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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Машинное обучение»

Тема: Предобработка данных.

Студент гр. 6302 _____ Барбарич И.Г.

Руководитель _____ Жангиров Т. Р.

Санкт-Петербург
2020

Цель работы

Ознакомиться с методами кластеризации модуля Sklearn

Загрузка данных:

1. Загрузить датасет по ссылке: https://www.kaggle.com/arjunbhasin2013/ccdata . Данные представлены в виде csv файла. Датасет содержит пропущенные значения

2. Создать Python скрипт. Загрузить данные в датафрейм, убрав столбец с метками и откинув наблюдения с пропущенными значениями

[10]:
```
import pandas as pd
import numpy as np

data = pd.read_csv('E:/мага/1 семестр/машинное обучение/лаба 5/archive/CC GENERAL.csv').iloc[:,1:].dropna()
```

[11]:
```
data
```

[11]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PURCHASES_FREQUENCY	PURCHASES_INSTALLMENTS_FREQUENCY	CASH_ADVANCE_
0	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.166667	0.000000	0.083333	
1	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.000000	0.000000	0.000000	
2	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.000000	1.000000	0.000000	
4	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.083333	0.083333	0.000000	
5	1809.828751	1.000000	1333.28	0.00	1333.28	0.000000	0.666667	0.000000	0.583333	
...	...	...	...	...	...	...	...	...	...	
8943	5.871712	0.500000	20.90	20.90	0.00	0.000000	0.166667	0.166667	0.000000	
8945	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.000000	0.000000	0.833333	
8947	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.833333	0.000000	0.666667	
8948	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.000000	0.000000	0.000000	
8949	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	0.666667	0.666667	0.000000	

8636 rows × 17 columns

[ ]:

DBSCAN

Так как разные признаки лежат в разных шкалах, то стандартизируем данные

[22]:
```
from sklearn import preprocessing

data = np.array(data, dtype='float')
min_max_scaler = preprocessing.StandardScaler()

scaled_data = min_max_scaler.fit_transform(data)
scaled_data
```

[22]:
```
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]])
```

Проведем кластеризацию методов DBSCAN при параметрах по умолчанию. Выведем метки кластеров, количество кластеров, а также процент наблюдений, которые кластеризовать не удалось

```
[38]: #Проведем кластеризацию методов DBSCAN при параметрах по умолчанию. Выведем
      #метки кластеров, количество кластеров, а также процент наблюдений, которые
      #кластеризовать не удалось
      from sklearn.cluster import DBSCAN, OPTICS

      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, 31, 32, 33, 34, 35, -1}
      36
      0.7512737378415933
```

Опишите все параметры, которые принимает DBSCAN:

Eps (float), default=0.5 - Максимальное расстояние между двумя образцами для того, чтобы один рассматривался как находящийся в окрестности другого.

min_samples (int), default=5 - Число выборок (или общий вес) в окрестности точки, рассматриваемой в качестве основной точки. Это включает в себя и саму точку.

Metric (string), or callable, default='euclidean' - Метрика, используемая при вычислении расстояния между экземплярами в массиве объектов. Если метрика является строкой или вызываемой, она должна быть одной из опций, разрешенных

metric_params (dict), default=None - Дополнительные аргументы ключевого слова для функции метрики.

Additional keyword arguments for the metric function.

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 передается в BallTree или cKDTree. Это может повлиять на скорость построения и запроса, а также на объем памяти, необходимый для хранения дерева. Оптимальное значение зависит от характера задачи.

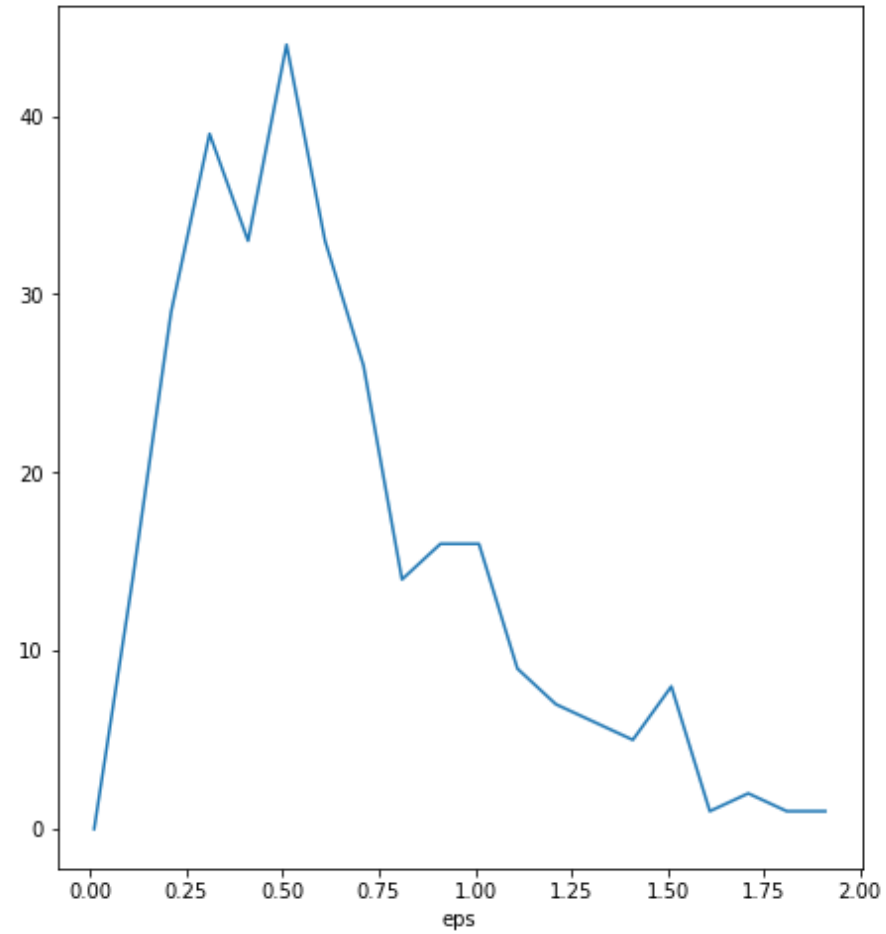
pfloat, default=None - Мощность метрики Минковского, которая будет использоваться для вычисления расстояния между точками. Если нет, то p=2 (эквивалентно евклидову расстоянию).

