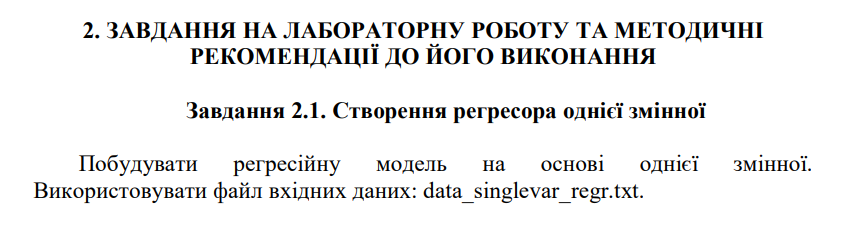
**ЛАБОРАТОРНА РОБОТА № 3**

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО

НАВЧАННЯ

**Посилання на GitHub: https://github.com/FrancIwanicki/OAI.git**

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

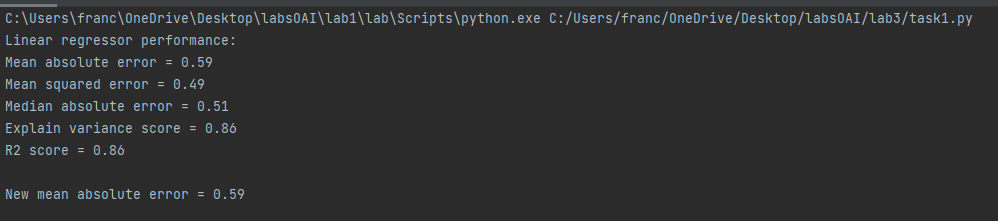


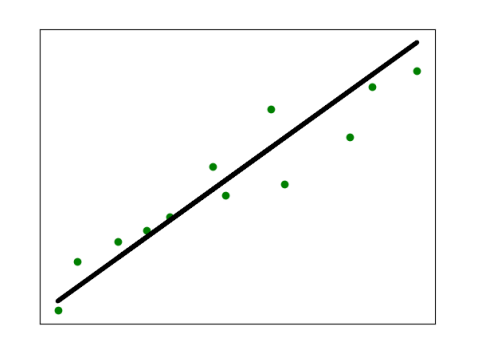
**Лістинг програми:**

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]  
  
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)  
  
# 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'  
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))

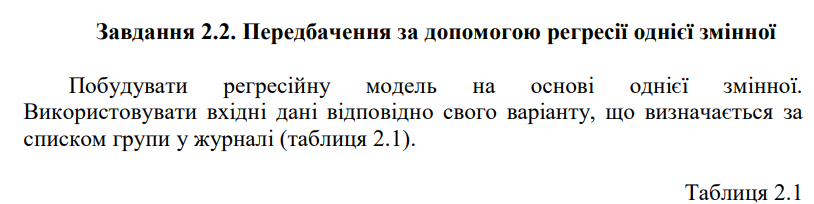
**Результат виконання:**

****



Регресійний аналіз за допомогою лінійної регресії

**Висновок:** Модель лінійної регресії була піддана тренуванню на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка – 0.59, середня квадратична різниця – 0.49, медіана абсолютних помилок – 0.51, оцінка поясненої дисперсії – 0.86. При відновленні моделі з використанням збереженого файлу нова абсолютна помилка також дорівнює 0.59, це означає, що збережена та завантажена модель демонструють == результати.



