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)

Студенты гр. 6304 Григорьев И.С.

Преподаватель Жангиров Т.Р.

Санкт-Петербург

2020

Цель работы

Ознакомиться с методами кластеризации модуля *Sklearn*.

Ход работы

Загрузка данных

Датасет загружен в датафрейм. Вид данных представлен на рис. 1.

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

8636 rows × 17 columns

Рисунок 1 – Исходные данные

DBSCAN

1. Данные стандартизированы, т.к. лежат в разных шкалах.
2. Проведена кластеризация методом DBSCAN при параметрах по умолчанию. Выведены метки кластеров, количество кластеров, а также процент наблюдений, которые кластеризовать не удалось, что показано на рис. 2. В табл. 1 представлены все параметры, которые принимает DBSCAN.

```
clustering = DBSCAN().fit(scaled_data)
print('Метки кластеров', set(clustering.labels_))
print('Количество кластеров', len(set(clustering.labels_)) - 1)
print('Процент выпавших наблюдений', list(clustering.labels_).count(-1) / len(list(clustering.labels_
)))
```

```
Метки кластеров {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 2
3, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, -1}
Количество кластеров 36
Процент выпавших наблюдений 0.7512737378415933
```

Рисунок 2 – Кластеризация DBSCAN при параметрах по умолчанию

Таблица 1 – Параметры DBSCAN

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.
algorithm: {'auto', 'ball_tree', 'kd_tree', 'brute'}, default = 'auto'	Алгоритм, который будет использоваться для вычисления точечных расстояний и поиска ближайших соседей.

3. Построены график количества кластеров и процента не кластеризованных наблюдений в зависимости от максимальной рассматриваемой дистанции (минимальное значение точек, образующих кластер, оставлено по умолчанию) и график количества кластеров и процента не кластеризованных наблюдений в зависимости от минимального значения количества точек, образующих кластер (максимальная рассматриваемая дистанция между наблюдениями оставлена по умолчанию). Графики представлены на рис. 3.

