МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В. И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Машинное обучение»

Тема: Кластеризация (DBSCAN, OPTICS)

Студент гр. 1310 _____ Комаров Д. Е.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2025

Постановка задачи

Цель работы: ознакомление с методами кластеризации модуля Sklearn

Выполнение лабораторной работы

Загрузка данных

Для выполнения лабораторной работы используем набор данных «Credit Card Dataset for Clustering». Данный набор данных содержит записи о кредитных картах клиентов. Загрузим данные в датафрейм, удалив из него столбец с метками и наблюдения с пропущенными значениями. Получилось 8950 наблюдений. Фрагмент результата представлен на рисунке 1.

	BALANCE	BALANCE_FREQUENCY	PURCHASES	...	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
0	40.900749	0.818182	95.40	...	139.509787	0.000000	12
1	3202.467416	0.909091	0.00	...	1072.340217	0.222222	12
2	2495.148862	1.000000	773.17	...	627.284787	0.000000	12
4	817.714335	1.000000	16.00	...	244.791237	0.000000	12
5	1809.828751	1.000000	1333.28	...	2407.246035	0.000000	12
...	...	...	...	...	...	...	...
8943	5.871712	0.500000	20.90	...	43.473717	0.000000	6
8945	28.493517	1.000000	291.12	...	48.886365	0.500000	6
8947	23.398673	0.833333	144.40	...	82.418369	0.250000	6
8948	13.457564	0.833333	0.00	...	55.755628	0.250000	6
8949	372.708075	0.666667	1093.25	...	88.288956	0.000000	6

Рисунок 1 – Данные для кластеризации

Так как различные признаки имеют различные шкалы измерения, то стандартизируем данные.

DBSCAN

Суть метода DBSCAN заключается в поиске точек, которые имеют достаточное количество соседей на некотором расстоянии. Такие точки вместе с их соседями образуют кластер. Остальные точки, которые не имеют достаточно соседей и не соседствуют с образующими кластер точками считаются шумом и не являются членами какого-либо кластера.

Проведем кластеризацию данных методом DBSCAN при параметрах по умолчанию. Было получено 36 кластеров, однако 75% наблюдений кластеризовать не удалось.

Основными параметрами DBSCAN являются *eps*, *metric, algorithm, min_samples*. Параметр *eps* отвечает за максимальную дистанцию, при которой 2 наблюдения могут быть отнесены к одному кластеру. Параметр *metric* за то,

как будет определяться расстояние между наблюдениями. Параметр *min_samples* отвечает за минимальное количество наблюдений, образующих кластер. Параметр *algorithm* отвечает за выбор алгоритма поиска соседних наблюдений.

Построим график зависимости количества кластеров от параметра *eps*. Данный график представлен на рисунке 2.

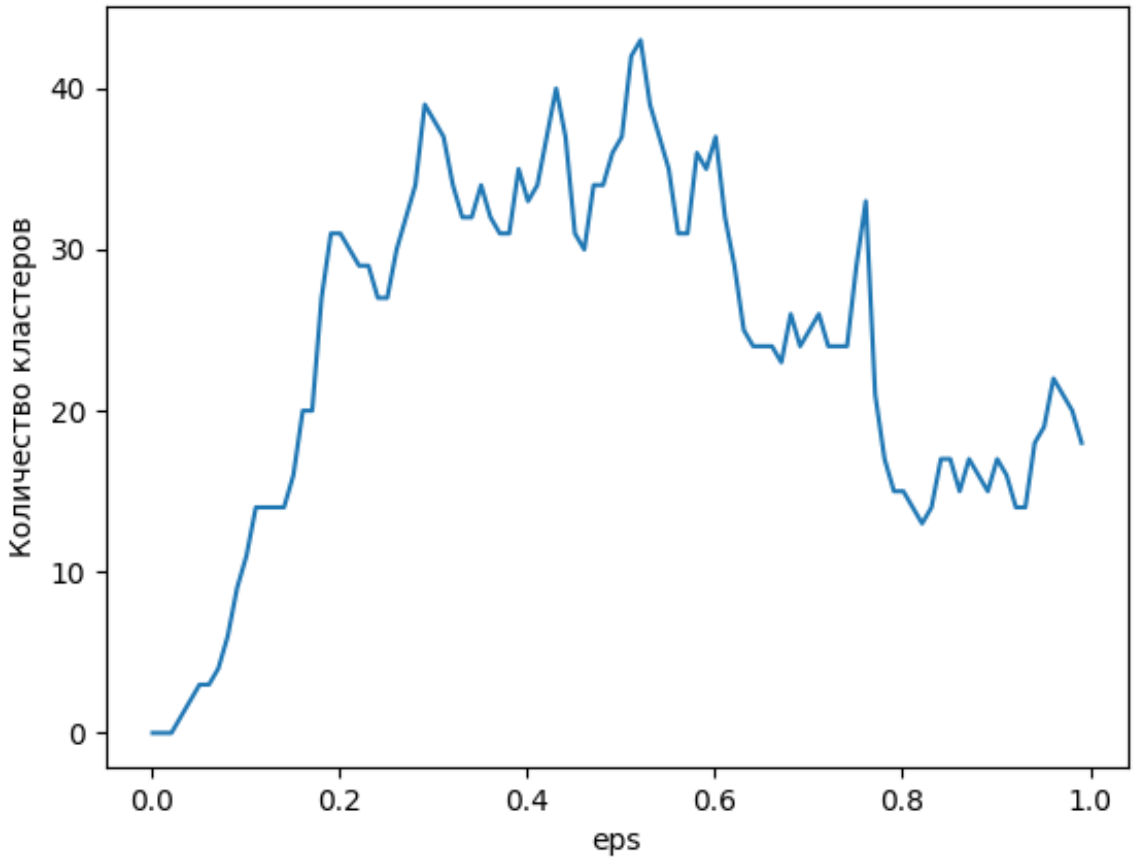

Рисунок 2 – График зависимости количества кластеров от параметра *eps*

График зависимости процента не кластеризованных наблюдений от параметра *eps* представлен на рисунке 3.

