ЛАБОРАТОРНА РОБОТА № 7
ДОСЛІДЖЕННЯ МЕТОДІВ НЕКОНТРОЛЬОВАНОГО НАВЧАННЯ
Варіант 13
Хід роботи:

Завдання 1: Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data_clustering.txt.

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans


# Завантаження вхідних даних
X = np.loadtxt("Лабораторна робота 7/data_clustering.txt", delimiter=",")
num_clusters = 5

# Включення вхідних даних до графіка
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("Вхідні дані")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks(())
plt.yticks(())

# Створення об'єкту KMeans
kmeans = KMeans(init="k-means++", n_clusters=num_clusters, n_init=10)

# Навчання моделі кластеризації KMeans
kmeans.fit(X)

# Визначення кроку сітки
step_size = 0.01

# Відображення точок сітки
x_vals, y_vals = np.meshgrid(
    np.arange(x_min, x_max, step_size), np.arange(y_min, y_max, step_size)
)

# Передбачення вихідних міток для всіх точок сітки
output = kmeans.predict(np.c_[x_vals.ravel(), y_vals.ravel()])
```

```
# Графічне відображення областей та виділення їх кольором
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",
)

# Відображення вхідних точок
plt.scatter(X[:, 0], X[:, 1], marker="o", facecolors="none", edgecolors="black",
s=80)

# Відображення центрів кластерів
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("Межі кластерів")
plt.xlim(x_min, x_max)
plt.ylim(y_min, y_max)
plt.xticks()
plt.yticks()
plt.show()
```

Виконання:

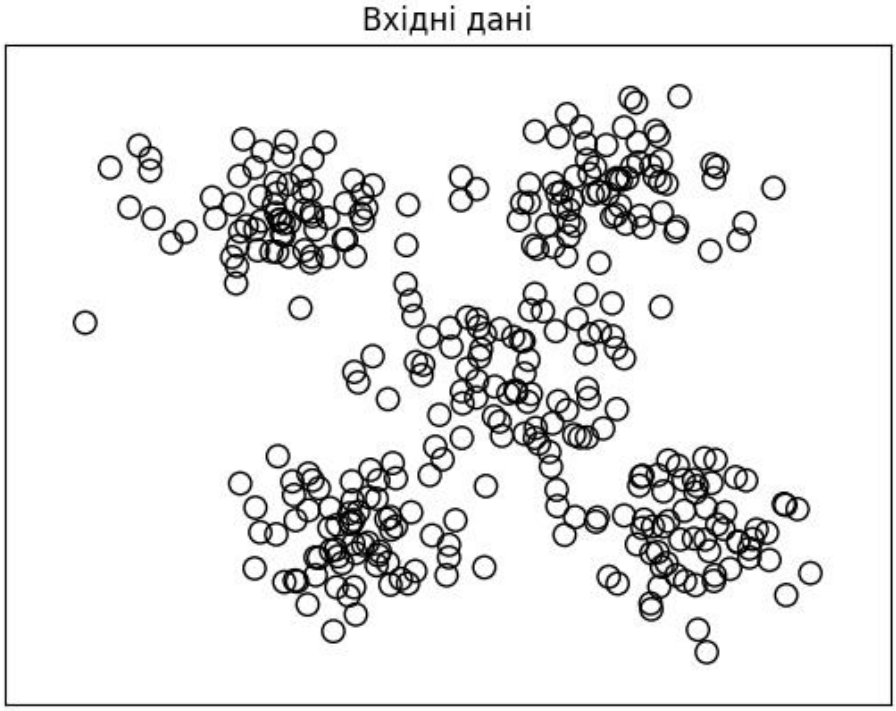

Рисунок 1.1 – Вхідні дані

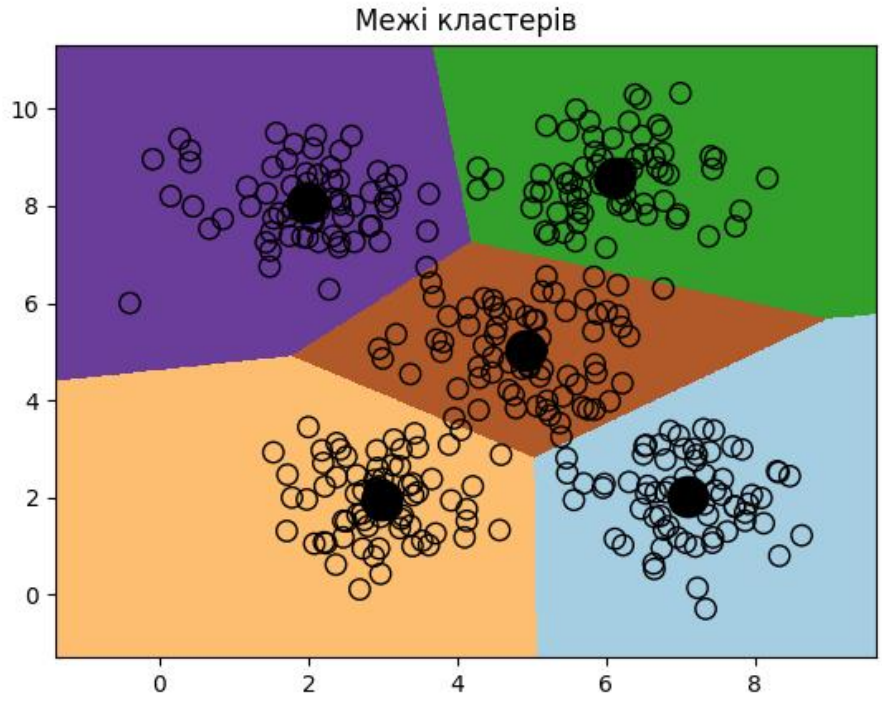

Рисунок 1.2 – Виконання програми

