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

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 «Прикладная математика и информатика»
(код и наименование направления подготовки / специальности)

Компьютерные технологии и математическое моделирование
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Исследование и реализация алгоритма поиска ассоциативных правил в большом наборе данных

Обучающийся А.С. Анисифоров _____
(Инициалы Фамилия) (личная подпись)

Руководитель к. т. н, доцент, Н.А. Сосина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к. ф. н., доцент, Н.В. Ященко
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема выпускной квалификационной работы – «Исследование и реализация алгоритма поиска ассоциативных правил в большом наборе данных».

Ключевые слова: исследование алгоритма, поиск ассоциативных правил, большие наборы данных.

Разработка эффективных алгоритмов и их оптимизация для работы с большими данными являются важными задачами, которые имеют как научное, так и практическое значение.

Объектом исследования бакалаврской работы является большой набор данных.

Предметом исследования бакалаврской работы является алгоритм поиска ассоциативных правил.

Цель бакалаврской работы – исследование и программная реализация алгоритма поиска ассоциативных правил в большом наборе данных.

Методы исследования – методы интеллектуального анализа данных, методы и технологии разработки программного обеспечения.

Работа состоит из введения, трех глав, заключения и списка используемой литературы и используемых источников.

Исследование и реализация алгоритмов поиска ассоциативных правил вносят вклад в развитие теории анализа данных и машинного обучения.

Практическая реализация таких алгоритмов позволяет решать реальные задачи в различных отраслях.

Выпускная квалификационная работа состоит из 40 страниц текста, 20 рисунков, 3 таблиц и 25 источников.

Abstract

The title of the graduation work is Research and implementation of an Algorithm for Association Rule Mining in Large Datasets

Keywords: algorithm research, association rule mining, Big Data.

The development and optimization of efficient algorithms for working with large datasets are important tasks with both scientific and practical relevance.

The object of the graduation work is a large dataset.

The subject of the graduation work is an algorithm for association rule mining.

The aim of the work is to study and implement an algorithm for discovering association rules in a large dataset.

Research methods include data mining techniques and software development methods and technologies. The work consists of an introduction, three chapters, a conclusion, and a list of references and sources used.

The research and implementation of association rule mining algorithms contribute to the advancement of data analysis and machine learning theory.

The practical application of such algorithms enables solving real-world problems across various industries.

The graduation work consists of 40 pages of text, 20 figures, 3 tables and 25 sources.

Оглавление

Введение

Точное, полное и быстрое определение потребностей клиентов и наличие рекомендаций по товарам и услугам являются решающими моментами с точки зрения повышения уровня удовлетворенности клиентов в различных областях человеческой деятельности.

Из-за значительного увеличения количества транзакций и клиентов анализ затрат по времени и потреблению компьютерной памяти становится выше. Чтобы повысить производительность рекомендаций по товарам и услугам используется подход, основанный на поиске и анализе правил ассоциации в больших наборах данных. «Для решения данной задачи используется методы и алгоритмы интеллектуального анализа данных, направленные на выявление часто встречающихся закономерностей, корреляций или ассоциаций в наборах данных в различных реляционных и транзакционных базах данных.

Алгоритм поиска ассоциативных правил в машинном обучении имеют решающее значение для извлечения ценной информации из данных, особенно в сценариях, где традиционные математические подходы могут оказаться непригодными» [10].

Актуальность темы исследования и реализации алгоритмов поиска ассоциативных правил в больших наборах данных обусловлена их широким применением в различных сферах, необходимостью обработки растущих объемов данных и развитием технологий анализа данных. Разработка эффективных алгоритмов и их оптимизация для работы с большими данными являются важными задачами, которые имеют как научное, так и практическое значение.

Объектом исследования бакалаврской работы является большой набор данных.

Предметом исследования бакалаврской работы является алгоритм поиска ассоциативных правил.

Цель бакалаврской работы – исследование и программная реализация алгоритма поиска ассоциативных правил в большом наборе данных.

Для достижения данной цели необходимо выполнить следующие задачи:

- выполнить постановку задачи исследования алгоритма поиска ассоциативных правил в большом наборе данных;
- проанализировать алгоритмы поиска ассоциативных правил в большом наборе данных;
- разработать и протестировать программу, реализующую алгоритм поиска ассоциативных правил в большом наборе данных.

«Методы исследования – методы интеллектуального анализа данных (Data mining), методы и технологии разработки программного обеспечения» [10].

Практическая значимость бакалаврской работы заключается в разработке программы, позволяющий производить поиск ассоциативных правил в большом наборе данных.