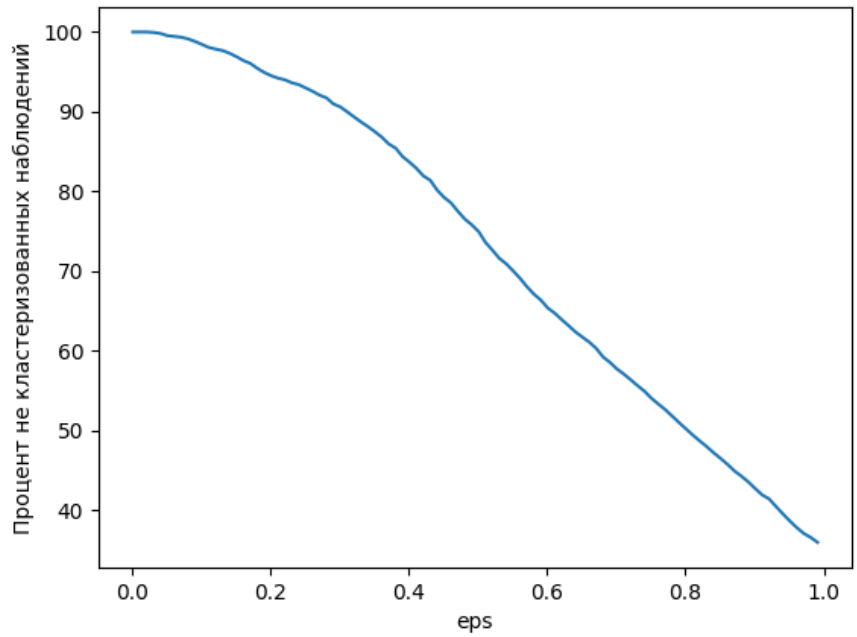

Рисунок 3 – График зависимости процента не кластеризованных наблюдений от параметра *eps*

Построим график зависимости количества кластеров от параметра *min_samples*. Данный график представлен на рисунке 4.

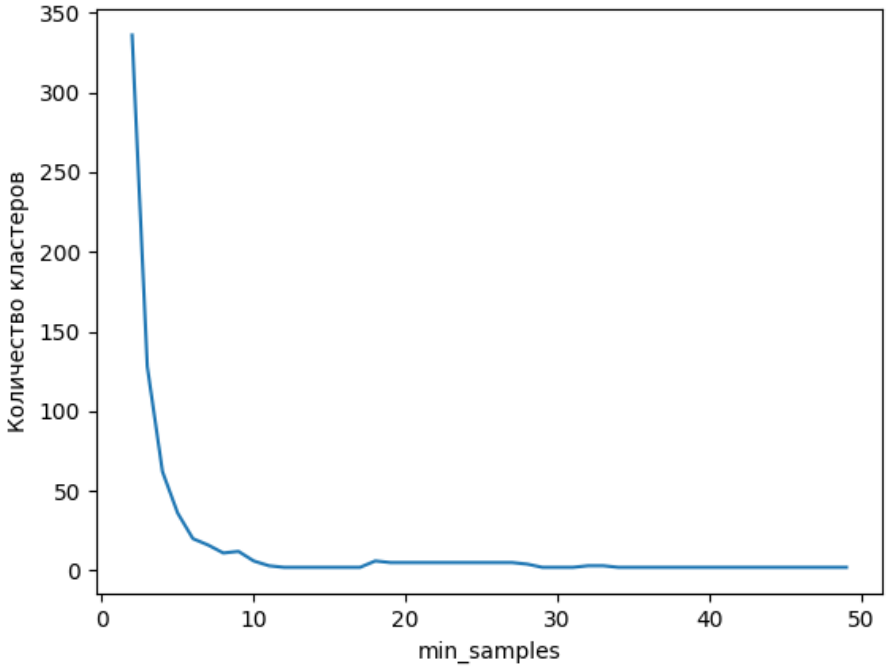

Рисунок 4 – График зависимости количества кластеров от параметра *min_samples*

График зависимости процента не кластеризованных наблюдений от параметра *min_samples* представлен на рисунке 5.

