**ЗВІТ**

**З лабораторної роботи №3**

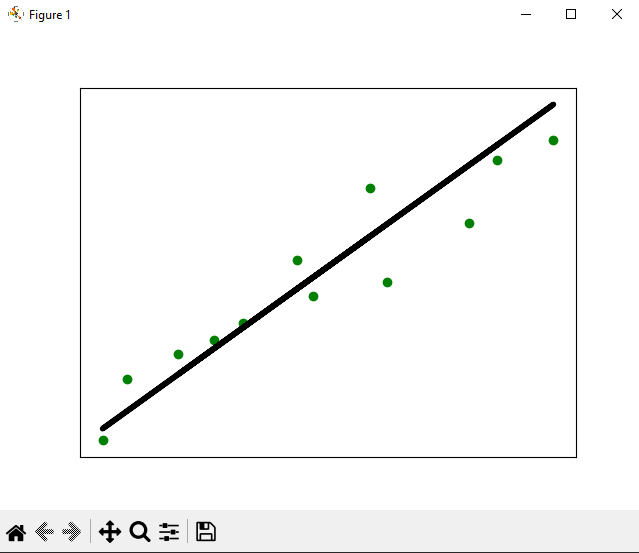
**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ**

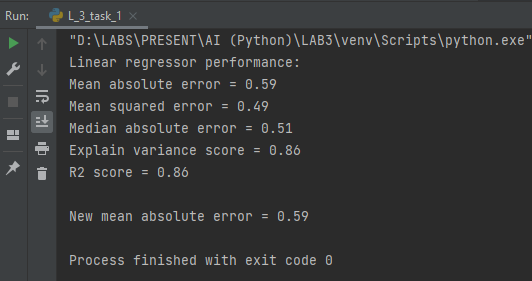
Студента КН-20-1 навчальної групи

Кірія Даніли Олеговича варіант №6

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

**Завдання 1.** Створення регресора однієї змінної



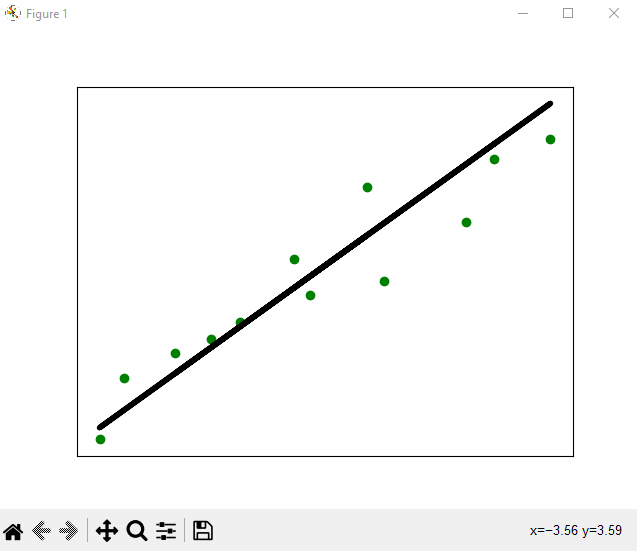


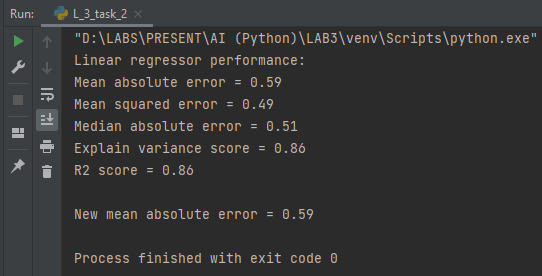
Код програми:

import pickle  
import matplotlib  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
matplotlib.use('TkAgg')  
# 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  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =",  
round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