Завдання 2: Виконайте кластеризацію K-середніх для набору даних Iris, який включає три типи (класи) квітів ірису (Setosa, Versicolour і Virginica) з чотирма атрибутами: довжина чашолистка, ширина чашолистка, довжина пелюстки та ширина пелюстки. У цьому завданні використовуйте sklearn.cluster.KMeans для пошуку кластерів набору даних Iris.

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.metrics import pairwise_distances_argmin
from sklearn.datasets import load_iris


iris = load_iris()
X = iris["data"]
y = iris["target"]

# Створюємо модель
kmeans = 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="lloyd",
)
# Навчаємо
kmeans.fit(X)

# Прогнозуємо кластери
y_kmeans = kmeans.predict(X)

# Візуалізуємо результати
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 пошуку 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:
        # Визначаємо підходящі кластери
        labels = pairwise_distances_argmin(X, centers)
```

```
        # Визначаємо центри
        new_centers = np.array([X[labels == i].mean(0) for i in range(n_clusters)])

        # Зупиняємось, якщо центри не змінні
        if np.all(centers == new_centers):
            break
        centers = new_centers
    return centers, labels


# Кластеризація
centers, labels = find_clusters(X, 3)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

# Кластеризація зі зміненим random_state
centers, labels = find_clusters(X, 3, rseed=0)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

# Кластеризація з 3 кластерами
labels = KMeans(3, random_state=0).fit_predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, s=50, cmap="viridis")

plt.show()
```

Виконання:

