

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

ФАКУЛЬТЕТ КОМПЬЮТЕРНЫХ НАУК

Еремейкин Петр Александрович

**РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ,
ОРИЕНТИРОВАННОГО НА ПОЛЬЗОВАТЕЛЯ, ДЛЯ ПРОВЕДЕНИЯ
КЛАСТЕР-АНАЛИЗА**

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению подготовки 01.04.02 Прикладная математика и информатика
образовательная программа «Науки о данных»

Научный руководитель

д.т.н., проф.

Б.Г. Миркин

И.О. Фамилия

Москва 2018

Аннотация

В данной работе рассматривается вопрос программной реализации современных интеллектуальных алгоритмов кластер-анализа. Внимание уделено как теоретическому описанию эффективных алгоритмов, так и разработке программного кода модулей системы с дружелюбным графическим пользовательским интерфейсом.

Во введении обоснована необходимость создания разрабатываемого комплекса, описана текущая ситуация в области кластеризации и основные проблемы, возникающие при применении популярных алгоритмов кластеризации.

В теоретической части описаны пять реализованных алгоритмов: *ik-means*, A-Ward, A-Ward _{$p\beta$} , dePDDP, BiKM-R, указаны характерные свойства и математические приёмы, лежащие в основе каждого из них. Для прояснения наиболее сложных для понимания особенностей, некоторые алгоритмы разобраны на иллюстративных примерах. В конце теоретической части приводятся общие рекомендации по выбору наиболее подходящего алгоритма для различных случаев.

В главе, посвящённой описанию программы вынесены вопросы взаимодействия с программой с точки зрения пользователя. Рассмотрены базовые операции и приведены пояснения к некоторым окнам графического интерфейса, с которыми сталкивается пользователь при выполнении этих операций.

Особенности программной реализации рассматриваются в главе 4. Эта глава содержит обоснование выбора средств реализации, и используемых программных библиотек. Некоторые вопросы освещены с помощью фрагментов кода. Для алгоритмов приведены схемы вычислений с необходимыми пояснениями.

Для демонстрации принципов работы программы, особенностей предварительной подготовки данных и интерпретации результатов рассмотрен иллюстративный пример. На нем показаны полученные кластеры и изложена интерпретация каждого из кластеров в соответствии с характерными признаками.

В заключении подведены итоги работы и обозначены возможные направления исследований и усовершенствований проекта.

Содержание

1	Введение	4
2	Теоретическая часть	7
2.1	Постановка задачи кластеризации	8
2.2	Аномальная кластеризация	9
2.3	Алгоритм A-Ward	13
2.3.1	Специальный критерий остановки	16
2.4	Алгоритм A-Ward _{$p\beta$}	18
2.5	Алгоритм dePDDP	22
2.6	Алгоритм BiKM-R	25
2.7	Нормализация данных	28
2.7.1	Общая формула нормализации данных	29
2.7.2	Преобразование начала отчёта	30
2.7.3	Преобразование масштаба шкалы	30
2.8	Генератор синтетических данных	30
2.9	Интерпретация результатов	32
2.9.1	Оценка качества разбиений	32
2.9.2	Характеристика Silhouette Width	32
2.9.3	Adjusted Rand Index	33
2.9.4	Характеризация кластеров	33
2.10	Общие рекомендации по выбору алгоритма	35
3	Пользовательское описание программы INDACT	37
3.1	Этапы работы с программой	37
3.2	Основные сведения о пользовательском интерфейсе	38
3.3	Загрузка данных	40
3.4	Генерация синтетических данных	41
3.5	Нормализация данных	43
3.6	Кластеризация	44
3.7	Анализ результатов	46

4	Структурное описание программы INDACT	49
4.1	Выбор методологии разработки	49
4.2	Выбор программных средств	50
4.2.1	Язык программирования	50
4.2.2	Библиотека графического интерфейса	53
4.2.3	Библиотеки для работы с данными	55
4.2.4	Тестирование	55
4.2.5	Средства дистрибуции	57
4.3	Структура системы	58
4.4	Структура входных данных	62
4.5	Реализация алгоритмов	63
4.5.1	Инициализация аномальными кластерами	63
4.5.2	A-Ward	69
4.5.3	A-Ward _{$p\beta$}	72
4.5.4	dePDDP	72
4.5.5	BiKM-R	72
5	Демонстрационный пример	74
5.1	Формирование данных	74
5.2	Описание данных	75
5.3	Нормализация	76
5.4	Кластеризация	76
5.5	Интерпретация результатов	77
6	Заключение	82

1 Введение

В настоящее время наблюдается интенсивное развитие информационных технологий, появляются новые программные решения, применяемые в широком спектре областей. Если раньше информационные технологии были областью интересов узкого круга специалистов, то сейчас установилась тенденция к повсеместному распространению прикладных программных продуктов.

Современные компании вынуждены опираться на применение информационной инфраструктуры и использовать преимущества цифровых технологий для поддержания конкурентоспособности своих продуктов или услуг. В процессе эксплуатации информационных систем накапливаются массивы данных, обработка и интерпретация которых может принести компании коммерческую выгоду.

Каждый случай обработки данных, как правило, требует индивидуального подхода, не существует универсальной последовательности операций для любой задачи. Поэтому, обработка данных сопряжена с привлечением интеллектуальных усилий от высококвалифицированных специалистов. Программные системы анализа данных призваны облегчить этот труд и предоставляют в распоряжение специалиста наиболее востребованные процедуры обработки данных. Особенно актуально применение таких систем для решения прикладных задач, которые, несмотря на свою индивидуальность, зачастую однотипны и всё же имеют некоторые общие этапы решения. Применяя готовый, тщательно отлаженный и документированный программный код, сокращается время на анализ данных, а также снижается вероятность возникновения ошибок.

Прогресс технических средств в области сбора и обработки информации приводит к росту размеров массивов данных, которые требуется обрабатывать для удовлетворения потребностей компаний. Поэтому растёт роль методов агрегации данных и выделения в них общих закономерностей или структур. К таким методам, в частности относятся методы кластеризации.

Под кластеризацией понимают выделение объектов из таблицы наблюдений в множества, называемые кластерами, которые объединяют наиболее сходные объекты, при этом различные объекты должны попадать в разные кластеры [1]. По принадлежности заданного нового объекта к определенному кластеру можно сделать предположения о его ключевых свойствах. Задачи кластеризации часто встают в самых разных областях,

например, при обработке изображений или биологических структур а также социальных групп [2].

Вероятно, наиболее широко известный и популярный метод кластеризации — k -means [3]. Этот метод основан на поочерёдной минимизации квадратичного критерия по двум группам переменных: центрам кластеров и принадлежности объектов кластерам. На основе аналогичного квадратичного критерия предложены целый ряд эффективных алгоритмов, в том числе и иерархические, например, Ward, Bisecting k -means.

Однако, несмотря на популярность k -means, он обладает существенными недостатками. Первый заключается в том, что перед запуском алгоритма требуется знать число кластеров, которое должно быть выявлено. На практике, далеко не всегда число кластеров известно заранее. Например, в случае анализа интернет данных, сформированных из журналов посещений, и формирования групп схожих сайтов не предполагается наличие априорной информации. В таком случае придётся или рассматривать другие алгоритмы, или опираться на эмпирические зависимости.

Второй недостаток заключается в том, что результат работы k -means сильно зависит от инициализации. В некоторых случаях неправильная инициализация может приводить к неудовлетворительным результатам. Этот недостаток обычно устраняют путём многократного запуска алгоритма для различных начальных условий, но при этом увеличивается и продолжительность кластеризации. Например, в случае обработки данных в режиме онлайн, такой способ может оказаться неприемлемым. В то же время были предложены эффективные методы инициализации, которые позволяют оценить число кластеров с небольшими дополнительными временными затратами.

Третий существенный недостаток k -means состоит в снижении качества получаемого разбиения для случая зашумлённых данных. Многие данные, например составленные на основе измерений физических величин, содержат случайные погрешности, которые требуется учитывать при кластеризации для получения удовлетворительного результата. Отсутствие какого-либо механизма учёта шума в данных стимулирует исследователей предлагать усовершенствования k -means.

Описанные выше недостатки k -means породили множество модификаций этого алгоритма, а также послужили импульсом для проведения работ над новыми алгоритмами, основанными на квадратичном критерии. Например, иерархический алгоритм

Ward [4] использует квадратичный критерий для агломеративного построения кластерной структуры. В свою очередь, Ward также имеет свои недостатки, которые частично были унаследованы от k -means, поэтому были предложены алгоритмы Ward_p и A-Ward_{pβ} [5], развивающие идею взвешенной кластеризации. Они продемонстрировали высокую эффективность при обработке зашумлённых данных, но, тем не менее, для канонической формулировки этих алгоритмов проблема выбора числа кластеров осталась неразрешённой. В дивизивных алгоритмах BiKM-R и dePDDP [6] используется автоматический критерий остановки, благодаря чему число кластеров определяется во время кластеризации.

Таким образом, за последнее время появилось большое число новых и эффективных алгоритмов кластеризации, многие из них ещё не реализованы в популярных библиотеках, таких как `scipy` для языка Python или `Clustering Toolbox` для MATLAB. Авторы новых алгоритмов заявляют о их высокой эффективности и становится очевидно, что со временем эти алгоритмы найдут своё применение для задач определённой специфики. В данной работе рассматривается разработка программного обеспечения, в состав которого входит набор современных интеллектуальных алгоритмов кластеризации, основанных на критерии наименьших квадратов. Разработанная программа получила название “Система интеллектуальной кластеризации данных” (Intelligent Data Clustering Toolkit, INDACT).

Программная система обладает графическим пользовательским интерфейсом, что делает её простой в применении даже для специалистов, не обладающих широкими познаниями в области программирования. В то же время пользовательский интерфейс представляет собой лишь надстройку над разработанной базовой библиотекой, которая включает в себя упомянутые алгоритмы. Эта библиотека имеет открытый код, снабжённый необходимой документацией, и тем самым допускает использование в других программных продуктах.

2 Теоретическая часть

В теоретической части будут подробно описаны основные алгоритмы, реализованные в системе INDACT. Алгоритмы кластеризации по принципу работы разделяют на две категории: плоские и иерархические. К плоским относится, например, популярный k -means. Данная работа затрагивает в основном вторую категорию. Иерархические алгоритмы, в отличие от плоских, формируют вложенную структуру кластеров. Информация о взаимной вложенности кластеров может быть полезна для некоторых практических приложений, например, при исследовании биологических видов, вложенность кластеров может отражать филогенетическое родство. Это интересное свойство повышает интерес исследователей к иерархическим алгоритмам.

Различают два вида иерархических алгоритмов: агломеративные (объединяющие) и дивизивные (разделяющие), которые соответственно реализуют восходящее и нисходящее направление формирования результирующего разбиения. Агломеративные алгоритмы рассматривают исходные данные как множество кластеров, состоящих из единственного объекта, который одновременно является центром. Итеративно происходит объединение двух ближайших кластеров, пока не будет выполнен заданный критерий останова. Дивизивные алгоритмы в противоположность агломеративным начинают работу с одного кластера, в который включены все объекты данных и разделяют его на более мелкие кластеры.

Программная система INDACT предоставляет пользователю выбор из четырёх современных алгоритмов кластеризации, два из которых агломеративные, а два — дивизивные. Агломеративные алгоритмы A-Ward и A-Ward _{$p\beta$} основаны на классическом критерии Ward, но расширяют традиционный подход с помощью предварительного этапа, называемого аномальной кластеризацией. Задача аномальной кластеризации состоит в предварительной разведке кластерной структуры и вычленении аномальных кластеров, которые расположены далеко от центра данных. Алгоритм A-Ward _{$p\beta$} , кроме того, предполагает взвешивание признаков, что может рассматриваться как механизм учёта случайных погрешностей. Два дивизивных алгоритма — dePDDP и BiKM-R используют автоматические критерии останова, выполнение которых проверяется на каждой итерации. Они хорошо подходят для тех случаев, когда число кластеров заранее неизвестно.

Кроме кластеризации в состав программной системы включён модуль для генерации данных. Этот модуль позволяет сгенерировать при помощи небольшого числа управляющих параметров синтетические данные и на их примере проверить работу различных алгоритмов.

2.1 Постановка задачи кластеризации

В работе используется представление данных в виде таблицы объект–признак. Такой формат данных широко распространён для практических приложений и встречается во многих реальных ситуациях. Программа INDACT не позволяет обрабатывать другие варианты представления данных.

Пусть имеется N объектов и у каждого объекта определены значения V признаков. Вообще говоря, признаки могут принимать как числовые значения, так и номинальные. Далее подразумевается, что если исходные данные имеют номинальные признаки, то их следует представить с помощью выбранного метода в виде числовых значений. Например, номинальный признак может быть разложен на несколько бинарных, каждый бинарный признак обозначает наличие (значение 1) или отсутствие (значение 0) соответствующего значения исходного признака. Множество всех объектов будем обозначать Y . Тогда эти данные могут быть представлены в виде таблицы следующего вида:

$$Y = \begin{pmatrix} y_1 \\ \dots \\ y_N \end{pmatrix} = \begin{pmatrix} y_{11} & \dots & y_{1V} \\ \dots & \dots & \dots \\ y_{N1} & \dots & y_{NV} \end{pmatrix}$$

Требуется получить разбиение $S = \{C_1, \dots, C_K\}$, состоящее из K кластеров, которые не пересекаются и покрывают всё множество объектов Y . Кластер будем обозначать прописной буквой C , а его центр — строчной c . В общем случае значение K не известно, хотя встречаются ситуации в которых число кластеров задано. Например, число кластеров может быть известно исходя из общих закономерностей предметной области. Чёткой формулировки относительно того, что должно быть включено в кластеры не существует. Общая идея состоит в том чтобы сходные объекты были включены в один кластер, а несходные не принадлежали одному кластеру. В схожесть объектов в различных приложениях может вкладываться различный математический смысл, например,

схожесть объектов может определяться геометрической близостью или совпадением некоторых основных переменных.

2.2 Аномальная кластеризация

Аномальная кластеризация [7] имеет большое значение для всего программного комплекса, так как этот подход широко используется как составляющая часть более сложных алгоритмов, например, аномальная инициализация является первым шагом A-Ward и A-Ward_{pβ}. Этот этап позволяет предварительно “разведать” структуру данных и тем самым сформировать исходные предположения о возможном числе кластеров.

Алгоритмы, основанные на k -means требуют явного задания числа кластеров и начальных центров. Как правило, число кластеров определяется исходя из общих зависимостей и представлений пользователя о предметной области. Начальные центры кластеров определяются по результатам нескольких запусков для случайного положения с последующим выбором наилучшего результата. Зачастую у пользователя не имеется никаких представлений о предметной области относительно возможного числа кластеров. Поэтому в настоящее время идёт интенсивная работа над разработкой различных методов, которые позволяли бы определить число кластеров исходя исключительно из самих данных.

Аномальный кластер-анализ является одним из таких методов. Он основан на поочерёдном поиске аномальных групп и исключении их из данных, до тех пор, пока не останется ни одного объекта. Под аномальной группой понимается множество объектов, которые далеко отстоят от глобального центра данных. Количество найденных аномальных групп объектов может служить приближением числа кластеров. Существуют алгоритмы, которые непосредственно используют найденное число кластеров в результате аномальной инициализации, например, ik -means [7], а также алгоритмы, которые только опираются на этот результат для первой итерации и могут производить отличное число кластеров, например, A-Ward_{pβ}.

Метод аномальных кластеров можно рассматривать как модифицированный частный случай k -means при числе кластеров $K = 2$. При этом инициализация начального положения центров кластеров жёстко определена — один кластер всегда имеет центр в глобальном центре всех данных и не изменяется во время работы алгоритма, а центр

второго кластера, который называется аномальным, инициализируется в наиболее удалённой от глобального центра точке. В процессе работы центр аномального кластера уточняется аналогично традиционному k -means. Отличие метода аномальных кластеров от k -means заключается в том, что один центр остаётся неизменным при на всех итерациях. На каждом шаге в аномальный кластер включаются те точки, которые лежат ближе к центру аномального кластера, чем к глобальному центру данных. После центр аномального кластера обновляется чтобы соответствовать среднему по всем включённым объектам. Когда центр аномального кластера стабилизировался и не изменяется, происходит исключение найденного кластера и продолжается работа с оставшимися данными. Исключение аномальных групп объектов происходит до тех пор, пока не останется ни одного объекта.

Формально алгоритм аномального кластер-анализа можно записать в следующем виде:

А Л Г О Р И Т М # 1: Аномальная кластеризация (ik-means)

1. Подготовка. Задаться пороговым значением минимальной численности аномального кластера Θ . Вычислить глобальный центр данных $a = (a_1, \dots, a_V)$:

$$a_v = \frac{1}{N} \sum_{i=1}^N y_{iv}$$

Если имеются исходные представления о норме, использовать их для вычисления глобального центра. Стандартизировать таблицу данных сдвигом начала координат в точку a .

2. Инициализация аномального центра . Определить наиболее удалённую от начала координат точку s . Эта точка является начальным центром аномального кластера.

3. Обновление аномального кластера . Объекты, которые расположены ближе к центру s , чем к началу координат включить в аномальный кластер. Если в аномальный кластер не было внесено изменений, перейти к шагу 5.

4. Обновление центра . Вычислить новый центр аномального кластера как покомпонентное среднее всех объектов, включённых в него. Перейти к шагу 3.

5. *Сохранение центра*. Если число объектов в аномальном кластере больше заданного порогового значения Θ , сохранить центр и ассоциированный с ним аномальный кластер в список результатов.

6. *Исключение аномального кластера*. Исключить из данных все объекты, которые принадлежат аномальному кластеру. Если в таблице Y все ещё остаются объекты, перейти к шагу 2.

7. *Кластеризация*. Выполнить алгоритм k -means [3] на исходных данных Y . При этом использовать центры аномальных кластеров, сохранённых на шаге 5 в качестве начальных. Результатом работы ik -means является полученное разбиение.

В [1] показано, что аномальная кластеризация минимизирует критерий, подобный критерию метода k -means:

$$W(S, c) = \sum_{i \in C} d(y_i, c) + \sum_{i \notin C} d(y_i, 0),$$

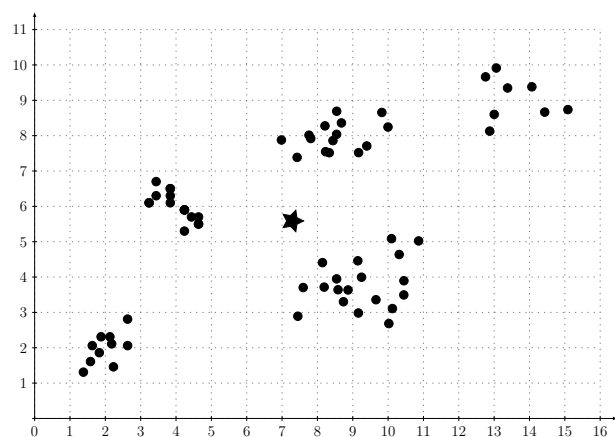
где: C — искомый кластер;

c — центр кластера ;

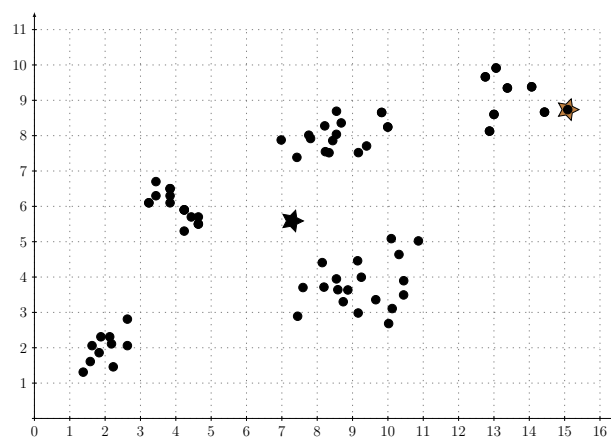
d — квадрат евклидовского расстояния.

Стоит заметить, что хотя аномальная кластеризация и позволяет предварительно “разведать” структуру данных, но на применение этого метода также имеются ограничения. Например, следует учитывать, что получаемая структура аномальных кластеров сгущается ближе к началу координат. Для сглаживания этого недостатка рекомендуется отбрасывать слишком маленькие аномальные группы, число объектов в которых меньше заданного значения Θ .

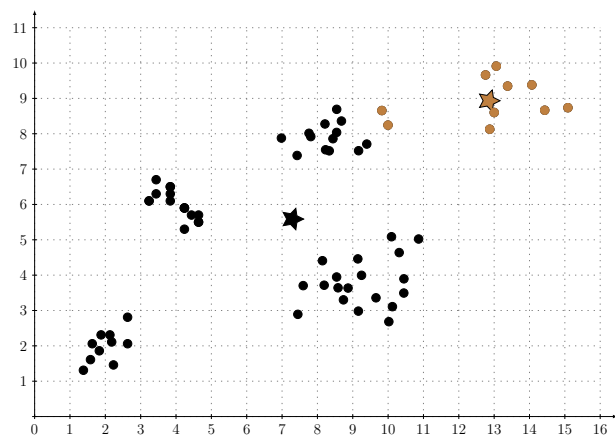
Рассмотрим работу алгоритма ik -means на простом примере. Пусть имеются двумерные данные, изображённые на рисунке 1(а). Первым шагом алгоритм определяет наиболее удалённую точку от глобального центра данных (на рисунке 1 обозначается чёрной звёздочкой) как показано на рисунке 1(б). Когда начальный центр аномального кластера определён, выполняются шаги некоторой вариации k -means при $K = 2$ и жёстко зафиксированным центром данных. При этом центр аномального кластера



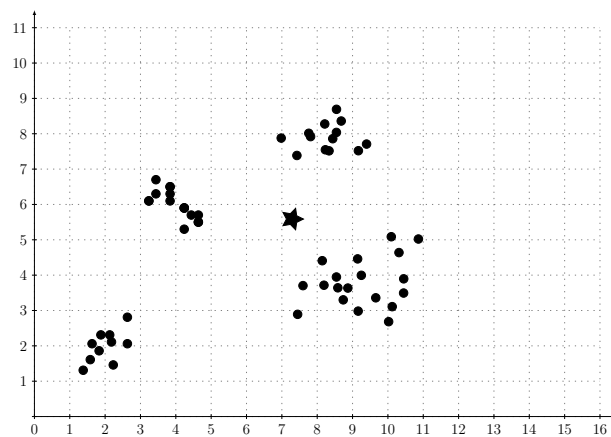
(а) Исходные данные



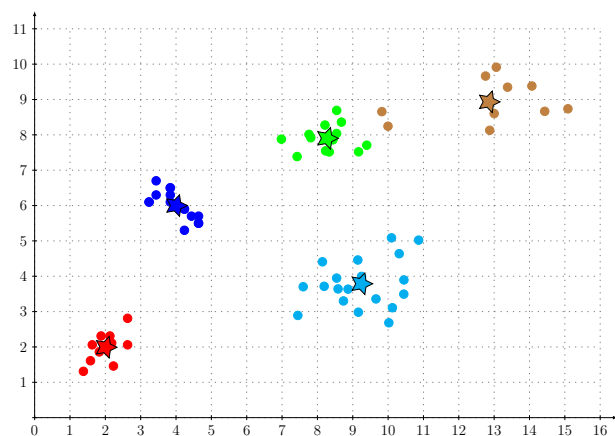
(б) Выбор начального центра аном. кластера



(в) Аномальный кластер сформирован



(г) Аномальный кластер исключён



(д) Итоговое разбиение аном. кластерами

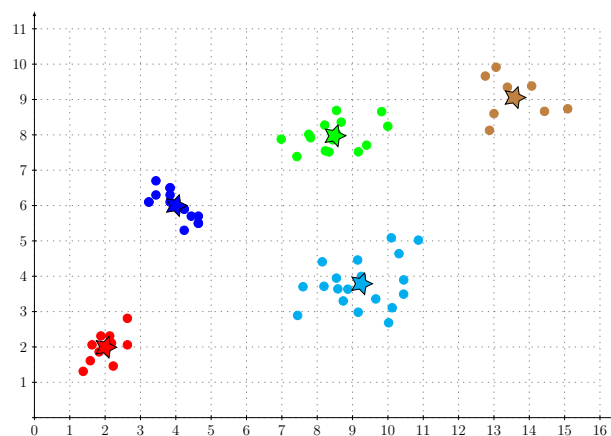
(е) Результат после выполнения k -means

Рисунок 1 – ik -means: этапы выделения аномальных кластеров
 Каждому кластеру соответствуют точки определённого цвета в двумерном пространстве.

