МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка системы оптимизации расположения товаров в торговой точке на основе интеллектуального анализа данных о покупках»

Обучающийся В.А. Дьяченко
(Инициалы Фамилия) (личная подпись)

Руководитель к.т.н., В.С. Климов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант И.Ю. Усатова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2022

Аннотация

Темой бакалаврской работы является «Разработка системы оптимизации расположения товаров в торговой точке на основе интеллектуального анализа данных о покупках».

Выпускная квалификационная работа состоит из введения, трех глав, заключения, 3 таблиц, 4 формул, 24 рисунков и списка литературы из 25 источников, включая зарубежные, общим объемом в 44 страницы.

Ключевым вопросом бакалаврской работы является создание удобного и понятного инструмента для анализа большого объема данных в ретейле, который поможет улучшить тактику развития покупательского опыта в рознице и значительно увеличит объем продаж путем использования полученных данных для улучшенной планировки торгового зала.

Целью бакалаврской работы является разработка системы оптимизации расположения товаров на основе интеллектуального анализа данных о покупках для улучшения покупательского опыта.

Объектом исследования бакалаврской работы являются методы и алгоритмы интеллектуального анализа данных.

Предметом исследования бакалаврской работы является применение алгоритмов интеллектуального анализа данных для изучения данных о покупках.

Выпускная квалификационная работа может быть разделена на следующие логически взаимосвязанные части: обзор предметной области, а также описание методов интеллектуального анализа данных; подробный разбор задач аффинитивного анализа данных; анализ основных алгоритмов, теоретическая схема их работы и подробное сравнение; программная реализация системы и ее тестирование.

В заключении вынесены выводы о проделанной работе.

Abstract

The title of the graduation work is «Development of a system for optimizing the arrangement of goods at the point of sale based on data mining using shopping data».

This graduation work consists of an introduction, three chapters, conclusion, 3 tables, 4 formulas, 24 figures and a list of references of 25 sources, including foreign, a total amount of 44 pages.

The key issue of the thesis is the creation of a comfortable and comprehensible tool for analyzing large amounts of data in retail, that will help to improve the tactics of customer experience development in retail and significantly increase sales by using the obtained data to enhance the layout of the sales area.

The aim of the work is to develop a system for optimizing the arrangement of goods based on intelligent analysis of shopping data to improve the customer experience.

The object of the graduation work is methods and algorithms of data mining.

The subject of the graduation work is the application of data mining analysis algorithms in order to study purchase data.

The graduation work may be divided into several logically connected parts which are: an overview of the subject area, as well as the description of data mining techniques; a comprehensive look at the goals of affinity data analysis; analysis of the major algorithms, a theoretical scheme of their work and a detailed comparison; software implementation of the system and its testing.

It can be concluded that this work is relevant not only for the field of retail, but can also be used for the analysis of the customer experience of any area of trade.

Содержание

Введение 5

1 Применение технологий интеллектуального анализа данных для решения задач розничной торговли 7

1.1 Обзор предметной области 7

1.2 Обзор существующих подходов в интеллектуальном анализе данных .. 9

2 Математическая модель аффинитивного анализа 13

2.1 Общие ассоциативные правила 13

2.2 Классификация существующих алгоритмов поиска ассоциативных правил 17

2.2.1 Алгоритм Apriori 18

2.2.2 Анализ Apriori-подобных алгоритмов (FP-Growth, Eclat) 22

2.3 Сравнительный анализ алгоритмов поиска частых наборов 26

3 Разработка программного обеспечения для реализации алгоритма 30

3.1 Выбор фреймворка для реализации интерфейса программы 30

3.2 Программная реализация 33

3.3 Тестирование программы 37

Заключение 41

Список используемой литературы и используемых источников 42

Введение

На сегодняшний день довольно большое количество информации, используемой в таких различных областях как как розничная торговля, банковский сектор, медицина и так далее, хранится в больших базах данных, но не используется полноценно. Именно поэтому процесс извлечения полезной информации является очень важным.

Особенно актуально это для анализа рыночной корзины, известного также как поиск ассоциативных правил или аффинитивный анализ. Он помогает маркетологу понять поведение покупателей, например, какие продукты покупаются вместе. Существуют различные методы и алгоритмы, позволяющие провести этот анализ.

Исходя из практики анализа рыночной корзины, наиболее эффективными средствами анализа данных являются методы и алгоритмы интеллектуального анализа данных.

Актуальность данной работы состоит в создании удобного и понятного инструмента для анализа большого объема данных в ретейле, который поможет улучшить тактику развития покупательского опыта в рознице.

Таким образом, цель исследования – разработка системы оптимизации расположения товаров на основе интеллектуального анализа данных о покупках для улучшения покупательского опыта.

Поставленная цель решается путем решения следующих задач:

– исследование существующих методов анализа покупательской корзины;

– изучение математической модели аффинитивного анализа данных;

– программная реализация системы оптимизации расположения товаров на основе интеллектуального анализа данных.

В первой главе рассматриваются анализ предметной области, основные подходы в проведении интеллектуального анализа данных, а также структура анализа рыночной корзины.

Во второй главе будут подробно рассмотрены общие принципы аффинитивного анализа, основанные на нем алгоритмы, а также проведена их сравнительная характеристика.

В третьей главе описывается программная реализация системы, выбор фреймворков и ее непосредственное тестирование.

В заключении подводятся итоговые результаты исследования, проделанной работы и их соответствие поставленным задачам.

1 Применение технологий интеллектуального анализа данных для решения задач розничной торговли

1.1 Обзор предметной области

На сегодняшний день одной из наиболее важных проблем в сфере розничной торговли является мерчандайзинг. Он предназначен для определения набора выставляемых товаров и способов их выкладки. Другими словами мерчандайзинг – это часть маркетингового процесса, определяющая, как именно товар будет продавятся в торговой точке. Его деятельность направлена на стимуляцию продаж в рознице, посредством построения результативного контакта между действующим покупателем и конечным товаром в торговом зале [3].

Основной целью мерчандайзинга, как и целью любого магазина, являются продажи, а именно направленность на представление товара таким образом, чтобы эффективность продажи продукта повышалась без помощи консультанта в торговом зале. В подавляющем большинстве случаев покупатель не имеет намерения покупать тот или иной бренд, вне зависимости от влияния рекламы или маркетинга, решение о совершении покупки он принимает непосредственно у витрины. Следовательно, сама витрина должна выглядит так, чтобы покупатель был заинтересован купить представленный на ней товар.

Классификация основных подходов в сфере мерчандайзинга представлена на рисунке 1.

Рисунок 1 – Основные подходы в сфере мерчандайзинга

Визуальный мерчандайзинг опирается на зрительное восприятие, посредством оформления витрин на основе концепций дизайна, технический берет за основу подходы маркетинга. Приоритетным среди них является перекрёстный подход, известный также как Smart Merchandising или же Интеллектуальный Мерчандайзинг.

Интеллектуальный мерчандайзинг – это использование инновационных технологий и тактик, основанных на анализе данных розничной или иной торговли, для стратегического размещения и упорядочивания продуктов. Данные, которые можно собрать с помощью инструментов интеллектуального мерчандайзинга, при помощи обнаружение корреляционных связей среди огромного количества записей о бизнес-транзакциях, также полезны для принятия бизнес-решений по пополнению запасов, проведению товарных кампаний и улучшению покупательского опыта [8].

Поиск частых наборов элементов в анализируемых транзакциях позволяет обнаружить ассоциации и корреляции между элементами в больших транзакционных или реляционных наборах данных. Решить эту задачу помогает интеллектуальный анализ данных.

1.2 Обзор существующих подходов в интеллектуальном анализе данных

Существует множество подходов для проведения интеллектуального анализа данных. Для более глубоко понимания данной области следует для начала рассмотреть уже имеющиеся шаблоны или модели поиска закономерностей. В основном выделяют три основных группы: паттерны и правила, добычу данных и приложения. Общая схема представлена на рисунке 2. Некоторые исследования, однако, разделяют эти группы: например, различным приложениям может потребоваться исследование по разным шаблонам, что, естественно, приводит к разработке новых методологий поиска [16].

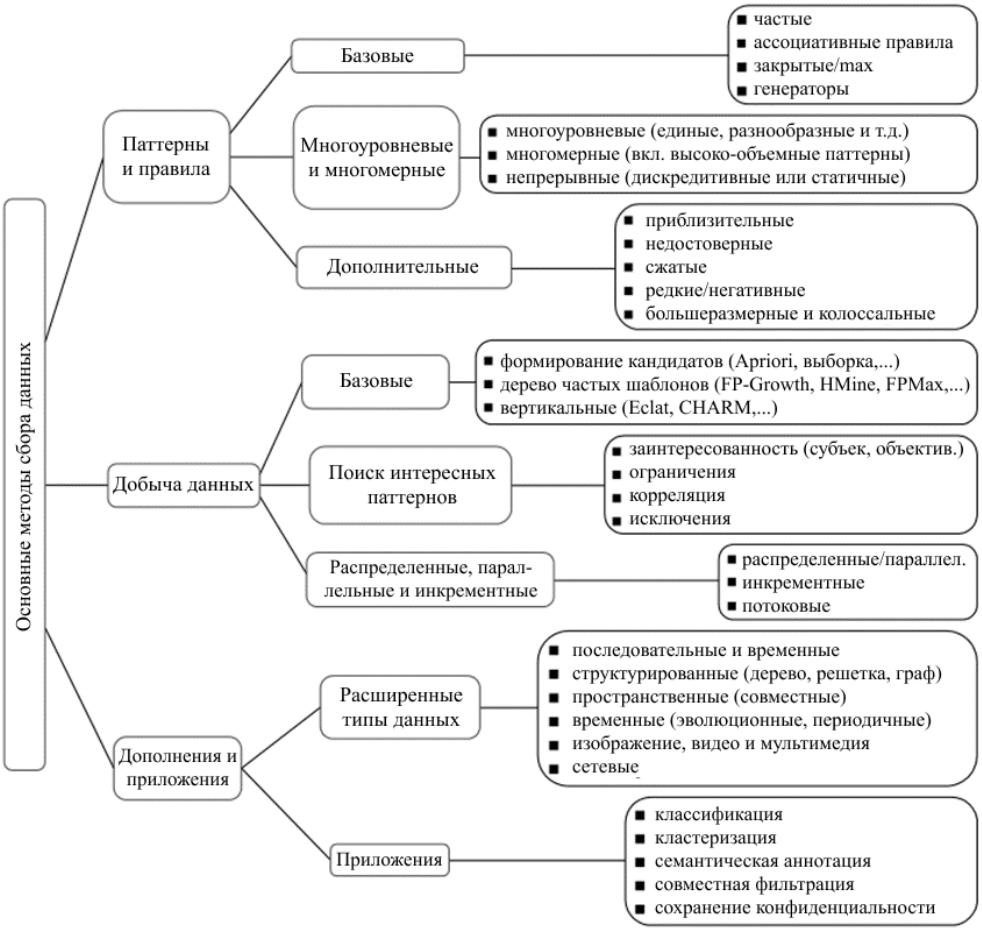

Рисунок 2 – Общая схема шаблонов поиска закономерностей

Исходя из разнообразия паттернов, поиск закономерностей можно классифицировать по следующим критериям:

– базовые закономерности: обычно часто встречающийся элемент может иметь несколько альтернативных форм, включая простую частую, закрытую или max-паттерн. Элемент p является max-паттерном, если не существует ни одной частой формы p. Часто встречающиеся паттерны также могут быть отображены в ассоциативных или других видах правил, основанных на интересующих критериях;

– на основе уровней абстракции, задействованных в паттерне: модели или ассоциативные правила могут содержать в себе элементы находящиеся на высоком, низком или нескольких уровнях абстракции;

– в зависимости от количества измерений, задействованных в правиле или паттерне: если элементы или атрибуты в ассоциативном правиле или паттерне ссылаются только на одно измерение, то это правило/паттерн ассоциации является одномерным. В противном случае это уже будет правило/паттерн многомерной ассоциации;

– основанные на типах значений, обрабатываемых в паттерне или правиле: если правило включает ассоциации между наличием или отсутствием элементов, оно является правилом булевой ассоциации, если правило описывает ассоциации между количественными элементами или атрибутами, то оно является правилом количественной ассоциации;

– на основе ограничений или критериев, используемых для поиска селективных шаблонов: обнаруживаемые шаблоны или правила могут быть основаны на ограничениях (то есть удовлетворять набору заданных пользователем ограничений), приблизительными, сжатыми, близкими к совпадениям (то есть теми, которые подсчитывают количество опор близких или почти совпадающих наборов элементов) и так далее.

Чаще всего интеллектуальный анализ данных рассматривается как одна большая область Data Mining. Data Mining – это процесс сортировки больших массивов данных для выявления закономерностей и взаимосвязей в

раннее неизвестных и практически полезных знаниях, которые необходимы для принятия решений в различных бизнес-проблемах. Методы и инструменты Data Mining позволяют предприятиям прогнозировать будущие тенденции и принимать более обоснованные бизнес-решения.

Data Mining является ключевой частью общей аналитики данных и одной из основных дисциплин в науке о данных, которая использует передовые методы анализа для поиска полезной информации в этих наборах. На более детальном уровне Data Mining является одним из этапов процесса обнаружения знаний в базах данных Knowledge Discovery in Databases (KDD) – методологии науки о данных для сбора, обработки и анализа данных. Иногда понятия «Data Mining» и «KDD» называют взаимозаменяемыми, но чаще всего их рассматривают как разные вещи [2].

В свою очередь Data mining разделяют на пять больших групп, а именно:

– классификация: при классификации сначала изучаются характеристики вновь представленного объекта, а затем его относят к заранее определенному классу, например, классифицируют кредитных заявителей как с низким, средним или высоким риском;

– ассоциация: основной целью ассоциации является установление взаимосвязи между элементами, существующими на рынке. Типичными примерами моделирования ассоциаций являются анализ рыночной корзины и программы перекрестных продаж;

– прогнозирование: здесь происходит прогнозирование некоторых неизвестных или отсутствующих значений атрибутов на основе иной информации. Например: Прогнозирование стоимости продаж на следующую неделю на основе имеющихся данных;

– кластеризация: в этом случае Data Mining организует данные в значимые подгруппы (кластеры) таким образом, что точки внутри группы похожи друг на друга и максимально отличаются от точек в других группах. Это неконтролируемая классификация;

– анализ выбросов: в данной технике Data Mining используется для выявления и объяснения исключений. Например, в случае кредитных заявителей, выброс может быть некоторой транзакцией, которая происходит необычно [4].

Наиболее опциальной техникой в нашем случае будет является поиск ассоциаций, так как именно с его помощью происходит наиболее детальный и точный анализ транзакций покупателей. Типичным примером поиска наборов частых элементов является анализ рыночной корзины.

Анализ рыночной корзины (Market Basket Analysis) или аффинитивный анализ – это метод математического моделирования, основанный на теории, что если приобретается определенная группа товаров, то с большой вероятностью приобретётся и другая группа. Данный тезис впервые был сформулирован Р. Агравалом в 1993 году. Этот процесс анализирует покупательские привычки клиентов путем поиска ассоциаций между различными товарами, которые покупатели кладут в свои «корзины». Обнаружение этих ассоциаций может помочь ритейлерам разработать маркетинговые стратегии, основываясь на знании о том, какие именно товары покупатели часто приобретают вместе. Например, люди, покупающие муку и сахар, также склонны купить яйца, потому что большая часть из них планирует испечь пирог [19].

Розничная компания может использовать эту информацию для изменения планировки магазина так, чтобы товары, которые чаще всего встречаются вместе в транзакции, располагались близко друг к другу для улучшения покупательского опыта. Главным инструментом в обнаружении данных взаимосвязей являются ассоциативные правила.

2 Математическая модель аффинитивного анализа

2.1 Общие ассоциативные правила

Ассоциативные правила – это одна из очень важных концепций машинного обучения, используемая в анализе рыночной корзины. Инвестирование времени и ресурсов в продуманное размещение товаров, подобное этому, не только сокращает время покупателя на выбор товара, но и напоминает ему о том, какие соответствующие товары он может быть заинтересован купить, тем самым помогая магазинам в процессе перекрестных продаж. Ассоциативные правила помогают выявить все подобные связи между товарами из огромных баз данных [22]. Важно отметить следующее: правила не извлекают предпочтения индивидуума, а скорее находят взаимосвязи между множеством элементов каждой отдельной транзакции.

Если говорить более подробно, то правила не связывают различные транзакции пользователей во времени для выявления взаимосвязей. Список товаров с уникальными идентификаторами транзакций (от всех пользователей) изучается как одна группа [9]. Это помогает при размещении товаров в отделах.

Ассоциативное правило состоит из условия {A} и следствия {B}, оба из которых представляют собой список элементов. Теперь рассмотрим его структуру более подробно.

Пусть $I = \{I_1 , I_2 ,..., I_m\}$ – набор элементов или событий, а T – это набор транзакций базы данных, где каждая транзакция t - это непустой набор элементов, такой, что $t \subseteq I$. Каждая транзакция, связанная с идентификатором, называется TID (Transaction ID set) [11]. Считается, что транзакция T содержит A, если $A \subseteq T$. Правило ассоциации - это импликация вида $A \Rightarrow B$, где $A \subset I, B \subset I, A = \emptyset, B = \emptyset, и\ A \cap B = \varphi$. Пример формирования таблицы транзакций, представлен на рисунке 3.

T	Транзакции
t_1	$\{i_3, i_9, i_{10}, i_{13}\}$
t_2	$\{i_1, i_3, i_{10}\}$
t_3	$\{i_2, i_9\}$
…	…
t_m	$\{i_3, i_9, \ldots, i_{10}, i_{13}\}$

Рисунок 3 – Таблица транзакций

Правило $A \Rightarrow B$ выполняется на множестве транзакций T с поддержкой s (support), где s – процент транзакций в T, которые содержат $A \cup B$ (т.е. объединение множеств A и B или, скажем, и A, и B). Иными словами поддержка дает представление о том, насколько часто набор элементов встречается во всех транзакциях. Это принимается за вероятность $T(A \Rightarrow B)$.

$$supp(A \Rightarrow B) = T(B|A) = \frac{|\{t \in T; (\mathrm{A} \cup \mathrm{B}) \subseteq t\}|}{|T|} \quad (1)$$

Другой важной метрикой является достоверность (confidence). Достоверность определяет вероятность появления следствия {A} в корзине, учитывая, в ней уже имеется условие {B}. Правило $A \Rightarrow B$ имеет достоверность c в наборе транзакций T, где c – процент транзакций в T, содержащих A, которые также содержат B. Это принимается за условную вероятность $T(B|A)$.

$$conf(A \Rightarrow B) = T(B|A) = \frac{supp(\mathrm{A} \Rightarrow \mathrm{B})}{supp(A)} \quad (2)$$

Правила, которые удовлетворяют как минимальному порогу поддержки (min_sup), так и минимальному порогу достоверности (min_conf), называются сильными [16].

Набор элементов, содержащий k элементов, называется k-элементный набор. Набор {мука, молоко} является набором из двух элементов. Частота появления набора элементов – это количество транзакций, содержащих этот набор. Если поддержка набора элементов I удовлетворяет предварительно

определенному минимальному порогу поддержки min_sup, то I является частым набором элементов. Множество частых наборов из k элементов принято обозначать L_k.

Рассмотрим пример поиска ассоциативных правил, используя данные из таблицы 1.

