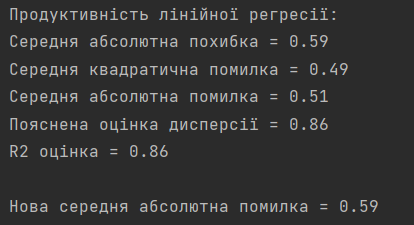
**Кушер Дмитро Євгенович ЗІПЗк-22-1**

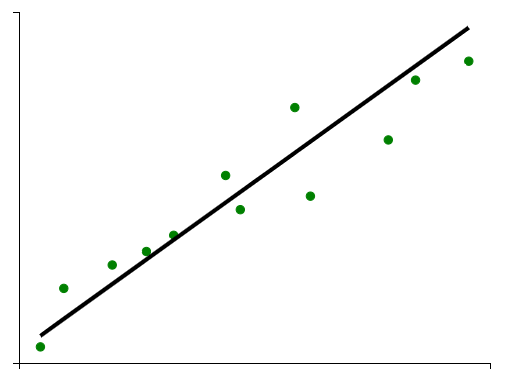
**Лаб 3**

**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("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 2))  
output\_model\_file = 'model.pkl'  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

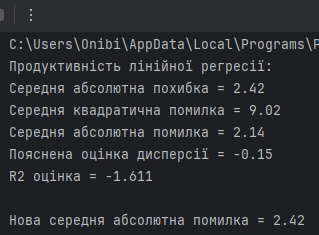


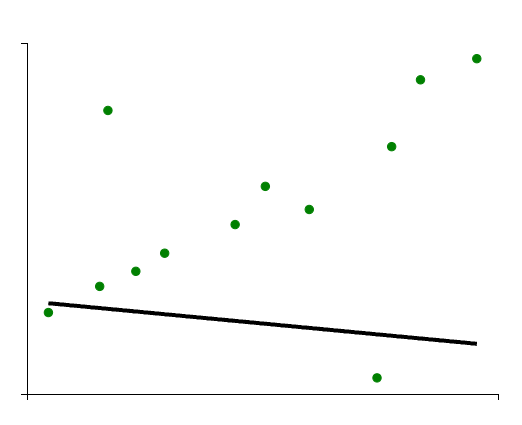


**Завдання 2.2**

Передбачення за допомогою регресії 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\_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("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 3))  
output\_model\_file = 'model.pkl'  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

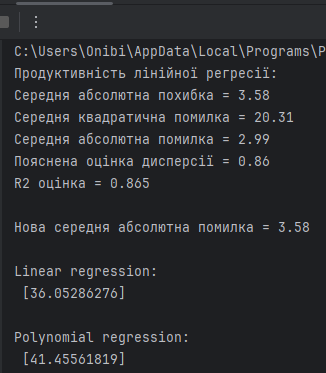




**Завдання 2.3**

Створення багатовимірного регресора

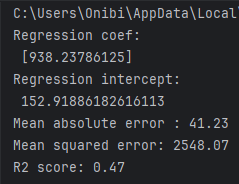
import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import PolynomialFeatures  
# Вхідний файл, який містить дані  
input\_file = 'data\_multivar\_regr.txt'  
# Завантаження даних  
data = 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:]  
# Створення об'єкта лінійного регресора  
regressor = linear\_model.LinearRegression()  
regressor.fit(X\_train, y\_train)  
# Прогнозування результату  
y\_test\_pred = regressor.predict(X\_test)  
print("Продуктивність лінійної регресії:")  
print("Середня абсолютна похибка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Середня квадратична помилка =", round(sm.mean\_squared\_error(y\_test, y\_test\_pred), 2))  
print("Середня абсолютна помилка =", round(sm.median\_absolute\_error(y\_test, y\_test\_pred), 2))  
print("Пояснена оцінка дисперсії =", round(sm.explained\_variance\_score(y\_test, y\_test\_pred), 2))  
print("R2 оцінка =", round(sm.r2\_score(y\_test, y\_test\_pred), 3))  
output\_model\_file = 'model.pkl'  
# Збереження моделі  
with open(output\_model\_file, 'wb') as f:  
 pickle.dump(regressor, f)  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nНова середня абсолютна помилка =", round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))  
# 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)  
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))