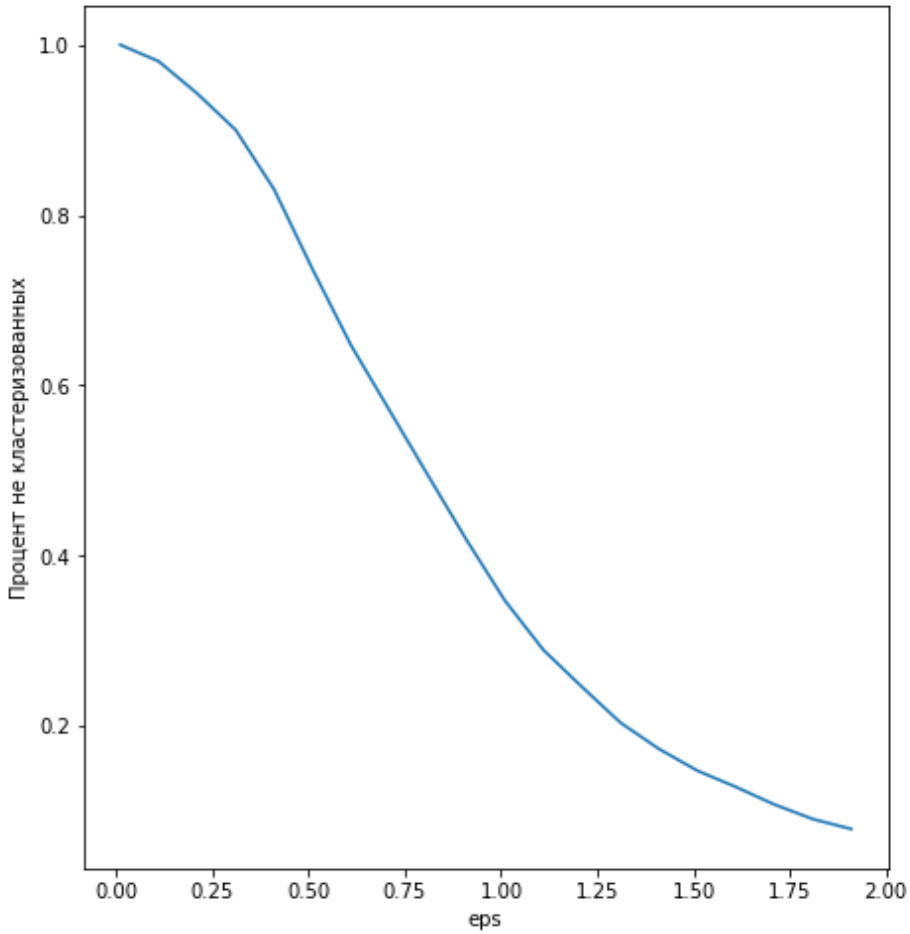
n_jobsint, default=None

Количество выполняемых параллельных заданий. None означает 1, если только в контексте joblib.parallel_backend. -1 означает использование всех процессоров.

4. Постройте график количества кластеров и процента не кластеризованных наблюдений в зависимости от максимальной рассматриваемой дистанции между наблюдениями. Минимальное значение количества точек образующих, кластер оставить по умолчанию

[63]:
```
#Постройте график количества кластеров и процента не кластеризованных наблюдений
#в зависимости от максимальной рассматриваемой дистанции между наблюдениями.
#Минимальное значение количества точек образующих, кластер оставить по
#умолчанию

eps_ = []
clust = []
non_clust = []

for eps in np.arange(0.01, 2.0, 0.1):
    clustering = DBSCAN(eps=eps).fit(scaled_data)
    clust.append(len(set(clustering.labels_)) - 1)
    non_clust.append(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
    eps_.append(eps)

fig, ax = plt.subplots(1, 2, figsize=(16,8))

ax[0].plot(eps_, clust)
ax[0].set_xlabel('eps')
ax[0].set_ylabel('')

ax[1].plot(eps_, non_clust)
ax[1].set_xlabel('eps')
ax[1].set_ylabel('Процент не кластеризованных')

plt.show()
```

