

Министерство науки и высшего образования  
Российской Федерации  
Федеральное государственное бюджетное образовательное  
учреждение высшего образования «Рыбинский государственный  
авиационный технический университет имени П. А. Соловьева»

## ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И СИСТЕМЫ УПРАВЛЕНИЯ

Математического и программного обеспечения электронных  
вычислительных средств

### ОТЧЁТ

по дисциплине:

«Методы и алгоритмы анализа данных»

на тему:

«Кластеризация»

Выполнил: студент группы ИВМ-24

Морозов А. А.

Руководитель: ассистент

Вязниковцев Д. А.

Рыбинск 2024

## Содержание

Цель работы .....	3
1 Первичная обработка данных .....	4
2 Снижение размерности .....	5
2.1 Метод главных компонент .....	5
2.2 Метод стохастического вложения соседей с t-распределением.....	6
2.3 Сравнение двух методов.....	7
3 Методы кластеризации .....	8
3.1 Метод <i>K</i> -средних .....	8
2.5 Метод <i>DBSCAN</i> .....	11
Вывод.....	13

## Цель работы

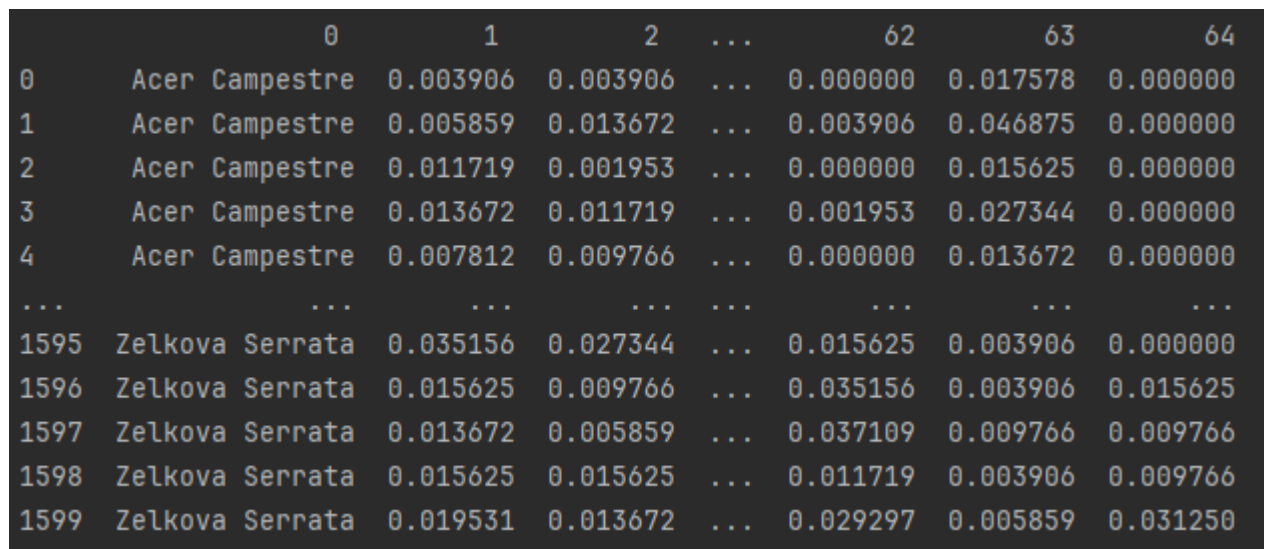
Целями данной лабораторной работы являются:

- изучение теории о кластерном анализе;
- первичная обработка данных;
- работа с методами главных компонент, стохастического вложения соседей с  $t$ -распределением,  $K$ -средних и *DBSCAN*;

## 1 Первичная обработка данных

В начале необходимо загрузить датасет *data\_Mar\_64.txt* и вывести его в консоль (рисунок 1.1):

```
data = pd.read_csv("data_Mar_64.txt", header=None, sep=',',  
quotechar="")  
print(data)
```