Таблица 1 – Пример набора товаров в транзакции

1	персики, сок, конфеты, сливы
2	персики, сок, конфеты
3	персики, сок
4	чай, сок, конфеты, сливы, кофе
5	чай, сок, конфеты, кофе
6	чай, сок, кофе
7	персики, груши
8	персики, чай, груши
9	сок, кофе, конфеты
10	сливы, кофе, чай

Товар «кофе» содержится в 5 транзакциях из 10, следовательно, используя формулу (1) поддержка будет равна:

$$supp(\text{кофе}) = \frac{5}{10} = 0{,}5$$

Для того, чтобы выяснить вероятность, что вместе с «кофе» покупатель возьмет еще и «чай», нужно рассчитать их совместную поддержку. Кофе и чай встречаются вместе в 4 транзакциях, следовательно:

$$supp(\text{кофе} \Rightarrow \text{чай}) = \frac{4}{10} = 0{,}4$$

Поддержка товара «кофе» равна 0,5. Таким образом, опираясь на формулу (2) достоверность правила кофе ⇒ чай будет равна:

$$conf(\text{кофе} \Rightarrow \text{чай}) = \frac{0{,}4}{0{,}5} = 0{,}8$$

Следовательно, можно с большой вероятностью утверждать, что те, кто купил кофе, скорее всего возьмут и чай.

Но правил поддержки и достоверности недостаточно для отсеивания не интересующих нас ассоциативных правил. Чтобы устранить этот недостаток, можно использовать меру корреляции. Корреляционное правило измеряется не только поддержкой и достоверностью, но и корреляцией между наборами элементов A и B. Таким правилом является лифт (lift).

Лифт – это простая мера корреляции, которая задается следующим образом. Появление набора элементов A не зависит от появления набора предметов B, если $T(A \Rightarrow B) = T(A)P(B)$; в противном случае, наборы элементов A и B являются зависимыми и коррелируют как события. Лифт между появлением A и B можно рассчитать следующим образом:

$$lift(A \Rightarrow B) = \frac{\mathrm{T(A \cup B)}}{\mathrm{T(A)T(B)}} = \frac{supp(A \Rightarrow B)}{\mathrm{supp(A)supp(B)}} \quad (3)$$

Если полученное значение уравнения меньше 1, то возникновение A отрицательно коррелирует с появлением B, что означает, что появление одного из них, скорее всего, приводит к отсутствию другого. Если полученное значение больше 1, то A и B положительно коррелируют, то есть возникновение одного из них подразумевает возникновение другого. Если полученное значение равно 1, то A и B независимы и корреляция между ними отсутствует [23].

Обратимся к таблице 1 и найдем значение лифта для правила кофе ⇒ чай:

$$lift(\text{кофе}, \text{чай}) = \frac{0{,}8}{0{,}5} = 1{,}6$$

Так как значение больше 1, можно утверждать, что кофе и чай чаще покупают вместе, чем по отдельности.

Еще одной мерой корреляции является показатель левередж (leverage). Он рассчитывает разницу между наблюдаемой частотой появления A и B

вместе и частотой, которая ожидалась бы, если бы A и B были независимы. Значение левереджа, равное 0, указывает на независимость [6].

$$levr(A \Rightarrow B) = T(B|A) = supp(A \Rightarrow B) - supp(A)supp(B) \quad (4)$$

Таким образом значение для товаров «кофе» и «чай» будет равно:

$$levr(\text{кофе}, \text{чай}) = 0{,}4 - 0{,}25 = 0{,}15$$

Метрики лифт и левередж помогают выявить ассоциативные правила с еще большей точностью, однако не всегда являются обязательными и особо надежными.

2.2 Классификация существующих алгоритмов поиска ассоциативных правил

Существует множество алгоритмов для поиска ассоциативных правил. Они работают на статических данных и находят хорошие ассоциативные правила на основе таких различных метрик как поддержка, достоверность, лифт и так далее, которые были рассмотрены раннее. В действительности, поиск частых наборов товаров - это процедура, которая помогает найти те наборы продуктов, которые чаще всего покупают вместе [5].

Алгоритмы для поиска частых элементов можно разделить на три категории, которые представлены на рисунке 4.

Рисунок 4 – Классификация алгоритмов

Алгоритмы на основе объединения (joined-based) расширяют список элементов в более крупный набор элементов до минимального порогового значения поддержки, которое определяет пользователь. Алгоритмы на основе дерева (tree-based) используют лексикографическое дерево, которое позволяет собирать наборы элементов различными способами, например, по глубине порядке, а алгоритмы наращивания паттернов (pattern growth) создают наборы элементов в зависимости от выявленных в данный момент частых шаблонов и расширяют их. Первые две группы алгоритмов имеют схожий подход к решению поставленной задачи, так как при проходе базы данных они формируют наборы элементов длиной k, основываясь на наборах длинной k-1 [15]. Таким образом, вышеперечисленные алгоритмы можно разделить на две основные группы:

– подход, основанный на генерации и тестировании набора кандидатов (candidate set generation-and-test);

– подход, основанный на наращивании паттернов (pattern-growth).

Основополагающим алгоритмом среди них является алгоритм Apriori.

2.2.1 Алгоритм Apriori

Алгоритм Apriori был предложен Р. Агравалом и Р. Срикантом в 1994 году для поиска наборов частых элементов для булевых ассоциативных правил [AS94b]. В его основе лежит подход candidate set generation-and-test. Apriori использует итерационный подход, известный как поиск по уровням, где k-наборы используются для поиска (k+1)-наборов [21].

Первым шагом находится набор единичных частых наборов элементов путем сканирования базы данных, затем происходит группировка тех элементов, которые удовлетворяют минимальной поддержке. Полученное множество обозначается L_1. Далее L_1 используется для нахождения L_2 , набора двойных частых элементов, который используется для нахождения L_3 и так далее, пока не будет найдено больше ни одного частого k-набора. Для нахождения каждого L_k требуется провести один полный обход базы данных [7]. Псевдокод алгоритма представлен на рисунке 5.

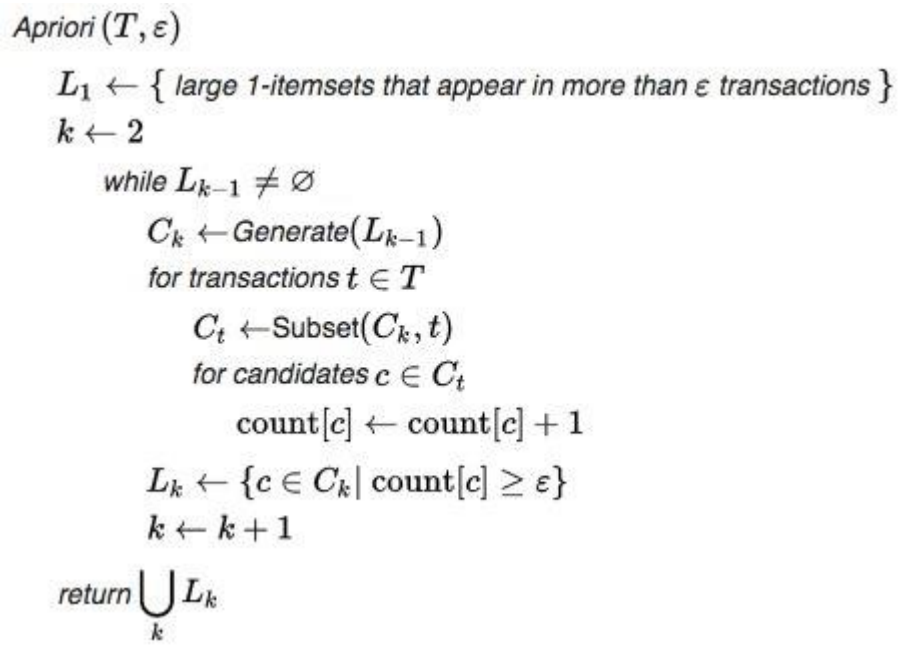

Рисунок 5 – Псевдокод алгоритма Apriori

Чтобы найти L_k, набор кандидатов k-элементов создается путем соединения L_{k-1} с самим собой. Это множество элементов обозначается C_k. В первой итерации набор элементов A непосредственно составляет первый набор элементов-кандидатов C_1 . Предположим, что $A=\{a_1,a_2,...,a_m\}$, тогда $C_1=\{\{a_1\},\{a_2\},...,\{a_m\}\}$. На k-й итерации, набор элементов-кандидатов C_k этой итерации формируется в соответствии с набором частых элементов L_{k-1}, найденным на последней итерации.

По определению, если набор элементов I не удовлетворяет минимальному порогу поддержки, min_sup, то I не является частым, то есть $T(I) < min\_sup$. Если к набору элементов I добавить элемент A, то полученный набор элементов (то есть $I \cup A$) не может встречаться чаще, чем I. Поэтому $I \cup A$ также не является частым, то есть $T(I \cup A) < min\_sup$, следовательно, все непустые подмножества частого набора элементов также должны быть частыми [12]. Другими словами, если какое-либо множество длины k не является частым в базе данных, ни одно из его супер множеств длины k+1 не может быть частым.

Это свойство принадлежит к особой категории свойств, называемых антимонотонностью – если множество не может пройти тест, то все его подмножества также не пройдут тот же тест. Данное свойство помогает значительно сократить количество проходов для поиска. Его иллюстрация представлена на рисунке 6.

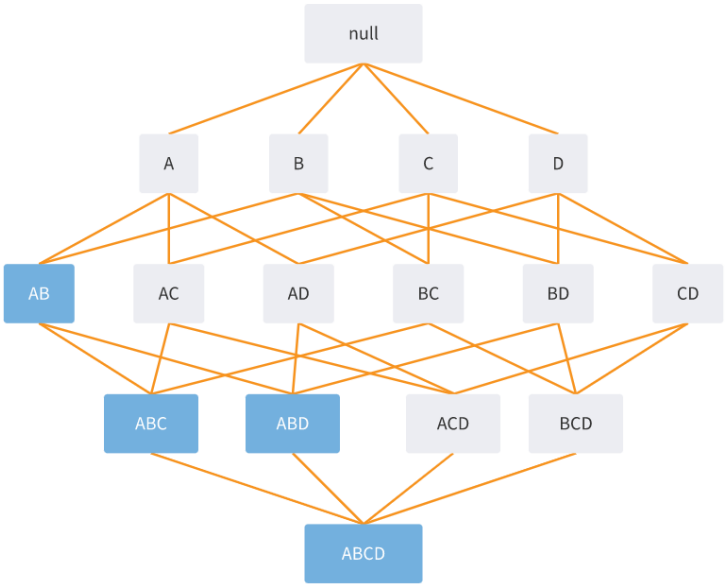

Рисунок 6 – Свойство анти-монотонности

Рассмотрим пример работы алгоритма, изображенные на рисунке 7. В качестве min_sup возьмем значение равное 3. Первым этапом происходит генерация кандидатов, состоящих из одного элемента. Далее рассчитывается их поддержка. Если ее значение меньше значения min_sup набор отбрасывается. Оставшиеся наборы будут часто встречающимися одноэлементными наборами. Тоже самое повторяется и с двух-, трех- и так далее элементными наборами до тех пор, пока наборов удовлетворяющих min_sup не останется.