**Завдання 2.2.** Передбачення за допомогою регресії однієї змінної

Варіант 1

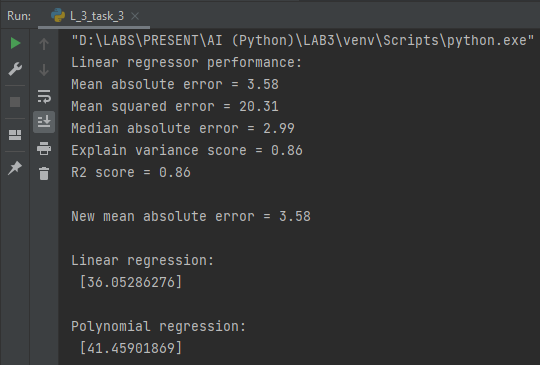




Код програми:

import pickle  
import matplotlib  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
  
matplotlib.use('TkAgg')  
# Вхідний файл, який містить дані  
input\_file = 'data\_regr\_1.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  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =",  
round(sm.mean\_absolute\_error(y\_test, y\_test\_pred\_new), 2))

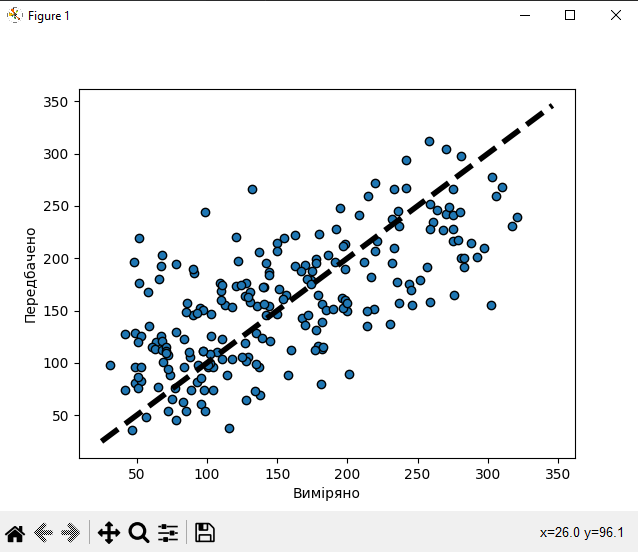
**Завдання 2.3.** Створення багатовимірного регресора

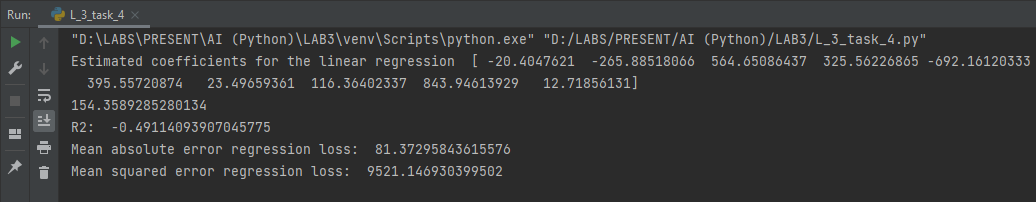


Код програми:

import pickle  
import numpy as np  
from sklearn import linear\_model  
import sklearn.metrics as sm  
import matplotlib.pyplot as plt  
from sklearn import linear\_model  
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("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))  
  
# Завантаження моделі  
y\_test\_pred\_new = regressor.predict(X\_test)  
print("\nNew mean absolute error =",  
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



