МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»

Высшая школа электроники и компьютерных наук

Кафедра вычислительной математики и высокопроизводительных
вычислений

РАБОТА ПРОВЕРЕНА
Рецензент, к. т. н.,
доцент
_____/А.С. Волосников
«____»_____2020 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, к.ф.-м.н.,
доцент
_____/Н.М. Япарова
«____»_____2020 г.

Исследование методов искусственного интеллекта в кластерном анализе большого объёма данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–090401.2020.680.ПЗ ВКР

Руководитель работы, доцент,
к.ф.-м.н., доцент
_____/ Н.М. Япарова
«____»_____2020 г.
Автор работы
Студент группы КЭ-230
_____/ Р.М. Исмагилов
«____»_____2020 г.

Нормоконтролер, доцент, к.ф.-
м.н., доцент
_____/ С.У. Турлакова
«____»_____2020 г.

Челябинск 2020

АННОТАЦИЯ

Исмагилов Р.М. Исследование методов кластерного анализа для большого объема данных Челябинск: ЮУрГУ, КЭ-230, 68 с., 31 ил.10 таб., библиогр. список – 41 наим.

Данная выпускная квалификационная работа посвящена сравнительному анализу методов кластерного анализа данных. В теоретической части был произведен обзор существующих методов кластерного анализа. Рассмотрены возможные модификации этих методов и критерии их выбора. Так же были подробно описаны этапы кластерного анализа. Практическая часть посвящена реализации выбранных методов на языке программирования Python

Основные задачи проекта:

1. Рассмотреть существующие методы кластеризации;
2. Рассмотреть критерии выбора метода кластеризации;
3. Выбрать несколько популярных методов кластеризации;
4. Подготовить данные к кластеризации;
5. Провести кластеризацию данных выбранным методом.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность темы.

Благодаря развитым методам сбора информации увеличиваются объёмы различных баз данных. В связи с этим увеличивается и потребность в группировании этой самой информации для последующего извлечения выгоды. Этим и занимается кластерный анализ. Кластерный анализ – это общее название для большого набора статистических методов, которые направленны на обнаружение групп в выборке объектов, которые называют кластерами. Существенным отличием кластерного анализа от иных методов какой-либо группировки состоит в том, что группы заранее неизвестны.

Кластерный анализ нашел себе применение во многих областях науки. Начиная от маркетинга с задачами по сегментации потребителей [1] заканчивая социологией и даже медициной [2] помогая классифицировать препараты, симптомы и самих пациентов.

Кластерный анализ выполняет следующие задачи:

1. Разработка типологии или классификации;
2. Исследование полезных концептуальных схем группирования объектов;
3. Порождение гипотез на основе исследования данных;
4. Проверка гипотез или исследования для определения, действительно ли типы (группы), выделенные тем или иным способом, присутствуют в имеющихся данных.

Цель данной работы: построение адаптивного метода обработки большого объёма данных на основе сравнительного анализа методов анализа данных, с выявлением оптимального метода обработки.

Для реализации цели работы решаются следующие задачи:

1. Отбор методов кластеризации;
2. Разработка программного обеспечения, реализующего выбранные методы;
3. Формирование критериев оптимальности параметров кластерного анализа;
4. Выбор наиболее оптимального метода обработки.

Подобные исследования неоднократно проводились, [3-6] но работы в качестве данных использовали некие готовые наборы данных (data sets). Уникальность этой работы в том, что сравнение методов кластеризации производится на реальных данных о клиентах различных магазинов, где результат работы не очевиден.

Обзор литературы

В основном темой кластерного анализа занимаются «за рубежом». Поэтому основными источниками являются зарубежные работы. Это статья «Some methods for classification and analysis of multivariate observations» от автора MacQueen J. Данная работа посвящена методу K - means описывая его простоту реализации и возможность применения на больших объёмах данных. Так же, это статья, посвященная методу кластеризации BIRCH от авторов Tian Z, Raghu R, Miron L с названием «BIRCH: an efficient data clustering method for very large databases». Авторы описывают принципы работы нового, на то время, метода кластеризации, приводит способы оптимизации метода для меньшего потребления памяти и примеры работы на реальных данных, в частности изображений дерева. И основной можно назвать публикацию авторов L. McInnes, J. Healy, S. Astels, с названием «hdbscan: Hierarchical density based clustering». Публикация посвящена описанию модификации существующего метода кластеризации DBSCAN и должна решить проблемы с подбором неочевидного параметра «эпсилон». Работа описывает недостатки метода DBSCAN и решает их, добавляя дополнительные шаги к алгоритму кластеризации. В публикации демонстрируется что метод HDBSCAN может лучше справляться с выделением кластеров, различающихся друг от друга плотностями.

Современные работы по теме кластерного анализа больше представлены в виде статей посвященных каким-либо модификациям существующих методов кластеризации. Например, статья о LBIRCH [13] посвященная улучшению метода BIRCH так же к этому можно отнести статью о подборе параметра «эпсилон» для DBSCAN [8] или статью про улучшение DBSCAN для лучшей кластеризации пространственных баз данных.

1. Теоретическая часть

1.1 Описание различных методов кластеризации

Методы кластеризации можно разделить на пять категорий:

- Методы разбиения;
- Иерархические методы;
- Плотностные методы;
- Сеточные методы;
- Модельные методы.

Методы разбиения

Для заданного k (количества кластеров) алгоритм создает начальные группы, затем итерационным методом меняет группы так, чтобы результаты группировки были лучше, чем в предыдущей итерации. Алгоритм стремится к такому результату, при котором объекты в группе будут схожи друг с другом, а объекты из разных групп различаться. Примерами реализации такого подхода являются алгоритмы К – means [7] и К – medoids [8]. В алгоритме К – means сходство рассматривается по отношению к центру масс кластера – среднему значению координат объектов кластера в пространстве данных. К – medoids в свою очередь использует для представления центра кластера не центр масс, а представительный объект – один из объектов кластера. Данные эвристические методы применимы для нахождения сферических кластеров в малых и средних по размеру базах данных. Для кластеризации больших баз или кластеризации данных со сложными формами требуется корректировки методов.

Иерархические методы

Суть метода в создании дерева вложенных кластеров. В соответствии с направлениями декомпозиции алгоритмы разделяют на методы:

- Дивизимные;
- Агломеративные.

Для агломерационной иерархической кластеризации декомпозиция происходит «снизу вверх». Такой алгоритм начинает работу с того, что каждому объекту назначается свой кластер. В последующих итерациях происходит объединение смежных кластеров друг с другом до тех пор, пока все объекты не окажутся в одном кластере или будут выполнены иные условия завершения.

Дивизимная кластеризация в свою очередь реализует декомпозицию «сверху вниз». Первым этапом алгоритма будет объединение всех объектов в один кластер, а в последующих итерациях будет происходить разбиение кластеров на меньшие части до тех пор, пока не будет достигнуто условие завершения.

Недостатком иерархических методов является отсутствие возможности отменить слияния или разделение кластера после завершения итерации. Из-за этого нельзя исправить неправильное решение что может привести к низкому качеству кластеризации. Так же ввиду необходимости проверять и оценивать количество кластеров после каждого слияния или разделения, метод плохо масштабируется. Примерами алгоритмов, основанных на иерархическом методе, являются BIRCH [9] и CHAMELEON [10].

Плотностные методы

Цель данного метода — это обнаружение кластеров произвольной формы. Основная идея метода заключается в условии, до тех пор, пока плотность смежной области (объекта или числа точек данных) превышает определенный порог, продолжать кластеризацию. Для создания кластера необходимо чтобы вокруг некоторой точки на заданном расстоянии должно находится заданное количество других точек. Данный метод хорошо подходит

для фильтрации различных выбросов и как было указанно ранее для нахождения кластеров произвольной длинны.

Представителями такого метода кластеризации являются DBSCAN [11] и OPTICS. Последний является более предпочтительным вариантом т.к. устраняет недостаток DBSCAN заключающийся в трудности кластеризации данных с большими разницами в плотностях.

Сетевые методы

Общая идея методов заключается в том, что пространство объектов разбивается на конечное число ячеек, образующих сетевую структуру, в рамках которой выполняются все операции кластеризации. Главное достоинство методов этой группы в малом времени выполнения, которое обычно не зависит от количества объектов данных, а зависит только от количества ячеек в каждом измерении пространства.

Алгоритм CLIQUE [12], адаптированный под кластеризацию данных высокой размерности, является одним из классических сетевых алгоритмов. Метод основан на том предположении, что если в многомерном пространстве данных распределение объектов не равномерно – встречаются регионы плотности и разрежения, то проекция региона плотности в подпространство с меньшей размерностью будет частью региона плотности в этом подпространстве.

Модельные методы

Методы этого семейства предполагают, что имеется некоторая математическая модель кластера в пространстве данных и стремятся максимизировать сходство этой модели и имеющихся данных. Часто при этом используется аппарат математической статистики. Алгоритм может создавать

функцию плотности, отражающую пространственное распределение точек данных для определения местоположения кластера. Основываясь на стандартной статистике, он автоматически определяет количество кластеров.

Модельные методы разделяются на 2 типа:

- Статистические методы;
- Нейронные сети.

Одним из представителей данного метода является алгоритм EM. Алгоритм EM [13] основан на предположении, что исследуемое множество данных может быть смоделировано с помощью линейной комбинации многомерных нормальных распределений. Его целью является оценка параметров распределения, которые максимизируют функцию правдоподобия, используемую в качестве меры качества модели.

1.2. Этапы выполнения кластерного анализа.

Процесс кластеризации некоторого объёма данных можно представить в виде нескольких этапов:

1. формулировка проблемы;
2. выбор меры расстояния;
3. выбор метода кластеризации;
4. принятие решения о количестве кластеров;
5. интерпретация и профилирование кластеров;
6. оценка достоверности кластеризации.

Формулировка проблемы

Основная задача данного этапа — это отбор данных и их преобразование. Проблема состоит в том, чтобы отыскать именно ту совокупность переменных, которая бы наилучшим образом отражала понятие сходства. В идеале переменные должны выбираться в соответствии с ясно сформулированной теорией лежащей в основе классификации т.к. бездумный отбор по принципу «чем больше, тем лучше» может привести к некорректным результатам.

Выбор меры расстояния

Цель кластеризации — группирование схожих объектов. Поэтому для того, чтобы оценить, насколько они похожи или непохожи, необходимо использовать некую единицу измерения. Наиболее распространенный метод заключается в том, чтобы в качестве такой меры использовать расстояния между двумя объектами. Объекты с меньшими расстояниями между собой

больше похожи, чем объекты с большими расстояниями. Существует несколько способов вычисления расстояния между двумя объектами.

Это:

1. Евклидово расстояние

Наиболее распространенная функция расстояния. Представляет собой геометрическим расстоянием в многомерном пространстве:

$$p(x, x') = \sqrt{\sum_{i}^{n} (x_i - x_i')^2}$$

2. Квадрат евклидова расстояния

Применяется для придания большего веса более отдаленным друг от друга объектам. Это расстояние вычисляется следующим образом:

$$p(x, x') = \sum_{i}^{n} (x_i - x_i')^2$$

3. Расстояние городских кварталов (манхэттенское расстояние)

Это расстояние является средним разностей по координатам. В большинстве случаев эта мера расстояния приводит к таким же результатам, как и для обычного расстояния Евклида. Однако для этой меры влияние отдельных больших разностей (выбросов) уменьшается (т.к. они не возводятся в квадрат). Формула для расчета манхэттенского расстояния:

$$p(x, x') = \sum_{i}^{n} |x_i - x_i'|$$

4. Степенное расстояние

Применяется в случае, когда необходимо увеличить или уменьшить вес, относящийся к размерности, для которой соответствующие объекты сильно отличаются. Степенное расстояние вычисляется по следующей формуле:

$$p(x, x') = \sqrt[r]{\sum_{i}^{n} (x_i - x'_i)^p}$$

где r и p – параметры, определяемые пользователем. Параметр p ответственен за постепенное взвешивание разностей по отдельным координатам, параметр r ответственен за прогрессивное взвешивание больших расстояний между объектами. Если оба параметра – r и p — равны двум, то это расстояние совпадает с расстоянием Евклида.

1.3. Критерии выбора метода кластеризации

На сегодняшний момент существуют сотни различных методов кластеризации. Для выбора нужного метода необходимо учитывать множество факторов, которые включают в себя:

Цель кластеризации

Для разных целей могут быть эффективно применимы разные методы. Так, например, если цель состоит в том, чтобы найти хорошие места для размещения магазина то целью будет кластеризация клиентов таким образом, чтобы сумма расстояний до центра кластеров была минимальна. В таком случае следует выбрать «метод разбиения», например K – means и C – means. Однако этот метод не подойдет для задач анализа растровых данных и распознавания изображений т.к. в таких случаях стоит задача кластеризации данных со сложными формами. Подходящим методом будет «плотностной метод».

Баланс между качеством и скоростью

Как правило, выбор метода кластеризации — это компромисс между качеством работы и быстродействием. [14] На скорость кластеризации сильно влияет объем данных и некоторые точные алгоритмы, качественно обрабатывающие небольшие базы данных, могут быть не способны справиться с большими объёмами за необходимый промежуток времени. В таком случае приходят на помощь различные методы сжатия данных с последующей их кластеризацией. Сжатие данных, как правило, сопровождается потерями что ведет к падению качества получаемых кластеров. [15] При таком подходе главной задачей является сжатие данных

таким образом чтобы существенно увеличить скорость кластеризации, сильно не теряя в качестве результата.

Характеристики данных

О похожести двух объектов данных мы можем судить на основе их характеристик. И если характеристики представлены в числовом виде, то без особых проблем мы можем рассчитать расстояния между ними. В настоящее время большинство методов рассчитано на обработку именно числовых характеристик. Дело обстоит иначе если характеристики представлены в виде двоичном, порядковом и иных видов. Для таких данных этого, как правило, ставят в соответствие некие числовые значения, которые впоследствии уже можно подвергнуть кластеризации. [16]

Когда мы говорим о размерности данных, подразумеваем количество характеристик в каждом объекте. Многие методы хорошо работают с данными малой размерности, при этом испытывая трудности если размерность данных увеличивается. Так, за увеличением размерности следует увеличение времени кластеризации и снижение качества итоговых кластеров. Решением является либо уменьшение размерности, либо подбор подходящего под требования метода.

Количество шумов

Некоторые методы кластеризации очень чувствительны к шумам и выбросам. Про это нужно не забывать при выборе метода, особенно если исходные данные сильно «зашумлены».

1.4. Описание метода K – means

K – means – это один из самых популярных методов используемых для кластеризации большого объёма данных. Данный метод относится к «методам разделения» и его суть заключается в том, чтобы классифицировать объекты на k кластеров. Метод начинает свою работу с того, что случайным образом распределяет центры кластеров. В последующих итерациях положение этих центров будет изменяться и производиться пересчет принадлежности данных к кластеру. Данный процесс повторяется до тех пор, пока не будет соблюдено условие остановки. Как правило, условием остановки является ситуация, при которой положение центров кластеров меняется на очень малое значение.

Суть алгоритма в том, что он стремится минимизировать суммарное квадратичное отклонение точек кластеров от центров этих кластеров:

$$V = \sum_{i=1}^{k} \sum_{x \in S_i} (x - \mu_i)^2$$

где k – число кластеров, S_i – полученные кластеры, $i = 1,2,3 \ldots, k$, а μ_i – центры масс всех векторов x из кластера S_i. V – сумма квадратичных ошибок.

Метод K –means, как и любой «метод разделения» требует заранее известное количество кластеров.

Шаги:

1. Случайно выбираем расположение центров кластеров;
2. Рассчитываем расстояние между каждым объектом и центрами кластеров. Присваиваем объект к ближайшему кластеру;
3. Для всех кластеров пересчитываем центр;
4. Повторять 2,3 пункты до тех пор, пока центры кластеров не перестанут смещаться.

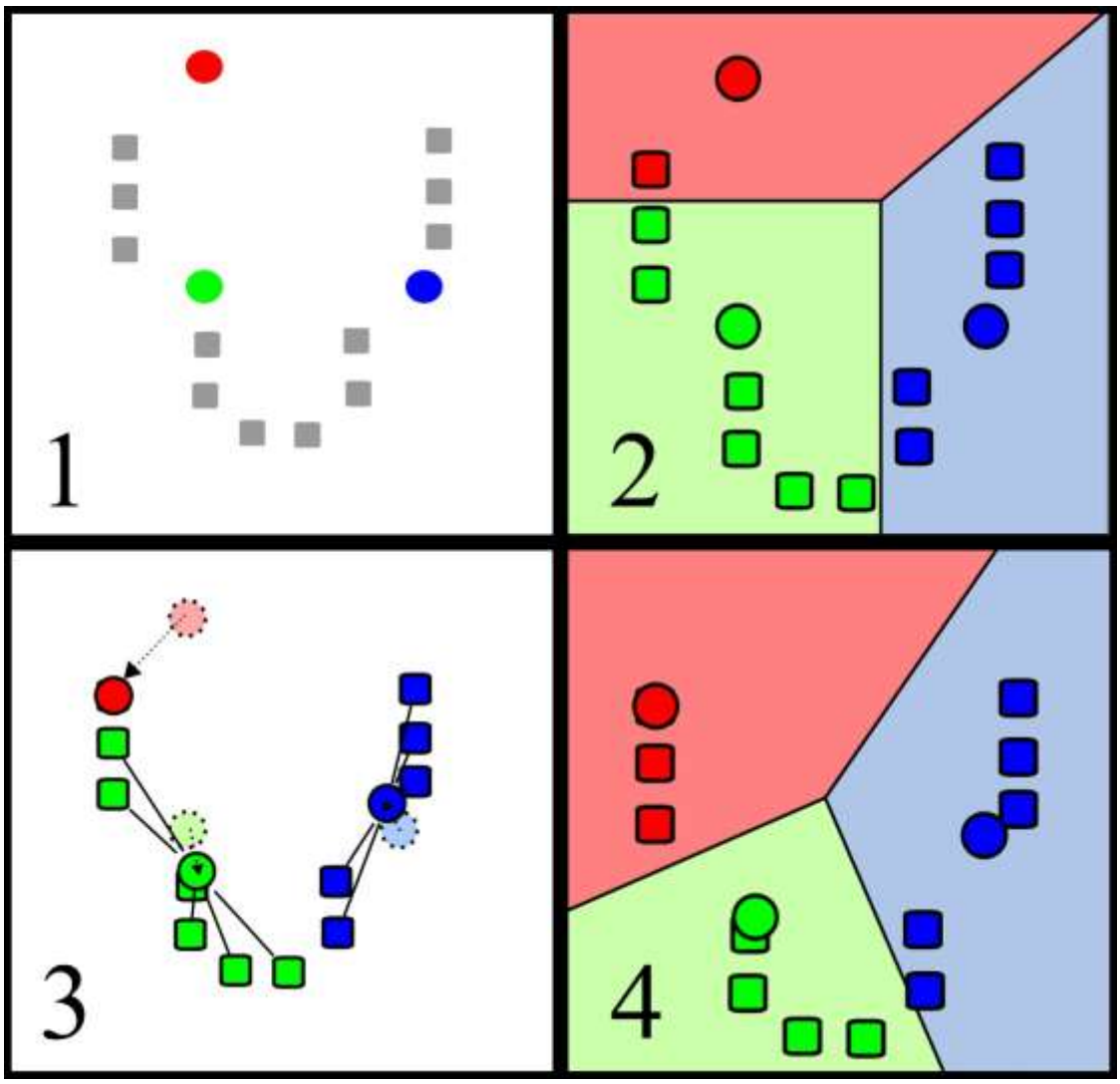

Рисунок 1 – Демонстрация работы метода

На рисунке 1 продемонстрирована работа метода где:

1. Центры 3х кластеров выбираются случайным образом;
2. Соотнесение точек к кластерам относительно центров;
3. Вычисление новых центров кластеров;
4. Соотнесение точек к кластерам уже относительно новых центров.

