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

ОТЧЕТ

по лабораторной работе №6

по дисциплине «Машинное обучение»

Тема: Кластеризация (DBSCAN, OPTICS)

Студент гр. 8304 _____ Кирьянов Д.И.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2021

Цель работы.

Ознакомится с методами кластеризации модуля Sklearn.

Ход работы.

Загрузка данных

Был загружен исходный набор данных, убрав столбец с метками и откинув наблюдения с пропущенными значениями. Результат работы представлен на рисунке 1.

	BALANCE	BALANCE_FREQUENCY	...	PRC_FULL_PAYMENT	TENURE
0	40.900749	0.818182	...	0.000000	12
1	3202.467416	0.909091	...	0.222222	12
2	2495.148862	1.000000	...	0.000000	12
4	817.714335	1.000000	...	0.000000	12
5	1809.828751	1.000000	...	0.000000	12
...	...	...	...	...	...
8943	5.871712	0.500000	...	0.000000	6
8945	28.493517	1.000000	...	0.500000	6
8947	23.398673	0.833333	...	0.250000	6
8948	13.457564	0.833333	...	0.250000	6
8949	372.708075	0.666667	...	0.000000	6

Рисунок 1 – Загруженные данные.

DBSCAN

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

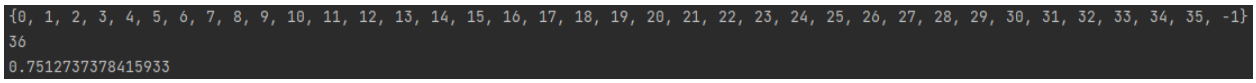

Рисунок 2 – Информация о кластерах

Параметры DBSCAN:

eps – максимальное расстояние между двумя элементами.

min_samples – число элементов в окрестности точки, чтобы считать ее основной.

metric – метрика для расчета расстояния между элементами.

metric_params – дополнительные параметры для метрики.

algorithm – алгоритм, который будет использоваться для вычисления точечных расстояний и поиска ближайших соседей.

leaf_size – может повлиять на скорость конструкции и запроса, а также на, требуемую для построения дерева, память.

p – степень метрики Миньковского, которая будет использоваться для вычисления расстояния между точками.

n_jobs – количество процессов для распараллеливания.

```
Parameters
----------
eps : float, default=0.5
    The maximum distance between two samples for one to be considered
    as in the neighborhood of the other. This is not a maximum bound
    on the distances of points within a cluster. This is the most
    important DBSCAN parameter to choose appropriately for your data set
    and distance function.

min_samples : int, default=5
    The number of samples (or total weight) in a neighborhood for a point
    to be considered as a core point. This includes the point itself.

metric : str, or callable, default='euclidean'
    The metric to use when calculating distance between instances in a
    feature array. If metric is a string or callable, it must be one of
    the options allowed by :func:`sklearn.metrics.pairwise_distances` for
    its metric parameter.
    If metric is "precomputed", X is assumed to be a distance matrix and
    must be square. X may be a :term:`Glossary <sparse graph>`, in which
    case only "nonzero" elements may be considered neighbors for DBSCAN.
```

```
    .. versionadded:: 0.17
       metric *precomputed* to accept precomputed sparse matrix.

metric_params : dict, default=None
    Additional keyword arguments for the metric function.

    .. versionadded:: 0.19

algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'
    The algorithm to be used by the NearestNeighbors module
    to compute pointwise distances and find nearest neighbors.
    See NearestNeighbors module documentation for details.

leaf_size : int, default=30
    Leaf size passed to BallTree or cKDTree. This can affect the speed
    of the construction and query, as well as the memory required
    to store the tree. The optimal value depends
    on the nature of the problem.

p : float, default=None
    The power of the Minkowski metric to be used to calculate distance
    between points. If None, then ``p=2`` (equivalent to the Euclidean
    distance).

n_jobs : int, default=None
    The number of parallel jobs to run.
    ``None`` means 1 unless in a :obj:`joblib.parallel_backend` context.
    ``-1`` means using all processors. See :term:`Glossary <n_jobs>`
    for more details.
```

Построен график количества кластеров и процента не кластеризованных наблюдений в зависимости от максимальной рассматриваемой дистанции между наблюдениями.