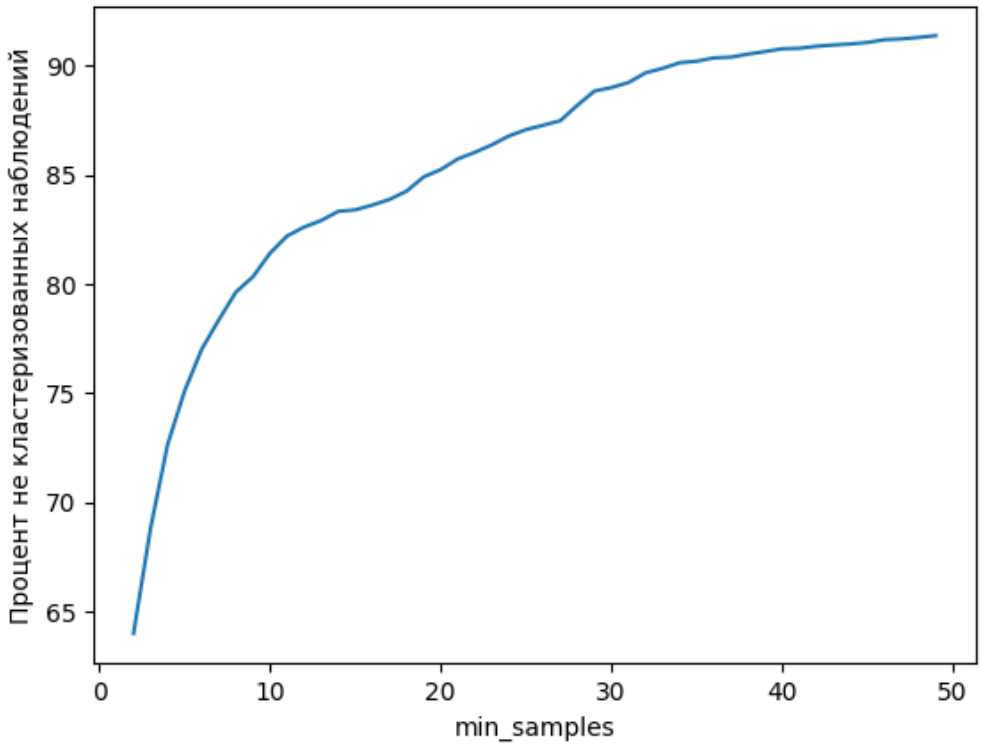

Рисунок 5 – График зависимости процента не кластеризованных наблюдений от параметра *min_samples*

Как можно увидеть из рисунков 3 и 5, процент не кластеризованных наблюдений монотонно убывает при увеличении максимальной дистанции, при которой 2 наблюдения могут быть отнесены к одному кластеру и монотонно возрастает при увеличении минимального количества наблюдений, образующего кластер. Из рисунков 2 и 4 можно заметить, что количество кластеров монотонно убывает при увеличении минимального количества наблюдений, образующего кластер. При увеличении максимальной дистанции, при которой 2 наблюдения могут быть отнесены к одному кластеру, количество кластеров на различных промежутках как возрастает, так и убывает, что логично, так как с одной стороны в таком случае могут появляться новые кластеры из наблюдений, которые ранее располагались слишком далеко друг от друга, чтобы образовать кластер, с другой стороны некоторые кластеры объединяются между собой.

Подберём такие значения параметров *eps* и *min_samples*, при которых количество кластеров получается от 5 до 7 и процент не кластеризованных наблюдений не превышает 12%. Такими параметрами являются *eps=2* и

min_samples=3. При них образуется 6 кластеров, а не кластеризованных наблюдений 6%. Понизим размерность данных до 2 методом главных компонент и визуализируем результаты кластеризации. Результат представлен на рисунке 6.