Метод кластеризации K – means всегда сходится к локальному минимуму. Количество итераций зависит от начальных центров кластеров. Сложность алгоритма по времени, нужному для сходимости, равна $O(nmkt)$. [17] Где n – количество точек, m – количество свойств (атрибутов), k –

количество кластеров, t – количество итераций метода. Требование к памяти достаточно скромные $O((m+k)n)$.

Проблемы метода K – means [18]:

1. Необходимо знать начальное количество кластеров;
2. Результат и скорость метода зависят от начального положения центров кластеров;
3. Метод не способен адекватно обработать объект, если тот принадлежит сразу к нескольким кластерам в равной степени.

На данный момент существуют работы, которые посвящены улучшению метода K means. Так, например работа «Research on k-means Clustering Algorithm» [19] посвящена оптимизации скорости работы метода. Суть оптимизации в том, чтобы лишний раз не пересчитывать принадлежность данных к кластеру что существенно может сократить время работы. Так же время работы можно сократить, оптимизируя выбор начального расположения центров. [20 - 22]

1.5. Описание метода C – means

Метод C means является методом нечеткой кластеризации. Нечеткая кластеризация во многих ситуациях более естественна чем строгая кластеризация. Так, например объекты на границах между несколькими кластерами будут принадлежать не одному кластеру, а всем с определенным коэффициентом между 0 и 1. Сам метод C means был разработан в 1973 году. [23] и улучшен в 1981 [24]

В целом шаги, выполняемые методом C means очень похожи на шаги у метода K means. Это:

1. Случайно выбираем центры кластеров;
2. Рассчитываем матрицу принадлежности элементов к кластерам:

$$r_{ij} = \frac{\mathcal{N}(d(x_i, c_j)|\mu = 0, \sigma)}{\sum_j^k \mathcal{N}(d(x_i, c_j)|\mu = 0, \sigma)},$$

где x_i – i -й элемент множества, c_j – центр кластера j, $d(x_i, c_j)$ – расстояние между точками x_i и c_j, $\mathcal{N}$ – плотность вероятности нормального распределения в точке $d(x_i, c_j)$;

3. Пересчитываем центры кластеров

$$c_j \leftarrow \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}};$$

4. Рассчитываем функцию потерь:

$$\mathrm{J} = \sum_j^k \sum_i^N d\left(x_i, c_j\right)^2 r_{ij};$$

До тех пор, пока значение функции потерь уменьшается то повторять цикл.

Сложность алгоритма по времени, нужному для сходимости, равна $O(nmk\hat{}2t)$. [25] Где n – количество точек, m – количество свойств

(атрибутов), k – количество кластеров, t – количество итераций метода. Требование к памяти такие же, как и у K – means $O\big((m+k)n\big)$.

Алгоритм C means можно считать в определенном смысле улучшенной версией метода K means. Однако из-за свойства нечеткости метод работает гораздо медленней. Точность метода так же страдает от выбросов и шумов в данных. Поэтому существуют различные модификации этого метода, улучшающие либо скорость работы [26], либо стойкость к шумам [27], либо качество кластеризации [28]. Но и в этом случае соблюдается некий баланс и улучшение одного параметра приводит к ухудшению другого.

1.6. Описание метода HDBSCAN

HDBSCAN - это алгоритм кластеризации, разработанный Кампелло, Мулави и Сандером. Он расширяет DBSCAN, превращая его в алгоритм иерархической кластеризации, затем кластеры извлекаются на основании стабильности. [29]

Данный алгоритм кластеризации можно разбить на следующие шаги:

1. Преобразовать пространство в соответствии с плотностью (или разрежённостью);
2. Построить минимальное остовное дерево взвешенного по расстоянию графа;
3. Построить кластерную иерархию связанных компонентов;
4. Уплотнение иерархии кластеров на основе минимального размера кластера;
5. Получение стабильных кластеров из сгущенного дерева.

Для нахождения кластеров, нужно найти острова с более высокой плотностью точек среди моря редких шумов. Предположение о существовании шума важно т.к. реальные данные не являются идеальными и как правило имеют выбросы, поврежденные записи и шум. Ядром алгоритма кластеризации является кластеризация методом одиночной связи. Она может быть весьма чувствительной к шуму: одна точка данных в неправильном месте может действовать как мост между островками, склеивая их вместе. Очевидно, нужно, чтобы алгоритм был устойчивым к шуму, поэтому нужно найти способ «понизить уровень моря» перед запуском единого алгоритма связи.

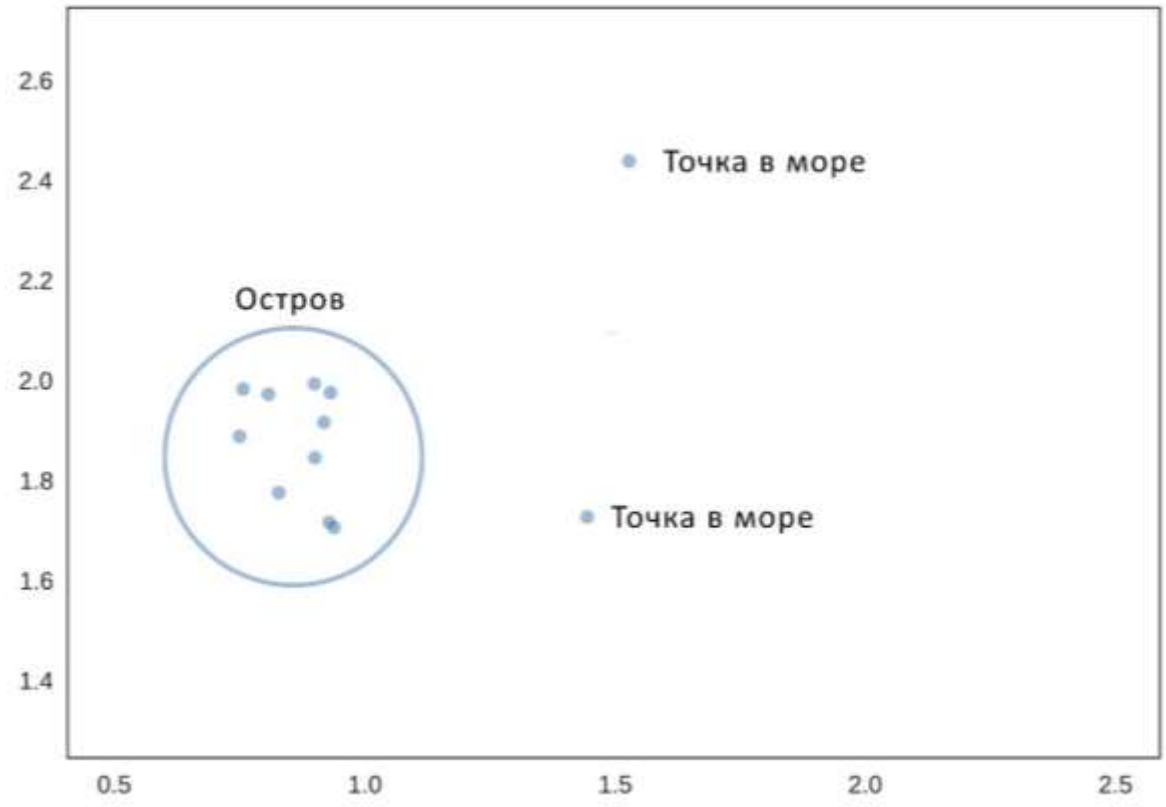

Рисунок 2– Точки на острове и точки в море

Как можно охарактеризовать «море» и «суша» без кластеризации? Можно рассчитать оценку плотности. Соответственно точки с более низкой плотностью считать, как «море». Цель данного этапа не в том, чтобы идеально отделить «море» от «суши», а в том, чтобы помочь нашему ядру кластеризации быть более устойчивым к шуму. Следующим этапом будет «понижение уровня моря». Этим этапом мы хотим добиться чтобы точки в море стали еще более отдаленными друг от друга и от «суши».

Для этого воспользуемся какой-нибудь простой оценкой плотности. Для этой цели вполне подходит расстояние K - ближайших соседей. Обозначим расстояние до центра кластера точки x для параметра k как $core_k(x)$. Остается только разнести точки с низкой плотностью (большим расстоянием до центра кластера) еще дальше друг от друга. Простой способ сделать это - определить новую метрику расстояния между точками, которая называется расстоянием

взаимной достижимости (mutual reachability distance). Определяется это расстояние следующим образом:

$$d_{mreach-k}(a,b) = \max\{core_k(b), core_k(b), d(a,b)\}$$

где d (a, b) - исходное метрическое расстояние между a и b. При этом точки с высокой плотностью (с малым расстояние до центра кластера) остаются на том же расстоянии друг от друга, но менее плотные точки отталкиваются так, чтобы находиться на расстоянии, $core_k(x)$ от любой другой точки. Это эффективно «понижает уровень моря», разнося разреженные «морские» точки, оставляя «сушу» нетронутой. Однако нужно быть осторожным с параметром k, большие значения интерпретируют больше точек как находящиеся в «море». Разберем данный этап на примере. Возьмем параметр k равное 6. Далее для произвольно выбранной точки нарисуем круг с радиусов равным «расстоянию до центра кластера» в который входят все шесть точек (включая центральную).

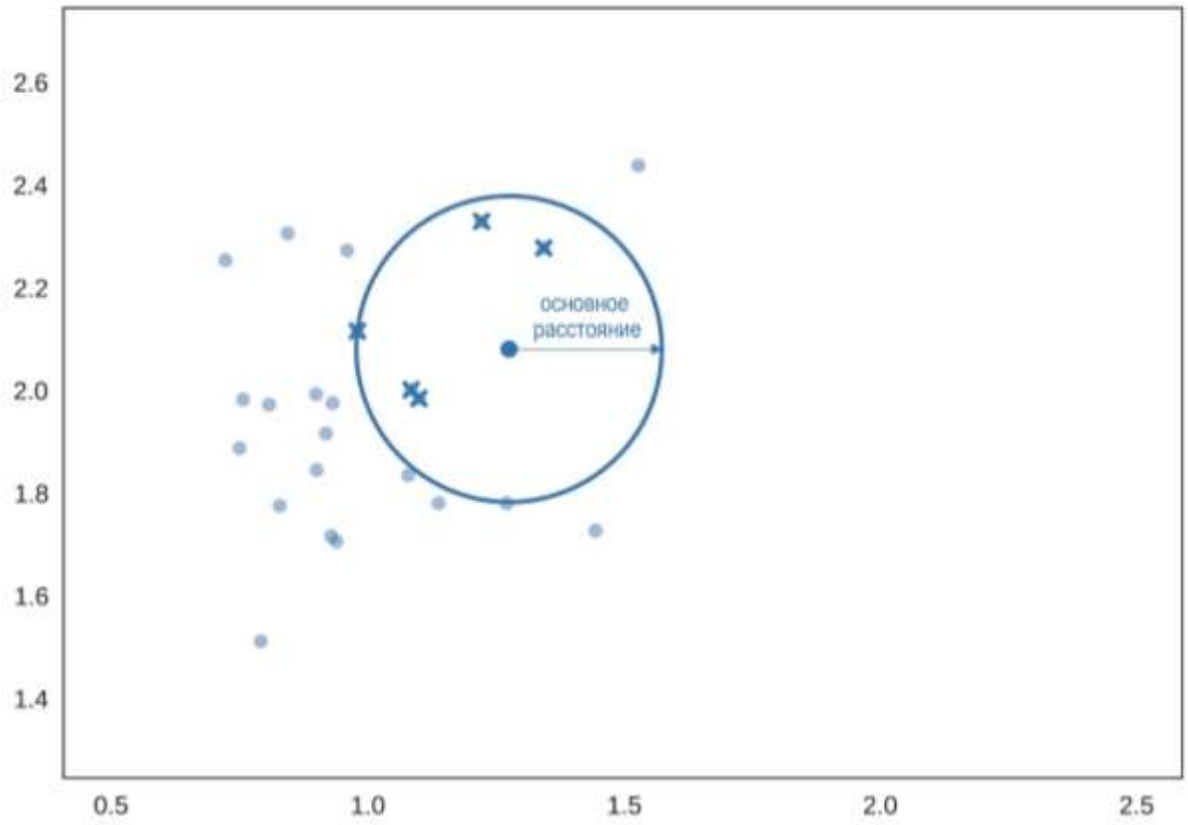

Рисунок 3 – Основное расстояние

Далее возьмем еще две точки и повторим то, что сделали в прошлый раз.

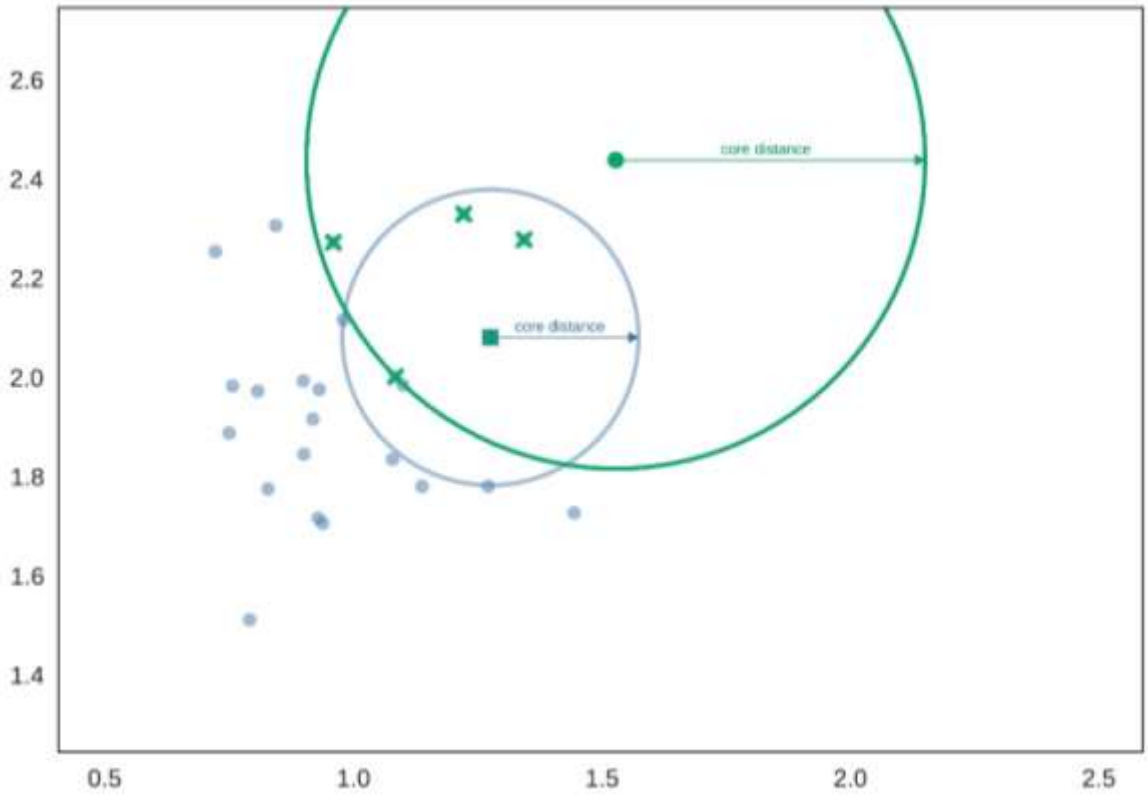

Рисунок 4 – Этапы расчета расстояния взаимной достижимости

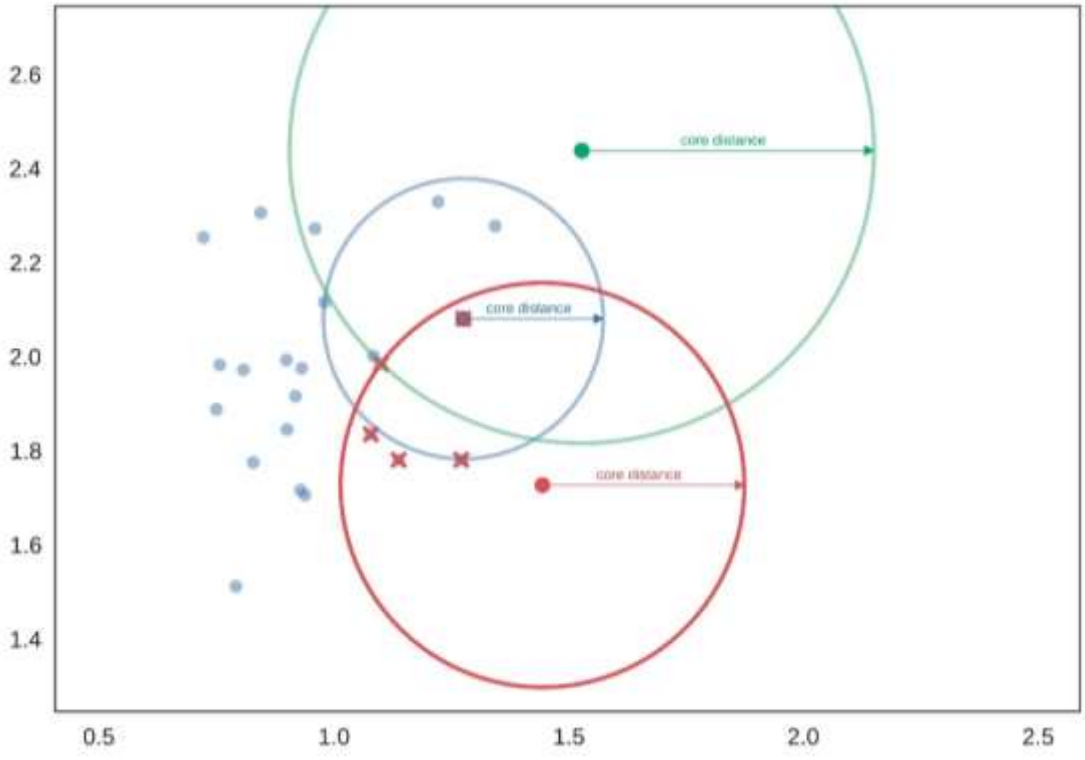

Рисунок 5 – Этапы расчета расстояния взаимной достижимости

Теперь для расчета расстояния взаимной достижимости между синими и зелеными центральными точкам проведем линию, соединяющую их.