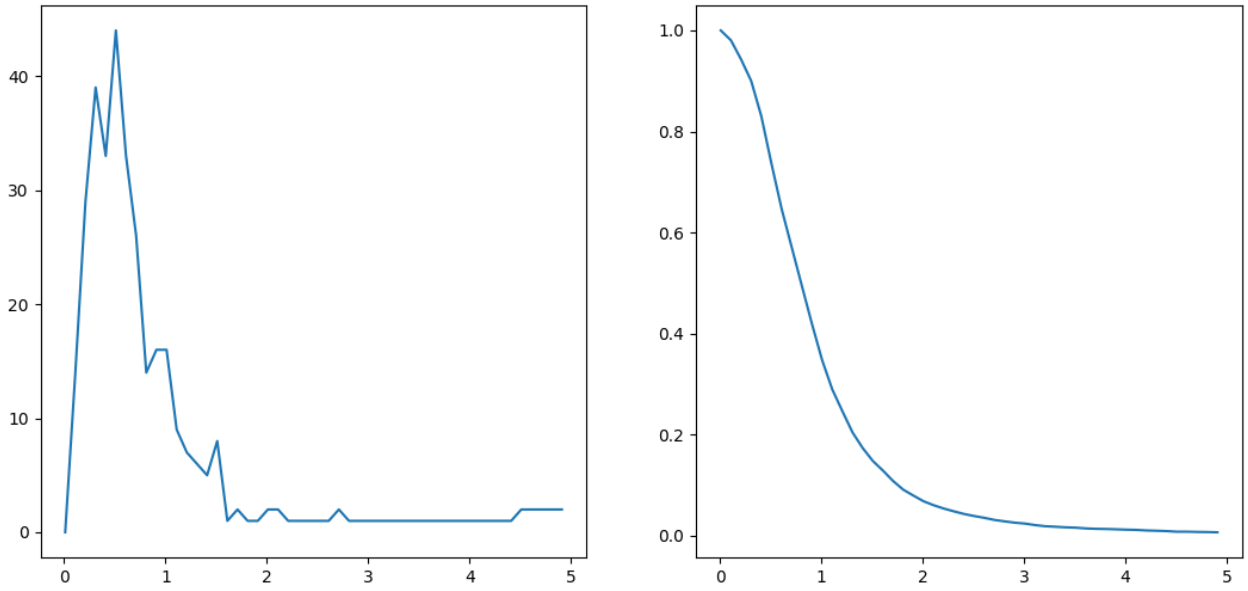

Рисунок 3 – График зависимости кол-ва кластеров и процента не кластеризованных наблюдений от макс. дистанции.

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

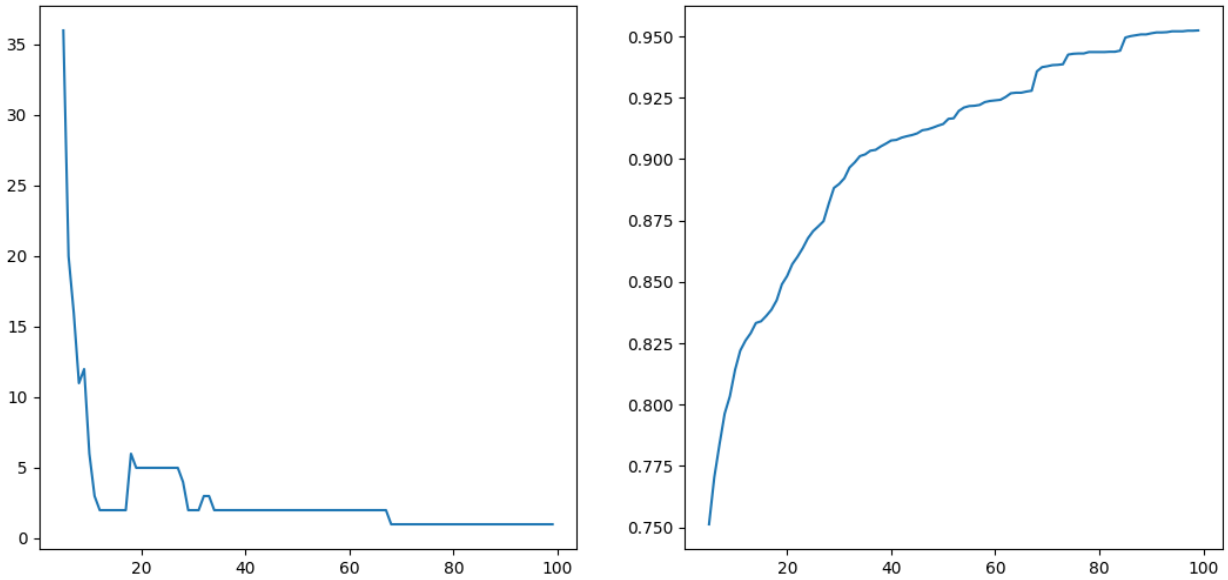

Рисунок 4 – График зависимости кол-ва кластеров и процента не класт. данных от мин. значения кол-ва точек, образующих кластер.

Определены значения минимального кол-ва точек и максимального расстояния при которых кол-во кластеров равно 6, а процент не кластеризованных наблюдений не превышает 12. Полученные величины: eps = 2, min_samples = 3.

Понижена размерность данных до 2, используя метод главных компонент. Визуализированы результаты кластеризации, полученные в пункте 6.