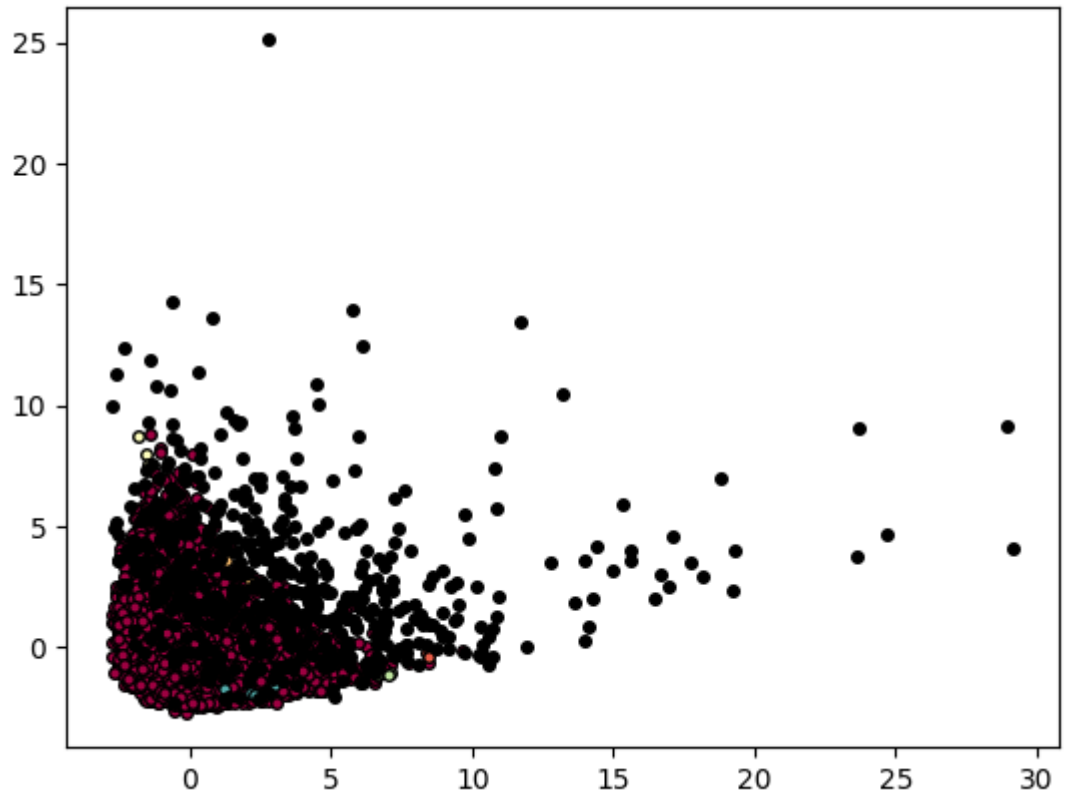

Рисунок 6 – Данные после кластеризации при помощи DBSCAN

Как можно увидеть из рисунка 6, большая часть точек была объединена в один кластер (красные). Также образовалось несколько маленьких кластеров. Черные точки являются шумом и не принадлежат ни одному кластеру.

OPTICS

Метод OPTICS, в отличие от DBSCAN, находит расстояния достижимости для каждой точки. Затем при помощи этих расстоянии достижимости и иных методов кластеризации находятся кластеры. Преимуществом данного алгоритма является то, что он способен определять кластеры различной плотности.

Основными параметрами OPTICS являются *min_samples, max_eps*, *metric*, *cluster_method, algorithm*. Параметр *min_samples* отвечает за

минимальное количество наблюдений, образующих кластер. Параметр *max_eps* отвечает за максимальное расстояние между двумя наблюдениями, при котором они считаются соседними. Параметр *metric* определяет то, как будет вычисляться расстояние между наблюдениями. Параметр *algorithm* определяет алгоритм, по которому будут вычисляться соседние точки. Параметр *cluster_method* определяет, каким образом результат работы алгоритма преобразуется в кластеры.

Подберём такие параметры *min_samples* и *max_eps*, при которых количество кластеров получается от 5 до 7 и процент не кластеризованных наблюдений не превышает 12%. Такими параметрами являются *max_eps*=3 и *min_samples*=3. При них образуется 5 кластеров, а не кластеризованных наблюдений 2%. Понизим размерность данных до 2 методом главных компонент и визуализируем результаты кластеризации. Результат представлен на рисунке 7.

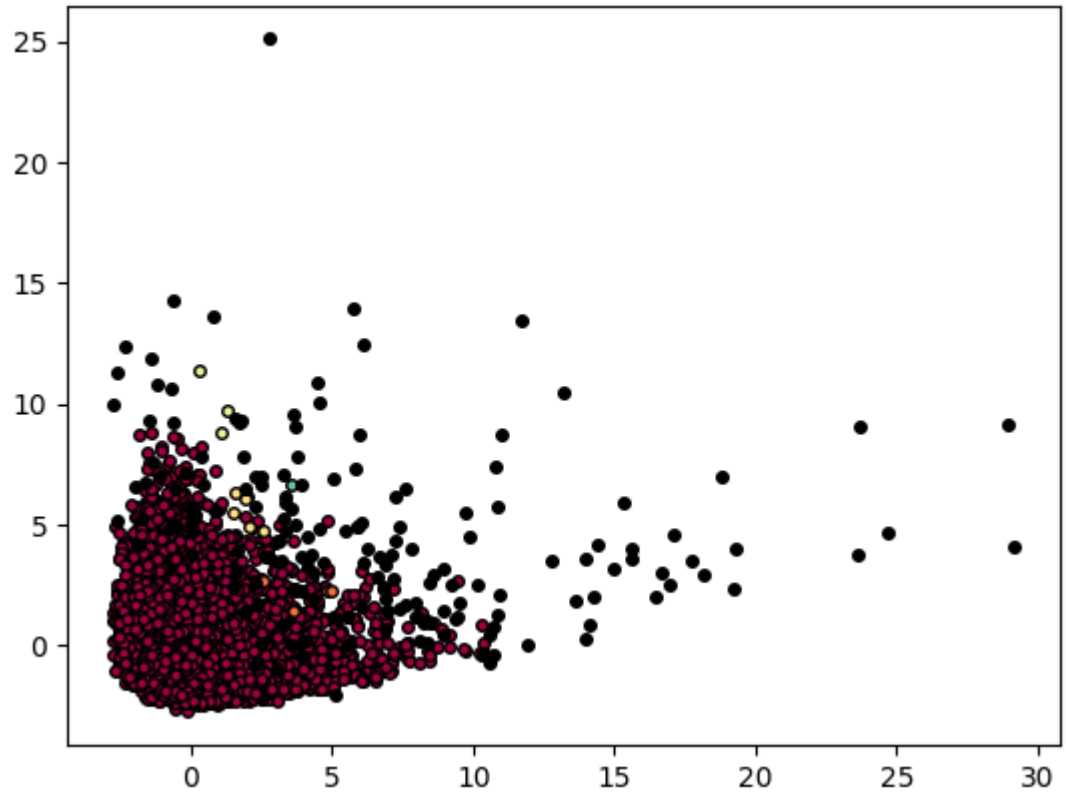

Рисунок 7 – Данные после кластеризации OPTICS

Как можно заметить, результат, полученный OPTICS схож с полученным DBSCAN, однако удалось кластеризовать больший процент наблюдений.

