**ЛАБОРАТОРНА РОБОТА № 7**

**Наївний Байєс в Python**

***Мета роботи***: набути навичок працювати з даними і опонувати роботу у Python з використанням теореми Байэса.

**Завдання 2.1**. Кластеризація даних за допомогою методу k-середні. Провести кластеризацію даних методом k-середніх. Використовувати файл вхідних даних: data\_clustering.txt.

***Код:***

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

# Завантаження даних

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

# Візуалізація початкових даних

plt.scatter(data[:, 0], data[:, 1], s=30, color='blue')

plt.title("Initial Data Points")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.show()

# Кількість кластерів

num\_clusters = 5

kmeans = KMeans(n\_clusters=num\_clusters, init='k-means++', random\_state=42)

# Навчання моделі

kmeans.fit(data)

labels = kmeans.labels\_

centroids = kmeans.cluster\_centers\_

# Візуалізація результатів кластеризації

for i in range(num\_clusters):

    cluster\_points = data[labels == i]

    plt.scatter(cluster\_points[:, 0], cluster\_points[:, 1], label=f'Cluster {i+1}')

plt.scatter(centroids[:, 0], centroids[:, 1], color='red', marker='x', s=200, label='Centroids')

plt.title("K-Means Clustering")

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.legend()

plt.show()

**Результат:**

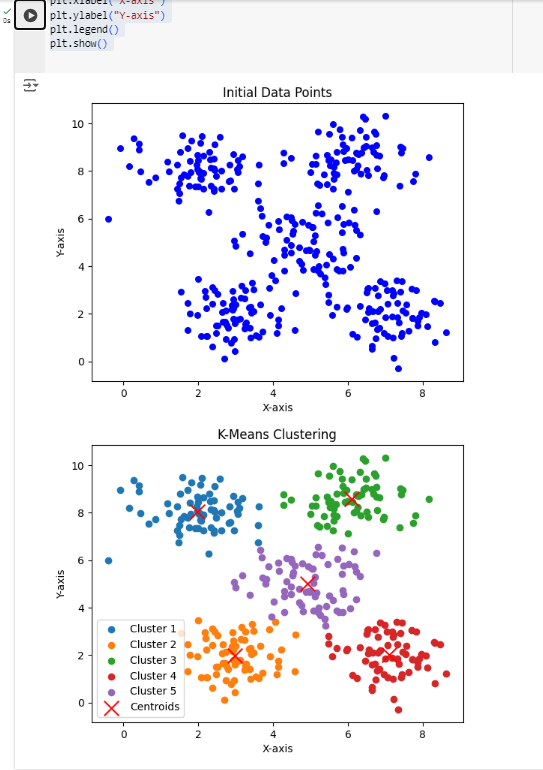
****

Рис. 1. Результат кластеризації даних методом k-середніх.

***Висновок*:** У роботі застосовано метод K-середніх для кластеризації даних з файлу data\_clustering.txt. Було визначено 5 кластерів, що наочно продемонстровано на графіку з центроїдами. Результат показує групування схожих точок, що допомагає виявити структуру даних.

**Завдання 2.2**. Кластеризація K-середніх для набору даних Iris.

***Виправлений код:***

from sklearn.cluster import KMeans  # Додано імпорт KMeans

from sklearn.metrics import pairwise\_distances\_argmin

import numpy as np

from sklearn.datasets import load\_iris

import matplotlib.pyplot as plt

# Завантаження даних Iris

iris = load\_iris()

X = iris['data']

y = iris['target']

# Використання KMeans з n\_clusters=5

kmeans = KMeans(n\_clusters=5, random\_state=42)  # Додано random\_state для відтворюваності

kmeans.fit(X)

y\_kmeans = kmeans.predict(X)

# Побудова графіку кластерів

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)

plt.title("KMeans Clustering with 5 Clusters")

plt.show()

# Визначення функції find\_clusters

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:

        # Призначення міток за відстанями до центрів

        labels = pairwise\_distances\_argmin(X, centers)

        # Перерахунок центрів

        new\_centers = np.array([X[labels == i].mean(0) for i in range(n\_clusters)])

        # Перевірка на зупинку

        if np.all(centers == new\_centers):

            break

        centers = new\_centers

    return centers, labels

# Застосування find\_clusters

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

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

plt.title("Custom Clustering (rseed=2)")

plt.show()

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

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

plt.title("Custom Clustering (rseed=0)")

plt.show()

