Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика, искусственный интеллект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Метод разбиения категориальных данных на основе агломеративного подхода иерархической кластеризации»

Студент ИУ7-83Б _____ К. Э. Ковалец
(Группа) (Подпись, дата) (И. О. Фамилия)

Руководитель ВКР _____ Н. В. Новик
(Подпись, дата) (И. О. Фамилия)

Нормоконтролер _____ Д. Ю. Мальцева
(Подпись, дата) (И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно–пояснительная записка к выпускной квалификационной работе содержит 78 страниц, 15 иллюстраций, 5 таблиц, 30 источников, 3 приложения.

В данной работе представлена разработка метода разбиения категориальных данных на основе агломеративного подхода иерархической кластеризации.

Описаны существующие методы кластеризации данных и проведено их сравнение по выделенным критериям. Описаны существующие критерии связи кластеров в иерархическом методе разбиения данных. Рассмотрены существующие меры расстояний между объектами и проведено их сравнение. Описаны методы оценки качества кластеризации. Разработан метод разбиения данных на основе агломеративного подхода иерархической кластеризации. Разработано программное обеспечение для демонстрации работы созданного метода. Проведено сравнение разработанного метода разбиения с аналогами с помощью существующих методов оценки качества кластеризации.

Ключеые слова: кластеризация, кластер, разбиение, иерархический метод, метод k-прототипов, категориальные данные, расстояние Говера, полная связь, метод локтя, метод оценки силуэтов.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Разбиение больших массивов категориальных данных является актуальной задачей в системах анализа данных в различных областях, таких как маркетинг, медицина, наука и другие. Цель кластеризации категориальных данных состоит в обнаружении скрытых закономерностей и разделении данных на группы, которые имеют схожие характеристики. Разбиение может быть использовано для ряда задач, таких как сегментация клиентской базы, предсказание потребительского поведения, снижение рисков, анализ и классификация текстовых документов.

В отличие от числовых данных, для категориальных не существует естественной метрики, которая может быть использована в качестве меры расстояния между объектами. Это делает кластеризацию категориальных данных более сложной задачей по сравнению с кластеризацией числовых данных.

Целью выпускной квалификационной работы является разработка метода разбиения категориальных данных на основе агломеративного подхода иерархической кластеризации.

Для достижения поставленной цели необходимо выполнить следующие задачи:

— описать существующие методы кластеризации данных и сравнить их по выделенным критериям;

— описать существующие критерии связи кластеров в иерархическом методе разбиения данных;

— рассмотреть существующие меры расстояний между объектами и провести их сравнение;

— описать методы оценки качества кластеризации;

— разработать метод разбиения данных на основе агломеративного подхода иерархической кластеризации;

— разработать программное обеспечение для демонстрации работы созданного метода;

— провести сравнение разработанного метода разбиения с аналогами с помощью существующих методов оценки качества кластеризации.

1 Аналитический раздел

1.1 Описание предметной области

Кластеризация [1] — это алгоритмический процесс разбиения набора данных на кластеры, состоящие из объектов, схожих между собой. При этом объекты из различных кластеров должны быть как можно более отличными друг от друга. Ключевой разницей кластеризации от классификации является отсутствие явно заданного набора групп, которое определяется в процессе выполнения алгоритма.

В кластеризации используются различные типы данных, которые могут быть разделены на две основные категории: количественные и категориальные. В зависимости от типа данных, могут быть использованы различные методы кластеризации.

Категориальные данные [2] — это набор данных, который хранит качественную информацию, закодированную в виде категорий. Категории не являются числами, а представляют собой описательные значения, такие как цвет, пол, марка автомобиля, рейтинг продукта и так далее.

Количественные данные [2] — это числовые данные, которые могут быть измерены и упорядочены. К ним относятся такие переменные, как возраст, рост, вес, расстояние, цена. Эти данные могут быть представлены как в дискретной, так и в непрерывной форме.

Сам процесс кластеризации состоит из трех шагов:

— Построение матрицы несходства — это важный этап в кластеризации, в котором определяется, насколько объекты отличаются друг от друга с помощью выбранной меры близости.

— Выбор метода кластеризации — определение подхода, который будет использоваться для группировки объектов на основе их сходства.

— Оценка и интерпретация — оценка полученных результатов и их интерпретация в соответствии с поставленной задачей.

Каждый из этих шагов важен и может повлиять на результат кластеризации. Точность и эффективность разбиения зависят от правильного выбора метода кластеризации и меры близости при построении матрицы несходства.

1.2 Методы разбиения

1.2.1 Иерархический метод

Иерархическая кластеризация [3] — это метод разбиения данных, который создает вложенные кластеры путем их последовательного слияния или разделения. Каждый шаг приводит к увеличению кластеров или объединению уже существующих.

Процесс иерархической кластеризации можно визуализировать с помощью дендрограммы, которая представляет собой дерево с отдельными объектами на нижнем уровне и объединенными кластерами на более высоких уровнях. Каждый уровень дендрограммы представляет объединение объектов или кластеров на предыдущем уровне.

Этот метод построения кластеров подразделяется на два основных подхода:

— агломеративный;

— дивизионный.

Преимущества иерархической кластеризации.

— Для работы метода число кластеров не должно быть задано изначально.

— Дендрограмма, которая является результатом иерархической кластеризации, позволяет визуализировать объединение кластеров.

— Каждый шаг объединения кластеров сохраняется, что позволяет более детально изучать данные.

Недостатки иерархической кластеризации.

— Различные меры близости могут давать различные результаты кластеризации, что может приводить к разным и нестабильным результатам.

— Для больших объемов данных возникают вычислительные проблемы, связанные с пересчетом расстояний между объектами на каждом этапе класстеризации.

1.2.1.1 Агломеративный подход

При агломеративном подходе [4] разбиение начинается с того, что каждый объект рассматривается как отдельный кластер. Затем на каждом шаге объединяются две наиболее близкие группы, пока все объекты не будут объединены в один кластер.

Алгоритм агломеративного подхода иерархической кластеризации можно представить в следующем виде.

1. Вычисление матрицы расстояний между всеми парами объектов (матрицы несходства). Для этого используется выбранная мера близости.
2. Размещение каждого объекта в отдельный кластер.
3. Выбор двух наиболее близких кластеров на основе матрицы несходства.
4. Объединение выбранных кластеров в один новый кластер.
5. Вычисление расстояния между новым кластером и всеми остальными кластерами (дополнение матрицы несходства).
6. Повторение шагов 3-5 до тех пор, пока все объекты не будут объединены в один кластер.
7. Построение дендрограммы, которая иллюстрирует процесс объединения кластеров.

1.2.1.2 Дивизионный подход

При дивизионном подходе [4] разбиение начинается с того, что все объекты находятся в одном кластере. Затем на каждом шаге наиболее разнородная группа разделяется на два новых более гомогенных кластера.

Алгоритм дивизионного подхода иерархической кластеризации можно представить в следующем виде.

1. Вычисление матрицы расстояний между всеми парами объектов (матрицы несходства). Для этого используется выбранная мера близости.
2. Размещение всех объектов в одном кластере.

3. Выбор самого разнородного кластера на основе матрицы несходства.

4. Разделение выбранного кластера на два новых более гомогенных кластера.

5. Вычисление расстояния между новыми кластерами и всеми остальными кластерами (дополнение матрицы несходства).

6. Повторение шагов 3-5 до тех пор, пока каждый объект не будет находиться в своем собственном кластере.

7. Построение дендрограммы, которая иллюстрирует процесс разделения кластеров.

1.2.2 K-средних

Метод k-средних [5] — это один из наиболее распространенных методов кластеризации, который разбивает набор данных на заранее определенное число кластеров k. Каждый кластер в этом методе представлен своим центром.

Алгоритм можно представить в следующем виде.

1. Выбирается число кластеров k.

2. Случайным образом выбираются k центров кластеров.

3. Для каждого объекта в наборе данных определяется ближайший к нему центр кластера (с помощью матрицы несходства).

4. Для каждого кластера вычисляется центроид — вектор, элементы которого представляют собой средние значения соответствующих признаков, вычисленных по всем записям кластера.

5. Центроиды становятся новыми центрами кластеров.

6. Шаги 3–5 повторяются до тех пор, пока центры кластеров не перестанут изменяться или алгоритм не превысит максимальное количество шагов.

7. В результате каждому объекту будет присвоен номер кластера, к которому он наиболее близок.

Преимущества метода разбиения k-средних.

— Является одним из наиболее простых алгоритмов кластеризации.

— Не требует предварительной разметки данных.

— Можно получить результаты разбиения с использованием различных метрик качества кластеризации.

— Может масштабироваться для кластеризации больших объемов данных.

Недостатки метода разбиения k-средних.

— Требуется заранее задать число кластеров k, что является недостатком в тех случаях, когда нет явной информации о количестве кластеров.

— Чувствителен к исходной инициализации центроидов кластеров, что может привести к различным результатам кластеризации при разных начальных условиях.

— Чувствителен к выбросам в данных. Если выбросы содержатся в кластере, то центр кластера будет смещаться ближе к ним и удаляться от большинства наблюдений в группе.

— Подходит только для вещественных чисел.

1.2.3 K-режимов

Алгоритм k-режимов [6] представляет собой модификацию алгоритма k-средних. Основная разница между двумя алгоритмами заключается в том, что k-средних разработан для работы с количественными числовыми данными, а k-режимов — для работы с категориальными данными.

Алгоритм k-средних использует расстояние между объектами, чтобы определить близость между ними и сгруппировать их в кластеры. Расстояния обычно измеряются по евклидовой метрике.

Алгоритм k-режимов имеет схожий подход, но вместо расстояний использует меры сходства между объектами, которые основаны на анализе сходства между категориями. Чаще всего в качестве такой метрики используют расстояние Хэмминга.

Другое отличие между рассматриваемыми алгоритмами заключается в способе определения центров кластеров. Метод k-средних определяет центры путем вычисления среднего значения для каждого кластера, тогда как k-режимов определяет центр кластера как объект, который является наиболее типичным для кластера.

Алгоритм можно представить в следующем виде.

1. Выбирается число кластеров k.
2. Случайным образом выбираются k объектов из набора данных в качестве начальных центров кластеров.
3. На каждой итерации каждый объект присваивается наиболее близкому по сходству к центру кластера (режиму). Измерение сходства в этом алгоритме осуществляется на основе расстояния Хэмминга.
4. Для каждого режима рассчитывается наиболее часто встречающиеся значения каждого признака.
5. Создаются новые режимы. Это делается путем выбора наиболее типичного объекта в данном кластере с учетом наиболее часто встречающихся значений каждого признака.
6. Шаги 3–5 повторяются до тех пор, пока центры кластеров не перестанут изменяться или алгоритм не превысит максимальное количество шагов.
7. В результате каждому объекту будет присвоен номер кластера, к которому он наиболее близок.

Преимущества метода разбиения k-режимов.

— Подходит для работы с категориальными данными.

— Простота реализации.

— Не требует метрики расстояния. Вместо этого он использует меру сходства между объектами на основе их категориальных признаков.

— Устойчив к выбросам, так как работает по принципу минимизации суммарного количества различий в категориях каждого кластера. Это означает, что он не зависит от расстояний между точками, а только от схожести категорий внутри кластеров.

Недостатки метода разбиения k-режимов.

— Требуется заранее задать число кластеров k, что является недостатком в тех случаях, когда нет явной информации о количестве кластеров.

— Чувствителен к исходной инициализации центроидов кластеров, что может привести к различным результатам кластеризации при разных начальных условиях.

— Проблемы в работе с разреженными данными.

— Не подходит для вещественных чисел.

1.2.4 K-прототипов

Алгоритм кластеризации k-прототипов [7] — это комбинация метода k-средних и алгоритма k-режимов для кластеризации данных, содержащих как числовые, так и категориальные признаки. Он применяется для данных, где распределение признаков может быть как числовым (например, возраст, доход), так и категориальным (например, пол, цвет, марка автомобиля).

Алгоритм можно представить в следующем виде.

1. Выбирается число кластеров k.

2. Случайным образом выбираются k центров кластеров.

3. Для каждого объекта в наборе данных определяется ближайший к нему центр кластера. Для вычисления матрицы несходства используются комбинированные метрики, которые учитывают расстояние между категориальными и числовыми данными.

4. Для каждого кластера вычисляется центроид — вектор, элементы которого представляют собой средние значения (для числовых переменных)

или наиболее часто встречающиеся значения (для качественных переменных) соответствующих признаков, вычисленные по всем записям кластера.

5. Центроиды становятся новыми центрами кластеров.

6. Шаги 3–5 повторяются до тех пор, пока центры кластеров не перестанут изменяться или алгоритм не превысит максимальное количество шагов.

7. В результате каждому объекту будет присвоен номер кластера, к которому он наиболее близок.

Преимущества метода разбиения k-прототипов.

— Может обрабатывать данные, содержащие как числовые, так и категориальные признаки.

— Простота реализации.

— Более устойчив к выбросам, чем алгориитм k-средних, в случае работы с категориальными данными, так как он не будет зависеть от расстояний между точками, а только от схожести категорий внутри кластеров (как и алгоритм k-режимов).

Недостатки метода разбиения k-прототипов.

— Требуется заранее задать число кластеров k, что является недостатком в тех случаях, когда нет явной информации о количестве кластеров.

— Чувствителен к исходной инициализации центроидов кластеров, что может привести к различным результатам кластеризации при разных начальных условиях.

1.2.5 C-средних

Метод разбиения c-средних [8] представляет собой обобщение метода кластеризации k-средних в ситуации, когда точки данных могут принадлежать нескольким кластерам одновременно. Данный метод относится к нечетким

алгоритмам кластеризации. При нечетком разбиении каждая точка имеет вероятность принадлежности к каждому кластеру, а не полностью принадлежит только одной группе.

Алгоритм можно представить в следующем виде.

1. Выбирается число кластеров k.
2. Случайным образом выбираются k центров кластеров.
3. Рассчитывается расстояние между каждой точкой данных и каждым центром кластера (с помощью матрицы несходства).
4. На основе вычисленных расстояний определяется степень (вероятность) принадлежности каждой точки конкретному кластеру (в диапазоне от 0 до 1).
5. Определяются новые центры кластеров на основе вычисленных степеней принадлежности.
6. Шаги 3–5 повторяются до достижения условия сходимости (например, пока сумма квадратов расстояний между объектами и центрами кластеров не перестанет изменяться).
7. В результате, каждая точка данных получит степень принадлежности к каждому из кластеров.

Преимущества метода разбиения с-средних.

— Позволяет определять нечеткую принадлежность каждой точки данных к кластеру.

— Имеет возможность задавать коэффициент нечеткости, который определяет степень размытости кластеров.

— Более устойчив к шумам в данных, чем жесткие алгоритмы разбиения, так как объекты имеют определенную степень принадлежности к каждому кластеру, а не закреплены за отдельной группой.

— Более устойчив к выбору начальных центров, чем алгоритм k-средних, благодаря нечеткой принадлежности точек к кластерам.

Недостатки метода разбиения c-средних.

— Требуется заранее задать число кластеров k, что является недостатком в тех случаях, когда нет явной информации о количестве кластеров.

— Определение оптимального значения параметра нечеткости (m) требует дополнительного тестирования.

— Имеет большую вычислительную сложность, чем классический алгоритм k-средних.

1.2.6 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [9] — это метод кластеризации, который использует плотность данных для группировки объектов в кластеры. Основной идеей DBSCAN является нахождение областей высокой плотности точек в данных, и разделение их на отдельные кластеры. Данный метод позволяет находить кластеры произвольной формы в данных, не требуя заранее заданного количества групп.

На вход DBSCAN помимо матрицы несходства принимает два основных параметра: радиус ϵ-окрестности и минимальное количество соседних точек в пределах заданного радиуса (minPts). Эти параметры определяют границы и ширину области высокой плотности точек, которые должны быть объединены в кластеры.

Алгоритм можно представить в следующем виде.

1. Выбирается случайная точка из нерассмотренных объектов данных.

2. Вычисляется количество соседей вокруг выбранной точки на расстоянии, не превышающем заданный радиус ϵ, и проверяется, достаточно ли этого количества соседей для добавления точки в кластер.

3. Если количество соседей больше или равно заданному порогу minPts, точка считается «основной» и добавляется в текущий кластер. Также просматриваются все соседи этой точки, и если какой-то из них также является «основной» точкой, то он добавляется в текущий кластер.

4. Если количество соседей меньше порога minPts, но точка находится в радиусе ϵ от другой «основной» точки, она считается «граничной» и добавляется в текущий кластер как граничная. Однако, эта точка не исследуется дальше на наличие «основных» соседей и не может стать их источником.

5. Если точка не имеет достаточно соседей на установленном расстоянии ϵ, и не находится в окружении граничных точек, она считается «шумом» и отбрасывается.

6. Повторяется процесс до тех пор, пока все объекты в данных не будут просмотрены.

7. Если два кластера имеют пересечение, то они объединяются в один.