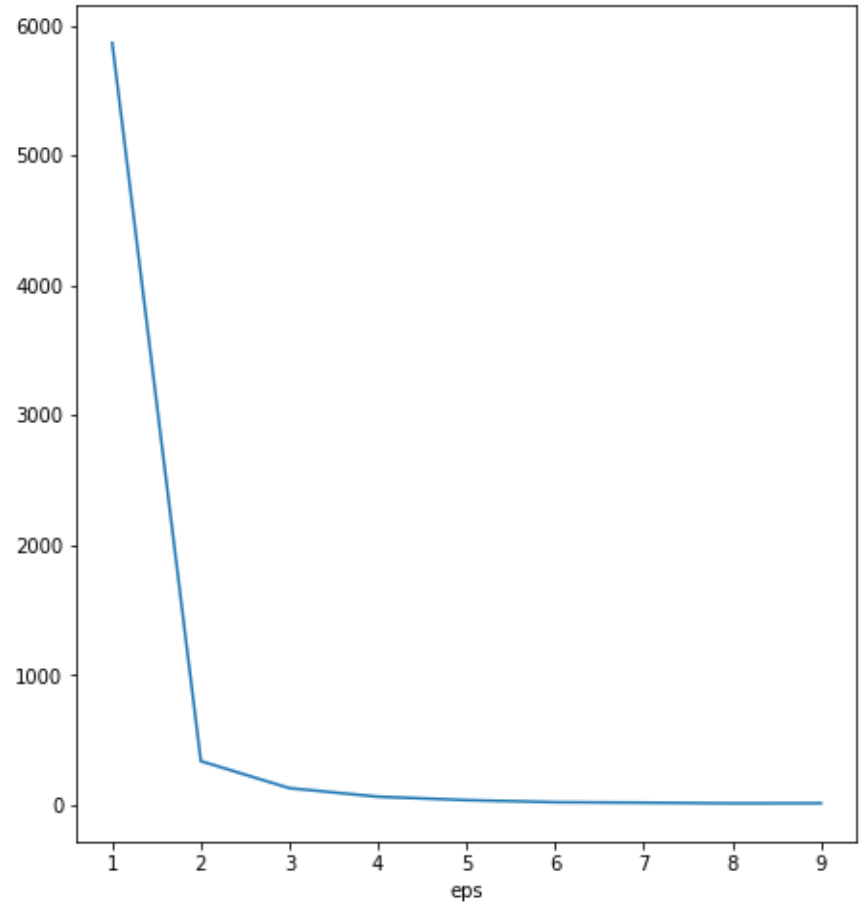


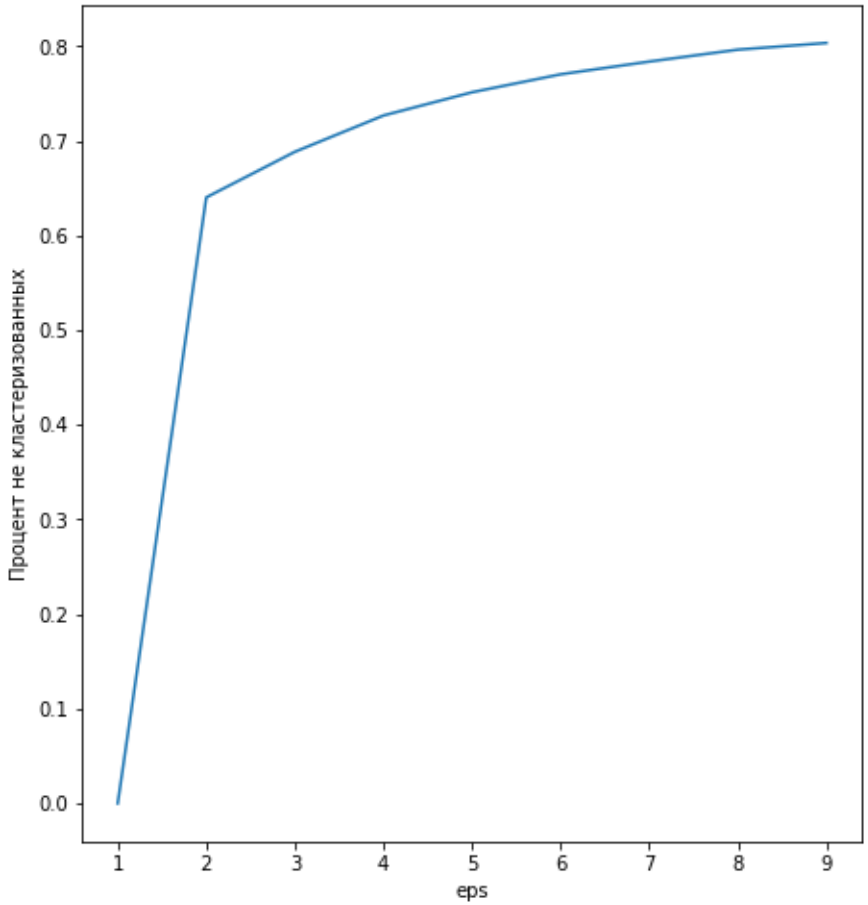


5. Постройте график количества кластеров и процента не кластеризованных наблюдений в зависимости от минимального значения количества точек, образующих кластер. Максимальную рассматриваемую дистанцию между наблюдениями оставьте по умолчанию

[67]:
```
# Постройте график количества кластеров и процента не кластеризованных наблюдений
#в зависимости от минимального значения количества точек, образующих кластер.
#Максимальную рассматриваемую дистанцию между наблюдениями оставьте по
#умолчанию

sample_ = []
clust = []
non_clust = []

for sample in np.arange(1, 10, 1):
    clustering = DBSCAN(min_samples=sample).fit(scaled_data)
    clust.append(len(set(clustering.labels_)) - 1)
    non_clust.append(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))
    sample_.append(sample)

fig, ax = plt.subplots(1, 2, figsize=(16,8))

ax[0].plot(sample_, clust)
ax[0].set_xlabel('eps')
ax[0].set_ylabel('')

ax[1].plot(sample_, non_clust)
ax[1].set_xlabel('eps')
ax[1].set_ylabel('Процент не кластеризованных')

plt.show()
```

```
plt.show()
```





6. Определите значения параметров, при котором количество кластеров получается от 5 до 7, и процент не кластеризованных наблюдений не превышает 12%.

eps eps

[77]:
```
#Определите значения параметров, при котором количество кластеров получается от 5
#до 7, и процент не кластеризованных наблюдений не превышает 12%.

scan_f = pd.DataFrame(columns=['min_samples', 'eps', 'clusters', 'procent'])

for min_samples in np.arange(1, 10, 1):
    for eps in np.arange(0.01, 2.0, 0.1):
        clustering = DBSCAN(eps=eps, min_samples=min_samples).fit(scaled_data)
        len_clusters = len(set(clustering.labels_)) - 1
        procent_non_cluster = list(clustering.labels_).count(-1) / len(list(clustering.labels_))

        scan_f = scan_f.append({'min_samples': min_samples, 'eps': eps, 'clusters': len_clusters, 'procent': procent_non_cluster}, ignore_index=True)
new_data = scan_f[(scan_f.clusters >= 5) & (scan_f.clusters <= 7) & (scan_f.procent <= 0.12)]
new_data
```

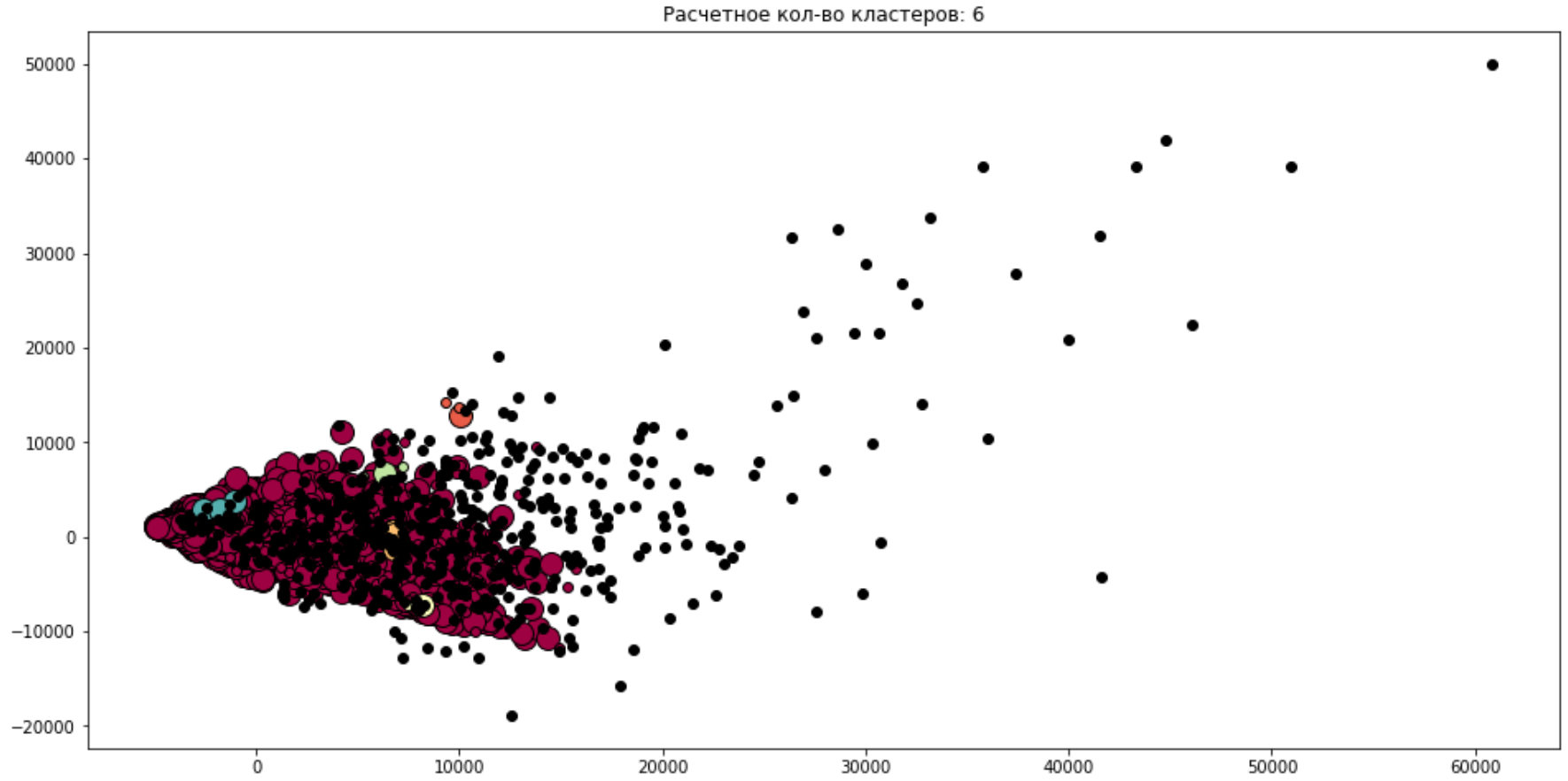
[77]:

	min_samples	eps	clusters	procent
77	4.0	1.71	6.0	0.099352

7. Понизьте размерность данных до 2 при используя метод главных компонент. Визуализируйте результаты кластеризации полученные в пункте 6 (метки должны быть получены на данных до уменьшения размерности). гайд по визуализации

[91]:
```
#Понизьте размерность данных до 2 при используя метод главных компонент.
#Визуализируйте результаты кластеризации полученные в пункте 6 (метки должны быть
#получены на данных до уменьшения размерности)

pca = PCA(n_components=2)
reduced_data  = pca.fit_transform(data)

clustering = DBSCAN(eps=2, min_samples=3, n_jobs=-1).fit(scaled_data)
core_samples_mask = np.zeros_like(clustering.labels_, dtype=bool)
core_samples_mask[clustering.core_sample_indices_] = True
labels = clustering.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]

plt.figure(figsize=(16, 8))
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = reduced_data[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=14)

    xy = reduced_data[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

plt.title('Расчетное кол-во кластеров: %d' % n_clusters_)
plt.show()
```

```
plt.show()
```



OPTICS

Опишите параметры метода OPTICS, а также какими атрибутами он обладает

Параметры:

min_samples (int > 1 или float между 0 и 1, по умолчанию=5) - Число выборок в окрестности точки, рассматриваемой в качестве базовой точки.
max_eps (float, default=np.Inf) - Максимальное расстояние между двумя образцами для того, чтобы один из них рассматривался как находящийся в окрестности другого.
metric str or callable, по умолчанию= ' minkowski’ - Метрика, используемая для вычисления расстояния.
P (int), по умолчанию=2 - gараметр для метрики
metric_paramsdict, default=None - Дополнительные аргументы ключевых слов для метрической функции.
cluster_method (str), default= ' xi’ - Метод экстракции используется для извлечения кластеров с использованием расчетной достижимости и упорядоченности. Возможные значения “ "xi”и "dbscan".
EPS (float), по умолчанию=нет - Максимальное расстояние между двумя образцами для того, чтобы один из них рассматривался как находящийся в окрестности другого.
Xi (float) между 0 и 1, по умолчанию=0,05 - Определяет минимальную крутизну на участке достижимости , образующем границу кластера.
min_cluster_size (int > 1 или float между 0 и 1), default=None - Минимальное число образцов в оптическом кластере, выраженное как абсолютное число или доля от числа образцов (округленное до значения не менее 2).
leaf_sizeint, по умолчанию=30 - Размер листа передается в BallTreeOp KDTree. Это может повлиять на скорость построения и запроса, а также на объем памяти , необходимый для хранения дерева.
n_jobsint, default=None - Количество параллельных заданий для выполнения поиска соседей. None означает 1, если только в joblib.parallel_backendконтексте. -1 означает использование всех процессоров.

Атрибуты

labels_ - ndarray of shape (n_samples,) - Метки кластеров для каждой точки набора данных задаются функцией fit ().

ordering_ - ndarray of shape (n_samples,) - Кластер упорядочил список выборочных индексов.
core_distances_ - ndarray of shape (n_samples,) - Расстояние, на котором каждый образец становится базовой точкой, индексируется по порядку объектов.

cluster_hierarchy_ - ndarray of shape (n_clusters, 2) - Список кластеров в виде [start, end] в каждой строке, со всеми индексами включительно.

Найдите такие параметры метода OPTICS (*max_eps *и min_samples) при которых, чтобы получить результаты близкие к результатам DBSCAN из пункта 6

[120]:
```
clustering = OPTICS(max_eps=1.7, min_samples=4, cluster_method='dbscan', n_jobs=-1).fit(scaled_data)

len_clusters = len(set(clustering.labels_)) - 1
#procent_non_cluster = list(clustering.labels_).count(-1) / len(list(clustering.labels_))

print(len_clusters)
```

```
5
```

[121]:
```
procent_non_cluster = list(clustering.labels_).count(-1) / len(list(clustering.labels_))
print(procent_non_cluster)
```

```
0.10282538212135248
```

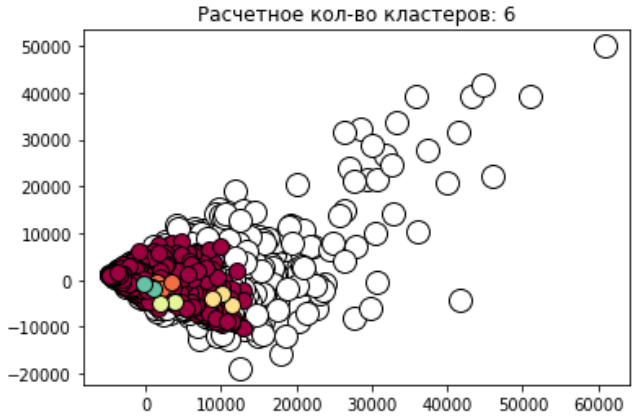
В чем отличия от метода OPTICS от метода DBSCAN

В Optics параметр eps не учитывается, если и учитывается, то только задает максимальное значение, в отличие от DBSCAN, где значение eps является важным параметром.