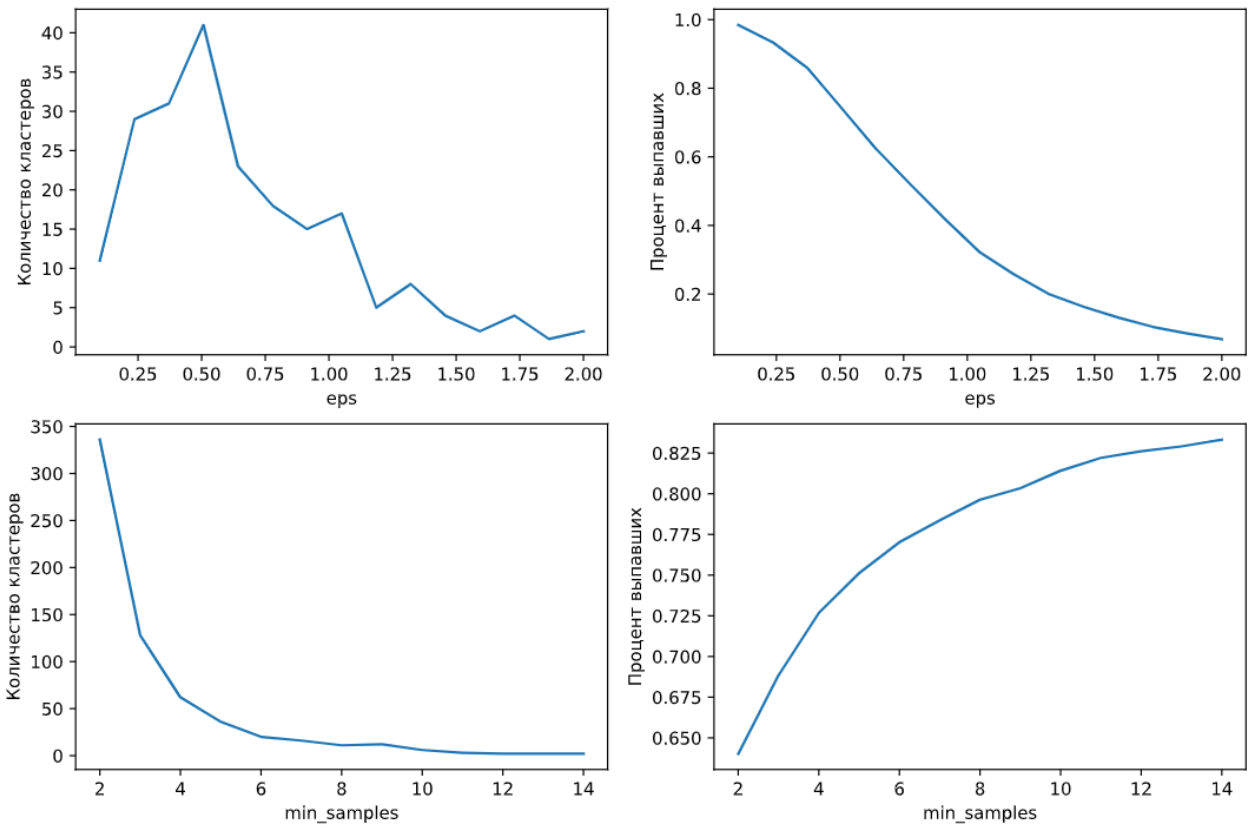

Рисунок 3 – Зависимости количества кластеров и процента выпавших наблюдений от eps и min_samples

С увеличением eps выпавших наблюдений становится все меньше, в конце концов все точки будут находится в одном кластере. Увеличение min_samples приводит к увеличению количества выпавших наблюдений и к уменьшению количества кластеров.

4. Определены значения параметров, при которых количество кластеров получается от 5 до 7, и процент не кластеризованных наблюдений не превышает 12%. Результат приведен на рис. 4.

```
clustering = DBSCAN(eps=2, min_samples=3).fit(scaled_data)
print('Метки кластеров', set(clustering.labels_))
print('Количество кластеров', len(set(clustering.labels_)) - 1)
print('Процент выпавших наблюдений', list(clustering.labels_).count(-1) / len(list(clustering.labels_
)))
```

```
Метки кластеров {0, 1, 2, 3, 4, 5, -1}
Количество кластеров 6
Процент выпавших наблюдений 0.06287633163501621
```

Рисунок 4 – Кластеризация DBSCAN

5. Размерность данных понижена до 2 с помощью метода главных компонент. Результаты кластеризации данных пониженной размерности представлены на рис. 5.