8. В итоге получается набор кластеров и точки, которые не вошли ни в один кластер, и считаются шумом.

Преимущества метода разбиения DBSCAN.

— Устойчивость к шуму и выбросам, так как данный метод позволяет обрабатывать и отделять точки, которые не принадлежат кластерам и являются выбросами.

— Нет необходимости знать количество кластеров до начала разбиения.

— Может обрабатывать кластеры произвольной формы и размеров.

— Может обрабатывать данные, содержащие как числовые, так и категориальные признаки.

Недостатки метода разбиения DBSCAN.

— Чувствительность к выбору начальных параметров: радиусу ϵ-окрестности и минимальному количеству соседних точек в пределах заданного радиуса (minPts).

— Не может обработать данные с различной плотностью.

1.2.7 Минимальное покрывающее дерево

Кластеризация на основе минимального покрывающего дерева [10] (МПД) — это метод кластеризации, который основывается на построении минимального остовного дерева (МОД) [11] для набора данных. МОД представляет собой подмножество ребер связного графа, которое соединяет все вершины графа и имеет минимальную сумму весов. Кластеры образуются путем удаления ребер, длина которых превышает пороговое значение, из минимального остовного дерева.

Алгоритм можно представить в следующем виде.

1. Строится полный граф данных, где каждый объект представлен вершиной, а вес ребра между двумя вершинами равен расстоянию между ними.

2. Строится минимальное остовное дерево графа, используя алгоритм Прима или Крускала.

3. Остовное дерево разделяется на кластеры путем удаления ребер с весами, превышающими пороговое значение.

4. В результате каждая отрезанная ветвь дерева будет представлять собой отдельный кластер.

Преимущества метода разбиения на основе минимального покрывающего дерева.

— Нет необходимости знать количество кластеров до начала разбиения.

— Является устойчивым к шуму, так как использует МОД. В результате работы алгоритма ребра с большими весами будут отброшены.

— Каждая отрезанная ветвь дерева показывает иерархические отношения между объектами в кластере, что позволяет получить более четкое представление о свойствах данных.

— Может обрабатывать данные, содержащие как числовые, так и категориальные признаки.

Недостатки метода разбиения на основе минимального покрывающего дерева.

— Необходимо заранее знать пороговое значение расстояний между объектами.

— Не обеспечивает равномерное разбиение на кластеры.

1.2.8 Сравнение методов разбиения

Сравнение методов разбиения данных предлагается проводить по следующим критериям.

1. Возможность обрабатывать данные, содержащие числовые значения.
2. Возможность обрабатывать данные, содержащие категориальные признаки.
3. Нет необходимости знать количество кластеров до начала разбиения.
4. Тип алгоритма кластеризации.
5. Форма кластеров после применения алгоритма.
6. Входные данные.
7. Выходные данные.

Результаты сравнения методов класстеризации данных приведены в таблицах 1.1–1.2.

Таблица 1.1 – Сравнение рассмотренных методов разбиения данных (часть 1)

Крит.	**Иерархический**	**K-средних**	**K-режимов**	**K-прототипов**
Кр. 1	+	+	–	+
Кр. 2	+	–	+	+
Кр. 3	+	–	–	–
Кр. 4	Иерархический	Центроидный	Центроидный	Центроидный
Кр. 5	Произвольная	Гиперсфера	Гиперсфера	Гиперсфера
Кр. 6	Матрица несходства, число кластеров (необязательно)	Массив объектов, число кластеров	Массив объектов, число кластеров	Массив объектов, число кластеров
Кр. 7	Бинарное дерево кластеров	Кластеры с номерами объектов	Кластеры с номерами объектов	Кластеры с номерами объектов

Таблица 1.2 – Сравнение рассмотренных методов разбиения данных (часть 2)

Крит.	**C-средних**	**DBSCAN**	**МПД**
Кр. 1	+	+	+
Кр. 2	–	+	+
Кр. 3	–	+	+
Кр. 4	Центроидный	На основе плотности	На основе графов
Кр. 5	Гиперсфера	Неравномерная	Произвольная
Кр. 6	Массив объектов, число кластеров	Матрица несходства, радиус ϵ, кол-во соседей minPts	Матрица несходства, пороговое значение расстояний
Кр. 7	Центры кластеров, матрица принадлежности объектов к кластерам	Набор кластеров и точки, которые не вошли ни в один кластер	Древовидная структура кластеров

1.3 Критерии связи

В случае использования иерархических алгоритмов встает вопрос, как объединять между собой кластеры, как вычислять «расстояния» между ними. Для решения этой задачи используются различные критерии связи [12], которые определяют правило, по которому происходит объединение групп. Выбор критерия связи зависит от природы данных, целей исследования и типа кластеров, которые нужно выделить. Рассмотрим самые популярные метрики.

Одиночная связь [13]. Расстояние между двумя кластерами — кратчайшее расстояние между двумя точками в каждом кластере.

Такая связь считается простой в реализации, но может приводить к образованию длинных цепочек объектов. Это происходит потому, что связь не учитывает сходство между всеми объектами внутри каждого кластера и может объединять две группы, состоящие из объектов, не похожих друг на друга. Данный метод подвержен проблемам шума.

Полная связь [14]. Расстояние между двумя кластерами — самое длинное расстояние между двумя точками в каждом кластере.

Этот критерий связи обеспечивает более однородные и компактные кластеры, чем одиночная связь. Полная связь учитывает расстояние между объектами внутри каждой группы, что делает его менее чувствительным к выбросам. Однако этот метод более затратен по вычислительным ресурсам (чем одиночная связь) из-за того, что требует вычисления расстояний между всеми парами объектов внутри каждого кластера. Также полная связь может привести к объединению групп с большим числом объектов.

Средняя связь [15]. Расстояние между двумя кластерами — это среднее расстояние между каждой точкой в одном кластере до каждой точки в другом кластере.

Данный метод уменьшает вероятность объединения кластеров с большим числом объектов. Связь по средним значениям учитывает не только расстояние между ближайшими объектами внутри каждого кластера, но и среднее расстояние между всеми парами объектов. Такой подход обеспечивает большую устойчивость к выбросам, чем односвязная связь. Однако объединение кластеров может быть не таким компактным, как при использовании полной связи.

1.4 Меры расстояний

В кластеризации используются различные меры расстояний для определения схожести между объектами (точками, векторами) и расчета расстояний между кластерами [10].

Выбор меры расстояний в кластеризации в значительной степени зависит от типа данных и задачи, которую мы решаем. Некоторые меры расстояний могут применяться только с определенными типами данных. Например, евклидово расстояние работает на непрерывных числовых данных, но не подходит для категориальных и бинарных признаков. Число измерений в данных также может повлиять на выбор меры расстояний. Например, расстояние Манхэттен лучше себя показывает на большом количестве признаков.

Различные меры расстояний могут приводить к различным результатам кластеризации. Рассмотрим самые популярные из них.

1.4.1 Евклидово расстояние

Евклидово расстояние [16] является одной из наиболее распространенных мер расстояний, используемых в кластеризации. Эта метрика позволяет измерять расстояние между двумя точками в многомерном пространстве. Для двух точек X и Y в n-мерном пространстве формула евклидова расстояния определяется следующим образом:

$$d(X, Y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}, \tag{1.1}$$

где $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$ — координаты точек X и Y соответственно.

Евклидово расстояние широко используется для кластеризации данных в пространстве вещественных чисел. Однако оно не подходит для работы с категориальными и бинарными данными, из-за нарушения условия непрерывности признаков.

1.4.2 Квадрат евклидова расстояния

Квадрат евклидова расстояния [16] — это мера расстояний между двумя точками, которая измеряется как сумма квадратов разницы между координатами точек на каждом измерении. Для двух точек X и Y в n-мерном пространстве формула квадрата евклидова расстояния определяется следую-

щим образом:

$$d^2(X, Y) = (x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2, \tag{1.2}$$

где $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$ — координаты точек X и Y соответственно.

Квадрат евклидова расстояния также широко используется в кластеризации данных. Он менее чувствителен к выбросам, чем обычное евклидово расстояние. Выбор данной меры может привести к потере точности в случае, если значения признаков сильно различаются. Квадрат евклидова расстояния также подходит только для работы с числовыми данными.

1.4.3 Расстояние городских кварталов

Расстояние городских кварталов (Манхэттенское расстояние) [17] — это мера расстояния между двумя точками, измеряемая как сумма разницы между координатами точек на каждом измерении. Для двух точек X и Y в n-мерном пространстве формула расстояния городских кварталов определяется следующим образом:

$$d(X, Y) = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|, \tag{1.3}$$

где $x_1, x_2, \ldots, x_n$ и $y_1, y_2, \ldots, y_n$ — координаты точек X и Y соответственно.

Такое расстояние названо в честь структурных особенностей городских кварталов: движение разрешено только вправо и вверх на квадратной сетке, ограниченной городскими кварталами, нельзя перемещаться по кратчайшему прямому пути.

Манхэттенское расстояние учитывает не только разность значений признаков, но и их взаимное расположение, что в некоторых случаях может быть более значимым. Это расстояние используется для кластеризации точек в пространстве и не подходит для работы с категориальными данными. Недостатком является то, что расстояние городских кварталов не может учитывать важность разных признаков и предполагает, что каждый признак имеет одинаковый вес.

1.4.4 Расстояние Чебышева

Расстояние Чебышева [18] — это мера расстояния между двумя точками, которая используется в кластеризации и определяется как максимальное значение разницы между значениями двух точек по каждому измерению. Для двух точек X и Y в n-мерном пространстве расстояние Чебышева можно определить следующим образом:

$$d(X, Y) = \max_{i=1}^{n} |x_i - y_i|, \tag{1.4}$$

где x_i и y_i — координаты i-того измерения точек X и Y соответственно.

Расстояние Чебышева выбирается для работы в тех случаях, когда важно учитывать максимальное отклонение каждой координаты каждого объекта в группе. Методы кластеризации, использующие данное расстояние, устойчивы к выбросам в данных. Измерение максимального отклонения справедливо только для числовых значений, поэтому расстояние Чебышева не подходит для работы с категориальными данными.

1.4.5 Расстояние Минковского

Расстояние Минковского [19] — это обобщение евклидова расстояния и расстояния городских кварталов. Для двух точек X и Y в n-мерном пространстве формула расстояния Минковского порядка p определяется следующим образом:

$$d(X, Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{1/p}, \ p \geq 1, \tag{1.5}$$

где x_i и y_i — координаты i-того измерения точек X и Y соответственно, p — параметр степени, который определяет, как взаимодействуют различные признаки при вычислении расстояния.

Когда $p = 1$, формула расстояния Минковского дает оценку расстояние Манхэттена, когда $p = 2$, она дает евклидово расстояние, при $p = \infty$ метрика обращается в расстояние Чебышева.

1.4.6 Степенное расстояние

Степенное расстояние [10] — это обобщение формулы расстояния Минковского. Для двух точек X и Y в n-мерном пространстве формула степенного расстояния определяется следующим образом:

$$d(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p\right)^{1/r}, \ p \geq 1, \ r \geq 1, \tag{1.6}$$

где x_i и y_i — координаты i-того измерения точек X и Y соответственно, p — показатель степени, определяющий, как сильно каждый признак учитывается в общем расстоянии, r — параметр, который определяет масштаб расстояния.

Степенное расстояние используется для кластеризации числовых данных. Данная метрика используется для измерения расстояния между объектами на основе нелинейных отношений между признаками.

1.4.7 Расстояние Хэмминга

Расстояние Хэмминга [20] — мера сходства между объектами, которая основана на анализе сходства между категориями. Данная метрика показывает количество различающихся позиций для строк с одинаковой длинной. Формула расстояния Хэмминга в кластеризации для двух векторов $\vec{x}$ и $\vec{y}$ длины n может быть записана следующим образом:

$$d(\vec{x},\vec{y}) = \sum_{i=1}^{n} \text{sign}|x_i - y_i|, \tag{1.7}$$

где x_i и y_i — значения i-ых элемента векторов $\vec{x}$ и $\vec{y}$ соответственно.

Таким образом, расстояние Хэмминга подходит для работы с бинарными данными. Для категориальных данных оно может быть рассчитано на основе количества несовпадений категорий между объектами. Однако для работы с числовыми данными следует выбирать другое расстояние.

1.4.8 Расстояние Говера

Расстояние Говера [21] — метрика сходства в кластеризации, позволяющая выполнять расчет расстояний между объектами, содержащими как

числовые, так и категориальные признаки. Формальная запись расстояния Говера для двух векторов $\vec{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip})$ и $\vec{x_j} = (x_{j1}, x_{j2}, \ldots, x_{jp})$ может быть записана следующим образом:

$$d(\vec{x_i}, \vec{x_j}) = \frac{\sum_{k=1}^{p} s_{ijk} \cdot \delta_{ijk}}{\sum_{k=1}^{p} \delta_{ijk}}, \tag{1.8}$$

где δ_{ijk} — величина, принимающая значение 1, если $\vec{x}$ и $\vec{y}$ можно сравнить по k-ому признаку, и 0, если сравнить нельзя, $s_{ijk} \in [0, 1]$ — оценка сходства двух признаков (чем признаки ближе друг к другу, тем оценка ближе к 0), p — количество признаков в векторах.

Формула расстояния Говера, когда отсутствующих значений не существует, может быть записана следующим образом:

$$d(\vec{x_i}, \vec{x_j}) = \frac{\sum_{k=1}^{p} s_{ijk}}{p}, \tag{1.9}$$

Оценка сходства двух признаков s_{ijk} определяется по-разному для каждого типа данных.

1. Для числовых перемеенных:

$$s_{ijk} = \frac{|x_{ik} - x_{jk}|}{R_k}, \tag{1.10}$$

   где R_k — диапазон значений признака k, x_{ik}, x_{jk} — значения k-ых признаков векторов $\vec{x_i}$, $\vec{x_j}$ соответственно.

2. Для категориальных переменных:

$$s_{ijk} = \begin{cases} 0, & \text{если } x_{ik} = x_{jk}; \\ 1, & \text{иначе,} \end{cases} \tag{1.11}$$

   где x_{ik}, x_{jk} — значения k-ых признаков векторов $\vec{x_i}$, $\vec{x_j}$ соответственно.

1.4.9 Сравнение мер расстояний

Сравнение мер расстоняний предлагается проводить по следующим критериям.

1. Возможность находить расстояние между числовыми данными.

2. Возможность находить расстояние между категориальными данными (без предварительной обработки категориальных признаков и кодирования качественных параметров числами).

Результаты сравнения мер расстояний приведены в таблице 1.3.

Таблица 1.3 – Сравнение рассмотренных мер расстояний

Мера расстояний	**Критерий 1**	**Критерий 2**
Евклидово расстояние	+	-
Квадрат евклидова расстояния	+	-
Расстояние городских кварталов	+	-
Расстояние Чебышева	+	-
Расстояние Минковского	+	-
Степенное расстояние	+	-
Расстояние Хэмминга	-	+
Расстояние Говера	+	+

1.5 Методы оценки качества кластеризации

1.5.1 Метод локтя

Метод локтя [22] — один из самых распространенных методов оценки качества разбиения, который используется для определения оптимального количества кластеров. Этот метод получил своё название по форме графика зависимости среднего расстояния в пределах группы от количества кластеров.

Идея метода заключается в том, чтобы найти на графике такую точку (точку «локтя»), после которой уменьшение среднего расстояния между элементами в кластере будет не так заметно. Оптимальное число кластеров будет находиться в точке «локтя».

Этот метод оценки качества разбиения применяется, если важнейшим фактором для анализа является компактность кластеров, то есть сходство внутри групп.

1.5.2 Метод оценки силуэтов

Метод оценки силуэтов [23] — это метод оценки качества разбиения, основанный на вычислении коэффициента силуэта для каждого объекта набора данных. График силуэтов показывает, насколько близко каждая точка внутри одной группы расположена к точкам ближайшего соседнего кластера.

Коэффициент силуэта для каждого объекта i может быть вычислен следующим образом:

$$s_i = \frac{b_i - a_i}{max(a_i, b_i)}, \tag{1.12}$$

где a_i — среднее расстояние между объектом i и другими объектами в том же кластере, b_i — среднее расстояние между объектом i и объектами из ближайшего кластера.

Значения коэффициентов силуэтов s_i находятся в диапазоне $[-1, 1]$ и могут быть интерпретированы следующим образом:

— если значение близко к 1, то объект находится в хорошо разделенном кластере;

— если значение близко к -1, то объект ошибочно находится в другом кластере;

— если значение близко к 0, то объект находится между кластерами.

Оптимальным считается такое количество кластеров, при котором достигаеся максимальное среднее значение коэффициентов силуэтов.

1.6 Постановка задачи

В рамках выполнения выпускной квалификационной работы требуется реализовать метод разбиения категориальных данных на основе агломеративного подхода иерархический кластеризации. При создании метода необходимо определить:

— входные и выходные данные алгоритма кластеризации;

— критерий связи кластеров в иерархической части разрабатываемого метода;

— меру расстояний при вычислении матрицы несходста.