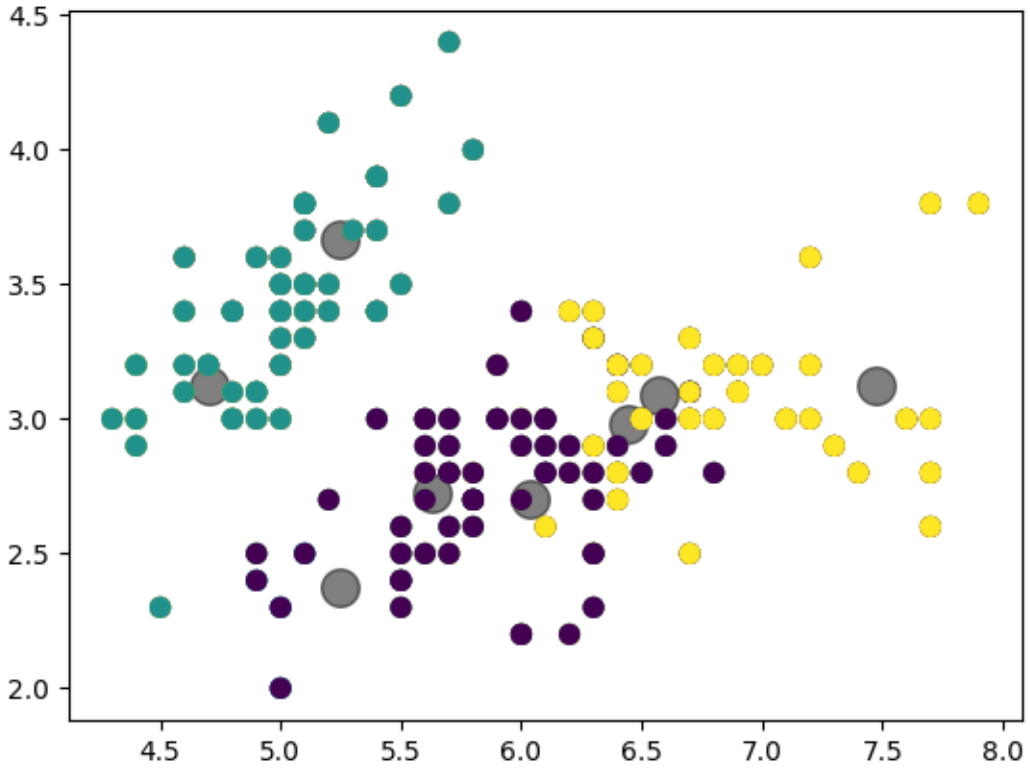

Рисунок 2.1 – Виконання програми

Завдання 3: Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

Код програми:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth

# Завантаження
X = np.loadtxt("Лабораторна робота 7/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="black")

    # Відображення на графіку центру кластера
    cluster_center = cluster_centers[i]
    plt.plot(
        cluster_center[0],
        cluster_center[1],
        marker="o",
        markerfacecolor="black",
        markeredgecolor="black",
        markersize=15,
    )
    plt.title("Clusters")
    plt.show()
```

Виконання:

```
Centers of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]