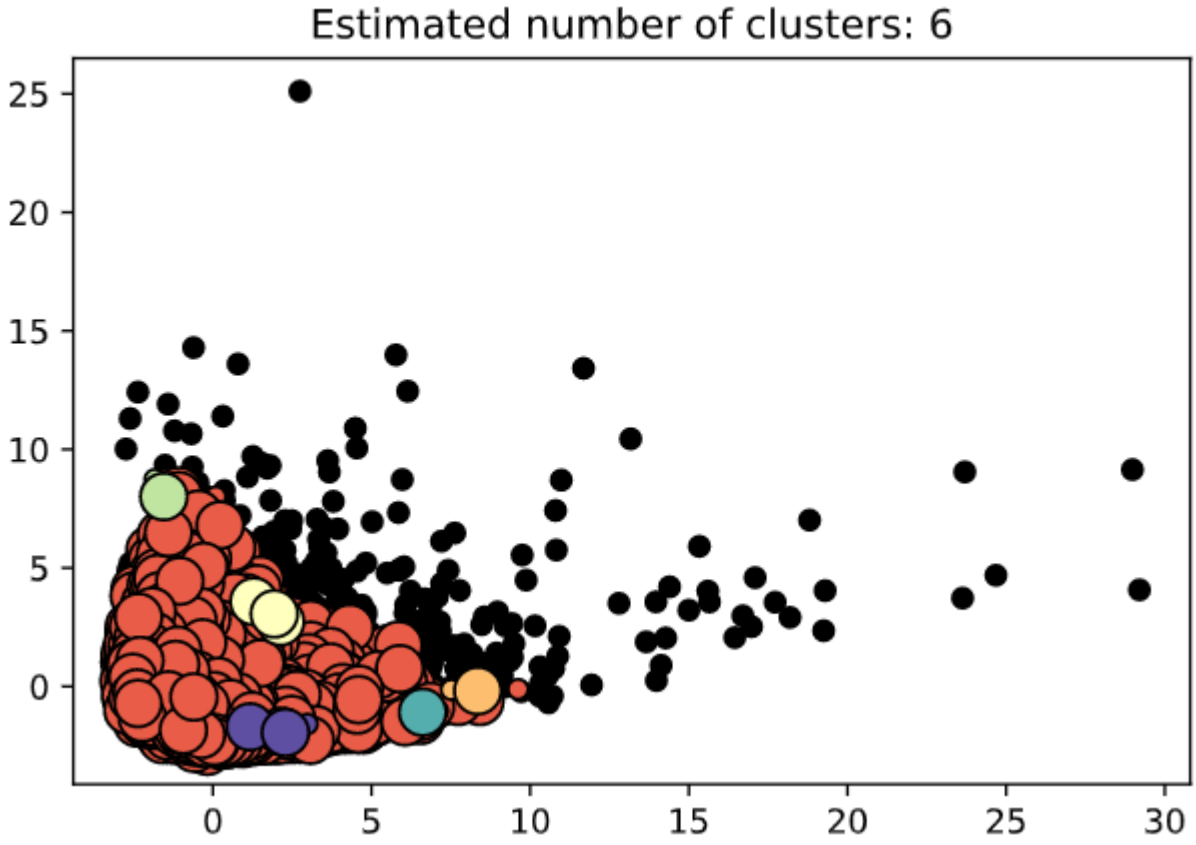

Рисунок 5 – Кластеризация данных пониженной размерности

OPTICS

1. В табл. 1 представлены все параметры, которые принимает DBSCAN.

Таблица 2 – Параметры DBSCAN

Параметр	Описание
max_eps: float, default=∞	Максимальное расстояние между двумя наблюдениями, чтобы один считался 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.

2. Результаты кластеризации методом OPTICS близки к результатам DBSCAN при cluster_method=dbscan, max_eps=2, min_samples=3. Результат представлен на рис. 6.

```
clust = OPTICS(min_samples=3, max_eps=2, cluster_method='dbscan').fit(scaled_data)
print('Метки кластеров', set(clust.labels_))
print('Количество кластеров', len(set(clust.labels_)) - 1)
print('Процент выпавших наблюдений', list(clust.labels_).count(-1) / len(list(clust.labels_)))
```

```
Метки кластеров {0, 1, 2, 3, 4, 5, -1}
Количество кластеров 6
Процент выпавших наблюдений 0.06310792033348772
```

Рисунок 6 – Результат кластеризации OPTICS

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.

3. График достижимости представлен на рис. 7.