		0	1	2	...	62	63	64
0	Acer Campestre	0.003906	0.003906	...	0.000000	0.017578	0.000000	
1	Acer Campestre	0.005859	0.013672	...	0.003906	0.046875	0.000000	
2	Acer Campestre	0.011719	0.001953	...	0.000000	0.015625	0.000000	
3	Acer Campestre	0.013672	0.011719	...	0.001953	0.027344	0.000000	
4	Acer Campestre	0.007812	0.009766	...	0.000000	0.013672	0.000000	
...	...	...	...	...	...	...	...	...
1595	Zelkova Serrata	0.035156	0.027344	...	0.015625	0.003906	0.000000	
1596	Zelkova Serrata	0.015625	0.009766	...	0.035156	0.003906	0.015625	
1597	Zelkova Serrata	0.013672	0.005859	...	0.037109	0.009766	0.009766	
1598	Zelkova Serrata	0.015625	0.015625	...	0.011719	0.003906	0.009766	
1599	Zelkova Serrata	0.019531	0.013672	...	0.029297	0.005859	0.031250	

Рисунок 1.1 – Загруженный датасет

Необходимо разделить датасет на 2. В первом датасете будет храниться целевая переменная, соответствующая столбцу с индексом 0, в закодированном формате, а в другом датасете оставшиеся столбцы:

```
X, y_name = data.iloc[:, 1:].values, data.iloc[:, 0].values  
y = LabelEncoder().fit_transform(y_name)
```

## 2 Снижение размерности

### 2.1 Метод главных компонент

Необходимо снизить размерность признакового пространства до двух. После изменения размерности отфильтруем значения целевой переменной по порогу до 14 и выведем получившиеся кластеры на экран (рисунок 1.2):

```
X_PCA = PCA(n_components=2, random_state=0).fit_transform(X)
X_filtered = X_PCA[y < 15]
y_filtered = y[y < 15]
print(f'X_PCA[0]:\n{round(X_PCA[0][0], 2), round(X_PCA[0][1], 2)}")
plt_show(X_filtered, y_filtered, "Метод главных компонент")
```