Предположим, что свойство анти-монотонности не используется. На этапе сравнения поддержки наборов с минимальным, остается только один набор, следовательно работа алгоритма завершается, так как формирование

новых наборов становится невозможным. Это наглядно продемонстрировано на области, выделенной серым цветом. Ниже этого участка представлен иной исход.

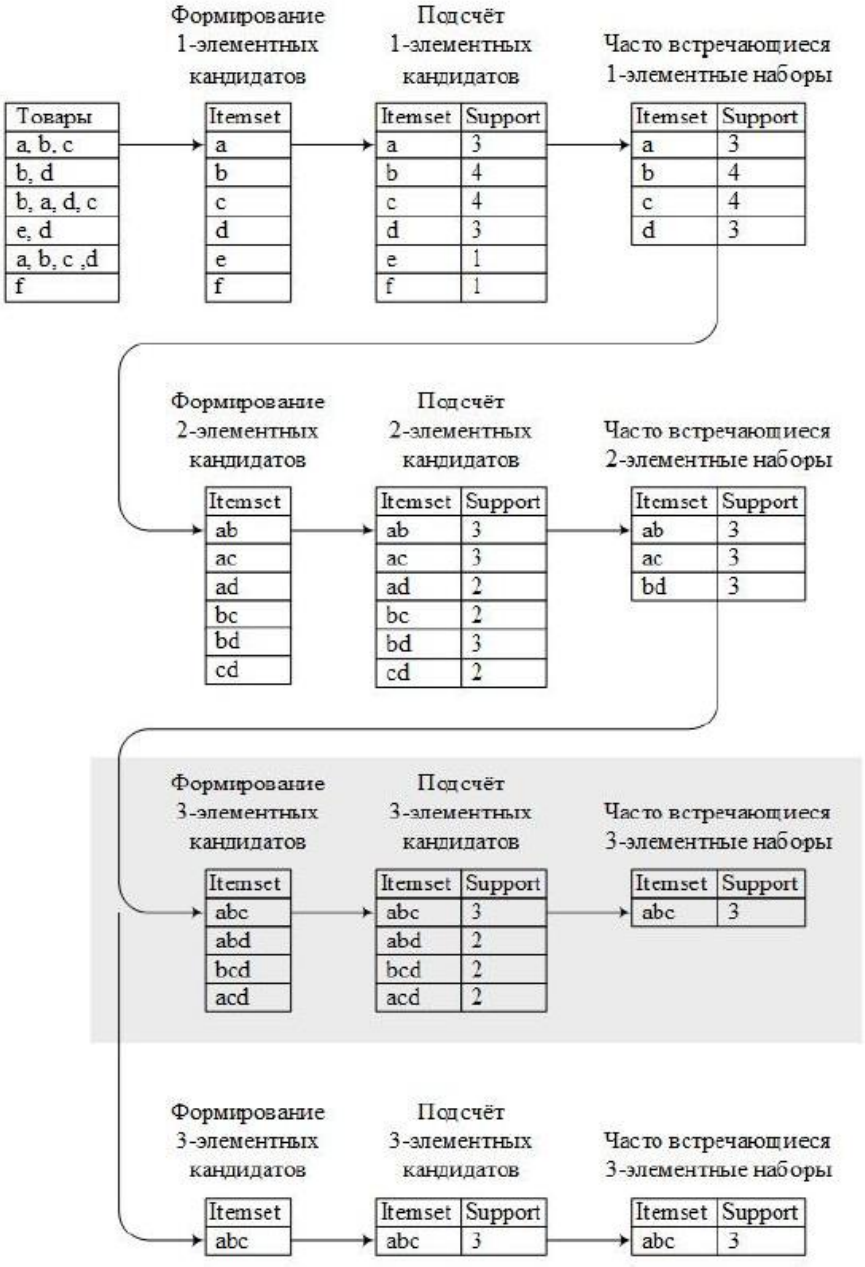

Рисунок 7 – Иллюстрация работы алгоритма Apriori

Преимущества алгоритма:

– это самый простой и понятный алгоритм среди всех алгоритмов поиска ассоциативных правил;

– получаемые правила интуитивно понятны и легко доносятся до конечного пользователя;

– он не требует маркированных данных, что дает возможность использовать его в различных ситуациях, поскольку такие данные часто наиболее доступны.

Недостатком алгоритма является время, необходимое для обработки большого количества наборов-кандидатов с большим количеством частых наборов элементов [24];

На основе данного алгоритма в последствии были созданы еще множество Apriori-подобных алгоритмов. Самыми распространенными из них стали FP-Growth и ECLAT.

2.2.2 Анализ Apriori-подобных алгоритмов (FP-Growth, Eclat)

Алгоритм FP-Growth (frequent pattern growth) является основополагающим для подхода «pattern-growth». Он сжимает базу данных в дерево частых шаблонов (FP-дерево) при этом сохраняя информацию об ассоциациях между наборами элементов. Процесс генерации FP-дерева представлен на рисунке 8. После сжатая база данных делится на набор баз данных условий (специальный тип проекции базы данных), затем с каждой из них проводится ассоциация с наиболее частым элементом [14].

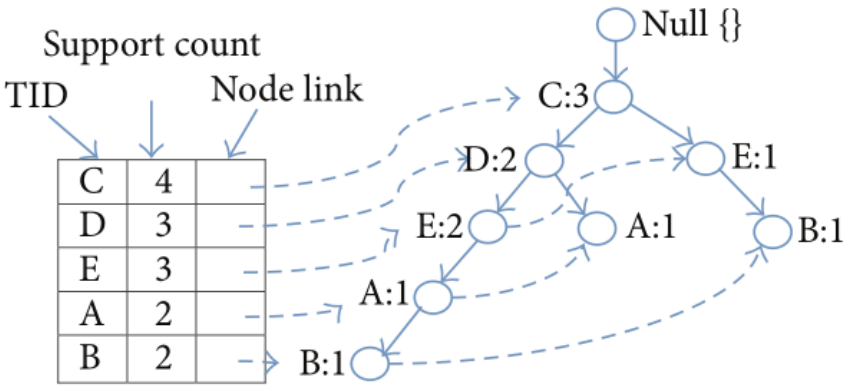

Рисунок 8 – Процесс генерации FP-дерева

Дерево частых шаблонов – это древовидная структура, которая создается на основе исходных наборов элементов базы данных. Целью FP-дерева является поиск наиболее часто встречающихся паттернов. Каждый узел FP-дерева представляет элемент множества [25]. Корневой узел представляет ноль, а нижние узлы – наборы баз данных условий.

Псевдокод алгоритма FP-Growth представлен на рисунке 9.

```
procedure FP_growth(Tree, α)
(1)   if Tree contains a single path P then
(2)     for each combination (denoted as β) of the nodes in the path P
(3)       generate pattern β ∪ α with support_count = minimum support count of nodes in β;
(4)   else for each a_i in the header of Tree {
(5)     generate pattern β = a_i ∪ α with support_count = a_i.support_count;
(6)     construct β's conditional pattern base and then β's conditional FP_tree Tree_β;
(7)     if Tree_β ≠ ∅ then
(8)       call FP_growth(Tree_β, β); }
```

Рисунок 9 – Псевдокод алгоритма FP-Growth

Преимущества:

– по сравнению с Apriori, который сканирует транзакции на каждой итерации, алгоритм FP-Growth сканирует базу данных только дважды;

– алгоритм FP-Growth превосходит алгоритм Apriori по скорости выполнения.

Недостатки:

– FP-Tree является более громоздким и сложным для построения, чем Apriori;

– если исходная база данных слишком велика, алгоритм может не поместиться в общей памяти.

Вторым по частоте использования алгоритмом, является Eclat (Equivalence Class Clustering and Bottom-Up Lattice Traversal) или Преобразование Классов Эквивалентности и Обход Решетки Снизу-Вверх. В то время как большинство алгоритмов придерживается применения общих

ассоциативных правил, Eclat не делает того же. Например, он не использует такую метрику как достоверность, которая необходима в альтернативных моделях. С другой стороны, это позволяет алгоритму быть более быстрым [17]. Иллюстрация работы алгоритма представлена на рисунке 10.

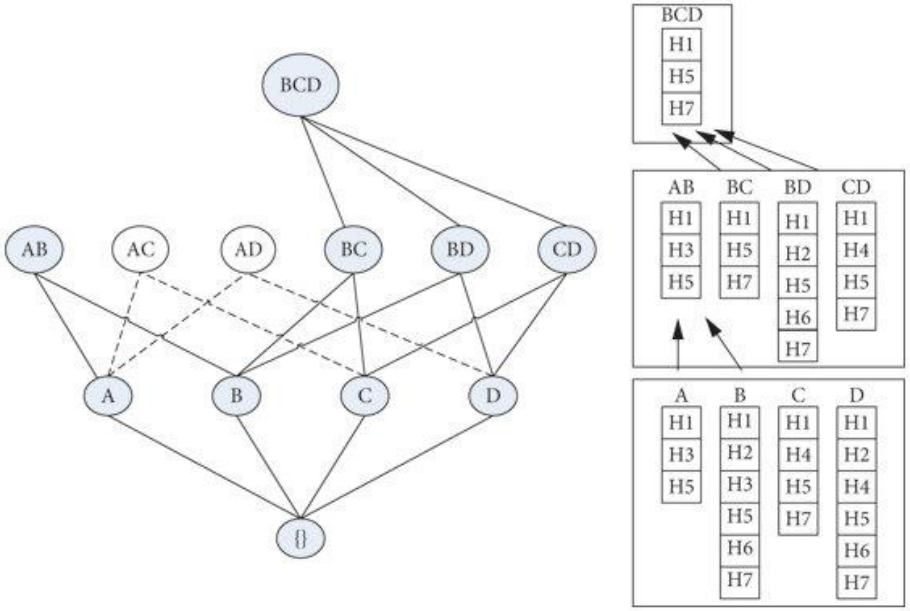

Рисунок 10 – Схема работы алгоритма Eclat

Данный алгоритм основан на использовании подхода генерации и тестировании набора кандидатов. Алгоритм находит элементы снизу, как при поиске в глубину, используя вертикальную базу данных, так как использование горизонтальной базы данных не подразумевается. Eclat сканирует базу данных только один раз, поэтому нет необходимости сканировать ее повторно. При работе он использует особую структуру данных – lattice или «решетка».