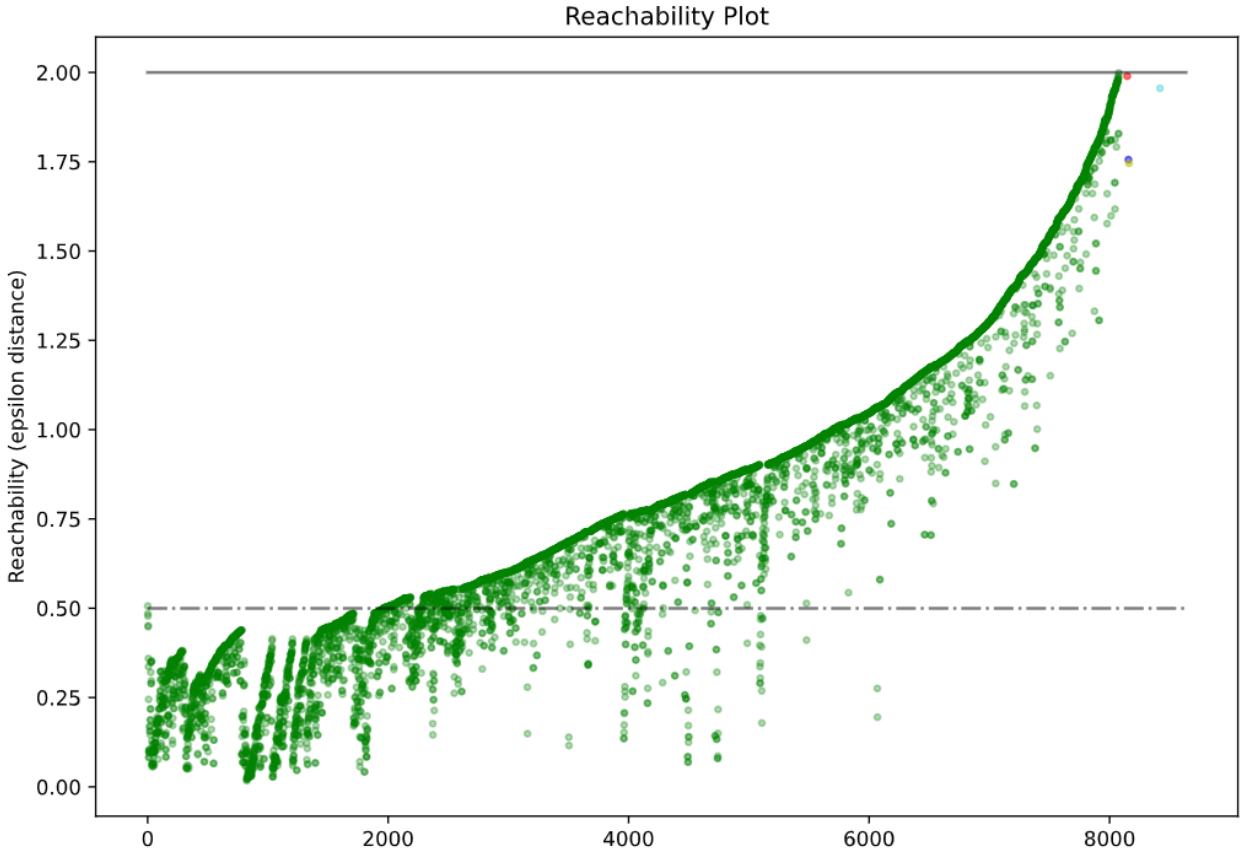

Рисунок 7 – График достижимости

4. Метод OPTICS исследован с использованием различных метрик, результаты представлены в табл. 3.

Таблица 3 – Результаты OPTICS для различных метрик

Метрика	Количество кластеров	Процент выпавших наблюдений
canberra	60	0.464
chebyshev	2	0.013
manhattan	55	0.395
euclidean	6	0.063
sqeuclidean	25	0.152

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.