Иерархические методы возвращают бинарное дерево кластеров, узлы которого содержат номера объектов, входящих в данную группу. На основе этих узлов можно вычислить центры кластеров, используемые в центроидных методах разбиения. Поэтому именно кластеризация центроидного типа может быть использованы для уточнения результатов иерархического разбиения в разрабатываемом гибридном методе.

Формальная постановка задачи в виде $IDEF0$-диаграммы представлена на рисунках 1.1–1.2.

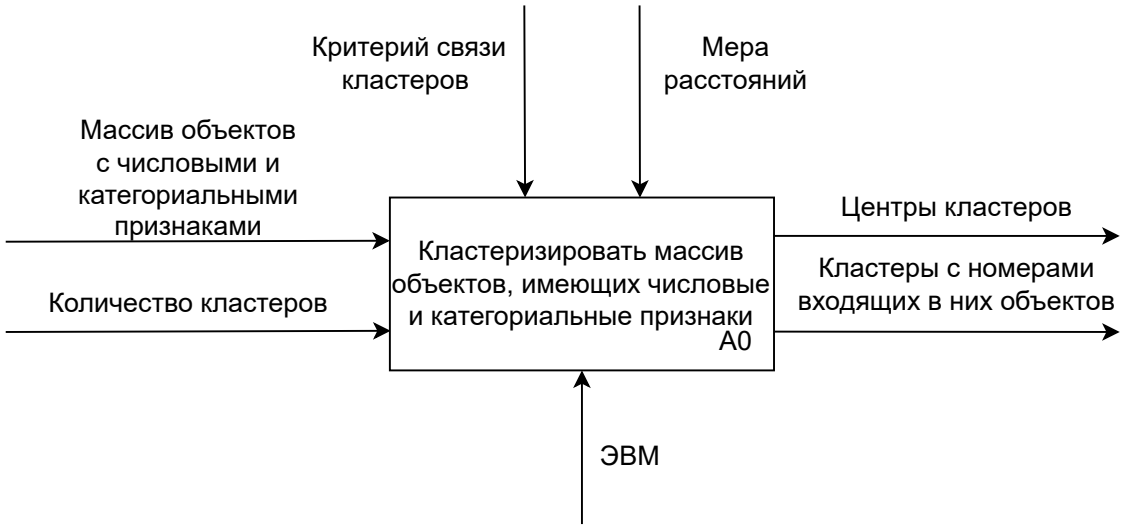

Рисунок 1.1 – IDEF0–диаграмма уровня A0

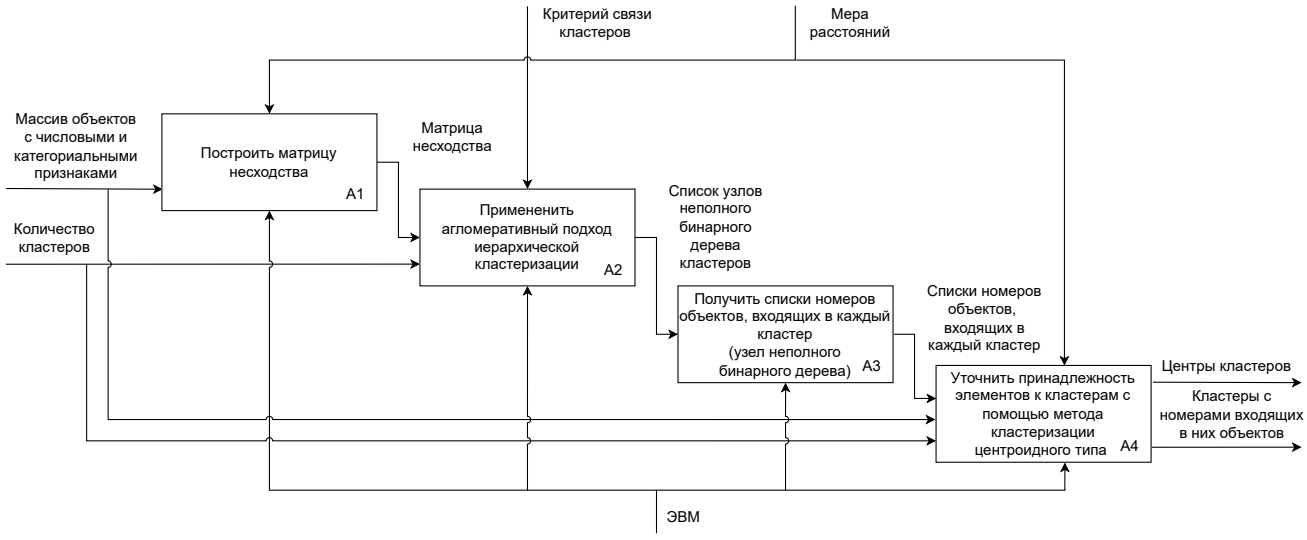

Рисунок 1.2 – IDEF0–диаграмма уровня A1

Вывод

В данном разделе была описана предметная область, разобраны методы разбиения данных, проведено их сравнение по выделенным критериям. Были рассмотрены меры расстояний, используемые для определения схожести между объектами, проведено их сравнение по типу обрабатываемых данных. Для иерархической кластеризации были рассмотрены критерии связи кластеров. Также была описана формальная постановка задачи и были разобраны методы оценки качества разбиения, с помощью которых будет производиться сравнения разрабатываемого метода с аналогами.

2 Конструкторский раздел

2.1 Требования к разрабатываемому методу разбиения данных

Для гибридного метода разбиения на основе агломеративного подхода иерархической кластеризации были выдвинуты следующие требования.

— Разрабатываемый метод должен уметь работать с числовыми параметрами помимо категориальных, так как качественные признаки могут присутствовать вместе с количественными. Такое допущение позволит использовать данный метод для решении большего числа задач.

— Разрабатываемый алгоритм кластеризации на вход должен получать массив объектов с числовыми и категориальными признаками, а также итоговое количество кластеров.

— В результате работы гибридный метод разбиения должен возвращать два списка: центры образованных класстеров и сами кластеры с номерами входящих в них объектов.

2.2 Требования к разрабатываемому ПО

Для демонстрации работы гибридного метода необходимо разработать ПО со следующими требованиями.

— Взаимодействие пользователя с ПО должно осуществляться с помощью графического интерфейса.

— Необходмо предусмотреть возможность изменения количества обрабатываемых объектов.

— Необходмо предусмотреть возможность изменения итогового количества кластеров.

— Пользователь должен иметь возможность сравнения гибридного метода разбиения с базовыми, на основе которых он был разработан, с помощью методов оценки качества кластеризации (методов локтя и оценки силуэтов).

2.3 Проектирование метода разбиения данных

При создании любого метода разбиения данных необходимо определить, каким образом будет определяться схожесть между объектами. Также, при использовании иерархического подхода в класстеризации, необходимо выбрать критерий связи кластеров.

Для разрабатываемого гибридного метода разбиения на основе агломеративного подхода иерархической кластеризации были выбраны следующие метрики.

— В качестве критерия связи кластеров в иерархической части разрабатываемого метода должна быть использована полная связь, так как она обеспечивает более однородные и компактные кластеры, чем одиночная или средняя.

— В качестве меры расстояний при вычислении матрицы несходства должно быть использовано расстояние Говера, так как оно единственное из рассмотренных подходит для работы как с числовыми, так и с категориальными данными. При вычислении данного расстоняния будем считать, что отсутствующих значений в параметрах не существует.

После применения иерархической кластеризации и преобразования полученных данных (блоки А2 и А3 на рисунке 1.2) необходимо уточнить принадлежность элементов кластерам с помощью метода кластеризации центроидного типа (блок А4 на рисунке 1.2). В качестве такого уточняющего метода было выбрано разбиение k-прототипов, так как оно единственное из методов кластеризации центроидного типа способно обрабатывать данные, содержащие как категориальные, так и числовые признаки.

2.4 Схемы разрабатываемого гибридного метода кластеризации

Схема гибридного метода кластеризации представлена на рисунке 2.1. Она состоит из четырех основных пунктов, три из которых далее будут рассмотрены более подробно.

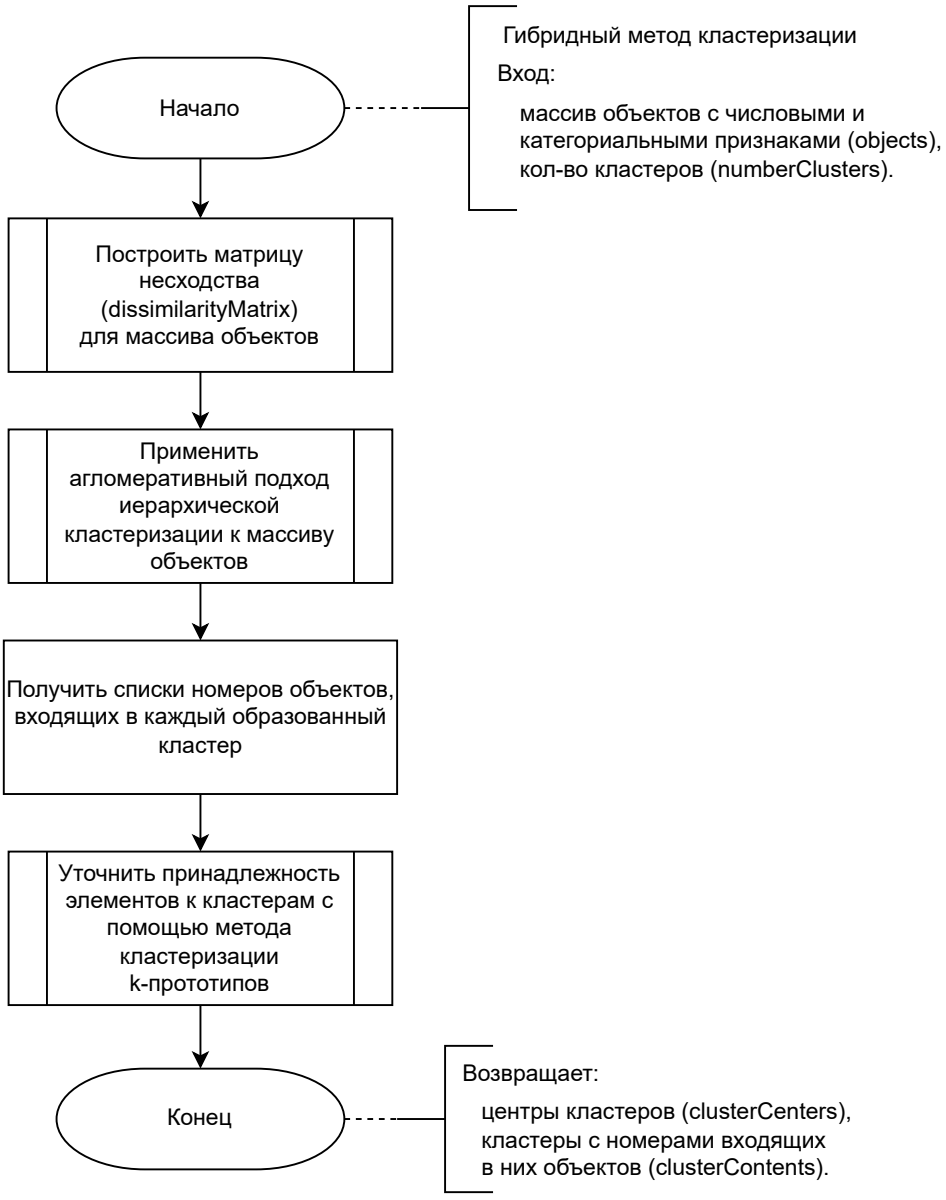

Рисунок 2.1 – Схема гибридного метода кластеризации

Схема нахождения матрицы несходства представлена на рисунке 2.2. Данная матрица показывает степень различия между объектами. Для определения расстояния между элементами матрицы используется расстояние Говера.

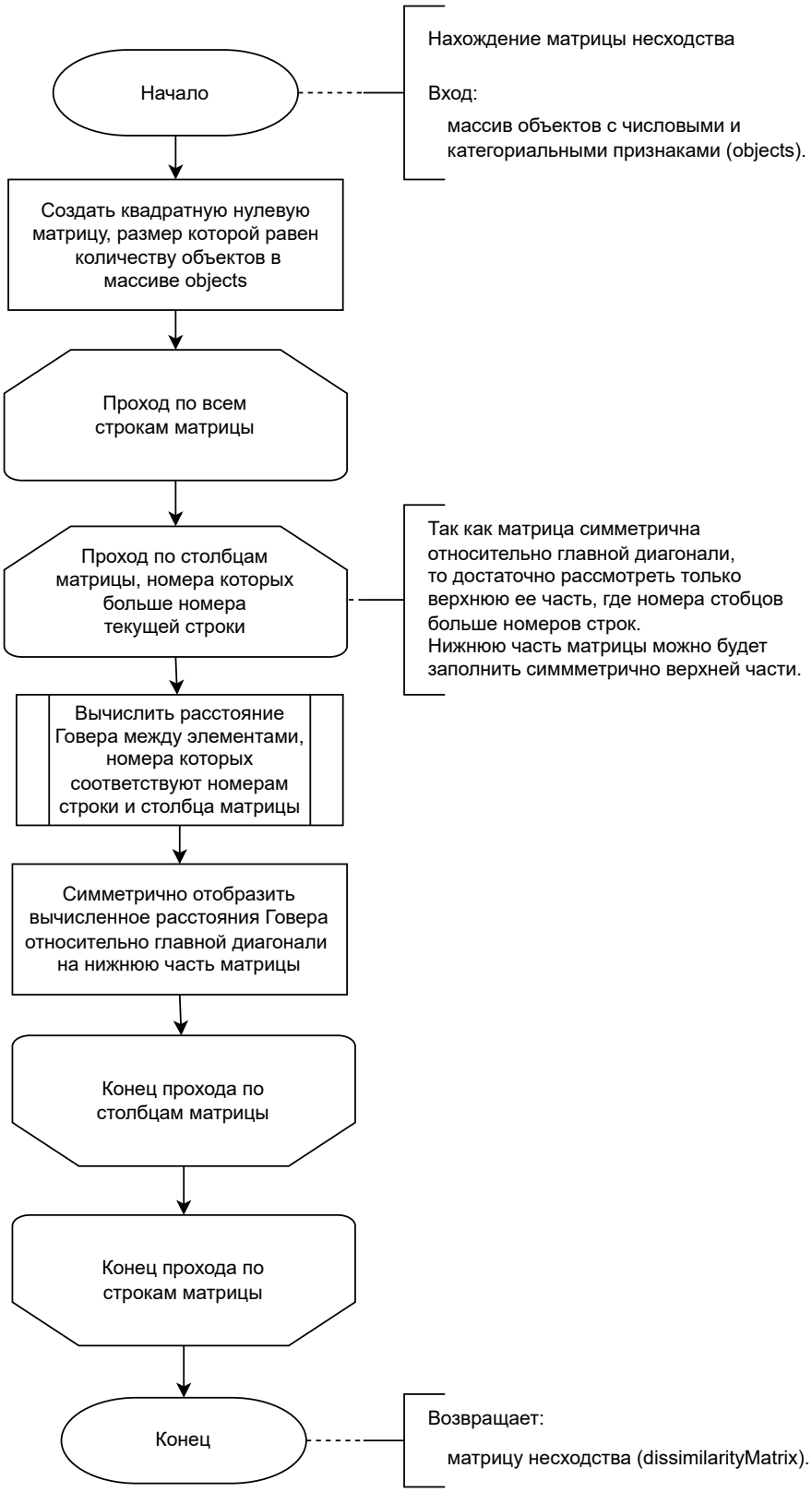

Рисунок 2.2 – Схема нахождения матрицы несходства

Схема нахождения расстояния Говера между двумя элементами с множеством признаков представлена на рисунке 2.3. Данное расстояние позволяет определить степень различия между объектами. Чем сильнее они отличаются друг от друга, тем ближе значение к 1.