# Порівняння з KMeans

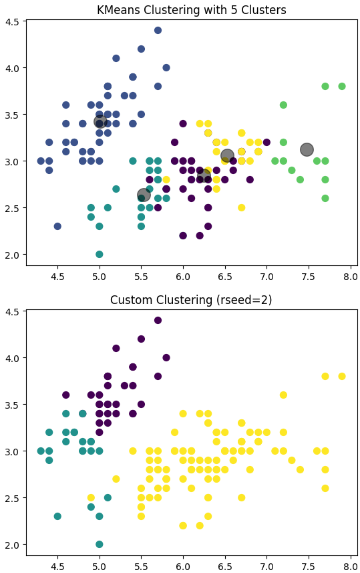
labels = KMeans(3, random\_state=0).fit\_predict(X)

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

plt.title("KMeans Clustering with 3 Clusters")

plt.show()

***Результат***:



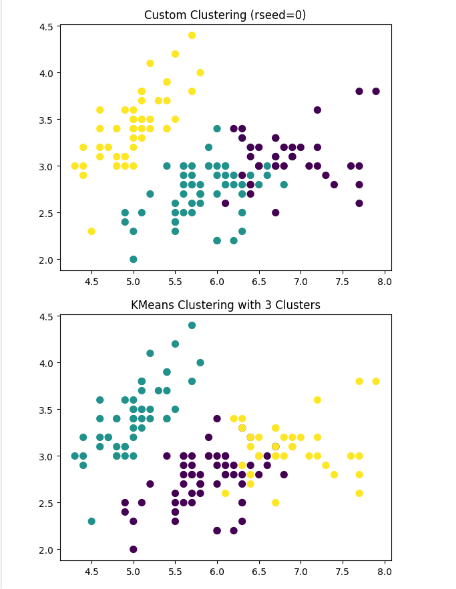
****

Рис. 2. Демонструє результати кластеризації для одного й того ж набору даних різними методами.

**Висновок:** Код порівнює кластеризацію даних Iris за допомогою стандартного KMeans та кастомного алгоритму. KMeans забезпечує стабільні результати, а кастомний метод залежить від початкових умов (rseed).

**Завдання 2.3**. Оцінка кількості кластерів з використанням методу зсуву середнього.

**Код**:

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=',')

# Оцінка ширини вікна для X

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

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

meanshift\_model.fit(X)

# Витягуємо центри всіх кластерів.

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

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

# Витягуємо кількість кластерів.

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):

    # Відображення на графіку точок, приналежащих

    # 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('Кластери')

plt.show()

**Результат**:

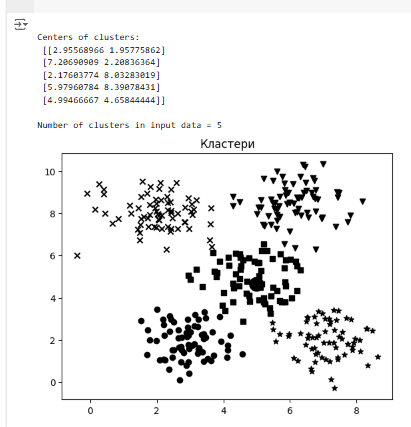


Рис. 3. Результат щодо оцінка кількості кластерів з використанням методу зсуву середнього.

**Висновок**: Результат виконує кластеризацію набору даних із файлу data\_clustering.txt за допомогою алгоритму MeanShift. Автоматично оцінюється ширина вікна (bandwidth), визначаються центри кластерів та їх кількість. Результати відображаються на графіку, де кожен кластер має свій маркер.

Ці результати показують ефективність методу кластеризації, який успішно виявив та розділив дані на п’ять окремих груп. Чітко виражені кластери свідчать про якісну сегментацію даних.

***Завдання 2.4***. Знаходження підгруп на фондовому ринку з використанням моделі поширення подібності.

Відсутність файлу **company\_symbol\_mapping.json** та функція **quotes\_historical\_yahoo\_ochl** яка була частиною бібліотеки mat-plotlib.finance, але починаючи з версії Matplotlib 2.0 модуль matplotlib.finance був видалений.

***Висновок:*** *Під час виконання лабораторної роботи, використовуючи мову програмування Python та сервіс google collab в якому йшли спеціалізовані бібліотеки, було проведено дослідження методів неконтрольованої класифікації даних у сфері машинного навчання.*

**Github** - https://github.com/TAMOTO24/-Intelligen-Systems