уточняется до схождения (рисунок 1(в)). Объекты, которые находятся ближе к центру аномального кластера, чем к глобальному центру исходных данных, включаются в состав аномального кластера и на после схождения очередной итерации, исключаются из данных без изменения глобального центра (см. рисунок 1(г)). Когда все объекты будут исчерпаны, для полученных аномальных кластеров выполняется классическая версия k -means с числом кластеров, равным числу найденных аномальных групп и соответствующими центрами. Результат работы ik -means изображён на рисунке 1(е).

Метод аномальной кластеризации позволяет достичь хороших результатов, но как было показано в недавних исследованиях [5], в случае задания стандартного порогового параметра $\Theta = 1$, выделяет избыточное число кластеров. Определение рационального значения Θ в настоящее время не имеет методической поддержки, поэтому для использования большого потенциала алгоритма было предложено его новое применение в роли предварительного этапа инициализации иерархических алгоритмов.

2.3 Алгоритм A-Ward

Алгоритм A-Ward является модернизацией широко известного алгоритма иерархической кластеризации Ward. Алгоритм Ward [4] относится к агломеративным, то есть он действует по принципу “снизу вверх”. На первом этапе принимается, что каждый отдельный объект $y_i \in Y$ представляет собой кластер, центром которого является сам объект y_i . Алгоритм Ward итеративно выбирает два ближайших кластера и объединяет их, таким образом уменьшая общее число кластеров на единицу на каждой итерации. Процесс повторяется, пока не будет достигнуто заранее заданное число кластеров. Опишем последовательность шагов Ward:

А Л Г О Р И Т М # 2: Ward

1. *Инициализация.* Задаться желаемым числом кластеров K^* . Установить начальное число кластеров равным числу объектов $K = N$. Исходное разбиение образованно элементарными кластерами $S = \{C_1, \dots, C_K\}$, в каждый из которых включён единственный объект $y_i \in Y$ из исходных данных.
2. *Слияние кластеров.* Выбрать два ближайших кластера $C_a, C_b \in S$ согласно

следующей формуле:

$$(1) \quad d_{Ward}(C_a, C_b) = \frac{N_1 N_2}{N_1 + N_2} \sum_{v=1}^V (c_{av} - c_{bv})^2,$$

где: N_1, N_2 — число объектов в кластерах C_a, C_b ;

c_{av}, c_{bv} — v -ая координата центра кластеров C_a, C_b .

Сформировать новый кластер C_{ab} , в который входят все объекты из C_a и C_b , удалив при этом старые кластеры C_a, C_b . Уменьшить текущее число кластеров K на 1.

3. Обновление центра. Вычислить новый центр сформированного кластера C_{ab} как покомпонентное среднее среди всех объектов этого кластера.

4. Условие остановки. Если текущее число кластеров больше желаемого $K > K^*$ и $K > 1$, перейти к шагу 2. В противном случае выдать текущее разбиение в качестве результата.

Следует обратить внимание на два важных свойства алгоритма Ward. Во-первых, как и любой иерархический алгоритм, Ward формирует вложенную структуру классов, что может быть использовано в некоторых практических приложениях, где вложенность кластеров может быть естественным образом объяснена в терминах предметной области. Во-вторых, в канонической формулировке алгоритм не требует инициализации, как например, k -means. Таким образом, отпадает необходимость многократного запуска для определения наилучшего разбиения при различных начальных условиях.

Но у Ward есть существенный недостаток — это продолжительность вычислений. В соответствии с описанной последовательностью шагов на каждой итерации алгоритма происходит вычисление расстояния между всеми кластерами. На первых итерациях число кластеров примерно равно числу объектов, это значит что требуется время, квадратично зависящее от N . Этот недостаток послужил толчком для исследования возможности применения аномальной кластеризации для сокращения вычислений.

Из экспериментальных исследований известно, что аномальная кластеризация порождает избыточное число кластеров в то время как Ward на каждом шаге сокращает число кластеров на единицу. Сочетание аномальной кластеризации и алгоритма Ward

позволит сократить время вычисления благодаря исключению стадии с большим числом маленьких кластеров. Такая модификация получила название A-Ward [5].

А Л Г О Р И Т М # 3: A-Ward

1. *Инициализация*. Установить $\Theta = 1$. Получить начальное число кластеров K и само разбиение S по алгоритму *ik-means* (алгоритм 1).
2. *Слияние кластеров*. Выбрать два ближайших кластера $C_a, C_b \in S$ согласно формуле (1). Сформировать новый кластер C_{ab} , в который входят все объекты из C_a и C_b , удалив при этом старые кластеры C_a, C_b . Уменьшить текущее число кластеров K на 1.
3. *Обновление центра*. Вычислить новый центр сформированного кластера C_{ab} как покомпонентное среднее среди всех объектов этого кластера.
4. *Условие остановки*. Если текущее число кластеров больше желаемого $K > K^*$ и $K > 1$, перейти к шагу 2. В противном случае выдать текущее разбиение в качестве результата.

На рисунке 2 продемонстрирован принцип работы алгоритма A-Ward для простых двумерных данных. Пусть после выполнения шага 1 была получена кластерная структура, показанная на рисунке 2(а). Исключительно в целях демонстрации зададим $K^* = 1$, то есть алгоритм продолжит работать пока не будет сформирован единственный кластер из всех объектов. Предположим что, после сравнения всех расстояний между кластерами, вычисленных по формуле (1) выяснилось, что красный и синий кластеры имеют наименьшее расстояние. Тогда они будут объединены в один кластер (для определённости присвоим ему красный цвет, хотя это не имеет значения). На следующем этапе будет произведено сравнение расстояний между оставшимися четырьмя кластерами и объединение ближайших, как показано на рисунке 2(б). Объединение кластеров продолжится, пока два последних кластера не будут слиты в единый ??.

Схематично процесс слияния кластеров можно изобразить дендограммой 2(д), которая показывает на каком этапе какие кластеры были объединены. Процесс иерархического кластер-анализа удобен с точки зрения остановки процесса: её можно произвести на любом этапе по заданному критерию, получив при этом соответствующее число кластеров. Таким образом, если было бы определено желаемое число кластеров, например, равное

трём, процесс был бы остановлен через две итерации и результирующему разбиению соответствовала бы картинка 2(в).

Предложенное усовершенствование позволяет существенно сократить время работы алгоритма, благодаря исключению трудоёмких начальных стадий с большим числом кластеров. Вместо исходного элементарного разбиения используется предварительно найденные аномальные кластеры, число которые существенно меньше числа объектов.

2.3.1 Специальный критерий остановки

В той формулировке алгоритма A-Ward, которая была приведена выше не удаётся избавиться от традиционного недостатка, присущего многим алгоритмам, основанных на k -means. Этот недостаток заключается в необходимости задать желаемое число кластеров. Однако, в направлении разрешения этой проблемы был сделан вклад в работе [8], в которой предложен специальный критерий остановки алгоритма. Предлагаемый способ завершения кластеризации в общем случае применим для иерархических алгоритмов с квадратичным критерием минимизации:

$$(2) \quad W(S, c) = \sum_{k=1}^K \sum_{i \in C_k} \sum_{v=1}^V (y_{iv} - c_{kv})^2,$$

где: W — минимизируемый квадратичный критерий;

S — кластерное разбиение, $S = C_1, \dots, C_K$;

K — число кластеров, $K = |S|$;

C_k — k -ый кластер разбиения;

c_{kv} — v -ая координата центра k -го кластера;

y_{iv} — v -ая координата центра i -го объекта;

Пусть на некотором этапе работы происходит объединение некоторых двух кластеров C_k и C_l с формированием нового кластера C_{kl} . Его центр можно вычислить по формуле: $c_{kl} = (N_k \cdot c_k + N_l \cdot c_l) / (N_k + N_l)$, где N_k и N_l обозначают число объектов в кластерах C_k и C_l соответственно. При объединении кластеров происходит увеличение критерия (2) на величину Δ :

$$(3) \quad \Delta(k, l) = W(S(k, l), c(k, l)) - W(S, c),$$

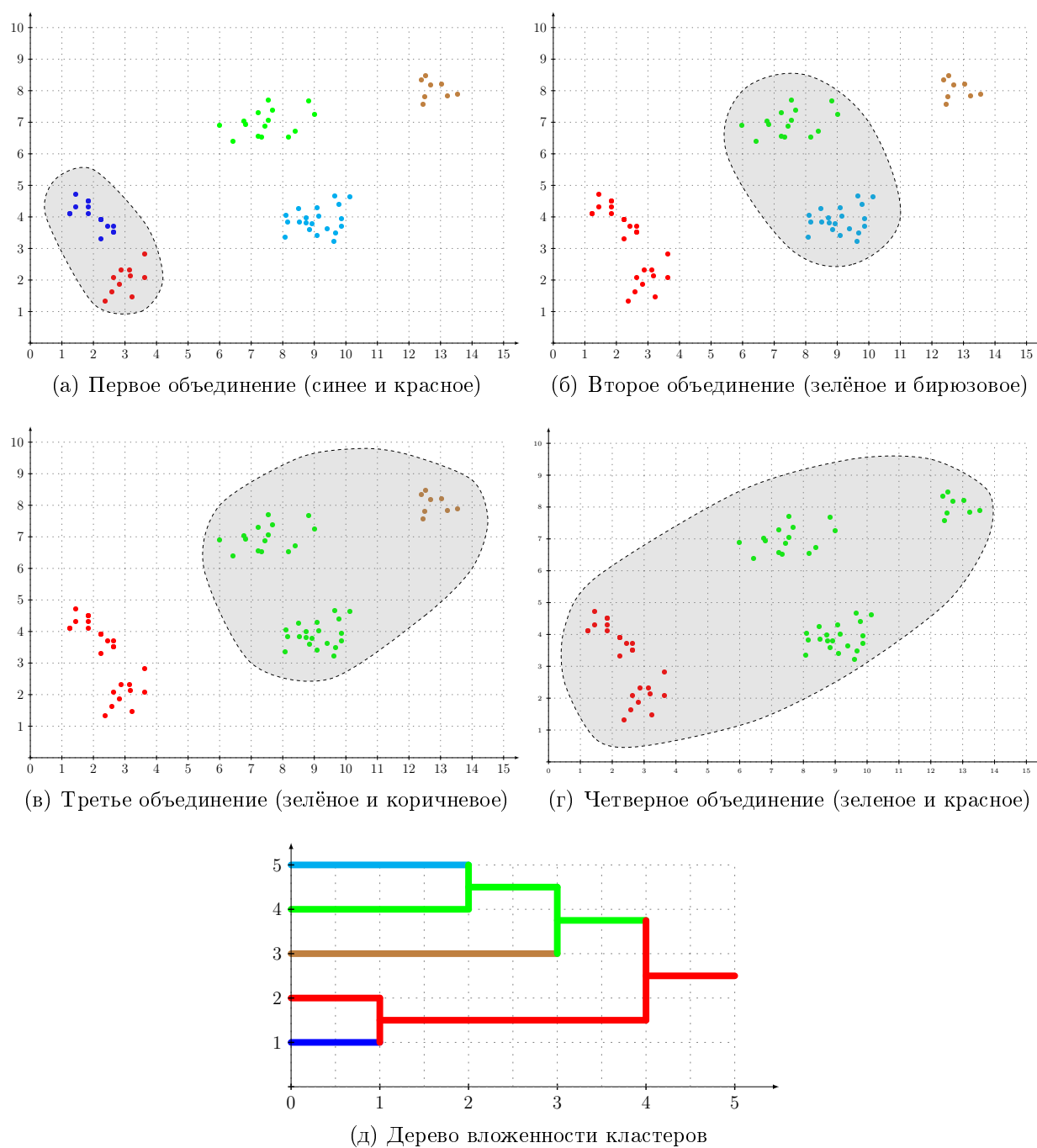


Рисунок 2 – A-Ward: этапы объединения кластеров, полученных в результате применения *ik-means*

Каждому кластеру соответствуют точки определённого цвета в двумерном пространстве.

Объединение кластеров обозначено серым затемнением с пунктирными границами. На дендограмме цвет ветви соответствует цвету точек кластера.

где: $S(k, l)$ — разбиение, полученное из S разбиения после слияния кластеров C_k и C_l ;

$c(k, l)$ — центры кластеров соответствующих разбиению $S(k, l)$;

Величина Δ всегда положительна, поскольку квадратичный критерий (2) уменьшается с ростом числа кластеров и достигает нуля при $K = N$. Величину изменения квадратичного критерия можно выразить через характеристики объединяемых кластеров:

$$\Delta(k, l) = \frac{N_k N_l}{N_k + N_l} d(c_k, c_l)$$

Как было отмечено в упомянутой статье [8], при вычислении приращения квадратичного критерия (2) по формуле (3) взаимно обнуляются все слагаемые, за исключением тех, которые относятся к кластерам $C_k, C_l, C_{kl} = C_k \cup C_l$. Таким образом, можно переписать выражение (3) следующим образом: $\Delta(k, l) = w(C_k \cup C_l) - w(C_k) - w(C_l)$, где $w(C) = \sum_{i \in C} d(y_i, c)$ — сумма квадратов евклидовских расстояний от точек кластера до его центра.

Предлагаемый специальный критерий имеет вид:

$$\Delta(k, l) < \alpha w(C_k \cup C_l)$$

где $\alpha = \frac{1}{2}$.

2.4 Алгоритм A-Ward_{pβ}

В реальных приложениях требуется анализировать данные с зашумлёнными признаками, которые были получены путём измерения некоторых физических параметров. В этом случае у алгоритмов Ward и A-Ward появляется неспособность отделения существенных признаков от шумовых. Для снижения влияния нерелевантных признаков предлагается ввести весовые коэффициенты, которые вычисляются на основании разброса значений признака: чем разброс больше, тем меньшую роль играет этот признак при кластеризации.

Модификация алгоритма A-Ward, учитывающая вес признака получила название A-Ward_{pβ}. В этой модификации также учтено обобщение для использования метрики Минковского произвольной степени. В обозначении алгоритма указано два параметра, буква p указывает на возможность изменения степени Минковского, а β — степени

весовых коэффициентов.

Как и в случае с A-Ward, для ускорения работы используется аномальная кластеризация. Алгоритм A-Ward_{pβ} потребовал разработки обобщённой версии метода аномального кластер-анализа с использованием весовых коэффициентов и для заданной метрики Минковского. Ниже описана версия алгоритма 1 для использования с алгоритмом A-Ward_{pβ}:

А Л Г О Р И Т М # 4: Аномальная инициализация для A – Ward_{pβ}

1. Инициализация. Задаться значениями параметров p и β . Вычислить глобальный центр данных c_Y как покомпонентный центр Минковского по всем объектам $y_i \in Y$.

2. Определение центра аномального кластера. Аномальный кластер принять пустым, $C_t = \emptyset$. Весовые коэффициенты распределить равномерно по всем признакам $w_{kv} = 1/V$ при $k = 1, 2$ и $v = 1, \dots, V$. За центр аномального кластера принять объект, который наиболее удалён от глобального центра c_Y по взвешенной метрике Минковского:

$$(4) \quad d_{p\beta}(y_i, c_k) = \sum_{v=1}^V w_{kv}^\beta |y_{iv} - c_{kv}|^p$$

3. Обновление аномального кластера. Каждый объект, который находится ближе к центру аномального кластера c_t , чем к глобальному центру данных c_Y , приписать к кластеру C_t . Если при этом C_t остался неизменным, перейти к шагу 6.

4. Обновление центра. Вычислить новый центр аномального кластера как покомпонентный центр Минковского по всем объектам $y_i \in C_t$.

5. Обновление весовых коэффициентов. Вычислить новые весовые коэффициенты по формуле:

$$(5) \quad w_{kv} = \frac{1}{\sum_{u=1}^V \left(\frac{D_{kv}}{D_{ku}} \right)^{\frac{1}{\beta-1}}},$$

где: $D_{kv} = \sum_{i \in C_k} |y_{iv} - c_{kv}|^\beta$ – разброс признака v в кластере C_k .

6. Сохранение параметров. Включить текущий центр аномального кластера c_t в список центров `c_list`, а веса w в список весов `w_list`.

7. Исключение аномального кластера. Исключить из Y все объекты $y_i \in C_t$. Если $Y \neq \emptyset$, перейти к шагу 2.

8. Выдача результата. Результатом работы алгоритма является разбиение S , состоящее из найденных аномальных кластеров, а также списки центров кластеров `c_list` и весов `w_list`.

Описанный алгоритм 4 является первым этапом, выполняемым при аномальной инициализации $A\text{-}Ward_{p\beta}$. На следующем этапе происходит минимизация расстояния между объектами и центроидами с использованием ранее найденных центроидов и весовых коэффициентов w . Для этого применяется алгоритм $imwk\text{-}means_{p\beta}$, который представляет собой модификацию $k\text{-}means$. Ниже описана последовательность шагов $imwk\text{-}means_{p\beta}$:

А Л Г О Р И Т М # 5: $imwk\text{-}means_{p\beta}$

1. Инициализация. Установить текущее разбиение пустым $S = \emptyset$, а число кластеров K равным длине списка `c_list`, который был получен при аномальной инициализации (алгоритм 4).

2. Формирование кластеров. Каждый объект $y_i \in Y$ поместить в кластер, центр которого c_k находится ближе всего к этому объекту. Близость объекта к центру кластера определяется по формуле (4). Если нет изменений в разбиении S , перейти к шагу 5.

3. Обновление центров. Вычислить новые координаты центра c_k каждого кластера C_k как покомпонентный центр Минковского всех объектов этого кластера $y_i \in C_k$.

4. Обновление весов. Вычислить новые веса w_{kv} по формуле (5) для $k = 1, \dots, K$ и $v = 1, \dots, V$. Перейти к шагу 2.

5. Выдача результата. Результатом работы алгоритма является разбиение S ,

а также списки центров кластеров `c_list` и весов `w_list`.

Третьим этапом работы $A\text{-}Ward_{p\beta}$ является непосредственно иерархическая кластеризация, при которой число кластеров сокращается до необходимого количества. Полученное разбиение S , а также центры кластеров и весовые коэффициенты используются для инициализации. Ниже приведён алгоритм $A\text{-}Ward_{p\beta}$.

А Л Г О Р И Т М # 6: $A\text{-}Ward_{p\beta}$

1. Инициализация. Параметры p и β остаются неизменными, которые были определены для $itwk\text{-}means_{p\beta}$ (алгоритм 5). Начальное состояние соответствует конечному для $itwk\text{-}means_{p\beta}$: исходный список центров кластеров `c_list` и весов `w_list` является результатом работы предыдущего этапа.

2. Объединение кластеров. Выбрать два ближайших кластера $C_a, C_b \in S$ и объединить их в новый C_{ab} . Близость кластеров определяется по следующей формуле:

$$(6) \quad d_{Ward_{p\beta}}(C_a, C_b) = \frac{N_a N_b}{N_a + N_b} \sum_{v=1}^V \left(\frac{w_{av} + w_{bv}}{2} \right)^\beta |c_{av} - c_{bv}|^p,$$

где: N_a, N_b — количество объектов в кластерах C_a и C_b соответственно

V — число признаков у каждого объекта $y_i \in Y$

w_{av}, w_{bv} — веса v -го признака в кластере C_a и C_b соответственно

c_{av}, c_{bv} — v -ая координата центров кластеров C_a и C_b соответственно

3. Обновление центра. Вычислить новое значение центра C_{ab} как покомпонентный центр Минковского по всем объектам $y_i \in C_{ab}$.

4. Обновление весов. Вычислить новые веса w_{kv} по формуле (5) для $k = 1, \dots, K$ и $v = 1, \dots, V$.

5. Условие остановки. Уменьшить текущее число кластеров на единицу. Если текущее число кластеров все ещё больше единицы или требуемого числа кластеров, перейти к шагу 2.

Алгоритм $A\text{-}Ward_{p\beta}$ продемонстрировал свою эффективность как на искусственно сгенерированных данных, так и на реальных. Особый интерес предоставляют возможности алгоритма относительно восстановления кластеров на зашумлённых данных.

Благодаря описанным нововведениям, алгоритм рассматривает значимость различных признаков с учётом дисперсии внутри кластера.

Вопрос определения наиболее подходящих значений параметров p и β на настоящее время изучен поверхностно. В частности, известно, что параметры должны задаваться для каждого данных индивидуально и могут сильно влиять на качество получаемого результата. Предложены базовые методы для нахождения рациональных значений при помощи перебора [9]. Указанная работа важна в контексте выбора критерия оценки качества разбиения без доступа к истинному разбиению. Авторы предлагают для этого использовать эмпирическую характеристику Silhouette Width (SW).

В настоящее время ведутся исследования относительно возможного сокращения времени вычислений при подборе параметров. В анализе данных распространён метод кросс-валидации, идея которого заключается в том, что массив данных сохраняет все свои основные свойства и признаки при случайном исключении из него некоторого числа объектов. Та же идея лежит в основе предполагаемого метода ускоренного перебора параметров. Для сокращения времени одного выполнения алгоритма, осуществляется переход к небольшой подвыборке данных, сформированной случайным образом. Предполагается, что результат подбора параметров на подвыборке будет не сильно отличаться от подбора по полным данным с точки зрения качества результирующего разбиения, но при этом будет достигнута существенная экономия времени. Описанные исследования к настоящему моменту ещё не опубликованы, но если упомянутая гипотеза подтвердится, для алгоритма будут сформированы все предпосылки его применения для решения практических задач.

2.5 Алгоритм dePDDP

Алгоритм dePDDP относится к иерархическим дивизивным и представляет собой усовершенствованную версию PDDP (Principal Direction Divisive Partitioning) [10]. Как и все дивизивные алгоритмы, dePDDP начинает работу рассматривая все данные как единственный кластер, который включает в себя все объекты. Процесс продолжается до тех пор, пока не будет выполнен заданный критерий остановки.

Итерационно происходит выбор одного кластера и его разбиение на два новых. Изначально в PDDP критерий разделения был относительно простым: в один кластер выбирались те объекты, проекции которых на главную компоненту [11] лежали на по-

ложительной полуоси, а остальные объекты составляли второй кластер. Впоследствии авторами работы [12] этот критерий был пересмотрен для того, чтобы учесть распределение данных. Новое предложение заключалось в том, чтобы производить разбиение по наиболее глубокому минимуму функции плотности при проецировании данных на первую главную компоненту.

При оценке функции плотности используется метод ядерной оценки. Для заданного кластера C все признаки центрируются по объектам $y_i \in C$, после чего определяется вектор главной компоненты как сингулярный вектор, соответствующий наибольшему сингулярному значению матрицы данных. Все объекты проецируются на ось главной компоненты и по формуле (7) вычисляется оценка функции плотности Розенבלата-Парзена:

$$(7) \quad \hat{f}(x_j) = \frac{1}{n h} \sum_{i=1}^n K((x_j - x_{n(i)})/h)$$

где: n — число объектов в кластере

$x_{n(i)}$ — i -ая точка в рассматриваемом кластере из n точек

$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$ — плотность нормального распределения

$h > 0$ — параметр окна, определяемый по формуле: $h = \sigma \left(\frac{4}{3n} \right)^{\frac{1}{5}}$,
где σ — среднекв. отклонение проекций в кластере.

Указанное нововведение, помимо основной цели, оказалось удачным одновременно для разрешения двух сопряжённых проблем: выбора очередного кластера для разбиения и остановки работы алгоритма. Было замечено, что минимум функции плотности может определять не только границу разбиения в пределах одного кластера, но и служить указателем на тот кластер, который должен быть разделен на текущей итерации. Таким образом, для разбиения выбирается кластер с наименьшим минимумом среди всех терминальных кластеров.

Если кластер имеет монотонную или выпуклую функцию плотности, то он не может быть разделен по данному критерию. Это наблюдение позволило определить естественный критерий останова алгоритма: алгоритм прекращает работу как только не

останется ни одного кластера, который можно было бы разбить по критерию минимума плотности. Сформулируем алгоритм dePDDP в виде последовательности шагов:

А Л Г О Р И Т М # 7: dePDDP

1. Инициализация. Исходное число кластеров K принять равным одному. Создать новый кластер, который включает в себя все объекты данных.

2. Оценка функции плотности. Для каждого нового кластера C_k в текущем разбиении найти главную компоненту по всем $y_i \in C_k$ и спроецировать на неё все объекты кластера. Построить оценочную функцию плотности $\hat{f}_k(x_i)$, руководствуясь формулой (7). Найти наиболее глубокий минимум функции плотности (если существует).

3. Выбор кластера. Из текущего разбиения выбрать кластер C_b для разделения. Для этого найти кластер, у которого наиболее глубокий минимум достигает наименьшего значения среди всех кластеров. Если такого кластера нет, перейти к шагу 5.