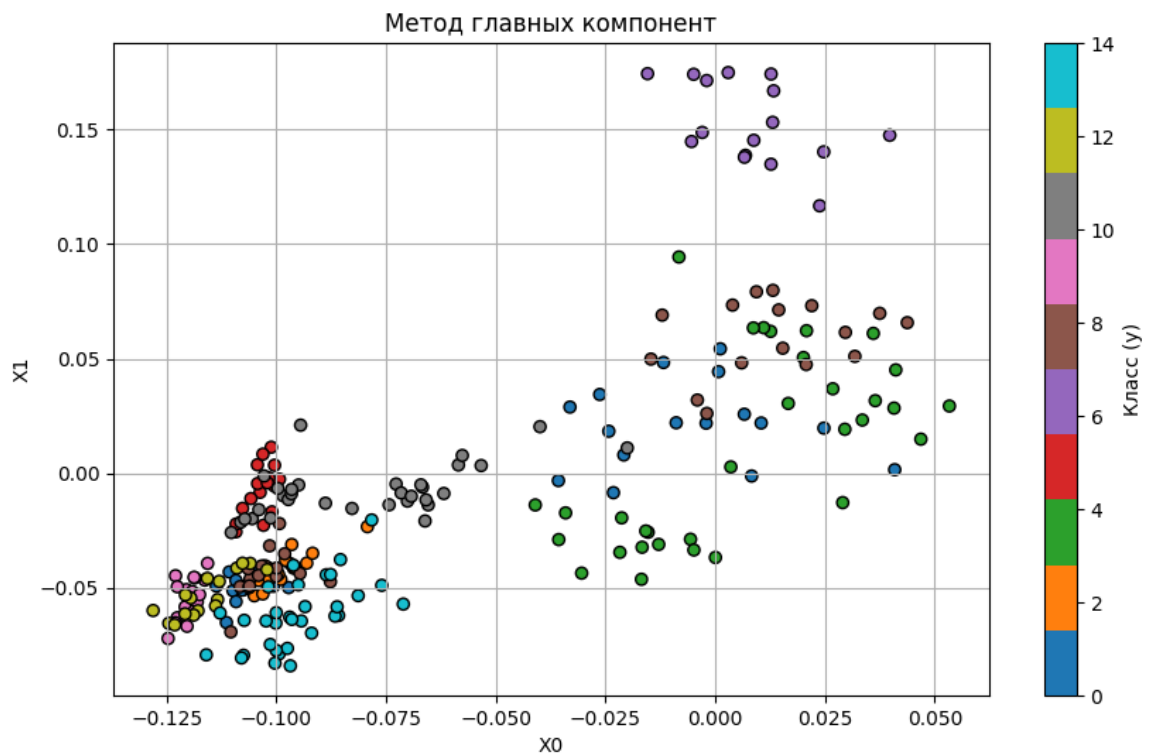


Рисунок 2.1 – Кластеризация методом главных компонент

Значение координат объекта с индексом 0 равно (-0.03, 0.03).

## 2.2 Метод стохастического вложения соседей с t-распределением

Для данного метода воспользуемся функцией *TSNE* из библиотеки *sklearn*:

```
X_tsne = TSNE(n_components=2, random_state=0).fit_transform(X)
X_filtered = X_tsne[y < 15]
plt_show(X_filtered, y_filtered, "Стохастическое вложение соседей с t-распределением")
print(f"X_tsne[0]:\n{round(X_tsne[0][0], 2), round(X_tsne[0][1], 2)}")
```

После выполнения кода выше появится график с кластерами (рисунок 2.2).