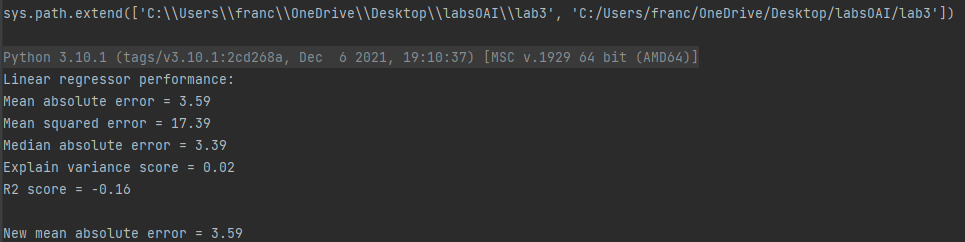


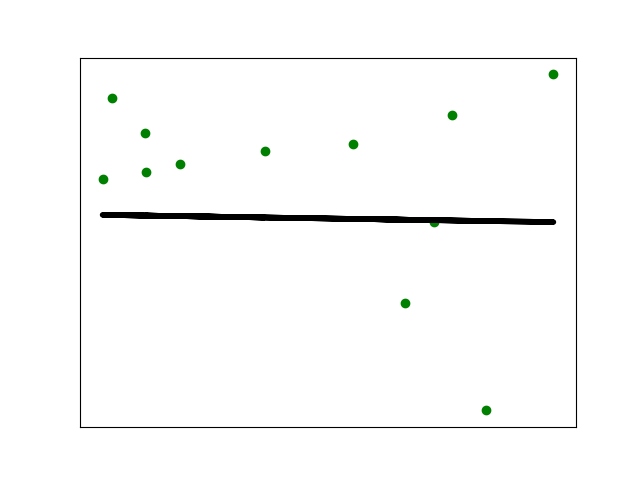
Варіант 3 файл: data\_regr\_3.txt

**Лістинг програми:**

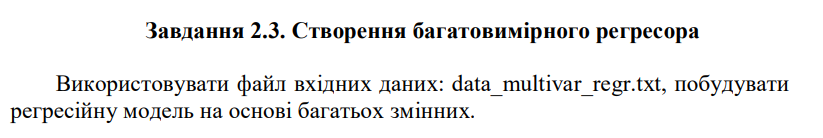
import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
input\_file = 'data\_regr\_3.txt'  
  
data = np.loadtxt(input\_file, delimiter=',')  
X, y = data[:, :-1], data[:, -1]  
  
num\_training = int(0.8 \* len(X))  
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 = 'model2.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))

**Результат** **виконання**:





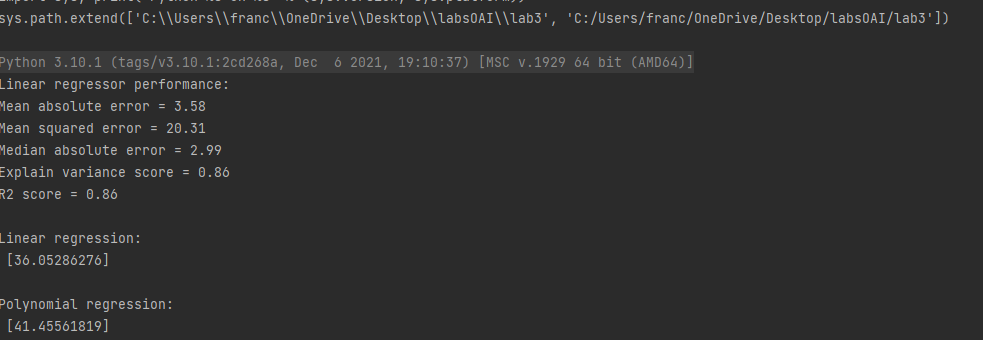
**Висновок:** Модель лінійної регресії була піддана тренуванню на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка – 3.59, середня квадратична різниця – 17.39 (свідчить про великі відхилення у прогнозах), медіана абсолютних помилок – 3.39, оцінка поясненої дисперсії – 0.02 (прогнози значно відрізняються від фактичних значень). Нова абсолютна помилка для моделі, відновленої з використанням збереженого файлу, також дорівнює – 3.59, що підтверджує, що збережена і завантажена модель повертає ті ж самі результати.



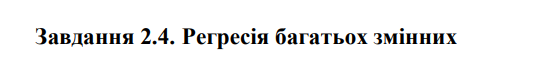
**Лістинг програми:**

from math import degrees  
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("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",  
 linear\_regressor.predict(datapoint))  
print("\nPolynomial regression:\n",  
 poly\_linear\_model.predict(poly\_datapoint))