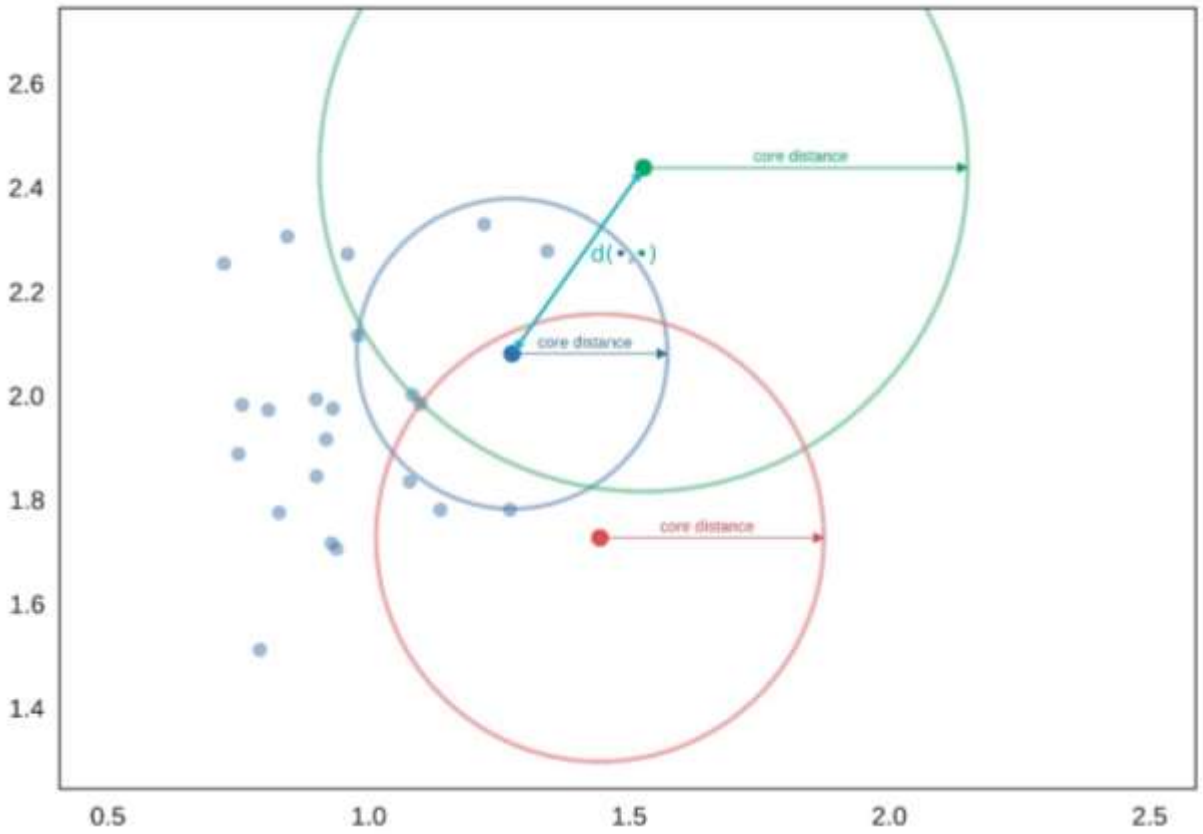

Рисунок 6 – Этапы расчета расстояния взаимной достижимости

Линия, соединяющая две центральные точки меньше, чем расстояние до центра кластера у зеленого круга. Поэтому расстояние взаимной достижимости для зеленой и синей точки будет равно радиусу зеленого круга.

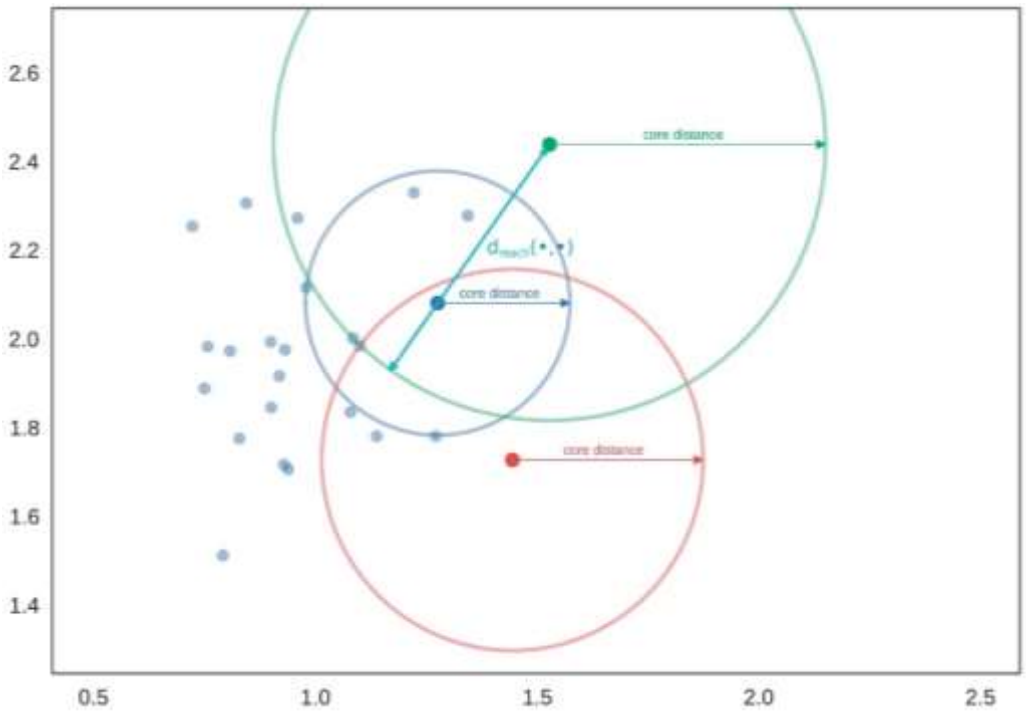

Рисунок 7 – Этапы расчета расстояния взаимной достижимости

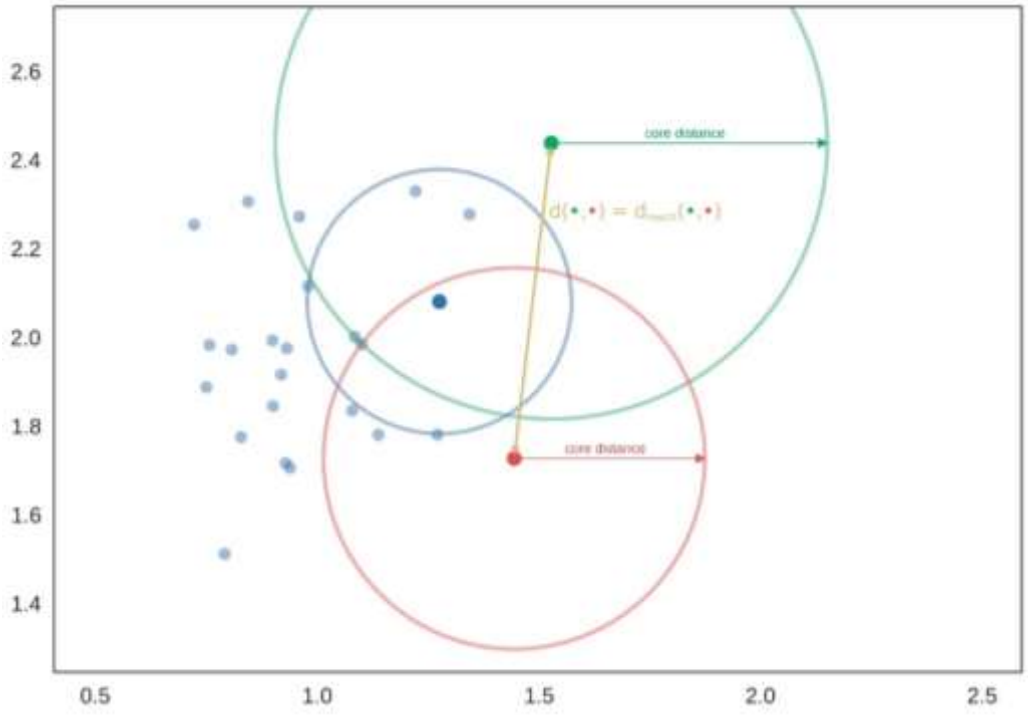

Рисунок 8 – Этапы расчета расстояния взаимной достижимости

Расстояние взаимной достижимости для зеленой и красной точки равно расстоянию между ними. Т.к. это расстояние больше, чем радиусы кругов.

После того как была получена метрика расстояния взаимной достижимости для данных, можно приступить к поиску островов с высокой плотностью точек. Очевидно, что разные острова могут иметь разную плотность. Поэтому рассмотрим данные как взвешенный граф с точками данных в виде вершин и ребром между двумя точками с весом, равным расстоянию взаимной достижимости этих точек.

Теперь будем изменять пороговое значение, двигаясь от большего к меньшему. Ребра, которые будут иметь вес выше порога нужно будет отбросить. По мере того, как мы отбрасываем ребра, граф будет разъединяться на связанные компоненты. Со временем у нас будет иерархия связанных компонентов на различных пороговых уровнях.

Однако реализация такого алгоритма достаточно ресурсоемкая задача из-за того, что количество рёбер равно n^2. Правильным решением будет поиск такого минимального набора ребер, при котором избавление от одного из ребер приводило к разъединению графа. Для этой цели как раз подходит так называемое «Минимальное остовное дерево».

Минимальное остовное дерево можно построить с помощью алгоритма Прима. Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

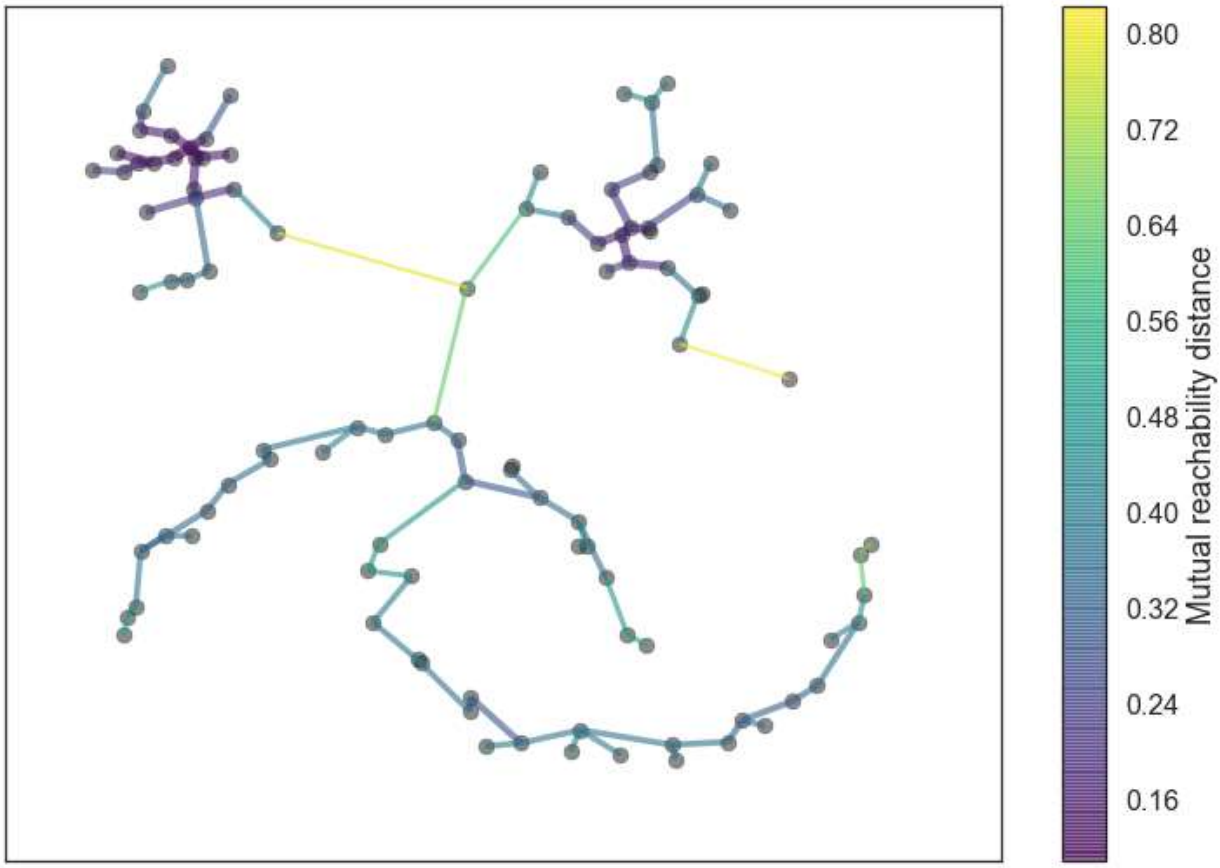

Рисунок 9 – Минимальное остовное дерево построенное на основе расстояния взаимной достижимости

Следующим шагом будет преобразование дерева в иерархию связанных компонентов. Для этого отсортируем ребра дерева по возрастанию расстояния. Тут и появляется проблема в идентификации кластера. Когда два кластера могут быть объединены в один. Но это можно обойти, используя систему непересекающихся множеств. Результат отобразим как дендрограмму.

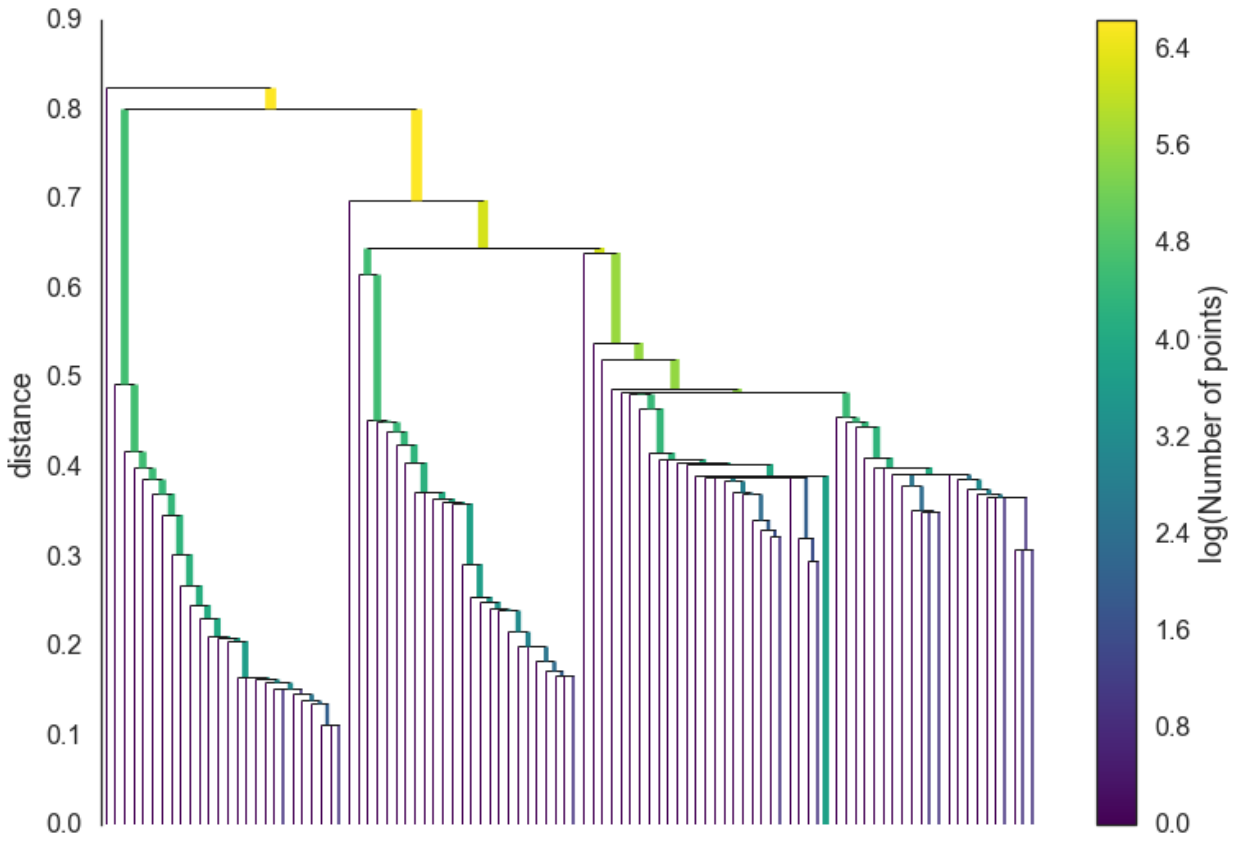

Рисунок 10 – Дендрограмма для дерева

Получить конкретные кластеры из дендограммы можно проведя линию по оси «расстояние». Так это происходит в алгоритме DBSCAN и считая кластеры, состоящие из одной точки как шум. Весьма очевидно, что появляется проблема. Где провести эту линию? За это в DBSCAN отвечает отдельный параметр. Дела обстоят хуже если искомые кластеры переменной плотности. В таком случае метод с «проведением линии» никак не подходит. Соответственно нужно каким-то образом «вырезать» дерево в различных местах. Тут и начинаются отличия HDBSCAN от DBSCAN.

Далее в HDBSCAN следует процедура сжатия дерева. Как можно увидеть на рисунке 9, часто бывает так, что расщепление кластера - это одна или две точки; однако вместо того, чтобы рассматривать его как кластер, разделяющийся на два новых кластера, будем рассматривать его как кластер, который «теряет точки». Но для начала нам нужно определить значение минимального размера кластера, которое берется в как входной параметр в

HDBSCAN. Далее можем пройтись по иерархии и при каждом разделении спрашивать, имеет ли один из новых кластеров, созданных разделением, меньше точек, чем минимальный размер кластера. Если это так, и новый кластер меньше, чем минимальный размер кластера, объявим его «точками, выпадающими из кластера», а более крупный кластер будет считаться кластером родителем, отмечая, какие точки «выпали из кластер» и на каком расстоянии это произошло. Если разделение делится на два кластера, каждый из которых, по крайней мере, равен минимальному размеру кластера, мы считаем, что это настоящий кластер, и пусть это разделение сохраняется в дереве. Пройдя через всю иерархию, мы в итоге получим гораздо меньшее дерево с небольшим количеством узлов, каждый из которых имеет данные о том, как размер кластера в этом узле уменьшается на разном расстоянии. Это можно визуализировать это как дендрограмму, похожую на рисунок 8.