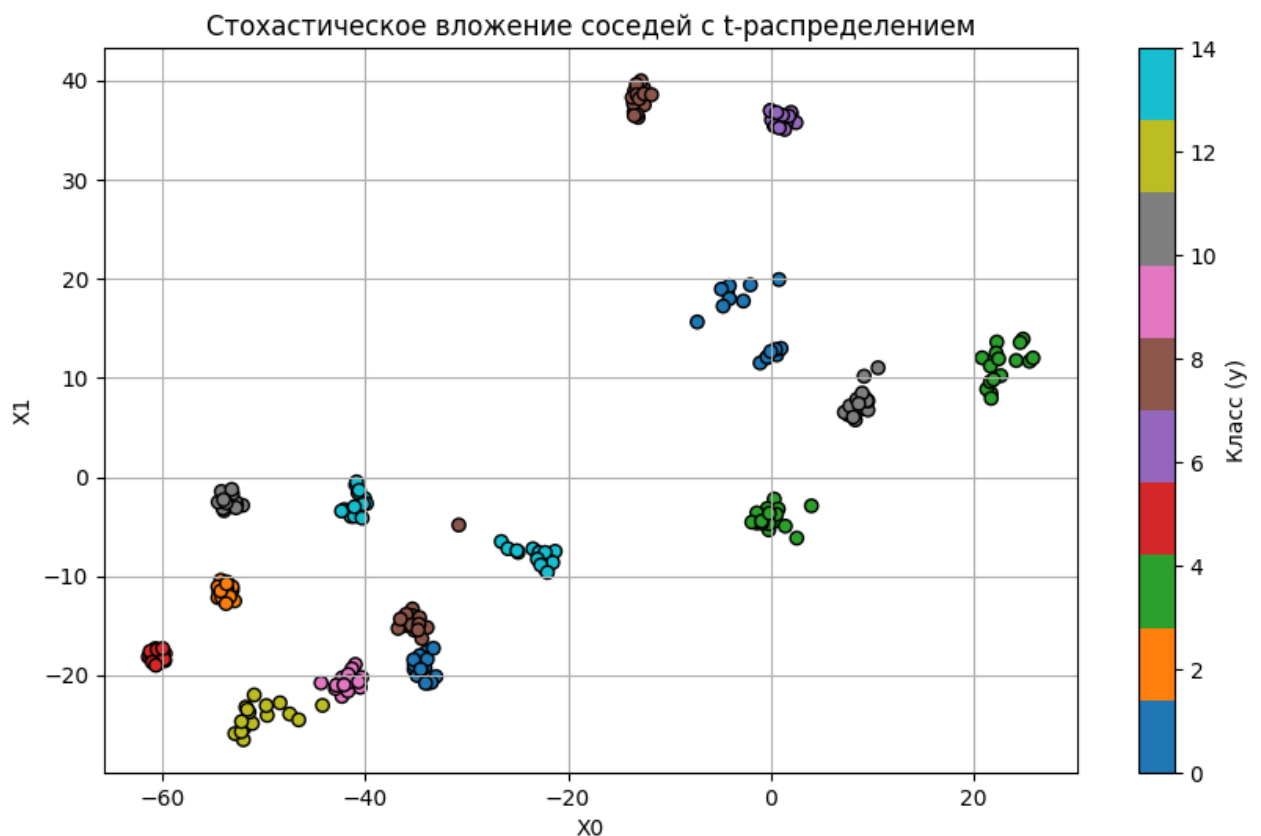


Рисунок 2.2 – Кластеризация методом *TSNE*

Значение координат объекта с индексом 0 равно (-7.26, 15.65).

## 2.3 Сравнение двух методов

Из рисунков 2.1 и 2.2 можно сделать вывод то, что с помощью метода *TSNE* удалось визуализировать объекты на плоскости и объекты разных классов визуально разделимы, а вот про метод главных компонент такое сказать нельзя.

## 3 Методы кластеризации

### 3.1 Метод $K$ -средних

Для данного метода реализуем класс *MyKMeans*:

```
class MyKMeans():
    def __init__(self, n_clusters=3, n_iters=100):
        self.n_clusters = n_clusters
        self.n_iters = n_iters

    def fit(self, X):
        np.random.seed(0)
        self.centers = np.random.uniform(low=X.min(axis = 0),
                                          high=X.max(axis = 0),
                                          size=(self.n_clusters, X.shape[1]))

        for it in range(self.n_iters):
            labels = self.predict(X)
            new_centers = np.array([X[labels == i].mean(axis=0) for i in
range(self.n_clusters)])
            if np.all(self.centers == new_centers):
                print(f"Алгоритм сошёлся на {it} итерации")
                break
            self.centers = new_centers

    def predict(self, X):
        labels = pairwise_distances_argmin(X, self.centers)
        return labels
```

Сгенерируем данные для кластеризации и кластеризируем их с помощью *MyKMeans* используя гиперпараметры  $n\_clusters=3$ ,  $n\_iters=100$ :

```
n_samples = 1000
noisy_blobs = datasets.make_blobs(n_samples=n_samples,
                                  cluster_std=[1.0, 3.0, 0.5],
```

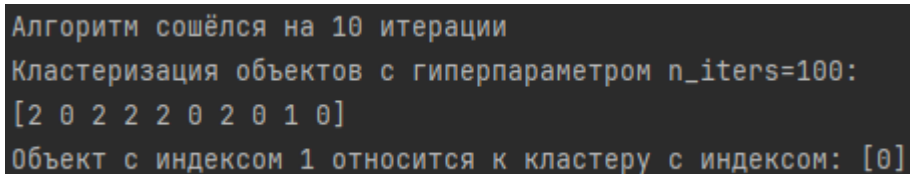


```

random_state=0)
X, y = noisy_blobs
cluster = MyKMeans()
cluster.fit(X)
print(f"Кластеризация объектов с гиперпараметром
n_iters=100:\n{cluster.predict(X)[:10]}")
print(f"Объект с индексом 1 относится к кластеру с индексом:
{cluster.predict(X[1].reshape(1, -1))}")
k_means100 = cluster.predict(X)
plt_show(X, k_means100, "Кластеризация при n_iters=100")