**Результат виконання:**

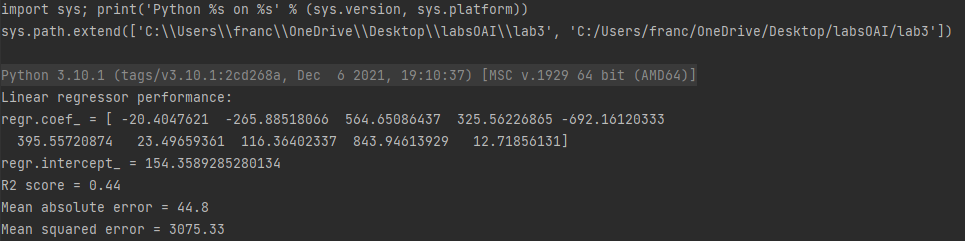
****

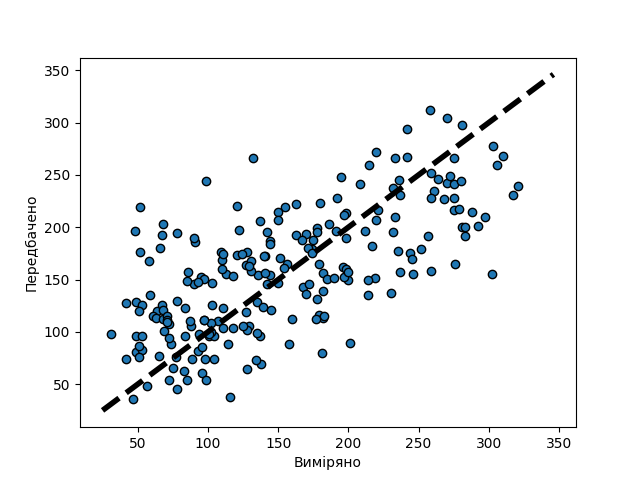
**Висновок:** Середня абсолютна помилка становить – 3.58,середня квадратична різниця дорівнює – 20.31 (значення вище ніж попереднє,що свідчить про великі відхолиння у прогнозах) ,медіана абсолютних помилок дорівнює – 2.99, оцінка поясненої дисперсії становить – 0.86 (прогнози схожі до фактичних значень.Для поліноміальної регресії використовується поліном 10-го ступеня, для вхідного значення [7.75, 6.35, 5.56] , поліноміальна регресійна модель прогнозує значення – 41.46, в той час як лінійна регресія прогнозує значення – 36.05.Обидві моделі мають схожі значення показників,що свідчить про те,що обидві моделі надають приблизно однакові результати.

****

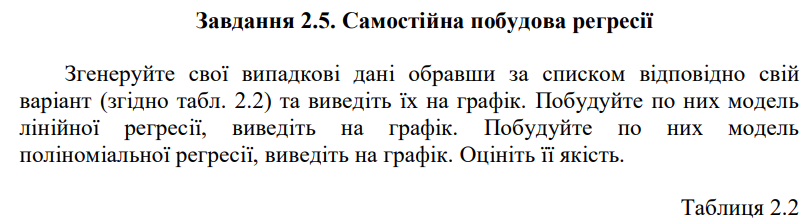
**Лістинг програми:**

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  
from sklearn.model\_selection import train\_test\_split  
  
diabetes = datasets.load\_diabetes()  
X = diabetes.data  
y = diabetes.target  
  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=0)  
  
regr = linear\_model.LinearRegression()  
regr.fit(X\_train, y\_train)  
y\_pred = regr.predict(X\_test)  
  
print("Linear regressor performance:")  
print("regr.coef\_ =", regr.coef\_)  
print("regr.intercept\_ =", regr.intercept\_)  
print("R2 score =", round(r2\_score(y\_test, y\_pred), 2))  
print("Mean absolute error =", round(mean\_absolute\_error(y\_test, y\_pred), 2))  
print("Mean squared error =", round(mean\_squared\_error(y\_test, y\_pred), 2))  
  
fig, ax = plt.subplots()  
ax.scatter(y\_test, y\_pred, 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 виконання:  
**

