Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Кластеризация»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Кирилович А. А.

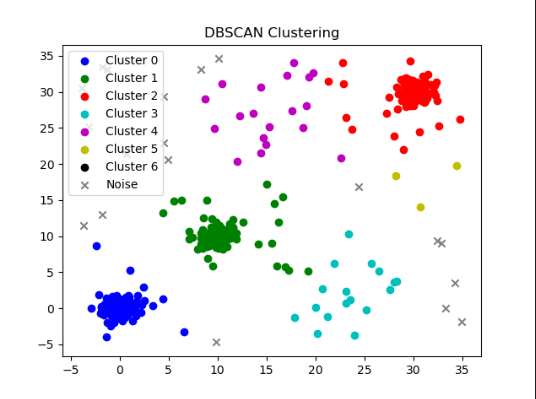
**Проверил:**

Туз И. С.

Брест 2023

**Цель:** Реализовать алгоритм кластеризации данных.

**Ход работы (Вариант 4)**

Метод: метод кластеризации на основе плотности DBSCAN

import matplotlib.pyplot as plt

import numpy as np

class DBSCAN:

def \_\_init\_\_(self, eps, min\_samples):

self.eps = eps

self.min\_samples = min\_samples

self.labels = None

self.visited = set()

def fit(self, X):

self.labels = np.full(X.shape[0], -1)

cluster\_id = 0

for i in range(X.shape[0]):

if i not in self.visited:

self.visited.add(i)

neighbors = self.get\_neighbors(X, i)

if len(neighbors) < self.min\_samples:

self.labels[i] = -1

else:

self.expand\_cluster(X, i, neighbors, cluster\_id)

cluster\_id += 1

def expand\_cluster(self, X, core\_point, neighbors, cluster\_id):

self.labels[core\_point] = cluster\_id

for neighbor in neighbors:

if neighbor not in self.visited:

self.visited.add(neighbor)

new\_neighbors = self.get\_neighbors(X, neighbor)

if len(new\_neighbors) >= self.min\_samples:

neighbors = neighbors.union(new\_neighbors)

if self.labels[neighbor] == -1:

self.labels[neighbor] = cluster\_id

def get\_neighbors(self, X, i):

neighbors = set()

for j in range(X.shape[0]):

if np.linalg.norm(X[i] - X[j]) < self.eps:

neighbors.add(j)

return neighbors

np.random.seed(40)

cluster1 = np.random.normal(loc=0, scale=1, size=(100, 2))

cluster2 = np.random.normal(loc=10, scale=1, size=(100, 2))

cluster3 = np.random.normal(loc=30, scale=1, size=(100, 2))

uniform\_points = np.random.uniform(low=-5, high=35, size=(100, 2))

X = np.concatenate([cluster1, cluster2, cluster3, uniform\_points])

dbscan = DBSCAN(eps=10, min\_samples=20)

dbscan.fit(X)

colors = ['b', 'g', 'r', 'c', 'm', 'y', 'k']

for i in range(len(set(dbscan.labels))):

cluster\_points = X[dbscan.labels == i]

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

plt.scatter(X[dbscan.labels == -1][:, 0], X[dbscan.labels == -1][:, 1], c='gray', marker='x', label='Noise')

plt.title('DBSCAN Clustering')

plt.legend()

plt.show()

**Вывод:** в ходе лабораторной работы я научился кластеризовать данные с помощью метода DBSCAN.