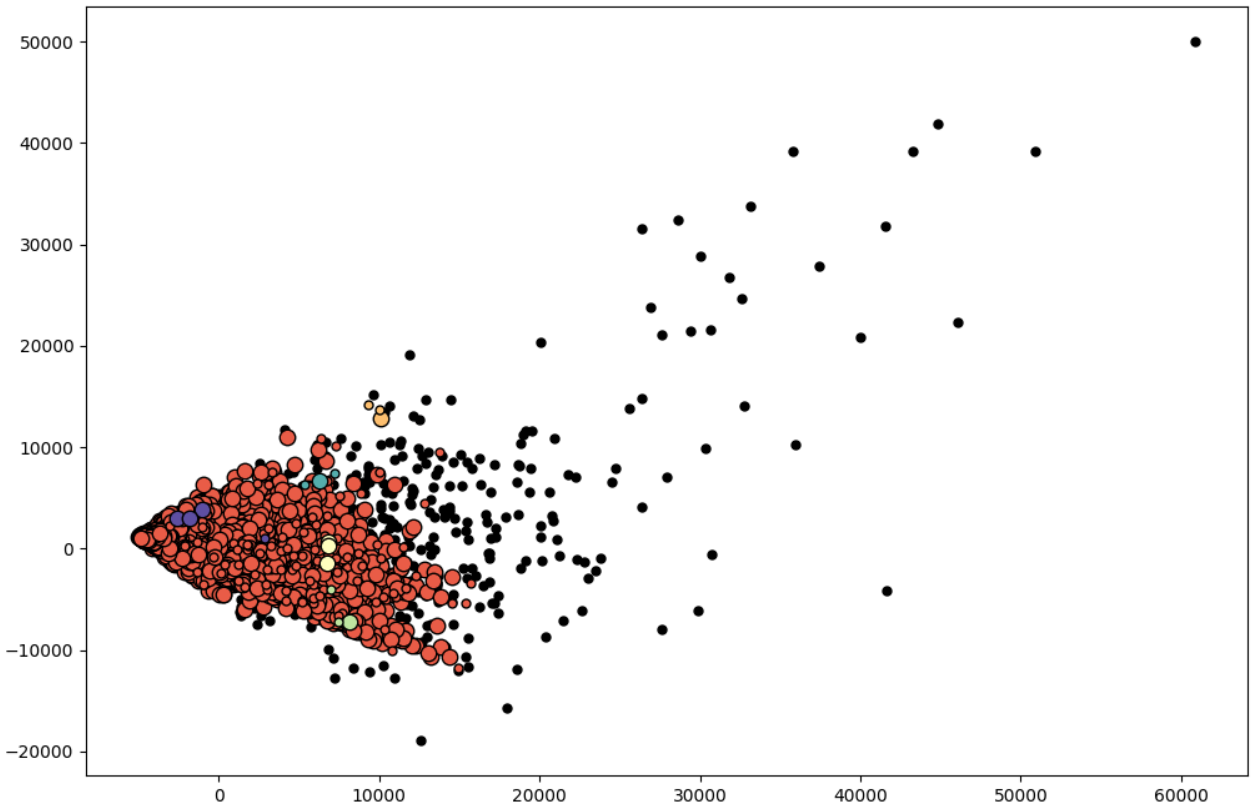

Рисунок 5 – Визуализация результатов кластеризации

OPTICS

Параметры OPTICS:

min_samples – число элементов в окрестности точки, чтобы она считалась основной.

max_eps – максимальное расстояние между элементами, чтобы они считались соседними.

metric – метрика для расчета расстояния между элементами.

p – параметр для метрики Миньковского.

metric_params – дополнительные параметры для метрики.

cluster_method – метод извлечения кластеров

eps – расстояние между элементами, чтобы они считались соседними. По умолчанию соответствует max_eps.

xi – определяет минимальную крутизну графика достижимости, которая составляет границу кластера.

predecessor_correction – коррекция кластеров в соответствии с предшественниками.

min_cluster_size – минимальное количество элементов в кластере OPTICS.

algorithm – алгоритм для поиска ближайших соседей.

leaf_size – может повлиять на скорость конструкции и запроса, а также на, требуемую для построения дерева, память.

memory – используется для кеширования вывода вычисления дерева.

n_jobs – количество параллельных процессов для поиска соседей.

```
Parameters
----------
min_samples : int > 1 or float between 0 and 1, default=5
    The number of samples in a neighborhood for a point to be considered as
    a core point. Also, up and down steep regions can't have more than
    ``min samples`` consecutive non-steep points. Expressed as an absolute
    number or a fraction of the number of samples (rounded to be at least
    2).

max_eps : float, default=np.inf
    The maximum distance between two samples for one to be considered as
    in the neighborhood of the other. Default value of ``np.inf`` will
    identify clusters across all scales; reducing ``max_eps`` will result
    in shorter run times.
```

```
metric : str or callable, default='minkowski'
    Metric to use for distance computation. Any metric from scikit-learn
    or scipy.spatial.distance can be used.

    If metric is a callable function, it is called on each
    pair of instances (rows) and the resulting value recorded. The callable
    should take two arrays as input and return one value indicating the
    distance between them. This works for Scipy's metrics, but is less
    efficient than passing the metric name as a string. If metric is
    "precomputed", X is assumed to be a distance matrix and must be square.

    Valid values for metric are:

    - from scikit-learn: ['cityblock', 'cosine', 'euclidean', 'l1', 'l2',
      'manhattan']

    - from scipy.spatial.distance: ['braycurtis', 'canberra', 'chebyshev',
      'correlation', 'dice', 'hamming', 'jaccard', 'kulsinski',
      'mahalanobis', 'minkowski', 'rogerstanimoto', 'russellrao',
      'seuclidean', 'sokalmichener', 'sokalsneath', 'sqeuclidean',
      'yule']

    See the documentation for scipy.spatial.distance for details on these
    metrics.

p : int, default=2
    Parameter for the Minkowski metric from
    :class:`~sklearn.metrics.pairwise_distances`. When p = 1, this is
    equivalent to using manhattan_distance (l1), and euclidean_distance
    (l2) for p = 2. For arbitrary p, minkowski_distance (l_p) is used.

metric_params : dict, default=None
    Additional keyword arguments for the metric function.

cluster_method : str, default='xi'
    The extraction method used to extract clusters using the calculated
    reachability and ordering. Possible values are "xi" and "dbscan".

eps : float, default=None
    The maximum distance between two samples for one to be considered as
    in the neighborhood of the other. By default it assumes the same value
    as ``max_eps``.
    Used only when ``cluster_method='dbscan'``.

xi : float between 0 and 1, default=0.05
    Determines the minimum steepness on the reachability plot that
    constitutes a cluster boundary. For example, an upwards point in the
    reachability plot is defined by the ratio from one point to its
    successor being at most 1-xi.
    Used only when ``cluster method='xi'``.

predecessor_correction : bool, default=True
    Correct clusters according to the predecessors calculated by OPTICS
    [2]_. This parameter has minimal effect on most datasets.
    Used only when ``cluster method='xi'``.

min_cluster_size : int > 1 or float between 0 and 1, default=None
    Minimum number of samples in an OPTICS cluster, expressed as an
    absolute number or a fraction of the number of samples (rounded to be
    at least 2). If ``None``, the value of ``min_samples`` is used instead.
    Used only when ``cluster_method='xi'``.
```

```
algorithm : {'auto', 'ball_tree', 'kd_tree', 'brute'}, default='auto'
    Algorithm used to compute the nearest neighbors:

    - 'ball_tree' will use :class:`BallTree`
    - 'kd_tree' will use :class:`KDTree`
    - 'brute' will use a brute-force search.
    - 'auto' will attempt to decide the most appropriate algorithm
      based on the values passed to :meth:`fit` method. (default)

    Note: fitting on sparse input will override the setting of
    this parameter, using brute force.

leaf_size : int, default=30
    Leaf size passed to :class:`BallTree` or :class:`KDTree`. This can
    affect the speed of the construction and query, as well as the memory
    required to store the tree. The optimal value depends on the
    nature of the problem.

memory : str or object with the joblib.Memory interface, default=None
    Used to cache the output of the computation of the tree.
    By default, no caching is done. If a string is given, it is the
    path to the caching directory.

n_jobs : int, default=None
    The number of parallel jobs to run for neighbors search.
    ``None`` means 1 unless in a :obj:`joblib.parallel_backend` context.
    ``-1`` means using all processors. See :term:`Glossary <n_jobs>`
    for more details.
```