****

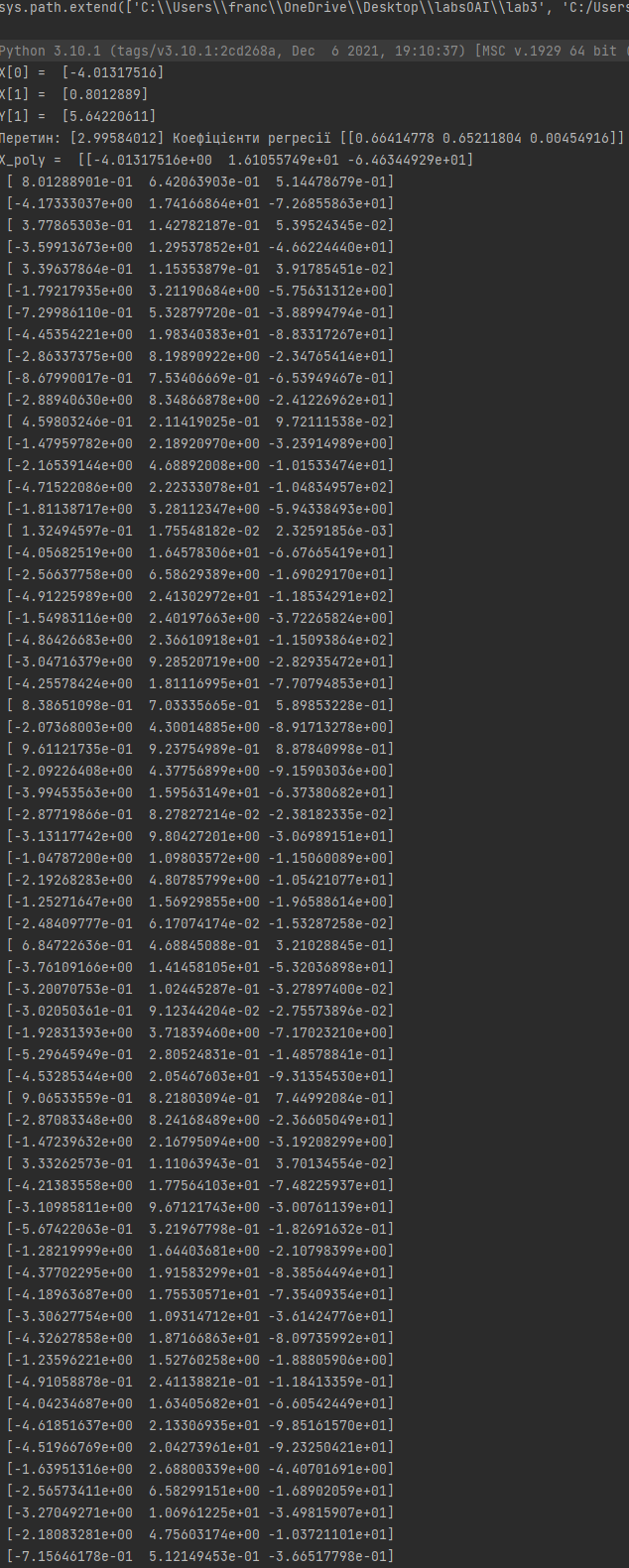
**Висновок:** Було виконано лінійну регресію для набору даних «Diabetes» та було отримано такі результати якості: **Коефіцієнти регресії** представлються як масив чисел,вони вказують на вагу кожної ознаки [-20.4047621,-265.88518, 564.65086437,325.56226865,692.16120333,395.55720874,23.49659361,116.36402337 843.94613929,12.71856131], **Перетин** рівний – 154.36 і представляє відсоток,на який зміщується пряма регресії, **Оцінка R2** дорівнює – 0.44 (модель пояснює близкьо 44% варіації в цільовій змінній), Середня Абсолютна різниця становить 44.8,Середня Квадратична Оцінка дорівнює – 3075.33 (показує різницю між прогнозованим і фактичним значеннями). Модель має обмежену ефективність,що підтверджує значення R2)

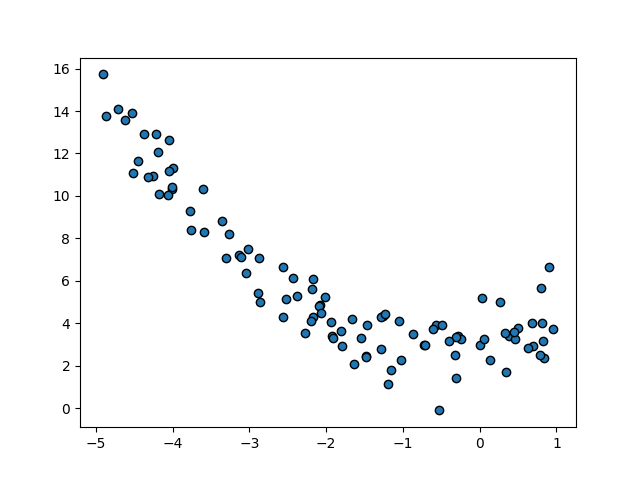


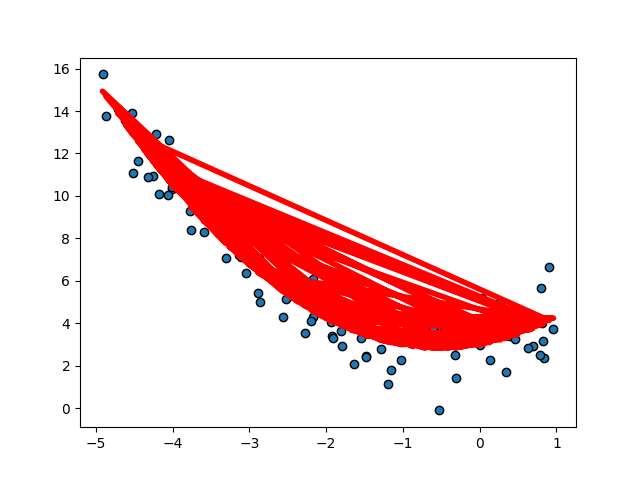
**Лістинг програми:**

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import linear\_model  
from sklearn.preprocessing import PolynomialFeatures  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 5  
y = 0.7 \* X \*\* 2 + X + 3 + np.random.randn(m, 1)  
  