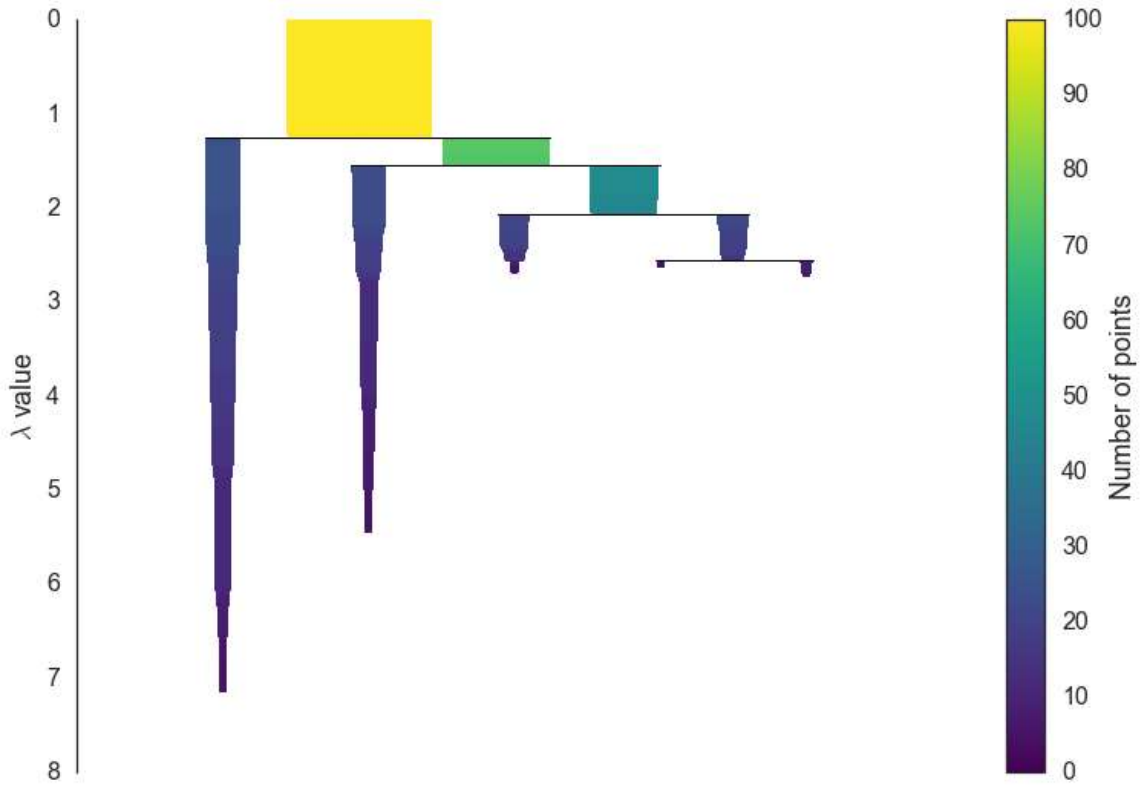

Рисунок 11 – Дендрограмма HDBSCAN
при минимальном размере кластера равному 5

Теперь нужно извлечь кластеры из дендограммы выше. Что бы это сделать нужно будет выбрать те кластеры, которые имеют наибольшую закрашенную область на графике. Так же нам нужно добавить еще одно требование: если кластер был выбран, то кластеры, являющиеся его потомками для выбора недоступны. Приступим к формализации.

Во-первых, для дальнейших расчетов в качестве меры будет брать не просто расстояние, а $\lambda = \frac{1}{distance}$. Так же определим значения λ_{birth} и λ_{death} где λ_{birth} значение, при котором кластер появился после разделения от родительского, а λ_{death} когда кластер разделился на более мелкие кластеры. Для каждой точки p в данном кластере определим значение λ_p которое будет означать при каком λ точка отделиться от кластера. Очевидно, что λ_p будет лежать в границах между λ_{birth} и λ_{death}. Теперь можно рассчитать параметр «стабильности» для каждого кластера.

$$\sum_{p \in cluster} (\lambda_p - \lambda_{birth})$$

Пометим все конечные узлы как кластеры и начнем прорабатывать дерево в обратном топологическом порядке. Если сумма стабильностей дочерних кластеров больше, чем стабильность родительского кластера, то мы устанавливаем стабильность кластера как сумму дочерних стабильностей. Если, стабильность родительского кластера больше, чем сумма его дочерних кластеров, тогда мы помечаем кластер как выбранный. У дочерних кластеров в таком случае пометка убирается. Как только достигаем корневого узла, возвращаем все выбранные кластеры как результат.

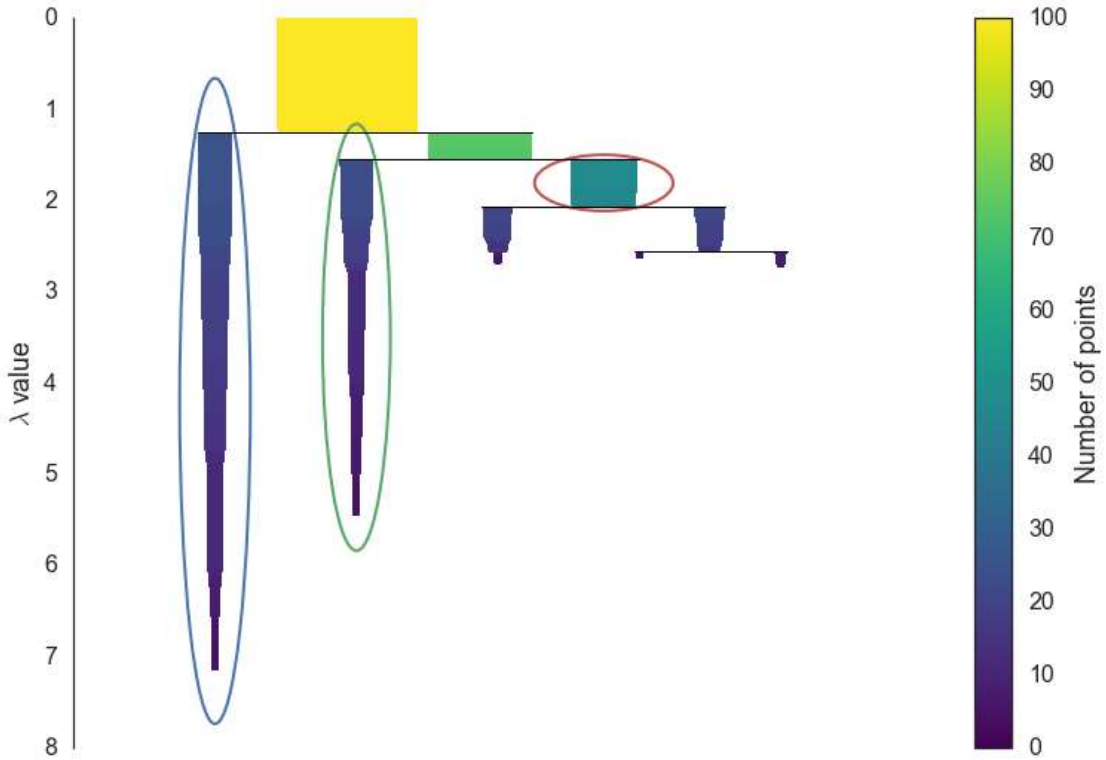

Рисунок 12 – Дендрограмма HDBSCAN

с помеченными кластерами

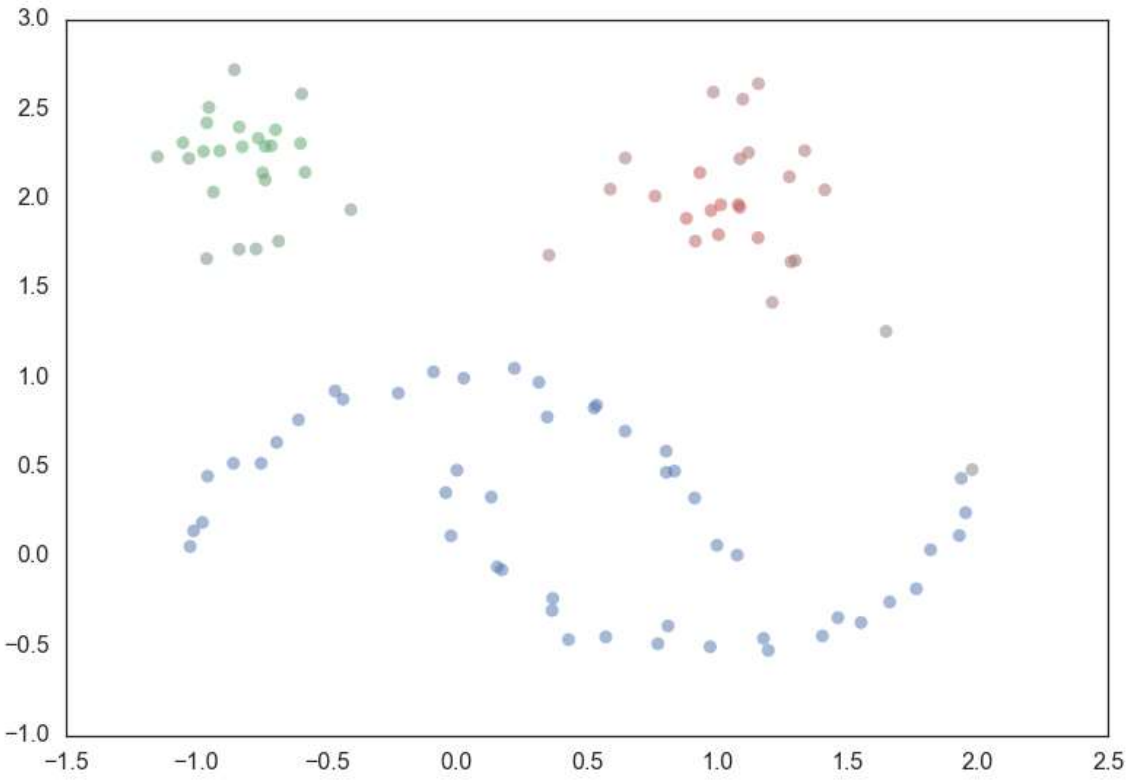

Рисунок 13 – Результат кластеризации HDBSCAN

HDBSCAN далеко не единственный алгоритм который пытается заменить неочевидный параметр «epsilon» в DBSCAN, тот самый параметр который отвечает за «проведение линии» на дендограмме, на иные, более интуитивно понятные параметры [30-33].

1.7. Описание метода BIRCH

BIRCH (англ. balanced iterative reducing and clustering using hierarchies) – алгоритм кластеризации основной целью которого это обработка больших массивов данных при малом потреблении ресурсов (оперативной памяти в частности). [34]

Экономное использование ресурсов достигается благодаря преобразованию данных в признак кластеризации (Clustering Feature). Признак кластеризации (CF) для точки $\overrightarrow{X_i}$ где $i = 1,2, \dots, N$, N – количество всех точек в пространстве размерностью d:

$$CF = (N, LS, SS)$$

$$\overrightarrow{LS} = \sum_{i=1}^{N} \overrightarrow{X_i}, \qquad \overrightarrow{SS} = \sum_{i=1}^{N} \left(\overrightarrow{X_i}\right)^2$$

Признаки кластеризации организуются в CF-дерево, высоко сбалансированное дерево с двумя параметрами: коэффициентом ветвления B и порогом T. Каждый нелистовой узел состоит максимум из B входов вида $[CF_i, child_i]$,где $child_i$ является указателем на его i-ого потомка, а CF_i является признаком кластеризации, представляющим связанный подкластер. Лист содержит не более L входов, каждый вида$[CF_i]$. Он также имеет два указателя, prev и next, которые используются для соединения в цепь все листы. Размер дерева зависит от параметра T. Требуется, чтобы узел A вмещался на страницу размера P. B и L определяются значением P. Таким образом, P может меняться для настройки производительности. Это очень компактное представление набора данных, поскольку каждый лист не является отдельной точкой данных, а является подкластером.

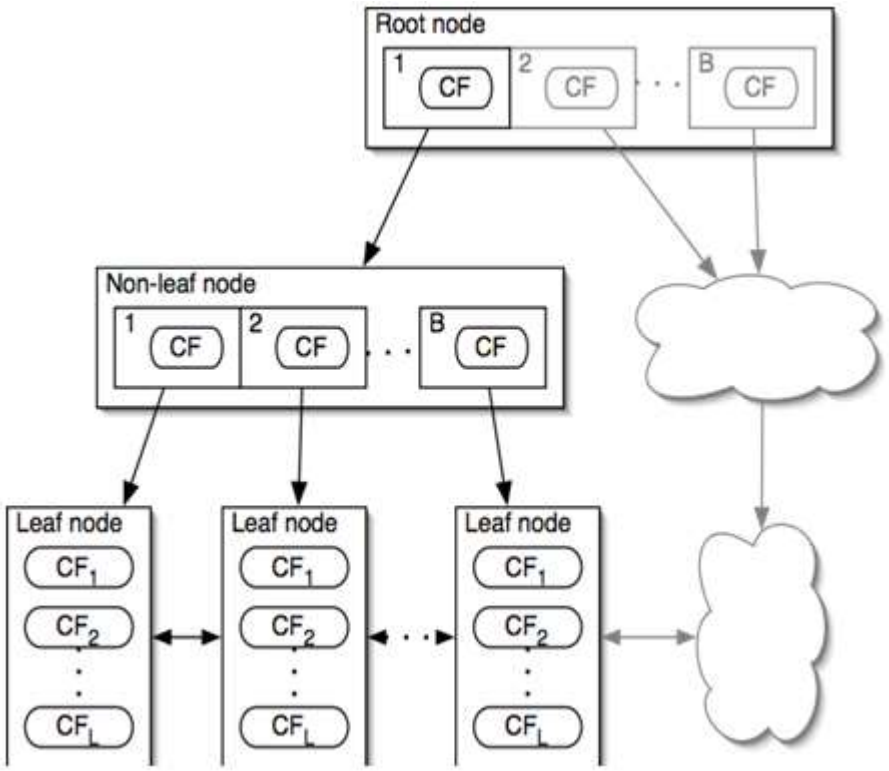

Рисунок 14 – Схема CF – дерева

Опишем алгоритм вставки одного CF элемента в CF – дерево.

1. Нахождение листа. Начиная с корня, спускаемся по CF дереву выбирая ближайший дочерний узел в соответствии с выбранной метрикой расстояния.
2. Изменим лист. Достигнув листа, находим ближайшую запись и проверяем может ли она «поглотить» CF элемент без пересечения порогового условия. Если может, то нужно обновить CF запись, иначе добавляем новую. Если на листе недостаточно места для этой новой записи, то нужно разделить листовой узел. Разделение узлов выполняется путем выбора двух наиболее удаленных друг от друга записей в качестве начальных значений и перераспределения оставшихся записей по расстоянию.
3. Измените путь к листу: вспомните, как каждый «не листовой» узел сам по себе является CF, состоящим из CF всех его дочерних элементов.

Поэтому после вставки записи CF в лист мы обновляем информацию CF для каждой неконечной записи на пути к листу. В случае разделения, нужно вставить новую «не листовую» запись в родительский узел и сделать так, чтобы она указывала на сформированный лист. Если согласно B, у родителя недостаточно места, то мы должны также разделить родителя и так далее до корня дерева.

Теперь приступим к описанию алгоритма кластеризации.

Фаза 1. Алгоритм сканирует данные и вставляет точки в дерево. Если ему не хватает памяти до завершения сканирования данных, он увеличивает пороговое значение и перестраивает новое, более маленькое CF-дерево, повторно вставляя листовые записи старого CF-дерева в новое CF-дерево. После того, как все старые листовые записи были повторно вставлены, сканирование данных и вставка в новое CF-дерево возобновляется с той точки, в которой оно было прервано.

Хороший выбор порогового значения может значительно сократить количество перестроений. Однако, если начальный порог слишком высок, мы получим менее детальное CF-дерево, чем это возможно с доступной памятью.

При желании мы можем выделить фиксированный объем пространства на жестком диске для обработки выбросов. Выбросы представляют собой листовые записи с низкой плотностью, которые считаются неважными по отношению к общей структуре кластеризации. Когда мы перестраиваем CF-дерево путем повторной вставки старых листовых записей, размер нового CF-дерева уменьшается двумя способами. Во-первых, мы увеличиваем пороговое значение, тем самым позволяя каждой записи листа поглощать больше точек. Во-вторых, мы рассматриваем некоторые листовые записи как потенциальные выбросы и записываем их на диск. Старая конечная запись считается потенциальным выбросом, если она имеет гораздо меньше точек данных, чем в среднем. Увеличение порогового значения или изменение распределения в

ответ на новые данные вполне может означать, что потенциальный выброс больше не считается выбросом. Как следствие, потенциальные выбросы сканируются, чтобы проверить, могут ли они быть повторно поглощены в дереве, не вызывая увеличение размера дерева.

Фаза 2 (помечена автором как необязательная). Учитывая, что определенные алгоритмы кластеризации работают лучше всего, когда число объектов находится в определенном диапазоне, мы можем сгруппировать переполненные подкластеры в более крупные, что приведет к общему меньшему CF-дереву.

Фаза 3. Почти любой алгоритм кластеризации может быть адаптирован для классификации признаков кластеризации вместо точек данных. Например, мы могли бы использовать K-средних для классификации наших данных, в то же время извлекая выгоду из BIRCH.

Фаза 4. Хотя дерево могло перестраиваться несколько раз, исходные данные сканировались лишь один раз. Фаза 4 включает в себя дополнительные проходы по данным для исправления неточностей, вызванных тем, что алгоритм кластеризации применяется к «грубой» сводке данных. Этап 4 также дает нам возможность избавиться от выбросов.

При кластеризации с самого начала нужно настроить 2 параметра. Это параметр «n_clusters» количество кластеров и «threshold» максимальный радиус подкластера. Проблема в том, что при определенных значениях параметра «threshold» нельзя найти то количество кластеров что указанно в «n_clusters». Эту проблему решена в алгоритме A-BIRCH. Которая является модификацией алгоритма BIRCH [35]. Существуют так же различные работы, которые занимаются улучшением работы алгоритма в общем [36-39].

1.8. Вывод по теоретической части

Данная часть была посвящена обзору существующих методов кластеризации и их разбиению на категории. Подробно были рассмотрены выбранные для последующего кластерного анализа методы. Показаны шаги алгоритмов и работы других авторов, направленные на улучшение каких-либо аспектов алгоритма.