4. Разделение. Разделить выбранный кластер C_b . При этом создать два новых кластера, которые содержат объекты, лежащие соответственно по левую и правую сторону от минимума функции плотности при проецировании на ось главной компоненты. Исходный кластер C_b прекращает существование. Перейти к шагу 2.

5. Выдача результата. Результатом работы dePDDP является текущее разбиение.

Рассмотрим иллюстративно работу алгоритма dePDDP на том же примере, что и A-Ward (см. рисунок 3). Данные состоят из пяти хорошо выделяемых кластеров. При этом на начальном этапе все данные принадлежат одному кластеру, который условно обозначим коричневым цветом, как показано на рисунке 3(а). Определим вектор главной компоненты (он направлен в том направлении, в котором разброс данных наибольший), который условно изображён в виде оси. Данные, спроецированные на ось главной компоненты, формируют оценку функции плотности с двумя минимумами, обозначенную на рисунке кривой линией. Выбирается наиболее глубокий минимум и осуществляется отделение синего кластера, как показано на рисунке 3(б). При этом оба

сформированных кластера имеют минимумы оценочной функции плотности. Выберем для разбиения тот, кластер, минимум которого глубже (допустим, это коричневый кластер). Очередное разбиение формирует кластер (коричневый на рисунке 3(в)), который не имеет ни одного минимума при проецировании данных на главную компоненту, это значит что кластер сформирован окончательно и выбивает из рассмотрения. Разбиение кластеров продолжается до тех пор, пока не останется ни одного кластера, который можно было бы разбить (рисунок 3(д)). Как только критерий останова выполнен, алгоритмом возвращается результирующее разбиение, показанное на рисунке ??.

Идея о разделении по минимуму функции плотности хорошо соответствует интуитивному понятию о кластерах, что было подтверждено в экспериментах с алгоритмом dePDDP на синтетических и реальных данных [12, 6].

2.6 Алгоритм BiKM-R

Алгоритм BiKM-R (Bisecting K-Means with the Random projections stopping rule) [6], так же как и dePDDP является иерархическим дивизивным. Среди дивизивных алгоритмов широкую известность получил простой алгоритм раздвоения по методу k -средних (Bisecting k -means) [13], модификацией которого является BiKM-R. Алгоритм раздвоения по методу k -средних использует квадратичный критерий для того чтобы разделить рассматриваемый кластер на два. Фактически этот подход является способом организации последовательного выполнения k -means (при $K = 2$) для того чтобы получить иерархическую структуру кластеров.

Авторами [6] были предложены следующие изменения в этот алгоритм:

- использовать аномальные кластеры для инициализации k -means
- остановку осуществлять по критерию, учитывающему число минимумов оценочной функции плотности при проецировании объектов кластера на случайные направления

Использование аномальной кластеризации применяется для того чтобы избавиться от случайной инициализации k -means, применяемого для деления кластера. Современные исследования относят необходимость инициализации к слабым сторонам k -means и утверждают о сильной зависимости результата кластеризации от правильного выбора исходных центров [11]. Благодаря применению метода аномальных класте-

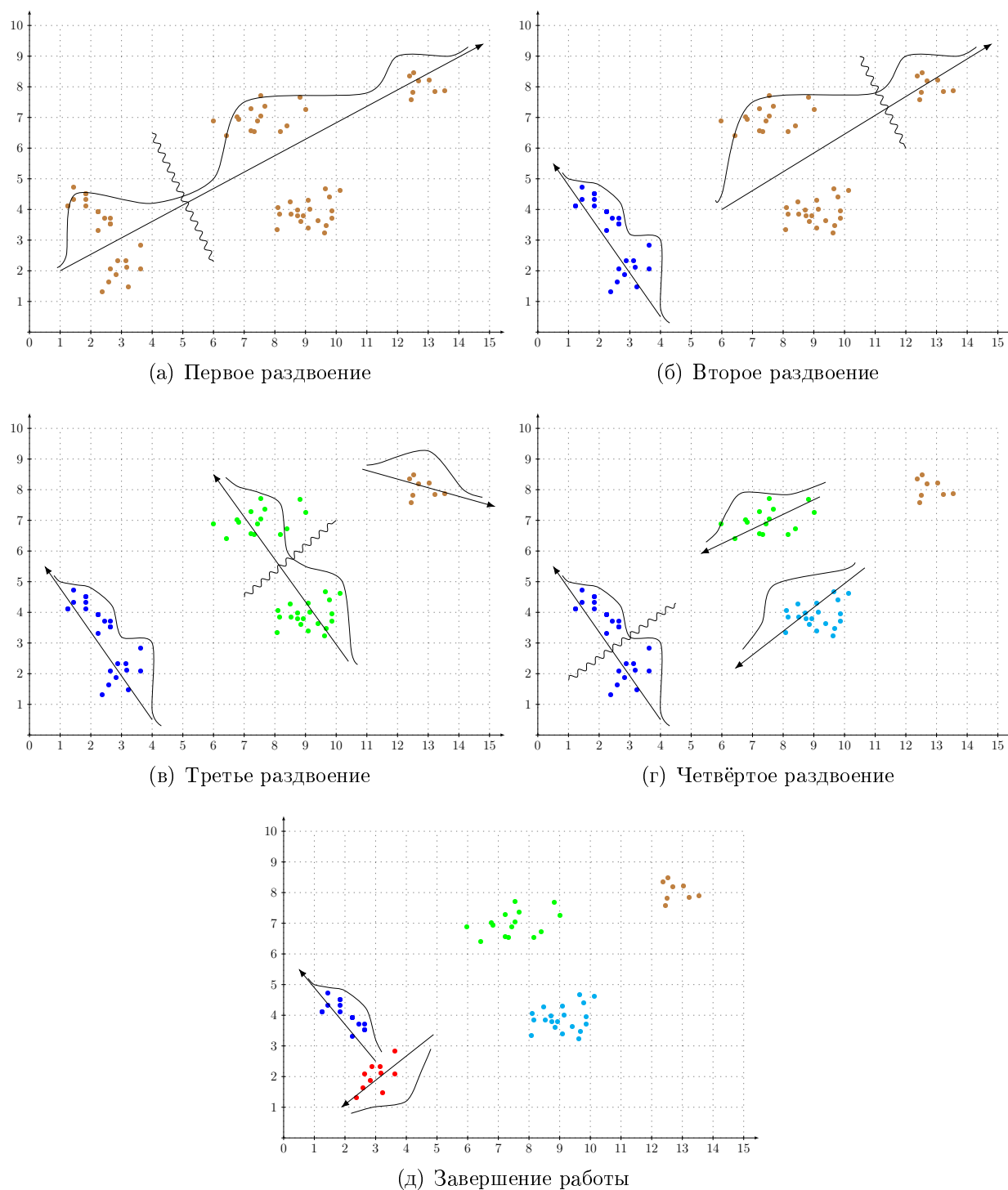


Рисунок 3 – dePDDP: Этапы работы алгоритма

Каждому кластеру соответствуют точки определённого цвета в двумерном пространстве. Осью показана первая главная компонента кластера. Чёрная кривая иллюстрирует оценку плотности данных, спроецированных на первую главную компоненту.

ров, исходные центры определены однозначно и неизменно. Причём полученные центры способствуют высокому качеству результирующего разбиения и автоматически подбираются для данных [1].

Для дивизивных методов характерно большое разнообразие критериев остановки. Например, в классических случаях применяется остановка по достижению заданного числа кластеров. Для остановки ViKM-R авторами предложен новый критерий, основанный на проецировании объектов кластеров на случайные направления. Пусть задано некоторое разбиение из K кластеров: $S = \{C_1, \dots, C_k\}$. Генерируется определённое число s случайных векторов p_t , $t = 1, \dots, s$. Вектора предлагается генерировать сферическим распределением Гаусса с математическим ожиданием в начале координат и $\sigma^2 = 1/V$, где V — число признаков. Каждый объект y_i каждого кластера C_k спроецировать на все направления p_t . Координата проекции вычисляется как скалярное произведение $x_t = \langle x, p_t \rangle$ после чего вычисляется функция плотности \hat{f}_k^t по методу ядерной оценки (формула (7)). Для каждого кластера вычисляется отношение ϵ_k числа функций \hat{f}_k^t , $t = 1, \dots, s$, которые имеют хотя бы один минимум к общему числу функций, то есть к s . Если для некоторого кластера C_k отношение ϵ_k меньше заданного пользователем порога ϵ , то кластер C_k не разбивается. В данной работе рассматривается вариант алгоритма при котором, если кластер на очередном шаге не был разбит, он не будет рассматриваться как кандидат на следующих итерациях. Однако, вообще говоря, возможен случай, когда кластер будет разбит на следующих итерациях, так как направления генерируются каждый раз случайно. Если для всех кластеров верно условие остановки $\epsilon_k < \epsilon$, то работа алгоритма прекращается. Авторы [6] описывают также использование указанного критерия для непосредственного выполнения разбиения, то есть предлагается выбирать тот кластер C_k , у которого значение ϵ_k наибольшее среди всех кластеров, при условии, что $\epsilon_k > \epsilon$ и осуществлять разбиение по наиболее глубокому минимуму. В программе INDACT реализован метод разбиения с использованием k -means при $K = 2$.

Описанный критерий остановки требует указания двух параметров. Во-первых, это число случайных направлений, на которые проецируются объекты кластеров. Авторы алгоритма без подробных пояснений предлагают принять число направлений, равным числу признаков. Такое число с одной стороны обеспечит полноту критерия, с дру-

гой стороны будет не слишком затратным с точки зрения времени вычислений. Вопрос выбора значения ϵ проработан весьма подробно с математическим обоснованием. Как оказалось, в среднем оправдан выбор значения $\epsilon = 0.32$.

Определим последовательность шагов для выполнения ViKM-R:

А Л Г О Р И Т М # 8: ViKM-R

1. Инициализация. Задаться пороговым значением ϵ , как правило, $\epsilon = 0.32$. Исходное число кластеров принять $K = 1$. Создать новый кластер, который включает в себя все объекты данных.

2. Генерация случайных направлений. Из сферического распределения Гаусса с нулевым математическим ожиданием и $\sigma^2 = 1/V$ сгенерировать $s = V$ случайных векторов.

3. Проецирование на случайные направления. Все объекты каждого кластера спроецировать на сгенерированные направления. Вычислить функцию плотности \hat{f}_k^t для каждого направления $t = 1, \dots, s$ по методу ядерной оценки (формула (7)). Для каждого кластера C_k найти долю направлений ϵ_k , функции плотности по которым имеют хотя бы один минимум.

4. Выбор кластера. Из текущего разбиения выбрать кластер C_b для которого доля ϵ_b максимальна и больше порогового значения ϵ . Если такой кластер не найден, перейти к шагу 6.

5. Разделение. Разделить кластер C_b с применением интеллектуального метода *ik-means*. Для того чтобы метод *ik-means* сгенерировал ровно два кластера, после аномальной инициализации выбрать два наибольших аномальных кластера в качестве центров *k-means*. После разделения исходный кластер C_b прекращает существование. Перейти к шагу 3.

6. Выдача результата. Результатом работы ViKM-R является текущее разбиение.

2.7 Нормализация данных

Нормализация данных играет значительную роль при применении алгоритмов кластеризации, основанных на критерии наименьших квадратов [1]. Поэтому в програм-

му INDAST включён модуль позволяющий применять различные способы масштабирования шкалы с сдвига начала координат. В данном разделе приведено математическое описание возможных вариантов выполнения нормализации данных.

2.7.1 Общая формула нормализации данных

Пусть имеются данные заданные в табличном виде:

$$(8) \quad Y = \begin{pmatrix} y_1 \\ \dots \\ y_N \end{pmatrix} = \begin{pmatrix} y_{11} & \dots & y_{1V} \\ \dots & \dots & \dots \\ y_{N1} & \dots & y_{NV} \end{pmatrix}$$

Как правило, нормализацию данных проводят отдельно для каждого признака. Обозначим v -ый столбец исходных данных в формуле (8) как Y_v , тогда:

$$Y = (Y_1, \dots, Y_v, \dots, Y_V)$$

Под нормализацией понимается преобразование исходных данных следующего вида:

$$Y \rightarrow Y' = (Y'_1, \dots, Y'_v, \dots, Y'_V),$$

$$Y'_v = \frac{Y_v - c_v}{r_v}$$

где: Y — исходные данные;

Y' — нормализованные данные;

Y'_v — нормализованный столбец данных, соответствующий исходному Y_v ;

$c_v = c(Y_v)$ — величина, зависящая от Y_v и определяющая преобразование начала отчёта;

$r_v = r(Y_v)$ — величина, также зависящая от Y_v и определяющая преобразование масштаба шкалы

2.7.2 Преобразование начала отчёта

Преобразование начала отчёта может быть произведено следующими способами:

- среднее $c_v = \frac{1}{N} \sum_{i=1}^N y_{iv}$
- минимум $c_v = \min_{i=1\dots N} y_{iv}$
- медиана $c_v = \text{median}_{i=1\dots N} y_{iv}$
- центр Минковского $c_v = \arg \min_x \sum_{i=1}^N |y_{iv} - x|^p$

В общем случае для вычисления центра Минковского при произвольном p не удаётся получить аналитическое решение. В программе INDACT используется итерационный алгоритм градиентного спуска.

2.7.3 Преобразование масштаба шкалы

Для преобразования масштаба шкалы предусмотрено на выбор три варианта:

- полуразмах $r_v = \frac{1}{2}(\max_i y_{iv} - \min_i y_{iv})$
- стандартное отклонение $r_v = \sqrt{\frac{\sum_{i=1}^N (y_{iv} - \bar{y}_v)^2}{N-1}}$
- абсолютное отклонение $r_v = \frac{1}{N} \sum_{i=1}^N |y_{iv} - \text{median}_{i=1\dots N} y_{iv}|$

2.8 Генератор синтетических данных

В программе INDACT предусмотрена возможность проведения численных экспериментов на синтетических данных. В данном разделе будет описан современный подход к генерации данных с использованием небольшого количества параметров, предложенный авторами статьи [6].

Как правило, синтетические данные генерируются либо с хорошо различимыми кластерами [14], либо для генерации применяется сложный алгоритм, в котором настраивается множество различных параметров [15]. Преимущества подхода, используемого в INDACT, заключаются в возможности одновременного регулирования разброса объектов внутри кластера и взаимное смешивание кластеров при помощи единственного параметра.

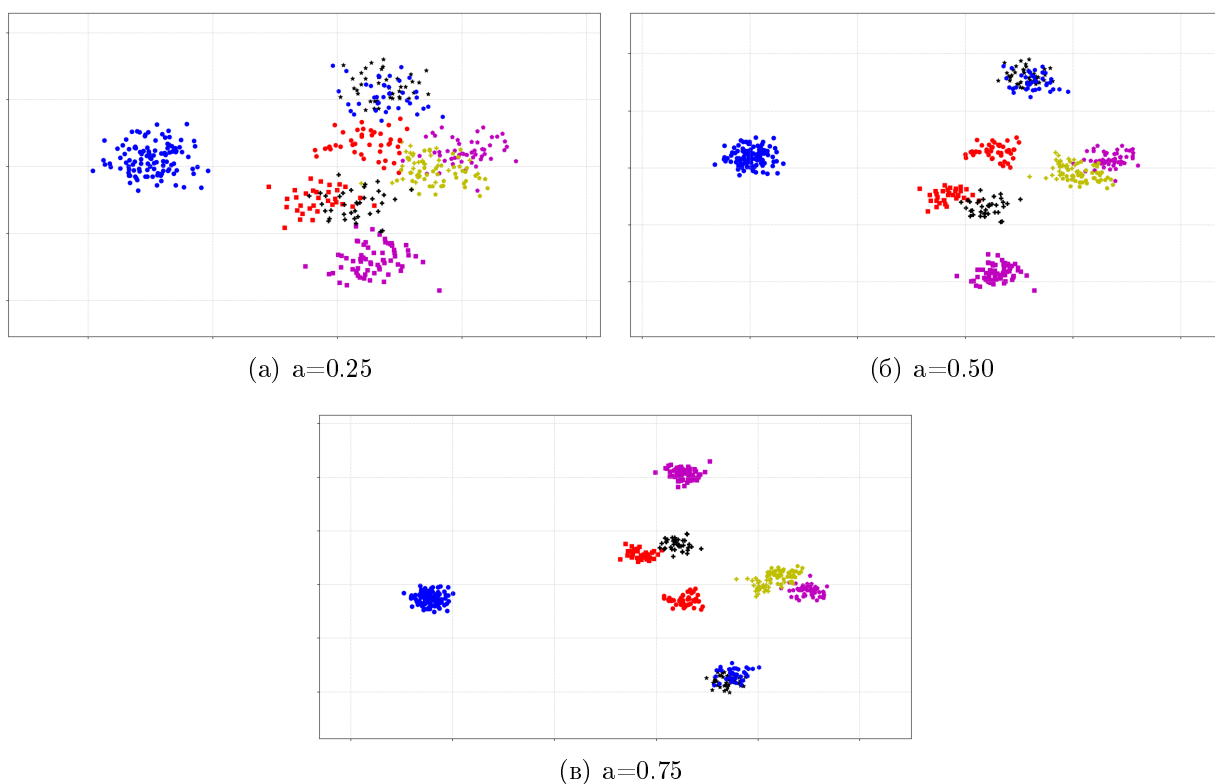


Рисунок 4 – Сгенерированные данные для трёх случаев параметра a
 Для трёх значений параметра $a = 0.25, 0.50, 0.75$ приведена диаграмма, изображающая проекции сгенерированных данных на главные компоненты. Данные сгенерированы при числе признаков, равным 10; общее число объектов — 500; минимальная численность кластера — 30.

Структура получаемых данных представляет собой заданное количество гауссовых кластеров K , сформированных при фиксированном общем числе объектов N и признаков V и заданным минимальным количеством объектов в каждом кластере m . Остаток нераспределенных объектов $\delta = N - K \cdot m$ размещается случайно и равномерно по всем кластерам. Для этого генерируется $K - 1$ псевдослучайных чисел в диапазоне от 0 до 1, к сгенерированным числам добавляется 0 и 1, после чего они упорядочиваются и вычисляется разность между соседними элементами в полученной последовательности. Эти разности определяют долю от нераспределенных объектов, которые будут дополнительно включены в K кластеров. Таким образом достигается строгое соблюдение общего числа объектов и равномерное распределение остатка по кластерам.

При определённой численности, кластер генерируется из многомерного распределения Гаусса. Среднее кластера выбирается случайно при равномерном распределении из множества $[-a, a]^V$, где $a \in [0, 1]$ — параметр, влияющий на взаимное смешивание

кластеров: чем меньше a , тем сложнее разделить сгенерированные кластеры. Ковариационная матрица формируется как диагональная с элементами, сгенерированными случайно из равномерного распределения в диапазоне $[0.05a, 0.1a]$. На рисунке показаны примеры сгенерированных данных

2.9 Интерпретация результатов

В системе INDACT встроен модуль генерации отчётов, который позволяет пользователю получить основную информацию относительно каждого кластера и разбиения в целом. В данном разделе будут описаны математические зависимости, которые лежат в основе характеристик, используемых в отчёте.

2.9.1 Оценка качества разбиений

При проведении кластер-анализа часто встаёт задача определения качества полученного разбиения. Например, вопрос о качестве разбиения может возникнуть при выборе наиболее подходящего алгоритма для анализируемых данных. В программе INDACT применяется две методики: индекс ARI и эмпирическая характеристика SW. Первая методика используется при сравнении разбиения с заданным эталонным. Эмпирическая характеристика SW не зависит от базового разбиения и может быть применена для одного разбиения при неизвестном эталонном. Однако, следует понимать, что SW не учитывает специфики предметной области и может в частных случаях давать искажённое представление о качестве разбиения. С другой стороны, на синтетических данных, для хорошо разделимых гауссовых кластеров, характеристика SW достаточно точно воспроизводит поведение индекса ARI относительно истинного разбиения, которое известно исходя из генерации кластеров.

2.9.2 Характеристика Silhouette Width

Эмпирическая характеристика Silhouette Width (SW) основывается на общем представлении о сильной близости внутри группы и хорошей разделимости кластеров. SW [16] принимает значения от -1 до 1 и для отдельного объекта вычисляется по следующей формуле:

$$SW(y_i) = \frac{b(y_i) - a(y_i)}{\max\{a(y_i), b(y_i)\}},$$

где: $a(y_i)$ — среднее расстояние между объектом $y_i \in C_k$ и всеми объектами, принадлежащими тому же кластеру C_k , что и y_i ;
 $b(y_i)$ — наименьшее среднее расстояние между объектом $y_i \in C_k$ и объектами, которые принадлежат другим кластерам.

Для разбиения характеристика SW определяется как среднее по всем объектам: $SW(S) = \frac{1}{N} \sum_{k=1}^K \sum_{i \in C_k} SW(y_i)$. Значения, наиболее близкие к 1 соответствуют наилучшим разбиениям. Как показали эксперименты, характеристика SW ведёт себя примерно тем же образом что и широко известный индекс ARI [17], который позволяет оценить качество разбиения, сравнивая его с истинным.

2.9.3 Adjusted Rand Index

Индекс ARI (Adjusted Rand Index) [17] является популярным способом сравнения эталонного и заданного разбиения. В условиях проводимого эксперимента используются синтетические данные, для которых известно истинное разбиение, поэтому для оценки эффективности применения эмпирической характеристики SW можно задействовать ARI. Формула для вычисления индекса записывается следующим образом:

$$ARI = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}{\frac{1}{2} \left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2} \right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2} \right] / \binom{n}{2}}$$

где: $n_{ij} = |C_i \cap C_j|$ — число объектов, входящий одновременно в i -ый кластер в первом разбиении и в j -ый — во втором.

$a_i = \sum_{j=1}^K |C_i \cap C_j|$ — число объектов, ходящих в i -ый кластер в первом разбиении

$b_j = \sum_{i=1}^K |C_i \cap C_j|$ — число объектов, ходящих в j -ый кластер во втором разбиении

Как и характеристика SW, индекс ARI принимает значения от -1 до 1. ARI достигает 1 только в том случае, если два разбиения совпадают.

2.9.4 Характеризация кластеров

По каждому полученному кластеру C_k в отчёте INDACT приводится его центр, вычисленный для каждого признака как среднее по всем объектам кластера:

$$c_{kv} = \frac{1}{|C_k|} \sum_{i \in C_k} y_{iv}$$

Центр кластера может вычисляться как по нормализованным, так и по исходным признакам. Для исходных признаков часто полезно знать разницу между глобальным центром данных и средним по кластеру, как в абсолютном выражении, так и в процентном:

$$D_{kv} = c_{kv} - \frac{1}{N} \sum_{i=1}^N y_{iv} \qquad D_{kv}, \% = \frac{D_{kv}}{1/N \sum_{i=1}^N y_{iv}} \cdot 100\%$$

Значимость кластера описывается его вкладом в квадратичный разброс данных. Квадратичный разброс данных и вклад в него заданного кластера C_k вычисляется по формулам, приведённым ниже:

$$T(Y) = \sum_{i=1}^N y_{iv}^2 \qquad contrib_k, \% = |C_k| \frac{\leq c_k, c_k >}{T(Y)} \cdot 100\%$$

Общий вклад кластеров в квадратичный разброс данных может быть получен суммированием вклада по каждому из кластеров: $contrib, \% = \sum_{k=1}^K contrib_k, \%$. Для того чтобы лучше понять что из себя представляют кластеры, в отчёте для каждого кластера отмечаются и выписываются признаки, которые превосходят по относительной разности заданный порог Θ_D . По-умолчанию пороговое значение выставлено равным 30%. Таким образом, в заданном кластере C_k признаки условно разделяются на:

- большие: $\{v : D_{kv}, \% > \Theta_D\}$
- маленькие: $\{v : -D_{kv}, \% > \Theta_D\}$

Вклад каждого кластера в квадратичный разброс данных может быть расписан по каждому признаку отдельно. Для определения вклада признака v в квадратичный разброс данных внутри кластера C_k применяется следующая формула:

$$contrib_{kv}, \% = |C_k| \frac{c_{kv}^2}{T(Y)} \cdot 100\%$$

Сумма вклада $contrib_{kv}, \%$ по всем признакам внутри кластера C_k равна общему

вкладу этого кластера в разброс данных: $\sum_{v=1}^V contrib_{kv}, \% = contrib_k, \%$. Для вычисления относительного вклада вычисляется вклад каждого признака v без учёта разбиения $contrib_v$ и определяется отношение вклада внутри кластера $contrib_{kv}, \%$ к $contrib_v$.

$$contrib_v, \% = \frac{\sum_{i=1}^N y_{iv}^2}{T(Y)} \qquad contrib_{kv}^{relative} = \frac{contrib_{kv}}{contrib_v}$$

2.10 Общие рекомендации по выбору алгоритма

В разделах 2.2—2.6 описаны алгоритмы, реализованные программной системой INDACT. Полный перечень алгоритмов, которые можно применить к данным, представлен ниже:

1. *ik*-means
2. A-Ward
3. A-Ward_{pβ}
4. dePDDP
5. BiKM-R

Алгоритмы *ik*-means, dePDDP, BiKM-R находят число кластеров автоматически, причём по-умолчанию в них используются определённые значения параметров, выбор которых определяет число получаемых кластеров. В *ik*-means по-умолчанию задан порог $\Theta = 1$ минимального числа объектов в аномальной группе, в алгоритме dePDDP задействована величина “окна Парзена” h , определяемая как указано в пояснениях к формуле (7), а в BiKM-R предопределена доля некорректных направлений $\epsilon = 0,32$. Значения параметров в dePDDP и BiKM-R найдены экспериментально и, как правило, не требуют регулировки. Значение параметра в *ik*-means $\Theta = 1$ обычно приводит к избыточному числу кластеров, что лежит в основе алгоритмов A-Ward и A-Ward_{pβ}, которые начинают именно с этих кластеров и последовательно их объединяют. При этом алгоритм A-Ward_{pβ} включает два параметра, настраиваемых вручную. Его использование рекомендуется только в случае, когда ожидается, что признаки слабо соответствуют искомой кластерной структуре. Нормально, пользователю следует начать с

алгоритма *ik*-means и применять A-Ward для последовательного объединения полученных кластеров, если *ik*-means дал слишком большое число кластеров. Если ожидается, что в данных много случайных, нехарактерных объектов, то лучше применять метод BiKM-R.