fig, ax = plt.subplots()  
ax.scatter(X, y, edgecolors=(0, 0, 0))  
plt.show()  
poly\_features = PolynomialFeatures(degree=3, include\_bias=False)  
X\_poly = poly\_features.fit\_transform(X)  
print("X[0] = ",X[0])  
print("X[1] = ",X[1])  
print("Y[1] = ",y[1])  
lin\_reg = linear\_model.LinearRegression()  
lin\_reg.fit(X\_poly, y)  
print("Перетин:",lin\_reg.intercept\_,"Коефіцієнти регресії", lin\_reg.coef\_)  
y\_pred = lin\_reg.predict(X\_poly)  
  
print("X\_poly = ",X\_poly)  
  
  
fig, ax = plt.subplots()  
ax.scatter(X, y, edgecolors=(0, 0, 0))  
plt.plot(X, y\_pred, color='red', linewidth=4)  
plt.show()

**Результат виконання:**

****

****



Графіки моделі

**Висновок:**Математичне рівняння для даної моделі:

y= 0.7x12+1.0x1+3+гауссов шум

Отримана модель регресії з передбаченими коефіцієнтами:

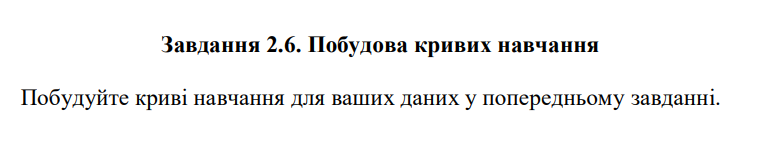
y =3.07618256+1.09246979x1+0.61181343x2−0.02138683x3

Результати регресії:  
X[0] = [-1.59591606]

X[1] = [-0.48231179] – значення Х для першого спостереження.

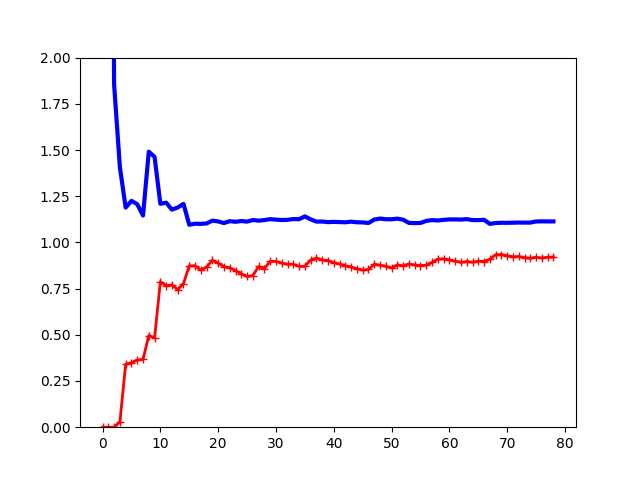
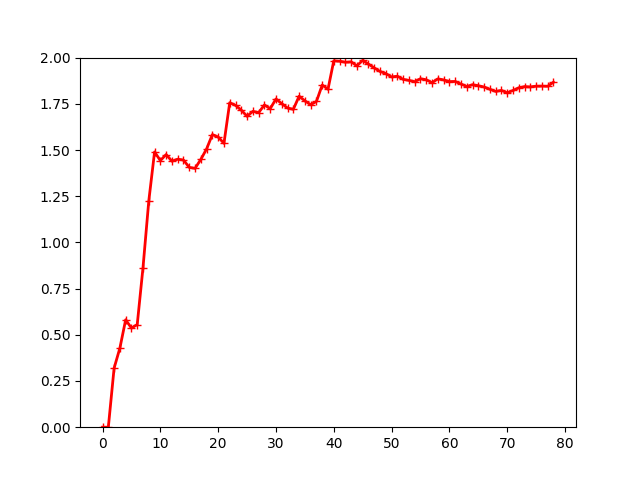
Y[1] = [3.4939998] – відповідне значення Y для першого спостереження.