Первым шагом идет составление списка наборов идентификаторов транзакций (TID) для каждого продукта. Следующим шагом является выбор значения min_sup. Далее вычисляется набор идентификаторов транзакций для каждой пары продуктов и фильтрация результатов, которые не достигают

минимальной поддержки. Все найденные наборы элементов помещаются в «решетку», после чего происходит обход базы данных в ширину [10]. Псевдокод алгоритма представлен на рисунке 11.

```
Bottom-Up (Fk ) {
1. for all Ii Fk
2. Fk +1 = Φ;
3. for all IjЄFk, i < j
4. N = Ii ∩ Ij;
5. if N .sup ≥ min_sup then
6. Fk +1 = Fk +1UN;
7. end
8. end
9. end
10. if Fk +1 ≠ Φ then
11. Bottom-Up(Fk +1);
12. end
13. }
```

Рисунок 11 – Псевдокод алгоритма Eclat

Количество элементов хранится в массиве $F_k = \{I_1, I_2, ..., I_n\}$. в качестве входных данных. На выходе получаются наборы часто встречающихся элементов. Поиск элементов начинается снизу вверх. В начале работы алгоритма элемент F_k+1 берется как пустая база данных. Далее находится поддержка отдельных элементов и сравнивается с поддержкой всех элементов с min_sup. Затем все элементы помещаются в F_k+1. Если массив не является пустым, проход «снизу-вверх» будет применяться к F_k+1.

Преимущества:

– алгоритм Eclat занимает мало памяти по сравнению с Apriori, поскольку в нем используется метод глубинного поиска;

– алгоритм Eclat превосходит алгоритм Apriori только при условии, что набор данных не слишком велик;

– алгоритм Eclat сканирует только текущий набор данных, который сканируется в алгоритме Eclat. В отличие от Apriori, где исходный набор данных сканируется на каждом этапе.

Недостатки:

– если список tidlist слишком велик, алгоритм Eclat может исчерпать память. Для пересечения длинных наборов TID требуется больше места в памяти и времени на обработку;

– если исходная база данных имеет горизонтальный формат, появляется необходимость перестраивать ее в вертикальную форму.

2.3 Сравнительный анализ алгоритмов поиска частых наборов

Для выбора наиболее подходящего алгоритма, который бы удовлетворял условиям поставленной задачи, необходимо провести их сравнительный анализ. Можно выделить следующие критерии для сравнения:

– эвристика алгоритма;

– простота/сложность реализации;

– распределение объемов затрачиваемой памяти;

– время выполнения расчетов;

– основной подход;

– используемая структура данных.

Критерии «основной подход» и «используемая структура данных» уже были рассмотрены раннее.

Обратимся к критерию эвристика алгоритма. Данный показатель играет важную роль в общей теории алгоритмов, однако их эффективность напрямую зависит от конкретных данных, их плотности и равномерности распределения, следовательно в нашем случае его использование является нецелесообразным.

Критерий распределение объемов затрачиваемой памяти зависит от сложности структур данных, используемых в алгоритме поиска часто встречающихся наборов данных. Поэтому для сравнения будет взят конечный вид базы данных просканированных элементов.

Критерий простота реализации является довольно субъективным, так как его понятия сильно размыты. Подтверждает данный тезис тот факт, что на данный момент существует множество одноименных библиотек с уже готовыми реализациями упомянутых алгоритмов, которые и будут использоваться в дальнейшем сравнении, следовательно сложность реализации считается равной.

Для определения скорости работы алгоритмов будет использован набор данных «Groceries» расположенный в репозитории цента машинного обучения Калифорнийского Университета (UCI Machine Learning Repository) по адресу https://archive.ics.uci.edu/ml/datasets.php. Содержимое данного файла представлено на рисунке 12.

```
citrus fruit,semi-finished bread,margarine,ready soups
tropical fruit,yogurt,coffee
whole milk
pip fruit,yogurt,cream cheese ,meat spreads
other vegetables,whole milk,condensed milk,long life bakery product
whole milk,butter,yogurt,rice,abrasive cleaner
rolls/buns
other vegetables,UHT-milk,rolls/buns,bottled beer,liquor (appetizer)
pot plants
whole milk,cereals
tropical fruit,other vegetables,white bread,bottled water,chocolate
citrus fruit,tropical fruit,whole milk,butter,curd,yogurt,flour,bottled water,dishes
beef
frankfurter,rolls/buns,soda
chicken,tropical fruit
butter,sugar,fruit/vegetable juice,newspapers
fruit/vegetable juice
packaged fruit/vegetables
chocolate
specialty bar
```

Рисунок 12 – Пример содержимого файла groceries.csv

Выборка содержит в себе информации о товарах из супермаркета, на основе сформированных чеков. Информация о товаре представлена в виде

его категории. Все элементы перечислены через запятую, на каждой строке расположена отдельная транзакция. Общее количество транзакций в файле 7438.

Сравнительный анализ выполнялся при помощи готовых библиотек Apriori, Eclat и FP-Growth. В качестве основной характеристики для сравнения было взято значение минимальной поддержки min_sup с шагом 0,1. Результаты времени проверки представлены в секундах. Полная сравнительная характеристика представлена в таблицах 2-3.

Таблица 2 – Сравнительная характеристика алгоритмов поиска частых наборов данных

Наименование алгоритма	Подход	Метод поиска частых наборов	Структура данных	Сохранение результатов в памяти
Apriori	Генерация и тестирование наборов кандидатов	Поиск в ширину	Массив данных ArrayList	Преобразованная версия исходной базы данных
FP-Growth	Наращивание паттернов	Поиск в глубину	Префиксное дерево FP-Tree	Набор условных FP-деревьев для каждого элемента
Eclat	Генерация и тестирование наборов кандидатов	Поиск в ширину	Использование решетки «lattice»	Наборы TID элементов в решетке

Таблица 3 – Сравнительная характеристика времени работы алгоритмов

Наименование алгоритма	Значение минимальной поддержки		
	0,7	0,8	0,9
Apriori	88.34	79.01	75.40
FP-Growth	88.06	78.24	74.05
Eclat	85.42	76.20	71.61

По результатам сравнительного анализа можно сделать вывод, что значительного отличия во времени работы алгоритмов нет. Как было упомянуто ранее эффективность их работы зависит только от исходной выборки и используемых модификаций. Наиболее подходящим методом поиска наборов частых элементов будет являться поиск в ширину, так как он более ориентирован на решение поставленной задачи, чем метод поиска в глубину. Для наглядности представления полученных результатов в качестве основного алгоритма для реализации системы оптимизации было принято решение в пользу алгоритма Apriori, так как преобразованная версия исходной базы данных, как и работа со структурой данных ArrayList, предоставит более наглядное представление о полученных результатах, в отличие от других рассмотренных алгоритмов.

3 Разработка программного обеспечения для реализации алгоритма

3.1 Выбор фреймворка для реализации интерфейса программы

Комфортная работа пользователя с программой обеспечивается наличием в ней графического интерфейса. В этом разработчикам помогает такой инструмент как фреймфорк. Фреймворки – это программное обеспечение, которое разрабатывается и используется разработчиками для создания приложений.. Выбор фреймворка во многом зависит от типа приложения, которое будет разрабатываться. Поскольку они часто создаются, тестируются и оптимизируются несколькими опытными инженерами и программистами, программные фреймворки являются универсальными, надежными и эффективными.

Использование фреймворка для разработки приложений позволяет сосредоточиться на высокоуровневой функциональности приложения. Это происходит потому, что о любой низкоуровневой функциональности позаботится сам фреймворк. Программные фреймворки облегчают жизнь разработчикам, позволяя им взять под контроль весь процесс разработки программного обеспечения или большую его часть с единой платформы [13].

Преимущества использования программного фреймворка:

– ощутимая помощь в разработке программного кода и использование паттернов проектирования;

– можно избежать дублирования и избыточного кода;

– помогает последовательно разрабатывать код с меньшим количеством ошибок;

– облегчает работу над сложными технологиями;

– несколько сегментов кода и функциональных возможностей предварительно построены и протестированы. Это делает приложения более надежными;

– время, необходимое для разработки приложения, значительно сокращается.

Так как для разработки был выбран объектно-ориентированный язык программирования Python, рассмотрим самые используемые для него фреймворки.

Kivy является кросс-платформенным фреймфорком для разработки мобильных приложений и других мультисенсорных приложений. Следовательно, Kivy можно определить как Python-фреймворк или Python-библиотеку для разработки приложений с пользовательским интерфейсом, имеющем открытым исходным кодом. Он имеет графический движок, построенный на OpenGL с самыми современными и быстрыми конвейерами. Он предоставляет набор инструментов, состоящий из 20 виджетов, таких как кнопки, ярлыки, макеты и так далее [18].

Основным недостатком Kivy является его невозможность компиляции или доступа к сложным кодам приложений, так как для этого требуется сочетание языков Python и kV вместе с помощью специальных алгоритмов. На основе данного недостатка был разработан язык Kivy, который имеет лучший синтаксис и представление всех функциональных возможностей в программе с более точным и организованным в виде дерева форматом. Пример его использование проиллюстрирован на рисунке 13.