Атрибуты OPTICS:

labels_ - метки кластера для каждой точки в датасете, заданной для fit().

reachability_ - расстояния достижимости для каждой выборки, индексированные по порядку объектов.

ordering_ - упорядоченный список выборочных индексов кластера.

core_distances_ - расстояние, на котором каждая выборка становится центральной точкой, индексируется по порядку объектов.

predecessor_ - точка, из которой была получена выборка, индексируется по порядку объектов.

cluster_hierarchy_ - список кластеров вида [начало, конец] в каждой строке, включая все индексы.

n_features_in_ - количество деталей, видимых во время посадки.

feature_names_in_ - названия особенностей, замеченных во время посадки. Определяется только, если X имеет имена функций, являющихся строками.

```
Attributes
----------
labels_ : ndarray of shape (n_samples,)
    Cluster labels for each point in the dataset given to fit().
    Noisy samples and points which are not included in a leaf cluster
    of ``cluster hierarchy `` are labeled as -1.

reachability_ : ndarray of shape (n_samples,)
    Reachability distances per sample, indexed by object order. Use
    ``clust.reachability_[clust.ordering_]`` to access in cluster order.

ordering_ : ndarray of shape (n_samples,)
    The cluster ordered list of sample indices.

core_distances_ : ndarray of shape (n_samples,)
    Distance at which each sample becomes a core point, indexed by object
    order. Points which will never be core have a distance of inf. Use
    ``clust.core_distances_[clust.ordering_]`` to access in cluster order.

predecessor_ : ndarray of shape (n samples,)
    Point that a sample was reached from, indexed by object order.
    Seed points have a predecessor of -1.

cluster hierarchy_ : ndarray of shape (n clusters, 2)
    The list of clusters in the form of ``[start, end]`` in each row, with
    all indices inclusive. The clusters are ordered according to
    ``(end, -start)`` (ascending) so that larger clusters encompassing
    smaller clusters come after those smaller ones. Since ``labels_`` does
    not reflect the hierarchy, usually
    ``len(cluster_hierarchy_) > np.unique(optics.labels_)``. Please also
    note that these indices are of the ``ordering_``, i.e.
    ``X[ordering_][start:end + 1]`` form a cluster.
    Only available when ``cluster_method='xi'``.

n_features_in_ : int
    Number of features seen during :term:`fit`.

    .. versionadded:: 0.24

feature_names_in_ : ndarray of shape (`n_features_in_`,)
    Names of features seen during :term:`fit`. Defined only when `X`
    has feature names that are all strings.
```

Найдены такие параметры метода OPTICS, при которых получается результат близкий к результату DBSCAN из пункта 6. Полученные величины: eps = 2, min_samples = 3, cluster_method = dbscan.

Визуализирован полученный результат, а также построен график достижимости.