На рисунке 8 представлен график достижимости, полученный при данных параметрах. На горизонтальной оси отображены наблюдения, а на вертикальной – их достижимое расстояние.

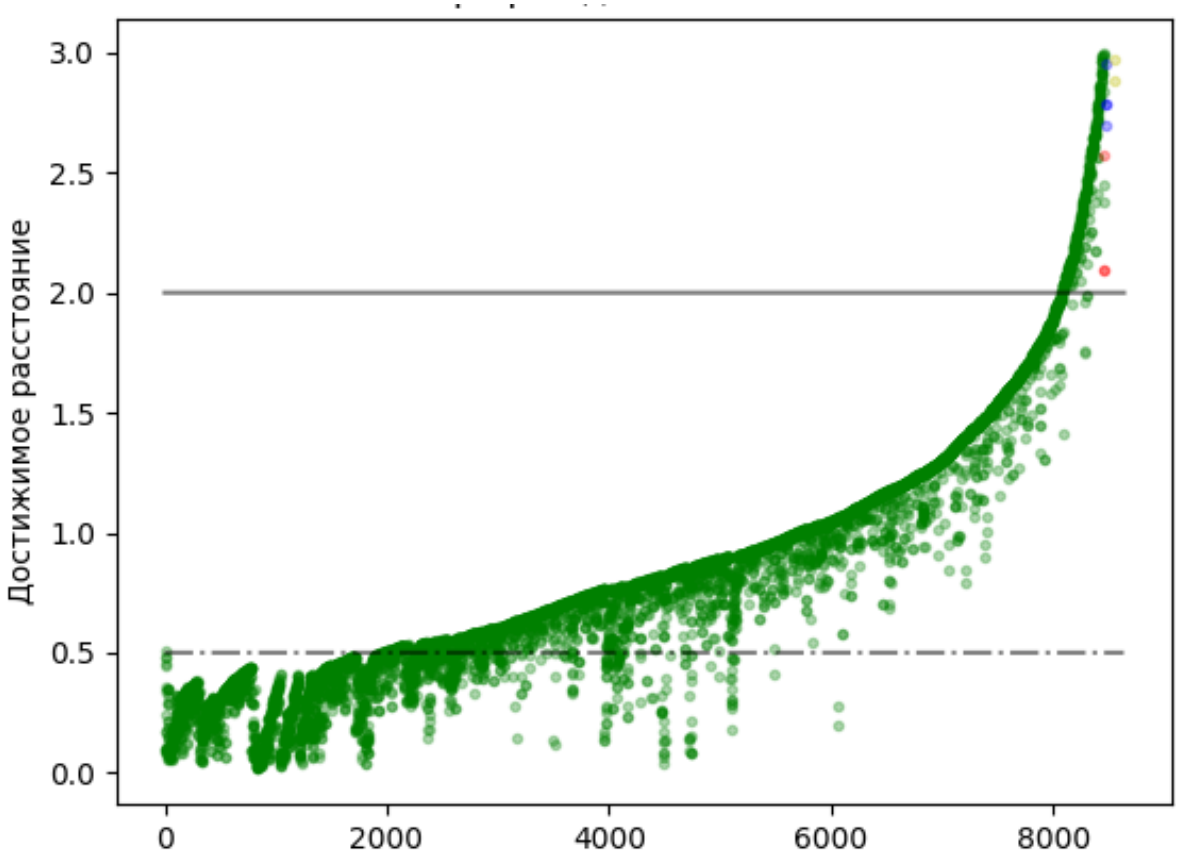

Рисунок 8 – График достижимости

Исследуем метод OPTICS при различных метриках. По умолчанию используется метрика *minkowski*. Изменим метрику на *euclidean*. Получилось 5 кластеров при 2% не кластеризованных наблюдений. Результат после понижения размерности до 2 представлен на рисунке 9.