3. Визуализируйте полученный результат, а также постройге график достижимости (reachable plot) гайд

```
0.10282538212135248
```

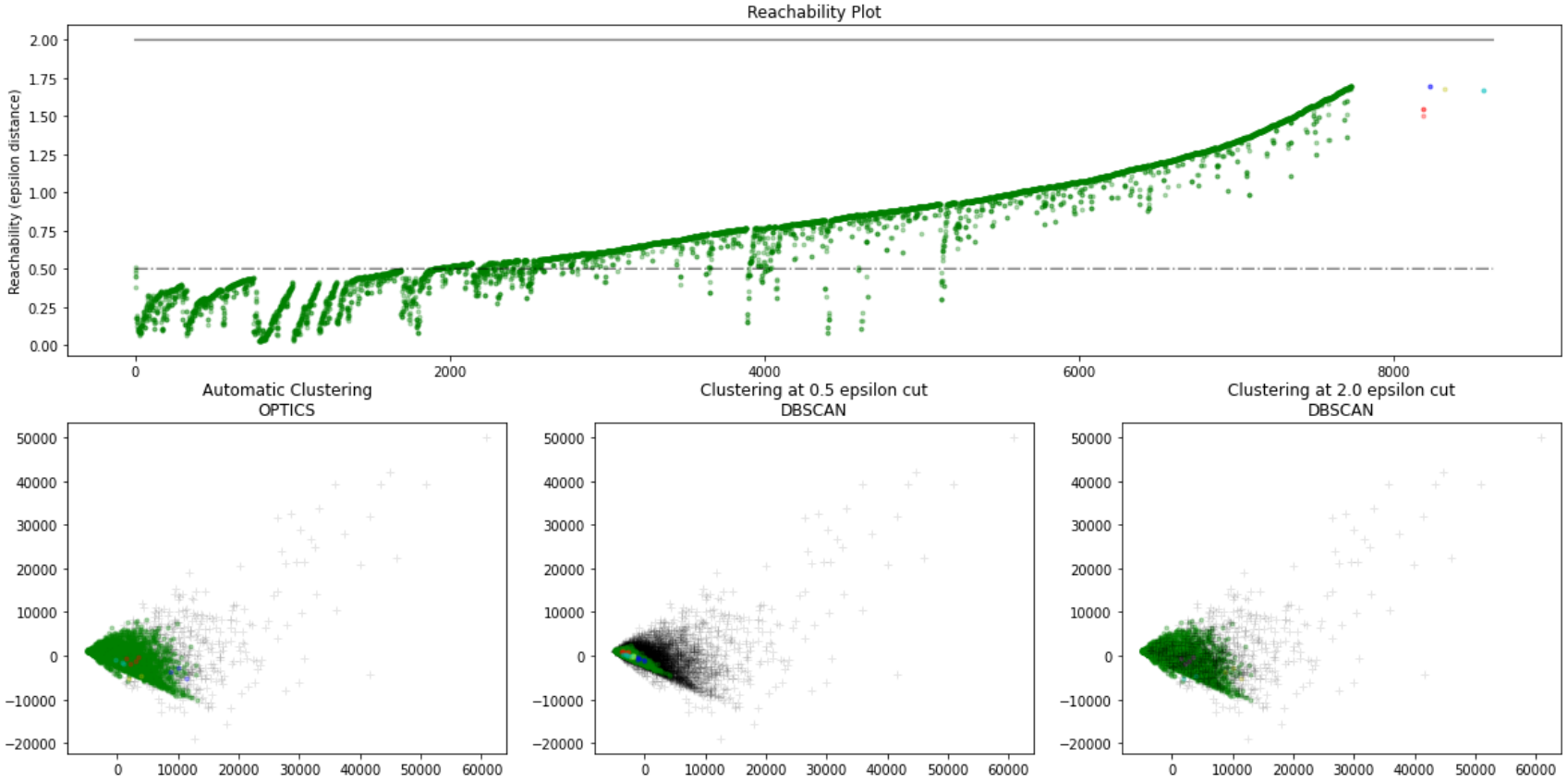
[135]:
```
labels = clustering.labels_

pca = PCA(n_components=2)
pca_data  = pca.fit_transform(data)

unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]

xy = pca_data[(labels == -1)]
plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor='w',
         markeredgecolor='k', markersize=14)

for k, col in zip(unique_labels, colors):
    if k == -1:
        continue

    class_member_mask = (labels == k)

    xy = pca_data[class_member_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=10)

plt.title('Расчетное кол-во кластеров: %d' % n_clusters_)
plt.show()
```



4. Исследуйте работу

```
<Figure size 432x288 with 0 Axes>
```

```
[147]: show_optics(clustering)
```



