МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

**ОТЧЕТ**

по лабораторной работе № 6

по дисциплине «Машинное обучение»

Тема: Кластеризация (DBSCAN, OPTICS)

Студент гр. 8304 _____ Сергеев А.Д.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

## Цель работы

Ознакомиться с методами кластеризации модуля *Sklearn*.

## Ход работы

### Загрузка данных

Данные загружены в датафрейм.

```
data = pd.read_csv('CC GENERAL.csv').iloc[:,1:].dropna()
data
```

| | BALANCE | BALANCE_FREQUENCY | PURCHASES | ONEOFF_PURCHASES | INSTALLMENTS_PURCHASES | CA |
|---|---|---|---|---|---|---|
| **0** | 40.900749 | 0.818182 | 95.40 | 0.00 | 95.40 | 0.0 |
| **1** | 3202.467416 | 0.909091 | 0.00 | 0.00 | 0.00 | 644 |
| **2** | 2495.148862 | 1.000000 | 773.17 | 773.17 | 0.00 | 0.0 |
| **4** | 817.714335 | 1.000000 | 16.00 | 16.00 | 0.00 | 0.0 |
| **5** | 1809.828751 | 1.000000 | 1333.28 | 0.00 | 1333.28 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... |
| **8943** | 5.871712 | 0.500000 | 20.90 | 20.90 | 0.00 | 0.0 |
| **8945** | 28.493517 | 1.000000 | 291.12 | 0.00 | 291.12 | 0.0 |
| **8947** | 23.398673 | 0.833333 | 144.40 | 0.00 | 144.40 | 0.0 |
| **8948** | 13.457564 | 0.833333 | 0.00 | 0.00 | 0.00 | 36. |
| **8949** | 372.708075 | 0.666667 | 1093.25 | 1093.25 | 0.00 | 127 |

Рисунок 1 – Исходные данные

### DBSCAN

1. Проведена кластеризация методом k-средних. Данные стандартизированы, т.к. лежат в разных шкалах.

```
data = np.array(data, dtype='float')
min_max_scaler = pp.StandardScaler()
scaled_data = min_max_scaler.fit_transform(data)
scaled_data
```

```
array([[-0.74462486, -0.37004679, -0.42918384, ..., -0.30550763,
        -0.53772694,  0.35518066],
       [ 0.76415211,  0.06767893, -0.47320819, ...,  0.08768873,
         0.21238001,  0.35518066],
       [ 0.42660239,  0.50540465, -0.11641251, ..., -0.09990611,
        -0.53772694,  0.35518066],
       ...,
       [-0.75297728, -0.29709491, -0.40657175, ..., -0.32957217,
         0.30614422, -4.22180042],
       [-0.75772142, -0.29709491, -0.47320819, ..., -0.34081076,
         0.30614422, -4.22180042],
       [-0.58627829, -1.09958965,  0.03129519, ..., -0.32709767,
        -0.53772694, -4.22180042]])
```

Рисунок 2 – Данные после стандартизации

2. Проведена кластеризация методом DBSCAN при параметрах по умолчанию.