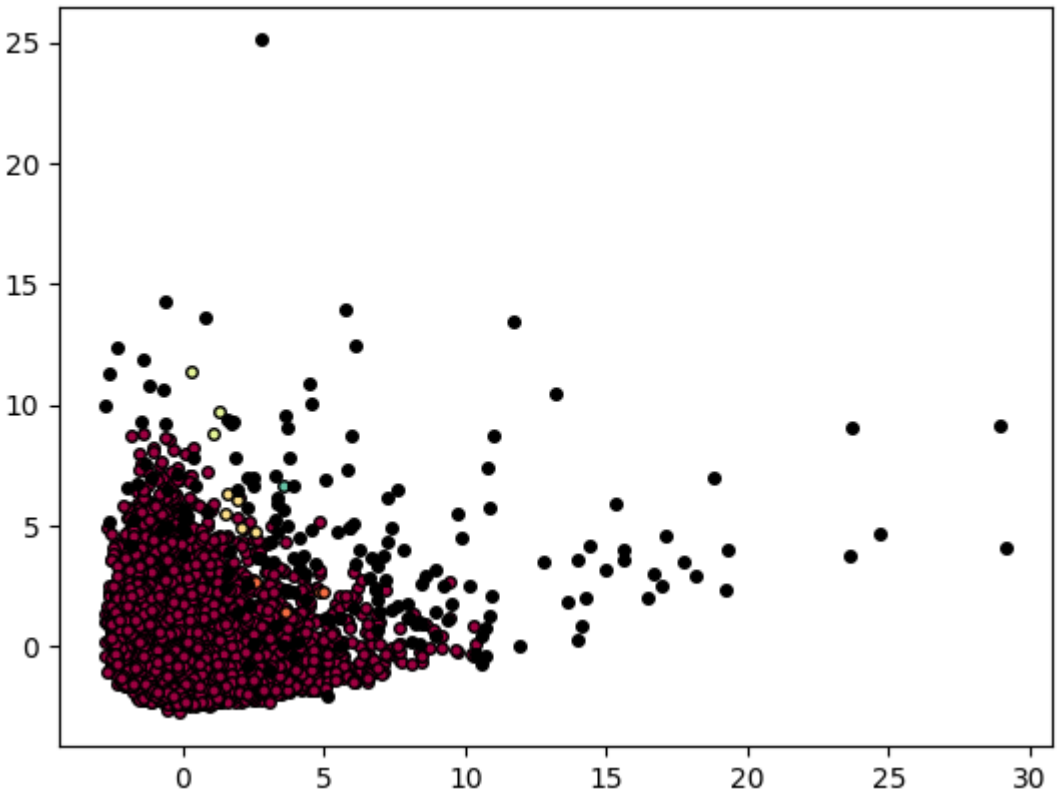

Рисунок 9 – Данные после кластеризации OPTICS с метрикой *euclidean*

Изменим метрику на *manhattan*. Не изменяя остальные параметры, получилось 27 кластеров при 20% не кластеризованных наблюдений. Результат после понижения размерности до 2 представлен на рисунке 10.

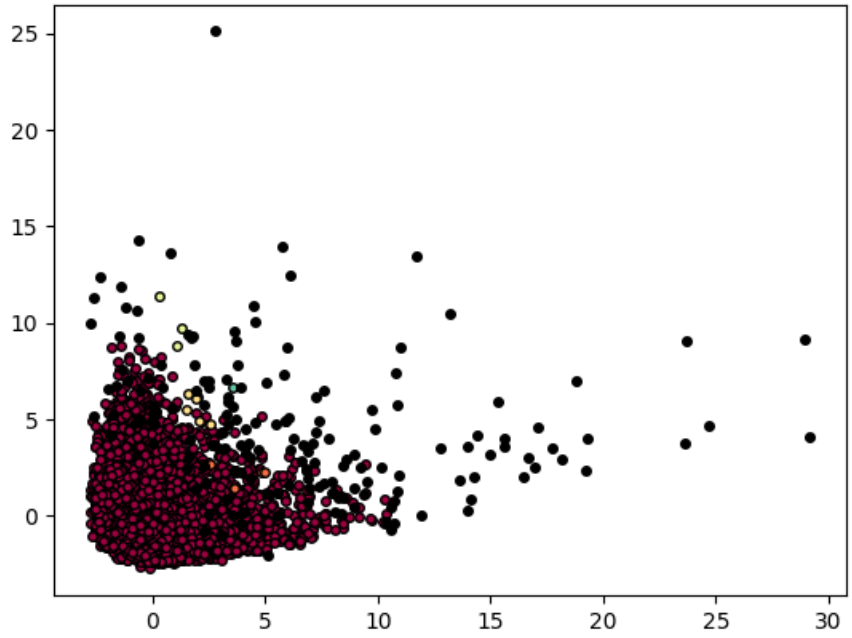

Рисунок 10 – Данные после кластеризации OPTICS с метрикой *manhattan*

Изменим метрику на *cosine*. Не изменяя остальные параметры, все данные объединились в один кластер. Результат после понижения размерности до 2 представлен на рисунке 11.

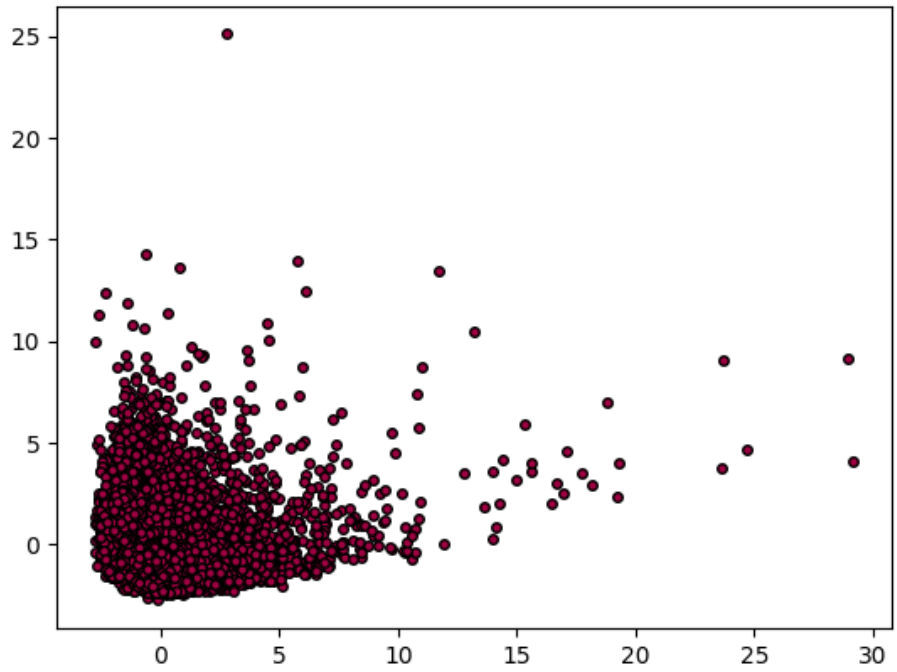

Рисунок 11 – Данные после кластеризации OPTICS с метрикой *cosine*

Изменим метрику на *cityblock*. Не изменяя остальные параметры, получилось 27 кластеров при 20% не кластеризованных наблюдений. Результат после понижения размерности до 2 представлен на рисунке 12.