2. Практическая часть

2.1. Программные средства для кластеризации

В качестве языка программирования для кластерного анализа был выбран Python. Популярность Python как языка для кластерного анализа во многом обеспеченна библиотекой Sklearn. [40]

Sklearn – свободно распространяемая библиотека для языка программирования Python. Библиотека обладает различными алгоритмами классификации, регрессии и кластеризации. Разработка библиотеки началась с 2013 года и продолжается до сих пор. Последняя версия библиотеки была выпущена в мае 2019 и имеет версию 0.21.0. [41]

2.2. Данные

Данные для кластерного анализа представлены в виде таблицы Excel. Сама таблица содержит информацию о клиентах магазинов с 2014 по 2018 годы. Каждый клиент имеет следующие характеристики:

1. Дата регистрации;
2. Количество совершенных покупок;
3. Дата последнего визита;
4. Общая стоимость всех совершенных покупок.

Таблица содержит 1849140 записей.

Весь процесс кластеризации можно разделить на несколько этапов:

1. Отбор и преобразование данных;
2. Очистка данных от ошибочных записей;
3. Нормализация данных;
4. Кластеризация;
5. Графическое отображение.

Отбор и преобразование данных

Как было указанно ранее, следовать принципу «чем больше характеристик, тем лучше» это неверная стратегия, которая может привести к не качественной кластеризации. Поэтому для кластеризации были выбраны характеристики «Количество совершенных покупок» «Общая стоимость всех совершенных покупок» и «Стаж клиента» где последняя характеристика высчитывается как разница между последним и первым посещением магазина.

Очистка данных от ошибочных записей

Таблица содержит некоторое количество ошибочных записей. В большинстве имеется ошибка в дате так, например около 27000 записей содержат дату регистрации как 1753-01-01 и около 8 записей говорят о том, что последняя покупка была раньше регистрации. От таких данных решено было избавится путем исключения их из набора данных для последующей кластеризации.

Таблица 1 – Статистика о очистке данных

	Файл: Data_1	**Файл: Data_2**	**Итого**
Количество записей	1000000	849140	1849140
Записей, где дата регистрации позже даты последней покупки	8	0	8
Записей, где дата регистрации раньше 2000 года	17508	9722	27 230
Процент ошибочных данных	1.750%	1.146%	1.473%
Количество записей после исключения ошибочных данных	982484	839418	1821902

Помимо очевидно неправильных записей так же существует проблема с неполнотой данных на некоторых интервалах.

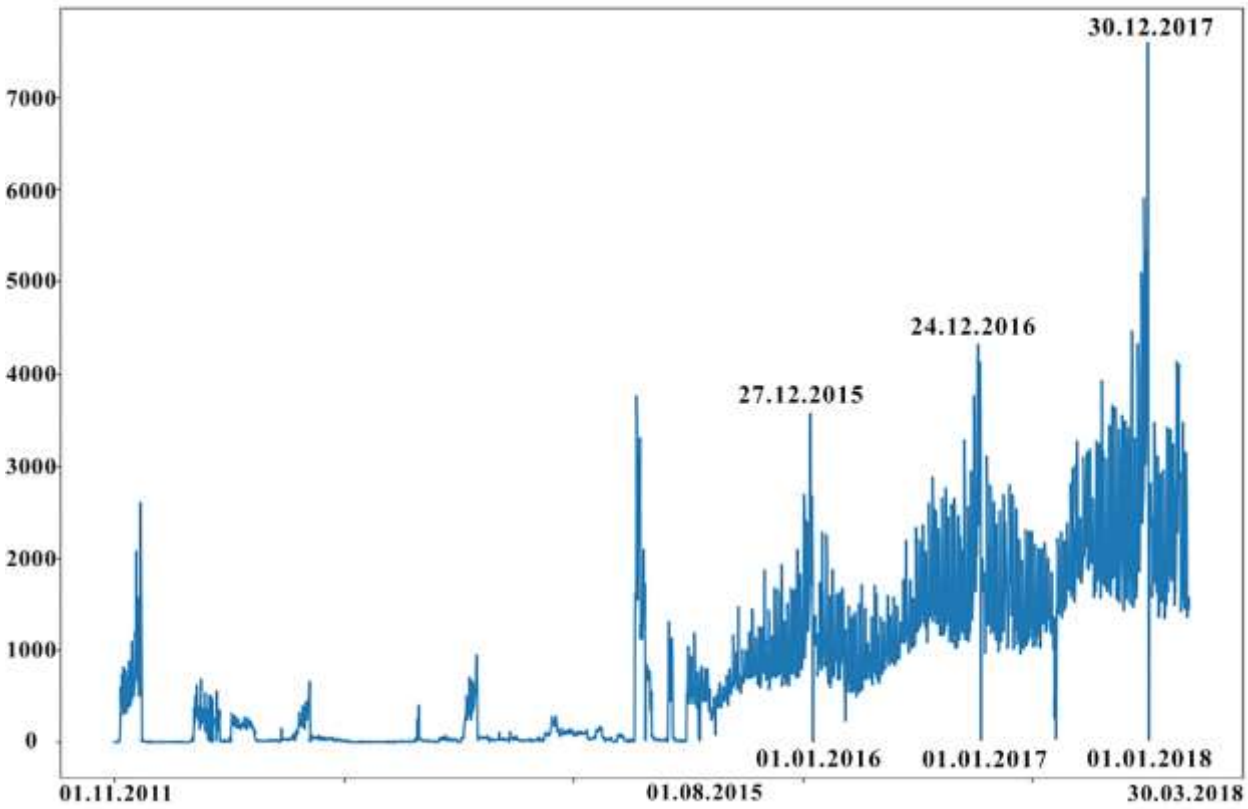

Рисунок 15 – График количества регистраций по дням

Как видно из рисунка 2 данные с 01.08.2015 – 30.03.2018 можно назвать предсказуемыми. Так, например, наблюдаются пики регистраций новых клиентов ближе к празднику «Новый год», закупка подарков и продуктов для праздника. И провалы первого января, когда большинство магазинов закрыты. Тоже самое нельзя сказать про отрезок с 01.11.2011 – 01.08.2015. Возможно, сбор данных в этом промежутке был затруднен или записи содержат неверную дату. В любом случае было решено удалить все записи с датой регистрации до 01.08.2015.

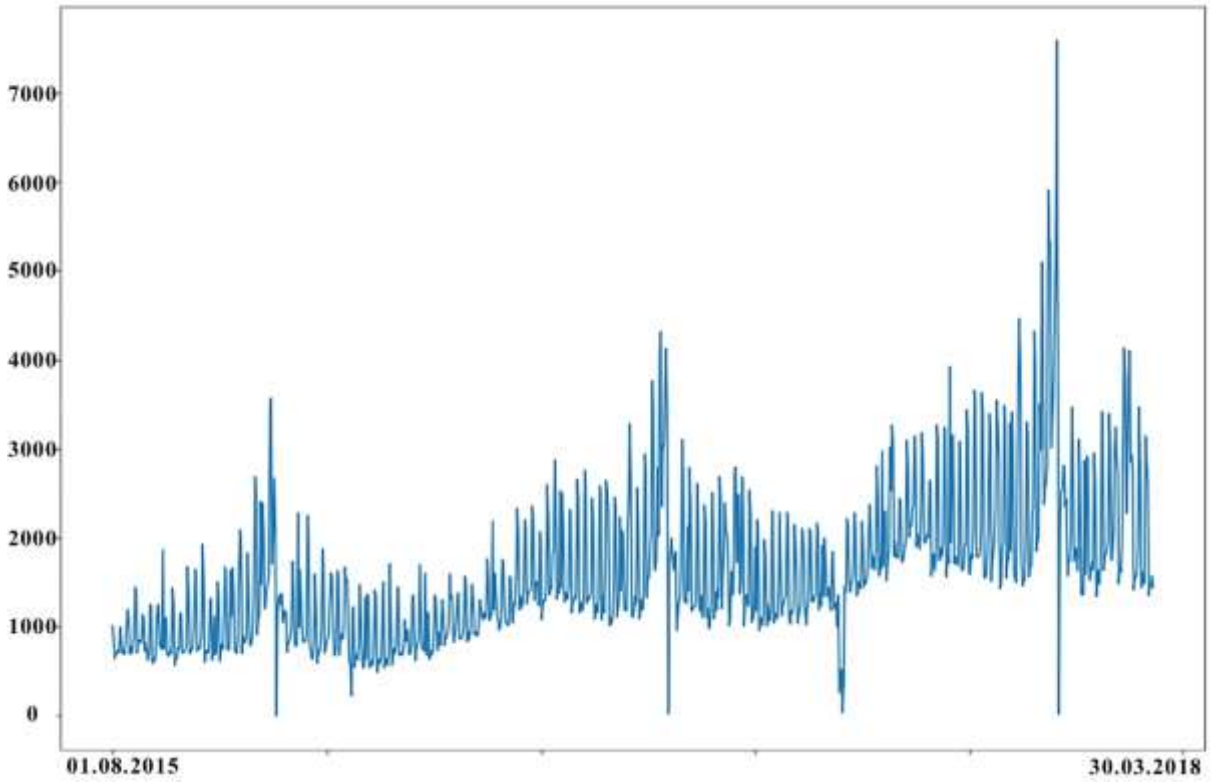

Рисунок 16 – График количества регистраций по дням для новых данных

Удалению подверглось 245057 записей. В общей сумме всех записей осталось 1576845.

Нормализация

Данные в таблице представлены в ненормализованном виде. Для нормализации решено было использовать нормализацию минимума-максимума. Для данной нормализации используется следующая формула

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)}$$

где X^* - нормализованное значение, min и max – минимальная и максимальная координата по всему множеству X

2.3. Результаты обработки K means

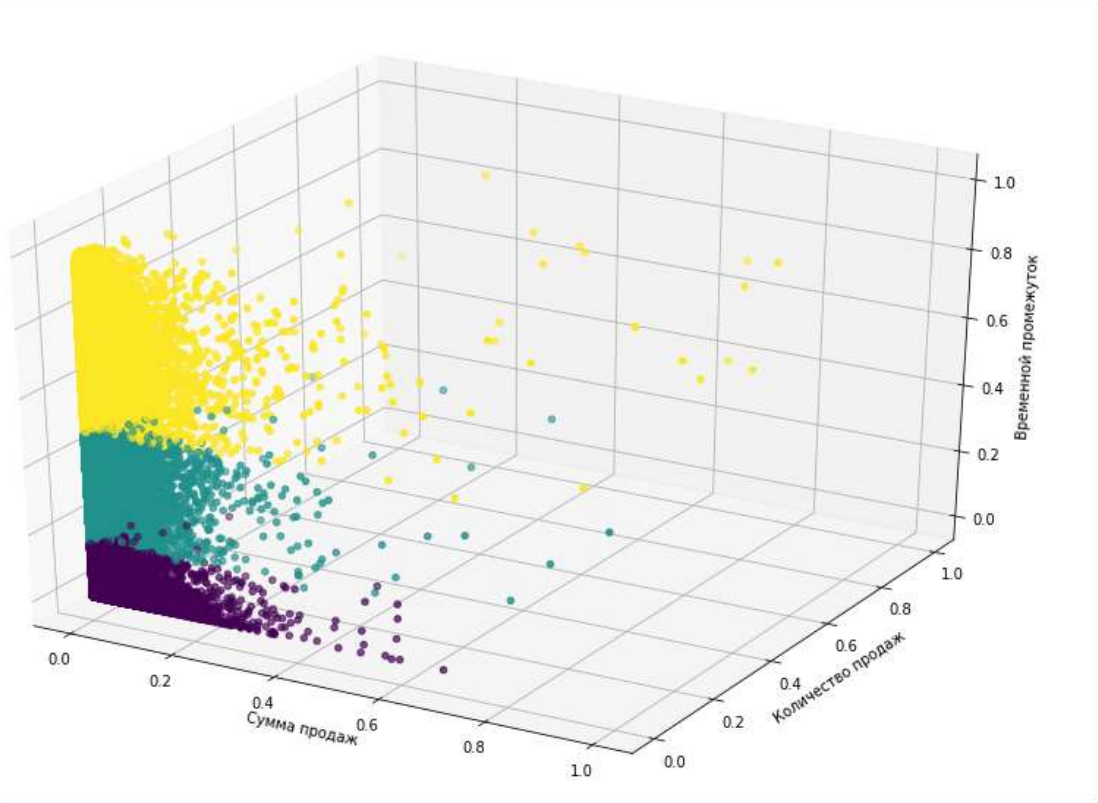

Рисунок 17 – Распределение данных по 3-м кластерам

Таблица 2 – Статистика для кластеризации

	1 (фиолетовый кластер)	2 (зеленый кластер)	3 (желтый кластер)
Средняя сумма покупок	20375.86	46659.57	73617.58
Среднее количество посещенных дней	9.58	300.90	645.61
Среднее количество совершенных покупок	1.39	3.28	4.20

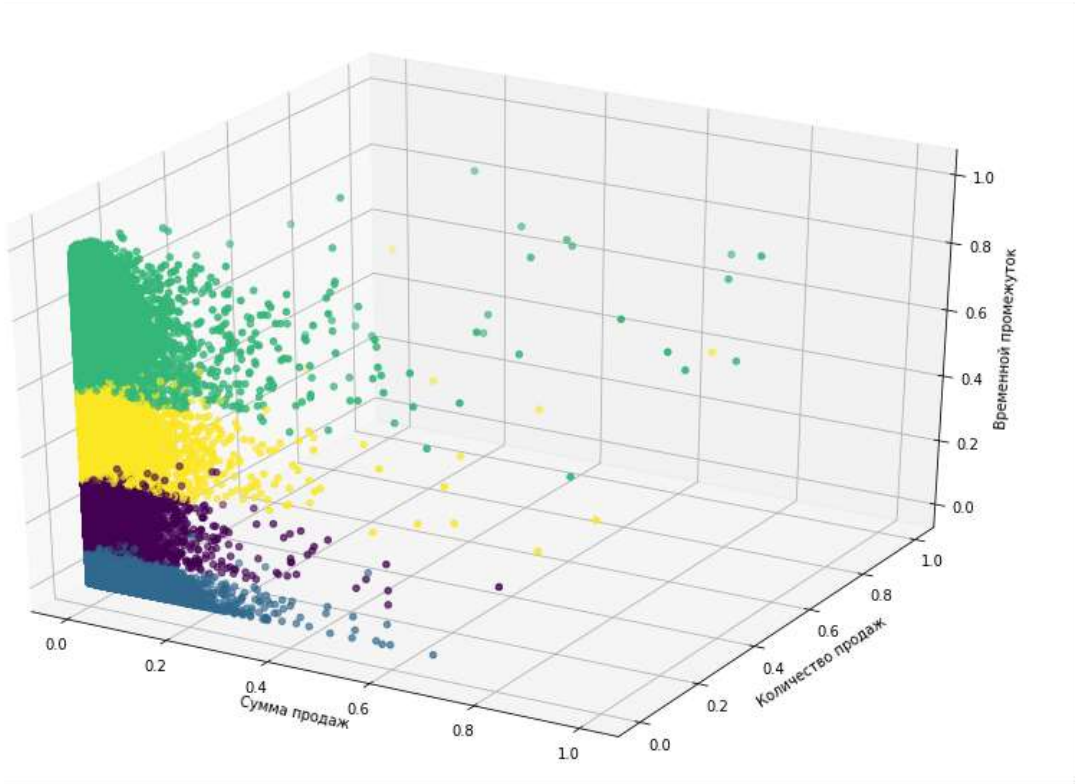

Рисунок 18 – Распределение данных по 4-м кластерам

Таблица 3 – Статистика для кластеризации

	1 (синий кластер)	2 (фиолетовый кластер)	3 (желтый кластер)	4 (зеленый кластер)
Средняя сумма покупок	19695.60	41479.52	53488.46	79535.06
Среднее количество посещенных дней	5.36	189.57	416.88	708.59
Среднее количество совершенных покупок	1.33	3.02	3.57	4.37

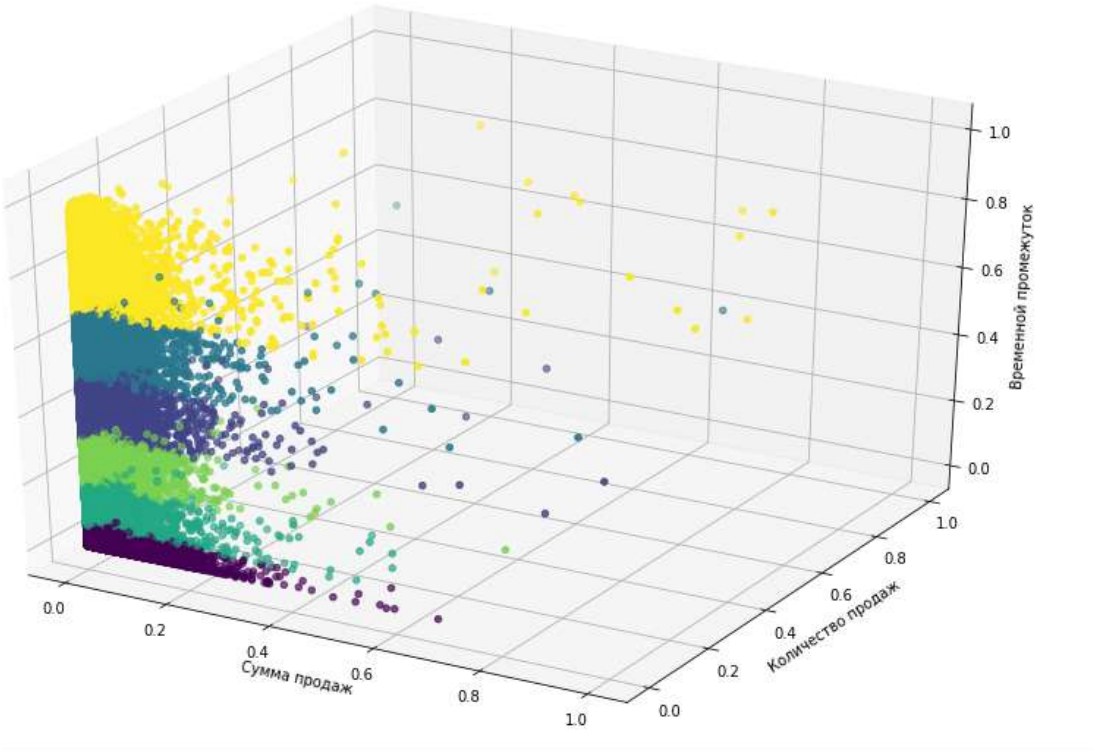

Рисунок 19 – Распределение данных по 6-ти кластерам

Таблица 4 – Время, затраченное на кластеризацию методом K – means

Количество кластеров	3	4	5	6
Время кластеризации (сек)	11.173	14.339	127	163

2.4. Результаты обработки C - means

Для кластеризации методом C means используется библиотека scikit-fuzzy. Этапы подготовки данных к кластеризации методом C means ничем не отличаются от метода K means. Поэтому можно сразу перейти к результатам кластеризации.