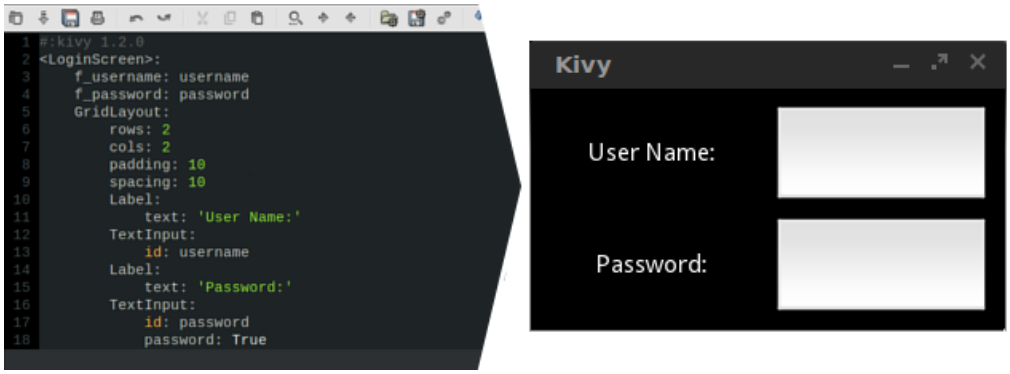

Рисунок 13 – Фреймворк Kivy

Вторым по частоте использования фреймворком является PyQt. PyQt – также является кросс-платформенным для разработки десктопных приложений со встроенными виджетами и инструментами, а также надежной поддержкой базы данных SQL. Таким образом, PyQt определяется как Python GUI фреймворк с набором привязок Python версий v2 и v3 и открытым исходным кодом.

В PyQt есть PyQtgraph, который построен на PyQt5 или PySide или NumPy, который также имеет быстрые конвейеры, так как его основу составляют библиотеки NumPy. Он предоставляет различные компоненты пользовательского интерфейса, такие как виджеты, кнопки, меню, ярлыки, таблицы и т.д., а также другие проектируемые виджеты, совместимые с PyQt, поскольку QT предоставляет все эти компоненты пользовательского интерфейса в Python [20].

Основным недостатком PyQt является отсутствие каких-либо специфических ресурсов. В PyQt, Qt не является каким-либо языком программирования, так как это инструмент. Главным преимуществом PyQt является возможность предварительного моделирования окон разрабатываемого приложение в редакторе Qt, который представлен на рисунке 14.

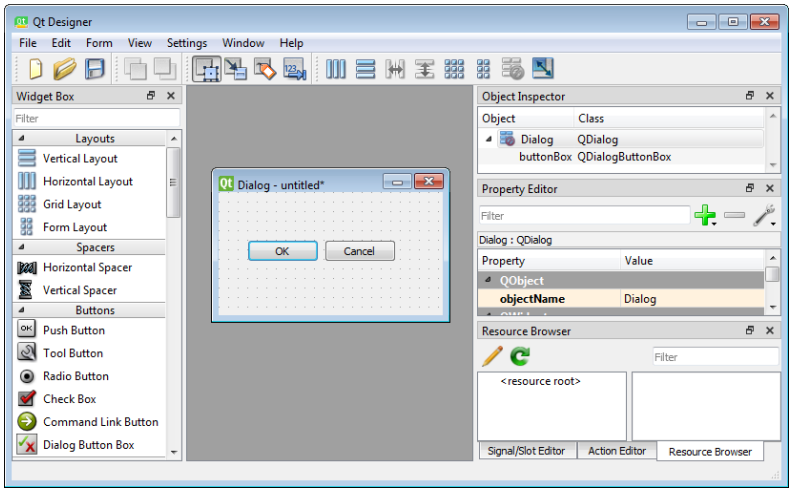

Рисунок 14 – Интерфейс редактора Qt

Так как Kivy более целенаправлен на разработку мобильных приложений, в качестве основного фреймворка при разработке был выбран PyQt.

3.2 Программная реализация

В ходе выполнения данной работы с использованием объектно-ориентированного языка программирования Python и фреймворка PyQt5 была разработана программа, реализующая алгоритм Apriori, которая находит ассоциативные правила на основе анализа множества транзакций.

Возможности программы:

– реализован интерфейс программы для удобства использования;

– реализован вызов справки, для лучшего понимания работы программы;

– по заданным пользователем значениям поддержки и достоверности производится поиск ассоциативных правил;

– вывод результата производится в отдельном окне.

Порядок работы разработанной программы представлен на рисунке 15.

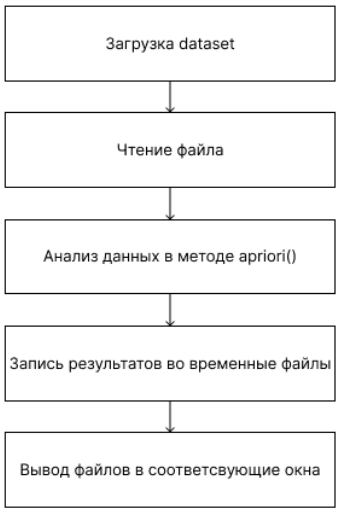

Рисунок 15 – Порядок работы программы

В ходе разработке программного кода, были использованы следующие библиотеки Python [1], представленные на рисунке 16:

– itertools является модулем, который предоставляет различные функции, работающие с итераторами для создания сложных итераторов

– numpy используется для работы с массивами. В ней также есть функции для работы в области линейной алгебры и преобразования матриц.

– matplotlib используется для визуализации данных и построения двумерных графиков.

– pandas используется для быстрого и понятного анализа данных, а также для работы с "реляционными" или "маркированными" данными.

– csv используется для возможности чтения баз данных

– apyori является простой реализацией алгоритма Apriori с помощью, предоставляемых в виде API и интерфейсов командной строки.

– sys служит для передачи информации о системе argv в QApplication

– os является методом для отображения содержимого директорий

– PyQt5 служит для отображения шаблонов интерфейса

```
import sys
import os
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import csv
from apyori import apriori
import itertools

from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtWidgets import QApplication, QWidget, QFileDialog, QTableWidget, QTableWidgetItem
```

Рисунок 16 – Импорт библиотек

Реализация алгоритма Apriori происходит в файле apriori.py, подробная структура которого представлена ниже.

Этап отсеивания элементов представлен на рисунке 17. Метод prune_dataset() сканирует количество каждого элемента в базе данных. Если элемент-кандидат не удовлетворяет минимальной поддержке, то он считается нечастым и удаляется. Данный метод выполняется для уменьшения размера наборов элементов-кандидатов.

```
def prune_dataset(olddf, len_transaction, tot_sales_percent):
    if 'tot_items' in olddf.columns:
        del(olddf['tot_items'])
    Item_count = olddf.sum().sort_values(ascending = False).reset_index()
    tot_items = sum(olddf.sum().sort_values(ascending = False))
    Item_count.rename(columns={Item_count.columns[0]:'Item_name',Item_count.columns[1]:'Item_count'}, inplace=True)
    selected_items = list(Item_count[Item_count.Tot_percent < tot_sales_percent].Item_name)
    olddf['tot_items'] = olddf[selected_items].sum(axis = 1)
    olddf = olddf[olddf.tot_items >= len_transaction]
    del(olddf['tot_items'])
    return olddf[selected_items], Item_count[Item_count.Tot_percent < tot_sales_percent]
```

Рисунок 17 – Программная реализация метода prune_dataset()

Далее программа обращается к методу apriori(), представленного на рисунке 18, который в свою очередь берет данные из методов generateLk(), generateCk() и rulegenerator().

```
def apriori():
    L, data = OneItemSets()
    flag = 1
    FreqItems = dict(L)
    while(len(L) != 0):
        fo = open(freqItemsets, "a+")
        for k, v in L.items():

            fo.write(str(k) + ' >>> ' + str(v) + '\n\n')
        fo.close()
        plotitemfreq(L)

        L, flag = generateCk(L, flag, data)
        FreqItems.update(L)
    rulegenerator(FreqItems)
```

Рисунок 18 – Программная реализация метода apriori()

Генерация множеств L_k метода generateLk() и C_k метода generateCk(), рассмотренных в главе 2 при анализе алгоритма Apriori, представлены на рисунках 19 и 20 соответственно. C_k - это супермножество L_k. Оно также является частью процесса отсеивания. Его члены могут быть или не быть частыми, но все частые k-элементы включаются в C_k. Создается путем объединения двух L_k. Если элемент в C_k принадлежит транзакции, он попадает в список C_t, затем C_t обрабатывается для формирования L для k-наборов частых элементов.

```
def generateCk(Lk_1, flag, data):
    Ck = []
    if flag == 1:
        flag = 0
        for item1 in Lk_1:
            for item2 in Lk_1:
                if item2 > item1:
                    Ck.append((item1, item2))

    else:
        for item in Lk_1:
            k = len(item)
        for item1 in Lk_1:
            for item2 in Lk_1:
                if (item1[:-1] == item2[:-1]) and (item1[-1] != item2[-1]):
                    if item1[-1] > item2[-1]:
                        Ck.append(item2 + (item1[-1],))
                    else:
                        Ck.append(item1 + (item2[-1],))
    L = generateLk(set(Ck), data)
    return L, flag
```

Рисунок 19 – Программная реализация метода generateLk()

```
def generateLk(Ck, data):
    count = {}
    for itemset in Ck:
        for transaction in data:
            if all(e in transaction for e in itemset):
                if itemset not in count:
                    count[itemset] = 1
                else:
                    count[itemset] = count[itemset] + 1
    count2 = {k: v for k, v in count.items() if v >= min_support}
    return count2
```

Рисунок 20 – Программная реализация метода generateCk()

Генерация ассоциативных правил происходит в методе rulegenerator(), представленном на рисунке 21. Все сгенерированные правила в последствии записываются в файл Rules.txt, которые и будет отображаться в окне произведенного расчета.

```
def rulegenerator(fitems):
    counter = 0
    for itemset in fitems.keys():
        if isinstance(itemset, str):
            continue
        length = len(itemset)
        union_support = fitems[tuple(itemset)]
        for i in range(1, length):
            lefts = map(list, itertools.combinations(itemset, i))
            for left in lefts:
                if len(left) == 1:
                    if ''.join(left) in fitems:
                        leftcount = fitems[''.join(left)]
                        conf = union_support / leftcount
                else:
                    if tuple(left) in fitems:
                        leftcount = fitems[tuple(left)]
                        conf = union_support / leftcount
                if conf >= min_confidence:
                    fo = open(Rules, "a+")
                    right = list(itemset[:])
                    for e in left:
                        right.remove(e)
                    fo.write(str(left) + ' (' + str(leftcount) + ')' + ' -> ' + str(right) + ' (' + str(fitems[''.join(right)]) + ')'
                    + ' [' + str(conf) + ']' + '\n')
                    print(str(left) + ' -> ' + str(right) + ' (' + str(conf) + ')')
                    counter = counter + 1
                    fo.close()
```

Рисунок 21 – Программная реализация метода rulegenerator()

Далее необходимо произвести тестирование разработанного программного кода.