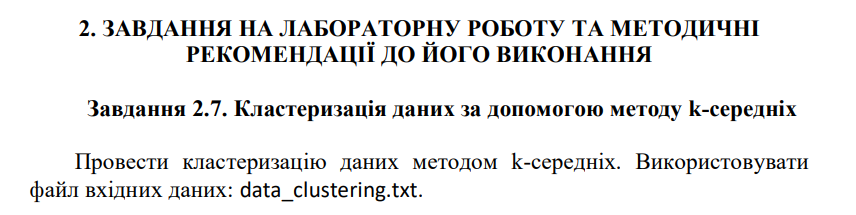
Перетин: [2.68172353] Коефіцієнти регресії [0.97095503,0.80659551,0.02275606]



**Лістинг програми:**

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn import linear\_model  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.pipeline import Pipeline  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 5  
y = 0.7 \* X \*\* 2 + X + 3 + np.random.randn(m, 1)  
  
def ivanytskyi\_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))  
 fig, ax = plt.subplots()  
 plt.ylim(0, 2)  
 ax.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label='train')  
 ax.plot(np.sqrt(val\_errors), "b-", linewidth=3, label='val')  
 plt.show()  
  
  
lin\_reg = linear\_model.LinearRegression()  
ivanytskyi\_plot\_learning\_curves(lin\_reg, X, y)  
  
polynomial\_regression = Pipeline([  
 ('poly\_features', PolynomialFeatures(degree=2, include\_bias=False)),  
 ('lin\_reg', linear\_model.LinearRegression()),  
])  
  
ivanytskyi\_plot\_learning\_curves(polynomial\_regression, np.array(X).reshape(-1, 1), y)

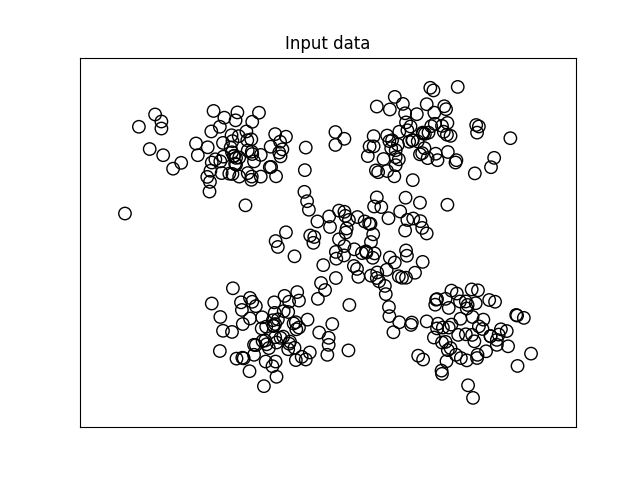
**Результат виконання: ** 

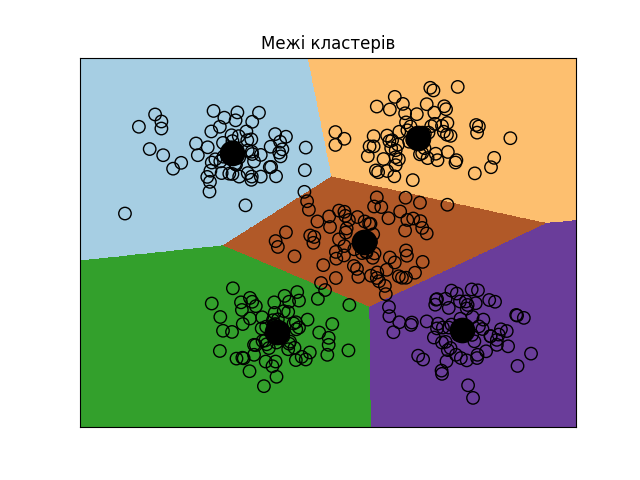
****

**Лістинг програми:**

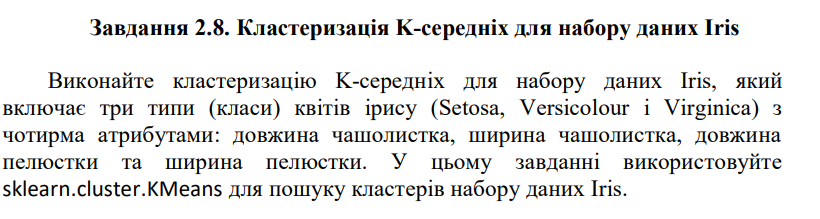
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
num\_clusters = 5  
  
plt.figure()  
plt.scatter(X[:, 0], X[:, 1], marker='o', facecolors='none', edgecolors='black', 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('Input data')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
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\_centers = kmeans.cluster\_centers\_  
plt.scatter(cluster\_centers[:, 0], cluster\_centers[:, 1],  
 marker='o', s=210, linewidths=4, color='black',  
 zorder=12, facecolors='black')  
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('Межі кластерів')  
plt.xlim(x\_min, x\_max)  
plt.ylim(y\_min, y\_max)  
plt.xticks(())  
plt.yticks(())  
plt.show()