3 Пользовательское описание программы INDACT

С точки зрения пользователя программная система INDACT представляет собой оконное приложение с графическим интерфейсом, в целом аналогичным интерфейсу большинства современных программ. Программа имеет одно основное окно, в котором одновременно отображаются и обрабатываемые и нормализованные данные, а также множество вспомогательных и диалоговых окон, возникающие в результате взаимодействия с пользователем.

INDACT распространяется в виде каталога бинарных файлов, который содержит все необходимые библиотеки уже скомпилированные для операционной системы, поэтому от пользователя не требуется устанавливать никакие дополнительные компоненты (в том числе интерпретаторы или модули).

В данном разделе продемонстрированы основные элементы интерфейса, а также некоторые диалоговые окна. Подробнее все функции системы описаны в руководстве пользователя. Также в руководстве пользователя предметно указаны требования к характеристикам персонального компьютера (ПК) и программному обеспечению, однако, кратко отметим, что INDACT работает практически на всех современных ПК под управлением Microsoft Windows 7 и выше.

3.1 Этапы работы с программой

Работа с программой в типичном случае состоит из четырёх базовых этапов, которые пользователь проходит последовательно как изображено на рисунке 5. Тем не менее, это не означает что указанная последовательность жёстко фиксирована и нет возможности вернуться на предыдущий этап. Если пользователь допустил ошибку или намерен опробовать различные варианты, он может заново вызвать функции с произвольного предыдущего этапа.

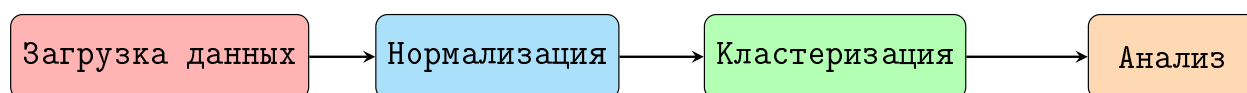


Рисунок 5 – Основные этапы работы с программой

На этапе загрузки данных пользователь должен выбрать текстовый файл с данными и загрузить его в программу. Текстовый файл представляет собой таблицу данных,

каждая строка которой соответствует строке файла, а столбцы разделены запятыми. После загрузки файла графический интерфейс отобразит загруженные данные и они станут доступны для работы. Единоновременно программой предусмотрена работа только с одним файлом.

Нормализация является подготовительным этапом, тем не менее, этот этап очень важен для успешной работы алгоритмов кластеризации. В диалоговом окне доступен выбор параметров нормализации описанных в разделе 2.7. Программа INDACT допускает изменение настроек нормализации практически на любом этапе обработки данных. В рамках нормализации в программной системе также рассматривается отбор признаков, которые будут участвовать в кластеризации. При отборе также решается вопрос о включении номинальных признаков, которые автоматически разбиваются на бинарные, после чего нормируются в соответствии с заданными настройками.

Этап кластеризации состоит в выборе подходящего алгоритма, задания параметров (при необходимости) и его выполнении. После выполнения алгоритма становится известно разбиение, которое ставит в соответствие каждому объекту номер кластера, которому объект принадлежит. Программа INDACT допускает работу одновременно с несколькими разбиениями. Таким образом, можно получить несколько различных результатов для разных значений параметров или алгоритмов и после этого приступить к анализу, выбрав наиболее подходящее разбиение.

Анализ результатов заключается в оценке полученных разбиений а также интерпретации кластеров. Для этого используется модуль генерации отчёта. Результирующий отчёт, нормированные и исходные признаки, а также признак, определяющий кластерную принадлежность объектов можно сохранить в текстовый файл. Кроме того, в рамках анализа функционалом системы предусмотрена возможность построения поля рассеивания по указанным признакам, а также SVD диаграммы с цветовым выделением кластеров и гистограммы заданного признака.

3.2 Основные сведения о пользовательском интерфейсе

Главное окно программы INDACT (см. рисунок 6) содержит главное меню (1), панель управляющих кнопок (2) и две таблицы: исходных (3) и нормализованных данных (4) .

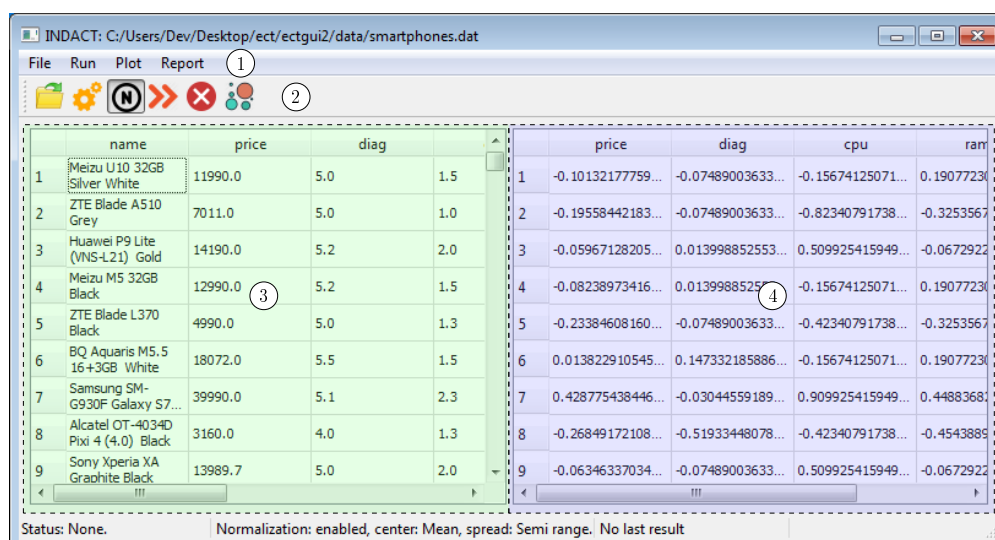


Рисунок 6 – Главное окно программы INDACT

Главное меню содержит большинство операций программы и имеет структуру вложенных подменю как показано на рисунке 7.

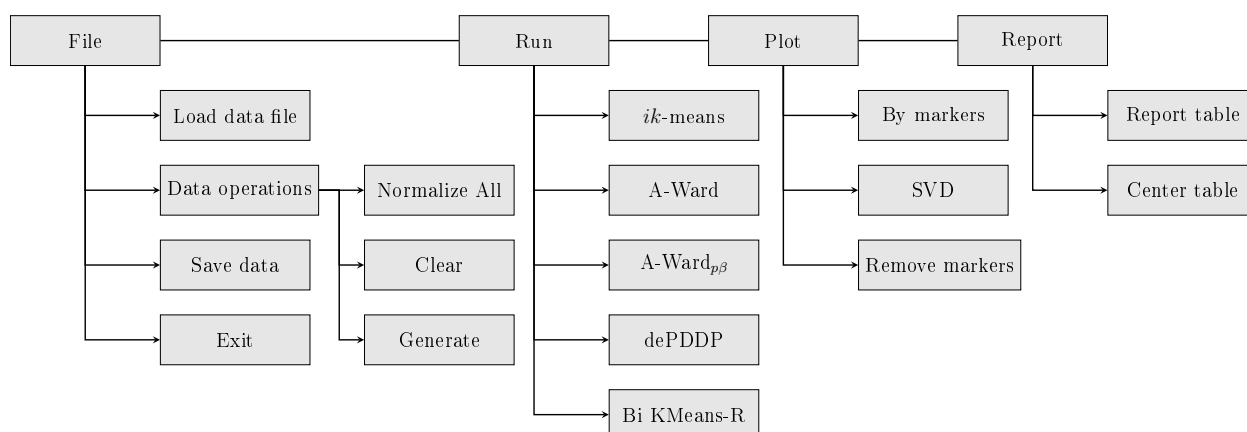


Рисунок 7 – Структура главного меню







Меню File содержит основные операции, такие как загрузка файла данных, операции с данными, сохранение данных и выход из программы. Операции с данными включают одновременную нормализацию нескольких признаков, очистку панели нормализованных данных и генерацию синтетических данных.

Пункт меню Run предоставляет пользователю возможность выбора одного из 5 реализованных алгоритмов. При выборе пункта меню, соответствующего алгоритму, при необходимости откроется окно ввода параметров алгоритма (некоторые алгоритмы, например, dePDDP, не требуют настройки), через которое осуществляется запуск на выполнение.

Меню Plot содержит базовые операции для графического представления данных. Например, у пользователя есть возможность присвоить признакам маркеры X и Y, определяющие признак, который будет отложен по оси абсцисс и ординат соответственно при построении поля рассеивания. Также пользователь может оценить общую структуру многомерных данных при помощи SVD диаграммы.


Для генерации отчёта предназначен пункт главного меню Report. Этот пункт содержит два действия для построения текстового отчёта и отчёта по центрам кластеров.

Вызов некоторых команд осуществляется с панели управляющих кнопок. Предусмотрены управляющие кнопки для следующих действий:

-  загрузить файл
-  настройки нормализации
-  вкл./выкл. нормализацию
-  нормализовать выбранные признаки
-  удалить выбранные признаки
-  отобразить/скрыть раздел результатов

3.3 Загрузка данных

Загрузка данных — начальный этап работы с программой. Для того чтобы указать программе из какого файла следует извлечь данные, пользователь выполняет процедуру загрузки текстового файла. Загружаемый файл должен отвечать определенным требованиям, в частности, каждая строка файла соответствует одному объекту и значения признаков разделены запятой. Расширение имени файла при загрузке во внимание не принимается, однако важно чтобы файл быть текстовым, а не бинарным. Подробно требования к файлу описаны в руководстве пользователя.

Для загрузки файла пользователь выбирает меню File → Load data file. Это действие приводит к открытию стандартного диалога загрузки файла. Того же эффекта можно добиться нажатием иконки  на управляющей панели.

На рисунке 8 показан диалог загрузки в операционной системе Windows. Пользователь должен проследовать в файловом диалоге по пути к загружаемому файлу и выделить его, после чего подтвердить выбор.

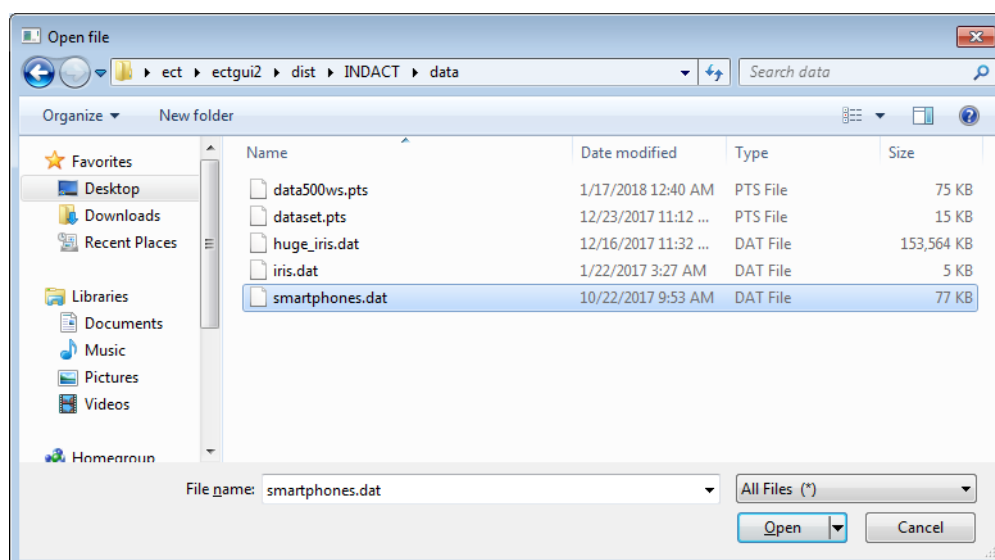


Рисунок 8 – Диалог загрузки данных

Загруженные данные отображаются в программе в таблице исходных данных. Пользователю следует убедиться, что загружен правильный файл, объекты и признаки отображаются верно. После успешной загрузки пользователь может начинать отбор признаков и их нормализацию.

3.4 Генерация синтетических данных

Генерация данных включена в состав системы для проведения численных экспериментов. Используется метод генерации, описанный в разделе 2.8, который позволяет зафиксировать наиболее значимые характеристики кластерной структуры.

Чтобы запустить модуль генерации данных, пользователь выбирает пункты меню File → Data operations → Generate. Диалоговое окно генератора данных, показанное на рисунке 9, позволяет ввести значения характеристик в текстовые поля в нижнем правом углу окна. Сразу же после ввода будет обновлена SVD диаграмма сгенерированных данных. Таким образом, пользователь может непосредственно оценивать влияние изменения параметра и подбирать значения, наиболее подходящие для его задачи.

Окно генератора требует ввода следующих 6 числовых параметров:

- Seed generation — начальная установка генератора случайных чисел, служит для обеспечения повторяемости результата;
- Minimum cluster cardinality — минимальное число объектов в каждом кластере;

- Number of clusters — число генерируемых кластеров;
- Number of objects — число объектов, распределяемых по кластерам;
- Features — количество признаков;
- Box parameter (α) — параметр, определяющий степень взаимного смешивания кластеров, чем больше значение, тем легче разделить сгенерированные кластеры.

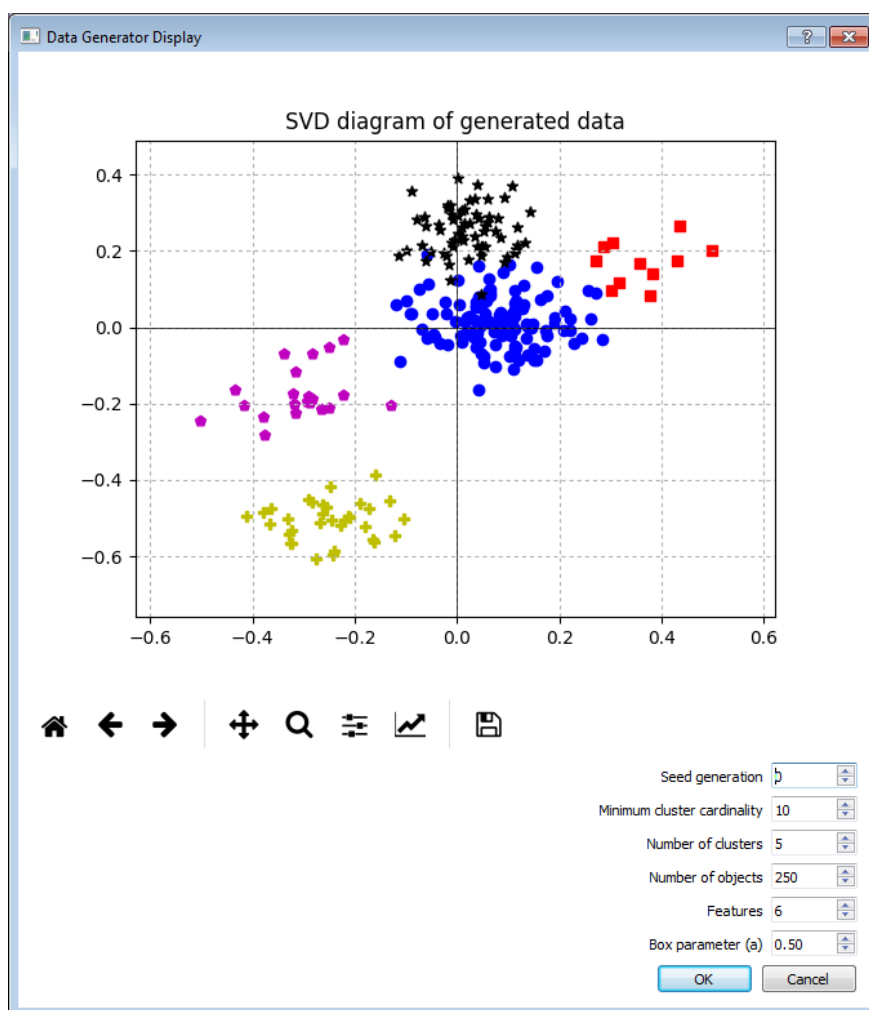



Рисунок 9 – Диалог генератора данных

После подтверждения ввода пользователю будет предложено выбрать путь для сохранения сгенерированных данных, а также выбрать способ сохранения признака, обозначающего кластерную принадлежность объекта. Этот признак можно сохранить в тот же файл, в котором содержатся сгенерированные данные в качестве отдельного столбца или сохранить его в другом текстовом файле.

3.5 Нормализация данных

Нормализация данных осуществляется в два этапа. На первом этапе происходит настройка параметров: способа преобразования начала отчёта и масштаба. Второй этап заключается в выборе признаков, которые будут участвовать в кластеризации. Для этого требуется перенести соответствующие признаки на панель нормализованных данных. Чтобы нормализовать отдельный признак, в контекстном меню следует выбрать пункт *Normalize*. Если необходимо нормализовать сразу группу признаков, удобнее воспользоваться кнопкой **»**, что позволит отметить для нормализации сразу несколько признаков. Изменить настройки нормализации можно нажав кнопку . При этом новые настройки применяются как для тех признаков, которые уже находятся в области нормализованных, так и для вновь нормализуемых. Окно настройки параметров показано на рисунке 10. В качестве справочной информации в это окно встроена основная формула нормализации (8). Отдельным флажком *Normalization enabled* переключается текущее состояние: *вкл./выкл.* Если нормализация выключена, то все выбранные признаки будут использованы в том виде, в котором они были загружены из файла.

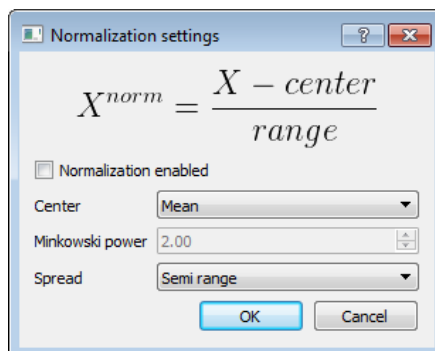


Рисунок 10 – Диалог настройки нормализации

Для задания параметров нормализации доступны следующие значения:

Center:

- No centring
- Mean
- Minimum
- Median
- Minkowski center

Mean:

- Unity
- Semi range
- Standard deviation
- Absolute deviation

Текстовое поле Minkowski power становится активно только в том случае, если для центрирования выбрано значение “Minkowski center”. Нормализованные данные отображаются в правой панели, как показано на рисунке 11.

	name	price	diag	
1	Meizu U10 32GB Silver White	11990.0	5.0	1.5
2	ZTE Blade A510 Grey	7011.0	5.0	1.0
3	Huawei P9 Lite (VNS-L21) Gold	14190.0	5.2	2.0
4	Meizu M5 32GB Black	12990.0	5.2	1.5
5	ZTE Blade L370 Black	4990.0	5.0	1.3
6	BQ Aquaris M5.5 16+3GB White	18072.0	5.5	1.5
7	Samsung SM-G930F Galaxy S7...	39990.0	5.1	2.3
8	Alcatel OT-4034D Pixi 4 (4.0) Black	3160.0	4.0	1.3
9	Sony Xperia XA Graphite Black	13989.7	5.0	2.0

	price	diag	cpu	ram
1	-0.10132177759...	-0.07489003633...	-0.15674125071...	0.1907723...
2	-0.19558442183...	-0.07489003633...	-0.82340791738...	-0.3253567...
3	-0.05967128205...	0.013998852553...	0.509925415949...	-0.0672922...
4	-0.08238973416...	0.013998852553...	-0.15674125071...	0.1907723...
5	-0.23384608160...	-0.07489003633...	-0.42340791738...	-0.3253567...
6	0.013822910545...	0.147332185886...	-0.15674125071...	0.1907723...
7	0.428775438446...	-0.03044559189...	0.909925415949...	0.4488368...
8	-0.26849172108...	-0.51933448078...	-0.42340791738...	-0.4543889...
9	-0.06346337034...	-0.07489003633...	0.509925415949...	-0.0672922...

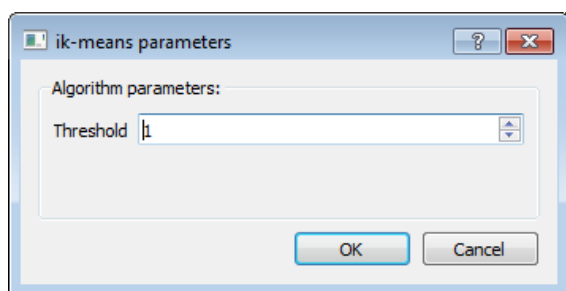
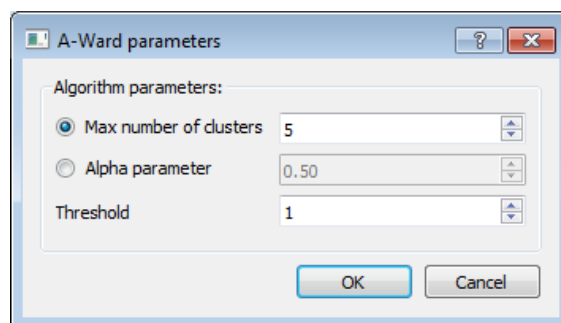
Status: None. Normalization: enabled, center: Mean, spread: Semi range. No last result

Рисунок 11 – Отображение нормализованных данных

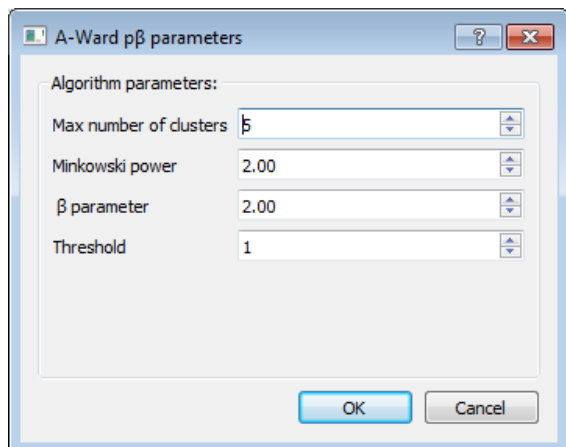
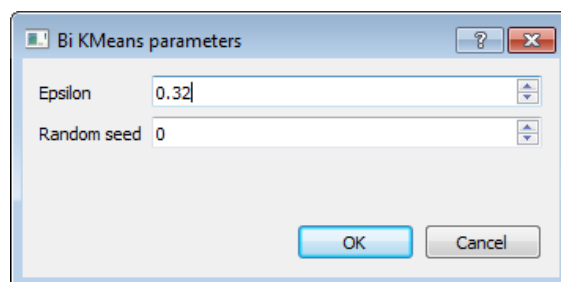
3.6 Кластеризация

Нормализованные данные готовы для выполнения алгоритма кластеризации. Для получения кластерного разбиения требуется выбрать один из алгоритмов кластеризации из меню Run. Каждый алгоритм, за исключением dePDDP обладает своим окном настроек параметров. Окна настроек параметров алгоритмов приведены на рисунке 12. Для получения более подробной информации относительно настройки параметров в диалоговых окнах следует обратиться к руководству пользователя.

