**ЛАБОРАТОРНА РОБОТА №3**

**ДОСЛІДЖЕННЯ МЕТОДІВ РЕГРЕСІЇ ТА НЕКОНТРОЬОВАНОГО НАВЧАННЯ**

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

**Хід роботи:**

**GitHub:** https://github.com/ipz201svo/AI

**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

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),

)

**Зображення, що містить знімок екрана, Барвистість, ряд, лазер

Автоматично згенерований опис**

Рис. 1.1 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис



Рис. 1.2 – Результат виконання програми

Модель лінійної регресії натренована на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка становить 0.59, середня квадратична різниця дорівнює 0.49,медіана абсолютних помилок дорівнює 0.51, оцінка поясненої дисперсії становить 0.86, нова абсолютна помилка для моделі, відновленої з використанням збереженого файлу також дорівнює 0.59.

**Завдання 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="yellow", 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))

Зображення, що містить знімок екрана, ряд, Барвистість

Автоматично згенерований опис

Рис. 2.1 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

Рис. 2.2 – Результат виконання програми

Модель лінійної регресії натренована на 80% даних і оцінена на залишкових 20%. Середня абсолютна помилка становить 2.42, середня квадратична різниця дорівнює 9.02 (значення вище ніж попереднє, що свідчить про великі відхилення у прогнозах), медіана абсолютних помилок дорівнює 2.14, оцінка поясненої дисперсії становить -0.15 (прогнози надто відмінні від фактичних значень).

**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:]

# Створення об'єкта лінійного регресора

linear = linear\_model.LinearRegression()

linear.fit(X\_train, y\_train)

# Прогнозування результату

y\_test\_pred = linear.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))

# 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", linear.predict(datapoint))

print("\nPolynomial regression:\n", poly\_linear\_model.predict(poly\_datapoint))

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

Рис. 3.1 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана, дизайн

Автоматично згенерований опис

Рис. 3.2 – Результат виконання програми

Середня абсолютна помилка становить 3.58, середня квадратична різниця дорівнює 20.31(значення вище ніж попереднє, що свідчить про великі відхилення 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.

**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("Linear regressor performance:")

print("Coef = ", regr.coef\_)

print("Intercept = ", regr.intercept\_)

print("R2 score =", round(r2\_score(ytest, ypred), 2))

print("Mean absolute error =", round(mean\_absolute\_error(ytest, ypred), 2))

print("Mean squared error =", round(mean\_squared\_error(ytest, ypred), 2))

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()

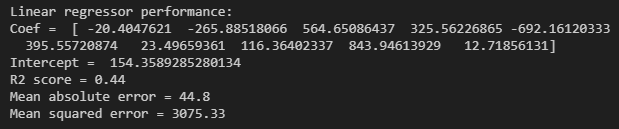


Рис. 4.1 – Результат виконання програми

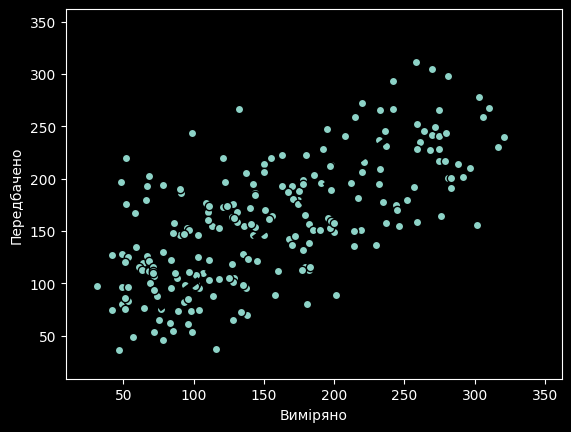


Рис. 4.1 – Результат виконання програми

Було виконано лінійну регресію для набору даних «Diabetes» та було отримано такі результати якості:

Коефіцієнти регресії представляються 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 близько 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.

**Завдання 5:** Самостійна побудова регресії.

import numpy as np

from sklearn import linear\_model

import sklearn.metrics as sm

from sklearn.preprocessing import PolynomialFeatures

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

m = 100

X = np.linspace(-3, 3, m)

y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

# Розбивка даних на навчальний та тестовий набори

X\_train, X\_test, y\_train, ytest = train\_test\_split(X.reshape((-1, 1)), y, test\_size

= 0.5, random\_state = 0)

lin\_reg = linear\_model.LinearRegression()

lin\_reg.fit(X.reshape((-1, 1)), y)

poly\_featueres = PolynomialFeatures(degree=2, include\_bias=False)