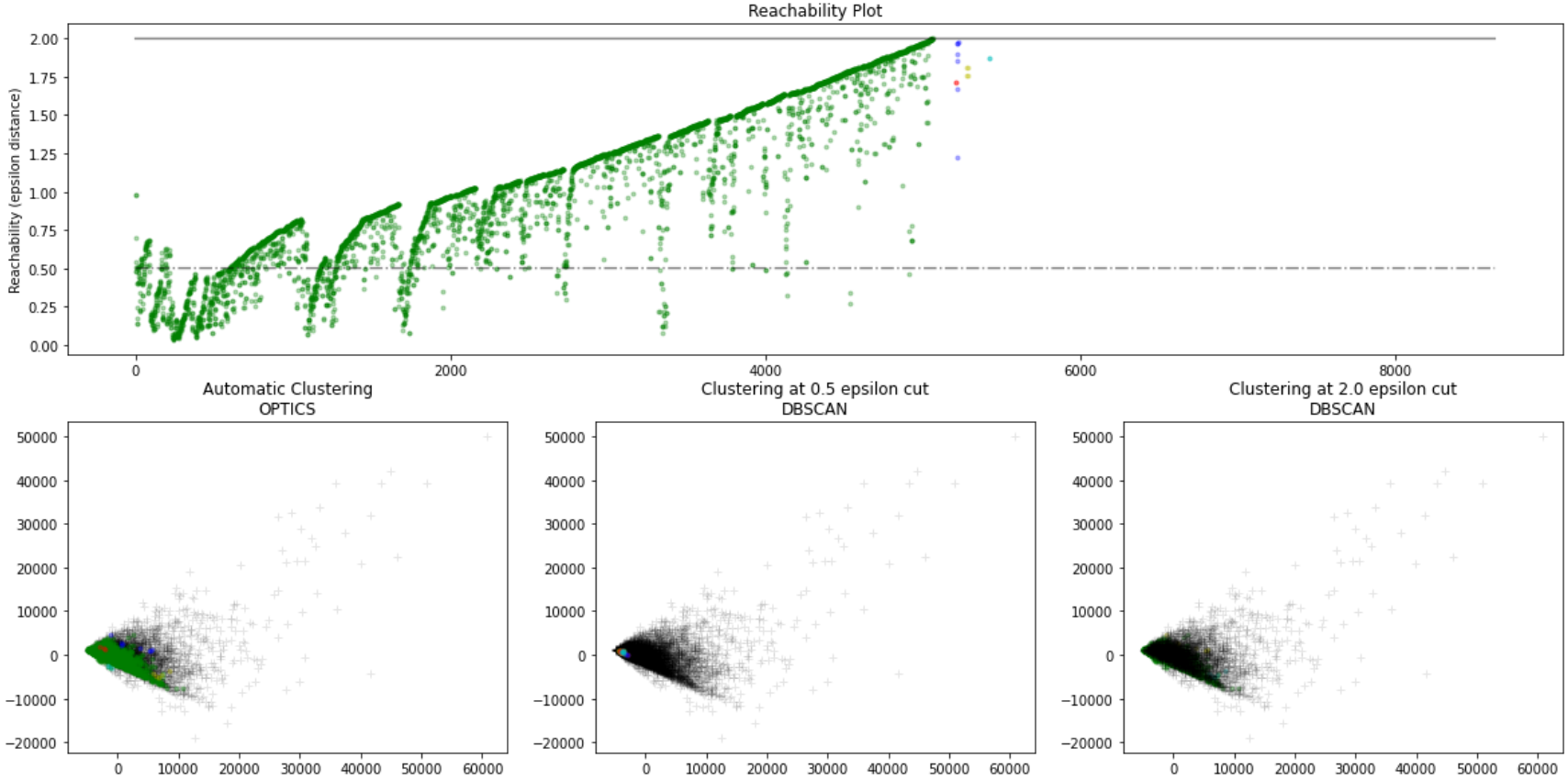
метода OPTICS с использованием различных метрик (выберите не менее 5 метрик)

[148]:
```
metrics = ['cityblock', 'cosine', 'chebyshev', 'l1', 'l2']
for metric in metrics:
    clustering = OPTICS(min_samples=3, max_eps=2, n_jobs=-1, cluster_method="dbscan", metric=metric).fit(scaled_data)
    num_of_clusters = len(set(clustering.labels_)) - 1
    not_classified = list(clustering.labels_).count(-1) / len(list(clustering.labels_))
    print('clusters: {}, not classified: {:.2f}'.format(num_of_clusters, not_classified))
    show_optics(clustering)
```

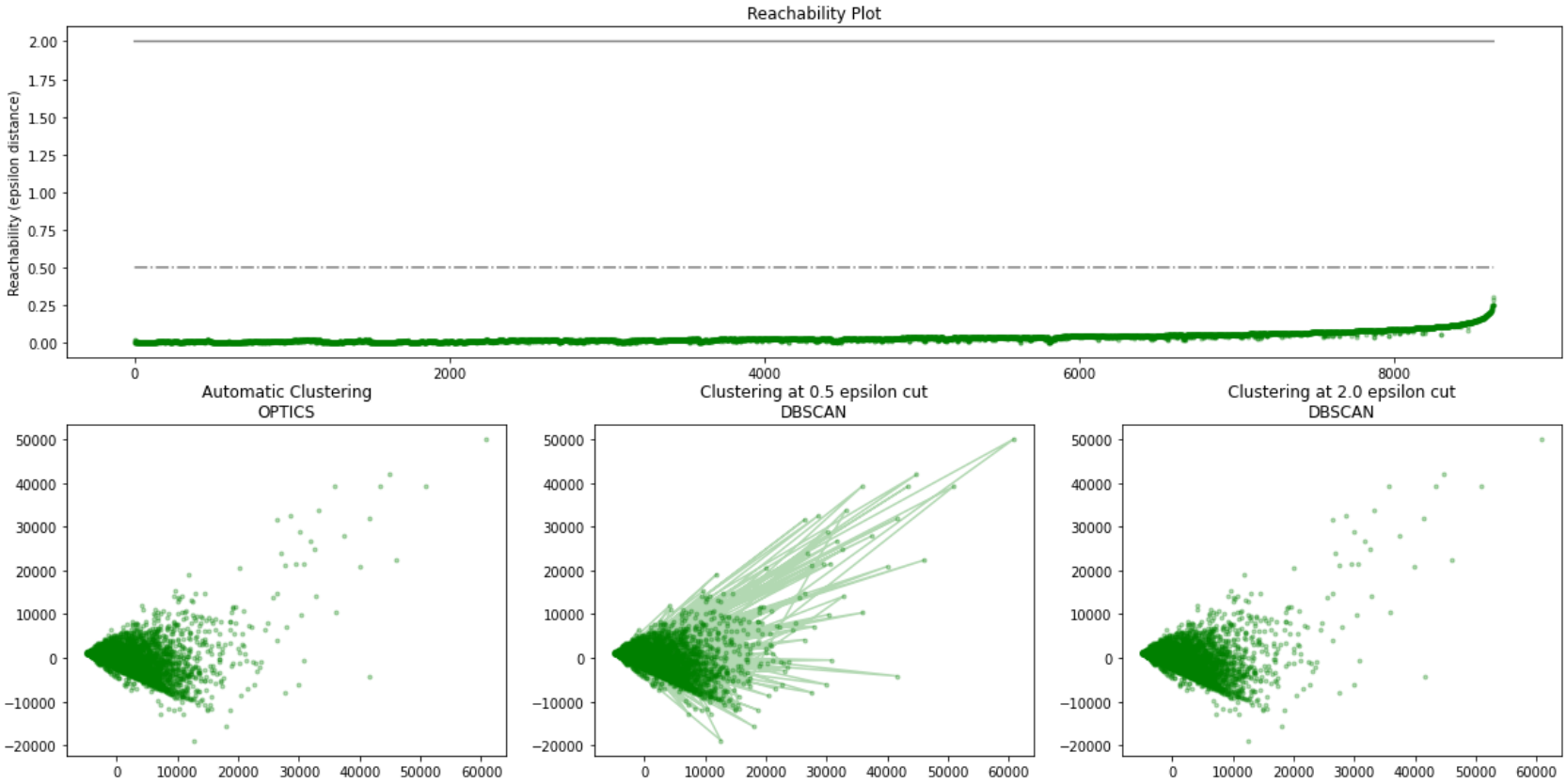
```
clusters: 55, not classified: 0.39
clusters: 0, not classified: 0.00
clusters: 2, not classified: 0.01
clusters: 55, not classified: 0.39
clusters: 6, not classified: 0.06
```

1.