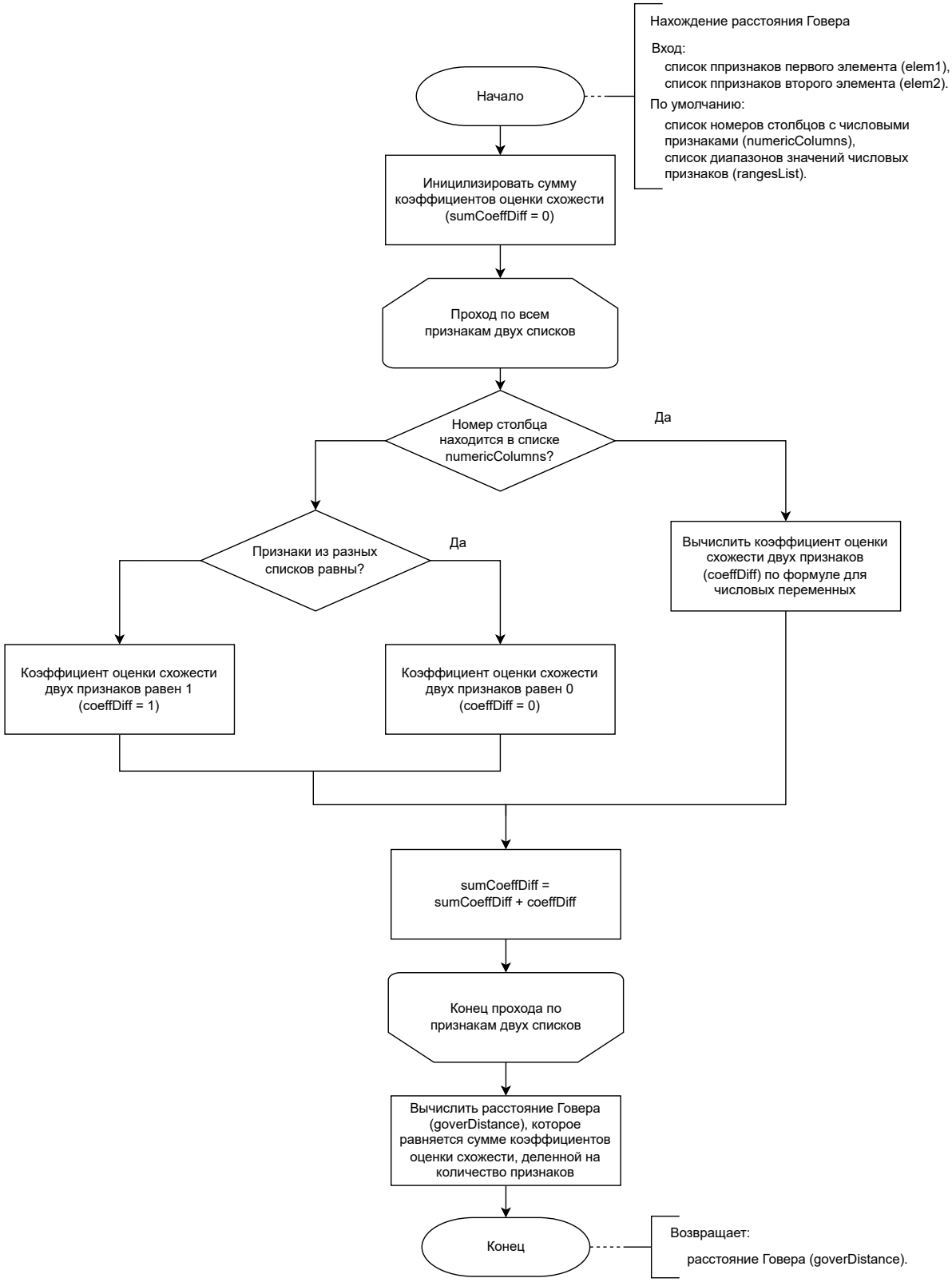

Рисунок 2.3 – Схема нахождения расстояния Говера

Схема иерархической части гибридного метода кластеризации представлена на рисунке 2.4. Это основная часть алгоритма, в результате которой уже будут получены кластеры в виде списка узлов неполного бинарного дерева.

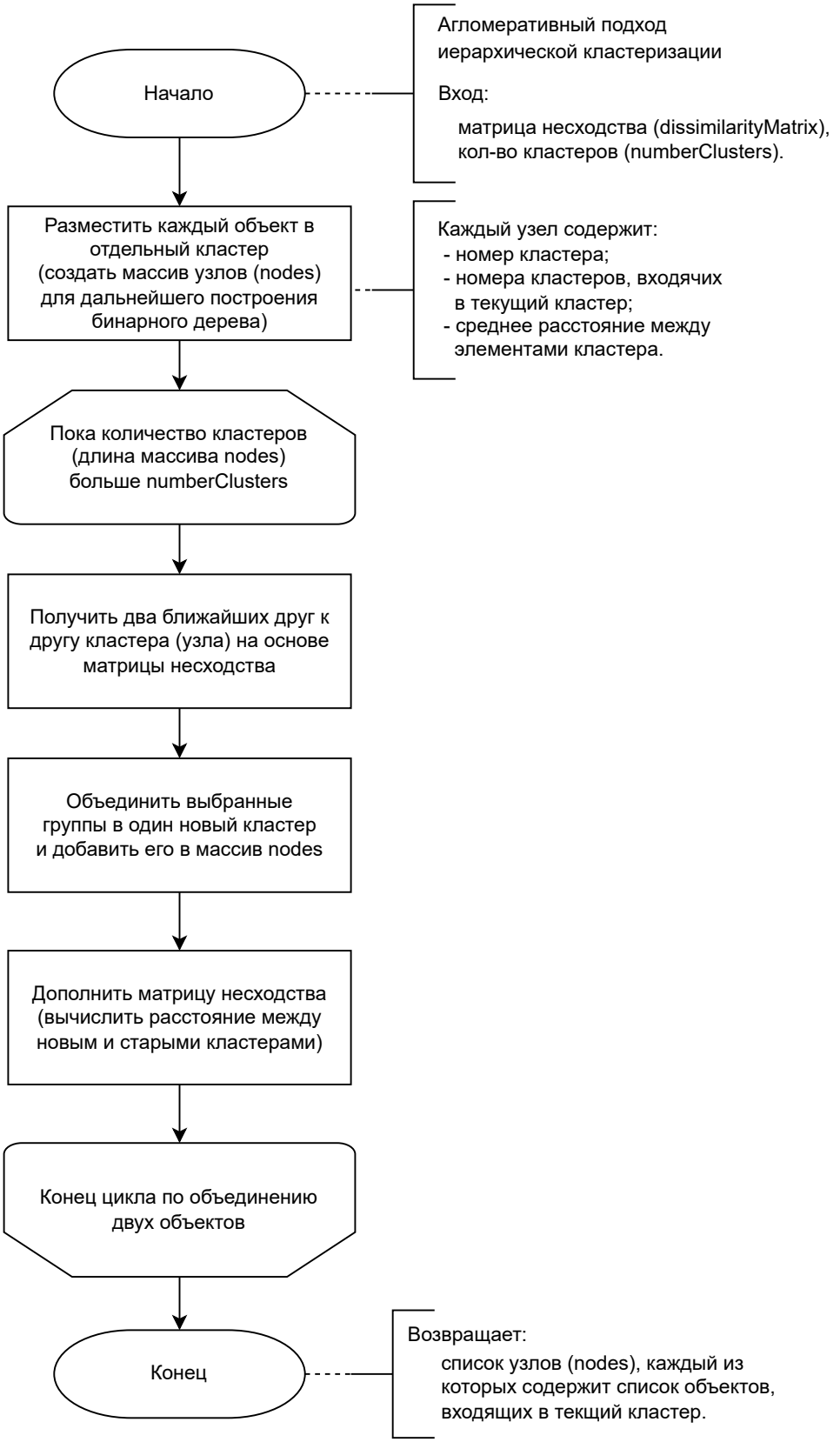

Рисунок 2.4 – Схема иерархической части гибридного метода кластеризации

Схема центроидной части гибридного метода кластеризации представлена на рисунке 2.5. Это последний этап гибридной кластеризации, в результате которого будет уточнена принадлежность элементов к кластерам.

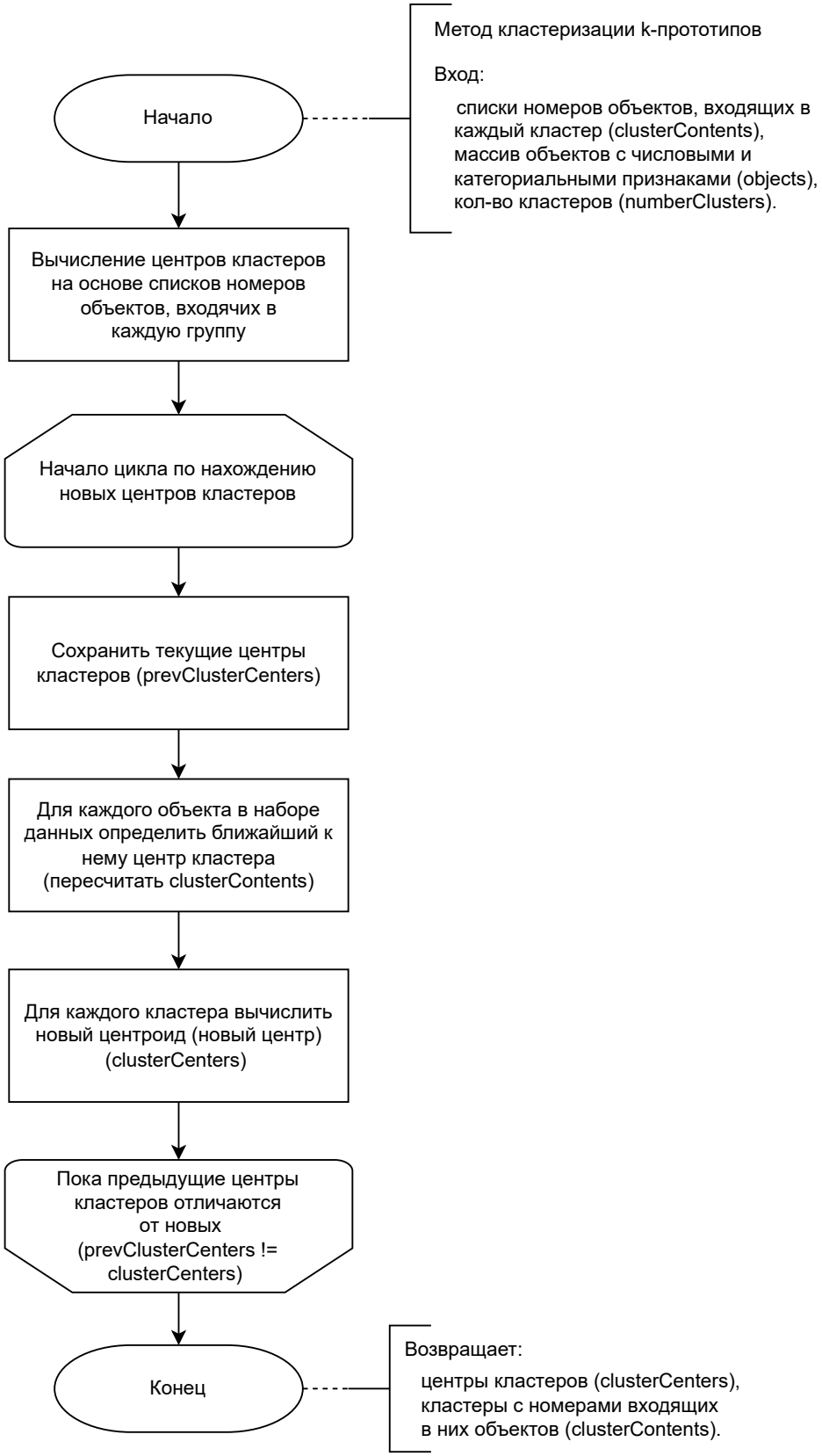

Рисунок 2.5 – Схема центроидной части гибридного метода кластеризации

Вывод

В данном разделе были предъявлены требования к разрабатываемому методу разбиения данных и к разрабатываемому ПО. Было произведено проектирование метода кластеризации. В качестве критерия связи кластеров в иерархической части метода была выбрана полная связь. Для вычисления матрицы несходства в качестве меры расстояний было выбрано расстояние Говера. Кроме того, в данном разделе были построены схемы для реализации гибридного метода разбиения.

3 Технологический раздел

3.1 Средства реализации ПО

В качестве языка программирования был выбран `Python` [24]. Это обусловлено наличием опыта работы с выбранным языком. Также для `Python` существует большое количество библиотек и документация на русском языке, а сам язык поддерживает объектно-ориентированную парадигму программирования.

При создании графического интерфейса для программного обеспечения была использована библиотека `tkinter` [25]. Она является кроссплатформенной и включена в стандартную библиотеку языка `Python` в виде отдельного модуля.

Для визуализации работы методов разбиения использовалась библиотека `matplotlib` [26] с модулем `matplotlib.pyplot` [27]. Для графического представления бинарного дерева, полученного в результате работы агломеративного подхода иерархического метода кластеризации, строилась дендрограмма с помощью модуля `cluster.hierarchy` [28] библиотеки `scipy` [29]. Также для построения таблиц с результатами разбиения использовалась библиотека `prettytable` [30].

3.2 Формат входных и выходных данных

На вход программа получает файл в формате `csv`, содержащий данные как с категориальными, так и с числовые признаками. Перед началом разбиения необходимо знать номера столбцов с количественными параметрами, а также диапазоны их значений. Возможность выбирать входной файл через графический интерфейс не предусмотрена. Пользователь на вход подает следующие параметры:

— количество итоговых кластеров;

— количество обрабатываемых объектов;

— количество прогонов для метода разбиения k-прототипов при сравнении методов.

Число обрабатываемых объетов не должно превышать количество строк в

входном файле, число итоговых кластеров должно быть не больше количества объетов. Все входные параметры должны являться натуральными числами.

На выходе будет построена таблица с результатами выбранного метода разбиения. Например, для гибридного метода будет выведена таблица полученных кластеров с их центрами и номерами входящих в них объектов. Также будет построен график, визуализирующий результаты разбиения. При проведении оценки качества кластеризации на выходе получим график сравнения методов, а также таблицу с вычисленными коэффициентами для каждого случая сравнения.

3.3 Реализация гибридного метода разбиения

Реализация гибридного метода кластеризации будет состоять из четырех основных этапов:

1. Построение матрицы несходства.
2. Применение агломеративного подхода иерархической кластеризации.
3. Получение списка номеров объектов, входящих в каждый кластер.
4. Уточнение принадлежности элементов к кластерам с помощью метода кластеризации k-прототипов.

На вход разрабатываемый метод получает массив объектов с числовыми и категориальными признаками, а также итоговое количество кластеров. Реализация класса `HybridClusterization` для гибридного метода разбиения данных на основе агломеративного подхода иерархической кластеризации представлена в листинге А.1.

На первом этапе необходимо построить матрицу несходства, которая необходима для определения степени различия между объектами и подается на вход следующему этапу кластеризации. Для определения расстояния между элементами матрицы используется расстояние Говера. Стоит отметить, что матрица симметрична относительно главной диагонали, поэтому достаточно построить только верхнюю ее часть, нижнюю же можно будет заполнить отображением. В листинге А.2 представлена реализация класса `Distance` для построения матрицы несходства и вычисления расстояния Говера между объектами.

На втором этапе матрица несходства передается в алгоритм агломеративного подхода иерархической кластеризации. В результате работы которого уже будут получены кластеры в виде списка узлов неполного бинарного дерева. В качестве критерия связи групп будет использована полная связь, то есть расстояние между кластерами будет считаться как самое длинное расстояние между двумя точками в каждом кластере. Данный этап считается основным, от результатов его работы во многом будет зависеть результат разрабатываемого метода. Реализация класса `HAClusterization` для иерархической части гибридного метода кластеризации представлена в листинге А.3.

Третий этап считается переходным от второго к четвертому. Он необходим для преобразования узлов неполного бинарного дерева в списки номеров объектов, входящих в каждый кластер.

Четвертый этап кластеризации отвечает за уточнение полученных ранее групп. Для вычисления расстояния от объектов до центроидов (центров кластеров) также будет использовано расстояние Говера. Данный этап является последним в разрабатываемом алгоритме. В итоге гибридный метод разбиения вернет центроиды и списки кластеров с номерами входящих в них объектов. Реализация класса `KPrototypesClusterization` для уточнения принадлежности элементов кластерам с помощью метода кластеризации центроидного типа представлена в листинге А.4.

Чтобы не перегружать предоставленные листинги, из всех реализованных классов были убраны методы и переменные, отвечающие за графическое отображение результатов разбиения.

3.4 Результаты работы ПО

Взаимодействие пользователя с ПО осуществляется с помощью графического интерфейса (рисунок 3.1), в котором предусмотрена возможность изменения количества обрабатываемых объектов, итогового числа кластеров, а также количества прогонов для k-прототипов при сравнении методов. При кластеризации данных пользователю предоставляется возможность выбирать один из трех реализованных алгоритмов разбиения. Также в ПО предусмотрена возможность оценки качества кластеризации с помощью метода оценки силуэтов и метода локтя.