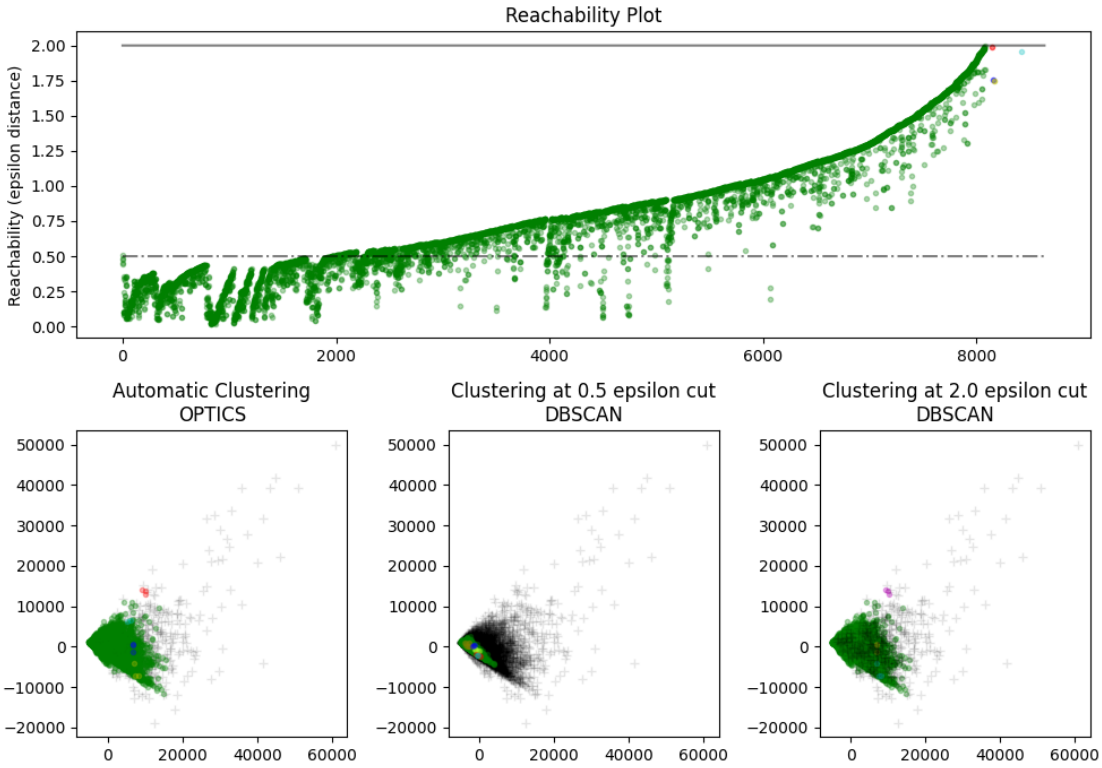

Рисунок 6 – Визуализированный результат.

Исследована работа метода OPTICS с использованием различных метрик (выбрано не менее 5 метрик).

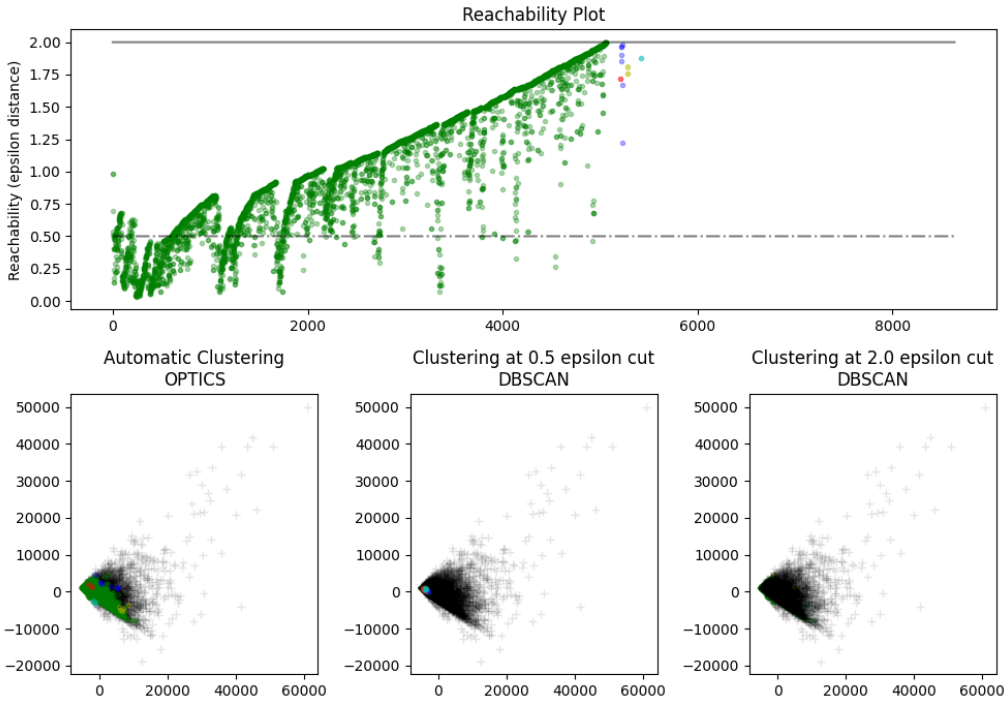

Рисунок 7 – График достижимости при использовании метрики “cityblock”

