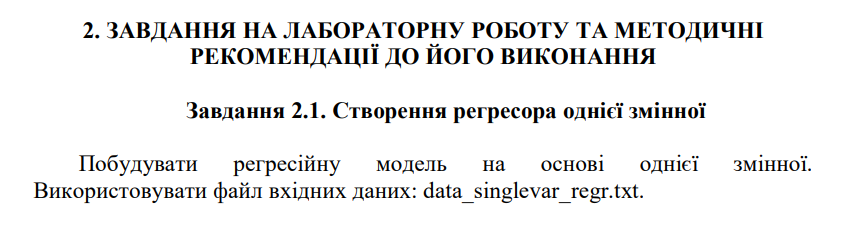
**ЛАБОРАТОРНА РОБОТА № 3**

ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО

НАВЧАННЯ

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

GitHub репозиторій: **https://github.com/SMTH666/OAI-Lab**



**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:**

**Изображение выглядит как текст, снимок экрана, Шрифт

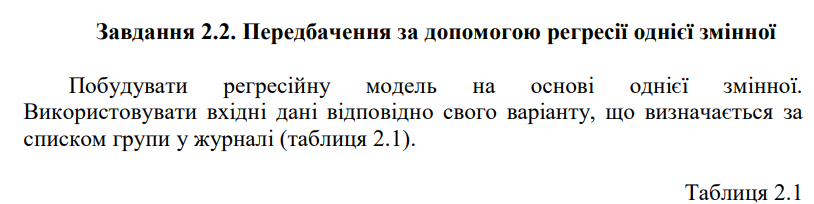
Автоматически созданное описание**

Изображение выглядит как линия, снимок экрана, диаграмма, График

Автоматически созданное описание

Регресійний аналіз за допомогою лінійної регресії

**Висновок:** Модель лінійної регресії була навчена на 80% даних та протестована на залишкових 20%. У цьому тесті були отримані наступні результати: середня абсолютна помилка - 0.59, середня квадратична різниця - 0.49, медіана абсолютних помилок - 0.51, оцінка поясненої дисперсії - 0.86. Після того, як модель була збережена та відновлена з використанням збереженого файлу, нова абсолютна помилка також склала 0.59. Це свідчить про те, що збережена та відновлена модель надають ті ж самі результати, що й оригінал.





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))

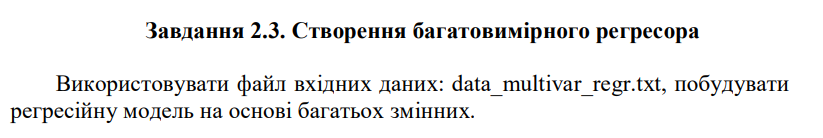
**Результат** **виконання**:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описаниеИзображение выглядит как снимок экрана, линия, Прямоугольник, диаграмма

Автоматически созданное описание

**Висновок:** Модель лінійної регресії була навчена на 80% доступних даних, а потім перевірена на залишкових 20%. У цьому тесті були отримані наступні метрики: середня абсолютна помилка дорівнює 3.59, середня квадратична різниця становить 17.39 (вище, ніж попередня, вказуючи на значні відхилення у прогнозах). Медіана абсолютних помилок складає 3.39, оцінка поясненої дисперсії дорівнює 0.02 (вказуючи на те, що прогнози значно відрізняються від фактичних значень). Після того, як модель була збережена та відновлена з використанням збереженого файлу, нова абсолютна помилка становить 3.59. Це свідчить про те, що збережена і відновлена модель дає ті ж самі результати, що й оригінал.



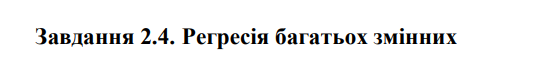
**Лістинг програми:**

from math import degrees  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
from sklearn.preprocessing import PolynomialFeatures  
from joblib import dump, load  
  
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))  
  
# Збереження моделі лінійної регресії  
dump(linear\_regressor, 'linear\_regressor\_model.joblib')  
  
# Відновлення моделі лінійної регресії  
loaded\_linear\_regressor = load('linear\_regressor\_model.joblib')  
  
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)  
  
# Збереження та відновлення моделі поліноміальної регресії  
dump(poly\_linear\_model, 'poly\_linear\_model.joblib')  
loaded\_poly\_linear\_model = load('poly\_linear\_model.joblib')  
  
print("\nLinear regression:\n", loaded\_linear\_regressor.predict(datapoint))  
print("\nPolynomial regression:\n", loaded\_poly\_linear\_model.predict(poly\_datapoint))