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

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30
36
0.7512737378415933
```

Рисунок 3 – Метко кластеров, их количество и процент нераспределенных элементов

| Параметр | Описание |
|---|---|
| eps: float, default=0.5 | Максимальное расстояние между двумя наблюдениями, чтобы один считался соседним с другим (радиус окрестности наблюдения). |
| min_samples: int, default=5 | Минимальное количество наблюдений в окрестности точки, чтобы считать ее базовой (включая саму точку). |
| metric: string or callable, default='euclidean' | Метрика для вычисления расстояния между экземплярами в массиве признаков. |
| metric_params: *dict, default=None* | Дополнительные аргументы ключевых слов для функции метрики. |
| algorithm: {'auto', 'ball_tree', 'kd_tree', 'brute'}, default = 'auto' | Алгоритм, который будет использоваться для вычисления точечных расстояний и поиска ближайших соседей. |
| leaf_size: *int, default=30* | Размер листа. Это может повлиять на скорость построения и запроса, а также на память, необходимую для хранения дерева. Оптимальное значение зависит от задачи. |
| P: *float, default=None* | Мощность метрики, используемой для вычисления расстояния между точками. Если *None*, то эквивалентно евклидову расстоянию. |
| n_jobs: *int, default=None* | Количество параллельных процессов для выполнения. |

3. Построен график зависимости количества кластеров и процента не кластеризованных наблюдений от максимальной рассматриваемой дистанции.

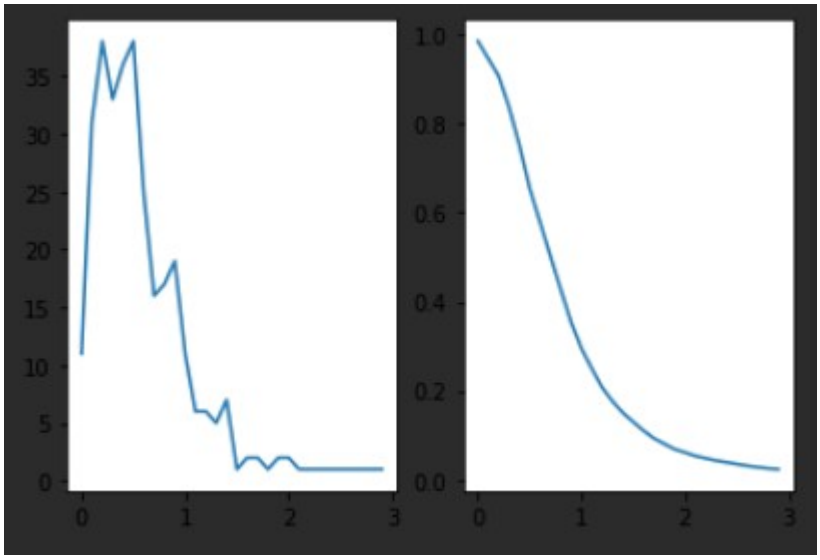


Рисунок 4 – Зависимост количества кластеров и процента выпавших наблюдений от eps

С увеличением eps все больше точек попадают в какой-либо кластер, в конце все точки будут находится в одном кластере.

4. Построен график зависимости количества кластеров и процента не кластеризованных наблюдений от минимального значения количества точек, образующих кластер.

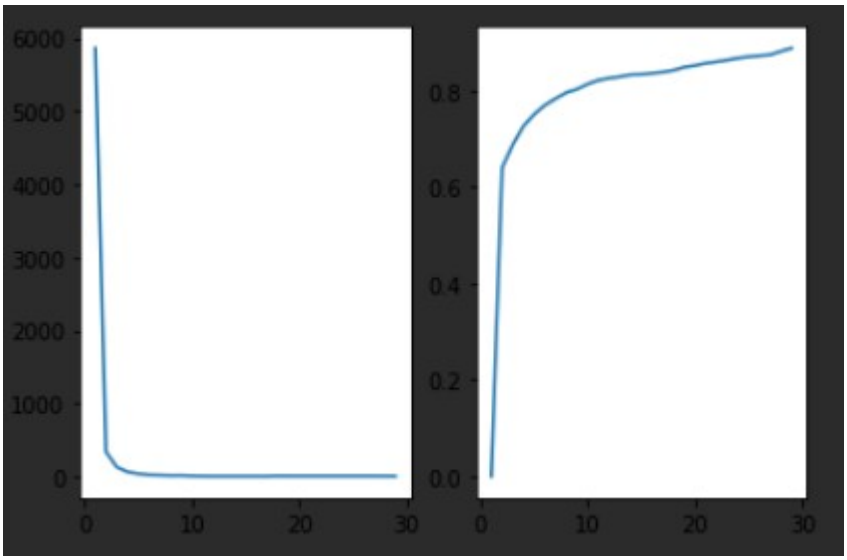


Рисунок 5 – Зависимост количества кластеров и процента выпавших наблюдений от min_samples

Увеличение min_samples приводит к увеличению количества выпавших наблюдений и к уменьшению количества кластеров.

5. Определены значения параметров, при которых количество кластеров получается от 5 до 7, и процент не кластеризованных наблюдений не

превышает 12%.

```
clustering = DBSCAN(eps=5.6, min_samples=2).fit(scaled_data)
print(f"Unclustered percent: {list(clustering.labels_).count(-1) / len(list(clustering.labels_))}")
print(f"Clusters number: {len(set(clustering.labels_)) - 1}")