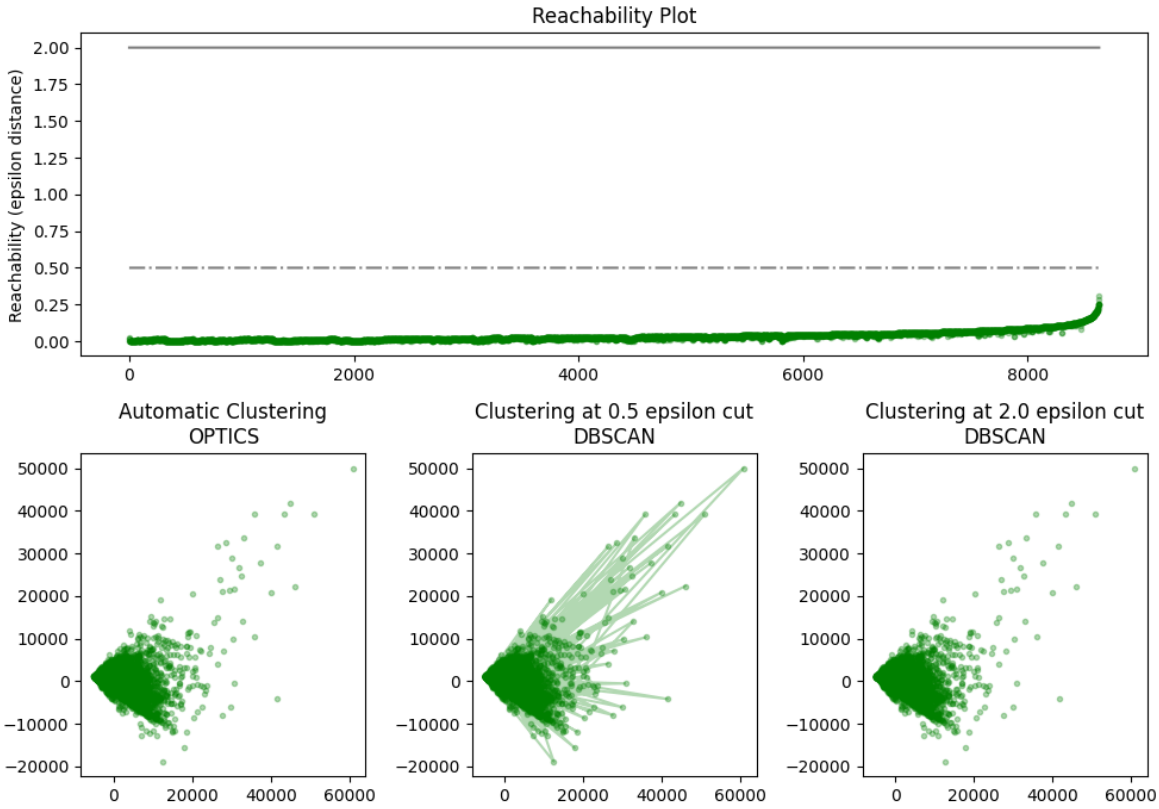

Рисунок 8 – График достижимости при использовании метрики “cosine”

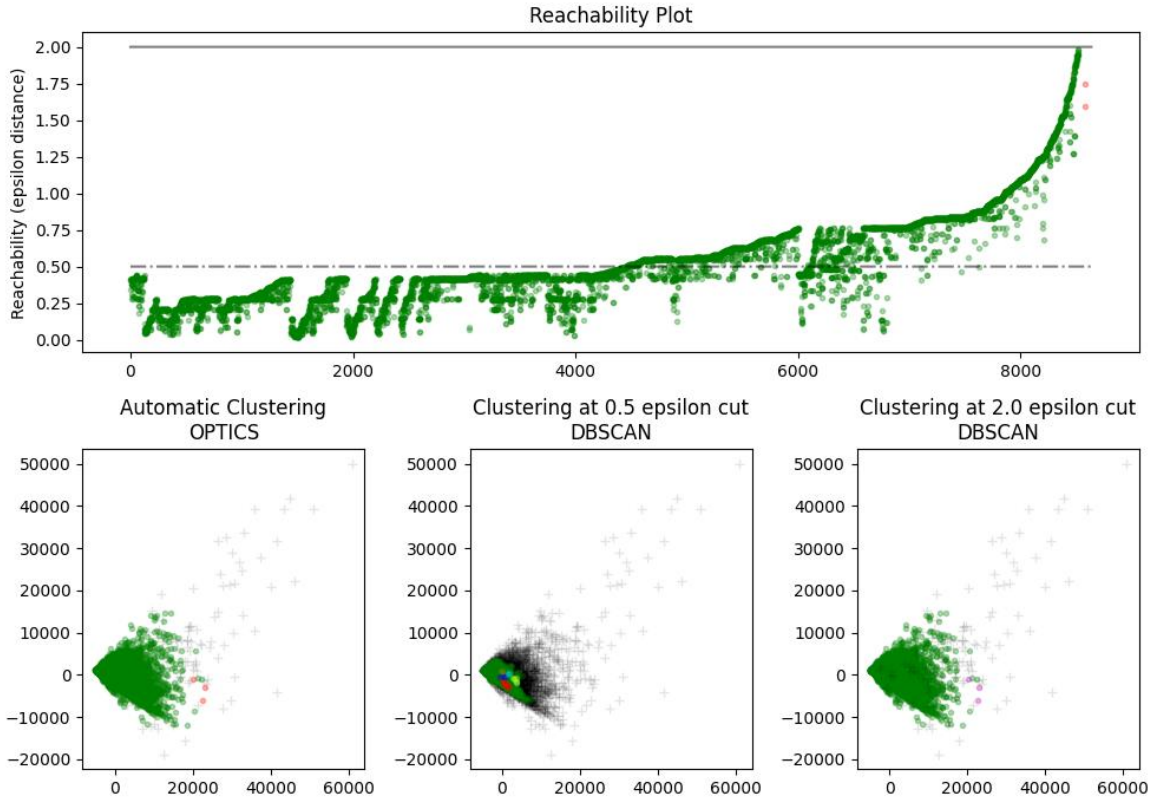

Рисунок 9 – График достижимости при использовании метрики “chebyshev”