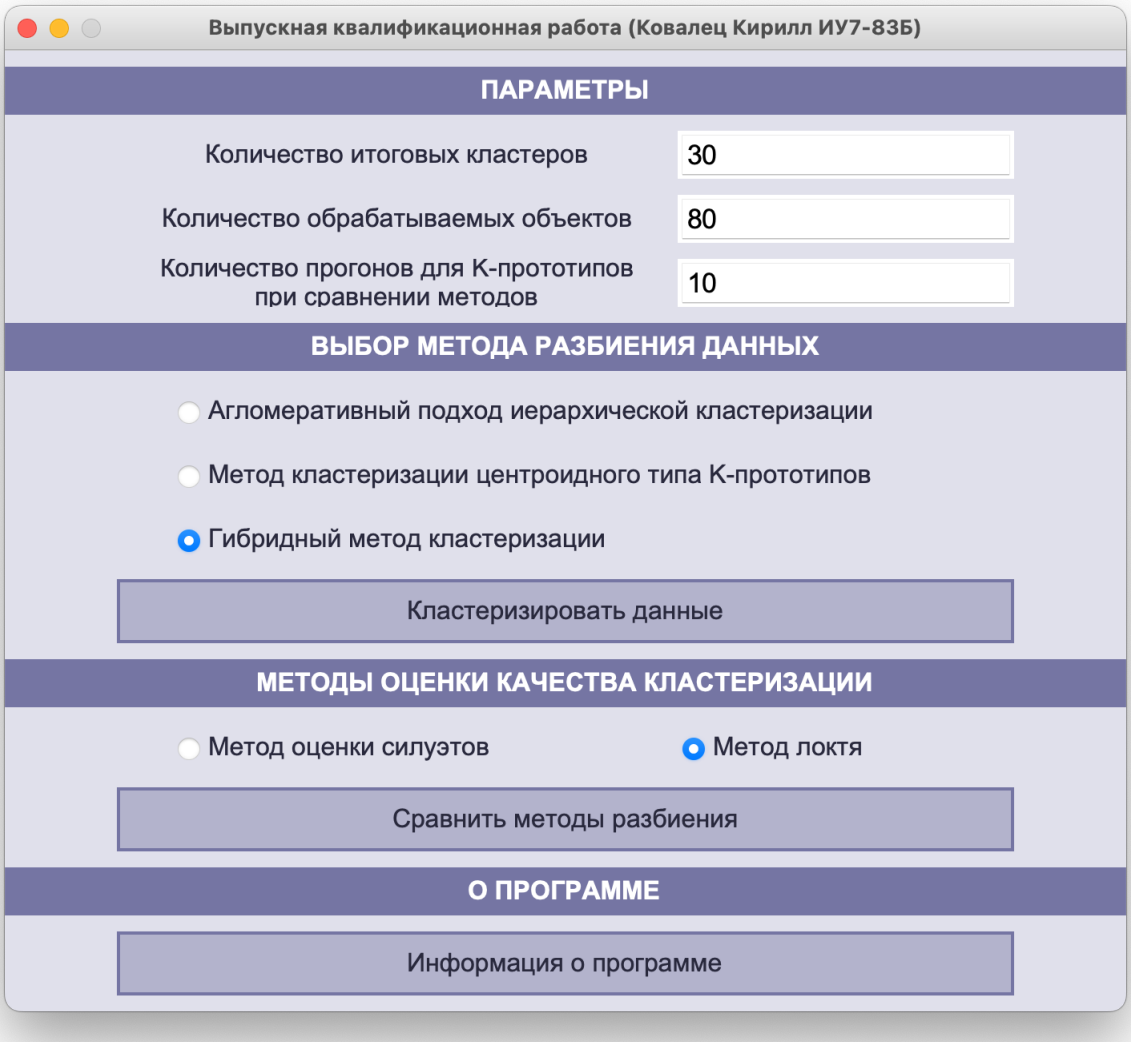

Рисунок 3.1 – Графический интерфейс разработанного ПО

Основной особенностью иерархических методов кластеризации являются возвращаемые ими значения. В отличие от центроидных методов, которые возвращают списки образованных кластеров с их центрами и номерами входящих в них объектов, иерархические возвращают бинарное дерево кластеров, для визуализации которого используют дендрограмму. Результат работы агломеративного подхода иерархической кластеризации представлен на рисунке 3.2.

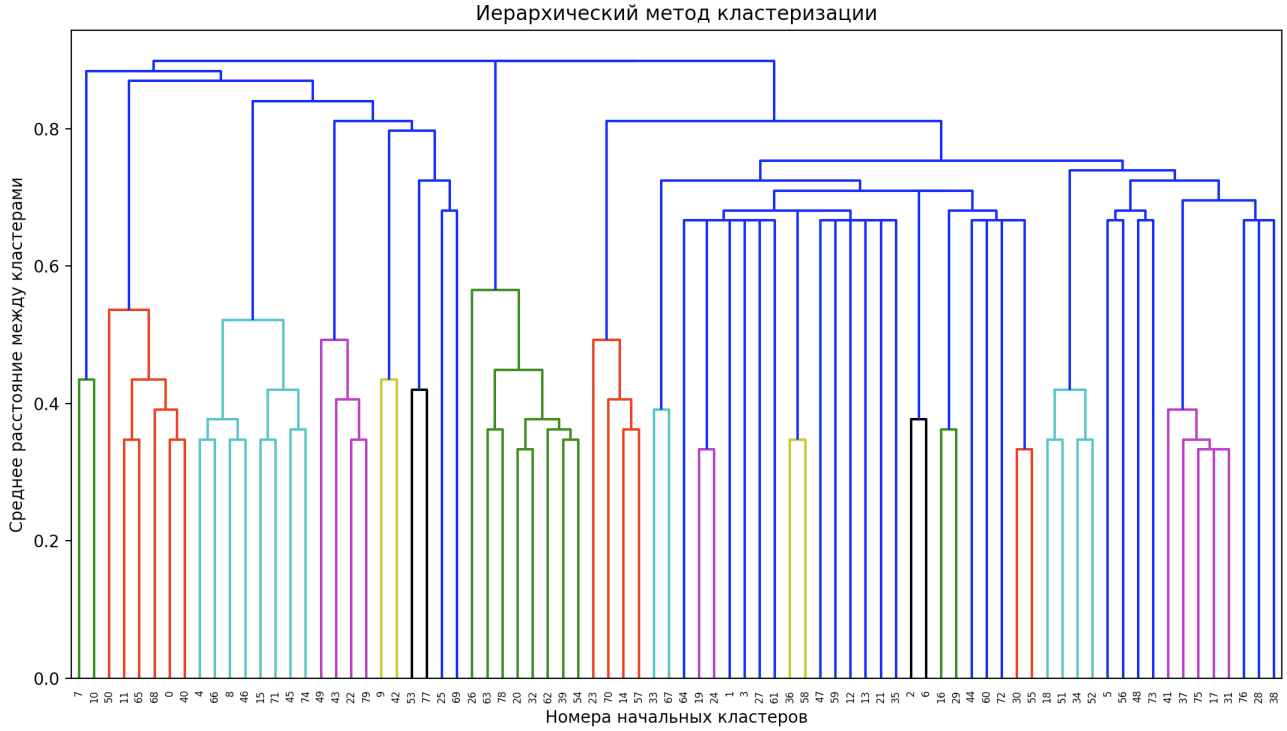

Рисунок 3.2 – Дендрограмма агломеративного подхода иерархической кластеризации

По полученной дендрограмме можно сделать вывод о том, что для кластеризации использовались данные, находящиеся друг от друга на расстоянии не менее 0.33.

Для центроидных методов разбиения возникла проблема визуализации результатов работы. Так как кластеризирумые данные содержат числовые и категориальные признаки, их нельзя представить в виде точек в n-мерном пространстве. Поэтому было принято решение продемонстрировать расстояния от объектов до центров их кластеров. Чем больше объектов находится на меньшем расстоянии до центра, тем лучше.

На рисунках 3.3–3.4 представлены результаты работы метода разбиения k-прототипов для 30 и 60 итоговых кластеров.

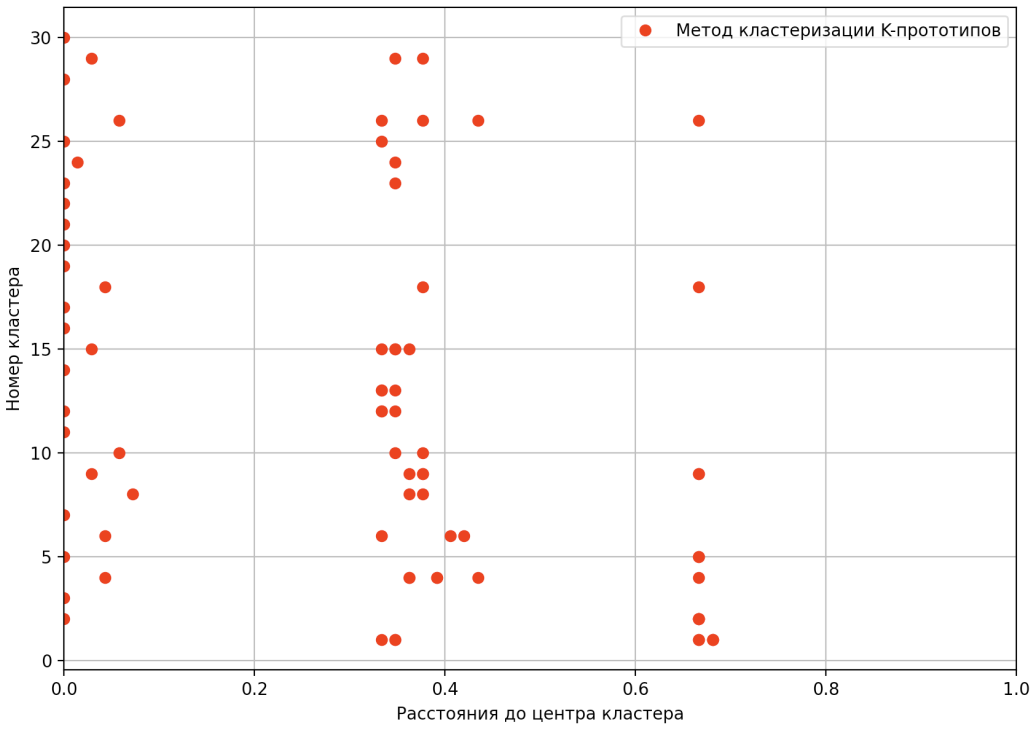

Рисунок 3.3 – Результаты кластеризации методом центроидного типа k-прототипов для 30 кластеров

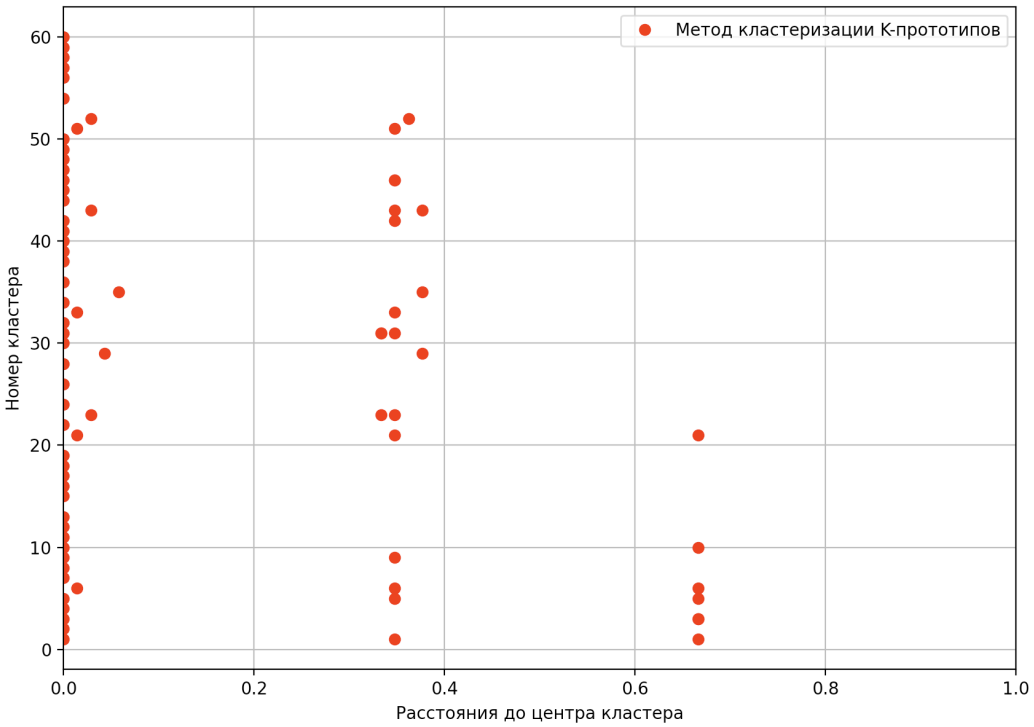

Рисунок 3.4 – Результаты кластеризации методом центроидного типа k-прототипов для 60 кластеров

На рисунках 3.5–3.6 представлены результаты работы разработанного гибридный метода разбиения для 30 и 60 итоговых кластеров.

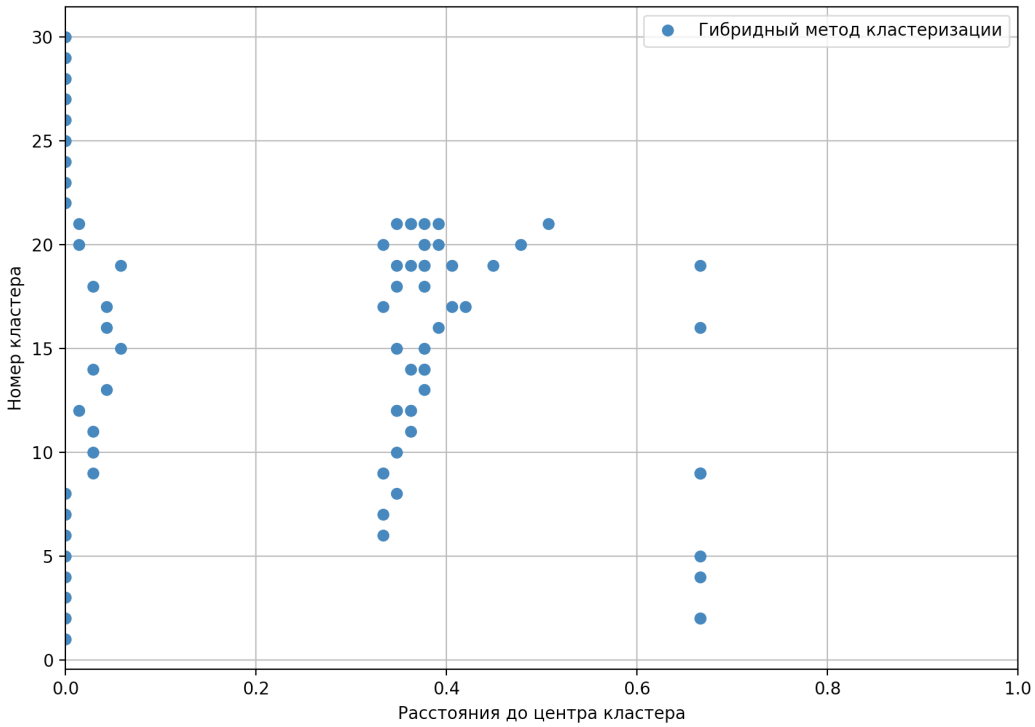

Рисунок 3.5 – Результаты кластеризации гибридным методом для 30 кластеров

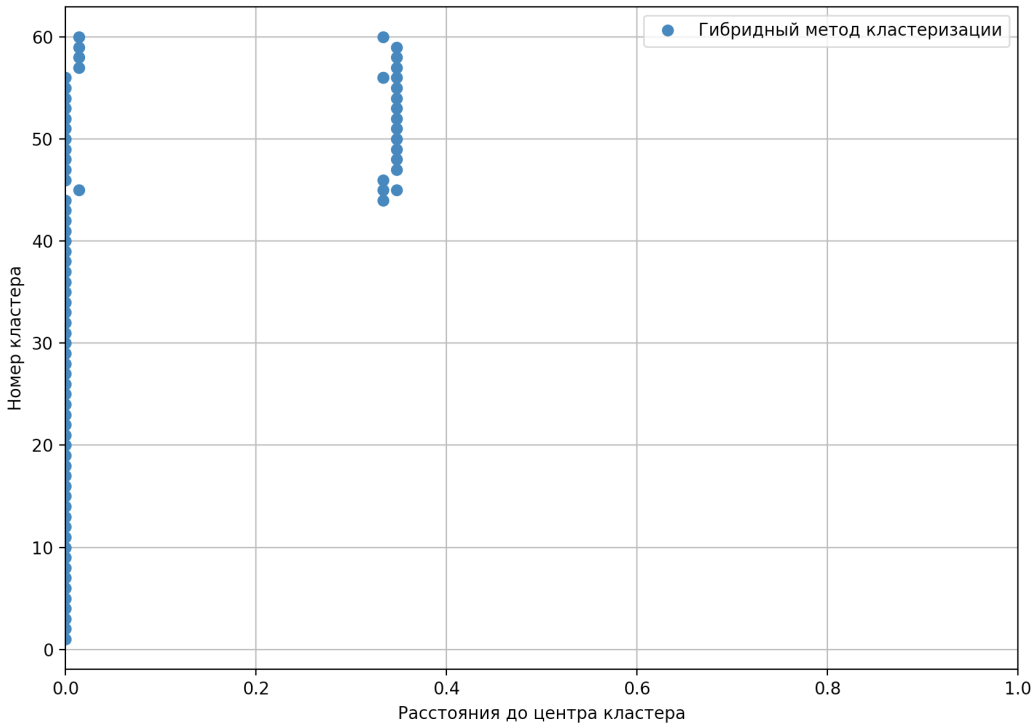

Рисунок 3.6 – Результаты кластеризации гибридным методом для 60 кластеров

Вывод

В данном разделе были рассмотрены средства реализации ПО, описан формат входных и выходных данных, реализован гибридный метод разбиения данных, а также приведены результаты работы ПО.