Результаты работы метода:

Таблица 5 – Время, затраченное на кластеризацию методом C means

Количество кластеров	3	4	5	6
Время кластеризации (сек)	15	28	32	59

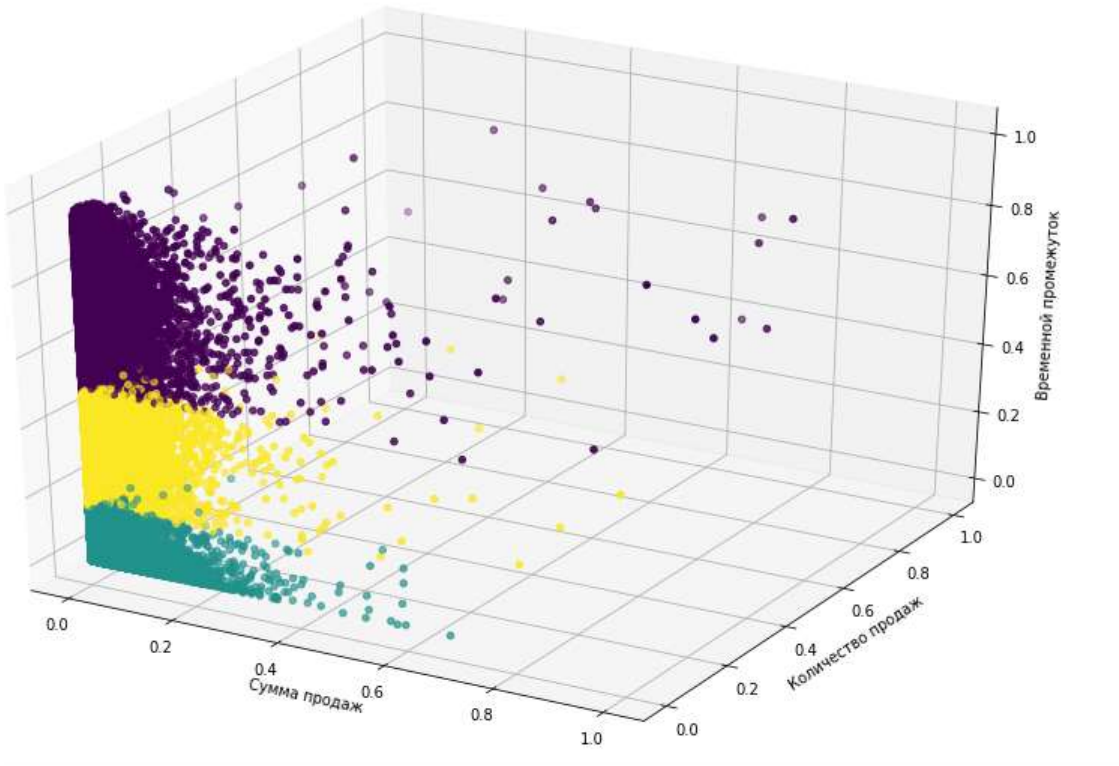

Рисунок 20 – Распределение данных по 3-м кластерам

Таблица 6 – Статистика для кластеризации

	1 (зеленый кластер)	2 (желтый кластер)	3 (фиолетовый кластер)
Средняя сумма покупок	20357.39	46723.67	73855.09
Среднее количество посещенных дней	9.40	301.33	649.39
Среднее количество совершенных покупок	1.39	3.28	4.21

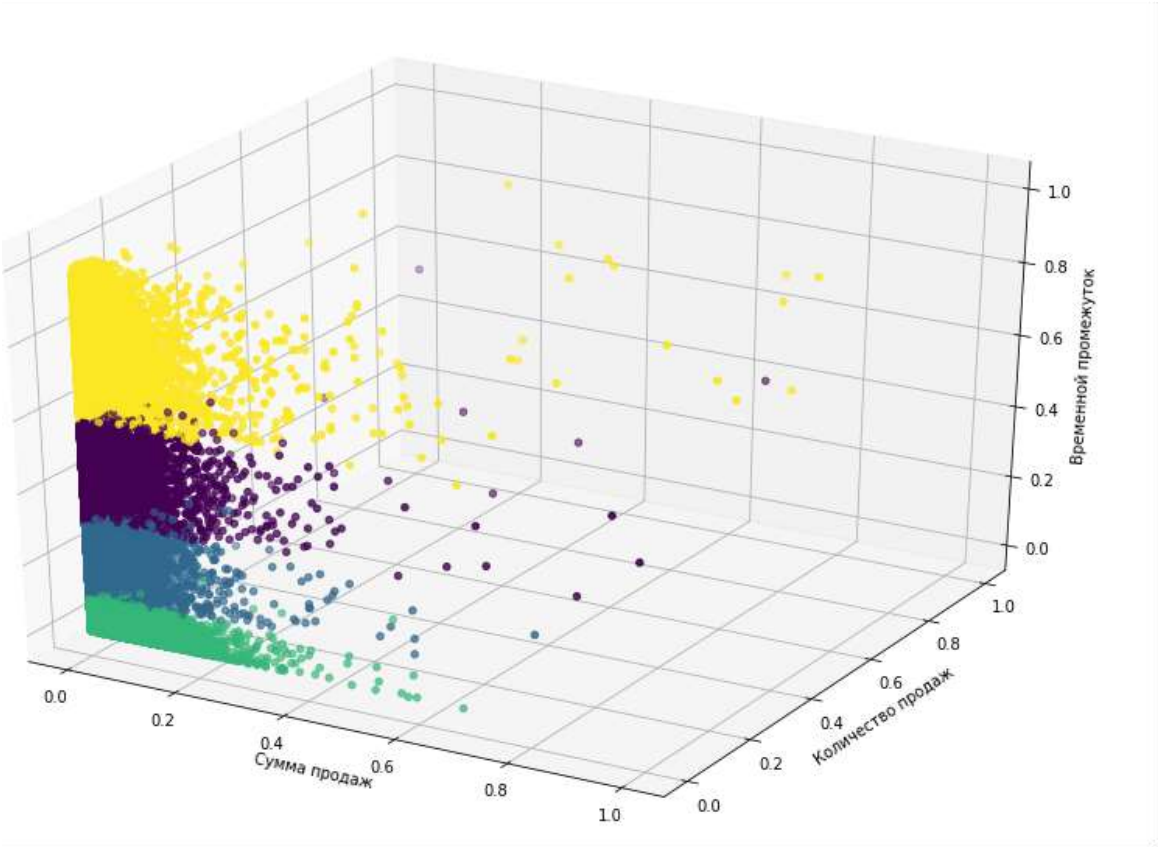

Рисунок 21 – Распределение данных по 4-м кластерам

Таблица 7 – Статистика для кластеризации

	1 (зеленый кластер)	2 (синий кластер)	3 (фиолетовый кластер)	4 (желтый кластер)
Средняя сумма покупок	19647.02	41188.14	53538.98	79620.77
Среднее количество посещенных дней	5.09	185.08	415.42	710.35
Среднее количество совершенных покупок	1.33	3.01	3.57	4.38

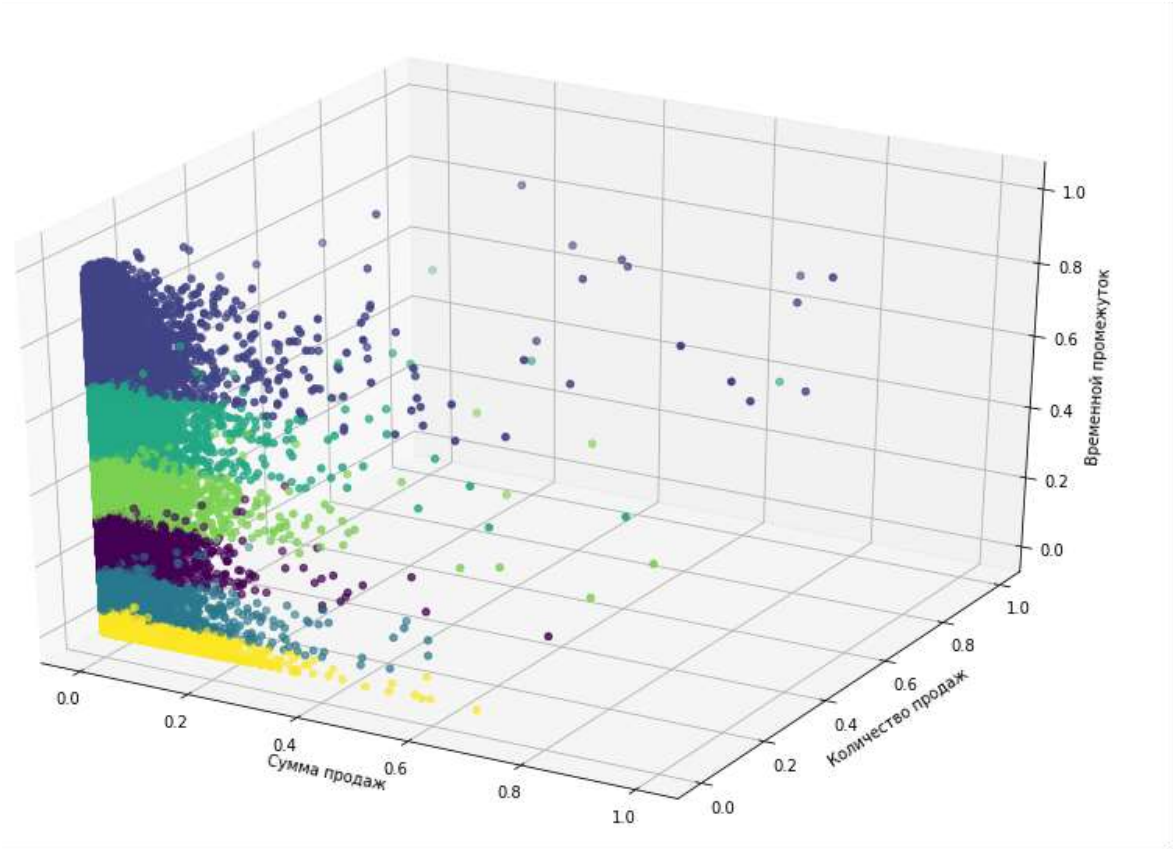

Рисунок 22 – Распределение данных по 6-ти кластерам

2.5. Результаты обработки BIRCH

Для начала рассмотрим результат кластеризации со стандартными настройками. Это:

Параметр threshold=0.5

Параметр n_clusters=3

И сразу же попытка кластеризовать с параметром threshold значение которого было 0.5 и выше вызывает следующее сообщение:

«ConvergenceWarning: Number of subclusters found (1) by Birch is less than (3). Decrease the threshold.»

Параметр «threshold». отвечает за максимальный радиус подкластера. Уменьшение этого параметра ведет к большему образованию новых подкластеров и наоборот. Как видно из ошибки выше большое значение привело к тому, что было найдено всего 2 кластера, тогда как мы искали 3. Уменьшим значение threshold до 0.15.