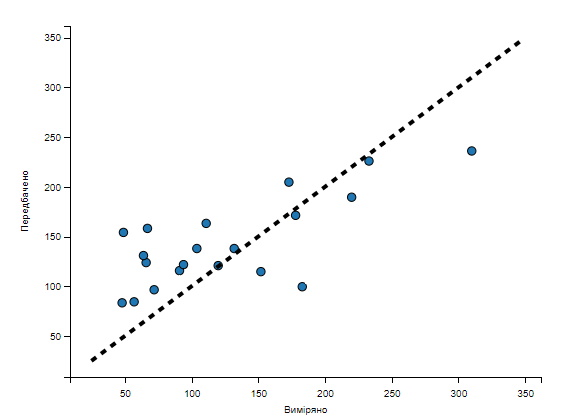


**Завдання 2.4**

Регресія багатьох змінних

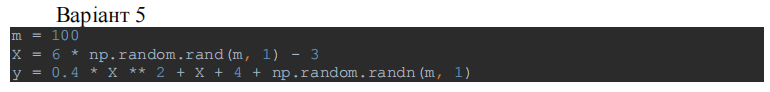
import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import datasets, linear\_model  
from sklearn.metrics import mean\_squared\_error, r2\_score, mean\_absolute\_error  
# Load the diabetes dataset  
diabetes\_X, diabetes\_y = datasets.load\_diabetes(return\_X\_y=True)  
# Use only one feature  
diabetes\_X = diabetes\_X[:, np.newaxis, 2]  
# Split the data into training/testing sets  
diabetes\_X\_train = diabetes\_X[:-20]  
diabetes\_X\_test = diabetes\_X[-20:]  
# Split the targets into training/testing sets  
diabetes\_y\_train = diabetes\_y[:-20]  
diabetes\_y\_test = diabetes\_y[-20:]  
# Create linear regression object  
regr = linear\_model.LinearRegression()  
# Train the model using the training sets  
regr.fit(diabetes\_X\_train, diabetes\_y\_train)  
# Make predictions using the testing set  
diabetes\_y\_pred = regr.predict(diabetes\_X\_test)  
# The coefficients  
print("Regression coef: \n", regr.coef\_)  
print("Regression intercept: \n", regr.intercept\_)  
# Середня абсолютна похибка  
print("Mean absolute error :", round(mean\_absolute\_error(diabetes\_y\_test, diabetes\_y\_pred), 2))  
# The mean squared error  
print("Mean squared error: %.2f" % mean\_squared\_error(diabetes\_y\_test, diabetes\_y\_pred))  
# The coefficient of determination: 1 is perfect prediction  
print("R2 score: %.2f" % r2\_score(diabetes\_y\_test, diabetes\_y\_pred))  
fig, ax = plt.subplots()  
ax.scatter(diabetes\_y\_test, diabetes\_y\_pred, edgecolors=(0, 0, 0))  
ax.plot([diabetes\_y.min(), diabetes\_y.max()], [diabetes\_y.min(), diabetes\_y.max()], 'k--', lw=4)  
ax.set\_xlabel('Виміряно')  
ax.set\_ylabel('Передбачено')  
plt.show()





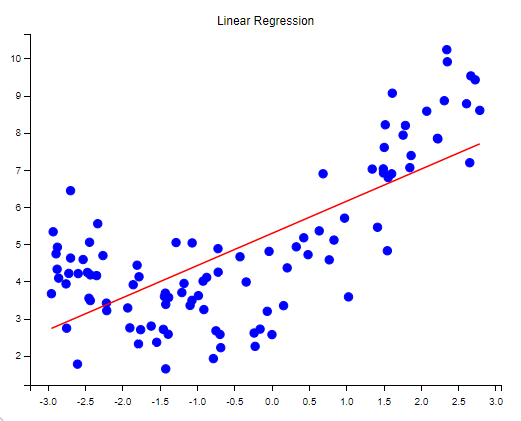
**Завдання 2.5**

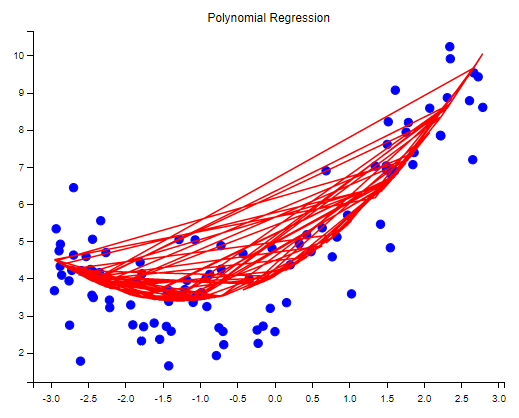
Самостійна побудова регресії



import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error, r2\_score  
from sklearn.model\_selection import train\_test\_split  
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)  
X = X.reshape(-1, 1)  
Y = y.reshape(-1, 1)  
lin = LinearRegression()  
lin.fit(X, y)  
poly = PolynomialFeatures(degree=2)  
X\_poly = poly.fit\_transform(X)  
poly.fit(X\_poly, y)  
lin2 = LinearRegression()  
lin2.fit(X\_poly, y)  
Y\_NEW = lin2.predict(X\_poly)  
r2 = r2\_score(Y, Y\_NEW)  
print('R2: ', r2)  
# Visualising the Linear Regression results  
plt.scatter(X, y, color='blue')  
plt.plot(X, lin.predict(X), color='red')  
plt.title('Linear Regression')  
plt.show()  
# Visualising the Polynomial Regression results  
plt.scatter(X, y, color='blue')  
plt.plot(X, lin2.predict(poly.fit\_transform(X)), color='red')  
plt.title('Polynomial Regression')  
plt.show()