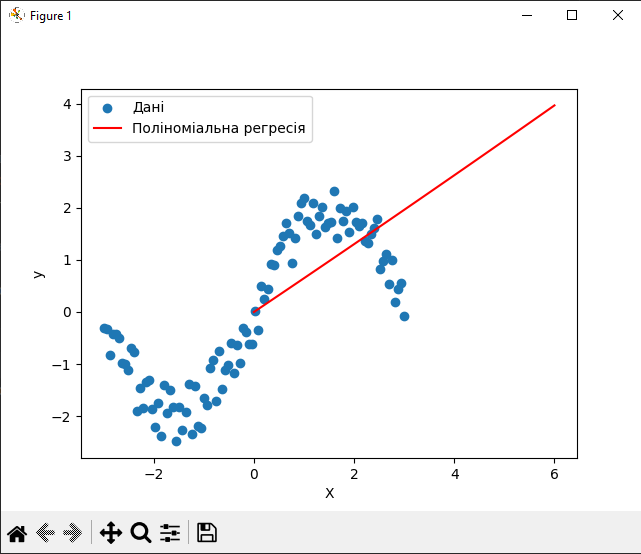
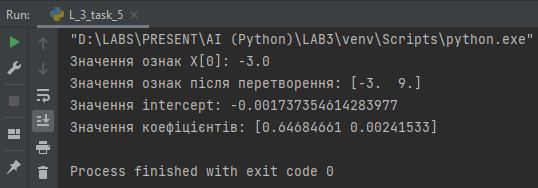


Код програми:

import matplotlib  
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  
  
matplotlib.use('TkAgg')  
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("Estimated coefficients for the linear regression ", regr.coef\_)  
print(str(regr.intercept\_))  
print("R2: ", r2\_score(ytrain, ypred))  
print("Mean absolute error regression loss: ", mean\_absolute\_error(ytrain, ypred))  
print("Mean squared error regression loss: ", mean\_squared\_error(ytrain, 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()

**Завдання 2.5.** Самостійна побудова регресії

Варіант 6

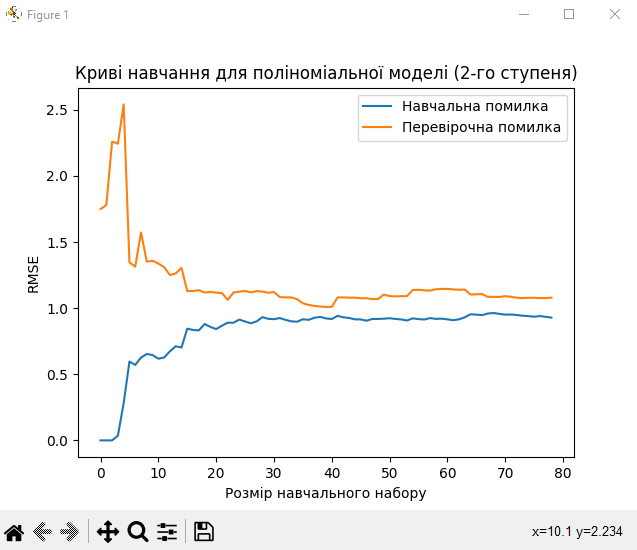
 

Рівняння отриманої поліноміальної моделі регресії має вигляд:  
y = 0.64684661⋅X2 + 0.00241533⋅X + (-0.001737354614283977)

Код програми:

import matplotlib  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LinearRegression  
  
matplotlib.use('TkAgg')  
  
m = 100  
X = np.linspace(-3, 3, m)  
y = 2 \* np.sin(X) + np.random.uniform(-0.6, 0.6, m)  
  
  
# Побудова поліноміальних ознак (квадратичних) для X  
poly\_features = PolynomialFeatures(degree=2, include\_bias=False)  
X\_poly = poly\_features.fit\_transform(X.reshape(-1, 1))  
  
# Виведення значень ознак X[0] та X\_poly на екран  
print("Значення ознак X[0]:", X[0])  
print("Значення ознак після перетворення:", X\_poly[0])  
  
# Підгонка лінійної моделі до розширених даних  
lin\_reg = LinearRegression()  
lin\_reg.fit(X\_poly, y)  
  
# Виведення значень коефіцієнтів полінома  
intercept = lin\_reg.intercept\_  
coefficients = lin\_reg.coef\_  
print("Значення intercept:", intercept)  
print("Значення коефіцієнтів:", coefficients)  
  
# Генерація значень для побудови графіку  
X\_plot = np.linspace(0, 6, 100)  
y\_plot = lin\_reg.predict(poly\_features.transform(X\_plot.reshape(-1, 1)))  
  