X\_poly = poly\_featueres.fit\_transform(X.reshape((-1, 1)))

poly\_reg = linear\_model.LinearRegression()

poly\_reg.fit(X\_poly, y)

print('Linear regression:')

print('Intercept: ', lin\_reg.intercept\_)

print('Coefficients: ', lin\_reg.coef\_)

print('\nPolynomial regression:')

print('Intercept: ', poly\_reg.intercept\_)

print('Coefficients: ', poly\_reg.coef\_)

plt.scatter(X, y)

plt.plot(X, lin\_reg.predict(X.reshape(-1, 1)), color='red', linewidth=2, label='linear')

plt.plot(X, poly\_reg.predict(X\_poly), color='blue', linewidth=2, label='polynomial')

plt.show()

Зображення, що містить знімок екрана, Барвистість

Автоматично згенерований опис

Рис. 5.1 – Результат виконання програми

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

Рис. 5.2 – Результат виконання програми

Модель у вигляді математичного рівняння

**Завдання 6:** Побудова кривих навчання.

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 = np.linspace(-3, 3, m)

y = np.sin(X) + np.random.uniform(-0.5, 0.5, m)

def 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))

plt.plot(np.sqrt(train\_errors), "r-+", linewidth=2, label='train')

plt.plot(np.sqrt(val\_errors), "b-", linewidth=3, label='val')

plt.show()

lin\_reg = linear\_model.LinearRegression()

plot\_learning\_curves(lin\_reg, np.array(X).reshape(-1, 1), y)

polynomial\_regression = Pipeline([

('poly\_features', PolynomialFeatures(degree=2, include\_bias=False)),

('lin\_reg', linear\_model.LinearRegression()),

])

plot\_learning\_curves(polynomial\_regression, np.array(X).reshape(-1, 1), y)

Зображення, що містить знімок екрана, Графік, схема, ряд

Автоматично згенерований опис

Рис. 6.1 – Результат виконання програми

Зображення, що містить знімок екрана, Графік, схема

Автоматично згенерований опис

Рис. 6.2 – Результат виконання програми

**Завдання 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="yellow", 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("Вхідні дані")

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")

plt.title("Границі кластерів та центри")

plt.xlim(x\_min, x\_max)

plt.ylim(y\_min, y\_max)

plt.xticks(())

plt.yticks(())

plt.show()

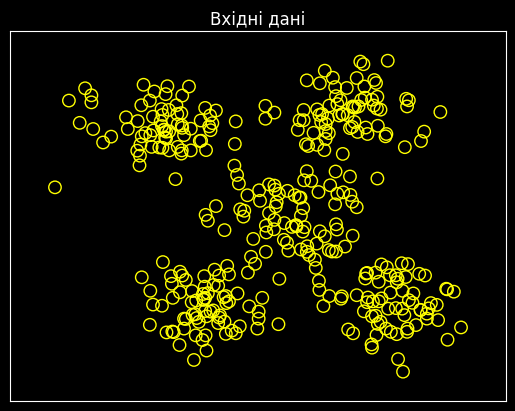


Рис. 7.1 – Результат виконання програми

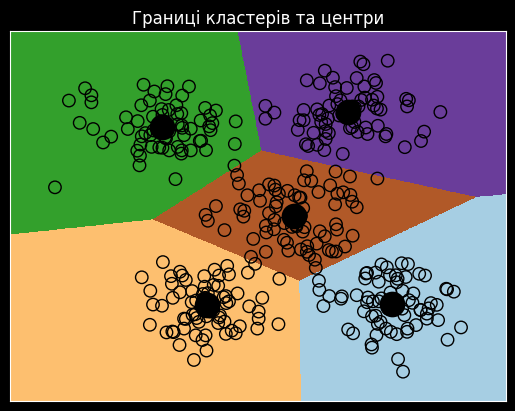


Рис. 7.2 – Результат виконання програми

Ми використали агоритм K-Means для кластеризації даних, було використано 5 кластерів.

**Завдання 8:** Кластеризація K-середніх для набору даних Iris.

from sklearn.svm import SVC

from sklearn.metrics import pairwise\_distances\_argmin

from sklearn.cluster import KMeans

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

import numpy as np

iris = load\_iris()

X = iris["data"]

y = iris["target"]

KMeans(

n\_clusters=8,

init="k-means++",

n\_init=10,

max\_iter=300,

tol=0.0001,

verbose=0,

random\_state=None,

copy\_x=True,

algorithm="auto",

)

# Створення моделі

kmeans = KMeans(n\_clusters = 5)

# Тренування моделі

kmeans.fit(X)

# Створення прогнозу

y\_kmeans = kmeans.predict(X)

# Відображення значень

plt.scatter(X[:, 0], X[:, 1], c=y\_kmeans, s=50, cmap="viridis")

centers = kmeans.cluster\_centers\_

plt.scatter(centers[:, 0], centers[:, 1], c="black", s=200, alpha=0.5)

# Функція для знаходження кластерів

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

centers, labels = find\_clusters(X, 3)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

centers, labels = find\_clusters(X, 3, rseed=0)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

labels = KMeans(3, random\_state=0).fit\_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

Зображення, що містить знімок екрана, Барвистість, Мультимедійне програмне забезпечення