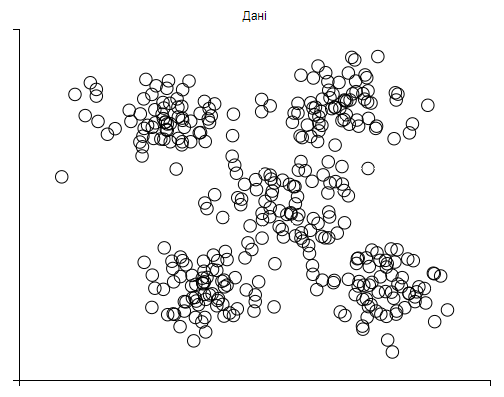


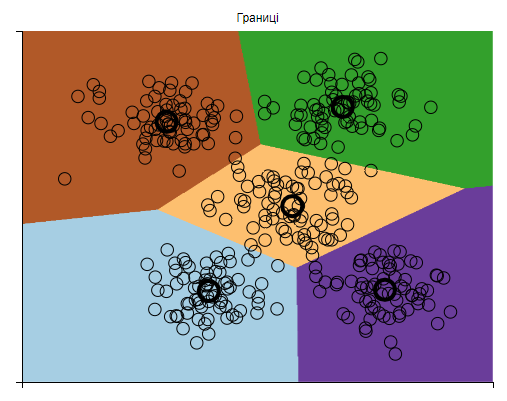


**Завдання 2.7**

Кластеризація даних за допомогою методу k-середніх

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
num\_clusters = 5  
plt.figure()  
plt.scatter(X[:, 0], X[:, 1], marker='o',  
 facecolors='none', edgecolors='black', s=80)  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max()+1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max()+1  
plt.title('Дані')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()  
kmeans = KMeans(init='k-means++', n\_clusters=num\_clusters, n\_init=10)  
kmeans.fit(X)  
step\_size = 0.01  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max()+1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max()+1  
x\_vals, y\_vals = np.meshgrid(  
 np.arange(x\_min, x\_max, step\_size), np.arange(y\_min, y\_max, step\_size))  
output = kmeans.predict(np.c\_[x\_vals.ravel(), y\_vals.ravel()])  
output = output.reshape(x\_vals.shape)  
plt.figure()  
plt.clf()  
plt.imshow(output, interpolation='nearest', extent=(x\_vals.min(), x\_vals.max(  
), y\_vals.min(), y\_vals.max()), cmap=plt.cm.Paired, aspect='auto', origin='lower')  
plt.scatter(X[:, 0], X[:, 1], marker='o',  
 facecolors='none', edgecolors='black', s=80)  
cluster\_center = kmeans.cluster\_centers\_  
plt.scatter(cluster\_center[:, 0], cluster\_center[:, 1], marker='o',  
 s=210, linewidths=4, color='black', zorder=12, facecolors='none')  
x\_min, x\_max = X[:, 0].min() - 1, X[:, 0].max()+1  
y\_min, y\_max = X[:, 1].min() - 1, X[:, 1].max()+1  
plt.title('Границі')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()





**Завдання 2.8**

Кластеризація K-середніх для набору даних Iris

import matplotlib.pyplot as plt  
from sklearn import datasets  
from sklearn.cluster import KMeans  
from sklearn.metrics import pairwise\_distances\_argmin  
import numpy as np  
iris = datasets.load\_iris()  
X = iris.data[:, :2]  
Y = iris.target  
kmeans = KMeans(n\_clusters=Y.max() + 1, init='k-means++', n\_init=10, max\_iter=300,  
 tol=0.0001, verbose=0, random\_state=None, copy\_x=True)  
kmeans.fit(X)  
y\_pred = kmeans.predict(X)  
print("n\_clusters: 3, n\_init: 10, max\_iter: 300, tol: 0.0001, verbose: 0, random\_state: None, copy\_x: True")  
print(y\_pred)  
plt.figure()  
plt.scatter(X[:, 0], X[:, 1], c=y\_pred, s=50, cmap='viridis')  
centers = kmeans.cluster\_centers\_  
plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5)  
plt.show()  
def find\_clusters(X, n\_clusters, rseed=2):  
 rng = np.random.RandomState(rseed)  
 i = rng.permutation(X.shape[0])[:n\_clusters]  
 centers = X[i]  
 while True:  
 labels = pairwise\_distances\_argmin(X, centers)  
 new\_centers = np.array([X[labels == i].mean(0) for i in range(n\_clusters)])  
 if np.all(centers == new\_centers):  
 break  
 centers = new\_centers  
 return centers, labels  
print("using find\_clusters():")  
centers, labels = find\_clusters(X, 3)  
print("n\_clusters: 3, rseed: 2")  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()  
centers, labels = find\_clusters(X, 3, rseed=0)  
print("n\_clusters: 3, rseed: 0")  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()  
labels = KMeans(3, random\_state=0).fit\_predict(X)  
print("n\_clusters: 3, rseed: 0")  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.show()