# Побудова графіку  
plt.scatter(X, y, label='Дані')  
plt.plot(X\_plot, y\_plot, label='Поліноміальна регресія', color='red')  
plt.xlabel('X')  
plt.ylabel('у')  
plt.legend()  
plt.show()

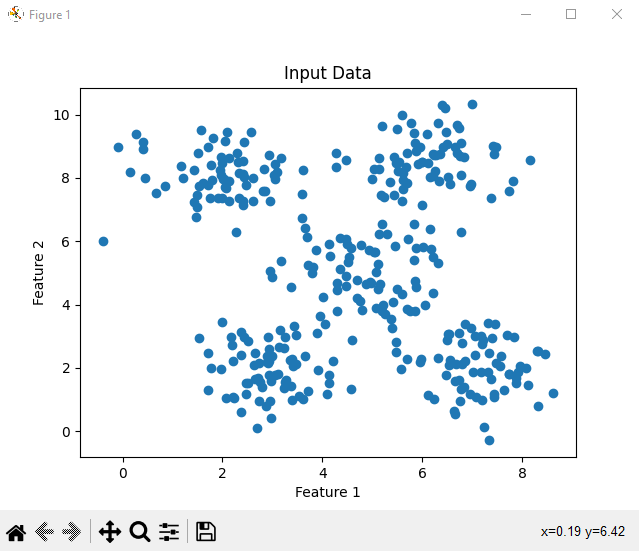
**Завдання 2.6.** Побудова кривих навчання

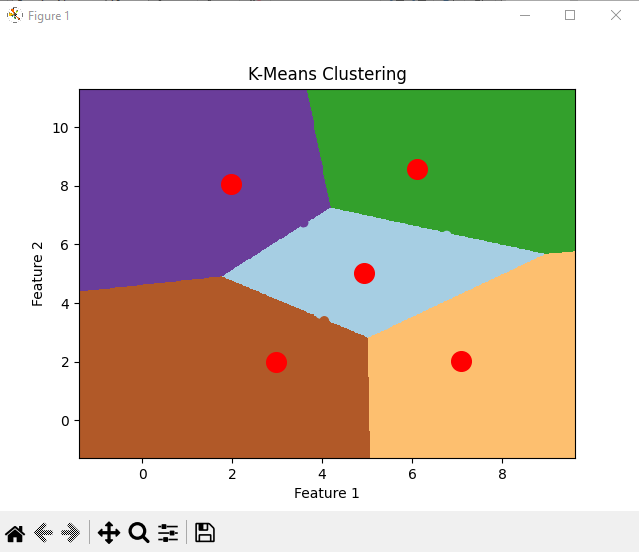


Код програми:

import matplotlib  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.preprocessing import PolynomialFeatures  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import mean\_squared\_error  
from sklearn.model\_selection import train\_test\_split  
  
matplotlib.use('TkAgg')  
  
m = 100  
X = np.linspace(-3, 3, m)  
y = 2 \* np.sin(X) + np.random.uniform(-0.6, 0.6, m)  
  
# Розділимо дані на навчальний та перевірочний набори  
X\_train, X\_val, y\_train, y\_val = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Функція для побудови кривих навчання  
def plot\_learning\_curves(model, X, y, X\_val, y\_val):  
 train\_errors, val\_errors = [], []  
 for m in range(1, len(X)):  
 model.fit(X[:m], y[:m])  
 y\_train\_predict = model.predict(X[:m])  
 y\_val\_predict = model.predict(X\_val)  
 train\_errors.append(mean\_squared\_error(y[:m], y\_train\_predict))  
 val\_errors.append(mean\_squared\_error(y\_val, y\_val\_predict))  
 plt.plot(np.sqrt(train\_errors), label="Навчальна помилка")  
 plt.plot(np.sqrt(val\_errors), label="Перевірочна помилка")  
 plt.legend()  
  