Number of clusters in input data = 5
```

Рисунок 3.1 – Виконання програми

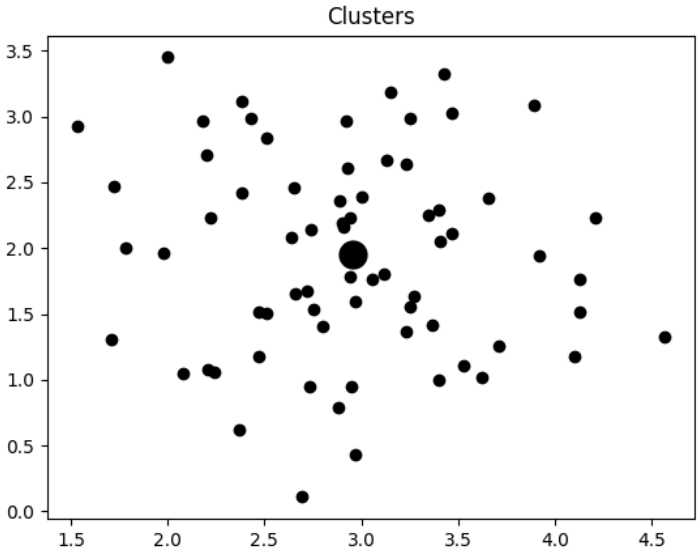

Рисунок 3.2 – Виконання програми

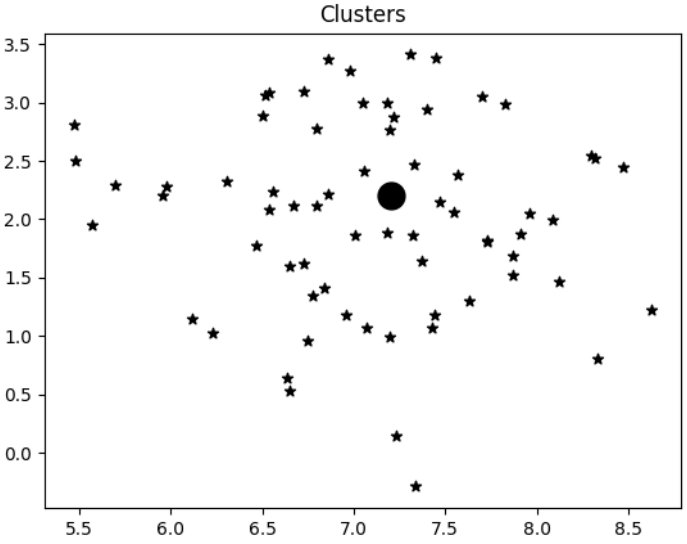

Рисунок 3.3 – Виконання програми

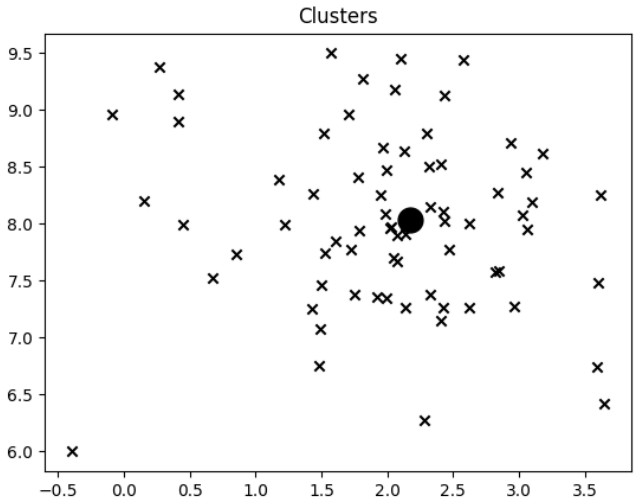

Рисунок 3.4 – Виконання програми

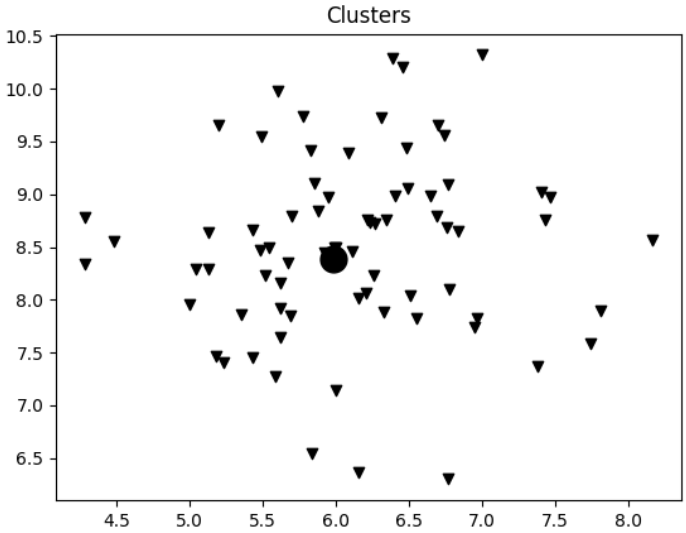

Рисунок 3.5 – Виконання програми

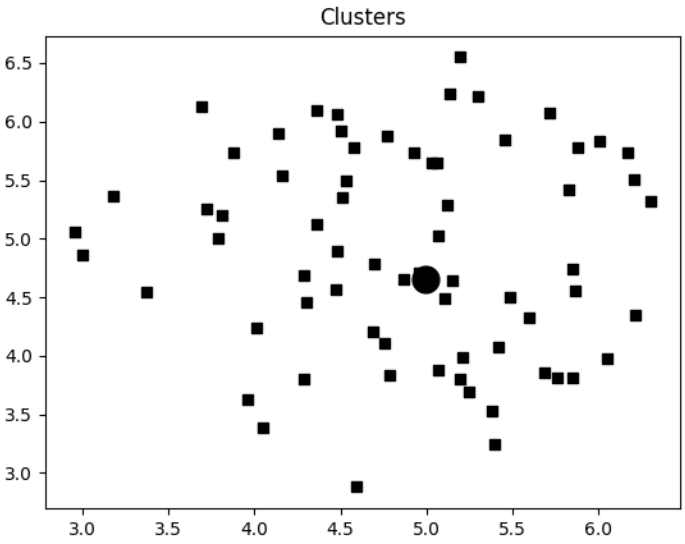

Рисунок 3.6 – Виконання програми

Завдання 4: Відповідно до рекомендацій, напишіть програму та оцініть максимальну кількість кластерів у заданому наборі даних за допомогою алгоритму зсуву середньою. Для аналізу використовуйте дані, які містяться у файлі data_clustering.txt.

Код програми:

```
import json
import numpy as np
import pandas as pd
from sklearn import covariance, cluster
import yfinance as yf  # Заміна `quotes_historical_yahoo_ochl`


# Вхідний файл із символічними позначеннями компаній
input_file = "Лабораторна робота 7/company_symbol_mapping.json"  # Завантажено з
https://github.com/PacktPublishing/Artificial-Intelligence-with-
Python/blob/master/Chapter%2004/code/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

# Завантаження архівних даних котирувань
start_date = "2003-07-03"
end_date = "2007-05-04"
quotes = []
for symbol in symbols:
    data = yf.download(symbol, start=start_date, end=end_date)
    if not data.empty:
        # Обчислення різниці між двома видами котирувань
        data["Price_Difference"] = data["Close"] - data["Open"]

        symbol_quotes = data[["Price_Difference"]].reset_index()
        symbol_quotes["Ticker"] = symbol

        quotes.append(symbol_quotes)

quotes = pd.concat(quotes, ignore_index=True)

pivot_df = quotes.pivot(index="Date", columns="Ticker", values="Price_Difference")

# Fill any missing values with zeros (or another method, if needed)
pivot_df.fillna(0, inplace=True)

# Extract the data as a 2D NumPy array
quotes_diff = pivot_df.values

# Нормалізація
X = quotes_diff.copy()
```

```
X /= X.std(axis=0)

# Створення моделі графа
edge_model = covariance.GraphicalLassoCV()

# Навчання моделі
edge_model.fit(X)

# Створення моделі кластеризації на основі поширення подібності
_, labels = cluster.affinity_propagation(edge_model.covariance_)
num_labels = labels.max()


# Виведення результатів
for i in range(num_labels + 1):
    cluster_symbols = [symbols[j] for j in range(len(labels)) if labels[j] == i]
    print("Cluster", i + 1, "==>", ", ".join(cluster_symbols))
```

Виконання:

```
Cluster 1 ==> CVC, HPQ, XRX, HD
Cluster 2 ==> CVX, YHOO, MTU, NOC, BA, PG, CL, TXN
Cluster 3 ==> XOM, COP, MSFT, CMCSA, AMZN, SNE, HMC, KO, K, BAC, CSCO, WMT, WBA, GSK
Cluster 4 ==> CAJ, MDLZ, GE
Cluster 5 ==> MAR
Cluster 6 ==> F, WFC, AIG, AAPL
Cluster 7 ==> TWX, MMM, MCD, PEP, JPM, AXP
Cluster 8 ==> TOT, VLO, IBM, TM, NAV, UN, GS, SAP
Cluster 9 ==> DELL, LMT
```

Рисунок 4.1 – Виконання програми

Посилання на GitHub: https://github.com/ipz215kvv/artificial-inteligence-systems