Данная работа состоит из введения, трех глав, заключения, списка используемой литературы и используемых источников.

Выпускная квалификационная работа состоит из 40 страниц текста, 20 рисунков, 3 таблиц и 25 источников.

Глава 1 Постановка задачи исследования алгоритма поиска ассоциативных правил в большом наборе данных

1.1 Математическое описание задачи поиска ассоциативных правил в большом наборе данных

«Поиск ассоциативных правил – это метод, используемый для выявления закономерностей в больших наборах данных. Он включает в себя поиск связей между переменными в данных и использование этих связей для прогнозирования или принятия решений» [17].

Цель анализа ассоциативных правил – выявить правила, описывающие отношения между различными элементами набора данных.

Рассмотрим математическую постановку задачи поиска ассоциативных правил.

«Ассоциативные правила представляют собой механизм нахождения логических закономерностей между связанными элементами (событиями или объектами).

Примерами приложения ассоциативных правил могут быть следующие задачи:

- выявление наборов товаров, которые в супермаркетах часто покупаются вместе или никогда не покупаются вместе;
- определение доли клиентов, положительно относящихся к нововведениям в их обслуживании; о определение профиля посетителей веб-ресурса;
- определение доли случаев, в которых новое лекарство показывает опасный побочный эффект и другие» [8].

Рассмотрим математическое описание задачи поиска ассоциативных правил в большом наборе данных.

Пусть имеется набор:

$I = \{i_1, i_2, , \dots, i_n\}$ из n двоичных атрибутов, которые назовем объектами.

Пусть дан набор транзакций вида:

$D = \{t_1, t_2, , \dots, t_n\}$, который назовем базой данных (БД).

Тогда ассоциативное правило определяется как импликация вида (1):

$$X \Rightarrow Y, \tag{1}$$

где $X, Y \subseteq I$

Правила ассоциации создаются путем тщательного анализа набора данных и поиска частых шаблонов «если/то».

Для оценки производительности алгоритмов интеллектуального анализа правил ассоциации обычно используются несколько показателей, представленных ниже.

«Поддержка (support) – показатель, насколько часто набор объектов обнаружится в базе данных. Определяется по формуле (2):

$$supp(X) = \frac{|\{t \in T; X \subseteq t\}|}{|T|}, \tag{2}$$

где T – набор транзакций БД.

Доверие (confidence) – показатель, насколько часто правило оказывается верным. Определяется по формуле (3)» [24]:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} . \tag{3}$$

Лифт (lift) правила количественно определяет, насколько вероятно возникновение последствий при наличии антецедента по сравнению с ситуацией, когда два события независимы. Определяется по формуле (4):

$$lift(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)} . \tag{4}$$

Еще одним показателем значимости ассоциативных правил является коэффициент интереса (conviction), определяемый по формуле (5):

$$conv(X \Rightarrow Y) = \frac{1 - supp(Y)}{1 - conf(X \Rightarrow Y)} \quad (5)$$

Он показывает, насколько сильно правило $X \Rightarrow Y$ отличается от случайного выбора.

«Следует учесть, что поиск ассоциативных правил совсем не тривиальная задача, как может показаться на первый взгляд. Одна из проблем – алгоритмическая сложность при нахождении часто встречающихся наборов элементов, т.к. с ростом числа элементов в I ($|I|$) экспоненциально растет число потенциальных наборов элементов» [2].

1.2 Методы поиска ассоциативных правил

Методы поиска ассоциативных правил – это техники, используемые для выявления интересных взаимосвязей и закономерностей в больших наборах данных, особенно в контексте анализа данных и машинного обучения [5].

Наиболее известным примером применения ассоциативных правил является анализ покупательского поведения, когда исследуется, какие товары часто покупаются вместе.

Рассмотрим основные методы поиска ассоциативных правил.

«Классический метод поиска ассоциативных правил рассматривает все возможные комбинации условий и следствий, оценивает для них поддержку и достоверность, а затем исключает все ассоциации, которые не удовлетворяют заданным ограничениям.

Число возможных ассоциаций с увеличением числа предметов растет экспоненциально. Если в базе данных транзакций присутствует k предметов и все ассоциации являются бинарными (то есть содержат по одному предмету в

условии и следствии), то потребуется проанализировать $k \cdot 2k - 1$ ассоциаций.

Поскольку реальные базы данных транзакций, рассматриваемые при анализе рыночной корзины, обычно содержат тысячи предметов, вычислительные затраты при поиске ассоциативных правил огромны.

Например, если рассматривать выборку, содержащую всего 100 предметов, то количество ассоциаций, образуемых этими предметами, составит $100 \cdot 299 \approx 6{,}4 \cdot 1031$.