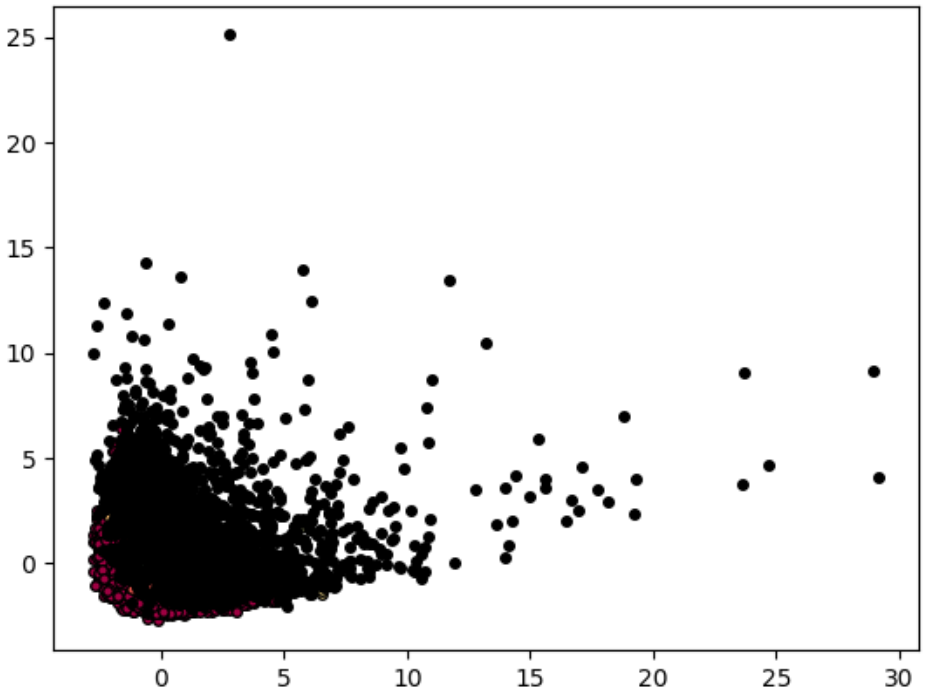

Рисунок 12 – Данные после кластеризации OPTICS с метрикой *cityblock*

Изменим метрику на *l1*. Не изменяя остальные параметры, получилось 27 кластеров при 20% не кластеризованных наблюдений. Результат после понижения размерности до 2 представлен на рисунке 13.

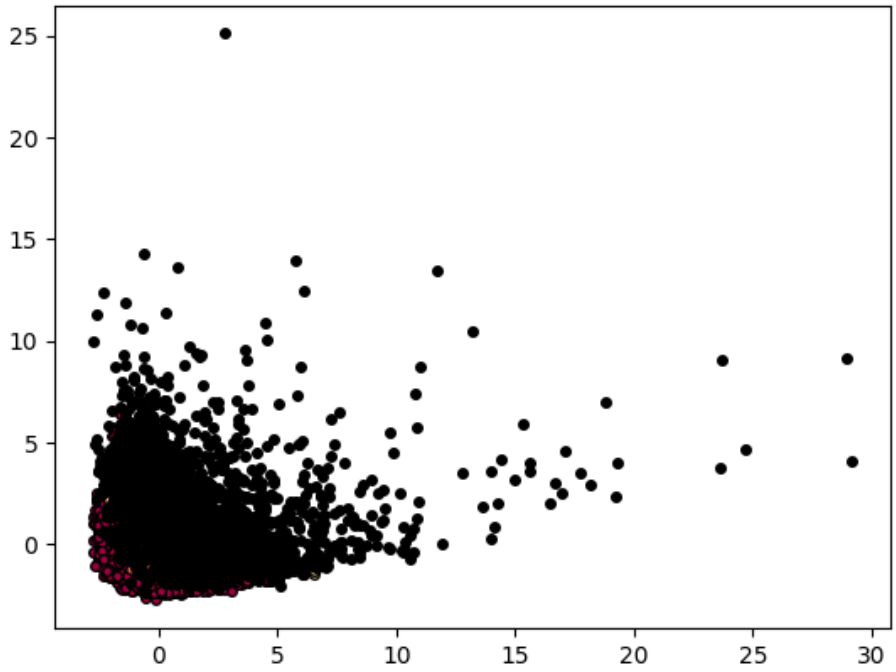

Рисунок 13 – Данные после кластеризации OPTICS с метрикой *l1*

Выводы

В ходе выполнения лабораторной работы было проведено ознакомление с методами кластеризации модуля Sklearn.

Был изучен метод DBSCAN, позволяющий находить кластеры на основании плотности расположения точек. Были построены графики зависимости количества получаемых кластеров и процента не кластеризованных наблюдений от минимального количества наблюдений в кластере и радиуса поиска соседних наблюдений и выявлено, как данные параметры влияют друг на друга. Были подобраны оптимальные параметры DBSCAN, а именно *eps*=2 и *min_samples*=3, при которых образуется 6 кластеров и 6% не кластеризованных наблюдений. После чего результат кластеризации после понижения размерности до 2 был визуализирован. Было выяснено, что большая часть наблюдений попала в один кластер. Также образовалось несколько маленьких кластеров.