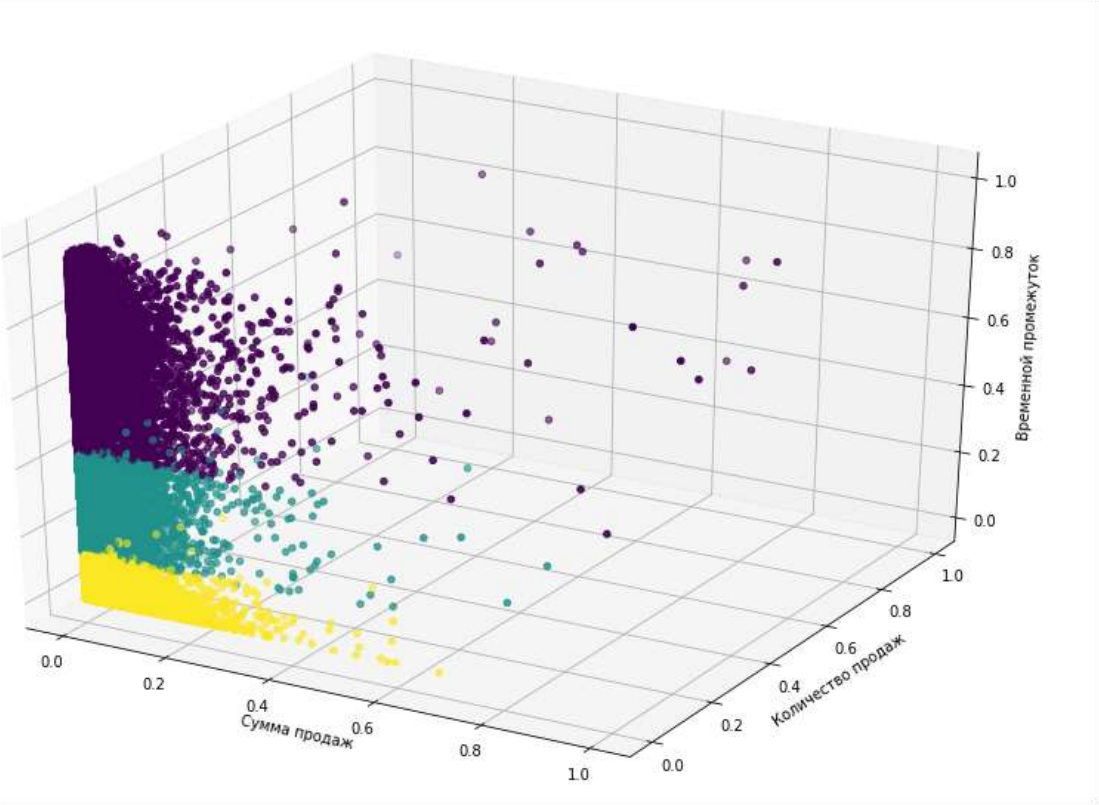

Рисунок 23 – Кластеризация Birch с параметром threshold 0.15

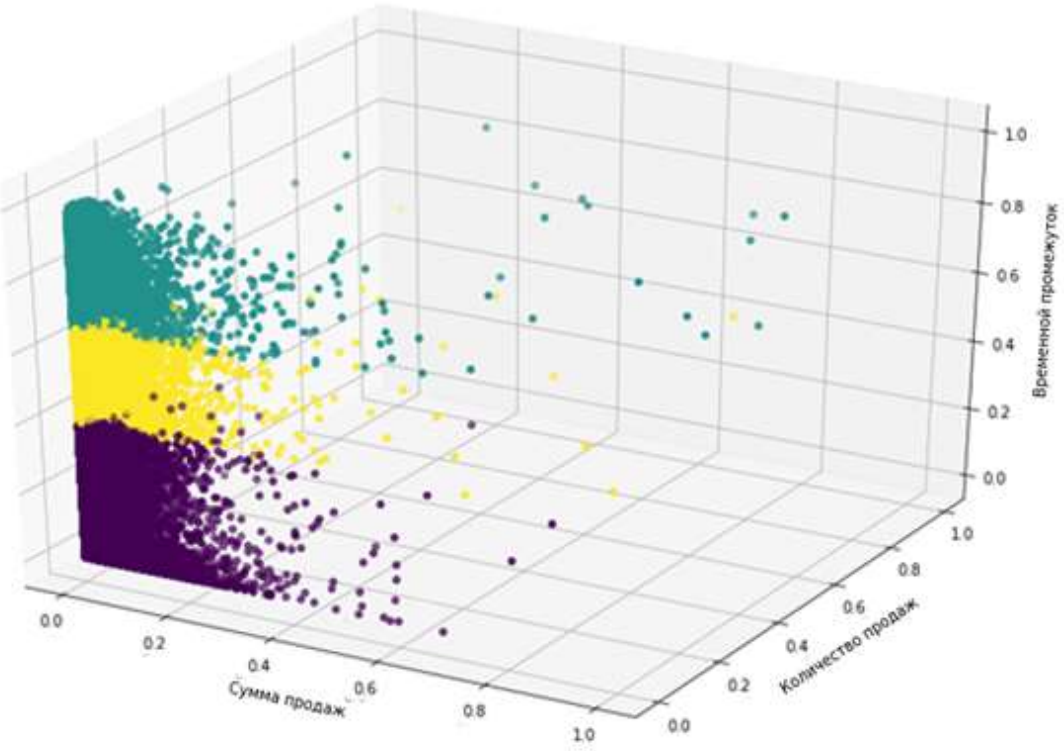

Рисунок 24 – Кластеризация Birch с параметром threshold 0.12

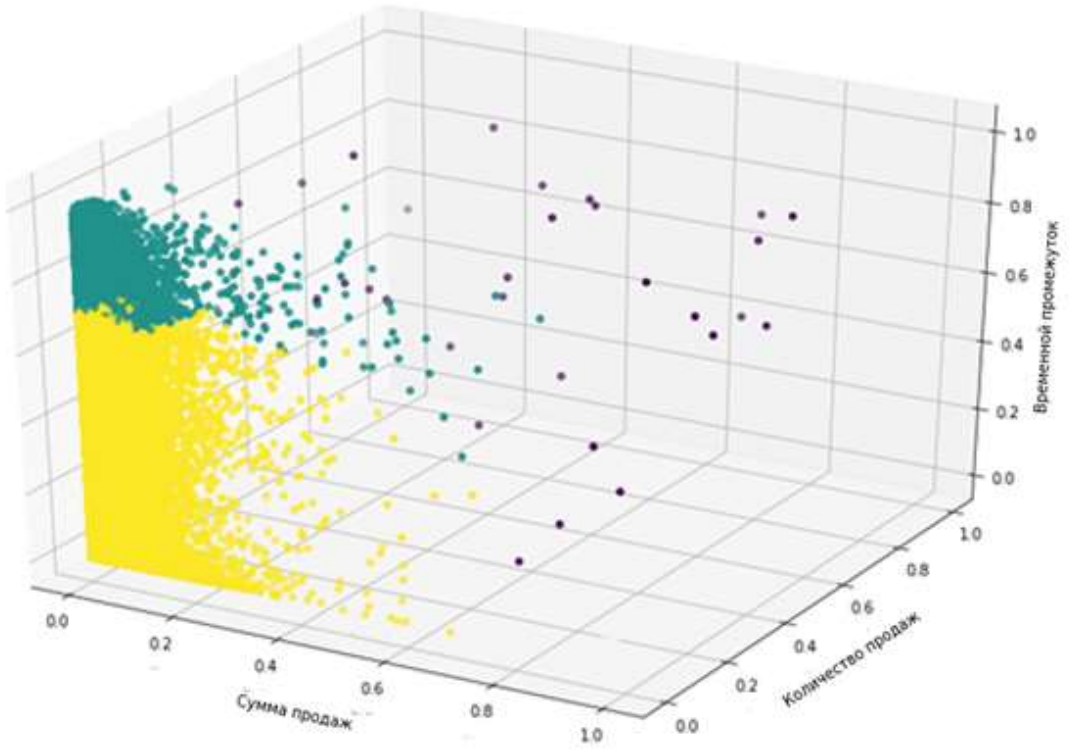

Рисунок 25 – Кластеризация Birch с параметром threshold 0.1

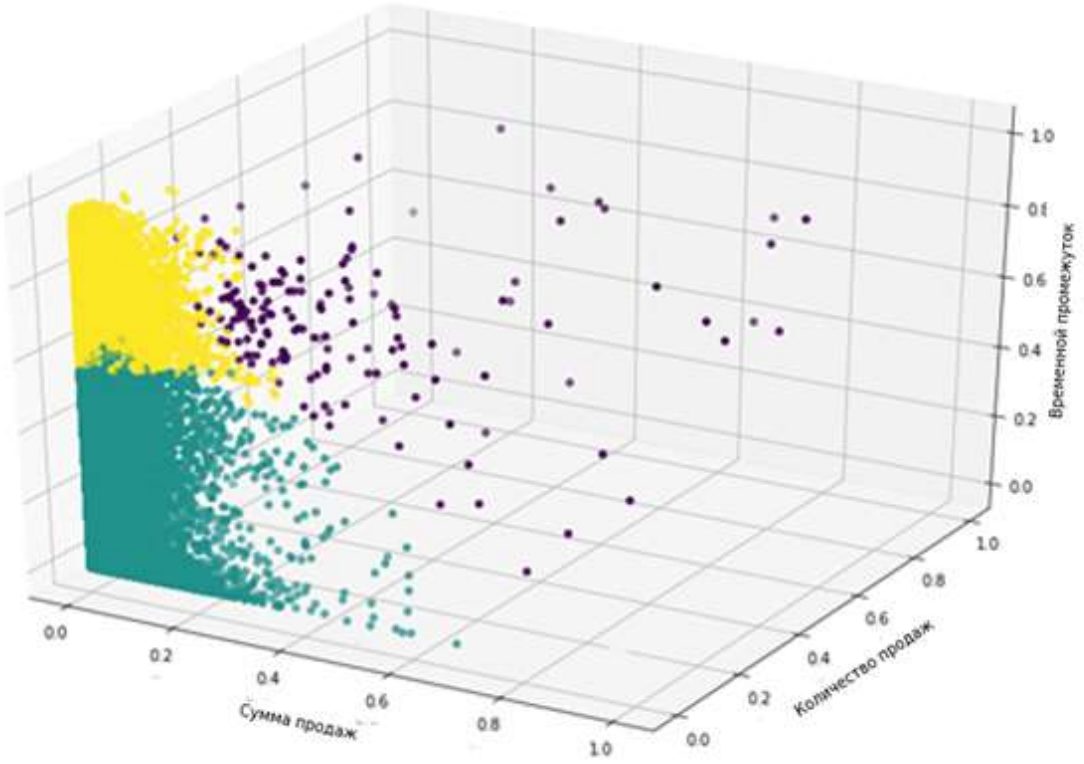

Рисунок 26 – Кластеризация Birch с параметром threshold 0.05

Как видно на рисунках выше при параметре threshold от 0.15 - 0.12 кластеры различаются друг от друга значением параметра «Временной промежуток». Тогда как при параметре threshold от 0.1 - 0.05 у одного из кластеров центройд смещен еще и по оставшимся двум осям. Какой можно сделать из этого вывод?

Каждая точка на графике отображает отдельного клиента. Алгоритм кластеризации на рисунках 23,24 разделил клиентов по тому, как долго они посещают магазин. С рисунками 25,26 другой результат.

Можно изменить количество искомых кластеров в параметре n_clusters. Получается вполне ожидаемый результат.

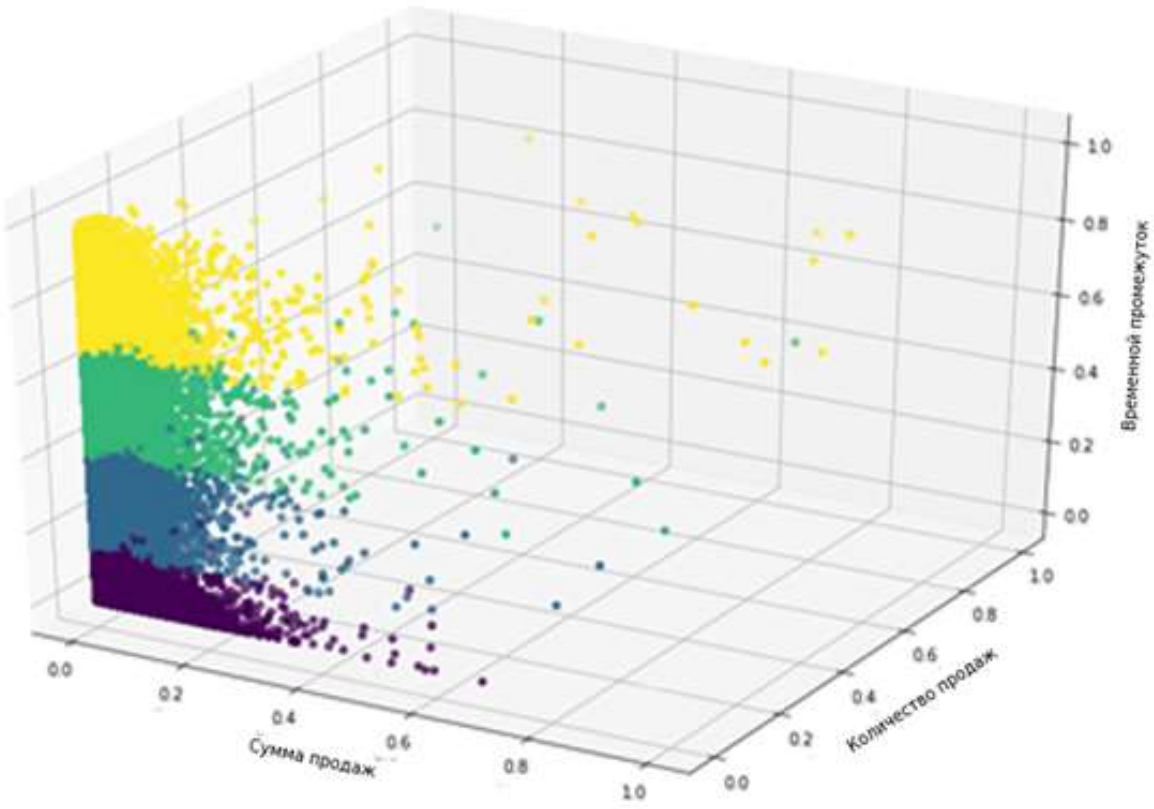

Рисунок 27 – Кластеризация Birch с параметром threshold 0.12 n_clusters=4

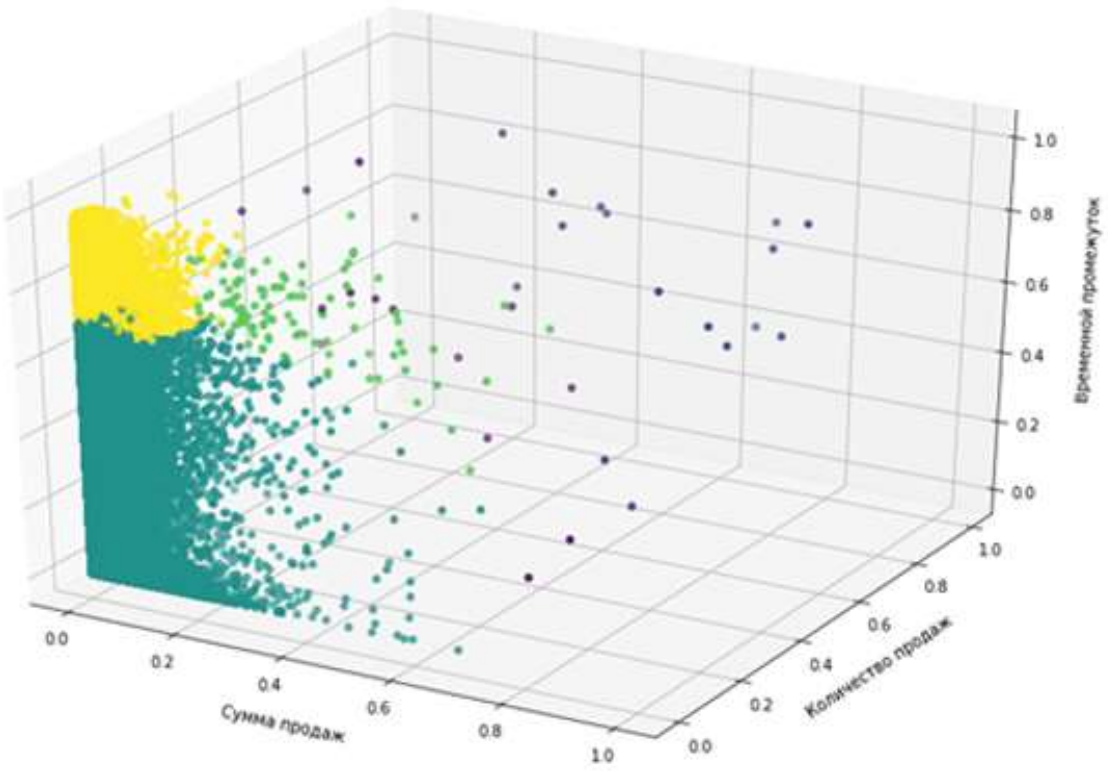

Рисунок 28 – Кластеризация Birch с параметром threshold 0.10

n_clusters=5

Остановимся на результате на рисунках 24 и 26. Посмотрим, как различаются кластеры друг от друга

Таблица 8 – Статистика для рисунка 24

	1 (фиолетовый кластер)	2 (зеленый кластер)	3 (желтый кластер)
Средняя сумма покупок	24618.41	86051.73	4231206.83
Среднее количество посещенных дней	655.93	783.18	59.08
Среднее количество совершенных покупок	120.38	4.60	1.67

Таблица 9 – Статистика для рисунка 26

	1 (желтый кластер)	2 (зеленый кластер)	3 (синий кластер)	3 (фиолетовый кластер)
Средняя сумма покупок	83568.15	43537.30	59697.65	20044.61
Среднее количество посещенных дней	755.89	242.64	488.89	7.318
Среднее количество совершенных покупок	4.52	3.14	3.77	1.36

Таблица 10 – Время затраченное на кластеризацию

threshold	0.15	0.1	0.05	0.12	0.10
n_clusters	3	3	3	4	5
Время, затраченное на кластеризацию (сек.)	26	26	45	26	27

2.6. Результаты обработки HDBSCAN

Для начала рассмотрим результат кластеризации со стандартными настройками. Это:

Параметр min_cluster_size=5

Параметр min_samples=None

Параметр cluster_selection_epsilon=0.0

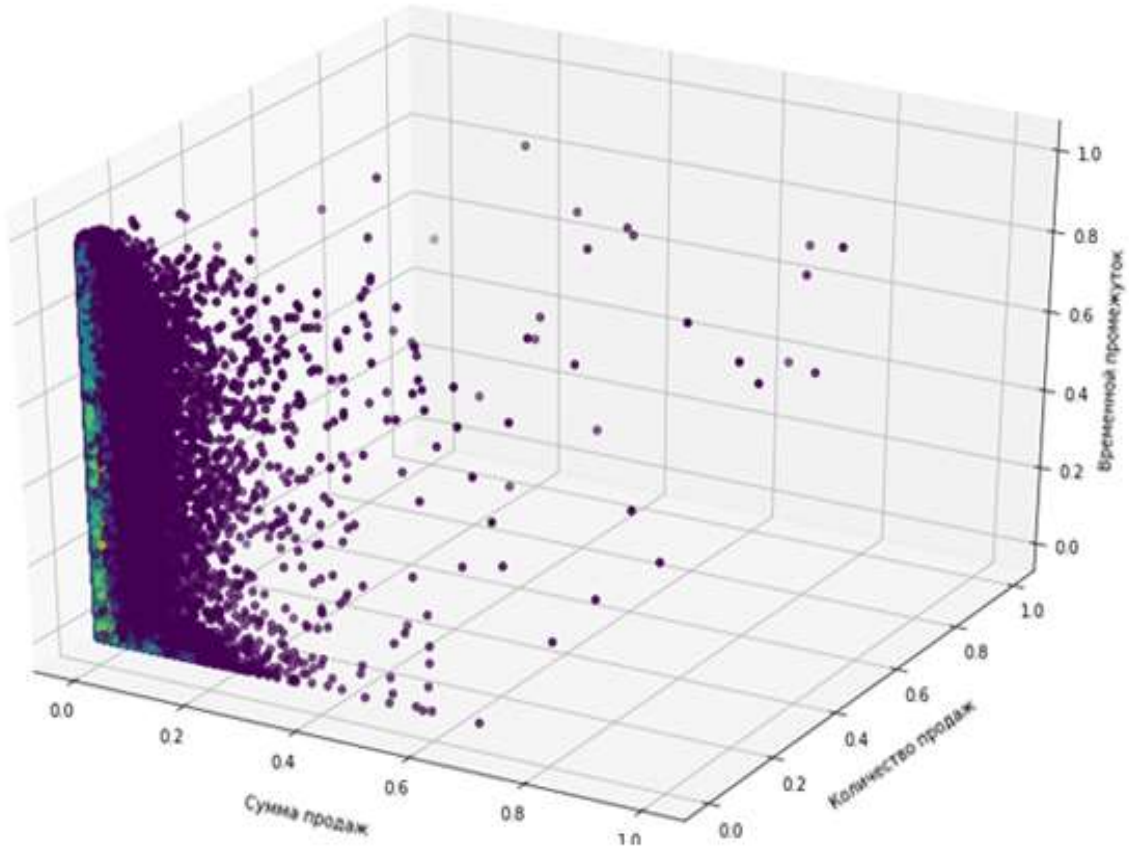

Рисунок 29 – Кластеризация HDBSCAN со стандартными параметрами

Результаты кластеризации с такими настройками тяжело назвать пригодными. Всего было найдено 57183 различных кластеров. В таком случае можно изменить параметр «cluster_selection_epsilon». Этот параметр влияет на то будут ли разделены кластеры в определённом диапазоне. Например, если установить значение 0,5, то кластеры, которые находятся на расстоянии менее 0,5 единиц буду объединены.

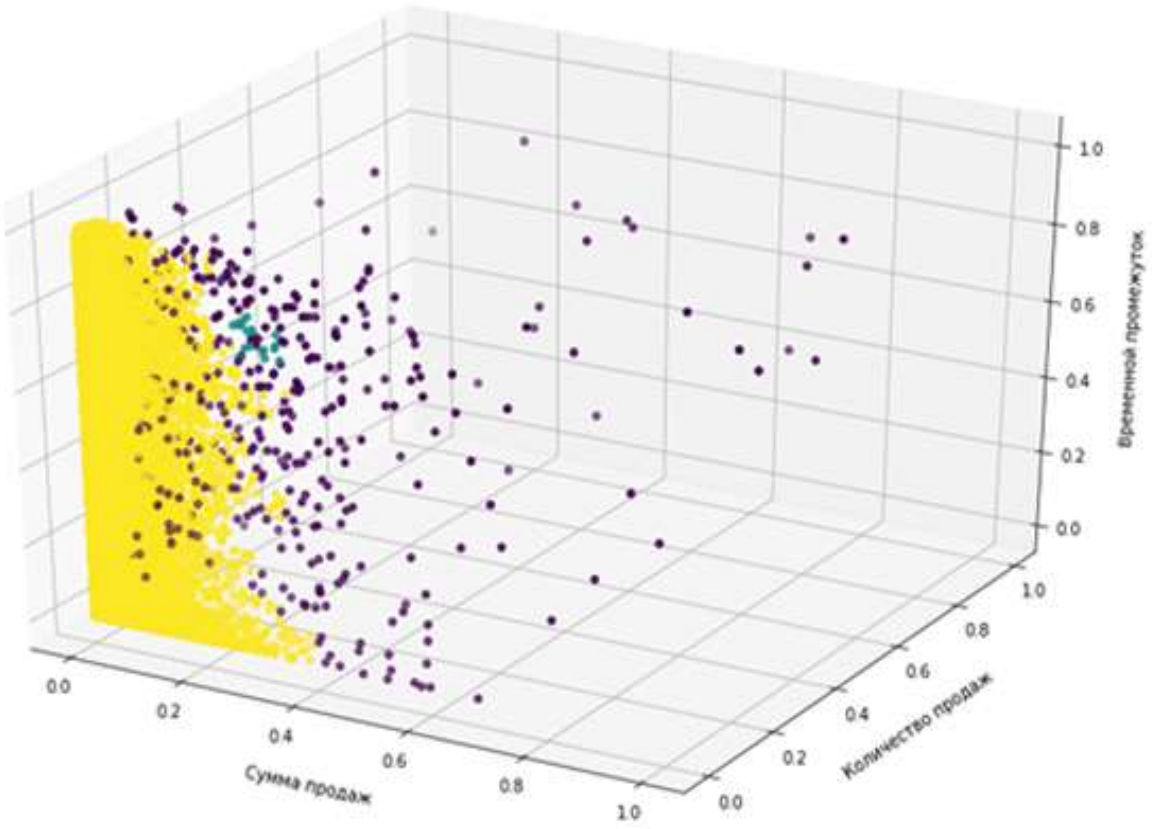

Рисунок 30 – Кластеризация HDBSCAN с параметром cluster_selection_epsilon=0.1

Как видно из рисунка 30 результат изменился если сравнивать с предыдущей кластеризацией. Однако много точек было помечено как выброс (фиолетовый цвет). Возможно было выбрано слишком большое значение cluster_selection_epsilon.