Как показал анализ, в процессе генерации ассоциативных правил широко используются методики, позволяющие уменьшить количество ассоциаций, которое требуется проанализировать. Одной из наиболее распространенных является концепция, основанная на обнаружении так называемых частых наборов, когда анализируются только те ассоциации, которые встречаются достаточно часто. На основе данной концепции построен известный алгоритм Apriori» [7].

Методы поиска ассоциативных правил на основе кластеризации представляют собой подход, который сочетает в себе техники кластеризации и ассоциативного анализа для выявления закономерностей и взаимосвязей в данных. Этот подход может быть особенно полезен, когда данные имеют сложные структуры или когда необходимо уменьшить размерность данных перед применением традиционных методов ассоциативного анализа.

В качестве примера рассмотрим алгоритм, называемый правилом ассоциации на основе кластера.

«Метод (Cluster-Based Association Rule или CBAR) заключается в создании таблиц кластеров путем сканирования базы данных один раз, а затем кластеризации записей транзакций в k-ю таблицу кластера, где длина записи равна k. Более того, большие наборы элементов генерируются путем контрастов с частичными таблицами кластера» [25]. Это не только сокращает значительные объемы данных, сокращая время, необходимое для выполнения сканирования данных, и требуя меньшего контраста, но и обеспечивает правильность полученных результатов.

Псевдокод алгоритма CBAR показан на рисунке 1.

Input: D, $MinSup$

Output: Answer (Answer = $\cup L_k$, for $1 \le k \le M$)

Begin

1) ***Cluster_Table_Create***(D, $MinSup$);
2) **for** (k=2; $L_{k-1} \neq \emptyset$; k++) **do** {
3) C_k =***Candidate_Itemset_Gen*** (L_{k-1});
4) L_k =***Large_Itemset_Gen*** (C_k);
5) }
6) Answer = $\cup L_k$;

End

Рисунок 1 – Псевдокод алгоритма CBAR

Некоторые алгоритмы могут комбинировать кластеризацию и ассоциативный анализ в одном процессе, что позволяет одновременно выявлять кластеры и ассоциативные правила. Например, алгоритмы, которые используют иерархическую кластеризацию для группировки объектов и одновременно выявляют ассоциативные правила внутри каждого кластера.

Рассмотрим методы поиска ассоциативных правил на основе теории графов.

«Алгоритмы, основанные на данном подходе, можно разделить на 2 группы: графовые модели и алгоритмы поиска правил, использующие графовые структуры» [12].

Можно использовать графовые модели, такие как графы частых наборов FIG (Frequent Itemset Graphs), для представления частых наборов элементов и их ассоциативных правил. Это позволяет эффективно находить связи между элементами.

На рисунке 2 представлен пример реализации концепция FIG [15].

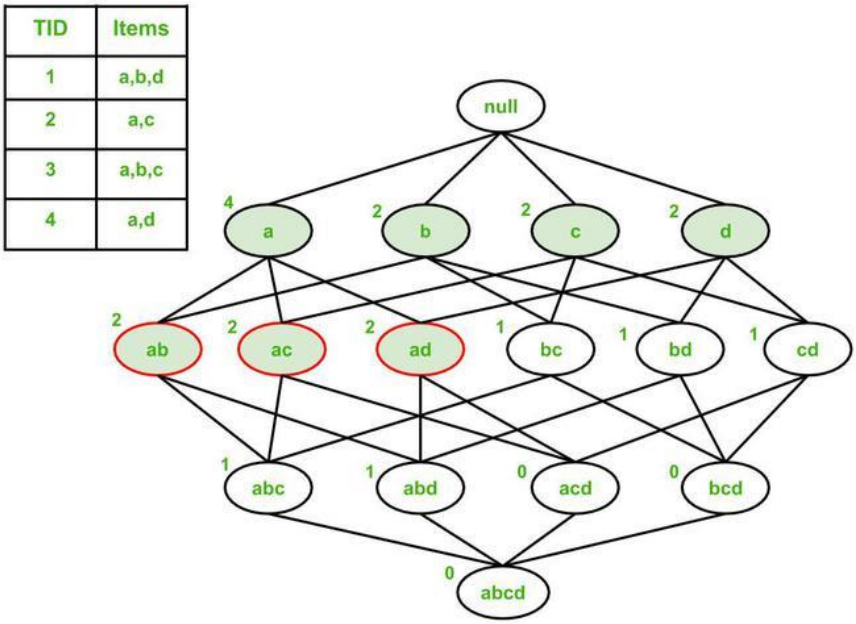

Рисунок 2 – Пример реализации концепции FIG

Количество поддержек показано в левом верхнем углу каждого узла. Предположим, что порог количества поддержек равен 50% , то есть каждый элемент должен встречаться в 2 или более транзакциях. На основе этого порога частыми наборами элементов являются a, b, c, d, ab, ac и ad (затененные узлы).