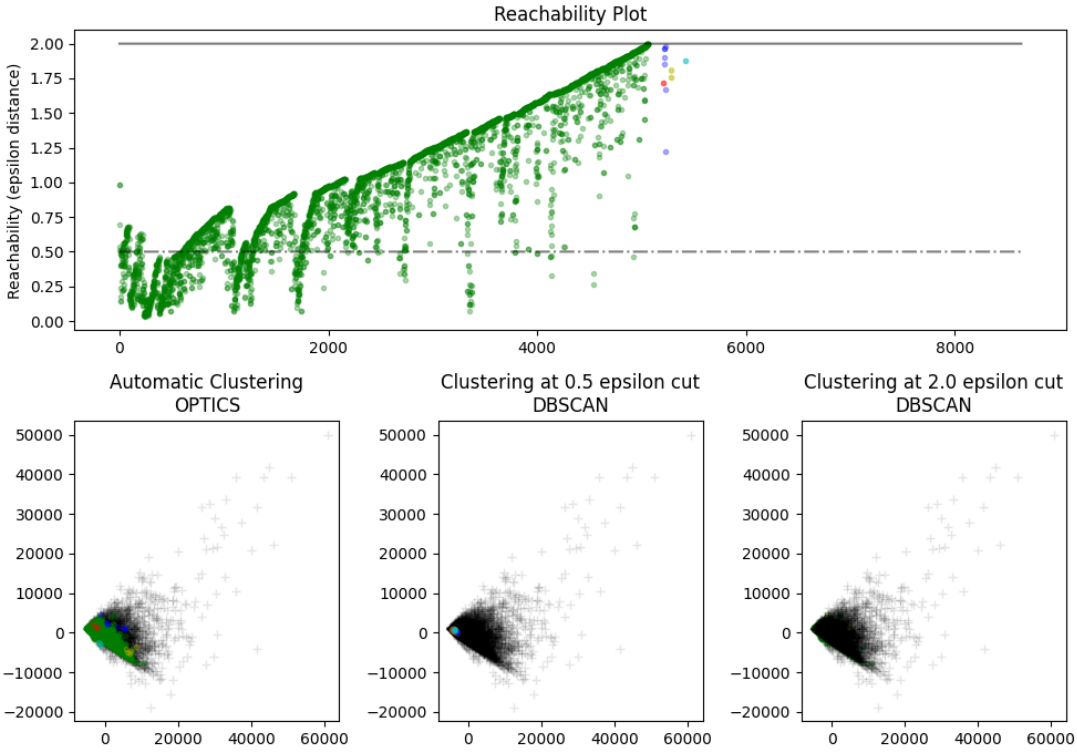

Рисунок 10 – График достижимости при использовании метрики “l1”

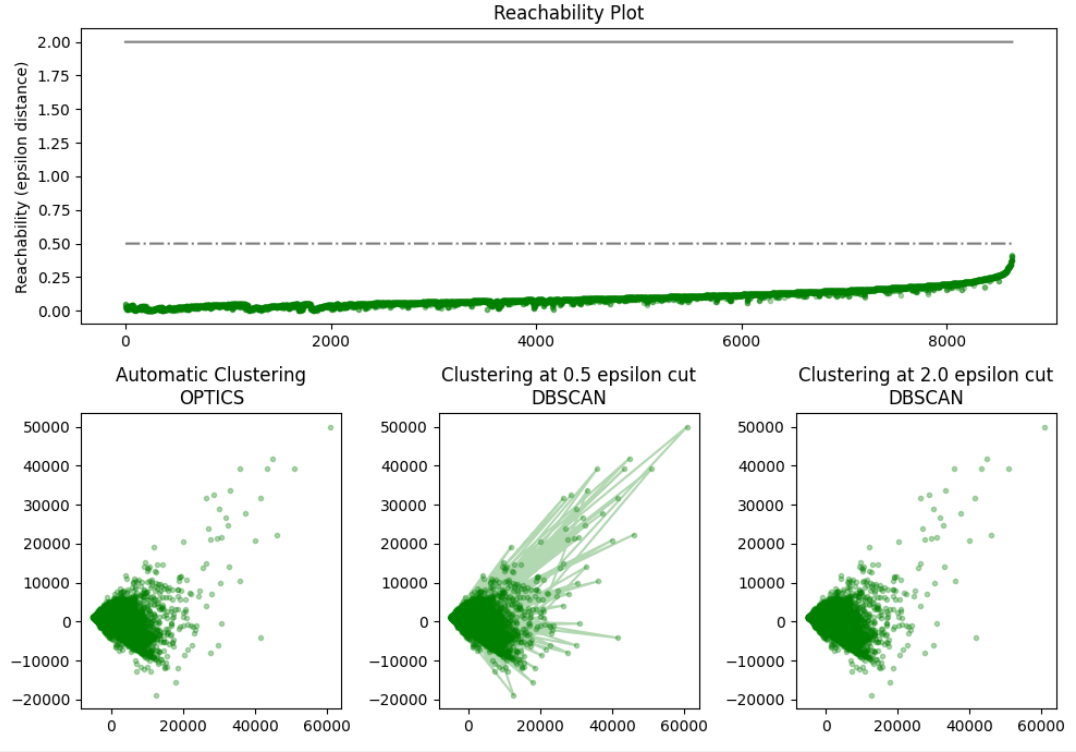

Рисунок 11 – График достижимости при использовании метрики “braycurtis”

Вывод

Был получен опыт работы с методами кластеризации.