poly\_features = PolynomialFeatures(degree=2, include\_bias=False)  
X\_poly = poly\_features.fit\_transform(X\_train.reshape(-1, 1))  
X\_poly\_val = poly\_features.transform(X\_val.reshape(-1, 1))  
  
lin\_reg = LinearRegression()  
plot\_learning\_curves(lin\_reg, X\_poly, y\_train, X\_poly\_val, y\_val)  
  
plt.xlabel('Розмір навчального набору')  
plt.ylabel('RMSE')  
plt.title('Криві навчання для поліноміальної моделі (2-го ступеня)')  
plt.show()

**Завдання 2.7.** Кластеризація даних за допомогою методу k-середніх



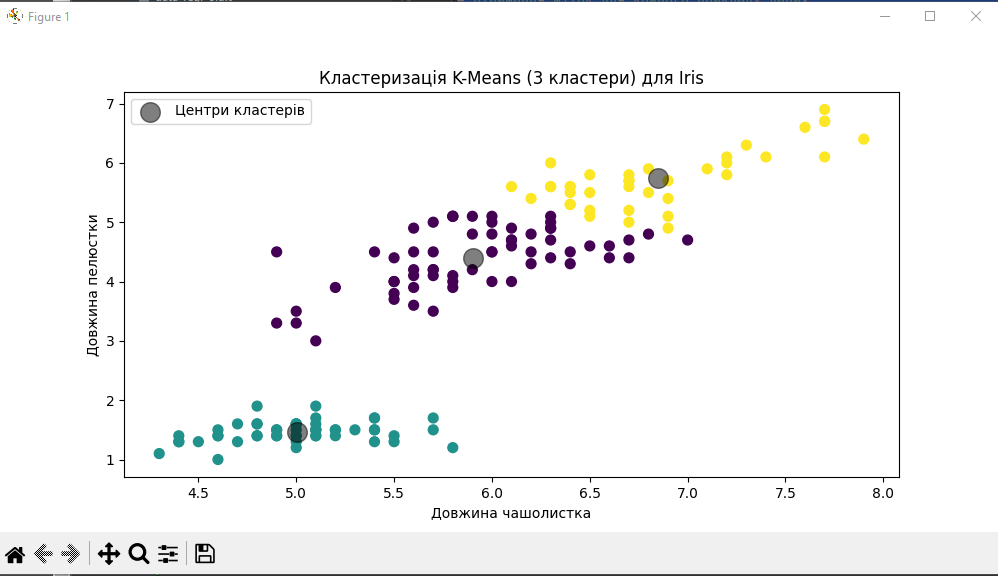


Код програми:

import matplotlib  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
  
matplotlib.use('TkAgg')  
  
# Завантаження вхідних даних  
data = np.loadtxt('data\_clustering.txt', delimiter=',')  
  
# Включення вхідних даних до графіка  
plt.scatter(data[:, 0], data[:, 1])  
plt.title('Input Data')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
plt.show()  
  
# Створення об'єкту КМеаns  
kmeans = KMeans(n\_clusters=5, init='k-means++', n\_init=10, random\_state=0)  
  
# Навчання моделі кластеризації КМеаns  
kmeans.fit(data)  
  
# Визначення кроку сітки  
h = 0.02  
  
# Визначення меж для сітки  
x\_min, x\_max = data[:, 0].min() - 1, data[:, 0].max() + 1  
y\_min, y\_max = data[:, 1].min() - 1, data[:, 1].max() + 1  
xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))  
  
# Передбачення вихідних міток для всіх точок сітки  
Z = kmeans.predict(np.c\_[xx.ravel(), yy.ravel()])  
Z = Z.reshape(xx.shape)  
  
# Графічне відображення областей та виділення їх кольором  
plt.figure(1)  
plt.imshow(Z, interpolation='nearest', extent=(xx.min(), xx.max(), yy.min(), yy.max()), cmap=plt.cm.Paired, aspect='auto', origin='lower')  
  