Метод разбиения k-прототипов и гибридный метод относятся к центроидным алгоритмам разбиения. Однако результаты кластеризации обоих методов для 60 кластеров сильно отличаюся. Гибридный метод показывает лучшие результаты, так как объекты находятся ближе к центрам кластеров, чем у k-прототипов (при проведении оценки качества кластеризации методом локтя среднее внутрикластерное расстояние для гибридного метода составило 0.1, в то время как для k-прототипов — 0.17). Это связано с рандомной инициализацией начальных центроидов в методе k-прототипов. В гибридном алгоритме центроиды вычисляются на основе кластеров, полученных в результате работы иерархической части метода.

4 Исследовательский раздел

4.1 Применение методов оценки качества кластеризации

4.1.1 Метод локтя

Данный метод оценки качества кластеризации применяется в тех случаях, когда важна компактность кластеров. Он оценивает среднее расстояние между объектами внутри групп. Метод локтя используют для определения оптимального количества кластеров путем нахождения на графике точки, после которой уменьшение среднего расстояния между элементами в группе будет не так заметно.

Реализация данного метода оценки качества кластеризации представлена в листинге Б.1. На рисунке 4.1 показаны результаты сравнения трех реализованных методов разбиения.

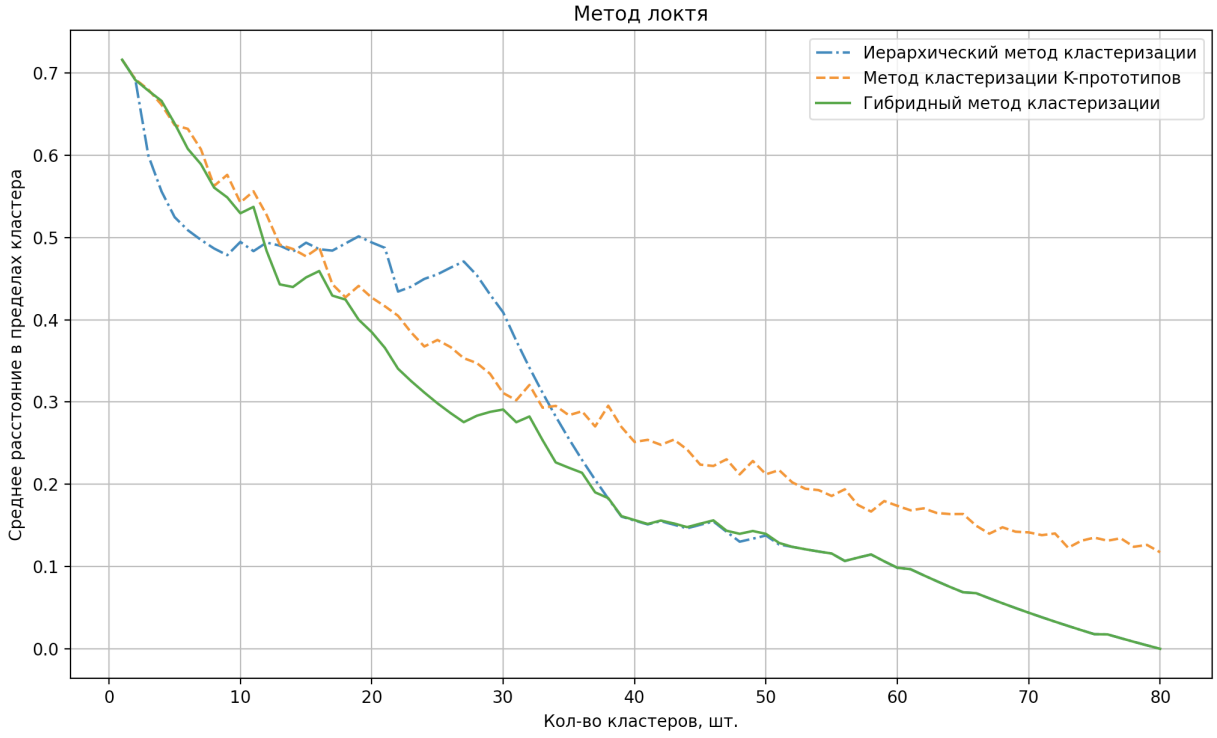

Рисунок 4.1 – Сравнение методов разбиения с помощью метода локтя

Результаты рассмотренного метода оценки качества разбиения приведены в таблице 4.1. В ней указаны только те строки, значения которых использовались в анализе результатов сравнения. Данная таблица показывает средние расстояния между элементами в кластерах, образованных разными

методами.

Таблица 4.1 – Результаты проведения оценки качества разбиения методом локтя

Кол-во кластеров	**Иерархический**	**K-прототипов**	**Гибридный**
11	0.483	0.552	0.537
12	0.494	0.531	0.484
13	0.49	0.53	0.443
26	0.463	0.345	0.287
27	0.471	0.347	0.276
28	0.454	0.344	0.283
30	0.409	0.311	0.291
33	0.311	0.295	0.254
34	0.282	0.292	0.227
35	0.255	0.291	0.22
36	0.23	0.291	0.214
37	0.206	0.269	0.19
38	0.183	0.284	0.183
39	0.161	0.26	0.161
40	0.156	0.258	0.157
41	0.151	0.264	0.152
42	0.155	0.227	0.156
60	0.099	0.166	0.099

Гибридный метод показал лучшие результаты при итоговом количестве кластеров от 12 до 37. Максимальная разница наблюдалась при 27 группах (у гибридного метода среднее внутрикластерное расстояние было в 1.26 раз меньше, чем у k-прототипов и в 1.71 раз меньше, чем у иерархического). При дальнейшем увеличении групп среднее расстояние для иерархического и гибридного методов было практически одинаковым. Это связано с рандомной инициализацией центров кластеров в методе k-прототипов, из-за чего центроидный метод показывал худшие результаты, чем иерархический, начиная с 34 групп (в 1.65 раз хуже при 40 и в 1.68 раз хуже при 60). Из-за этого уточнение значений на последнем этапе гибридного метода стало бесполезным. Отсюда следует, что гибридный метод разбиения данных надо применять

в тех случаях, когда метод k-прототипов показывает лучшие результаты в сравнении с иерархическим.

По результатам оценки качества кластеризации методом локтя можно сделать вывод о том, что оптимальным количеством групп будет 39, так как далее среднее расстояние будет уменьшаться незначительно (в 1.03 раза для 40 и 41 кластеров, тогда как для 38 и 39 кластеров оно уменьшилось в 1.13 и в 1.14 раз соответственно). Оценивались результаты иерархического метода разбиения, так как на них лучше всего видна точка «локтя».

4.1.2 Метод оценки силуэтов

Данный метод оценки качества кластеризации применяется в тех случаях, когда важно среднее расстояние между группами. Для этого вычисляется коэффициент силуэта в диапазоне $[-1, 1]$, определяющий, насколько близко каждая точка внутри одного кластера расположена к точкам ближайшей соседней группы. Чем ближе значение к 1, тем лучше кластеризированы объекты.

Реализация данного метода оценки качества кластеризации представлена в листинге В.1. На рисунке 4.2 показаны результаты сравнения трех реализованных методов разбиения.

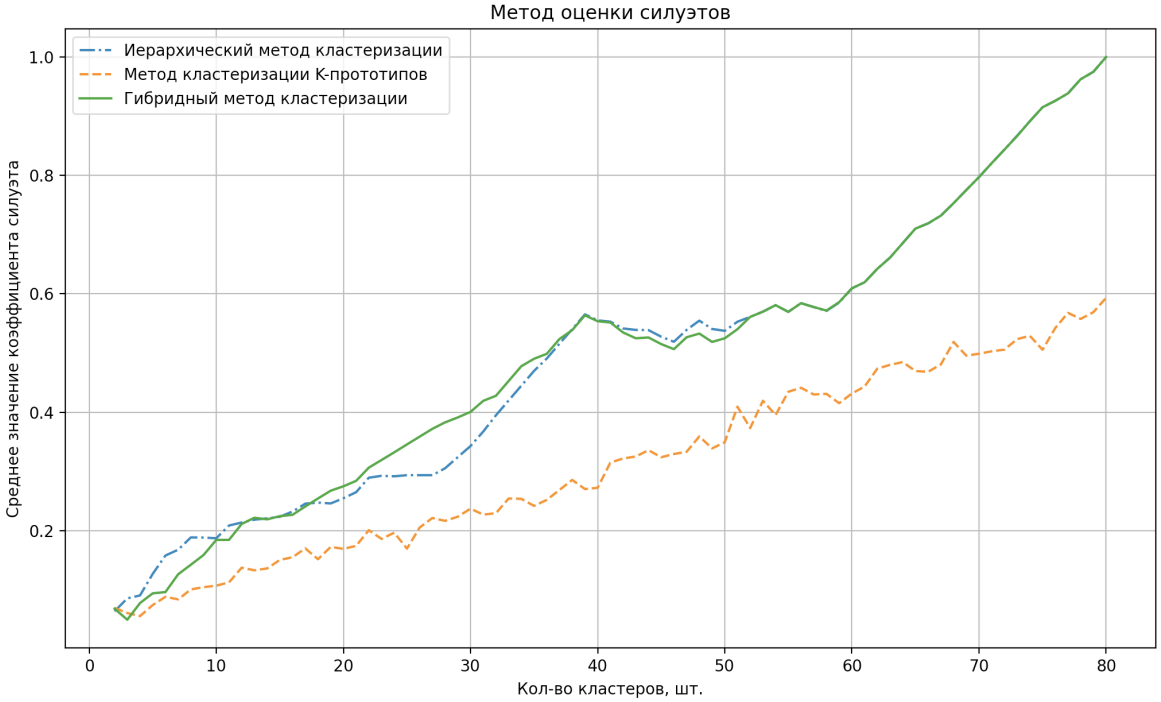

Рисунок 4.2 – Сравнение методов разбиения с помощью метода оценки силуэтов

Результаты метода оценки силуэтов приведены в таблице 4.2. В ней указаны только те строки, значения которых использовались в анализе результатов сравнения. Данная таблица показывает средние значения коэффициентов силуэта для элементов в кластерах, образованных разными методами.

Таблица 4.2 – Результаты проведения оценки качества разбиения методом оценки силуэтов

Кол-во кластеров	**Иерархический**	**K-прототипов**	**Гибридный**
17	0.246	0.165	0.241
18	0.247	0.174	0.254
19	0.246	0.164	0.268
30	0.343	0.257	0.401
36	0.491	0.25	0.499
37	0.516	0.295	0.524
38	0.54	0.273	0.539
39	0.565	0.285	0.564
40	0.555	0.283	0.554
52	0.561	0.427	0.561
53	0.57	0.385	0.57
54	0.581	0.4	0.581
60	0.609	0.417	0.609
80	1.0	0.588	1.0

Метод кластеризации k-прототипов показал наихудшие результаты в сравнение. Это связано с рандомной инициализацией центроидов, из-за чего некоторое кластеры образовывались близко друг к другу. Также принцип случайности при инициализации приводил к тому, что элементы неравномерно распределялись по группам, некоторые из которых вообще оставались пустыми. В отличие от центроидного метода, иерархический и гибридный изначально каждый объект закрепляли за своим кластером. Такой подход гарантировал, что пустых групп оставаться не будет.

Гибридный метод показал наилучшие результаты в промежутке от 18 до 37 итоговых кластеров. Максимальная разница наблюдалась при 27 группах (у гибридного метода коэффициент силуэта был в 1.27 раз больше, чем у иерархического и в 2.02 раза больше, чем у k-прототипов). Стоит отметить,

что когда число кластеров совпало с числом объектов, то среднее значение коэффициента силуэта для гибридного и иерархического методов равнялось 1. Это показывает, что каждый объект находился в своем кластере, и что пустых групп не было.

По результатам оценки качества кластеризации методом оценки силуэтов можно сделать вывод о том, что оптимальным количеством групп будет 39, так как в данной точке на графике наблюдается экстремум, после которого коэффициент силуэта вновь начинает расти относительно точки максимума только при 53 кластерах. В данной точке экстремума среднее значение коэффициента силуэта для гибридного и иерархического методов будет в 1.98 раз превышать показатель метода k-прототипов.

Выводы

В данном разделе проводилось сравнение реализоваанных алгоритмов разбиения с помощью существующих методов оценки качества кластеризации: метода локтя, метода оценки силуэтов.

Метод локтя показывает среднее расстояния между объектами внутри групп. Когда в наборе данных существует несколько объектов, идентичных относительно друг друга по вычисленному расстоянию (сами объекты необязательно должны быть одинаковыми), то центроидный метод разбиения будет возвращать более компактные кластеры, чем иерархический. Именно в таких случаях для достижения максимальной компактности лучше всего использовать гибридный метод кластеризации, так как он использует центроиды для уточнения полученных значений и не применяет рандомную инициализацию первичных центров кластеров, из-за чего является более предпочтительным в сравнении с иерархическим и k-прототипов.

В результате использования метода оценки силуэтов было выяснено, что максимально разделенные кластеры получаются в результате работы иерархического и гибридного методов разбиения. Гибридный также будет предпочтительнее (средние значения коэффициентов силуэта будут больше), когда в наборе данных существует несколько обособленных объектов, равноудаленных друг от друга.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы надо было разработать метод разбиения категориальных данных на основе агломеративного подхода иерархической кластеризации. Поставленная цель была достигнута. Также были выполнены следующие задачи.

1. Описаны существующие методы кластеризации данных (иерархический, k-средних, k-режимов, k-прототипов, C-средних, DBSCAN, МПД) и проведено их сравнение по выделенным критериям. Для уточнения результатов иерархического разбиения в гибридном методе был выбран центроидный метод k-прототипов.

2. Описаны существующие критерии связи кластеров (одиночная, полная, средняя) в иерархическом методе разбиения данных. При создании гибридного метода была использована полная связь, которая вычисляет расстояние между двумя кластерами как самое длинное расстояние между двумя точками в каждой группе. Полная связь была выбрана, так как она обеспечивает более компактные кластеры, чем одиночная или средняя.

3. Рассмотрены существующие меры расстояний между объектами (Евклидово расстояние, квадрат Евклидово расстояния, расстояние городских кварталов, расстояние Чебышева, расстояние Минковского, степенное расстояние, расстояние Хэмминга, расстояние Говера) и проведено их сравнение по возможности обрабатывать данные, содержащие категориальные и числовые признаки. Для гибридного метода кластеризации было выбрано расстояние Говера, так как оно едиственное из рассмотренных может обрабатывать оба типа признаков.

4. Описаны методы оценки качества кластеризации (метод локтя, метод оценки силуэтов). Первый используют, когда важна компактность кластеров, второй, когда важно среднее расстояние между группами.

5. Разработан метод разбиения данных на основе агломеративного подхода иерархической кластеризации.

6. Разработано программное обеспечение для демонстрации работы созданного метода.

7. Проведено сравнение разработанного метода разбиения с аналогами с помощью существующих методов оценки качества кластеризации. С помощью метода локтя было выяснено, что когда в наборе данных существует несколько обособленных объектов, равноудаленных друг от друга, то для достижения максимальной компактности лучше всего использовать гибридный метод разбиения. В результате использования метода оценки силуэтов было показано, что максимально разделенные кластеры получаются в результате работы иерархического и гибридного методов разбиения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Кугаевских А.* КЛАССИЧЕСКИЕ МЕТОДЫ МАШИННОГО ОБУЧЕНИЯ // Санкт-Петербург, ИТМО. — 2022. — С. 13—14.
2. *Поручиков М.* АНАЛИЗ ДАННЫХ // Самара, Самарский университет им. С.П. Королева. — 2016. — С. 9.
3. *Черезов Д.* ОБЗОР ОСНОВНЫХ МЕТОДОВ КЛАССИФИКАЦИИ И КЛАСТЕРИЗАЦИИ ДАННЫХ // Воронеж, Воронежский государственный университет. — 2020. — С. 28.
4. Understanding the concept of Hierarchical clustering Technique [Электронный ресурс]. — Режим доступа URL: `https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec` (Дата обращения: 12.04.2023).
5. K-means Clustering: Algorithm, Applications, Evaluation Methods, and Drawbacks [Электронный ресурс]. — Режим доступа URL: `https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a` (Дата обращения: 12.04.2023).
6. The k-modes as Clustering Algorithm for Categorical Data Type [Электронный ресурс]. — Режим доступа URL: `https://medium.com/geekculture/the-k-modes-as-clustering-algorithm-for-categorical-data-type-bcde8f95efd7` (Дата обращения: 12.04.2023).
7. *Jia Z.* Weighted k-Prototypes Clustering Algorithm Based on the Hybrid Dissimilarity Coefficient // Гуанчжоу, Nanfang College of Sun Yat-sen University. — 2020. — С. 2—4.
8. *Rajkumar K.* Fuzzy clustering and Fuzzy C-Means partition cluster analysis and validation studies on a subset of CiteScore dataset // International Journal of Electrical and Computer Engineering (IJECE). — 2019. — С. 18.
9. *Liu Z.* A Density-Based Spatial Clustering of Application with Noise Algorithm and its Empirical Research // Highlights in Science, Engineering and Technology. — 2022. — С. 174—178.
10. *Ершов К.* Анализ и классификация алгоритмов кластеризации // Москва, МГТУ им. Н.Э. Баумана. — 2016. — С. 274—277.