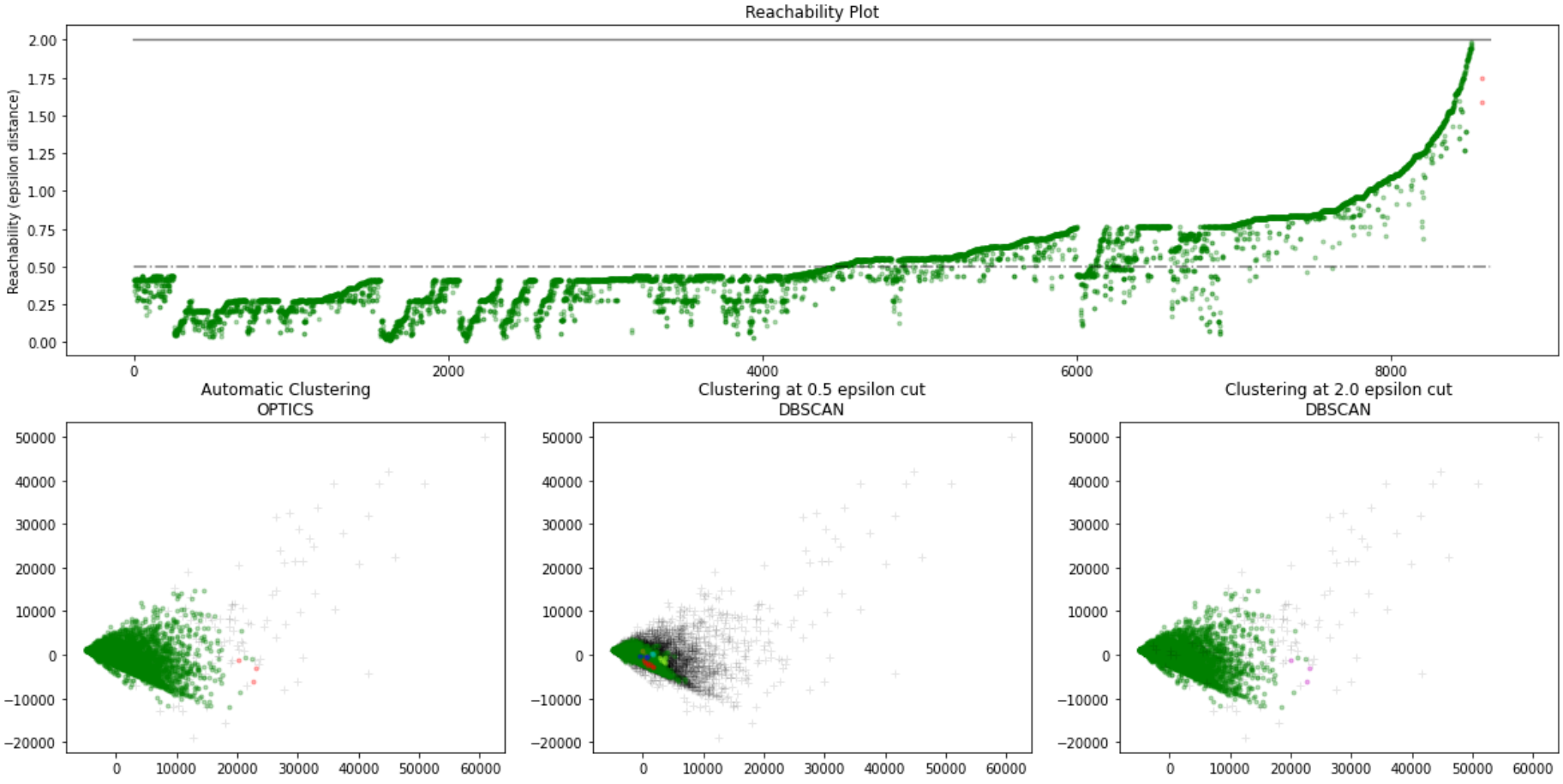
```
clusters: 6, not classified: 0.06
```



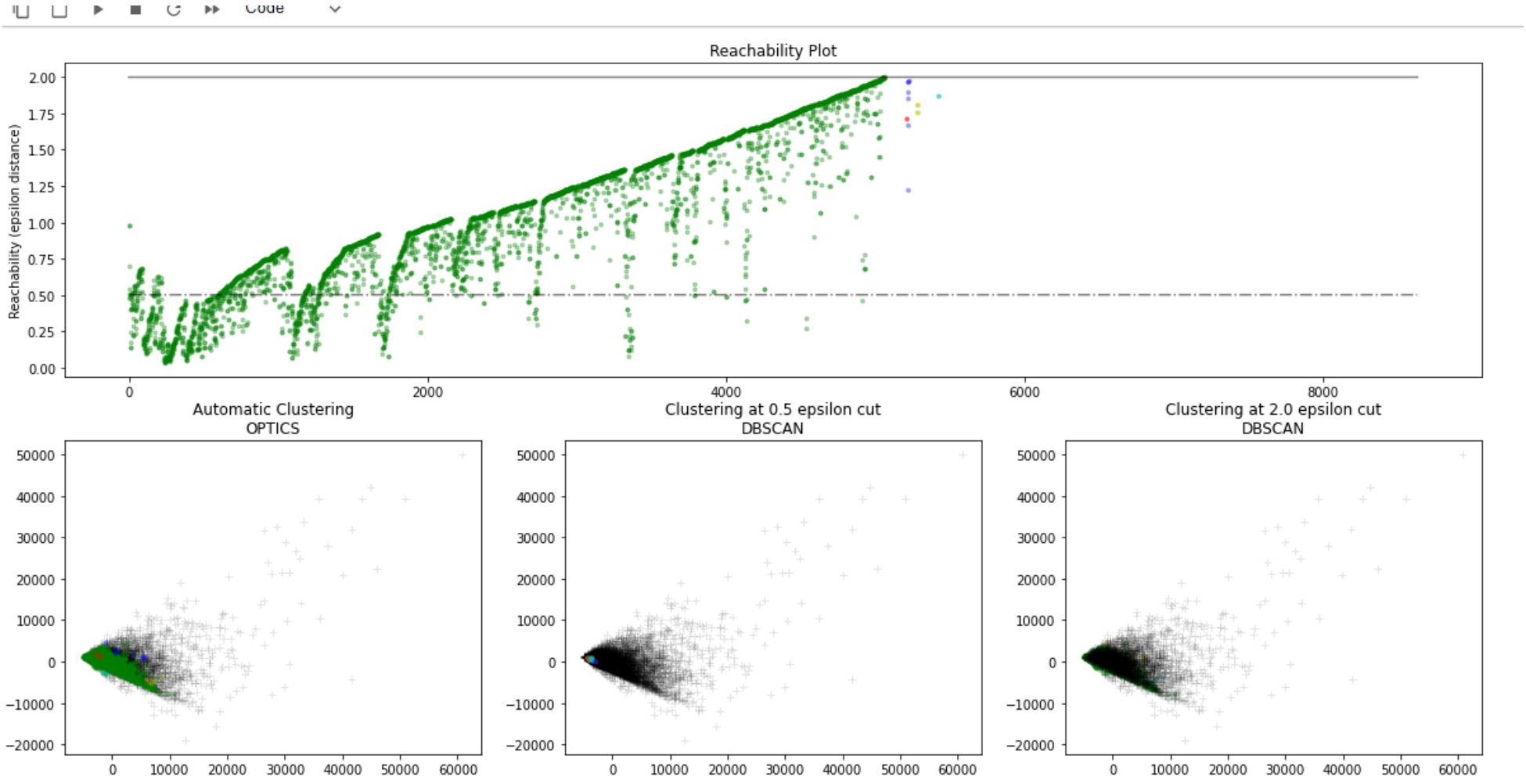
2.