# Відображення вхідних точок  
plt.scatter(data[:, 0], data[:, 1], c=kmeans.labels\_, cmap=plt.cm.Paired)  
plt.title('K-Means Clustering')  
plt.xlabel('Feature 1')  
plt.ylabel('Feature 2')  
  
# Відображення центрів кластерів  
plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1], s=200, c='red', label='Cluster Centers')  
  
plt.show()

Метод кластеризації k-середніх успішно розділив вхідні дані на п'ять кластерів і візуалізував їх межі. Ця процедура дозволила нам чітко розуміти, як дані розподілені між цими п'ятьма групами. Кластери були чітко виділені, і такий підхід може бути корисним для подальшого аналізу або використання в реальних завданнях.

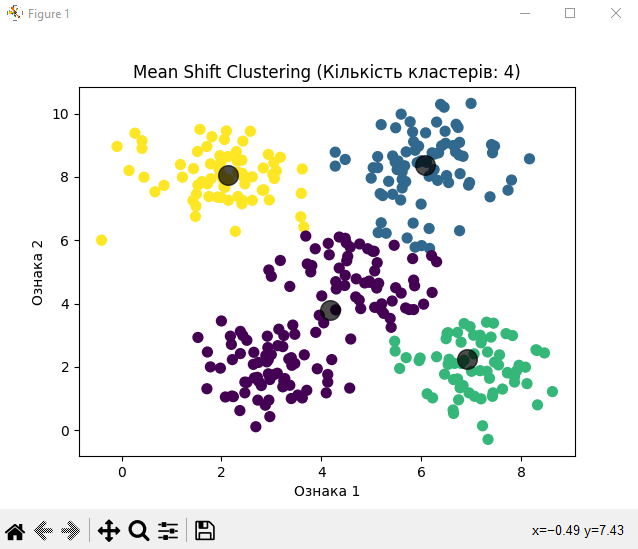
**Завдання 2.8.** Кластеризація K-середніх для набору даних Iris



Код програми:

import matplotlib  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
from sklearn.datasets import load\_iris  
  
matplotlib.use('TkAgg')  
  
# Завантаження даних Iris  
iris = load\_iris()  
X = iris.data # Ознаки (довжина і ширина чашолистка і пелюстки)  
y = iris.target # Мітки класів  
  
# Створення об'єкту KMeans для кластеризації на 3 кластери  
kmeans = KMeans(n\_clusters=3, random\_state=0)  
  
# Навчання моделі кластеризації KMeans  
kmeans.fit(X)  
  
# Отримання міток для кожного прикладу даних  
y\_kmeans = kmeans.predict(X)  
  
# Візуалізація результатів  
plt.figure(figsize=(10, 5))  
  
# Відобразимо довжину чашолистка проти довжини пелюстки з кольорами на основі міток кластерів  
plt.scatter(X[:, 0], X[:, 2], c=y\_kmeans, cmap='viridis', s=50)  
plt.xlabel('Довжина чашолистка')  
plt.ylabel('Довжина пелюстки')  
plt.title('Кластеризація K-Means (3 кластери) для Iris')  
  
# Відобразимо центри кластерів  
centers = kmeans.cluster\_centers\_  
plt.scatter(centers[:, 0], centers[:, 2], c='black', s=200, alpha=0.5, label='Центри кластерів')  
plt.legend()  
  
plt.show()

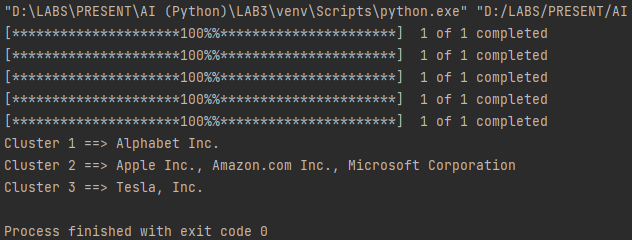
**Завдання 2.9.** Оцінка кількості кластерів з використанням методу зсуву середнього