Unclustered percent: 0.002779064381658175
Clusters number: 6
```

Рисунок 6 – значения eps и min_samples, при которых достигается требуемый процент не кластеризованных измерений и количество кластеров.

6. Размерность данных понижена до 2 с помощью метода главных компонент.

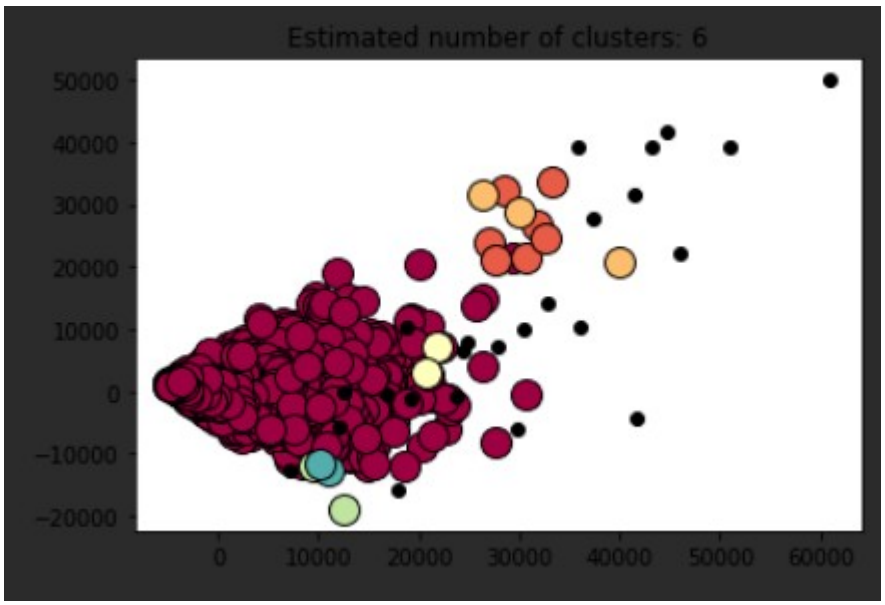


Рисунок 7 – Кластеризация данных пониженной размерности

## OPTICS

| Параметр | Описание |
|---|---|
| max_eps: float, default=inf | Максимальное расстояние между двумя наблюдениями, чтобы одно считалось соседним с другим (радиус окрестности наблюдения). |
| min_samples: int>1 or float in [0, 1], default=5 | Количество наблюдений в окрестности точки, чтобы считать ее базовой. |
| metric: string or callable, default=’minkowski’ | Метрика для вычисления расстояния. |
| p: int, default = 2 | Параметр для метрики Минковского. |
| cluster_method: string, default = ‘xi’ | Метод  извлечения кластеров. Также можно поставить 'dbscan'. |

| | |
|---|---|
| eps | Максимальное расстояние между двумя наблюдениями, чтобы один считался соседним с другим (радиус окрестности наблюдения). Нужен только при cluster_method=dbscan. |
| xi: float in [0,1], default=0.05 | Определяет минимальную крутизну на графике достижимости, который составляет границу кластера. |
| predecessor_correction: bool, default=True | Коррекция кластеров в соответствии с предшественниками, рассчитанными OPTICS. Этот параметр оказывает минимальное влияние на большинство наборов данных. Используется только когда cluster_method = 'xi'. |
| min_cluster_size: int>1 or float in [0, 1], default=None | Минимальное количество выборок в кластере OPTICS, выраженное в виде абсолютного числа или доли от количества выборок (округленное до не менее 2). Если None, вместо этого используется значение min_samples. Используется только когда cluster_method = 'xi'. |
| algorithm: {‘auto’, ‘ball_tree’, ‘kd_tree’, ‘brute’}, default = ‘auto’ | Алгоритм, который будет использоваться для вычисления точечных расстояний и поиска ближайших соседей. |
| leaf_size: *int, default=30* | Размер листа. Это может повлиять на скорость построения и запроса, а также на память, необходимую для хранения дерева. Оптимальное значение зависит от задачи. |
| Memory: *str or object with the joblib.Memory interface, default=None* | Путь к каталогу кэширования. Используется для кэширования выходных данных вычисления дерева. По умолчанию кэширование не выполняется. |
| n_jobs: *int, default=None* | Количество параллельных заданий, выполняемых для поиска соседей. |

1. Результаты кластеризации методом OPTICS близки к результатам DBSCAN при cluster_method=dbscan, max_eps=2, min_samples=3.