**Результат виконання:**

****

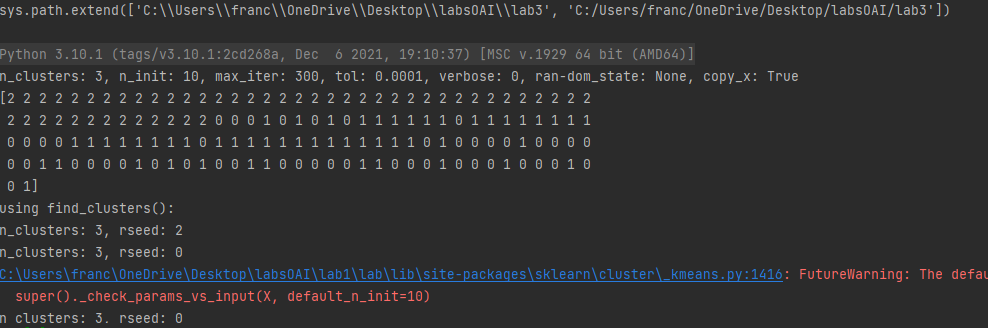
****

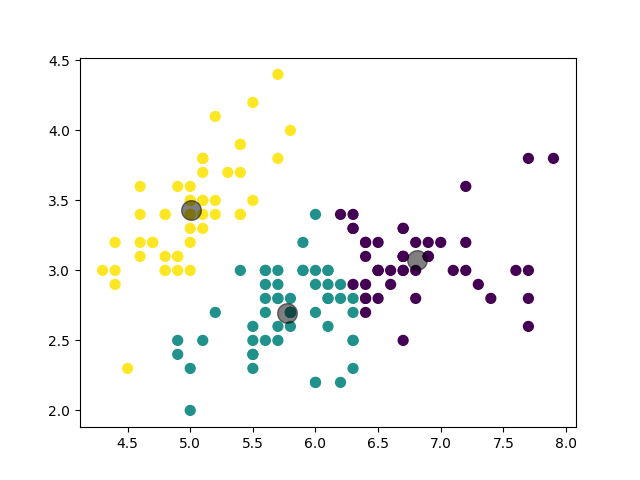
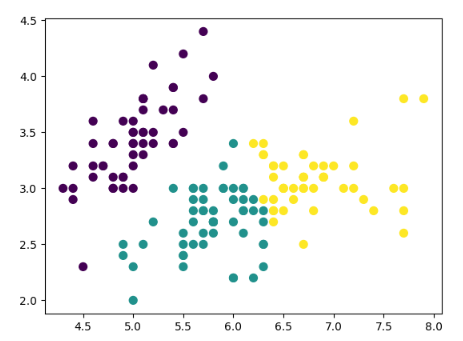
**Висновок:** Ми використали агоритм K-Means для кластеризації даних, було використано 5 кластерів, на графіку показано розділення простору на кластери та центри кластерів.

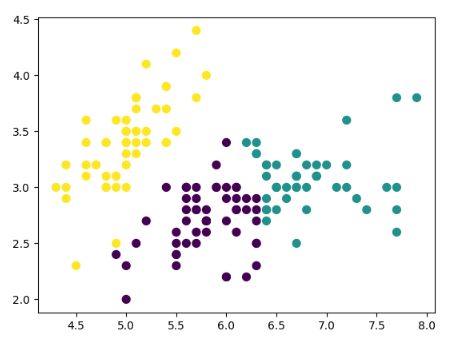


Лістинг програми:

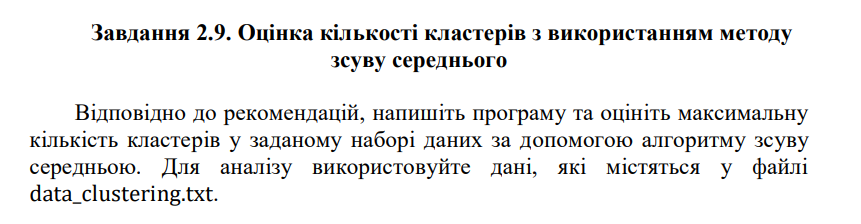
import matplotlib.pyplot as plt  
from sklearn import datasets  
from sklearn.cluster import KMeans  
from sklearn.metrics import pairwise\_distances\_argmin  
import numpy as np  
iris = datasets.load\_iris()  
X = iris.data[:, :2]  
Y = iris.target  
kmeans = KMeans(n\_clusters=Y.max() + 1, init='k-means++', 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, ran-dom\_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 ivanytskyi\_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 = ivanytskyi\_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 = ivanytskyi\_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()