**Результат виконання:**

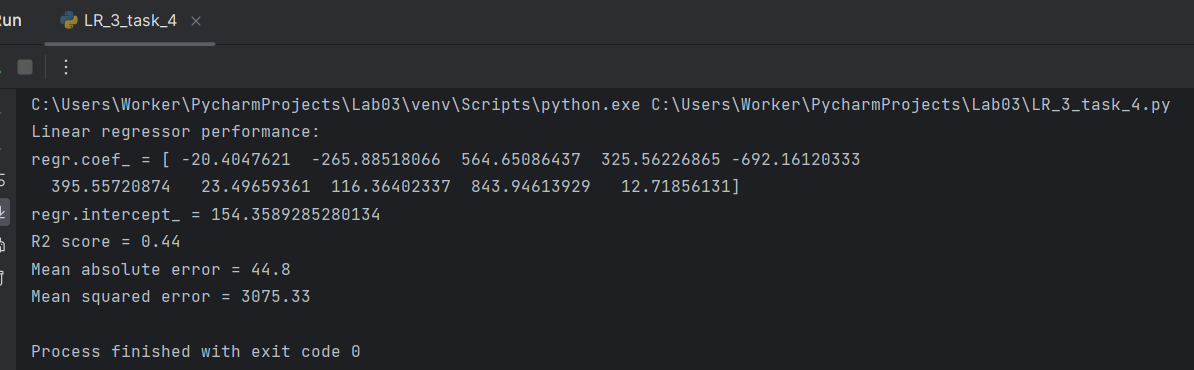
**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Висновок:** Середнє абсолютне відхилення складає 3.58, середньоквадратична різниця - 20.31 (це значення вище, ніж попереднє, що вказує на значні розбіжності у прогнозах). Медіана абсолютних помилок становить 2.99, оцінка поясненої дисперсії - 0.86 (прогнози схожі на фактичні значення). Для поліноміальної регресії використовується поліном 10-го ступеня, і для вхідного значення [7.75, 6.35, 5.56] поліноміальна регресійна модель передбачає значення 41.46, в той час як лінійна регресія передбачає значення 36.05. Обидві моделі мають схожі показники, що свідчить про те, що вони надають приблизно однакові результати.****

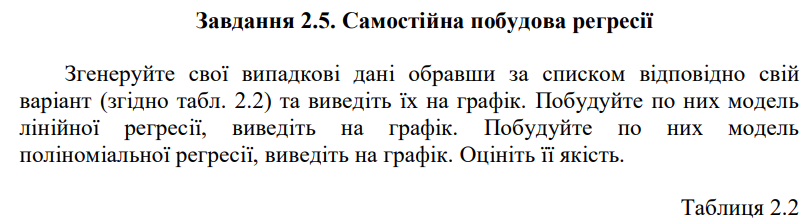
**Лістинг програми:**

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  
from sklearn.preprocessing import StandardScaler  
  
diabetes = datasets.load\_diabetes()  
X = diabetes.data  
y = diabetes.target  
  
# Масштабування ознак для поліпшення роботи лінійної регресії  
scaler = StandardScaler()  
X = scaler.fit\_transform(X)  
  
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("Coefficients:", regr.coef\_)  
print("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:  
**

**Изображение выглядит как текст, снимок экрана, диаграмма, линия

Автоматически созданное описание**

**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 -0.97055556 -12.64686835 26.85770273 15.48541795 -32.92275106 18.81473774 1.11761899 5.53487216 40.14242421 0.60496027],**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 

**Лістинг програми:**

import matplotlib.pyplot as plt  
import numpy as np  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LinearRegression  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 4  
y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
fig, ax = plt.subplots()  
ax.scatter(X, y, edgecolors=(0, 0, 0))  
plt.show()  
  
# Трансформація ознак за допомогою поліноміальних ознак  
poly\_features = PolynomialFeatures(degree=2, 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])  
  
# Навчання лінійної регресії на поліноміальних ознаках  
lin\_reg = LinearRegression()  
lin\_reg.fit(X\_poly, y)  
  
print("Перетин:", lin\_reg.intercept\_)  
print("Коефіцієнти регресії:", lin\_reg.coef\_)  
  
# Передбачення на основі навченої моделі  
y\_pred = lin\_reg.predict(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:**

**Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

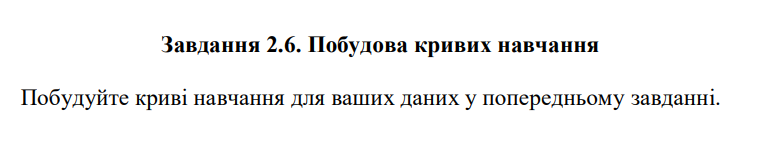
**Изображение выглядит как снимок экрана, диаграмма