Reachability Plot
Reachability (epsilon distance)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
Automatic Clustering
OPTICS
Clustering at 0.5 epsilon cut
DBSCAN
Clustering at 2.0 epsilon cut
DBSCAN
50000
40000
30000
20000
10000
0
−10000
−20000
0
10000
20000
30000
40000
50000
60000

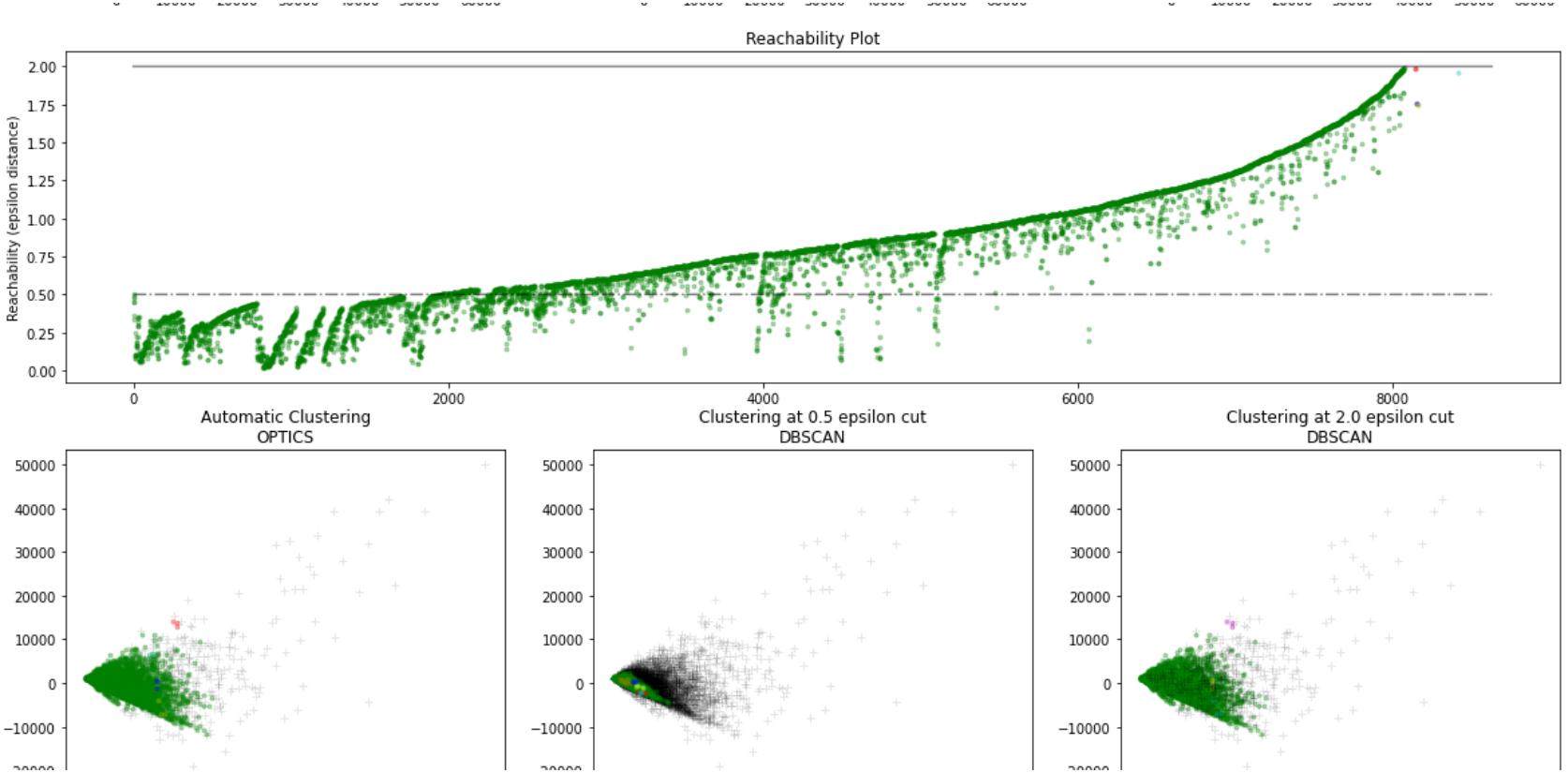
3.


Reachability Plot
Reachability (epsilon distance)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
Automatic Clustering
OPTICS
Clustering at 0.5 epsilon cut
DBSCAN
Clustering at 2.0 epsilon cut
DBSCAN
50000
40000
30000
20000
10000
0
−10000
−20000
0
10000
20000
30000
40000
50000
60000

4.


Code
Reachability Plot
Reachability (epsilon distance)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
Automatic Clustering
OPTICS
Clustering at 0.5 epsilon cut
DBSCAN
Clustering at 2.0 epsilon cut
DBSCAN
50000
40000
30000
20000
10000
0
−10000
−20000
0
10000
20000
30000
40000
50000
60000

5.


Reachability Plot
Reachability (epsilon distance)
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0
2000
4000
6000
8000
Automatic Clustering
OPTICS
Clustering at 0.5 epsilon cut
DBSCAN
Clustering at 2.0 epsilon cut
DBSCAN
50000
40000
30000
20000
10000
0
−10000