После ввода необходимых параметров и подтверждения ввода запускается алгоритм кластеризации. На относительно небольших данных (до 500 объектов) кластеризация обычно не занимает более 20–30 сек. При успешной отработке алгоритма автоматически откроется дополнительная панель результатов. Благодаря этой панели имеется возможность хранить сразу несколько разбиений, переключаясь между ними чтобы исследовать свойства каждого из них. На рисунке 13 приведён вид окна с открытой панелью результатов после однократного запуска dePDDP. Как можно видеть на этом рисунке, панель содержит столбец dePDDP_1, который отображает соответствие каждого объекта одному из сформированных кластеров, используя для этого его номер.

(a) *ik-means*

(б) A-Ward

(в) A-Ward_{pβ}

(г) BiKM-R

Рисунок 12 – Диалоговые окна для задания параметров алгоритмов

INDACT: C:/Users/Dev/Desktop/ect/ectgui2/data/smartphones.dat

File Run Plot Report

	name	price		price	diag		dePDDP_1
1	Meizu U10 32GB Silver White	11990.0	1	-0.10132177759...	-0.07489003	1	1
2	ZTE Blade A510 Grey	7011.0	2	-0.19558442183...	-0.07489003	2	1
3	Huawei P9 Lite (VNS-L21) Gold	14190.0	3	-0.05967128205...	0.013998852	3	2
4	Meizu M5 32GB Black	12990.0	4	-0.08238973416...	0.013998852	4	1
5	ZTE Blade L370 Black	4990.0	5	-0.23384608160...	-0.07489003	5	1
6	BQ Aquaris M5.5 16+3GB White	18072.0	6	0.013822910545...	0.147332185	6	1
7	Samsung SM-G930F Galaxy S7...	39990.0	7	0.428775438446...	-0.03044559	7	3
8	Alcatel OT-4034D	3160.0	8	-0.26849172108...	-0.51933448	8	1

Status: Nor Normalization: enabled, center: Mean, spread: Semi rang Last result: (1.47 s) dePDDP with no paramete...

Рисунок 13 – Отображение результата кластеризации

Примечательно, что результат кластеризации может быть получен и косвенным образом. Например, данные уже содержат признак, который определяет разбиение на множестве объектов. В таком случае этот признак можно занести в панель результатов из контекстного меню, выбрав пункт To labels. Этот результат будет обладать тем же функционалом для анализа, что и результаты, полученные путём выполнения алгоритма кластеризации. Таким образом, программа может быть использована как средство анализа результата, полученного сторонними средствами и сохранённым в текстовом файле.

3.7 Анализ результатов

Для анализа результатов программой INDACT генерируется настраиваемый текстовый отчёт. В отчёт включены основные характеристики кластеров и разбиения в целом, в том числе оценки качества. Для вывода диалогового окна (рисунк 14) с текстовым отчётом требуется в главном меню выбрать пункты Report → Report table. При перед открытием окна у пользователя будет возможность выбрать то, разбиение относительно которого требуется провести анализ. Как было упомянуто выше в разделе 3.6, программа может одновременно хранить несколько разбиений, однако в момент построения отчёта требуется указать одно из доступных.

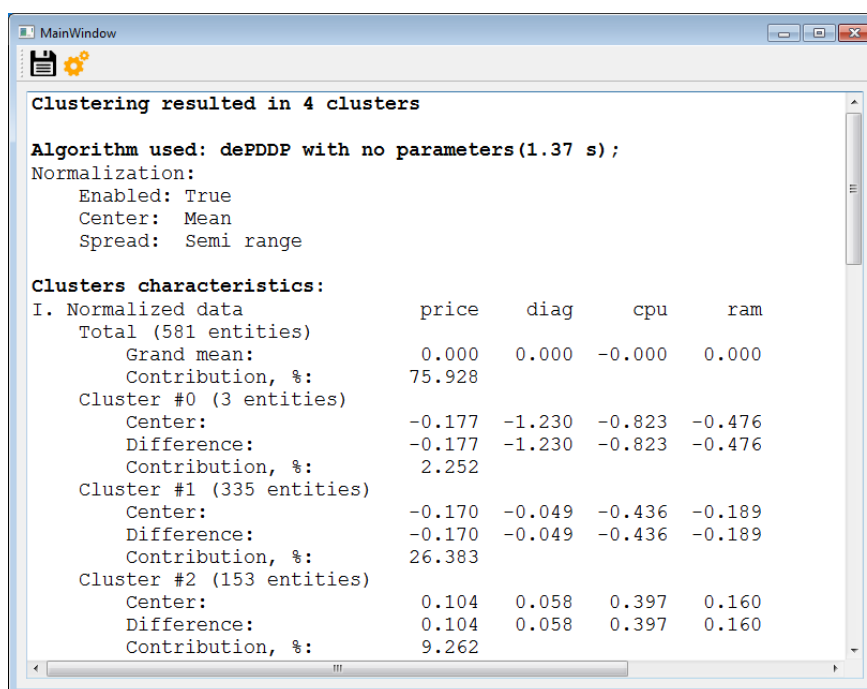




Рисунок 14 – Диалоговое окно отчёта

Общая структура отчёта представлена ниже:

- Число полученных кластеров
- Применённый алгоритм с указанием параметров
- Используемые настройки нормализации
- (Опционально) Оценка разбиений: SW, ARI.
- Характеристика кластеров в нормализованных признаках
- Характеристика кластеров в исходных признаках
- Перечень отличительных признаков кластеров
- Таблица вклада каждого кластера в квадратичный разброс данных по признакам
- Таблица вклада каждого признака в квадратичный разброс без учёта кластеризации
- Таблица относительного вклада каждого кластера в квадратичный разброс данных по признакам
- (Опционально) Таблица сопряженности
- (Опционально) Перечень объектов заданных кластеров

По-умолчанию в отчёт не включаются величины, вычисление которых может занять продолжительное время. Например, к таким величинам относятся характеристики разбиений SW и ARI. Включить вычисление этих характеристик а также задать эталонное разбиение можно при помощи диалогового окна настроек, которое вызывается нажатием кнопки  в окне отчёта. Когда отчёт сформирован надлежащим образом, пользователь может сохранить его в текстовый файл при помощи кнопки .

Также к этапу анализа можно отнести построение SVD диаграммы, поля рассеивания и гистограммы. SVD диаграмма может быть построена при помощи выбора в главном меню пункта Plot → SVD. Для построения поля рассеивания от пользователя требуется предварительно выставить так называемые метки, которые определяют роль признака. Метка X задаёт признак по оси абсцисс, а метка Y — по оси ординат. Задание

меток осуществляется из контекстного меню соответствующего признака (подробнее см. руководство пользователя). Функция построения гистограммы также входит в контекстное меню признака.

4 Структурное описание программы INDACT

В данном разделе рассматриваются технические особенности системы INDACT. Перед разработчиком стоит задача написать программу, реализующую сложные алгоритмы вычислений и предоставить пользователю удобный и современный графический интерфейс. При этом желательно, чтобы требования к конфигурации ПК, на котором будет запускаться программы были как можно менее строгими.

Для решения этой задачи потребуется аккуратный выбор программных средств, таких как язык программирования и программные библиотеки. От выбора программных средств во многом зависит трудоёмкость создания продукта и его некоторые потребительские качества, например, кроссплатформенность и необходимость установки дополнительных компонент. Кроме того удачный выбор используемых программных библиотек позволит существенно сократить объем вновь разрабатываемого кода, что положительно скажется на отказоустойчивости продукта и уменьшит возможность возникновения ошибок.

4.1 Выбор методологии разработки

Разработка программных систем осуществляется как правило в рамках какой-либо методологии. Методология определяет общую концепцию организации работ, роль заказчика в процессе разработки, порядок внесения изменений и выпуска версий. Конечно, преимущества различных методологий в основном заметны при работе над крупными и затратными проектами, которые требуют координации усилий больших коллективов специалистов. Тем не менее, даже при разработке относительно несложных программ полезно придерживаться определённой дисциплины в организации работ, что позволит написать структурированный код, пригодный к поддержке на протяжении долгого времени.

В настоящее время наиболее популярны методологии Agile, TDD, RAD и другие. Гибкая методология Agile [18] ориентирована на динамическое формирование требований и итеративную разработку. Говорить о методологии Agile целесообразно в том случае, если над программным продуктом трудится несколько небольших групп разработчиков. В начале 2000-ых появилась и начала интенсивно развиваться методология разработки через тестирование (test-driven development TDD)[19], которая в некото-

рой степени является противоположностью Agile. В ней подразумевается написание тестов до того как будет готов тестируемый программный код, соответственно требования должны быть известны заранее. При аккуратном следовании методологии TDD разработанный продукт отличается высоким качеством и устойчивостью к сбоям, однако затраты на написание тестов могут существенным образом замедлить разработку. В условиях сжатых сроков при небольшой стоимости продукта наиболее эффективно показала себя методология быстрой разработки приложений RAD (rapid application development) [20]. RAD основана на инкрементальном прототипировании и широком применении средств визуального проектирования. Как и в случае с Agile, RAD позволяет изменять требования к системе по мере выработки мнения заказчика относительно того или иного технического решения. В целом RAD отвечает ограничениям, которые следует принять во внимания при разработке INDACT: необходимо выполнить проект в сжатые сроки; изначально требования к программе определены нечетко, возможно их изменение; проект выполняется без финансирования; графическому интерфейсу пользователя отводится важная роль.

4.2 Выбор программных средств

4.2.1 Язык программирования

Выбору языка программирования следует уделить большое внимание. В настоящее время для решения различных задач программисту доступен широкий выбор языков программирования. Существующие языки программирования сильно различаются по возможностям, гибкости и области применения. Многие языки, ранее считавшиеся передовыми, в наше время практически не применяются, так как они были вытеснены более современными. Сразу ограничимся наиболее распространёнными языками, что позволит, в случае необходимости, производить доработку программы разными программистами.

Аналитическая фирма RedMonk регулярно отслеживает тенденции в области информационных технологий и публикует рейтинг языков программирования. Рейтинг составляется с учётом числа заданных вопросов на известном информационном ресурсе stackoverflow.com и количестве проектов, загруженных на хостинг github.com. Указанные ресурсы пользуются большой популярностью в информационном сообществе и рейтинг, составленный таким образом, вполне подходит для оценки популярности

языков программирования. Согласно рейтингу RedMonk за 2018 год [21], наиболее распространены следующие языки:

1. JavaScript
2. Java
3. Python
4. PHP
5. C#
6. C++

Исключим из приведённого перечня JavaScript и PHP, которые востребованы в большей степени для разработки веб-проектов. Таким образом, язык программирования для решения поставленной задачи будем выбирать из Java, Python, C#, C++. Каждый из этих языков отличают свои особенности.

Java — строго типизированный язык с автоматической сборкой мусора, применяется для серверных и корпоративных приложений. Java имеет широкую поддержку, как с точки зрения всевозможных справочных ресурсов, так и с точки зрения реализованных прикладных библиотек с открытым исходным кодом. К недостаткам языка можно отнести в некоторой степени избыточную по современным меркам “многословность”. Возможно, этот недостаток проистекает из строгой типизации. С другой стороны, строгая типизация выступает преимуществом в той сфере, в которой как раз и применяется Java наиболее интенсивно — для написания ответственных компонент распределённых приложений. Это позволяет отсеять многие ошибки ещё на этапе компиляции программы.

Набирающий популярность Python имеет большое число почитателей среди специалистов по анализу данных. С одной стороны, этот язык позволяет разработчику использовать традиционные подходы и конструкции, например, объектно-ориентированное программирование и классы. С другой стороны, в языке имеются чрезвычайно гибкие структуры, которые характерны для скриптовых языков. Динамическая типизация избавляет программиста от необходимости при объявлении каждый

раз указывать тип переменных и в общем облегчает процедуру разработки. Также на языке реализованы многие удобные библиотеки обработки и анализа данных.

Относительно C# можно сказать, что в целом концептуально он не сильно отличается от Java. Однако, отдельно следует отметить следующую особенность: C# разработан корпорацией Microsoft, поэтому о выполнении кода C# на различных платформах (Linux, Mac) приходится размышлять только в теоретическом плане. На практике не существует официальных сред выполнения C# кода для операционных систем за исключением Windows.

С помощью C++ можно решить, как правило, любую задачу, однако это не означает что он одинаково удобен и для системного программирования и для написания небольшой программы с графическим интерфейсом. В C++ отсутствует автоматическая сборка мусора, выделение и освобождение памяти осуществляется в ручном режиме. Это обстоятельство может создавать большие неудобства. Если программист не будет аккуратно следить за выделяемой памятью, произойдёт “утечка” памяти и программа может завершиться аварийным выходом. Это лишь один аспект из многих, который следует учитывать при программировании на C++. Таким образом, при использовании C++ снижается скорость разработки, повышается трудоёмкость отладки и от программиста требуется наличие определённых навыков.

По результатам проведённого анализа было принято решение сделать выбор в пользу Python. Многие возможности, обеспечиваемые языком Python, такие как лексические замыкания, значения параметров по умолчанию, неявная динамическая типизация и автоматическая сборка мусора, ускоряют разработку и позволяют сконцентрироваться непосредственно на реализуемом алгоритме. В то же время существует множество функциональных и хорошо документированных программных библиотек для Python. Наиболее известные из них — NumPy для работы с многомерными массивами, Pandas для преобразования табличных данных и SciPy для научных вычислений.

Для выполнения программы, написанной на Python происходит трансляция каждой инструкции в промежуточную форму — байт-код. Байт-код выполняется в виртуальной машине Python. Виртуальные машины написаны для различных операционных систем, тем самым достигается независимость разработанного Python кода от используемой платформы. При этом несколько страдает производительность, но этот вопрос

нельзя однозначно закрыть в пользу языков, компилируемых в машинный код, ведь виртуальная машина может оптимизировать как выделение памяти, так и выполнение различных инструкций.

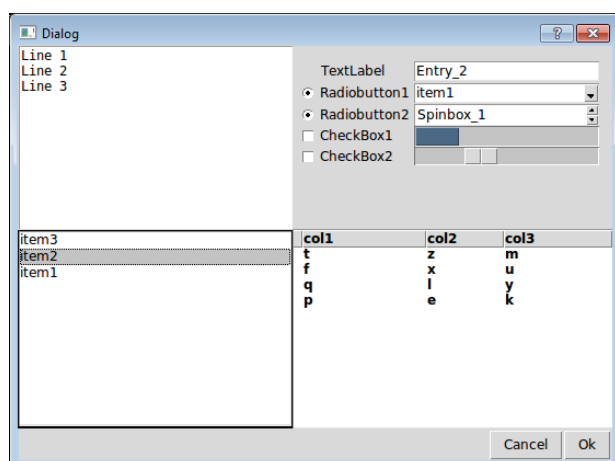
4.2.2 Библиотека графического интерфейса

Разработка программ с графическим пользовательским интерфейсом сильно упрощается при использовании готовых библиотек. Как правило, такие библиотеки позволяют создавать оконные приложения, на которых размещаются типовые элементы интерфейса: кнопки, переключатели, надписи и т.п. Практически все библиотеки предоставляют некоторый достаточный набор таких элементов, привычных пользователю современных операционных систем, некоторые библиотеки удобны для разработки собственных элементов управления. Поэтому при выборе библиотеки графического интерфейса будем руководствоваться в основном такими критериями, как удобство разработки, доступность справочной информации, кроссплатформенность. Для Python наиболее распространены следующие библиотеки: Tkinter, wxPython, PyQt. Примеры окон приложений, отрисованных каждой из указанных библиотек в операционной системе Windows 7 приведены для сравнения на рисунке 15. Как показал анализ с учётом рекомендаций на тематических ресурсах, наиболее сбалансированным решением представляется PyQt.

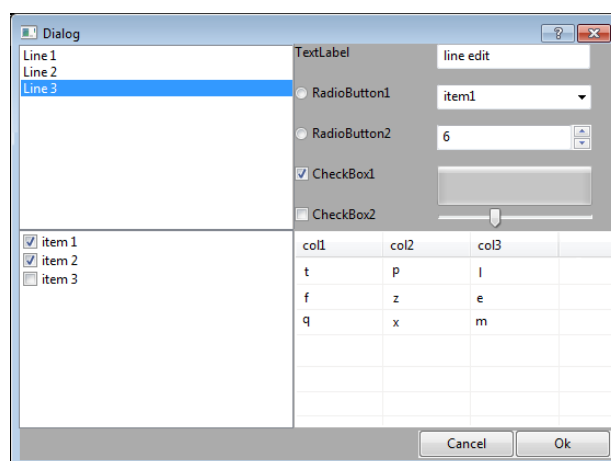
Среди недостатков Tkinter часто выделяют сложность отладки. Также распространено мнение, что эта библиотека не в полной мере соответствует представлениям о современном интерфейсе. С другой стороны Tkinter входит в стандартную поставку Python, не требует установки дополнительных модулей, достаточно прост, гибок и функционален.

wxPython — это библиотека, которую приходится устанавливать отдельно. Опытные разработчики отмечают относительно слабую проработку документации, сложность поиска справочной информации. В то же время wxPython страдает от некоторых проблем, связанных с переносимостью программы для работы под различными операционными системами. К положительным сторонам wxPython относят богатый состав типовых элементов и лучшее графическое оформление по сравнению с Tkinter.

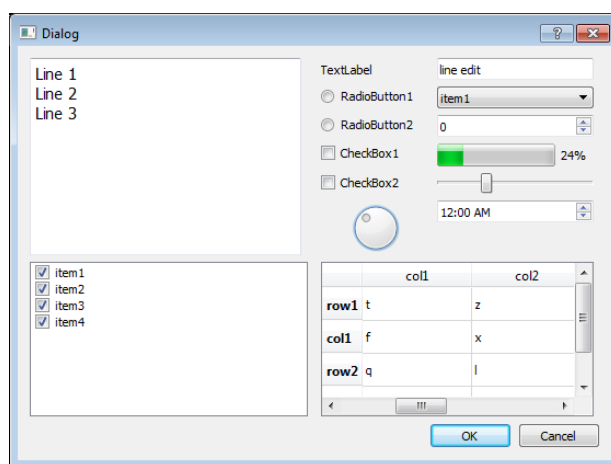
Широкую известность получила кроссплатформенная библиотека Qt, которая написана на C++ и изначально предназначена для этого языка. История этой библио-



(a) Tkinter



(б) wxPython



(в) PyQt

Рисунок 15 – Сравнение библиотек графического интерфейса

теки начинается с 1995 года, с тех пор библиотека регулярно обновляется, при этом устраняются ошибки, добавляются усовершенствования и нововведения. Документация библиотеки составлена тщательно, а большое число опытных пользователей охотно делятся рекомендациями и эффективными подходами использования средств библиотеки. Теперь воспользоваться преимуществами библиотеки можно и из языка Python. Программная библиотека PyQt представляет собой оболочку библиотеки Qt в Python классы. Удобство применения PyQt также обеспечивается наличием CASE (Computer-Aided Software Engineering) инструментов для этой библиотеки. Например, Qt Designer позволяет компоновать пользовательский интерфейс в том виде, в котором он будет представлен пользователю без необходимости писать программный код. Основным недостатком, который отмечают пользователи — лицензионная политика разработчика биб-

лиотеки. Для свободного программного обеспечения используется лицензия GNU GPL, но для коммерческих проектов требуется приобретение платной лицензии [22].

4.2.3 Библиотеки для работы с данными

Для работы с данными применяется в целом типичный набор Python библиотек: Pandas, NumPy, SciPy, scikit-learn и Matplotlib. Первая библиотека используется для удобства преобразования и отображения табличных данных. NumPy в основном применяется непосредственно для реализации алгоритмов и матричных вычислений, например в NumPy входит функция SVD разложения. SciPy позволяет численно определять минимум заданной функции, что в частности используется для вычисления центра Минковского. Библиотека scikit-learn имеет в своём составе функцию для эффективного вычисления индекса ARI. Отображение данных в виде различного рода диаграмм (SVD, гистограммы, поля рассеивания) удобно реализовать при помощи библиотеки Matplotlib.

4.2.4 Тестирование

Тестирование — важный этап разработки ПО. Правильно организованное тестирование способствует обеспечению высокого качества программного продукта а также сокращению времени на отладку. Наибольший эффект достигается при автоматизированном тестировании, что позволяет проверять работоспособность программы после каждого изменения в коде, тем самым все возникающие ошибки сразу же обнаруживаются. Следует понимать, что написание тестов также требует определённых усилий и для сложных программ невозможно обеспечить полное покрытие тестами всего кода.

В первую очередь необходимо составлять тесты на наиболее ответственные программные компоненты, от которых зависит работоспособность системы в целом. Таким компонентом в системе INDACT является модуль кластеризации. Ошибки в графическом интерфейсе, как правило, не столь значимы и критичны, они проявляются в ограниченном виде и слабо влияют на работоспособность незатронутых функций. К тому же их относительно легко обнаружить вручную. Тестирование графического интерфейса практически реализуемо, но в данном случае не целесообразно и излишне трудоёмко.

Для тестирования модуля кластеризации применяется библиотека `pytest`. Преимущества этой библиотеки заключаются в богатом функционале, включающем в себя удобные функции для проверки утверждений, параметризацию тестов, подключаемые плагины, а также в генерации подробных отчётов и независимости от API (`pytest` не требует импорта модулей фреймворка).

Тесты основаны на сопоставлении результата работы разрабатываемого алгоритма и достоверной процедуры на заранее определённых данных. Под достоверной процедурой понимается способ кластеризации с высокой степенью доверия. Например, при фиксированных данных небольшой размерности можно провести вычисления вручную и сохранить результат в текстовый файл, который будет сверяться с результатом работы написанной программы. Второй вид достоверной процедуры — использование сторонних Python библиотек для проверки частных случаев более общих алгоритмов. По такому пути можно пойти, например, при тестировании взвешенной версии k -means при $p = \beta = 2$. Для этого надо сначала произвести аномальную кластеризацию и определить центры и число кластеров. Полученные кластеры используются как для инициализации разработанного алгоритма при $p = \beta = 2$, так и для библиотечного `sklearn.cluster.KMeans`. Таким образом, в частном случае $itwk\text{-}means_{p\beta}$ является алгоритмом k -means и по этому частному случаю можно производить первичный контроль. Третий случай достоверной процедуры представляет собой использование MATLAB кода от авторов алгоритмов. Этот код запускается через командную строку в среде MATLAB прямо во время выполнения теста. Результат возвращается путём захвата стандартного вывода. Конечно, третий способ предпочтителен, он не требует фиксации данных. Но не следует пренебрегать и первыми двумя, ведь и в коде авторов алгоритмов могут содержаться ошибки.

Приведем простой пример кода, который на самом деле используется для проверки работоспособности A-Ward.

Листинг 1 — Тестирование A-Ward

```
1 from clustering.agglomerative.a_ward import AWard
2 from tests.tools import transformation_exists
3
4
5 def test_a_ward(data_cs_k_star_res):
6     cs = data_cs_k_star_res.cs
7     k_star = data_cs_k_star_res.k_star
```

```
8     actual = data_cs_k_star_res.res
9
10    run_a_ward = AWard(cs, k_star)
11    result = run_a_ward()
12    assert transformation_exists(actual, result)
```

Первая строка импортирует тестируемый алгоритм, а вторая — функцию проверки равенства разбиений. Эта функция способна определить фактическое равенство двух разбиений, даже если кластеры пронумерованы разными способами. Тестирующая функция `test_a_ward` принимает обобщённый параметр `data_cs_k_star_res`, значение которого определяется конфигурационным файлом и автоматически вставляется фреймворком `pytest`. Передаваемый параметр содержит ссылку на обрабатываемые данные и кластерную структуру, полученную при аномальной инициализации, а также терминальное число кластеров и результат выполнения достоверной процедуры. После распаковки обобщённого параметра происходит запуск алгоритма A-Ward и сопоставление полученного результата с истинным разбиением при помощи функции `transformation_exists`. Если существует такой способ перенумеровать полученное разбиение, чтобы в точности получить истинное, тест считается пройденным, в противном случае фреймворком `pytest` автоматически формируется отчёт об ошибке при тестировании с указанием различий в рассматриваемых разбиениях.

4.2.5 Средства дистрибуции

Распространение программ, написанных на Python, отличаются некоторой спецификой. Для того чтобы программа заработала на компьютере заказчика требуется обеспечить все закономерности и установить необходимые библиотеки. Нельзя оставлять без внимания даже версии устанавливаемых библиотек, в некоторых случаях от этого зависит работает программа или нет. Всё вышесказанное относится и к самому интерпретатору Python. Такая ситуация, как правило, вызывает раздражение заказчика и отнимает ресурсы разработчиков на поддержку процесса установки. Существуют инструменты, которые фиксируют все зависимости разработанной программы и устанавливают их на компьютере пользователя в автоматическом режиме. Указанные инструменты не гарантируют безошибочного процесса установки, ведь сами библиотеки могут требовать наличие каких-либо дополнительных компонент, причём не обязательно Python.