Автоматически созданное описание**

**Изображение выглядит как кровать, дизайн

Автоматически созданное описание с низким доверительным уровнем**

Графіки моделі



**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  
  
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))  
 fig, ax = plt.subplots()  
 ax.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label='train')  
 ax.plot(np.sqrt(val\_errors), "b-", linewidth=3, label='val')  
 plt.ylim(0, 2)  
 plt.legend(loc='upper right')  
 plt.show()  
  
m = 100  
X = 6 \* np.random.rand(m, 1) - 4  
y = 0.5 \* X \*\* 2 + X + 2 + np.random.randn(m, 1)  
  
  
lin\_reg = linear\_model.LinearRegression()  
plot\_learning\_curves(lin\_reg, X, y)  
  
polynomial\_regression = Pipeline([  
 ('poly\_features', PolynomialFeatures(degree=2, include\_bias=False)),  
 ('lin\_reg', linear\_model.LinearRegression()),  
])  
  
plot\_learning\_curves(polynomial\_regression, X, y)

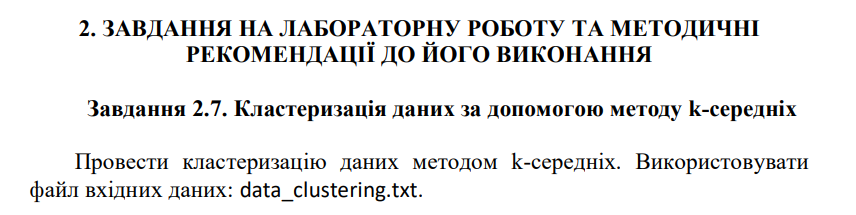
**Результат виконання:**

Изображение выглядит как текст, График, линия, диаграмма

Автоматически созданное описание

Изображение выглядит как текст, диаграмма, График, линия

Автоматически созданное описание

****

**Лістинг програми:**

import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn import metrics  
  
# Завантаження даних  
X = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
# Кількість кластерів  
num\_clusters = 5  
  
# Візуалізація вхідних даних  
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(())  
plt.show()

**Результат виконання:**

**Изображение выглядит как шаблон

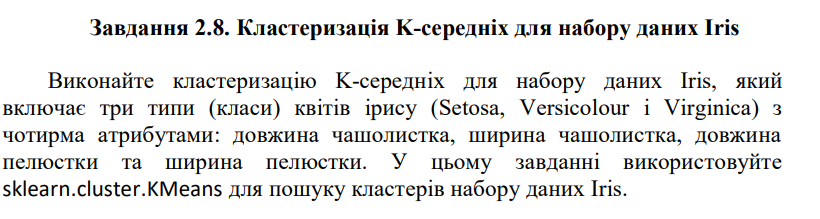
Автоматически созданное описание**

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()

**Изображение выглядит как текст, снимок экрана, графический дизайн, Графика

Автоматически созданное описание**

**Висновок:** Я використав 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 з параметрами  
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)  
  
# Візуалізація результатів KMeans  
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.title('KMeans Clustering')  
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  
  
# Застосування власної функції для кластеризації  
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.title('Custom Clustering (n\_clusters: 3, rseed: 2)')  
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.title('Custom Clustering (n\_clusters: 3, rseed: 0)')  
plt.show()  
  
# Використання KMeans з вказаною кількістю кластерів та random\_state  
labels = KMeans(3, random\_state=0).fit\_predict(X)  
print("using KMeans():")  
print("n\_clusters: 3, random\_state: 0")  
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap='viridis')  
plt.title('KMeans Clustering (n\_clusters: 3, random\_state: 0)')  
plt.show()

Результат виконання:  
Изображение выглядит как снимок экрана, текст

Автоматически созданное описание

Изображение выглядит как снимок экрана, диаграмма, Красочность, линия

Автоматически созданное описание

Изображение выглядит как снимок экрана, диаграмма, Красочность, текст

Автоматически созданное описание

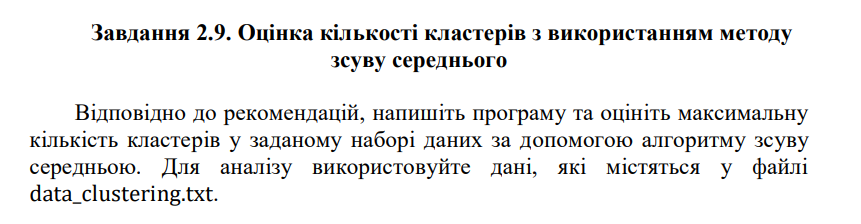
Изображение выглядит как снимок экрана, диаграмма, Красочность, линия

Автоматически созданное описание

Изображение выглядит как снимок экрана, диаграмма, Красочность, линия

Автоматически созданное описание

**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), cycle(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()

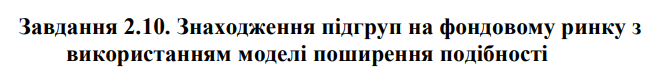
**Результат виконання:  
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, Красочность

Автоматически созданное описание**

**Висновок:** Алгоритм Mean Shift успішно використовується для кластеризації даних. В результаті було отримано п'ять кластерів, і їх центри були виведені на екран та відображені на графіку. Кластери виділені різними кольорами для легкості візуалізації та аналізу структури даних.



Неможливо виконати завдання через відсутність файлу company\_symbol\_mapping.json.

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