```

После выполнения кода выше в консоли выведется информация о кластеризации (рисунок 2.3) и отобразится график на экране (рисунок 2.4).



```

Алгоритм сошёлся на 10 итерации
Кластеризация объектов с гиперпараметром n_iters=100:
[2 0 2 2 2 0 2 0 1 0]
Объект с индексом 1 относится к кластеру с индексом: [0]

```

Рисунок 3.1 – Информация о кластеризации методом  $K$ -средних с параметром  $n\_iters$  равным 100

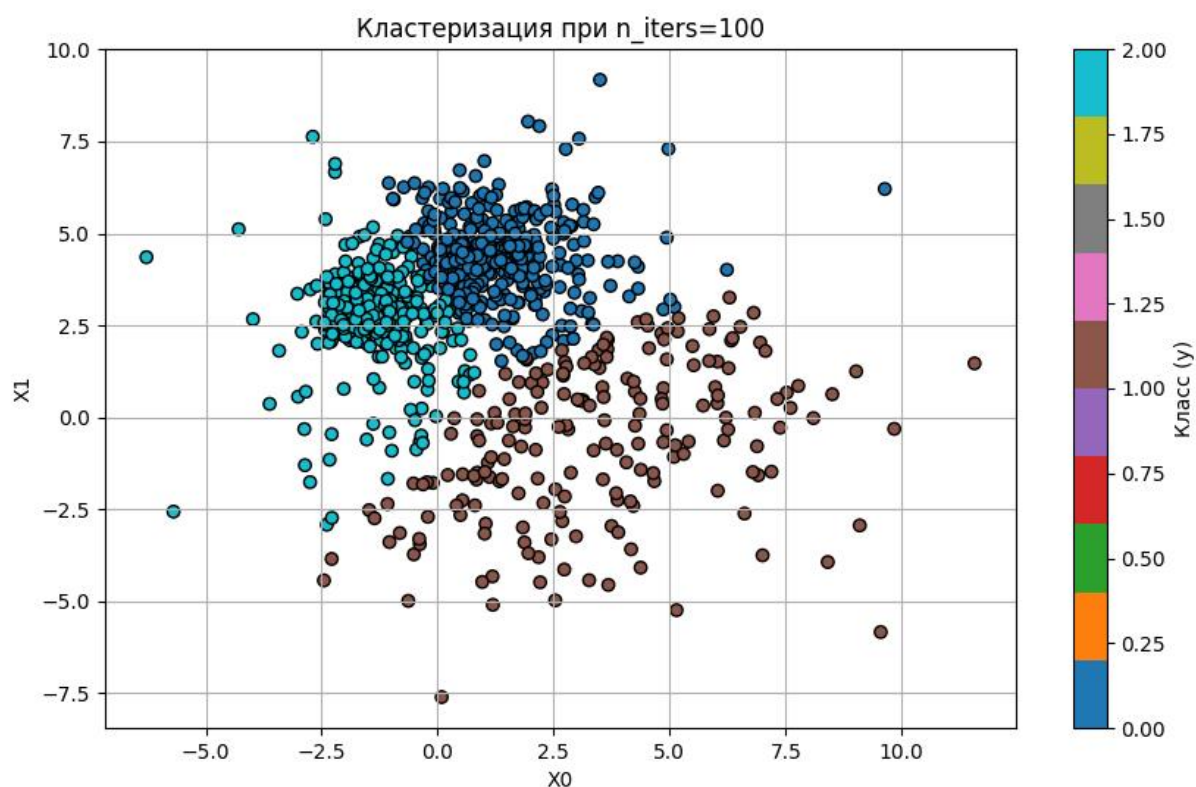


Рисунок 3.2 – Кластеризация методом  $K$ -средних с параметром  $n\_iters$  равным 100

При изменении параметра  $n\_iters$  со 100 на 5, то кластеризация поменяется. На рисунке 2.5 показана информация о кластеризации, на рисунке 2.6 изображён график кластеризации.

```
Кластеризация объектов с гиперпараметром n_iters=5:
[2 0 2 2 2 2 2 0 1 0]
Объект с индексом 1 относится к кластеру с индексом: [0]
```