Практически исключить возникновение указанных ошибок позволила следующая процедура формирования распространяемого пакета. Программист устанавливает у себя на рабочей машине две виртуальные операционные системы, той же версии, что и у заказчика. Установки какого либо программного обеспечения на одну из операционных систем не производится, назовём эту ОС контрольной. Вторая операционная система служит для создания распространяемого каталога бинарных файлов. В неё устанавливаются все библиотеки, необходимые для запуска INDACT. Утилита `pyinstaller` [23] также отслеживает все зависимости программы, но при этом создаётся единый исполняемый файл, который включает в себя все выявленные библиотеки в той версии, которая установлена на данный момент. Полученный файл сохраняется в контрольной, “чистой” операционной системе. Если программа в контрольной ОС функционирует исправно, сформированный каталог бинарных файлов может быть отправлен заказчику без опасений относительно требуемого стороннего программного обеспечения. Заказчику останется только запустить полученный бинарный файл и начать работу.

4.3 Структура системы

Вся программа INDACT разделена на два модуля. Первый модуль содержит функции, непосредственно реализующие различные алгоритмы кластеризации и не зависит от второго модуля, в котором находится код для отображения графического интерфейса. Такое решение позволяет разработать несколько реализаций интерфейса для написанной библиотеки кластеризации, причём необязательно графических и пользовательских. Например, можно разработать код, который будет являться MATLAB оболочкой и вызывать скажем, `dePDDP` из этого математического пакета.

На рисунке 16 показана структура модулей программы INDACT. На этой диаграмме отображаются только состав модулей и иерархия вложенности Python-пакетов, с помощью которых организованы исполняемые `.py` сценарии, при этом сами сценарии не отображаются. Два основных модуля — `ect` и `ectgui` призваны разделить реализацию алгоритмов кластеризации и графического интерфейса, как это было упомянуто выше.

Модуль `ect` разделён на три пакета, имплементирующие различные функциональные аспекты. Отдельные пакеты группируют код для тестирования и код генератора данных, соответственно `tests` и `generators`. Пакет тестирования в свою очередь раз-

бит на компоненты, относящиеся к тестированию конкретных алгоритмов, некоторые из которых для краткости на рисунке 16 скрыты. Наиболее важный пакет модуля `ect` — `clustering`, в котором сгруппированы сценарии для выполнения кластеризации. В отдельные пакеты выделены алгоритмы дивизивной и агломеративной кластеризации, при этом в пакете агломеративной кластеризации обособлены две группы сценариев, выполняющих этап аномального анализа: `ik_means` и `p_init`.

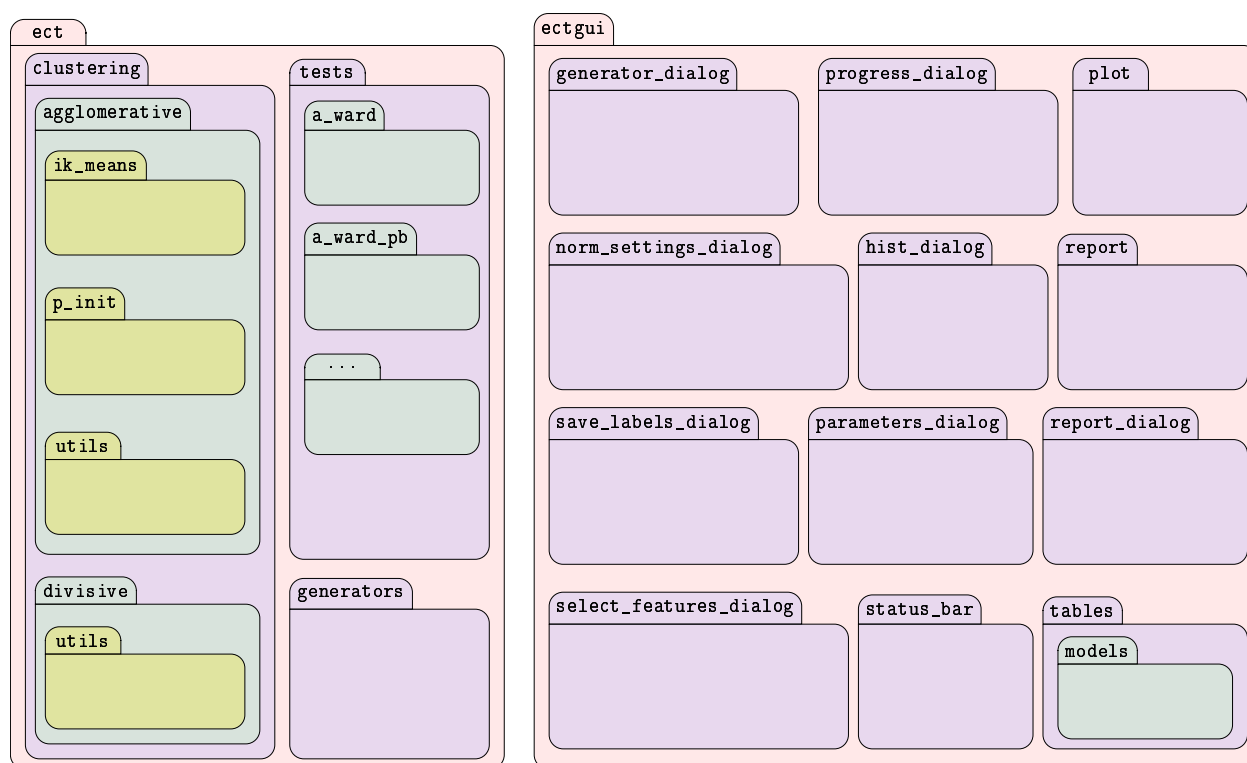
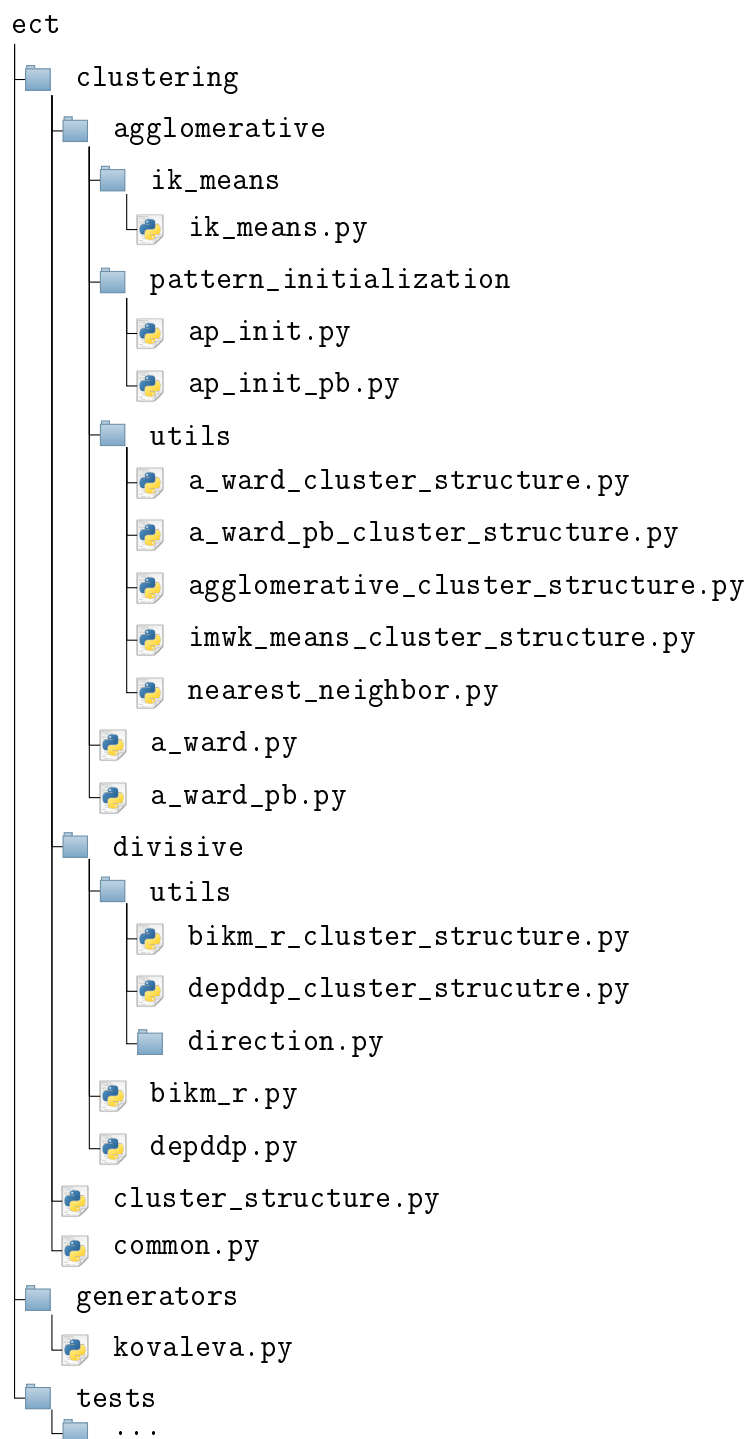


Рисунок 16 – Структура пакетов программы INDACT

Модуль `ectgui` имеет более простую, практически плоскую структуру, в которой каждый внутренний пакет соответствует коду, предназначенному для обработки определённого окна или графического элемента приложения. В программе INDACT применяются таблицы из библиотеки PyQt, основанные на моделях. Модель таблицы представляет собой Python-класс, который наследуется от `QtCore.QAbstractTableModel` и содержит определённые методы, управляющий поведением и наполнением таблицы. Эти классы и расположены в пакете `ectgui.tables.models`

Рассмотрим структуру модуля `ect` в подробностях. На рисунке 17 показан состав `ect` с указанием файлов Python сценариев. Сценарии, осуществляющие тестирование, не представляют большого интереса в данной работе, поэтому они пропущены. В насто-

Рисунок 17 – Подробная структура модуля `ect`

ящий момент в составе библиотеки кластеризации имеется один реализованный генератор данных, который был описан в разделе 2.8. Файл с кодом генератора находится в директории `generators`.

На начальных этапах программа INDACT разрабатывалась в функциональном

стиле, для каждого алгоритма была написана отдельная функция, которая принимала входные данные и необходимые аргументы, регулирующие работу алгоритма и возвращала результат в виде массива номеров кластеров, соответствующих исходным объектам. Однако, такой подход был признан неэффективным: некоторые реализуемые алгоритмы частично повторяют друг друга, что приводило или к копированию фрагментов кода, что естественно недопустимо, или к чрезмерной зависимости одних фрагментов кода от других. В первом случае при изменении какой-либо процедуры в одном алгоритме, например, при исправлении ошибки, все изменения приходится повторять и в скопированных фрагментах, что в свою очередь ведёт к возникновению новых проблем. Избыточная взаимозависимость кода замедляет процесс разработки и усложняет отладку.

С учётом вышеизложенных проблем было принято решение использовать преимущества объектного подхода. Были выделены классы для каждого из алгоритмов а также для обобщённой кластерной структуры. Под обобщённой кластерной структурой понимается совокупность групп объектов, заданная на известном множестве объектов, для которой определены некоторые процедуры, в ООП называемые также методами, и характеристики (атрибуты). Примерами процедур являются добавление и удаление новой группы объектов, а среди характеристик можно выделить, например, общее число групп. Наследование от класса обобщённой кластерной структуры позволяет использовать исходные методы и добавлять новые. Например, для реализации дивизивных алгоритмов к обобщенной кластерной структуре добавляется метод выбора наилучшего кластера. Примечательно, что благодаря принципу полиморфизма, реализация нового метода может отличаться в зависимости от алгоритма: одна реализация понадобится для dePDDP, другая — для ViKM-R. При этом код, вызывающий метод останется неизменным. Обобщенная кластерная структура описана в файле `clustering/cluster_structure.py`

Как упоминалось ранее, модуль кластеризации разбит по принципу работы. В пакете `agglomerative` содержится код, реализующий алгоритмы A-Ward и A-Ward_{pβ} вместе со сценариями для выполнения аномальной инициализации. Первый этап аномальной инициализации реализован двумя классами, которые описаны в файлах `ap_init.py` и `ap_init_pb`. Класс `APInitPB` для взвешенной аномальной инициализации насле-

дуются от `APInit`, переопределяя при этом два метода. Также объектный подход позволил избавиться от необходимости реализации двух алгоритмов *ik-means*: обобщенный алгоритм, использующий преимущества полиморфизма определен в файле `clustering/agglomerative/ik_means/ik_means.py`. Пакт `agglomerative.utils` содержит описание кластерных структур, специфичных для каждого алгоритма. Все эти описания наследуют и при необходимости переопределяют поведение обобщенной кластерной структуры.

Аналогичным образом устроен и пакет дивизивной кластеризации. В нём также алгоритмы описаны в соответствующих файлах Python-сценариев, а определение кластерных структур вынесено в пакет `divisive.utils`.

Работа модуля графического интерфейса в работе подробно рассматриваться не будет. Этот модуль в основном использует функционал библиотеки `PyQt` и, в отличие от модуля `ect`, не определяет каких-либо сложных алгоритмов, которые могли бы использоваться в других проектах.

4.4 Структура входных данных

Входными данными для рассматриваемых алгоритмов кластеризации является таблица объект-признак. В программной реализации эта таблица представлена в виде `NumPy`-массива, который хорошо подходит для хранения матрицы числовых значений и операций над ними. В то же время при анализе данных специалисты часто сталкиваются с необходимостью работать с номинальными признаками, которые принимают строковые значения. Эта проблема решается на уровне модуля графического интерфейса. Изначально предназначенная для работы с разнородными табличными данными Python библиотека `Pandas` имеет в своём составе функционал для загрузки данных из текстовых файлов, что делает её очень удобной для применения в модуле `ectgui`. В библиотеке `Pandas` для набора данных принято название `DataFrame`. Преобразование `DataFrame` в массив `NumPy` осуществляется за один вызов метода, что также очень удобно. Входные данные загружаются из текстового файла библиотекой `Pandas`, после нормализации признаков осуществляется их конвертация в массив `NumPy` и выполнение алгоритма кластеризации. Загрузка данных в библиотеке `Pandas` реализована достаточно гибко и позволяет настраивать различные параметры, но в программе `INDACT` реализована загрузка только из файлов, в которых каждая строка соответ-

стствует признаку, а значения в столбцах разделяются запятыми. Как правило, такие файлы имеют расширение `.csv`, но на самом деле расширение не накладывает строгих ограничений на содержание файла, поэтому данные могут загружаться и из файлов с другими расширениями, например, `.dat`. Дополнительные требования к названию признаков и номинальным значениям приведены в руководстве пользователя. Для наглядности ниже приведена типовая структура файла входных данных, скопированная из руководства пользователя:

Пример файла входных данных						
name,	price,	diag,	cpu,	ram,	stype,	vendor
Meizu U10 32GB,	11990.00,	5.0,	1.50,	3072,	IPS,	Meizu
ZTE Blade A510,	7011.00,	5.0,	1.00,	1024,	IPS,	ZTE
Huawei P9 Lite,	14190.00,	5.2,	2.00,	2048,	IPS,	Huawei
Meizu M5 32GB ,	12990.00,	5.2,	1.50,	3072,	IPS,	Meizu
ZTE Blade L370,	4990.00,	5.0,	1.30,	1024,	TFT,	ZTE
BQ Aquaris M5 ,	18072.00,	5.5,	1.50,	3072,	IPS,	BQ
...	,	...	,	...	,	...

4.5 Реализация алгоритмов

В данном разделе будут освещены нюансы реализации алгоритмов, приведены вычислительные схемы и фрагменты кода, где это необходимо.

4.5.1 Инициализация аномальными кластерами

Рассмотрение агломеративных алгоритмов следует начинать с аномальной инициализации, которая включает в себя два этапа: поиск аномальных кластеров и запуск *k-means* по найденным центрам кластеров. Диаграмма классов, реализующих первый этап приведена на рисунке 18. При составлении этой диаграммы стандартная нотация, предполагающая обозначение модификаторов доступа префиксными символами (+, -, #) не используется, так как язык Python не определяет область видимости с помощью модификаторов. Вместо модификаторов доступа принято руководствоваться соглашениями, по которым специальные методы выделяются двумя подчёркиваниями спереди и сзади, например `__init__`, а скрытые атрибуты отмечаются одинарным под-

чёркиванием спереди, например, `_origin`.

Класс обобщённой кластерной структуры назван `ClusterStructure`, он содержит основные методы, которые одинаково работают в большинстве реализаций. Для создания объекта `ClusterStructure` в качестве аргументов требуется передать массив данных, на которых определена структура кластеров.

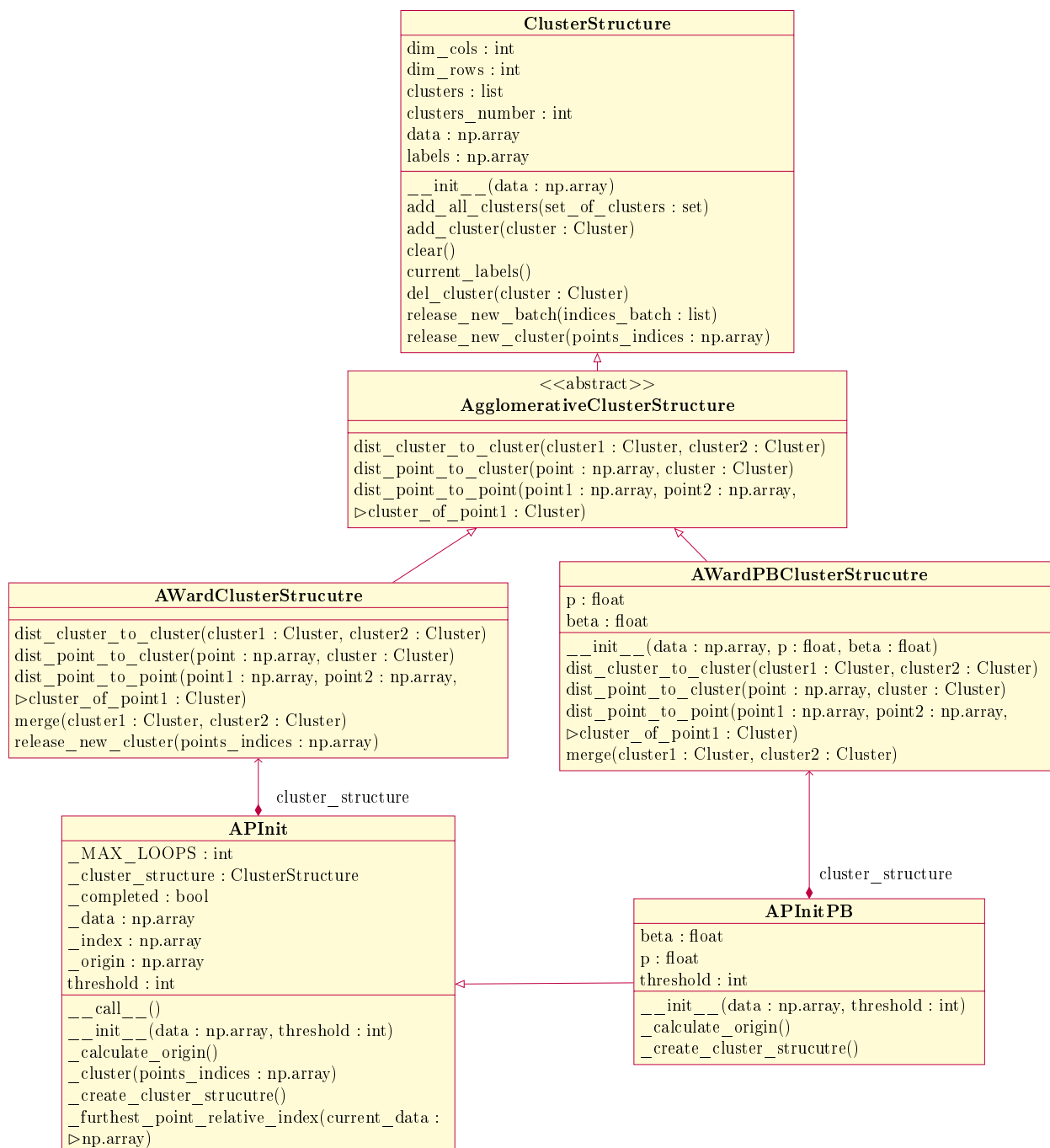


Рисунок 18 – Диаграмма классов для инициализации аномальными кластерами

Абстрактный класс `AgglomerativeClusterStructure` добавляет три абстрактных

метода, которые используются при реализации агломеративных алгоритмов. Эти методы определяют способ вычисления расстояний между двумя объектами, объектом и кластером и двумя кластерами. Классы кластерных структур `AWardClusterStructure` и `AWardPBClusterStructure` реализуют эти методы в соответствии с описанными математическими формулами для алгоритмов A-Ward и A-Ward_{pβ}. Также эти классы определяют метод слияния двух кластеров, которые не используются для аномальной инициализации, но играют важную роль в агломеративных алгоритмах.

Непосредственно алгоритм выделения аномальных групп содержится в классе `APInit`. При этом процедура вычисления глобального центра данных вынесена в отдельный метод, что позволяет переопределить его в классе `APInitPB`, наследующем `APInit`. Вычисление глобального центра данных в алгоритме аномальной инициализации для A-Ward_{pβ} является задачей минимизации метрики Минковского, в отличие от A-Ward, где достаточно просто вычислить среднее. Благодаря наследованию и введению абстрактной кластерной структуры, `APInitPB` практически не содержит кода, определяющего последовательность выполнения вычислений, а лишь адаптирует эту последовательность под более общий случай. Поэтому далее рассмотрение ограничим классом `APInit`.

При создании объекта `APInit` сохраняется ссылка на переданные в конструктор данные и целочисленный порог, определяющий минимальную численность аномальной группы. На каждой итерации работы алгоритма происходит исключение части объектов из рассматриваемых данных. Для сохранения взаимосвязи объектов, рассматриваемых на очередной итерации с исходными объектами в конструкторе инициализируется массив целочисленных индексов, длина которого равна длине исходных данных. При исключении объектов происходит одновременно и исключение соответствующих им индексов, поэтому всегда можно восстановить разбиение, полученное на исходных данных. Также в процессе инициализации вычисляется глобальный центр данных, остающийся неизменным на протяжении всего времени работы алгоритма, и создаётся пустая кластерная структура.

В Python функции можно рассматривать как объекты. Чтобы сделать вызываемый объект-функцию, программисту необходимо объявить метод `__call__()`. Как можно видеть из диаграммы классов на рисунке 18, объект класса `APInit` может быть вы-

зван как функция. Такой механизм вызова алгоритмов принят во всей разработанной библиотеке кластеризации. На рисунке 19 приведена схема вычислений, реализованная методом `APInit.__call__()`. На этой схеме опущены некоторые детали, несущественные для понимания работы. В целом синтаксис близок к синтаксису Python, но в некоторых случаях он изменён для наглядности и сокращения.