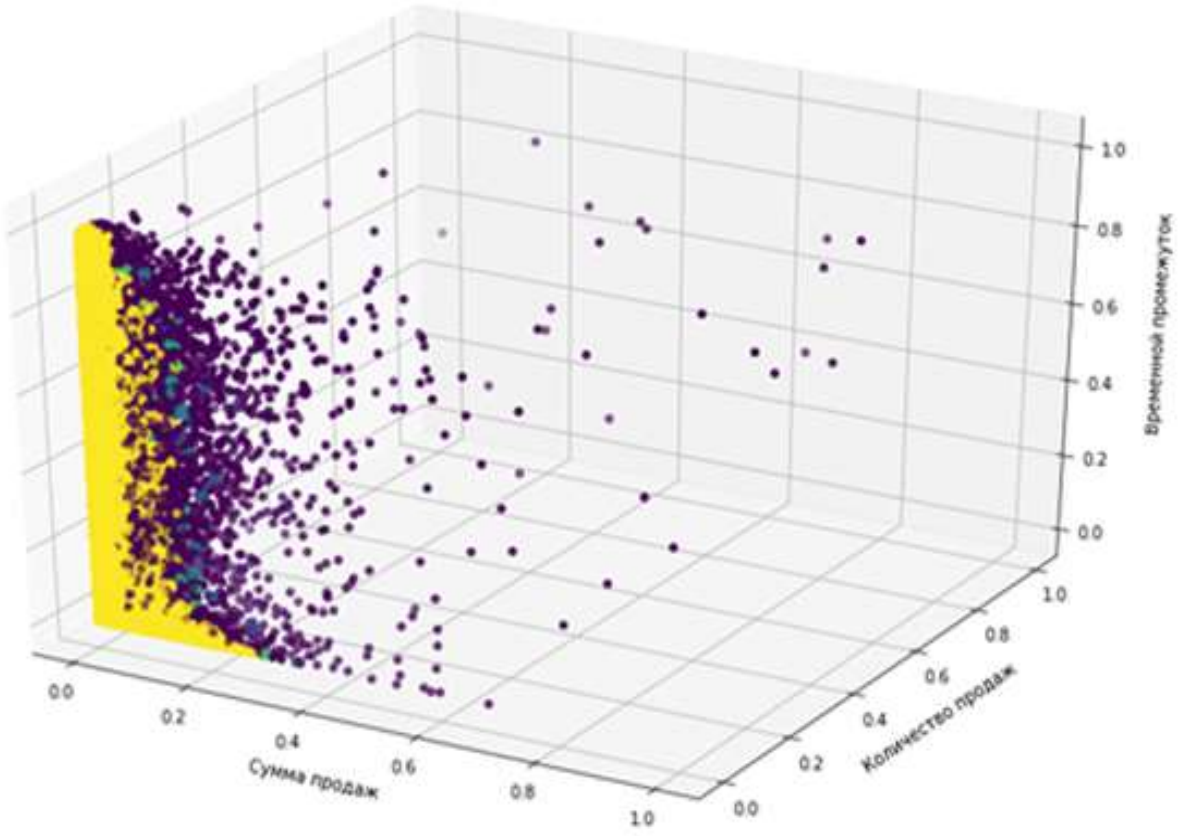

Рисунок 31 – Кластеризация HDBSCAN с параметром cluster_selection_epsilon=0.01

На рисунке 31 в общей сложности 44 кластера. Большинство кластеров содержит не более 30 точек. Можно попытается объединить мелкие кластеры установив лимит размера кластера через параметр min_cluster_size.

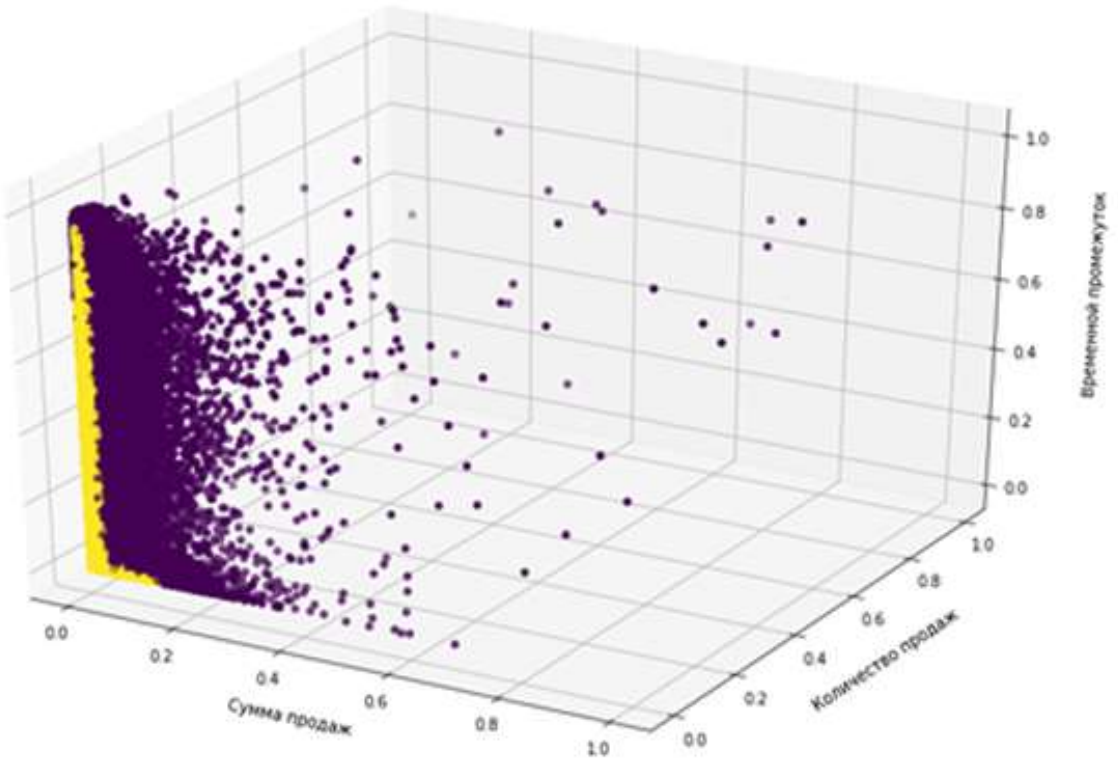

Рисунок 31 – Кластеризация HDBSCAN с параметром cluster_selection_epsilon=0.1 и min_cluster_size=30

Однако изменение параметра min_cluster_size привело к тому, что снова большая часть данных была помечена как выброс. Дальнейшее изменение параметров не привело к какому-либо удовлетворительному результату.

2.7. Вывод по практический части

В данной части была проведена подготовка данных к кластеризации и кластеризация. Этап подготовки данных выявил множество ошибочных записей, которые решено было удалить. Следующим этапом стала кластеризация и сравнение результатов. Методы C - means и K - means показали практический одинаковые результаты и говорить о том какой именно метод лучше подойдет для текущего набора данных тяжело. Метод BIRCH же дает более обширную вариативность результатов. Изменяя параметр threshold можно добиться схожих результатов что и в методах C - means и K – means. При этом манипулируя тем же параметром можно получить кластеры не только разделенные временным промежутком, т.е. не только разделяя клиентов по тому, как давно они посещают магазин, но и еще по тому, как много тратят и покупают в магазине. Увы метод HDBSCAN не привел к каким-либо полезным результатам. HDBSCAN больше нацелен на поиск отделяемых «островов» тогда как наши данные представляют собой один большой «остров». Исходя из полученных результатов можно сделать вывод что наиболее подходящим методом кластеризации для данного набора данных является метод BIRCH.

ЗАКЛЮЧЕНИЕ

В рамках данной работы был произведен обзор существующих методов кластерного анализа. Были рассмотрены категории методов кластеризации представлены их положительные и отрицательные стороны и типичные представители. Также рассмотрен сам процесс кластеризации расписаны и из каких шагов он состоит. В практической части было проведено сравнение нескольких методов кластеризации по результатам работы и был выбран более подходящий метод.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. G.J. Mclachlan Cluster analysis and related techniques in medical research // Statistical Methods in Medical Research - 1992 27-48p
2. G. Punj Cluster analysis in marketing research: Review and suggestions for application 1983 // Journal of Marketing Research - Vol. 20, No. 2 May, 1983, pp. 134-148
3. S. Panda Comparing fuzzy-C Means and K-Means clustering techniques: a comprehensive study/ S. Panda, S. Sahu, P.Jena, S. Chattopadhyay - Advances in Computer Science, Eng. & Appl., AISC 166, pp. 451–460
4. Е.С. Подвальный Сравнение алгоритмов кластерного анализа на случайном наборе данных / Е.С. Подвальный, А.В. Плотников
5. O.M. Abu Abbas Comparisons between data clustering algorithms // International Arab Journal of Information Technology 2008, pp 320-325
6. G. Sehgal Comparison of various clustering algorithms/G. Sehgal, K. Garg // International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014, pp 3074-3076
7. MacQueen, J. Some methods for classification and analysis of multivariate observations/ J. MacQueen // In Proc. 5th Berkeley Symp. On Math. Statistics and Probability, 1967. -С.281-297.
8. Kaufman, L. Clustering by means of Medoids, in Statistical Data Analysis Based on the l–Norm and Related Methods / L. Kaufman, P.J. Rousseeuw, Y. Dodge, 1987. -С.405-416.
9. Zhang, T. BIRCH: An Efficient Data Clustering Method for Very Large Databases / T. Zhang, R. Ramakrishnan, M. Linvy // In Proc. ACM SIGMOD Int. Conf. on Management of Data. ACM Press, New York, 1996. -С.103-114.
10. Karypis, G. CHAMELEON: A Hierarchical Clustering Algorithm Using Dynamic Modeling / G. Karypis, E.-H. Han, V. Kumar // Journal Computer Volume 32 Issue 8. IEEE Computer Society Press Los Alamitos, CA, 1999. -pp 68-75

11. Ester, M. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise / M. Ester, H.-P. Kriegel, J. Sander, X. Xu // In Proc. ACM SIGMOD Int. Conf. on Management of Data, Portland, OR, 1996. –C. 226-231.

12. Agrawal, R. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications / R. Agrawal, J. Gehrke, D. Gunopulos, P. Raghavan // In Proc. ACM SIGMOD Int. Conf. on Management of Data, Seattle, Washington, 1998. -C.94-105.

13. Demster, A. Maximum Likelihood from Incomplete Data via the EM Algorithm /A.P. Demster, N.M. Laird, D.B. Rubin //JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B, Vol. 39, No. 1, 1977. -C.1-38.

14. L. Dalton Clustering Algorithms: On Learning, Validation, Performance, and Applications to Genomics/ L. Dalton, V. Ballarin, M. Brun// Curr Genomics. 2009 pp 430–445

15. Halkidi M On clustering validation techniques/ Halkidi M, Batistakis Y, Vazirgiannis M // J. Intell. Inf. Syst. 2001 pp 107–145

16. J. McCaffrey Clustering Non-Numeric Data Using Python [Электронный документ] https://visualstudiomagazine.com/articles/2018/04/01/clustering-non-numeric-data.aspx Проверено 05.05.2020

17. David Arthur & Sergei Vassilvitskii. How Slow is the k-means Method? // Proceedings of the 2006 Symposium on Computational Geometry (SoCG). 2006 pp 10

18. Федин, Ф. О. Анализ данных. Часть 2. Инструменты Data Mining [Электронный ресурс]: учеб. пособие / Ф. О. Федин, Ф. Ф. Федин. – Москва: Московский городской педагогический университет, 2012. – 308 с.

19. Shi Na Research on k-means clustering algorithm / Shi Na, Liu Xumin, Guan yong //2010 Third International Symposium on Intelligent Information Technology and Security Informatics 2010, pp 63-67

20. Md. Sohrab Mahmud Improvement of K-means clustering algorithm with better initial centroids based on weighted average /Md. Sohrab Mahmud, Md.

Mostafizer Rahman, Md. Nasim Akhtar// 7th International Conference on Electrical and Computer Engineering 2012, pp 647 -650

21. Min Huang Improved K-means clustering center selecting algorithm/ Min Huang, Lei Yu, Ying Chen// Information Engineering and Applications 2012 pp 373-379

22. Madhu Yedla Enhancing, K-means Clustering Algorithm with Improved Initial Center/ Madhu Yedla, Srinivasa Rao Pathakota, T.M Srinivasa //International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 1 (2) 2010, pp 121-125.

23. Dunn J.C. A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters // Journal of Cybernetics. — 1973, pp 32–57

24. Bezdek, James C. Pattern Recognition with Fuzzy Objective Function Algorithms. 1981

25. Rao V.S. Comparative investigations and performance analysis of FCM and MFPCM algorithms on /Vuda Sreenivasa Rao. // Indian Journal of Computer Science and Engineering Vol 1 No 2, 145-151

26. X. Yan Power System Identification and adjustment of Bad Data Based on Data Mining. / X. Yan, L. Yan hong // JOURNAL OF ANHUIELECTRICAL ENGINEERING PROFESSIONAL TECHNIQUE college. VOL15, NO 3，pp 11-15

27. Liu YP Improvement and Optimization of a Fuzzy C-Means Clustering Algorithm. Systems Engineering and Electronics/ Shen Yi, Liu ZY//. Vol·22, No·4 2000，pp 1-3

28. Wu KL Alternative C-means clustering algorithms/ Wu KL, Yang MS//Pattern Recognition, 2002. pp 2267-227

29. L. McInnes, J. Healy, S. Astels, hdbscan: Hierarchical density based clustering In: Journal of Open Source Software, The Open Journal, volume 2, number 11. 2017

30. C. Dharni M An improvement of DBSCAN Algorithm to analyze cluster for large datasets / C. Dharni M. Bnasal, // 2013 IEEE International Conference in MOOC, Innovation and Technology in Education (MITE), Jaipur, 2013, pp. 42-46
31. Chowdhury Nirmalya Using an MST based Value for ε in DBSCAN Algorithm for Obtaining Better Result. / Chowdhury Nirmalya and Preetha Bhattacharjee // International Journal of Information Technology and Computer Science 6 (2014): 55-60
32. Duan, L A Local Density Based Spatial Clustering Algorithm with Noise / Duan, L., Xiong, D., Lee, J.H., & Guo, F. // IEEE International Conference on Systems, Man and Cybernetics, 5, 4061-4066 (2006)
33. Ahmed, K.N An Overview of Various Improvements of DBSCAN Algorithm in Clustering Spatial Databases / Ahmed K.N., Razak T.A.// [Электронный документ] https://www.semanticscholar.org/paper/An-Overview-of-Various-Improvements-of-DBSCAN-in-Ahmed-Razak/9d3b112276536dd29d9af0873db58a5f3f08b65c [Проверено 01.05.2020
34. Tian Z. BIRCH: an efficient data clustering method for very large databases. /Tian Zhang, Raghu Ramakrishnan, and Miron Livny// In Proceedings of the 1996 ACM SIGMOD international conference on Management of data (SIGMOD '96). Association for Computing Machinery, New York, NY, USA, 103–114.
35. Lorbeer B Variations on the Clustering Algorithm BIRCH. Big Data Research. / Lorbeer Boris, Kosareva Ana, Deva Bersant, Softić Dženan, Ruppel Peter, Küpper Axel // Journal of Cybernetics. — 1973, pp 32–57
36. Jiang S. Improved BIRCH clustering algorithm: Improved BIRCH clustering algorithm. / Jiang Shengyi, LI Xia // Journal of Computer Applications. 29. 293-296. 10.3724/SP.J.1087.2009.00293.
37. Ismael N Improved Multi Threshold Birch Clustering Algorithm/ Ismael Nidal Alzaalan Mahmoud, Ashour Wesam. // International Journal of Artificial Intelligence and Applications for Smart Devices. 2. 1-10. 10.14257/ijaiasd.2014.2.1.01.

38. Dongwei Guo LBIRCH: An Improved BIRCH Algorithm Based on Link/ Dongwei Guo, Jingwen Chen, Yingjie Chen, Zhiyu Li. // In Proceedings of the 2018 10th International Conference on Machine Learning and Computing (ICMLC 2018). Association for Computing Machinery, New York, NY, USA, 74–78.

39. H. Du An Improved BIRCH Clustering Algorithm and Application in Thermal Power /H. Du, Y. Li //2010 International Conference on Web Information Systems and Mining, Sanya, 2010, pp. 53-56,

40. Release History - 0.21.0 documentation scikit-learn. [Электронный документ] https://scikit-learn.org/stable/whats_new.html#version-0-21-0 Проверено 24.05.2019

41. Kaufman, L. Clustering by means of Medoids, in Statistical Data Analysis Based on the l–Norm and Related Methods / L. Kaufman, P.J. Rousseeuw, Y. Dodge, 1987. -C.405-416.

ПРИЛОЖЕНИЕ А

Листинг программы

```
#%tensorflow_version 1.x magic:
import matplotlib.pylab as plt
import numpy as np
import pandas as pd
import math
import tensorflow as tf
from tensorflow import keras
from google.colab import drive
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
import sklearn
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error
from datetime import datetime
drive.mount('/content/drive')
data1 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/BKP/data1.csv',sep=',')#[::-1]
data2 = pd.read_csv('/content/drive/My Drive/Colab Notebooks/BKP/data2.csv',sep=',')#[::-1]
data1.head()
allCleanData = data1
allCleanData=allCleanData.append(data2, ignore_index=True)
allCleanData.head()
data2.head()
scaler = MinMaxScaler(feature_range=(0, 1))


DDnp=np.asarray(allCleanData['DataGap'])
```

```
DDnp=DDnp.reshape(-1,1)
allCleanData['DataScaled']= scaler.fit_transform(DDnp)


DDnp=np.asarray(allCleanData['Сумма продаж'])
DDnp=DDnp.reshape(-1,1)
allCleanData['SummSellsScaled']= scaler.fit_transform(DDnp)


DDnp=np.asarray(allCleanData['Количество продаж '])
DDnp=DDnp.reshape(-1,1)
allCleanData['CountSellsScaled']= scaler.fit_transform(DDnp)
allCleanData.head()
result =
np.transpose((allCleanData['SummSellsScaled'].values,allCleanData['CountSellsSc
aled'].values,allCleanData['DataScaled'].values))
import hdbscan
startTime=datetime.now()
model = hdbscan.HDBSCAN(cluster_selection_epsilon=0.05)
all_predictions3 = model.fit(result)
print(datetime.now()- startTime)
fig = plt.figure()
ax = fig.gca(projection='3d')
plt.gcf().set_size_inches(15, 10)
ax.scatter( allCleanData['SummSellsScaled'],
allCleanData['CountSellsScaled'],allCleanData['DataScaled'],c=all_predictions3.la
bels_)
from sklearn.cluster import  Birch
model = Birch(threshold=0.12,branching_factor=50,n_clusters=4)
result =
np.transpose((allCleanData['SummSellsScaled'].values,allCleanData['CountSellsSc
aled'].values,allCleanData['DataScaled'].values))
all_predictions = model.fit_predict(result)
fig = plt.figure()
```

```
ax = fig.gca(projection='3d')

plt.gcf().set_size_inches(15, 10)

ax.scatter( allCleanData['SummSellsScaled'],
allCleanData['CountSellsScaled'],allCleanData['DataScaled'],c=all_predictions)
```