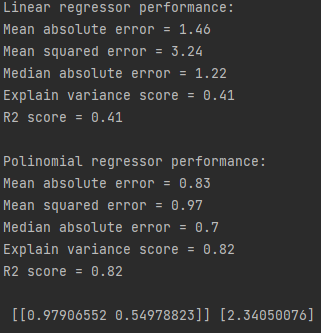
По графіку можна розуміти, що зі збільшенням фактичного рівня цукру в крові, збільшується і спрогнозований. Лінія показує місце з мінімальною різницею між спостережуваними й передбаченими показниками. Модель в 44% випадків може пояснити дисперсію між даними. Середня абсолютна похибка моделі 44.8, а середня квадратична похибка 3075.

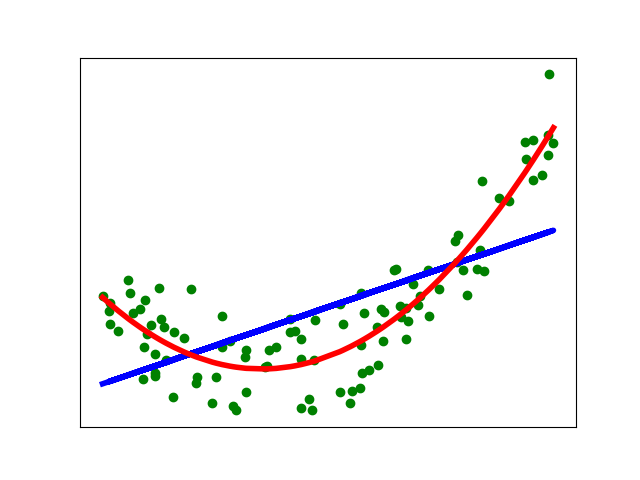
**Завдання №5.** Самостійна побудова регресії

№ за списком – 22

№ варіанту – 2

import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import PolynomialFeatures  
np.random.seed(0)  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.6 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
# Побудова графіка  
plt.scatter(X, y, color='green', label='data')  
# Лінійна регресія  
linear\_regression = linear\_model.LinearRegression()  
linear\_regression.fit(X, y)  
y\_linear\_pred = linear\_regression.predict(X)  
# Поліноміальна регресія  
polynomial\_features = PolynomialFeatures(degree=2, include\_bias=False)  
X\_poly = polynomial\_features.fit\_transform(X)  
linear\_regression\_poly = linear\_model.LinearRegression()  
linear\_regression\_poly.fit(X\_poly, y)  
y\_poly\_pred = linear\_regression\_poly.predict(X\_poly)  
# Побудова графіка  
plt.plot(X, y\_linear\_pred, color='blue', label='linear', linewidth=4)  
sort\_indices = np.argsort(X[:, 0])  
X\_sorted = X[sort\_indices]  
y\_poly\_pred\_sorted = y\_poly\_pred[sort\_indices]  
plt.plot(X\_sorted, y\_poly\_pred\_sorted, color='red', label='polynomial',  
linewidth=4)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
print("Linear regressor performance:")  
print("Mean absolute error =",  
round(sm.mean\_absolute\_error(y, y\_linear\_pred), 2))  
print("Mean squared error =",  
round(sm.mean\_squared\_error(y, y\_linear\_pred), 2))  
print("Median absolute error =",  
round(sm.median\_absolute\_error(y, y\_linear\_pred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(y,  
y\_linear\_pred), 2))  
print("R2 score =", round(sm.r2\_score(y, y\_linear\_pred), 2))  
print("\nPolinomial regressor performance:")  
print("Mean absolute error =",  
round(sm.mean\_absolute\_error(y, y\_poly\_pred), 2))  
print("Mean squared error =",  
round(sm.mean\_squared\_error(y, y\_poly\_pred), 2))  
print("Median absolute error =",  
round(sm.median\_absolute\_error(y, y\_poly\_pred), 2))  
print("Explain variance score =", round(sm.explained\_variance\_score(y,  
y\_poly\_pred), 2))  
print("R2 score =", round(sm.r2\_score(y, y\_poly\_pred), 2))  
print('\n', linear\_regression\_poly.coef\_,  
linear\_regression\_poly.intercept\_)





y = 0.58208917 \* x2 + 1.03175502 \* x + 2.1067843

**Завдання №6.** Побудова кривих навчання.

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import PolynomialFeatures  
  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 3  
y = 0.6 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
# Функція для побудови кривих навчання  
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=42)  
 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.legend()  
 plt.show()  
  
  
lin\_reg = LinearRegression()  
plot\_learning\_curves(lin\_reg, X, y)  
  
poly\_reg = Pipeline([  
 ("poly\_features", PolynomialFeatures(degree=10, include\_bias=False)),  
 ("lin\_reg", LinearRegression())])  
plot\_learning\_curves(poly\_reg, X, y)