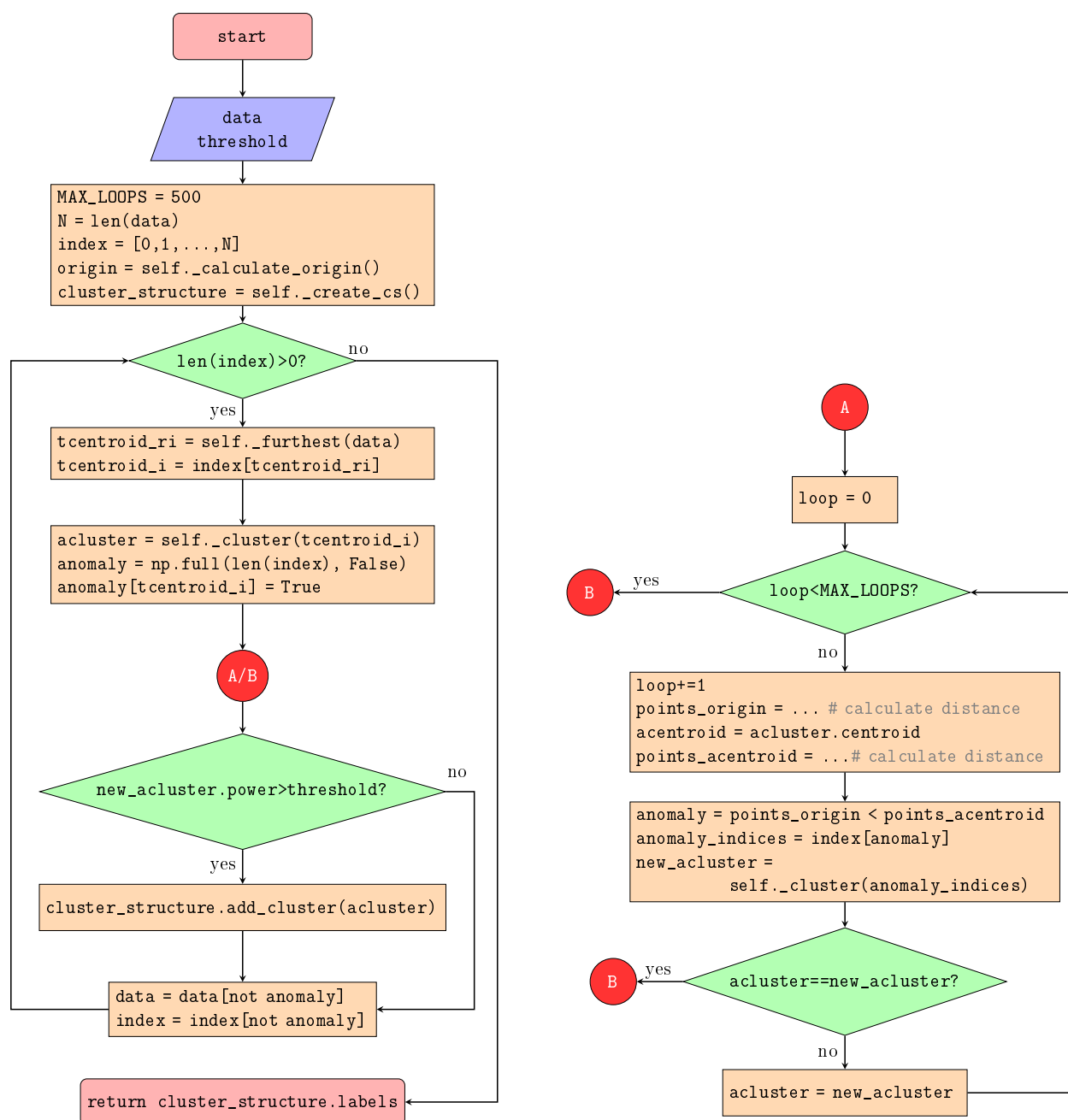


Рисунок 19 – Схема работы метода `APInit.__call__()`

Входными данными метода являются массив данных и целочисленный порог, определяющий минимальное число объектов в аномальной группе. Первый блок вычис-

лений производит инициализацию, определяя предельное число итераций внутреннего цикла, и другие переменные, используемые в работе алгоритма. Следующий блок задаёт главный цикл вычислений, итерации которого выполняются пока данные не исчерпаны. Внутри главного цикла определяется относительный индекс начального центра аномального кластера как индекс в текущих данных наиболее удалённого объекта. Для того чтобы на следующем шаге создать аномальный кластер из одного объекта требуется определить индекс этого объекта на исходных данных. Эта операция выполняется следующей строкой: `tcentroid_i = index[tcentroid_ri]`. После создания кластера, также определяется массив из логических переменных, который задаёт этот кластер на текущих данных. Количество итераций внутреннего цикла ограничено сверху переменной `MAX_LOOPS`. Этот цикл, начиная с найденного центра, формирует аномальный кластер путём попеременной минимизации квадратичного критерия. Если процедура сошлась до достижения предельного числа циклов, поток выполнения возвращается главному циклу. В главном цикле, после проверки сформированного кластера на соответствие критерию минимальной численности, он сохраняется в кластерную структуру, после чего из данных и индекса исключаются те объекты, которые входят в только что полученный аномальный кластер. После завершения работы метода, кластерная структура содержит искомое разбиение.

Аномальная инициализация с учётом весовых коэффициентов выполняется тем же кодом, но различаются реализации вызываемых методов. Класс `APInitPB` переопределяет поведение методов для создания кластерной структуры и вычисления глобального центра данных. Подставляя соответствующую кластерную структуру мы получаем другое поведение при вычислении расстояний между объектами и кластерами.

Перейдём к рассмотрению алгоритмов *ik*-means и *imwk*-means_{рв}. Оба эти алгоритма реализованы одним классом `IKMeans`, что стало возможно благодаря применению абстрактной кластерной структуры. Диаграмма классов приведена на рисунке 20. Различия в алгоритмах обеспечиваются подстановкой кластерных структур разных классов. Для реализации *ik*-means используется та же кластерная структура `AWardClusterStructure`, что и для выделения аномальных групп. Особенности учёта весовых коэффициентов вынуждают создать отдельную кластерную структуру для *imwk*-means_{рв}.

Код класса `IKMeans` достаточно прост и он полностью приведён в листинге 2. Конструктор принимает кластерную структуру как аргумент, при этом данные извлекаются из объекта кластерной структуры. Также класс имеет свойство, которое обеспечивает доступ для чтения кластерной структуры. Объекты класса `IKMeans` одновременно являются функциями, вызов которой приводит к выполнению алгоритма. Главный цикл ограничен по числу итераций значением 500. Если решение не сошлось, возвращается результат в том виде, в котором он был получен к моменту исчерпания предела в 500 итераций. Внутри главного цикла формируется словарь, который каждому кластеру ставит в соответствие объекты, которые необходимо в него включить. Создание такого словаря неизбежно потому что формирование кластеров должно происходить одновременно, так как от этого в общем случае зависят веса признаков. Внутренний цикл наполняет упомянутый словарь, перебирая данные по одному объекту. Для каждого объекта в строке 20 выбирается наиболее близкий кластер. Близость кластера к объекту определяется процедурой, заданной классом кластерной структуры, что обеспечивает повторное использование кода для обоих алгоритмов A-Ward и A-Ward_{pb}. Новые кластеры создаются также по правилам, определённым кластерной структурой (см. строку 23 листинга 2). Если созданные кластеры не равны, тем что были на предыдущей итерации, кластерная структура обновляется новыми кластерами. В противном случае процесс завершается выдачей номеров кластеров, соответствующих объектам.

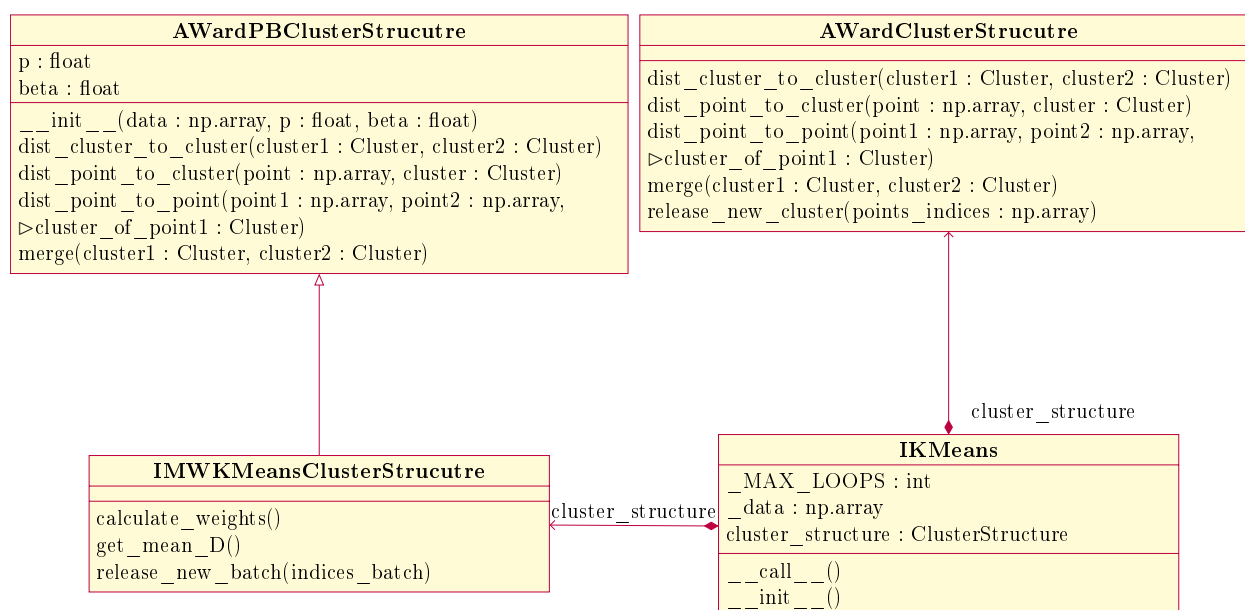


Рисунок 20 – Диаграмма классов для интеллектуальной инициализации

Листинг 2 — Код класса IKMeans

```
1 import numpy as np
2
3
4 class IKMeans:
5     _MAX_LOOPS = 500
6
7     def __init__(self, cluster_structure):
8         self._cs = cluster_structure
9         self._data = cluster_structure.data
10
11     @property
12     def cluster_structure(self):
13         return self._cs
14
15     def __call__(self):
16         for loop in range(IKMeans._MAX_LOOPS):
17             clusters = self._cs.clusters
18             cluster_points = {cluster: [] for cluster in clusters}
19             for index, point in enumerate(self._data):
20                 nearest_cluster = min(clusters, key=
21                     lambda c: self._cs.dist_point_to_cluster(point, c))
22                 cluster_points[nearest_cluster].append(index)
23             new_clusters = self._cs.release_new_batch(
24                 [np.array(x) for x in cluster_points.values()])
25             if set(new_clusters) == set(clusters): # stop condition
26                 break
27             self._cs.clear()
28             self._cs.add_all_clusters(set(new_clusters))
29         return self.cluster_structure.current_labels()
```

4.5.2 A-Ward

Для увеличения производительности A-Ward был применён алгоритм цепи ближайших соседей (nearest-neighbor chain) [24]. На каждой итерации алгоритма Ward происходит поиск двух ближайших кластеров и их объединение. Нахождение пары ближайших кластеров требует в общем случае квадратичного времени. В то же время очередная итерация не сильно изменяет кластерную структуру и большая часть расстояний, вычисленных на начальных этапах, долгое время остаётся неизменным. Чтобы использовать эту информацию для ускорения Ward можно заранее вычислить ориентированный граф ближайших соседей (ГБС) и обновлять его при обновлении кластерной структуры. Как раз по такому принципу и работает алгоритм цепи ближайших соседей. Начиная с произвольного объекта происходит обход ГБС до тех пор, пока не будут найдены два кластера, которые взаимно являются ближайшими соседями, после чего происходит их объединение. Для нового кластера вычисляются ближайшие соседи и

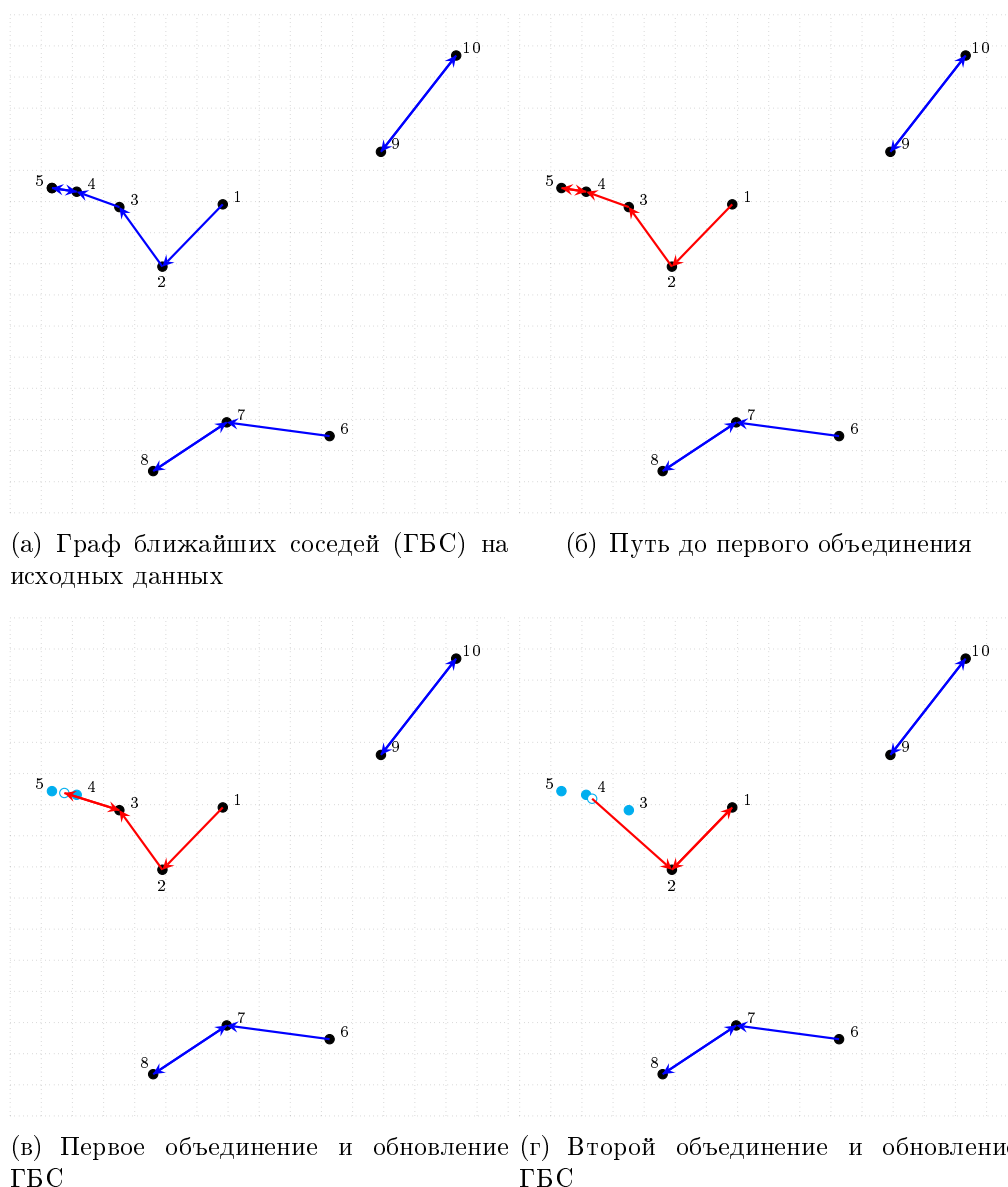


Рисунок 21 – Графическая трассировка алгоритма ближайших соседей

ГБС обновляется соответствующим образом. Недостаток алгоритма цепи ближайших соседей состоит в том, что обход происходит в порядке, не соответствующем порядку объединения кластеров алгоритмом Ward. При этом результирующая кластерная структура, всегда совпадает с результатом работы Ward.

На рисунке 21 приведена трассировка алгоритма цепи ближайших соседей. Алгоритм следует по графу ближайших соседей, хотя полностью его не строит. Для некоторых произвольных данных рисунок 21(а) изображает ГБС синими стрелками. Это сделано исключительно в иллюстративных целях. Алгоритмом выбирается некоторая произвольная начальная точка, в трассировке она обозначена как 1. Для начальной

точки вычисляется ближайший сосед (точка 2). Так как для точки 2 ближайший сосед не является точкой 1, происходит переход в точку 2. Перебор продолжается по пути в ГБС (обозначен красным цветом) до тех пор, пока не будут найдены две точки, являющиеся взаимными ближайшими соседями. Этот момент изображён на рисунке 21(б), а найденными точками являются точки 4 и 5. На рисунке 21(в) показано слияние точек 4 и 5 в бирюзовый кластер. После слияния процесс продолжается с того места, где было обнаружены точки, образующие кластер. Рисунок 21(г) изображает добавление к бирюзовому кластеру точки 3, при этом не происходит повторного вычисления и сравнения всех расстояний между объектами, а только между последними в цепи ближайших соседей.

Листинг 3 — Фрагмент кода, реализующего алгоритм цепи ближайших соседей

```
1 def __call__(self, ax):
2     stack = []
3     while self._cs.clusters_number > 1:
4         if not stack:
5             arbitrary_cluster = next(iter(self._cs.clusters))
6             stack.append(arbitrary_cluster)
7         top = stack[-1]
8         nearest, dist = self._find_nearest(top)
9         if nearest is None:
10            break
11        if nearest in stack:
12            merged = self._cs.merge(top, nearest)
13            self._merge_array.append([top, nearest, merged, dist])
14            stack.remove(top)
15            stack.remove(nearest)
16        else:
17            stack.append(nearest)
18        # sort merge_array by distance
19        self._merge_array.sort(key=lambda elem: elem[-1])
20    return self._merge_array
```

Фрагмент кода, реализующего алгоритм цепи ближайших соседей приведён в листинге 3. Этот фрагмент является методом `NearestNeighborChain` и превращает соответствующие объекты в объекты-функции. В строке 2 создаётся пустой список, который играет роль стека и хранит пройденных путь. Процесс объединения повторяется в главном цикле до тех пор пока остаются кластеры. Если стек опустел, из текущей кластерной структуры выбирается следующий кластер (см. строку 5) и он добавляется в стек. На каждой итерации из стека извлекается конечный кластер, для которого определяется ближайший кластер-сосед. Если сосед не может быть найден, выполнение алгоритма прерывается (строка 9). В том случае, если найденный сосед уже имеется

в пройденном пути, то он объединяется с конечным кластером, при этом объединяемые кластеры извлекаются из стека, иначе сосед добавляется в стек. Цикл замыкается. После завершения цикла осуществляется сортировка найденной последовательности объединений по расстоянию между кластерами (строка 19).

При заданном начальном числе кластеров n , время работы алгоритма ближайших соседей составляет $O(n^2)$ [24]. Несмотря на то, что описанная реализация требует сортировки результата, которая занимает линейно-логарифмическое время, асимптотическое время работы не изменяется: $O(n^2 + n \log(n)) = O(n^2)$.

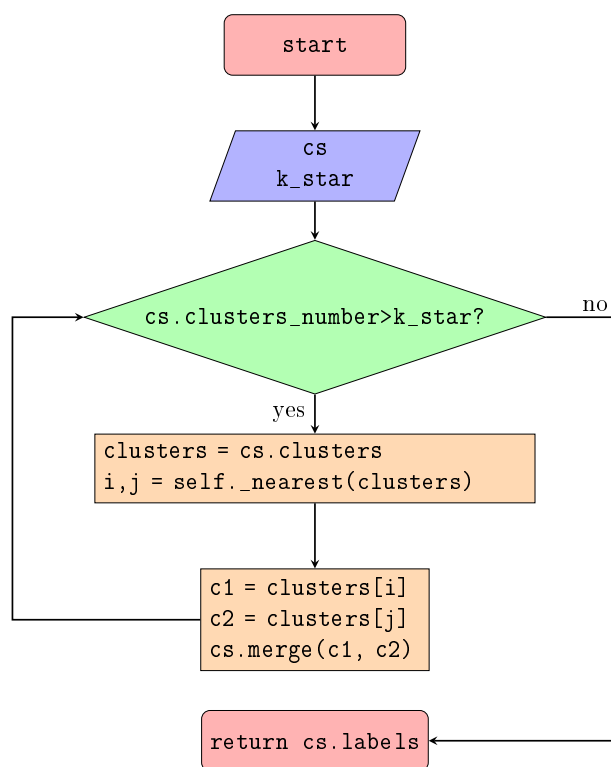
4.5.3 A-Ward_{pβ}

Реализация алгоритма A-Ward_{pβ} не использует преимущества метода цепи ближайших соседей, так как в A-Ward_{pβ} после каждого объединения происходит перераспределение весовых коэффициентов в кластерах и соответственным образом изменяются расстояния, что не может быть учтено в классической формулировке упомянутого метода. Так как все особенности вычисления расстояний в алгоритмах инкапсулируются классом кластерной структуры, то реализация A-Ward_{pβ} имеет достаточно общий вид и состоит из зацикленной последовательности поиска ближайших кластеров и их объединения, как показано на рисунке 22.

Переменная `cs` является ссылкой на кластерную структуру `AWardPBClusterStructure` (см. рисунок 18), а целочисленная переменная `k_star` определяет максимальное число кластеров, которое требуется найти. Условный блок, изображённый на рисунке 22 задаёт главный цикл и проверяет, достигнуто ли заданное число кластеров. Если, нет то выполняются внутренние операторы. Внутри цикла первый блок обработки отвечает за поиск индексов двух ближайших кластеров относительно расстояния, определяемого кластерной структурой. Второй блок получает объекты кластеров по индексу и осуществляет их слияние опять же с помощью вызова метода кластерной структуры.

4.5.4 dePDDP

4.5.5 BiKM-R

Рисунок 22 – Схема работы метода `AWardPB.__call__()`

5 Демонстрационный пример

В данном разделе описывается полный цикл работы программы на демонстрационных данных. Рассмотрены вопросы формирования данных из xml файла, их загрузка в программу, кластеризация и интерпретация результатов. Большое внимание уделено вопросу определения числа кластеров.

5.1 Формирование данных

Для демонстрации был сформирован набор данных о смартфонах, продаваемых интернет-магазином Озон, на основе xml файлов, открыто предоставляемых на веб-сайте [25]. В листинге 4 приведён фрагмент файла, из которого видна структура описания товаров. Этот файл содержит параметры моделей смартфонов, продававшихся в магазине в IV квартале 2017 года. Значения характеристик указываются между определёнными xml тегами, например, запись `<price>11990</price>` означает, что цена на данной модели составляет 11 990 руб. Аналогичным образом указаны и остальные параметры модели. В одном xml файле записана информация по нескольким сотням моделей, описание каждой модели сгруппировано внутри тега `<offer>`.

Листинг 4 — Фрагмент xml файла с описанием товара

```
1 <offers>
2   <offer id="138492682" available="true" group_id="138492684">
3     <price>11990</price>
4     <...>
5     <name>Meizu U10 32GB, Silver White</name>
6     <vendor>Meizu</vendor>
7     <param name="Вес" unit="г">330</param>
8     <...>
9     <param name="Диагональ">5</param>
10    <param name="Технология матрицы">IPS</param>
11    <param name="Процессор">MT6750 (8 ядер) 1,5 ГГц</param>
12    <param name="Встроенная память" unit="ГБ">32</param>
13    <param name="Оперативная память" unit="МБ">3072</param>
14    <...>
15  </offer>
16  <...>
17 </offers>
```

Вычленение интересующей нас информации из этого файла не составляет труда. Единственный параметр, который сохранён в неудобном формате — частота процессора. Однако, задача извлечения частоты процессора из строки также была успешно решена с применением регулярных выражений. В автоматическом режиме с помощью

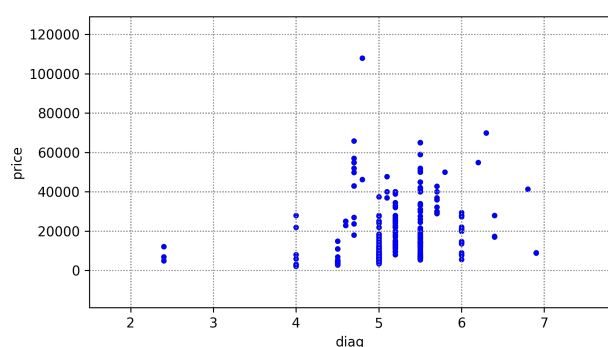
Python-скрипта xml файл был преобразован в плоскую таблицу объект-признак. При подготовке данных выяснилось, что файл содержит описание 581 модели но всего 386 из них обладают уникальными характеристиками, остальные отличаются только цветом.

5.2 Описание данных

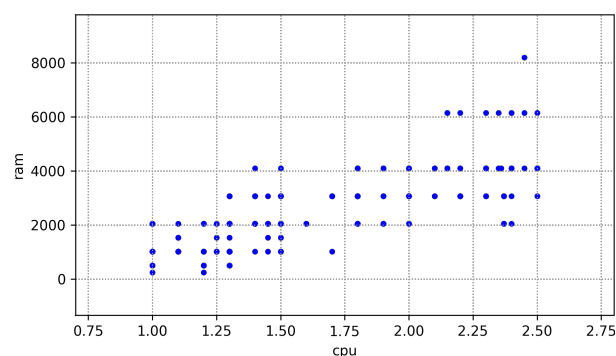
Данные, используемые в демонстрационном примере являются таблицей размерности 386×4 . Описание признаков приведено в таблице 1. Связи между некоторыми признаками показаны на графиках на рисунке 23.