Рисунок 3.53 – Информация о кластеризации методом  $K$ -средних с параметром  $n\_iters$  равным 5

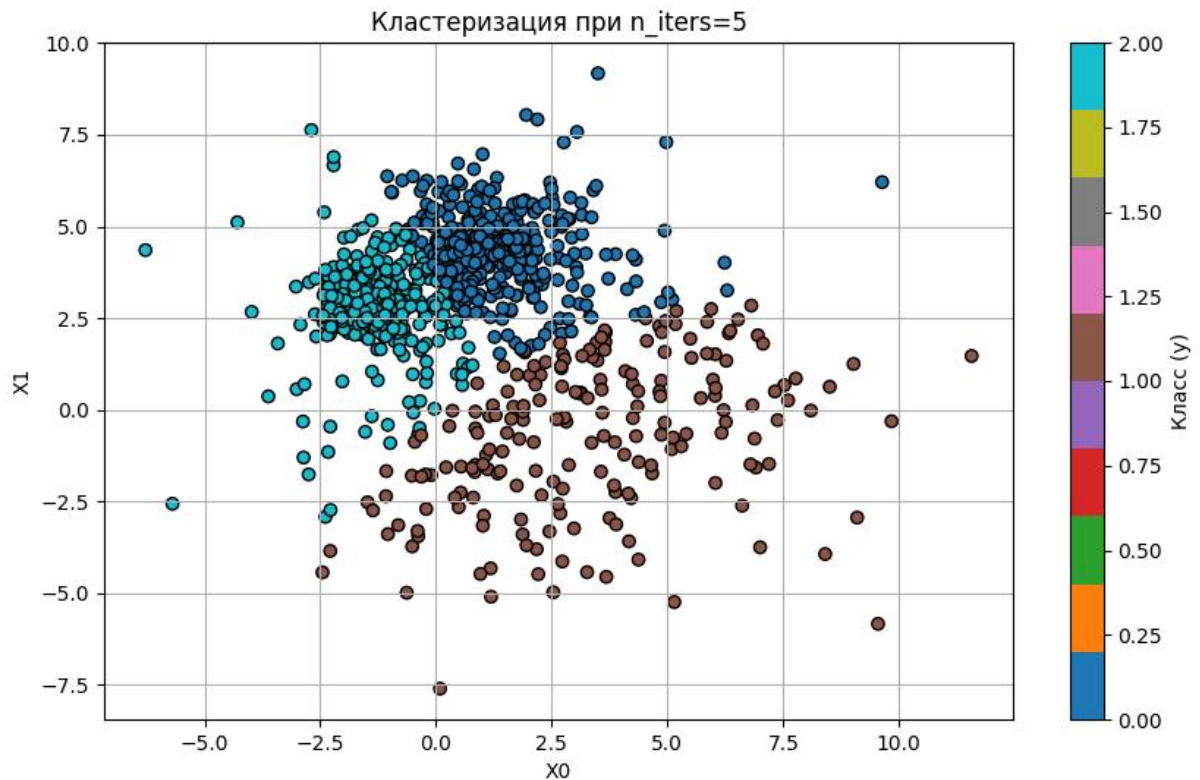


Рисунок 3.4 – Кластеризация методом  $K$ -средних с параметром  $n\_iters$  равным 5

При изменении параметра  $n\_iters$  со 100 на 5 у некоторых объектов изменилась и метка предсказываемого кластера и таких объектов было 17.