Код програми:

import matplotlib  
import numpy as np  
from sklearn.cluster import MeanShift, estimate\_bandwidth  
from sklearn.datasets import make\_blobs  
import matplotlib.pyplot as plt  
  
matplotlib.use('TkAgg')  
  
# Завантаження даних з файлу  
data = np.loadtxt('data\_clustering.txt', delimiter=',') # Завантаження даних з файлу  
  
# Оцінка оптимальної ширини вікна  
bandwidth = estimate\_bandwidth(data, quantile=0.2, n\_samples=len(data))  
  
# Створення об'єкта MeanShift для кластеризації з обраною шириною вікна  
ms = MeanShift(bandwidth=bandwidth, bin\_seeding=True)  
  
# Навчання моделі кластеризації Mean Shift  
ms.fit(data)  
  
# Визначення центрів кластерів  
cluster\_centers = ms.cluster\_centers\_  
  
# Оцінка кількості кластерів  
num\_clusters = len(cluster\_centers)  
  
# Візуалізація результатів  
plt.scatter(data[:, 0], data[:, 1], c=ms.labels\_, cmap='viridis', s=50)  
plt.scatter(cluster\_centers[:, 0], cluster\_centers[:, 1], c='black', s=200, alpha=0.7)  
  
plt.title(f'Mean Shift Clustering (Кількість кластерів: {num\_clusters})')  
plt.xlabel('Ознака 1')  
plt.ylabel('Ознака 2')  
plt.show()  
  
print(f'Оцінена кількість кластерів: {num\_clusters}')

**Завдання 2.10.** Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності





Код програми:

import datetime  
import json  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.covariance import GraphicalLassoCV  
from sklearn.cluster import affinity\_propagation  
import yfinance as yf # Використовуємо yfinance для отримання фінансових даних  
  
# Ваш шлях до файлу з прив'язками символічних позначень компаній  
with open('company\_symbol\_mapping.json', 'r') as f:  
 company\_symbols\_map = json.load(f)  
  
symbols, names = np.array(list(company\_symbols\_map.items())).T  
  
# ...  
# Завантаження архівних даних котирувань за допомогою yfinance  
start\_date = datetime.datetime(2010, 1, 1)  
end\_date = datetime.datetime(2021, 12, 31)  
  
quotes = []  
# Отримання котирувань для всіх компаній  
for symbol in symbols:  
 try:  
 quote = yf.download(symbol, start=start\_date, end=end\_date)['Adj Close']  
 if not quote.empty:  
 quotes.append(quote)  
 else:  
 print(f"No data available for [{symbol}]")  
 except Exception as e:  
 print(f"Failed download: [{symbol}]: {e}")  
  
# ...  
# Обчислення середнього значення котирувань для всіх компаній  
valid\_quotes = [quote.values for quote in quotes if not quote.empty]  
min\_length = min(len(quote) for quote in valid\_quotes)  
closing\_quotes = np.array([quote[:min\_length] for quote in valid\_quotes], dtype=float)  
  
# Обчислення різниці між відкриттям та закриттям  
quotes\_diff = closing\_quotes - closing\_quotes[:, 0][:, None]  
  
X = quotes\_diff.copy().T  
X /= X.std(axis=0)  
  
# Створення моделі графа  
edge\_model = GraphicalLassoCV()  
  
# Навчання моделі  
with np.errstate(invalid='ignore'):  
 edge\_model.fit(X)  
  
# Створення моделі кластеризації на основі поширення подібності  
\_, labels = affinity\_propagation(edge\_model.covariance\_)  
num\_labels = labels.max()  
  
# ...  
for i in range(num\_labels + 1):  
 cluster\_names = names[np.where(labels == i)]  
 print("Cluster", i + 1, "==>", ', '.join(cluster\_names))

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

GitHub: https://github.com/invicibleee/Artificial-intelligence.git