```
clustering = OPTICS(cluster_method='dbscan', max_eps=2, min_samples=3).fit(scaled_data)
print(f"Unclustered percent: {list(clustering.labels_).count(-1) / len(list(clustering.labels_))}")
print(f"Clusters number: {len(set(clustering.labels_)) - 1}")

Unclustered percent: 0.06310792033348772
Clusters number: 6
```

Рисунок 7 – значения eps и min_samples, при которых достигается требуемый процент не кластеризованных измерений и количество кластеров.

Процесс определения базовых точек в OPTICS идентичен DBSCAN, однако в OPTICS для точек вычисляются и сохраняются расстояния достижимости, на основе которых наблюдения выстраиваются в кластере, сохраняя при этом иерархическую структуру.

2. Построен график достижимости

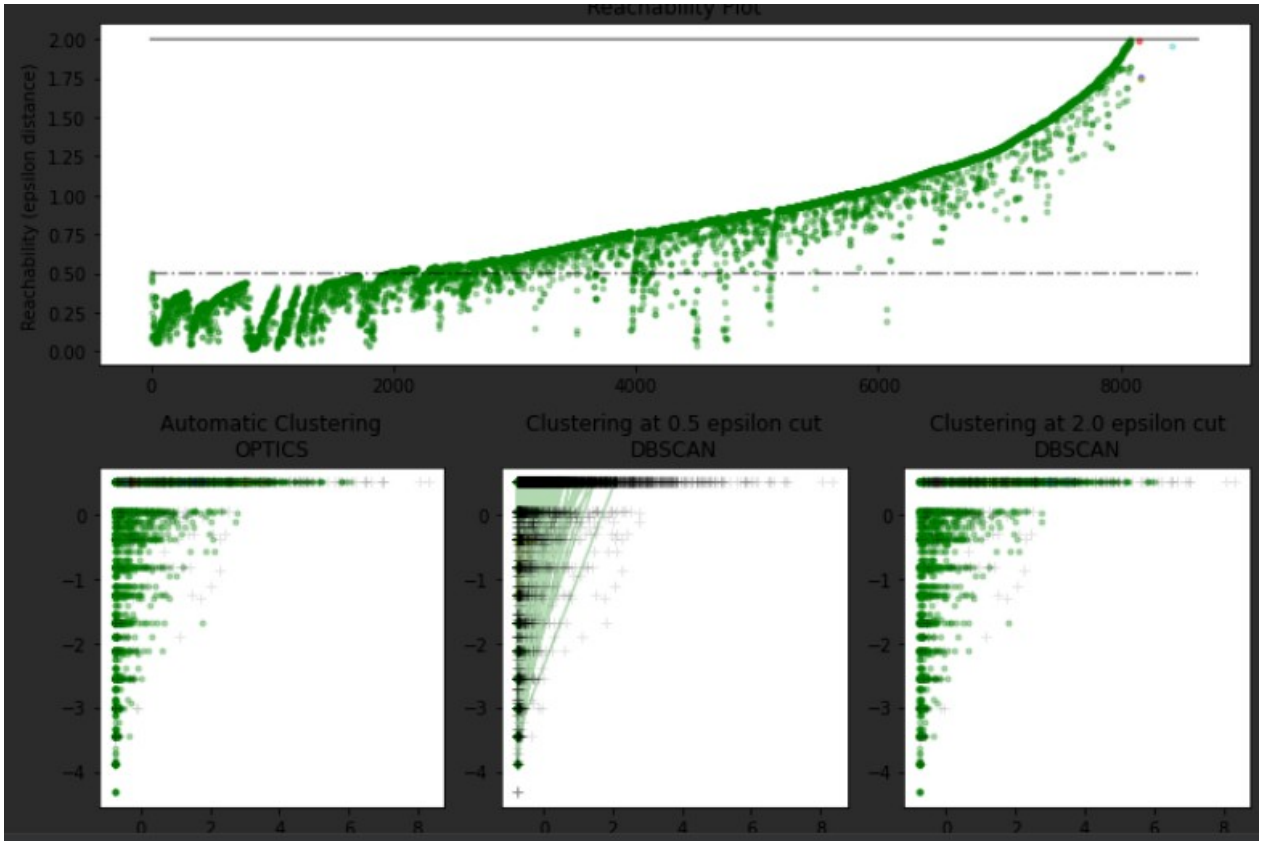


Рисунок 4 – График достижимости

3. Метод OPTICS исследован с использованием различных метрик

| Метрика | Количество кластеров | Процент выпавших наблюдений |
|---|---|---|
| canberra | 106 | 0.88722 |
| chebyshev | 143 | 0.86869 |
| manhattan | 99 | 0.90840 |
| euclidean | 111 | 0.89961 |
| sqeuclidean | 146 | 0.86962 |

**Выводы**

В ходе лабораторной работы изучены такие методы кластеризации модуля *Sklearn*, как DBSCAN и OPTICS. При cluster_method=“xi“ OPTICS разделяет данные на большое число кластеров, малое количество кластеров достигается только при большом количестве выпавших наблюдений.