Таблица 1 — Признаки данных

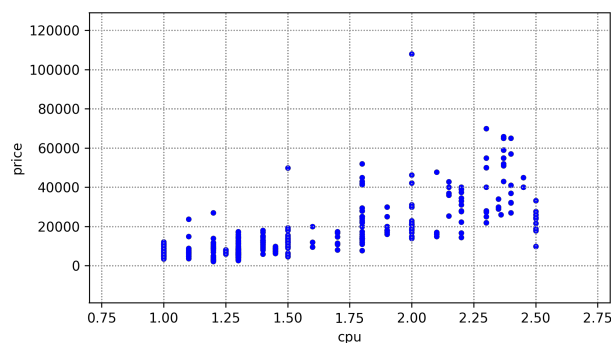
#	Название	Описание	Единица измерения
1	price	Цена данной модели смартфона в IV квартале 2017 года	руб.
2	diag	Размер диагонали экрана	дюйм
3	cpu	Частота центрального процессора (ЦП)	ГГц
4	ram	Объем оперативной памяти	Мб



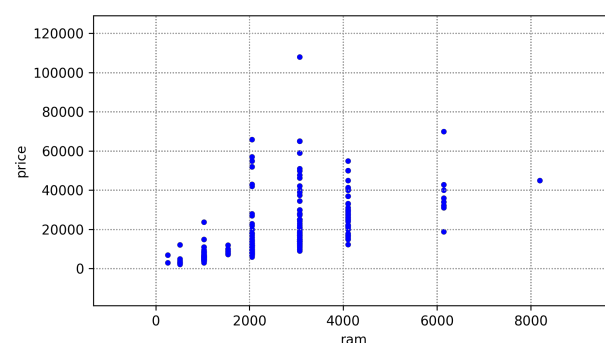
(а) Зависимость цены от диагонали экрана



(б) Зависимость объема ОЗУ от частоты ЦП



(в) Зависимость цены от частоты ЦП



(г) Зависимость цены от объема ОЗУ

Рисунок 23 – Зависимости некоторых признаков

Ниже приведён фрагмент файла исходных данных, загружаемого в программу INDACT:

name,	price,	diag,	cpu,	ram
Meizu U10 32GB Silver White,	11990.00,	5.0,	1.50,	3072
ZTE Blade A510 Grey,	7011.00,	5.0,	1.00,	1024
Huawei P9 Lite (VNS-L21) Gold,	14190.00,	5.2,	2.00,	2048
Meizu M5 32GB Black,	12990.00,	5.2,	1.50,	3072
ZTE Blade L370 Black,	4990.00,	5.0,	1.30,	1024
BQ Aquaris M5.5 16+3GB White,	18072.00,	5.5,	1.50,	3072
Samsung SM-G930F Galaxy S7 (32GB) Silver,	39990.00,	5.1,	2.30,	4096
Alcatel OT-4034D Pixi 4 (4.0) Black,	3160.00,	4.0,	1.30,	512
Sony Xperia XA Graphite Black,	13989.70,	5.0,	2.00,	2048
ZTE Blade L5 Plus Black,	6790.00,	5.0,	1.30,	1024
Meizu Pro 6 64GB Rose Gold,	25990.00,	5.2,	2.50,	4096
BQ Aquaris X5 Plus Black,	17290.00,	5.0,	1.80,	2048
Sony Xperia X Compact Mist Blue,	24989.90,	4.6,	1.80,	3072
ZTE Blade V7 Rose,	14590.00,	5.2,	1.30,	2048
Lenovo Vibe Shot (Z90A40) Red (PA1K0161RU)	16890.00,	5.0,	1.70,	3072
HTC 10 Lifestyle Topaz Gold,	27990.00,	5.2,	1.80,	3072
ZTE Nubia Z11 Max Grey,	22072.00,	6.0,	1.80,	4096
ZTE Blade L4 Pro Black,	7821.00,	5.0,	1.00,	1024
Samsung SM-G935F Galaxy S7 Edge (32GB) Black,	49990.00,	5.5,	2.30,	4096
Alcatel OT-4034D Pixi 4 (4.0) Black White,	3059.00,	4.0,	1.30,	512
Sony Xperia XA White,	13989.70,	5.0,	2.00,	2048

5.3 Нормализация

Значения признаков заданы в различных шкалах, поэтому требуется провести нормализацию данных. Как было описано ранее, функционал системы предполагает проведение нормализации с двумя параметрами: сдвиг и диапазон. Для рассматриваемых данных исходных предположений о способе нормализации нет, поэтому применена стандартная процедура с вычитанием среднего значения по столбцам и масштабированием по полуразмаху:

$$Y'_v = \frac{Y_v - \frac{1}{N} \sum_{i=1}^N y_{iv}}{\frac{1}{2}(\max_i y_{iv} - \min_i y_{iv})}, \quad v = 1, 2, 3, 4$$

5.4 Кластеризация

Для кластеризации демонстрационных данных будут опробованы 4 алгоритма: *ik-means*, *dePDDP*, *BiKM-R*, *A-Ward*. Применение на выбранных данных алгоритма *A-Ward*_{*p*β} не обосновано, так как он основан в основном предназначен для случаев с зашумлёнными данными.

Главным является вопрос о числе кластеров. Для его решения были применены

Таблица 2 — Попарный индекс ARI полученных разбиений

Число кластеров	Алгоритм	<i>ik</i> -means	dePDDP	BiKM-R	A-Ward
9	<i>ik</i> -means	1	0.28	0.36	0.67
4	dePDDP	-	1	0.25	0.46
4	BiKM-R	-	-	1	0.63
4*	A-Ward	-	-	-	1

три алгоритма, которые определяют число кластеров без участия пользователя. Алгоритмы dePDDP и BiKM-R привели к числу кластеров, равному 4, тогда как алгоритм *ik*-means при стандартном пороговом значении $\Theta = 1$ дал значительно большее число кластеров, 9. Полученный результат согласуется с теоретической частью, в которой утверждалось, что *ik*-means при стандартных настройках обнаруживает избыточное число кластеров. Исходя из этого было принято, что число кластеров равно 4, после чего использовался алгоритм A-Ward, выполняющий попарное объединение наиболее близких кластеров, полученных по методу *ik*-means, до тех пор, пока не получится 4 кластера.

Результаты попарного сравнения полученных разбиений с помощью индекса ARI сведены в таблице 2. Число кластеров для алгоритма A-Ward отмечено звёздочкой, это означает что оно фиксируется пользователем при запуске. Рисунок 24 изображает представления результатов кластеризации в координатах первых двух главных компонент. Как видно (в первую очередь по индексу ARI), разбиение по методу A-Ward наиболее похоже на остальные, поэтому далее предлагается рассматривать именно это разбиение.

5.5 Интерпретация результатов

Проведём анализ кластеров, полученных алгоритмом A-Ward. Для этого рассмотрим таблицу отдельных кластеров для нормализованных данных (далее эту таблицу). Как видно, в кластере 1 (62 объекта) доминирует значение 0.496 признака "Частота ЦП это значит, что данный кластер, в основном, выделяется этим признаком - в него входят смартфоны с максимальной частотой ЦП (2.321 ГГц, как видно из соответствующей таблицы средних значений в исходных признаках - ее тоже надо привести). Оказывается, остальные кластеры также характеризуются доминированием этого признака: 0.205 (кластер 26 82 объекта), -0.139 (кластер 3, 132 объекта) и -0.266 (кластер

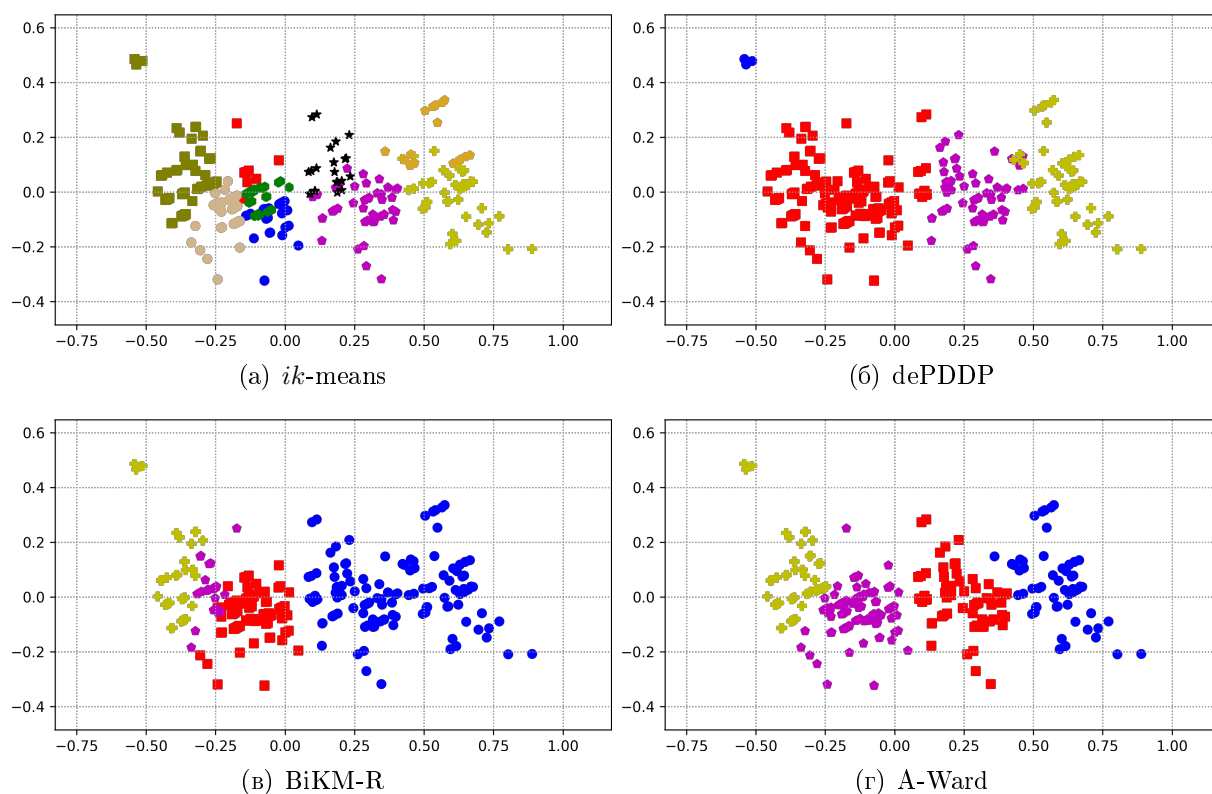


Рисунок 24 – SVD представления полученных разбиений

4, 110 объектов), причем остальные признаки ему подстать - слегка больше среднего в кластере 2, слегка меньше среднего в кластере 3 (правда, здесь Диагональ на уровне среднего) и меньше среднего, особенно по объему ОЗУ, в кластере 4. В целом, можно утверждать, что расслоение моделей произошло в основном по признаку Частота ЦП - для кластеров характерны значения 2.321, 1.885, 1.369 и 1.179, соответственно. Рис. иллюстрирует это заключение.

Можно посмотреть на разбиение по методу dePDDP как наиболее отличающееся от остальных согласно таблице (АРИ). По той же методике рассматриваем таблицу центров кластеров в нормализованных признаках (см. Таблицу...). Кластер 1 содержит всего 3 модели, которые выделяются уменьшенными значениями всх параметров, особенно Диагональ экрана, -0.614, т.е. 2.4 дюйма. Остальные три кластера более или менее соответствуют, по характеристикам их центров, кластерам 1, 2, 3 по методу A-Ward, рассмотренным выше. Обращает на себя внимание кластер 2 по методу dePDDP, объединивший более половины выборки, 253 модели - отсюда и отличия от остальных разбиений.

Таблица 3 — Фрагмент отчёта по результатам работы A-Ward

A-Ward

Clustering resulted in 4 clusters

Clusters characteristics:

I. Normalized data	price	diag	cpu	ram
Total (386 entities)				
Grand mean:	-0.000	-0.000	-0.000	0.000
Contribution, %:	77.686			
Cluster #1 (62 entities)				
Center:	0.219	0.043	0.496	0.208
Difference:	0.219	0.043	0.496	0.208
Contribution, %:	39.940			
Cluster #2 (82 entities)				
Center:	0.049	0.048	0.205	0.090
Difference:	0.049	0.048	0.205	0.090
Contribution, %:	8.585			
Cluster #3 (132 entities)				
Center:	-0.053	0.023	-0.139	-0.024
Difference:	-0.053	0.023	-0.139	-0.024
Contribution, %:	5.850			
Cluster #4 (110 entities)				
Center:	-0.096	-0.087	-0.266	-0.155
Difference:	-0.096	-0.087	-0.266	-0.155
Contribution, %:	23.311			
II. Raw data	price	diag	cpu	ram
Total (386 entities)				
Grand mean:	16284.932	5.161	1.577	2266.860
Contribution, %:	85.764			
Cluster #1 (62 entities)				
Center:	39381.852	5.353	2.321	3914.323
Difference:	23096.919	0.192	0.744	1647.462
Difference, %:	141.830	3.717	47.165	72.676
Contribution, %:	53.581			
Cluster #2 (82 entities)				
Center:	21459.248	5.376	1.885	2984.585
Difference:	5174.315	0.214	0.308	717.725
Difference, %:	31.774	4.150	19.537	31.662
Contribution, %:	21.238			
Cluster #3 (132 entities)				
Center:	10640.037	5.263	1.369	2075.152
Difference:	-5644.895	0.101	-0.209	-191.709
Difference, %:	-34.663	1.966	-13.230	-8.457
Contribution, %:	8.559			
Cluster #4 (110 entities)				
Center:	6183.326	4.772	1.179	1033.309
Difference:	-10101.606	-0.390	-0.399	-1233.551
Difference, %:	-62.030	-7.548	-25.272	-54.417
Contribution, %:	2.385			

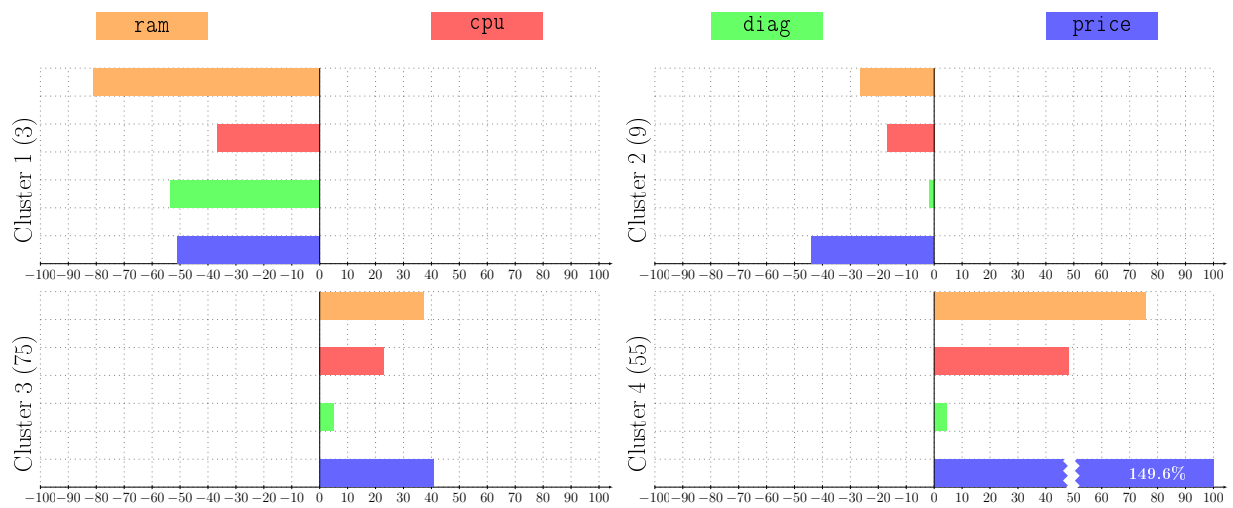
Таблица 4 — Фрагмент отчёта по результатам работы dePDDP

dePDDP

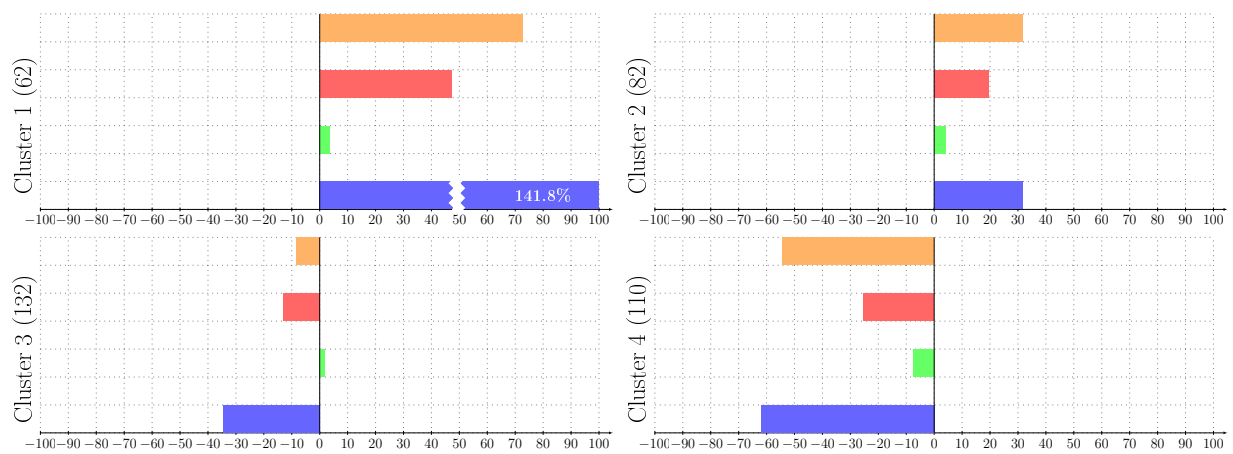
Clustering resulted in 4 clusters

Clusters characteristics:

I. Normalized data	price	diag	cpu	ram
Total (386 entities)				
Grand mean:	-0.000	-0.000	-0.000	0.000
Contribution, %:	72.391			
Cluster #1 (3 entities)				
Center:	-0.079	-0.614	-0.385	-0.232
Difference:	-0.079	-0.614	-0.385	-0.232
Contribution, %:	3.336			
Cluster #2 (253 entities)				
Center:	-0.068	-0.021	-0.177	-0.076
Difference:	-0.068	-0.021	-0.177	-0.076
Contribution, %:	20.348			
Cluster #3 (75 entities)				
Center:	0.063	0.057	0.241	0.107
Difference:	0.063	0.057	0.241	0.107
Contribution, %:	10.970			
Cluster #4 (55 entities)				
Center:	0.231	0.051	0.508	0.216
Difference:	0.231	0.051	0.508	0.216
Contribution, %:	37.738			
II. Raw data	price	diag	cpu	ram
Total (386 entities)				
Grand mean:	16284.932	5.161	1.577	2266.860
Contribution, %:	84.873			
Cluster #1 (3 entities)				
Center:	7990.000	2.400	1.000	426.667
Difference:	-8294.932	-2.761	-0.577	-1840.193
Difference, %:	-50.936	-53.501	-36.598	-81.178
Contribution, %:	0.106			
Cluster #2 (253 entities)				
Center:	9118.852	5.068	1.311	1664.506
Difference:	-7166.080	-0.094	-0.266	-602.354
Difference, %:	-44.004	-1.818	-16.875	-26.572
Contribution, %:	11.995			
Cluster #3 (75 entities)				
Center:	22923.560	5.420	1.939	3112.960
Difference:	6638.628	0.259	0.362	846.100
Difference, %:	40.765	5.010	22.958	37.325
Contribution, %:	22.147			
Cluster #4 (55 entities)				
Center:	40648.678	5.391	2.339	3984.291
Difference:	24363.746	0.230	0.762	1717.431
Difference, %:	149.609	4.447	48.315	75.763
Contribution, %:	50.625			



(a) dePDDP



(b) A-Ward

Рисунок 25 – Относительное отклонение от среднего признаков в кластерах

6 Заключение

Программная система интеллектуальной кластеризации INDACT включает в себя модуль графического интерфейса и библиотеку из 5 современных алгоритмов кластер-анализа, а также средств анализа полученных результатов. С помощью графического интерфейса INDACT позволяет проводить отбор признаков, нормализацию данных, первичный анализ с помощью полей рассеивания и гистограмм, а так же запускать выполнение реализованных алгоритмов кластеризации. На основе полученных результатов кластеризации пользователь может сгенерировать подробный отчёт для дальнейшего анализа и формулирования выводов. Библиотека алгоритмов содержит три двивзивных метода: *ik*-means, dePDDP и BiKM-R и два агломеративных: A-Ward и $A-Ward_{p\beta}$. Для каждого из алгоритмов описаны особенности и рекомендуемая область применения.

Система INDACT разработана с учётом современных тенденций в области разработки программного обеспечения и с применением передовых технических приёмов и инструментов. Код программы написан на высокоуровневом популярном языке программирования Python и опирается на принципы объектного подхода для увеличения степени повторного использования вычислительных процедур. Разработка пользовательского интерфейса произведена в графическом редакторе, что позволило ускорить процесс создания системы.

Разработанная система позволит облегчить применение реализованных алгоритмов для специалистов, не обладающих углублёнными знаниями в кластер-анализе.

Усовершенствование системы возможно в разных направлениях, из которых следует отметить следующие три: развитие функционала и добавление новых возможностей; испытания программы в реальных условиях с целью выявить пользовательские предпочтения по проведению и использованию результатов кластер-анализа; доработка автоматических тестов для увеличения покрытия и включения в покрытие графического интерфейса.

Список литературы

- [1] Миркин Б. Г. Введение в анализ данных. М.: Юрайт, 2015.
- [2] B. Mirkin. Clustering: A Data Recovery Approach. Computer Science and Data Analysis. London, UK: CRC Press, 2012.
- [3] Ball G.H. H. A clustering technique for summarizing multivariate data // Behavioral Science. 1967. no. 12. P. 153–155.
- [4] Joe H. W. Hierarchical Grouping to Optimize an Objective Function // Journal of American Statistical Association. 1963.
- [5] de Amorim R.C. Makarenkov V. M. A-Ward _{$p\beta$} : Effective hierarchical clustering using the Minkowski metric and a fast k-means initialisation // Information Sciences. 2016. Vol. 370–371. P. 343–354.
- [6] Kovaleva E.V. M. Bisecting K-Means and 1D Projection Divisive Clustering: A Unified Framework and Experimental Comparison // Journal of Classification. 2015. no. 10. P. 414–444.
- [7] Chiang M.M.-T. M. Intelligent choice of the number of clusters in k-means clustering: an experimental study with different cluster spreads // Journal of Classification. 2010. Vol. 27. P. 3–40.
- [8] Mirkin B. T. Capturing the right number of clusters with K-Means using the complementary criterion and affinity propagation // Journal of Classification. 2017.
- [9] de Amorim R.C. Shestakov A. M. M. The Minkowski central partition as a pointer to a suitable distance exponent and consensus partitioning // Pattern Recognition. 2017. Vol. 67. P. 62–72.
- [10] Boley D. Principal Direction Divisive Partitioning // Data Mining and Knowledge Discovery. 1998. P. 325–344.
- [11] Mirkin B. Core Concepts in Data Analysis: Summarization, Correlation, Visualization. 2010.

- [12] Tasoulis S.K. Tasoulis D.K. P. Enhancing Principal Direction Divisive Clustering // Pattern Recognition. 2010. no. 43. P. 3391–3411.
- [13] Steinbach M. Karypis G. K. A Comparison of Document Clustering Techniques // KDD Workshop on Text Mining. Vol. 400, no. 1. P. 525–526.
- [14] Milligan G.W. Clustering Validation: Results and Implications for Applied Analyses // Clustering and Classification. 1996. P. 341–375.
- [15] Steinley D. B. Initializing K-Means Batch Clustering: A Critical Evaluation of Several Techniques // Journal of Classification. 2007. no. 24. P. 99–121.
- [16] P. R. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis // Journal of Computational and Applied Mathematics. 1987. Vol. 20. P. 53–65.
- [17] Hubert L. A. Comparing partitions // Journal of Classification. 1985. Vol. 2. P. 193–218.
- [18] Highsmith J.A. Agile Software Development Ecosystems. Addison-Wesley Professional, 2002.
- [19] Beck K. Test-Driven Development by Example. Addison Wesley, 2003.
- [20] Hunter R. Kerr J.M. Inside RAD: How to Build a Fully Functional System in 90 Days or Less. McGraw-Hill, 1993.
- [21] The RedMonk Programming Language Rankings: January 2018. (дата обращения: 17.05.2018). [Электронный ресурс] — <http://redmonk.com/sogady/2018/03/07/language-rankings-1-18>.
- [22] What is PyQt? (дата обращения: 17.05.2018). [Электронный ресурс] — <https://www.riverbankcomputing.com/software/pyqt/intro>.
- [23] PyInstaller. (дата обращения: 17.05.2018). [Электронный ресурс] — <http://www.pyinstaller.org/>.
- [24] Murtagh F. A survey of recent advances in hierarchical clustering algorithms // The Computer Journal. 1983. № 26 (4). С. 354–359. DOI: 10.1093/comjnl/26.4.354.

-
- [25] Каталог XML - Ozon. (дата обращения: 09.10.2017). [Электронный ресурс] — http://www.ozon.ru/context/partner_xml/.