11. Minimum Spanning Tree Tutorials [Электронный ресурс]. — Режим доступа URL: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/ (Дата обращения: 12.04.2023).

12. Hierarchical Clustering / Dendrogram: Simple Definition, Examples [Электронный ресурс]. — Режим доступа URL: https://www.statisticshowto.com/hierarchical-clustering/ (Дата обращения: 18.04.2023).

13. Single-Link clustering clearly explained [Электронный ресурс]. — Режим доступа URL: https://harikabonthu96.medium.com/single-link-clustering-clearly-explained-90dff58db5cb (Дата обращения: 18.04.2023).

14. Manual Step by Step Complete Link hierarchical clustering with dendrogram [Электронный ресурс]. — Режим доступа URL: https://medium.com/analytics-vidhya/manual-step-by-step-complete-link-hierarchical-clustering-with-dendrogram-210c57b6afbf (Дата обращения: 18.04.2023).

15. Average Linkage Clustering [Электронный ресурс]. — Режим доступа URL: https://www.statistics.com/glossary/average-linkage-clustering/ (Дата обращения: 18.04.2023).

16. Евклидово расстояние в анализе кластерного типа [Электронный ресурс]. — Режим доступа URL: https://otus.ru/journal/evklidovo-rasstoyanie-v-analize-klasternogo-tipa/ (Дата обращения: 20.04.2023).

17. Euclidean and Manhattan distance metrics in Machine Learning [Электронный ресурс]. — Режим доступа URL: https://medium.com/analytics-vidhya/euclidean-and-manhattan-distance-metrics-in-machine-learning-a5942a8c9f2f (Дата обращения: 20.04.2023).

18. 9 Distance Measures in Data Science [Электронный ресурс]. — Режим доступа URL: https://towardsdatascience.com/9-distance-measures-in-data-science-918109d069fa (Дата обращения: 20.04.2023).

19. Расстояние Минковского [Электронный ресурс]. — Режим доступа URL: http://poivs.tsput.ru/ru/Math/Analysis/FunctionalAnalysis/DistanceMinkowski (Дата обращения: 20.04.2023).

20. What is Hamming Distance — Tutorialspoint [Электронный ресурс]. — Режим доступа URL: https://www.tutorialspoint.com/what-is-hamming-distance (Дата обращения: 20.04.2023).

21. Gower's Distance — Medium [Электронный ресурс]. — Режим доступа URL: https://medium.com/analytics-vidhya/gowers-distance-899f9c4bd553 (Дата обращения: 20.04.2023).

22. Метод локтя (Elbow Rule) [Электронный ресурс]. — Режим доступа URL: https://www.helenkapatsa.ru/mietod-loktia/ (Дата обращения: 22.04.2023).

23. Метод силуэта (Silhouette Method) [Электронный ресурс]. — Режим доступа URL: https://www.helenkapatsa.ru/silhouette-method/ (Дата обращения: 22.04.2023).

24. Welcome to Python [Электронный ресурс]. — Режим доступа URL: https://www.python.org (Дата обращения: 25.04.2023).

25. Tkinter — Python interface to Tcl/Tk [Электронный ресурс]. — Режим доступа URL: https://docs.python.org/3/library/tkinter.html (Дата обращения: 25.04.2023).

26. Matplotlib — Visualization with Python [Электронный ресурс]. — Режим доступа URL: https://matplotlib.org (Дата обращения: 25.04.2023).

27. matplotlib.pyplot — Matplotlib 3.5.3 documentation [Электронный ресурс]. — Режим доступа URL: https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html (Дата обращения: 25.04.2023).

28. Hierarchical clustering (scipy.cluster.hierarchy) [Электронный ресурс]. — Режим доступа URL: https://docs.scipy.org/doc/scipy/reference/cluster.hierarchy.html (Дата обращения: 30.04.2023).

29. SciPy [Электронный ресурс]. — Режим доступа URL: https://scipy.org (Дата обращения: 30.04.2023).

30. Модуль PrettyTable в Python, вывод табличных данных [Электронный ресурс]. — Режим доступа URL: https://docs-python.ru/packages/modul-prettytable-python/ (Дата обращения: 16.05.2023).

ПРИЛОЖЕНИЕ А
Реализация гибридного метода разбиения

Листинг А.1 — Реализация класса HybridClusterization

```
class HybridClusterization():
    objects: List[List[str]] # список объектов
    numberClusters: int      # количество кластеров
    clusterContents: List[List[int]] # массив, содержащий в себе k массивов с
    ↪ номерами элементов, принадлежащих соответствующему кластеру
    clusterCenters: List[List[Union[str, int]]] # массив, содержащий в себе k
    ↪ центров кластеров

    def __init__(self, objects: List[List[str]], numberClusters: int):
        self.objects = objects
        self.numberClusters = numberClusters
        self.clusterContents, self.clusterCenters = self.run()

    def run(self) -> List[List[int]]:
        distance = Distance()
        # построение матрицы несходства
        dissimilarityMatrix = distance.createDissimilarityMatrix(self.objects)
        # иерархическая часть метода
        haClusterization = HAClusterization(dissimilarityMatrix,
        ↪ self.numberClusters)
        # получение списка номеров объектов, входящих в каждый кластер
        clusterContents = [node.listClusterNumbers for node in
        ↪ haClusterization.nodes]
        # уточнение принадлежности элементов кластерам с помощью метода
        ↪ кластеризации центроидного типа
        kPrototypesClusterization = \
            KPrototypesClusterization(self.objects, self.numberClusters,
            ↪ clusterContents)

        return kPrototypesClusterization.clusterContents, \
               kPrototypesClusterization.clusterCenters
```

Листинг А.2 — Реализация класса Distance

```
class Distance():
    def __init__(self):
        self.numericColumns = NUMERIC_COLUMNS
        self.rangesList = RANGES_LIST

    def goverDistance(self, elem1: List[str], elem2: List[str]) -> float:
        '''
```

Продолжение листинга А.2

```
        Возвращает расстояние Говера между двумя элементами
        '''
        sumCoeffDiff = 0
        numberColumns = len(elem1)

        for columnNumber in range(1, numberColumns):
            # если столбец содержит числовые данные
            try:
                i = self.numericColumns.index(columnNumber)
                coeffDiff = abs(int(elem1[columnNumber]) -
                ↪  int(elem2[columnNumber])) \
                    / self.rangesList[i]

            # если столбец содержит категориальные данные
            except ValueError:
                if elem1[columnNumber] == elem2[columnNumber]:
                    coeffDiff = 0
                else:
                    coeffDiff = 1

            sumCoeffDiff += coeffDiff

        return sumCoeffDiff / (numberColumns - 1)

    def createDissimilarityMatrix(self, objects: List[List[str]]) ->
    ↪  List[List[float]]:
        '''
        Построение матрицы несходства с помощью расстояния Говера
        '''
        numbObjects = len(objects)
        matrix = [ [0] * numbObjects for _ in range(numbObjects) ]

        for i in range(numbObjects - 1):
            for j in range(i + 1, numbObjects):
                matrix[i][j] = self.goverDistance(objects[i], objects[j])
                matrix[j][i] = matrix[i][j]

        return matrix
```

Листинг А.3 — Реализация класса HAClusterization

```
class Node():
    def __init__(self, clusterNumber: int, listClusterNumbers: List[int],
    ↪  avgDistance: float, left=None, right=None):
        self.clusterNumber = clusterNumber          # номер кластера
```

Продолжение листинга А.3

```
        self.listClusterNumbers = listClusterNumbers # список номеров кластеров,
        ↪ входящих в текщий кластер
        self.avgDistance = avgDistance               # среднее расстояние между
        ↪ элементами кластера

        self.left  = left
        self.right = right


class HAClusterization():
    dissimilarityMatrix: List[List[float]] # матрица несходства
    nodes: List[Node]                      # список узлов дерева
    tree: Union[Node, None]                # корень дерева

    def __init__(self, dissimilarityMatrix: List[List[float]], numberClusters:
    ↪ int=1):
        self.dissimilarityMatrix = dissimilarityMatrix
        self.nodes = []
        self.addNodes()
        self.tree = self.buildTree(numberClusters)

    def addNodes(self) -> None:
        '''
        Размещение каждого объекта в отдельный кластер
        '''
        for i in range(len(self.dissimilarityMatrix)):
            self.nodes.append(Node(i, [i], 0))

    def nodeClusterNumber(self, nodeIndex: int) -> int:
        '''
        Возвращает номер кластера узла
        '''
        return self.nodes[nodeIndex].clusterNumber

    def getClusterNumbersWithMinDistance(self) -> List[int]:
        '''
        Возвращает индексы узлов, содержащих номера кластеров
        с минимальным растояние между ними
        '''
        iMinDistance = 0
        jMinDistance = 1

        for i in range(len(self.nodes) - 1):
            for j in range(i + 1, len(self.nodes)):
                if self.dissimilarityMatrix\
```

Продолжение листинга А.3

```
                    [self.nodeClusterNumber(i)]\
                    [self.nodeClusterNumber(j)] <\
                   self.dissimilarityMatrix\
                    [self.nodeClusterNumber(iMinDistance)]\
                    [self.nodeClusterNumber(jMinDistance)]:

                    iMinDistance = i
                    jMinDistance = j

        return iMinDistance, jMinDistance

    def calcAvgDistance(self, listClusterNumbers: List[int]) -> float:
        '''
        Возвращает среднее расстояние между элементами кластера
        '''
        sumDistances = 0
        countDistances = 0
        size = len(listClusterNumbers)

        for i in range(size - 1):
            for j in range(i + 1, size):
                sumDistances += self.dissimilarityMatrix\
                    [listClusterNumbers[i]][listClusterNumbers[j]]
                countDistances += 1

        return sumDistances / countDistances if countDistances != 0 else 0

    def calcAvgWithinClusterDistance(self) -> float:
        '''
        Возвращает среднее расстояние в пределах кластера
        '''
        sumAvgDistances = 0
        countAvgDistances = 0

        for node in self.nodes:
            sumAvgDistances += node.avgDistance
            countAvgDistances += 1

        return sumAvgDistances / countAvgDistances if countAvgDistances != 0 else
        ↪ 0

    def updateDissimilarityMatrix(self, clusterNumber1: int, clusterNumber2: int)
    ↪ -> None:
        '''
        Дополнение матрицы несходства
```

Продолжение листинга А.3

```
89          '''
90          # добавление столбца с расстояниями до нового кластера в матрицу несходства
91          for i in range(len(self.dissimilarityMatrix)):
92              # расстояние между двумя кластерами -- самое длинное расстояние между двумя точками в каждом кластере (полная связь)
93              maxDistance = max(self.dissimilarityMatrix[i][clusterNumber1],
94                                self.dissimilarityMatrix[i][clusterNumber2])
95
96              self.dissimilarityMatrix[i].append(maxDistance)
97
98          # добавление строки, симметричной добавленому столбцу
99          self.dissimilarityMatrix.append([self.dissimilarityMatrix[i][-1] for i in range(len(self.dissimilarityMatrix))])
100         self.dissimilarityMatrix[-1].append(0)
101
102     def buildTree(self, numberClusters: int) -> Union[Node, None]:
103         '''
104         Построение бинарного дерева кластеров
105         (если оно полное, то возвращается корень дерева)
106         '''
107         while len(self.nodes) > numberClusters:
108             iMinDistance, jMinDistance = self.getClusterNumbersWithMinDistance()
109
110             firstNode = self.nodes.pop(iMinDistance)
111             secondNode = self.nodes.pop(jMinDistance - 1)
112
113             listClusterNumbers = firstNode.listClusterNumbers
114             listClusterNumbers.extend(secondNode.listClusterNumbers)
115
116             self.nodes.append(
117                 Node(
118                     len(self.dissimilarityMatrix),
119                     listClusterNumbers,
120                     self.calcAvgDistance(listClusterNumbers),
121                     firstNode,
122                     secondNode
123                 )
124             )
125
126             self.updateDissimilarityMatrix(firstNode.clusterNumber, secondNode.clusterNumber)
127
128         return self.nodes[0] if numberClusters == 1 else None
```

Листинг А.4 — Реализация класса KPrototypesClusterization

```
class KPrototypesClusterization():
    objects: List[List[str]] # список объектов
    k: int                   # количество кластеров
    clusterContents: List[List[int]] # массив, содержащий в себе k массивов с номерами элементов, принадлежащих соответствующему кластеру
    clusterCenters: List[List[Union[str, int]]] # массив, содержащий в себе k центров кластеров

    def __init__(self, objects: List[List[str]], k: int, clusterContents: Union[List[List[int]], None]=None):
        self.objects = objects
        self.k = k
        self.clusterContents, self.clusterCenters = self.run(clusterContents)

    def generateClusterCenters(self) -> List[List[Union[str, int]]]:
        '''
        Возвращает массив с k центрами кластерами
        '''
        clusterCenters = []
        numbObjects = len(self.objects)
        numberColumns = len(self.objects[0])

        for _ in range(self.k):
            clusterCenter = []
            for columnNumber in range(numberColumns):
                field = self.objects[randint(0, numbObjects - 1)][columnNumber]
                clusterCenter.append(field)

            clusterCenters.append(clusterCenter)

        return clusterCenters

    def getAverageField(self, clusterNumbers: List[int], columnNumber: int) -> float:
        '''
        Возвращает среднее значение поля в столбце среди элементов кластера
        '''
        sumValue = 0
        for i in clusterNumbers:
            sumValue += int(self.objects[i][columnNumber])

        return sumValue / len(clusterNumbers)

    def getMostPopularField(self, clusterNumbers: List[int], columnNumber: int) -> str:
        '''
```

Продолжение листинга А.4

```
42          Возвращает самое встречаемое поле в столбце среди элементов кластера
43          '''
44          repetitions = {}
45          for i in clusterNumbers:
46              key = self.objects[i][columnNumber]
47
48              if key in repetitions.keys():
49                  repetitions[key] += 1
50              else:
51                  repetitions[key] = 1
52
53          maxRep = max(repetitions.values())
54          for key, value in repetitions.items():
55              if value == maxRep:
56                  return key
57
58      def recalculateClusterCenters(self,
59              oldClusterCenters: List[List[Union[str, float]]],
60              clusterContents: List[List[int]]) -> List[List[Union[str, float]]]:
61          '''
62          Возвращает массив с k центрами кластерами
63          '''
64          clusterCenters = []
65          numberColumns = len(self.objects[0])
66
67          for i in range(self.k):
68              if len(clusterContents[i]) > 0:
69                  clusterCenter = []
70                  for columnNumber in range(numberColumns):
71                      if columnNumber in NUMERIC_COLUMNS:
72                          field = self.getAverageField(clusterContents[i],
                            ↪  columnNumber)
73                      else:
74                          field = self.getMostPopularField(clusterContents[i],
                            ↪  columnNumber)
75
76                      clusterCenter.append(field)
77                  clusterCenters.append(clusterCenter)
78              else:
79                  # если кластер пуст
80                  clusterCenters.append(oldClusterCenters[i].copy())
81
82          return clusterCenters
83
```

Продолжение листинга А.4

```
84      def createMatrixDistances(self, clusterCenters: List[List[Union[str, float]]])
        ↪ -> List[List[float]]:
85          '''
86          Возвращает матрицу растояний от элементов до центров кластеров
87          '''
88          numbObjects = len(self.objects)
89          matrix = [ [0] * self.k for _ in range(numbObjects) ]
90          distance = Distance()
91
92          for i in range(numbObjects):
93              for j in range(self.k):
94                  matrix[i][j] = distance.goverDistance(self.objects[i],
                    ↪  clusterCenters[j])
95
96          return matrix
97
98      def distributeElemsIntoClusters(self, clusterCenters: List[List[Union[str,
        ↪ float]]]) -> List[List[int]]:
99          '''
100         Возвращает массив, содержащий в себе k массивов с номерами элементов,
    ↪   принадлежащих соответствующему кластеру
101         '''
102         clusterContents = [ [] for _ in range(self.k) ]
103         matrixDistances = self.createMatrixDistances(clusterCenters)
104
105         for i in range(len(matrixDistances)):
106             jMin = 0
107             minDistr = matrixDistances[i][jMin]
108
109             for j in range(1, self.k):
110                 if matrixDistances[i][j] < minDistr:
111                     jMin = j
112                     minDistr = matrixDistances[i][jMin]
113
114             clusterContents[jMin].append(i)
115
116         return clusterContents
117
118     def run(self, clusterContents: Union[List[List[int]], None]) ->
        ↪ List[List[int]]:
119         if clusterContents == None:
120             clusterCenters = self.generateClusterCenters()
121         else:
122             clusterCenters = self.recalculateClusterCenters([], clusterContents)
123
```