Был изучен метод OPTICS. Его основное преимущество перед DBSCAN – способность обнаруживать кластеры различной плотности. Для него также были подобраны оптимальные параметры, а именно *max_eps*=3 и *min_samples*=3, при которых образуется 5 кластеров, а не кластеризованных наблюдений 2%. После чего результат кластеризации после понижения размерности до 2 был визуализирован. Было выяснено, результат оказался схож с DBSCAN, однако удалось кластеризовать больший процент наблюдений. Также данный алгоритм был протестирован для различных метрик.

Код программы, написанной для выполнения лабораторной работы представлен в приложении А.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
import pandas as pd
import numpy as np
from sklearn import cluster
from sklearn import preprocessing
from sklearn import decomposition
import matplotlib.gridspec as gridspec
import matplotlib.pyplot as plt

def plotClusts(scaled_data,clustering):
    pca = decomposition.PCA(n_components = 2)
    pca_data = pca.fit_transform(scaled_data)

    unique_labels = set(clustering.labels_)
    core_samples_mask = np.zeros_like(clustering.labels_, dtype=bool)
       colors = [plt.cm.Spectral(each) for each in np.linspace(0, 1,
len(unique_labels))]

    for k, col in zip(unique_labels, colors):
        if k == -1:
            col = [0, 0, 0, 1]
        class_member_mask = clustering.labels_ == k
        xy = pca_data[class_member_mask & core_samples_mask]
                                            plt.plot(xy[:,          0],xy[:,
1],"o",markerfacecolor=tuple(col),markeredgecolor="k",markersize=8)
        xy = pca_data[class_member_mask & ~core_samples_mask]
                                            plt.plot(xy[:,          0],xy[:,
1],"o",markerfacecolor=tuple(col),markeredgecolor="k",markersize=4)
    plt.show()

data = pd.read_csv('CC GENERAL.csv').iloc[:,1:].dropna()
print(data)
scaled_data = preprocessing.StandardScaler().fit_transform(data)

clustering = cluster.DBSCAN().fit(scaled_data)
print(set(clustering.labels_))
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))

eps=[n/1000 for n in range(1,1000,10)]
yp,yc=[],[]
for x in eps:
    clustering = cluster.DBSCAN(eps=x).fit(scaled_data)
    yc.append(len(set(clustering.labels_)) - 1)
                      yp.append(list(clustering.labels_).count(-1)          /
len(list(clustering.labels_))*100)
plt.plot(eps,yc)
plt.xlabel("eps")
plt.ylabel("Количество кластеров")
plt.show()
plt.plot(eps,yp)
plt.xlabel("eps")
plt.ylabel("Процент не кластеризованных наблюдений")
plt.show()

ms=[n for n in range(2,50,1)]
yp,yc=[],[]
for x in ms:
    clustering = cluster.DBSCAN(min_samples=x).fit(scaled_data)
    yc.append(len(set(clustering.labels_)) - 1)
```

```
                        yp.append(list(clustering.labels_).count(-1)          /
len(list(clustering.labels_))*100)
plt.plot(ms,yc)
plt.xlabel("min_samples")
plt.ylabel("Количество кластеров")
plt.show()
plt.plot(ms,yp)
plt.xlabel("min_samples")
plt.ylabel("Процент не кластеризованных наблюдений")
plt.show()

clustering = cluster.DBSCAN(eps=2,min_samples=3).fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data, clustering)

clustering         =         cluster.OPTICS(min_samples=3,         max_eps=3,
cluster_method="dbscan").fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)

space = np.arange(len(scaled_data))
reachability = clustering.reachability_[clustering.ordering_]
labels = clustering.labels_[clustering.ordering_]

colors = ["g.", "r.", "b.", "y.", "c."]
for klass, color in enumerate(colors):
    Xk = space[labels == klass]
    Rk = reachability[labels == klass]
    plt.plot(Xk, Rk, color, alpha=0.3)
plt.plot(space[labels == -1], reachability[labels == -1], "k.", alpha=0.3)
plt.plot(space, np.full_like(space, 2.0, dtype=float), "k-", alpha=0.5)
plt.plot(space, np.full_like(space, 0.5, dtype=float), "k-.", alpha=0.5)
plt.ylabel("Достижимое расстояние")
plt.title("График достижимости")
plt.show()

clustering         =         cluster.OPTICS(min_samples=3,         max_eps=3,
cluster_method="dbscan",metric='euclidean').fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)

clustering         =         cluster.OPTICS(min_samples=3,         max_eps=3,
cluster_method="dbscan",metric='manhattan').fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)

clustering         =         cluster.OPTICS(min_samples=3,         max_eps=3,
cluster_method="dbscan",metric='cosine').fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)

clustering         =         cluster.OPTICS(min_samples=3,         max_eps=3,
cluster_method="dbscan",metric='cityblock').fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)
```

```
clustering = cluster.OPTICS(min_samples=3, max_eps=3,
cluster_method="dbscan",metric='l1').fit(scaled_data)
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
plotClusts(scaled_data,clustering)
```