3.3 Тестирование программы

Для тестирования разработанной программы была взята выборка «Groceries», упомянутая в предыдущей главе.

На рисунке 22 представлено главное окно программы. Чтобы произвести расчет пользователю необходимо через вкладку меню «Файл» подпункт «Загрузить данные» выбрать интересующую базу данных, которая после загрузки отобразится в окне «Исходные данные» для предварительного просмотра. В случае необходимости в этом же окне пользователь также

может изменить позиции в определенных транзакциях. Далее нужно ввести значение метрик достоверности и поддержки в качестве критериев поиска и нажать кнопку «Произвести расчет».

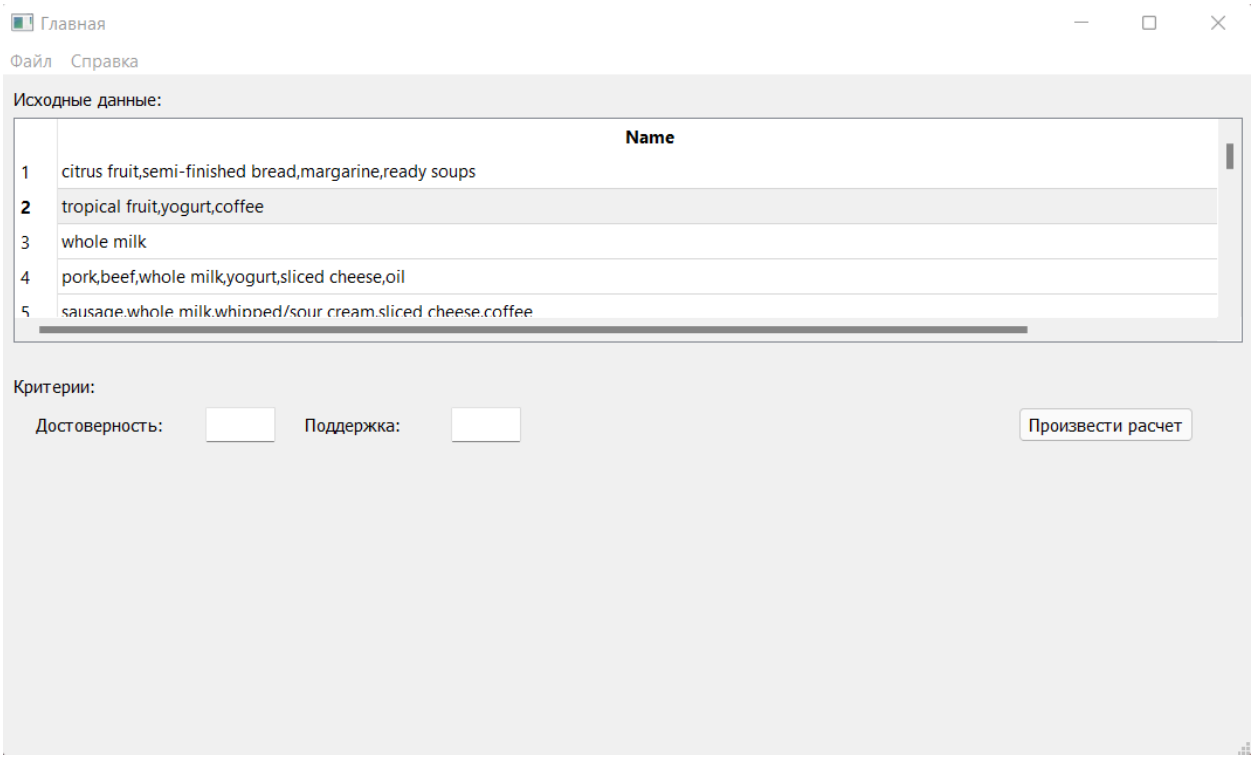

Рисунок 22 – Главное окно

В том случае, если у пользователя возникнут вопросы о том, как пользоваться программой, выбирать критерии и так далее, в меню на вкладке «Справка» реализована справочная информация. Ее окно представлено на рисунке 23. Разделы по которым могут возникнуть вопросы располагаются в левом окне. Для просмотра определённого пункта справки пользователю необходимо его выбрать и нажать на кнопку «Показать», соответствующая информация отобразиться в правом окне. Для выхода из справки и возврата к главному окну производится нажатие на кнопку «Выход».

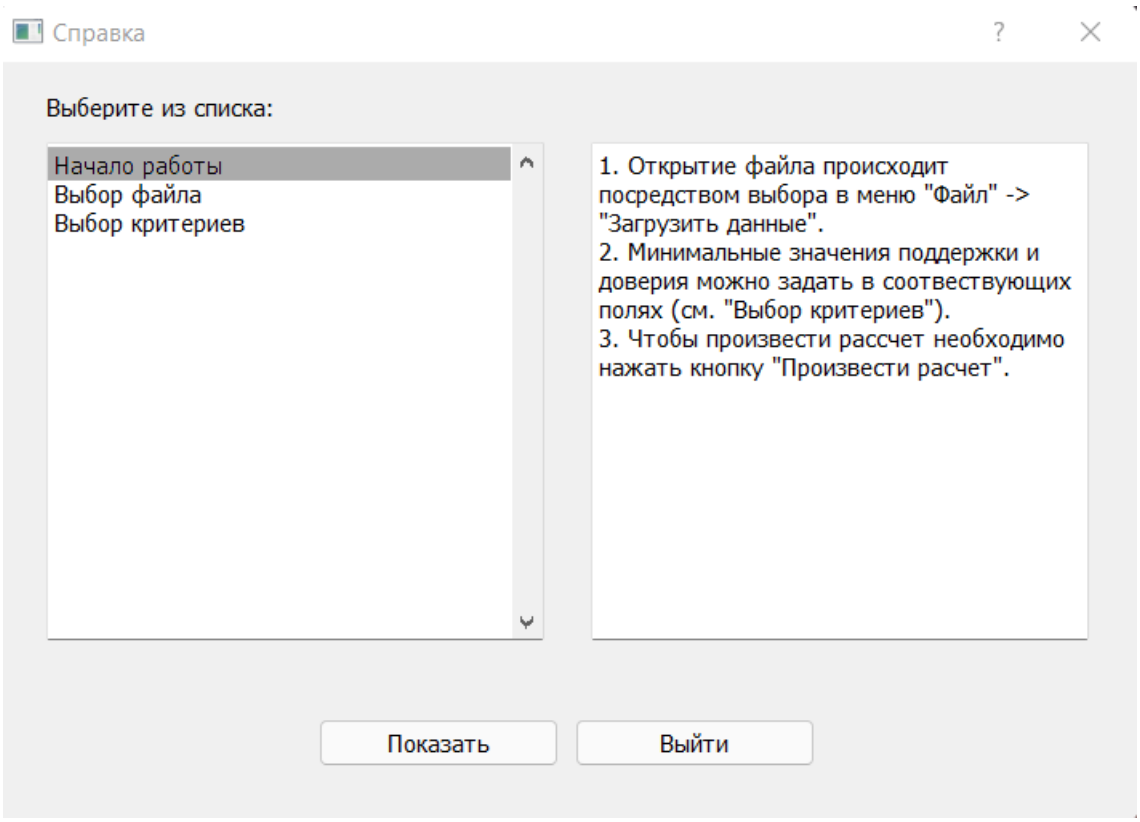

Рисунок 23 – Окно вызова справки

После того как пользователь нажал на кнопку «Произвести расчет» запускается сама работа алгоритма. По его окончанию окно сменяется на «Новый расчет», представленном на рисунке 24. Сверху в «Критерии поиска» отображаются введенные пользователем раннее критерии. В окне «Сгенерированные правила» выводятся все найденные ассоциативные правила и их значение поддержки (S) и доверия (C). В окне «Часто встречающиеся элементы» представлены товары, которые оказывались в корзине клиента чаще всего соответственно. Ниже в окне «Анализ полученных результатов» можно увидеть наглядные графики анализа проведенных транзакций, сгенерированных с помощью библиотеки Python «matplotlib».