Из этих 7 часто встречающихся наборов элементов 3 определены как максимально часто встречающиеся (имеют красный контур):

– ab: непосредственные супермножества abc и abd встречаются редко;
– ac: Непосредственные супермножества abc и acd встречаются редко;
– ad: Непосредственные супермножества abd и bcd встречаются редко.

Оставшиеся 4 частых узла (a, b, c и d) не могут быть максимально частыми, поскольку все они имеют по крайней мере 1 непосредственный надмножество, которое является частым.

Недостатком данного подхода является то, что количество поддерживаемых максимально частых наборов элементов не дает никакой

информации о количестве поддерживаемых их подмножеств. Это означает, что для определения количества поддерживаемых не максимально частых наборов элементов требуется дополнительный обход данных, что может быть нежелательным в некоторых случаях.

Алгоритмы, такие как «GSP (Graph-based Sequential Pattern mining), могут быть адаптированы для поиска ассоциативных правил, используя графовые структуры для анализа последовательностей и взаимосвязей» [15].

Псевдокод алгоритма GSP показан на рисунке 3.

```
setOfCandidateSeqs1 = setOfAllItems /* Создать последовательности длиной 1 */for (k = 1;
setOfCandidateSeqsk is not empty; k++) /* Рассмотрим последовательности длины k */for each
seq in setOfCandidateSeqsk /* Поддерживать последовательности, соответствующие
минимальной указанной поддержке*/
DetermineSupport(seq)setOfAllFrequentSeqsk = {seq | seq in setOfCandidateSeqsk has minimal
support}if setOfAllFrequentSeqsk is not emptysetOfCandidateSeqsk+1 =
GenerateCandidates(setOfAllFrequentSeqsk) /* Расширить последовательности */else
setOfCandidateSeqsk+1 = { }
```

Рисунок 3 – Псевдокод алгоритма GSP

«GSP использует уровневую парадигму для поиска всех шаблонов последовательностей в данных. Он начинается с поиска частых элементов размера один, а затем передает их в качестве входных данных для следующей итерации алгоритма GSP. База данных передается этому алгоритму несколько раз. На каждой итерации GSP удаляет все нечастые наборы элементов» [15].

Это делается на основе пороговой частоты, которая называется поддержкой. Сохраняются только те наборы элементов, частота которых больше количества поддержки.

После первого прохода GSP находит все частые последовательности длины 1, которые называются 1-последовательностями. Это делает входные данные для следующего прохода, это кандидат для 2-последовательностей.

«В конце этого прохода GSP генерирует все частые 2-последовательности, что делает входные данные для кандидатов 3-последовательностей. Алгоритм вызывается рекурсивно до тех пор, пока не будут найдены более частые наборы элементов» [15].

Недостатком GSP являются существенные затраты времени на сканирование базы данных.

Алгоритмы, основанные на теории графов, могут быть использованы для выявления закономерностей в сложных взаимосвязях.

Методы поиска ассоциативных правил играют важную роль в анализе данных и могут быть применены в различных областях, таких как маркетинг, розничная торговля, биоинформатика и другие.

Выбор конкретного метода поиска ассоциативных правил зависит от характера данных, объема данных и требований к эффективности.

Выводы по главе 1

Методы поиска ассоциативных правил играют важную роль в анализе данных и могут быть применены в различных областях, таких как маркетинг, розничная торговля, биоинформатика и другие. Выбор конкретного метода поиска ассоциативных правил зависит от характера данных, объема данных и требований к эффективности.

Глава 2 Обзор и анализ алгоритмов поиска ассоциативных правил

Рассмотрим свойства известных алгоритмов поиска ассоциативных правил: AIS, SETM и Apriori.

Алгоритм AIS был первым алгоритмом, предложенным Агравалом, Имелински и Свами для интеллектуального анализа правил ассоциации [20].

Алгоритм AIS состоит из следующих шагов.

Шаг 1. «Наборы элементов-кандидатов генерируются и подсчитываются на лету по мере сканирования базы данных.

Шаг 2. Для каждой транзакции определяется, какие из больших наборов элементов предыдущего прохода содержатся в этой транзакции.

Шаг 3. Новые наборы элементов-кандидатов генерируются путем расширения этих больших наборов элементов другими элементами в этой же транзакции.