Приложение А

Код программы на python

```
# To add a new cell, type '# %%'
# To add a new markdown cell, type '# %% [markdown]'
# %%
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans, DBSCAN, OPTICS
from sklearn import preprocessing
from sklearn.decomposition import PCA

# %%
data = pd.read_csv('CC_GENERAL.csv').iloc[:,1:].dropna()
data

# %%
k_means = KMeans(init='k-means++', n_clusters=3, n_init=15)
k_means.fit(data)
print(len(k_means.labels_[k_means.labels_ == 0]))
print(len(k_means.labels_[k_means.labels_ == 1]))
print(len(k_means.labels_[k_means.labels_ == 2]))

# %%
data = np.array(data, dtype='float')
min_max_scaler = preprocessing.StandardScaler()
scaled_data = min_max_scaler.fit_transform(data)

# %%
clustering = DBSCAN().fit(scaled_data)
print('Метки кластеров', set(clustering.labels_))
print('Количество кластеров', len(set(clustering.labels_)) - 1)
print('Процент выпавших наблюдений', list(clustering.labels_).count(-
1) / len(list(clustering.labels_)))

# %%
noise_eps, clusters_nums_eps = [], []
eps_v = np.linspace(0.1, 2, 15)

for eps in eps_v:
    dbscan = DBSCAN(eps=eps).fit(scaled_data)
    clusters_nums_eps.append(len(set(dbscan.labels_)) - 1)
    noise_eps.append(list(dbscan.labels_).count(-1) /  len(list(dbscan.labels_)))

noise_ms, clusters_nums_ms = [], []
min_samples_v = np.arange(2, 15, 1)
for min_samples in min_samples_v:
    dbscan = DBSCAN(min_samples=min_samples).fit(scaled_data)
    clusters_nums_ms.append(len(set(dbscan.labels_)) - 1)
    noise_ms.append(list(dbscan.labels_).count(-1) /  len(list(dbscan.labels_)))

# %%
fig, axes = plt.subplots(2, 2, figsize=(12, 8))

axes[0][0].plot(eps_v, clusters_nums_eps)
axes[0][0].set_xlabel('eps')
axes[0][0].set_ylabel('Количество кластеров')

axes[0][1].plot(eps_v, noise_eps)
axes[0][1].set_xlabel('eps')
```

```
axes[0][1].set_ylabel('Процент выпавших')

axes[1][0].plot(min_samples_v, clusters_nums_ms)
axes[1][0].set_xlabel('min_samples')
axes[1][0].set_ylabel('Количество кластеров')

axes[1][1].plot(min_samples_v, noise_ms)
axes[1][1].set_xlabel('min_samples')
axes[1][1].set_ylabel('Процент выпавших')

# %%
clustering = DBSCAN(eps=2, min_samples=3).fit(scaled_data)
print('Метки кластеров', set(clustering.labels_))
print('Количество кластеров', len(set(clustering.labels_)) - 1)
print('Процент выпавших наблюдений', list(clustering.labels_).count(-
1) / len(list(clustering.labels_)))

# %%
db = DBSCAN(eps=2, min_samples=3).fit(scaled_data)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

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))]

pca = PCA(n_components=2)
reduced_data = pca.fit_transform(scaled_data)
unique_labels.remove(-1)
unique_labels = [-1, *list(unique_labels)]

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=6, alpha=1)

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

plt.title('Estimated number of clusters: %d' % n_clusters_)

# %%
clust = OPTICS(min_samples=3, max_eps=2, cluster_method='dbscan').fit(scaled_data)
print('Метки кластеров', set(clust.labels_))
print('Количество кластеров', len(set(clust.labels_)) - 1)
print('Процент выпавших наблюдений', list(clust.labels_).count(-
1) / len(list(clust.labels_)))

# %%
space = np.arange(len(scaled_data))
reachability = clust.reachability_[clust.ordering_]
```

```
labels = clust.labels_[clust.ordering_]

plt.figure(figsize=(10, 7))

# Reachability plot
colors = ['g.', 'r.', 'b.', 'y.', 'c.']
for klass, color in zip(range(0, 5), 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., dtype=float), 'k-', alpha=0.5)
plt.plot(space, np.full_like(space, 0.5, dtype=float), 'k-.', alpha=0.5)
plt.ylabel('Reachability (epsilon distance)')
plt.title('Reachability Plot')

# %%
metrics = ['canberra', 'chebyshev', 'manhattan', 'sqeuclidean', 'euclidean']
for metric in metrics:
    clust = OPTICS(min_samples=3, max_eps=2, n_jobs=-
1, cluster_method="dbscan", metric=metric).fit(scaled_data)
    print('Количество кластеров', len(set(clust.labels_)) - 1)
    print('Процент выпавших наблюдений', list(clust.labels_).count(-
1) / len(list(clust.labels_)))
```