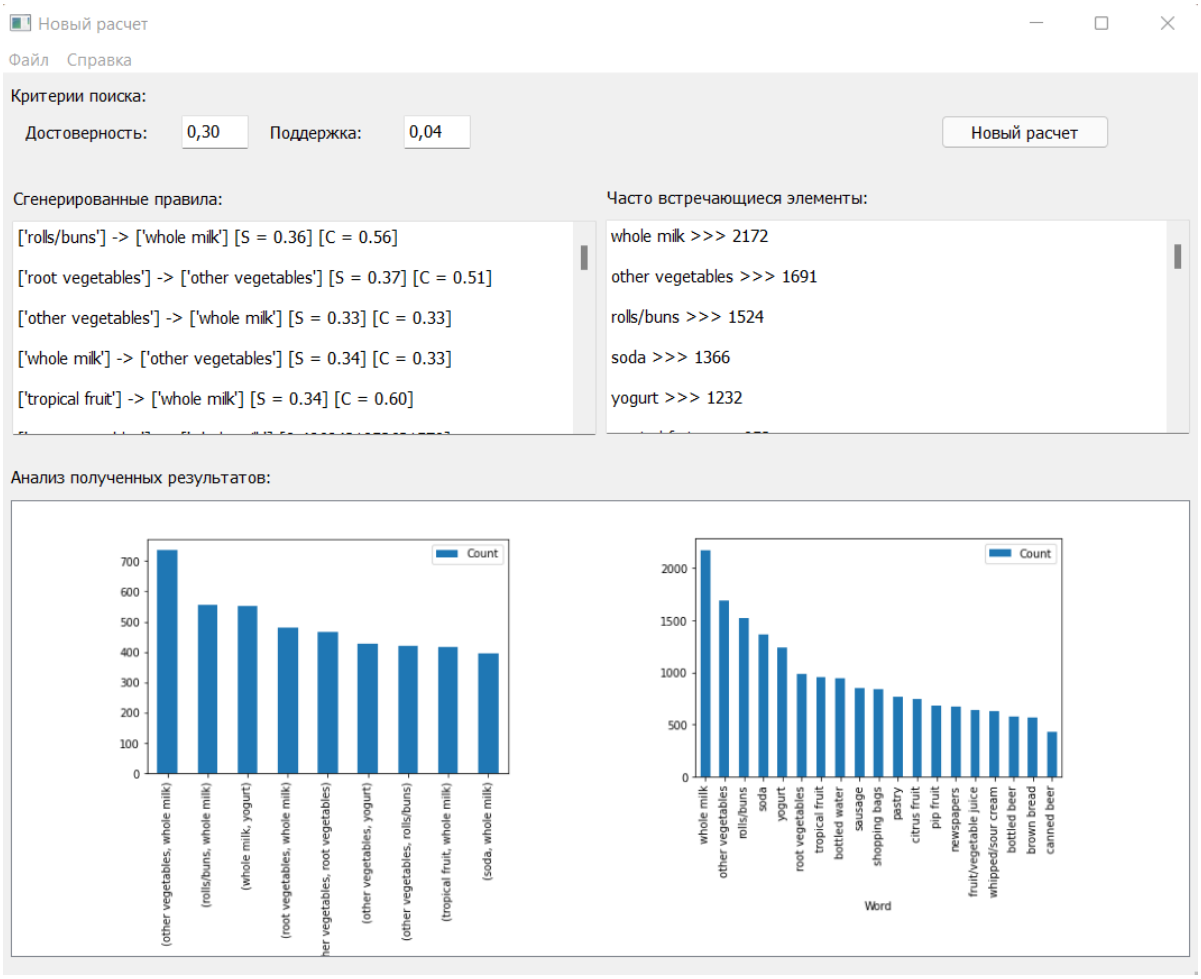

Рисунок 24 – Окно вывода результатов расчета

По результатам тестирования были получены различные наборы данных, расчет которых зависит от набранных пользователем критериев. Например, при заданных значениях поддержки и достоверности равных 0,3 и 0,04 соответственно, можно сделать вывод, что самым часто приобретаемым, как и самым часто встречающимся продуктом в паре будет являться элемент [whole milk], а наилучшими соседями на полках для данного продукта станут элементы [tropical fruit] и [rolls/buns]. Данный анализ можно провести для любого интересующего нас продукта вне зависимости от того, является ли он часто покупаемым или нет. Таким образом, опираясь на полученные результаты, мерчандайзер может составит подробную схему расположения часто встречающихся товаров в отделах для увеличения конверсии.

Заключение

В первом разделе был проведен полный анализ существующих подходов в современном мерчандайзинге и в интеллектуальном анализе больших объемов данных, а также было рассмотрено и классифицировано такое важное понятие как Data Mining. На основе проведенного анализа был сделан вывод, что при разработке системы оптимизации расположения товаров следует опираться на теорию анализа рыночной корзины покупателя, сформулированную Р. Агравалом.

Во втором разделе теоретически была рассмотрена модель аффинитивного анализа данных и ее основные метрики, также был приведен наглядный пример области их использования и расчета.

Были описаны основные алгоритмы, основанные на поиске ассоциативных правил, такие как Apriori, FP-Growth и Eclat, а именно схемы их работы и подробный анализ каждого из алгоритмов с выделением их преимуществ и недостатков. После на основе проведенных исследований и тестовых данных были составлены сравнительные таблицы, которые показали, что наилучшим решением в сфере анализа рыночной корзины, исходя из поставленной цели, является алгоритм Apriori.

В третьем разделе был осуществлен сравнительный анализ существующих фреймворков для разработки на языке Python, таких как Kivy и PyQt, для наилучшего опыта работы с программой. Далее подробно была рассмотрена программная реализация алгоритма Apriori и проведено последующее тестирование разработанной системы на основе тестового набора данных о транзакциях продуктового магазина.

Таким образом, в ходе выполнения выпускной квалификационной работы была разработана программа для оптимизации системы расположения товаров в торговой точке, на основе интеллектуального анализа данных о покупках, кроме того в полном объеме были выполнены все поставленные задачи.

Список используемой литературы и используемых источников

1. Документация по Python [Электронный ресурс] / URL: https://www.python.org/doc/ (дата обращения 10.03.22)
2. Макшанов А.В., Журавлев А.Е. Технологии интеллектуального анализа данных : учебное пособие; под ред. А. Воронова. : Лань, 2018. 212 с.
3. Нордфальт Й. Ритейл-маркетинг: Практики и исследования; под ред. А. Воронова. : Интеллектуальная Литература, 2015. 760 с.
4. Храмов, А. Г. Методы и алгоритмы интеллектуального анализа данных : учебное пособие : Самара : Самарский университет, 2019. 176 с.
5. Agarwal, J. Mining Frequent Quality Factors of Software System Using Apriori Algorithm / Jyoti Agarwal, Sanjay Kumar Dubey, Rajdev Tiwari // Proceedings of the International Conference on Data Engineering and Communication Technology (ICDECT 2016) // Springer Science+Business Media Singapore, 2017. pp. 481-490.
6. Al-Maolegi, M. An improved apriori algorithm for association rules / Mohammed Al-Maolegi, Bassam Arkok // International Journal on Natural Language Computing (IJNLC) vol. 3, 2014. pp. 138-148.
7. Apriori Algorithm for Association Rule Learning — How To Find Clear Links Between Transactions [Электронный ресурс]. URL: https://towardsdatascience.com/apriori-algorithm-for-association-rule-learning-how-to-find-clear-links-between-transactions-bf7ebc22cf0a (дата обращения: 12.04.2022).
8. An Introduction on Market Basket Analysis [Электронный ресурс]. URL: https://towardsdatascience.com/a-gentle-introduction-on-market-basket-analysis-association-rules-fa4b986a40ce (дата обращения: 12.04.2022).
9. Association rule learning. [Электронный ресурс] URL: https://en.wikipedia.org/wiki/Association_rule_learning#cite_note-apriori-8 (дата обращения: 30.03.2022).

10. Association Rule Mining using ECLAT Algorithm [Электронный ресурс]. URL: https://medium.com/machine-learning-and-artificial-intelligence/3-4-association-rule-mining-using-eclat-algorithm-b6e50aab2147 (дата обращения: 12.04.2022).

11. Chowdhary, S. A Comprehensive Analysis of Proprietary and Open Source Data Mining Tools / Sonia Rani Chowdhary, Mr Vikash // International Journal of Scientific Research in Computer Science, Engineering and Information Technology, 10.32628/CSEIT206210, 2020. pp. 111-119.

12. Dahbi, A. Using Multiple Minimum Support to Auto-adjust the Threshold of Support in Apriori Algorithm : Azzeddine Dahbi, Youssef Balouki, Taoufiq Gadi // Proceedings of the Ninth International Conference on Soft Computing and Pattern Recognition (SoCPaR 2017) // Springer International Publishing AG 2018. pp. 120-126.

13. Framework [Definition] Types of Frameworks [Электронный ресурс]. URL: https://hackr.io/blog/what-is-frameworks (дата обращения: 5.05.2022).

14. Gar, K. Comparing the Performance of Frequent Pattern Mining Algorithms / Dr. Kanwal Gar, Deepak Kumar // International Journal of Computer Applications vol. 69, 2013. pp. 654-661.

15. Heaton, J. Comparing Dataset Characteristics that Favor the Apriori, Eclat or FP-Growth Frequent Itemset Mining Algorithms // arXiv:1701.09042v1 [cs.DB], 2017. pp. 277-285.

16. Jiawei, H. Data Mining: Concepts and Techniques: The Morgan Kaufmann Series in Data Management Systems 3rd Edition : Morgan Kaufmann, 2011. 744 p.

17. Kaur, M. ECLAT Algorithm for Frequent Itemsets Generation / Manjit Kaur , Urvashi Grag // International Journal of Computer Systems, vol. 1, 2014. pp. 218-220.

18. Kivy: Cross-platform Python Framework [Электронный ресурс]. URL: https://kivy.org/#home (дата обращения: 30.04.2022).

19. Manpreet Kaur, Shivani Kang. Market Basket Analysis: Identify the changing trends of market data using association rule mining // International Conference on Computational Modeling and Security, Springer International Publishing AG, 2017. pp. 138-148.

20. Qt for Python [Электронный ресурс]. URL: https://doc.qt.io/qtforpython/ (дата обращения: 30.04.2022).

21. Rao, A. Application of market–basket analysis on healthcare / Abishek B. Rao, Jammula Surya Kiran1, Poornalatha G // Int J Syst Assur Eng Manag. 2021. pp. 548-555.

22. Sterne, J. Artificial Intelligence for Marketing: Practical Applications : Wiley; 1st edition, 2017. 368 p.

23. Understanding Association Mining and Market Basket Analysis with Apriori Algorithm using Python [Электронный ресурс]. URL: https://medium.com/analytics-vidhya/understanding-association-mining-and-market-basket-analysis-with-apriori-algorithm-using-python-e216c0d22f3e (дата обращения: 13.04.2022).

24. Wicaksono, D. The Comparison of Apriori Algorithm with Preprocessing and FP-Growth Algorithm for Finding Frequent Data Pattern in Association Rule / Deo Wicaksono, Muhammad Jambak // Sriwijaya International Conference on Information Technology and Its Applications, vol. 172. pp. 28-35.

25. Zeng, Y. Research of Improved FP-Growth Algorithm in Association Rules Mining / Yi Zeng, Shiqun Yin, Jiangyue Liu, Miao Zhang // Hindawi Publishing Corporation Scientific Programming, vol. 2015. pp 10-15.