ПРИЛОЖЕНИЕ А

Исходный код программы

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans, DBSCAN, OPTICS, cluster_optics_dbscan
from sklearn import preprocessing
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
from sklearn.decomposition import PCA


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

k_means = KMeans(init='k-means++', n_clusters=3, n_init=15)
no_labeled_data = data[1:]
k_means.fit(no_labeled_data)
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(set(clustering.labels_))
print(len(set(clustering.labels_)) - 1)
print(list(clustering.labels_).count(-1) / len(list(clustering.labels_)))

eps_ = np.arange(0.01, 5.0, 0.1)
info = []
for eps in eps_:
    clustering = DBSCAN(eps=eps).fit(scaled_data)
    labels_set = set(clustering.labels_)
    info.append([len(labels_set) - 1, list(clustering.labels_).count(-1) /
len(list(clustering.labels_))])

info = np.array(info)
fig, ax = plt.subplots(1, 2, figsize=(13,6))
ax[0].plot(eps_, info[:,0])
ax[1].plot(eps_, info[:,1])
plt.show()
samples = np.arange(5, 100, 1)
info = []
for sample in samples:
    clustering = DBSCAN(min_samples=sample).fit(scaled_data)
    labels_set = set(clustering.labels_)
    info.append([len(labels_set) - 1, list(clustering.labels_).count(-1) /
len(list(clustering.labels_))])
info = np.array(info)
fig, ax = plt.subplots(1, 2, figsize=(13,6))
ax[0].plot(samples, info[:,0])
ax[1].plot(samples, info[:,1])
plt.show()
samples = np.arange(1, 4, 1)
eps_ = np.arange(1.5, 2.5, 0.1)
info = {}
for sample in samples:
    for eps in eps_:
        clustering = DBSCAN(eps=eps ,min_samples=sample, n_jobs=-
1).fit(scaled_data)
        labels_set = set(clustering.labels_)
        info[(sample, eps)]= [len(labels_set) - 1,
list(clustering.labels_).count(-1) / len(list(clustering.labels_))]
```

```
for key, value in info.items():
    if value[0]>=5 and value[0]<=7 and value[1]<=0.12:
        print(key, value)

clustering = DBSCAN(eps=2, min_samples=3, n_jobs=-1).fit(scaled_data)
pca = PCA(n_components=2)
reduced_data = pca.fit_transform(data)
pca.explained_variance_ratio_
core_samples_mask = np.zeros_like(clustering.labels_, dtype=bool)
core_samples_mask[clustering.core_sample_indices_] = True
labels = clustering.labels_

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
# unique_labels = set(labels)
unique_labels = set(labels)
unique_labels.remove(-1)
unique_labels = [-1, *list(unique_labels)]
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
plt.figure(figsize=(12, 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=9)

    xy = reduced_data[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=5)
plt.show()
clustering = OPTICS(max_eps=2, min_samples=3,
cluster_method='dbscan').fit(scaled_data)
print(set(clustering.labels_))
print(len(set(clustering.labels_)) -1)
print(100 * list(clustering.labels_).count(-1) /
len(list(clustering.labels_)))

pca_data = PCA(n_components=2).fit_transform(data)
def plot_optics(clust):
    labels_050 = cluster_optics_dbscan(reachability=clust.reachability_,
                                       core_distances=clust.core_distances_,
                                       ordering=clust.ordering_, eps=0.5)
    labels_200 = cluster_optics_dbscan(reachability=clust.reachability_,
                                       core_distances=clust.core_distances_,
                                       ordering=clust.ordering_, eps=2)

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

    plt.figure(figsize=(10, 7))
    G = gridspec.GridSpec(2, 3)
    ax1 = plt.subplot(G[0, :])
    ax2 = plt.subplot(G[1, 0])
    ax3 = plt.subplot(G[1, 1])
    ax4 = plt.subplot(G[1, 2])
```

```
    # 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]
        ax1.plot(Xk, Rk, color, alpha=0.3)
    ax1.plot(space[labels == -1], reachability[labels == -1], 'k.',
alpha=0.3)
    ax1.plot(space, np.full_like(space, 2., dtype=float), 'k-', alpha=0.5)
    ax1.plot(space, np.full_like(space, 0.5, dtype=float), 'k-.', alpha=0.5)
    ax1.set_ylabel('Reachability (epsilon distance)')
    ax1.set_title('Reachability Plot')

    # OPTICS
    colors = ['g.', 'r.', 'b.', 'y.', 'c.']
    ax2.plot(pca_data[clust.labels_ == -1, 0], pca_data[clust.labels_ == -1,
1], 'k+', alpha=0.1)
    for klass, color in zip(range(0, 5), colors):
        Xk = pca_data[clust.labels_ == klass]
        ax2.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3)
    ax2.set_title('Automatic Clustering\nOPTICS')

    # DBSCAN at 0.5
    colors = ['g', 'greenyellow', 'olive', 'r', 'b', 'c']
    ax3.plot(pca_data[labels_050 == -1, 0], pca_data[labels_050 == -1, 1],
'k+', alpha=0.1)
    for klass, color in zip(range(0, 6), colors):
        Xk = pca_data[labels_050 == klass]
        ax3.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3, marker='.')
    ax3.set_title('Clustering at 0.5 epsilon cut\nDBSCAN')

    # DBSCAN at 2.
    colors = ['g.', 'm.', 'y.', 'c.']
    for klass, color in zip(range(0, 4), colors):
        Xk = pca_data[labels_200 == klass]
        ax4.plot(Xk[:, 0], Xk[:, 1], color, alpha=0.3)
    ax4.plot(pca_data[labels_200 == -1, 0], pca_data[labels_200 == -1, 1],
'k+', alpha=0.1)
    ax4.set_title('Clustering at 2.0 epsilon cut\nDBSCAN')

    plt.tight_layout()
    plt.show()

plot_optics(clustering)

metrics = ['cityblock', 'cosine', 'chebyshev', 'l1', 'braycurtis']

for metric in metrics:
    clustering = OPTICS(max_eps=2, min_samples=3, cluster_method="dbscan",
metric=metric).fit(scaled_data)
    plot_optics(clustering)
```