Автоматично згенерований опис

Рис. 8.1 – Результат виконання програми

Зображення, що містить знімок екрана

Автоматично згенерований опис

Рис. 8.2 – Результат виконання програми

**Завдання 9:** Оцінка кількості кластерів з використанням методу зсуву середнього.

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import MeanShift, estimate\_bandwidth

from itertools import cycle

X = np.loadtxt("data\_clustering.txt", delimiter=",")

bandwidth = estimate\_bandwidth(X, quantile=0.1, n\_samples=len(X))

meanshift\_model = MeanShift(bandwidth=bandwidth, bin\_seeding=True)

meanshift\_model.fit(X)

cluster\_centers = meanshift\_model.cluster\_centers\_

print("\nCenters of clusters:\n", cluster\_centers)

labels = meanshift\_model.labels\_

num\_clusters = len(np.unique(labels))

print("\nNumber of clusters in input data =", num\_clusters)

plt.figure()

markers = "o\*xvs"

for i, marker in zip(range(num\_clusters), markers):

plt.scatter(X[labels == i, 0], X[labels == i, 1], marker=marker, color="black")

cluster\_center = cluster\_centers[i]

plt.plot(

cluster\_center[0],

cluster\_center[1],

marker="o",

markerfacecolor="white",

markeredgecolor="yellow",

markersize=15,

)

plt.title("Clusters")

plt.show()

Зображення, що містить текст, Шрифт, знімок екрана, дизайн

Автоматично згенерований опис

Рис. 9.1 – Результат виконання програми

Зображення, що містить знімок екрана

Автоматично згенерований опис

Рис. 9.2 – Результат виконання програми

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.

**Завдання 10:** Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.

import datetime

import json

import numpy as np

from sklearn import covariance, cluster

import yfinance as yf

# Вхідний файл із символічними позначеннями компаній

input\_file = "company\_symbol\_mapping.json"

# Завантаження прив'язок символів компаній до їх повних назв

with open(input\_file, "r") as f:

company\_symbols\_map = json.loads(f.read())

symbols, names = np.array(list(company\_symbols\_map.items())).T

# Визначення архівних даних котирувань

start\_date = "2003-07-03"

end\_date = "2007-05-04"

# Завантаження архівних даних котирувань

quotes = []

valid\_symbols = []

for symbol in symbols:

try:

data = yf.download(symbol, start=start\_date, end=end\_date)

if not data.empty:

quotes.append(data)

valid\_symbols.append(symbol)

except Exception as e:

print(f"Failed to download data for {symbol}: {e}")

# Перевірка чи є валідні дані

if not quotes:

print(

"No valid data available for any symbol. Check your symbol mapping and data availability."

)

else:

# Оновлення символів на дійсні

symbols = valid\_symbols

# Вилучення котирувань, що відповідають відкриттю та закриттю біржі

opening\_quotes = np.array([quote["Open"].values for quote in quotes]).T

closing\_quotes = np.array([quote["Close"].values for quote in quotes]).T

# Обчислення різниці між двома видами котирувань

quotes\_diff = closing\_quotes - opening\_quotes

# Нормалізація даних

X = quotes\_diff.copy()

X /= X.std(axis=0)

# Створення моделі графа

edge\_model = covariance.GraphicalLassoCV()

# Навчання моделі

with np.errstate(invalid="ignore"):

edge\_model.fit(X)

# Створення моделі кластеризації на основі поширення подібності

\_, labels = cluster.affinity\_propagation(edge\_model.covariance\_)

num\_labels = labels.max()

# Виведення результатів

print("\nClustering of stocks based on difference in opening and closing quotes:\n")

for i in range(num\_labels + 1):

cluster\_indices = np.where(labels == i)[0]

cluster\_names = names[cluster\_indices]

if len(cluster\_names) > 0:

print("Cluster", i + 1, "==>", ", ".join(cluster\_names))

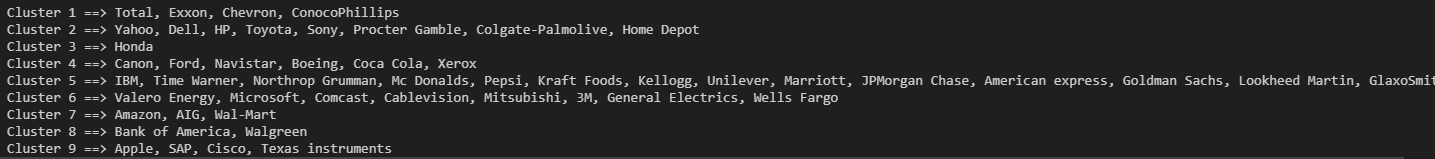


Рис. 10.1 – Результат виконання програми

**Висновок:** в ході виконання лабораторної роботи використовуючи спеціалізовані бібліотеки та мову програмування Python досліджено різні методи класифікації даних та навчився їх порівнювати