Результат виконання:  


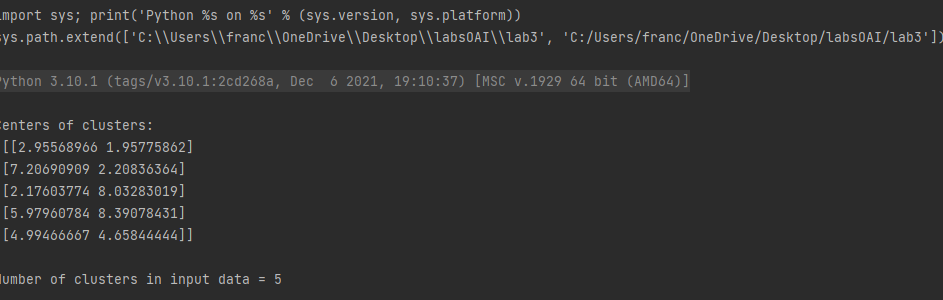


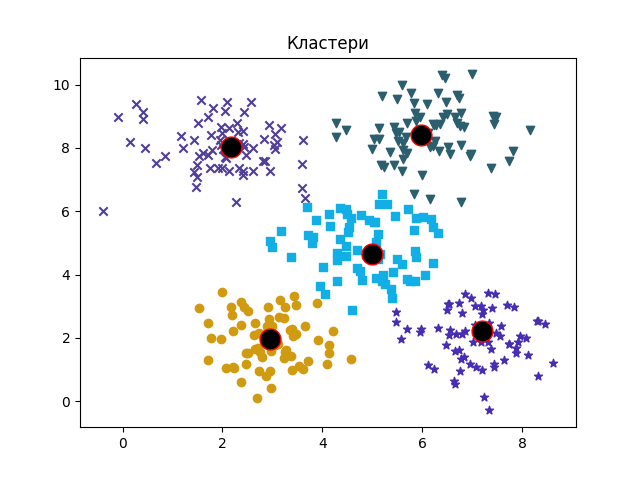
**Висновок:** K-Means алгоритм може бути використаний для кластеризації даних, і ви можете вибирати різні параметри та початкові умови для отримання різних результатів кластеризації. Точна кількість кластерів та початкові умови можуть бути важливими для правильного розділення даних на кластери

****

**Лістинг програми:**

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from itertools import cycle  
  
# Завантаження даних  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
# Оцінка ширини вікна для X  
bandwidth\_X = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))  
  
# Кластеризація даних методом зсуву середнього  
meanshift\_model = MeanShift(bandwidth=bandwidth\_X, bin\_seeding=True)  
meanshift\_model.fit(X)  
  
# Витягування центрів кластерів  
cluster\_centers = meanshift\_model.cluster\_centers\_  
print('\nCenters of clusters:\n', cluster\_centers)  
  
# Оцінка кількості кластерів  
labels = meanshift\_model.labels\_  
num\_clusters = len(np.unique(labels))  
print("\nNumber of clusters in input data =", num\_clusters)  
  
# Відображення на графіку точок та центрів кластерів  
plt.figure()  
markers = 'o\*xvs'  
for i, marker in zip(range(num\_clusters), markers):  
 # Відображення на графіку точок, що належать поточному кластеру  
 plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker,  
 color=np.random.rand(3,))  
  
 # Відображення на графіку центру кластера  
 cluster\_center = cluster\_centers[i]  
 plt.plot(cluster\_center[0], cluster\_center[1], marker='o',  
 markerfacecolor='black', markeredgecolor='red',  
 markersize=15)  
  
plt.title('Кластери')  
plt.show()

**Результат виконання:  
**

****

**Висновок:** Mean Shift алгоритм ефективно використовується для групування даних. В результаті його застосування було сформовано п'ять кластерів, а координати їх центрів були виведені на екран і зображені на графіку. Кожен кластер був відзначений окремим кольором для полегшення візуалізації та аналізу структури даних.

**Висновок до лабораторної роботи:** використовуючи спеціалізовані бібліотеки і мову програмування Python дослідив методи регресії та неконтрольованої класифікації даних у машинному навчанні.