Продолжение листинга А.4

```
124             while(True):
125                 prevClusterCenters = clusterCenters.copy()
126                 clusterContents = self.distributeElemsIntoClusters(clusterCenters)
127 
128                 clusterCenters = self.recalculateClusterCenters(prevClusterCenters,
                    ↪  clusterContents)
129                 if prevClusterCenters == clusterCenters:
130                     break
131 
132             return clusterContents, clusterCenters
```

ПРИЛОЖЕНИЕ Б

Реализация оценки качества кластеризации методом локтя

Листинг Б.1 — Реализация класса TestElbow

```
class EventMethod(Enum):
    HA_CLUST     = 0
    KP_CLUST     = 1
    HYBRID_CLUST = 2


class TestElbow():
    def __init__(self, objects: List[List[str]], numberOfRuns: int):
        self.objects = objects
        self.numberOfRuns = numberOfRuns
        self.numberObjects = len(objects)
        self.dissimilarityMatrix = self.createDissimilarityMatrix()

    def createDissimilarityMatrix(self) -> List[List[float]]:
        '''
        Возвращает матрицу несходства
        '''
        distance = Distance()
        dissimilarityMatrix = distance.createDissimilarityMatrix(self.objects)

        return dissimilarityMatrix

    def calcAvgDistance(self, listObjectNumbers: List[int]) -> float:
        '''
        Возвращает среднее расстояние между элементами кластера
        '''
        sumDistances = 0
        countDistances = 0
        size = len(listObjectNumbers)

        for i in range(size - 1):
            for j in range(i + 1, size):
                sumDistances += self
                ↪  .dissimilarityMatrix[listObjectNumbers[i]][listObjectNumbers[j]]
                countDistances += 1

        return sumDistances / countDistances if countDistances else 0

    def calcAvgWithinClusterDistance(self, clusterContents: List[List[int]]) ->
    ↪  float:
```

Продолжение листинга Б.1

```
39          '''
40          Возвращает среднее расстояние в пределах кластера
41          '''
42          sumAvgDistances = 0
43          countAvgDistances = 0
44
45          for listObjectNumbers in clusterContents:
46              # если кластер содержит элементы
47              if len(listObjectNumbers) > 0:
48                  sumAvgDistances += self.calcAvgDistance(listObjectNumbers)
49                  countAvgDistances += 1
50
51          return sumAvgDistances / countAvgDistances if countAvgDistances else 0
52
53      def calcAvgWithinClusterDistanceForMethod(self, method: EventMethod,
        ↪ numberClusters: int) -> float:
54          '''
55          Возвращает среднее расстояние в пределах кластера
56          '''
57          if method == method.KP_CLUST:
58              sumAvgDistances = 0
59              for _ in range(self.numberOfRuns):
60                  kPrototypesClusterization = KPrototypesClusterization(self.objects,
                    ↪ numberClusters)
61                  sumAvgDistances +=
                    ↪ self.calcAvgWithinClusterDistance(kPrototypesClusterization
                    ↪ .clusterContents)
62
63              avgWithinClusterDistance = sumAvgDistances / self.numberOfRuns
64
65          elif method == method.HA_CLUST:
66              haClusterization = HAClusterization(self.dissimilarityMatrix.copy(),
                ↪ numberClusters)
67              # получение списка номеров объектов, входящих в каждый кластер
68              clusterContents = [node.listClusterNumbers for node in
                ↪ haClusterization.nodes]
69              avgWithinClusterDistance =
                ↪ self.calcAvgWithinClusterDistance(clusterContents)
70          else:
71              hybridClusterization = HybridClusterization(self.objects,
                ↪ numberClusters)
72              avgWithinClusterDistance =
                ↪ self.calcAvgWithinClusterDistance(hybridClusterization
                ↪ .clusterContents)
73
```

Продолжение листинга Б.1

```
74          return avgWithinClusterDistance
75
76      def comparisonMethods(self) -> None:
77          '''
78          Сравнение методов разбиения
79          '''
80          listAvgDistanceHA = []
81          listAvgDistanceKP = []
82          listAvgDistanceHybrid = []
83
84          print()
85          for k in range(1, self.numberObjects + 1):
86              listAvgDistanceHA.append(self
                ↪  .calcAvgWithinClusterDistanceForMethod(EventMethod.HA_CLUST,
                ↪  k))
87              listAvgDistanceKP.append(self
                ↪  .calcAvgWithinClusterDistanceForMethod(EventMethod.KP_CLUST,
                ↪  k))
88              listAvgDistanceHybrid.append(self
                ↪  .calcAvgWithinClusterDistanceForMethod(EventMethod.HYBRID_CLUST,
                ↪  k))
89
90              print("Проведено сравнение {:d} кластеров из {:d}".format(k,
                ↪  self.numberObjects))
91
92          listClusterNumbers = [i for i in range(1, self.numberObjects + 1)]
93
94          self.printAvgDistanceTable(
95              listClusterNumbers, listAvgDistanceHA,
96              listAvgDistanceKP, listAvgDistanceHybrid)
97          self.buildGraph(
98              listClusterNumbers, listAvgDistanceHA,
99              listAvgDistanceKP, listAvgDistanceHybrid)
100
101     @staticmethod
102     def printAvgDistanceTable(
103             listClusterNumbers: List[int],
104             listAvgDistanceHA: List[float],
105             listAvgDistanceKP: List[float],
106             listAvgDistanceHybrid: List[float]) -> None:
107         '''
108         Построение таблицы со средними расстояниями в пределах кластера
109         '''
110         tableHeader = ["Кол-во кластеров", "Иерархический", "К-прототипов",
            ↪  "Гибридный"]
```

Продолжение листинга Б.1

```
111             table = PrettyTable(tableHeader)
112 
113             for i in range(len(listClusterNumbers)):
114                 table.add_row([
115                     listClusterNumbers[i],
116                     round(listAvgDistanceHA[i], 3),
117                     round(listAvgDistanceKP[i], 3),
118                     round(listAvgDistanceHybrid[i], 3)
119                 ])
120             print("\n  --- Таблица со средними расстояниями в пределах кластера ---")
121             print(table)
122 
123         @staticmethod
124         def buildGraph(
125                 listClusterNumbers: List[int],
126                 listAvgDistanceHA: List[float],
127                 listAvgDistanceKP: List[float],
128                 listAvgDistanceHybrid: List[float]) -> None:
129             '''
130             Построение графика зависимости среднего расстояния в пределах кластера от
    ↪  кол-ва кластеров
131             '''
132             plt.figure(figsize=(13, 7))
133             plt.title("Метод локтя")
134             plt.plot(listClusterNumbers, listAvgDistanceHA, '-.', label =
            ↪  'Иерархический метод кластеризации')
135             plt.plot(listClusterNumbers, listAvgDistanceKP, '--', label = 'Метод
            ↪  кластеризации K-прототипов')
136             plt.plot(listClusterNumbers, listAvgDistanceHybrid, '-', label =
            ↪  'Гибридный метод кластеризации')
137             plt.grid(True)
138             plt.legend()
139             plt.ylabel('Среднее расстояние в пределах кластера')
140             plt.xlabel('Кол-во кластеров, шт.')
141             plt.show()
```

ПРИЛОЖЕНИЕ В

Реализация оценки качества кластеризации методом оценки силуэтов

Листинг В.1 — Реализация класса TestSilhouettes

```
class TestSilhouettes():
    def __init__(self, objects: List[List[str]], numberOfRuns: int):
        self.objects = objects
        self.numberOfRuns = numberOfRuns
        self.numberObjects = len(objects)
        self.dissimilarityMatrix = self.createDissimilarityMatrix()

    def createDissimilarityMatrix(self) -> List[List[float]]:
        '''
        Возвращает матрицу несходства
        '''
        distance = Distance()
        dissimilarityMatrix = distance.createDissimilarityMatrix(self.objects)

        return dissimilarityMatrix

    def calcAvgDistance(self, objectNumber: int,
            listObjectNumbers: List[int], defaultValue: int=0) -> float:
        '''
        Возвращает среднее расстояние между объектом и переданными элементами
        '''
        sumDistances = 0
        countObjects = len(listObjectNumbers)

        for i in range(countObjects):
            sumDistances +=
            ↪  self.dissimilarityMatrix[objectNumber][listObjectNumbers[i]]

        return sumDistances / countObjects if countObjects else defaultValue

    def calcMaxDistance(self, objectNumber: int, listObjectNumbers: List[int]) ->
    ↪  float:
        '''
        Возвращает максимальное расстояние между объектом и переданными элементами
        '''
        distance = 0
        countObjects = len(listObjectNumbers)

        for i in range(countObjects):
```

Продолжение листинга B.1

```
38              distance = max(distance,
                ↪  self.dissimilarityMatrix[objectNumber][listObjectNumbers[i]])
39
40          return distance if countObjects else 1
41
42      def getNearestCluster(self, objectNumber: int,
43              clusterContents: List[List[int]]) -> List[int]:
44          '''
45          Возвращает список номеров объектов из ближайшего кластера
46          '''
47          minDistance = 1
48          nearestCluster = clusterContents[0]
49
50          for listObjectNumbers in clusterContents:
51              # расстояние между двумя кластерами -- самое длинное расстояние между
                ↪  двумя точками в каждом кластере (полная связь)
52              distance = self.calcMaxDistance(objectNumber, listObjectNumbers)
53              if distance < minDistance:
54                  minDistance = distance
55                  nearestCluster = listObjectNumbers
56
57          return nearestCluster
58
59      def calcSumCoeffsOneCluster(self,
60              listObjectNumbers: List[int],
61              clusterContents: List[List[int]]) -> float:
62          '''
63          Возвращает сумму значений коэффициента силуэта
64          для элементов текущего кластера
65          '''
66          sumCoeffs = 0
67
68          for objectNumber in listObjectNumbers:
69              # найдем список номеров объектов из ближайшего кластера
70              nearestCluster = self.getNearestCluster(objectNumber, clusterContents)
71              # создадим копию списка номеров элементов текущего кластера без с
                ↪  номера текущего объета
72              otherNumbers = listObjectNumbers.copy()
73              otherNumbers.remove(objectNumber)
74              # среднее расстояние между текущим объектом и объектами из ближайшего
                ↪  кластера
75              b = self.calcAvgDistance(objectNumber, nearestCluster, defaultValue=1)
76              a = self.calcAvgDistance(objectNumber, otherNumbers, defaultValue=0)
77              sumCoeffs += (b - a) / max(a, b)
78
```

Продолжение листинга B.1

```
79          return sumCoeffs
80
81      def calcAvgSilhouetteCoeff(self, clusterContents: List[List[int]]) -> float:
82          '''
83          Возвращает среднее значение коэффициента силуэта
84          для всех объектов при заданном кол-ве кластеров
85          '''
86          sumCoeffs = 0
87          countCoeffs = 0
88
89          for i in range(len(clusterContents)):
90              listObjectNumbers = clusterContents[i]
91              # если кластер содержит элементы
92              if len(listObjectNumbers) > 0:
93                  # создадим копию списка без массива с номерами элементов текущего
                    ↪ кластера
94                  otherClusterContents = clusterContents.copy()
95                  otherClusterContents.pop(i)
96
97                  sumCoeffs += self.calcSumCoeffsOneCluster(listObjectNumbers,
                    ↪ otherClusterContents)
98                  countCoeffs += len(listObjectNumbers)
99
100         return sumCoeffs / countCoeffs if countCoeffs != 0 else 0
101
102     def calcAvgSilhouetteCoeffForMethod(self, method: EventMethod, numberClusters:
        ↪ int) -> float:
103         '''
104         Возвращает среднее значение коэффициента силуэта
105         для всех объектов при заданном кол-ве кластеров
106         '''
107         if method == method.KP_CLUST:
108             sumAvgCoeffs = 0
109             for _ in range(self.numberOfRuns):
110                 kPrototypesClusterization = KPrototypesClusterization(self.objects,
                    ↪ numberClusters)
111                 sumAvgCoeffs +=
                    ↪ self.calcAvgSilhouetteCoeff(kPrototypesClusterization
                    ↪ .clusterContents)
112
113             avgSilhouetteCoeff = sumAvgCoeffs / self.numberOfRuns
114
115         elif method == method.HA_CLUST:
116             haClusterization = HAClusterization(self.dissimilarityMatrix.copy(),
                ↪ numberClusters)
```

Продолжение листинга В.1

```
117             # получение списка номеров объектов, входящих в каждый кластер
118             clusterContents = [node.listClusterNumbers for node in
                ↪  haClusterization.nodes]
119             avgSilhouetteCoeff = self.calcAvgSilhouetteCoeff(clusterContents)
120         else:
121             hybridClusterization = HybridClusterization(self.objects,
                ↪  numberClusters)
122             avgSilhouetteCoeff =
                ↪  self.calcAvgSilhouetteCoeff(hybridClusterization.clusterContents)
123
124         return avgSilhouetteCoeff
125
126     def comparisonMethods(self) -> None:
127         '''
128         Сравнение методов разбиения
129         '''
130         listSilhouetteCoeffHA = []
131         listSilhouetteCoeffKP = []
132         listSilhouetteCoeffHybrid = []
133
134         print()
135         for k in range(2, self.numberObjects + 1):
136             listSilhouetteCoeffHA.append(self
                ↪  .calcAvgSilhouetteCoeffForMethod(EventMethod.HA_CLUST,
                ↪  k))
137             listSilhouetteCoeffKP.append(self
                ↪  .calcAvgSilhouetteCoeffForMethod(EventMethod.KP_CLUST,
                ↪  k))
138             listSilhouetteCoeffHybrid.append(self
                ↪  .calcAvgSilhouetteCoeffForMethod(EventMethod.HYBRID_CLUST,
                ↪  k))
139
140             print("Проведено сравнение {:d} кластеров из {:d}".format(k,
                ↪  self.numberObjects))
141
142         listClusterNumbers = [i for i in range(2, self.numberObjects + 1)]
143
144         self.printAvgDistanceTable(
145             listClusterNumbers, listSilhouetteCoeffHA,
146             listSilhouetteCoeffKP, listSilhouetteCoeffHybrid)
147         self.buildGraph(
148             listClusterNumbers, listSilhouetteCoeffHA,
149             listSilhouetteCoeffKP, listSilhouetteCoeffHybrid)
150
151     @staticmethod
```

Продолжение листинга B.1

```
152     def printAvgDistanceTable(
153             listClusterNumbers: List[int],
154             listSilhouetteCoeffHA: List[float],
155             listSilhouetteCoeffKP: List[float],
156             listSilhouetteCoeffHybrid: List[float]) -> None:
157         '''
158         Построение таблицы со средними значениями коэффициента силуэта
159         '''
160         tableHeader = ["Кол-во кластеров", "Иерархический", "К-прототипов",
            ↪ "Гибридный"]
161         table = PrettyTable(tableHeader)
162 
163         for i in range(len(listClusterNumbers)):
164             table.add_row([
165                 listClusterNumbers[i],
166                 round(listSilhouetteCoeffHA[i], 3),
167                 round(listSilhouetteCoeffKP[i], 3),
168                 round(listSilhouetteCoeffHybrid[i], 3)
169             ])
170         print("\n  --- Таблица со средними значениями коэффициента силуэта ---")
171         print(table)
172 
173     @staticmethod
174     def buildGraph(
175             listClusterNumbers: List[int],
176             listAvgDistanceHA: List[float],
177             listAvgDistanceKP: List[float],
178             listAvgDistanceHybrid: List[float]) -> None:
179         '''
180         Построения графика зависимости среднего значения коэффициента силуэта от
    ↪   кол-ва кластеров
181         '''
182         plt.figure(figsize=(13, 7))
183         plt.title("Метод оценки силуэтов")
184         plt.plot(listClusterNumbers, listAvgDistanceHA, '-.', label =
            ↪ 'Иерархический метод кластеризации')
185         plt.plot(listClusterNumbers, listAvgDistanceKP, '--', label = 'Метод
            ↪ кластеризации К-прототипов')
186         plt.plot(listClusterNumbers, listAvgDistanceHybrid, '-', label =
            ↪ 'Гибридный метод кластеризации')
187         plt.grid(True)
188         plt.legend()
189         plt.ylabel('Среднее значение коэффициента силуэта')
190         plt.xlabel('Кол-во кластеров, шт.')
191         plt.show()
```

ПРИЛОЖЕНИЕ Г