## 2.5 Метод *DBSCAN*

Для данного метода воспользуемся функцией *DBSCAN* из библиотеки *sklearn*:

```
clusters = DBSCAN(eps=0.5).fit_predict(X)
print(f"Объект с индексом 1 принадлежит кластеру: {clusters[1]}")
print(f"Полученное количество кластеров равно {len(set(clusters) - {-1})} (без выбросов)")
print(f"Количество объектов, отнесенных к выбросам: {(clusters == -1).sum()}")
plt_show(X, clusters, "DBSCAN")
```

На рисунке 2.7 изображён вывод в консоли, а на рисунке 2.8 изображён вывод графика на экране после выполнения кода.

```
Объект с индексом 1 принадлежит кластеру: 0  
Полученное количество кластеров равно 11 (без выбросов)  
Количество объектов, отнесенных к выбросам: 152
```

Рисунок 3.5 – Информация о кластеризации методом *DBSCAN*

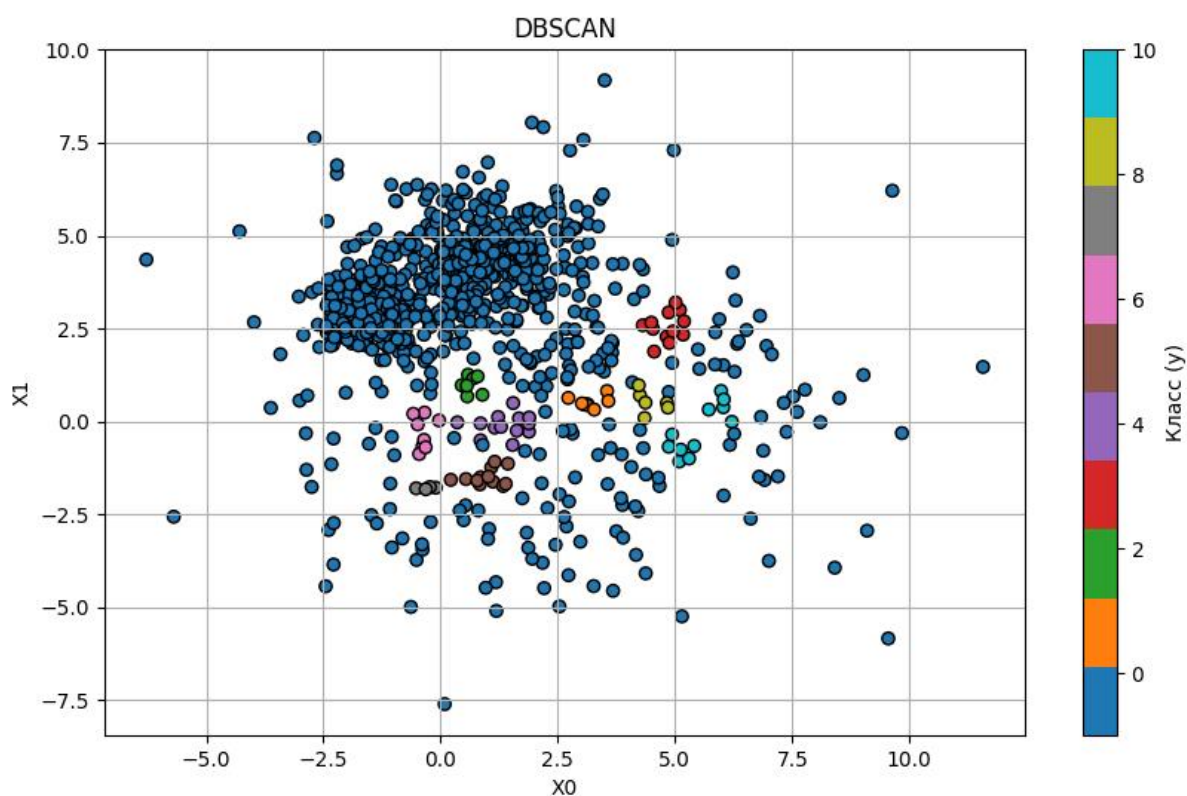


Рисунок 3.6 – Кластеризация методом *DBSCAN*

## Вывод

В ходе работы были получены результаты разных методов снижения размерности и кластеризации. Метод *TSNE* показал себя лучше, чем метод *PCA*. При кластеризации метод *DBSCAN* самостоятельно определил количество кластеров и убрал шумы, но для него необходимо указать максимальное расстояние между объектами в кластере.