Иными словами, AIS фокусируется на улучшении качества баз данных вместе с необходимой функциональностью для обработки запросов поддержки принятия решений. В этом алгоритме генерируются только правила ассоциации с одним элементом, что означает, что консеквент этих правил содержит только один элемент, например, могут быть сгенерированы такие правила, как $X \cap Y \Rightarrow Z$, но не такие правила, как $X \Rightarrow Y \cap Z$.

Базы данных сканировались много раз, чтобы получить частые наборы элементов в AIS. Чтобы сделать этот алгоритм более эффективным, был введен метод оценки для отсечения тех кандидатов наборов элементов, которые не имеют надежды быть большими, следовательно, можно избежать ненужных усилий по подсчету этих наборов элементов.

Поскольку предполагается, что все кандидаты наборов элементов и частые наборы элементов хранятся в основной памяти, управление памятью также предлагается для AIS, когда памяти недостаточно (рисунок 4)» [20].

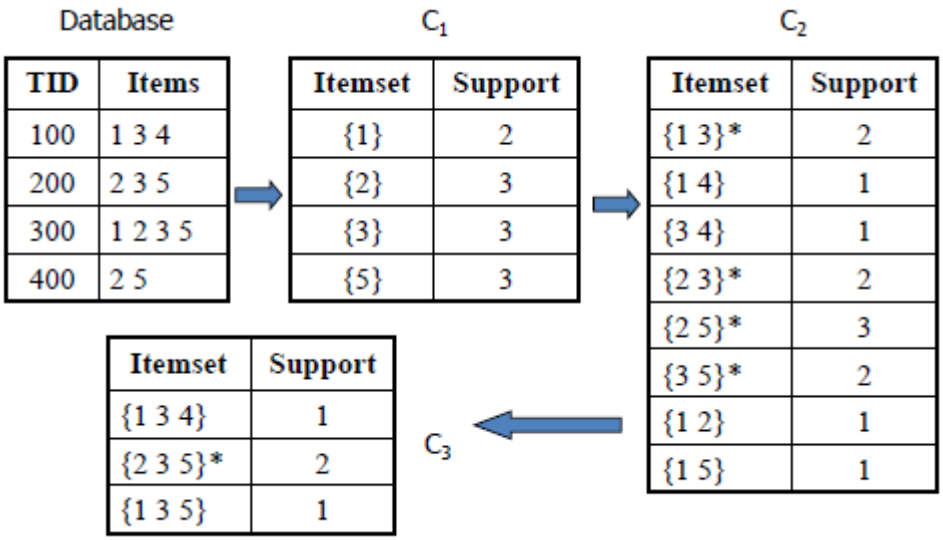

Рисунок 4 – Пример работы алгоритма AIS

«В алгоритме AIS частые наборы элементов были сгенерированы путем сканирования баз данных несколько раз. Количество поддержки каждого отдельного элемента было накоплено во время первого прохода по базе данных. На основе минимального количества поддержки те элементы, количество поддержки которых меньше его минимального значения, исключаются из списка элементов. Двухэлементные наборы кандидатов генерируются путем расширения частых одноэлементных наборов другими элементами в транзакции» [6]. Во время второго прохода по базе данных количество поддержки этих кандидатов двухэлементных наборов накапливается и проверяется на соответствие порогу поддержки. Аналогично эти (k+1)-элементные наборы кандидатов генерируются путем расширения частых k-элементных наборов элементами в той же транзакции.

Процесс генерации наборов кандидатов и частых наборов элементов повторяется до тех пор, пока любой из них не станет пустым.

Алгоритм SETM состоит из нижеследующих шагов.

Шаг 1. «Наборы элементов-кандидатов генерируются на лету по мере сканирования базы данных, но подсчитываются в конце прохода.

Шаг 2. Новые наборы элементов-кандидатов генерируются так же, как в

алгоритме AIS, но идентификатор TID генерирующей транзакции сохраняется с набором кандидатов в последовательной структуре.

Шаг 3. В конце прохода количество поддерживаемых элементов-кандидатов определяется путем агрегирования этой последовательной структуры.

В алгоритме SETM наборы элементов-кандидатов генерируются «на лету» по мере сканирования базы данных, но подсчитываются в конце прохода. Затем генерируются новые наборы элементов-кандидатов так же, как в алгоритме AIS, но идентификатор TID генерирующей транзакции сохраняется вместе с набором элементов-кандидатов в последовательной структуре. Это отделяет процесс генерации кандидатов от подсчета.

В конце прохода количество поддерживаемых наборов элементов-кандидатов определяется путем агрегирования последовательной структуры (рисунок 5)» [6].

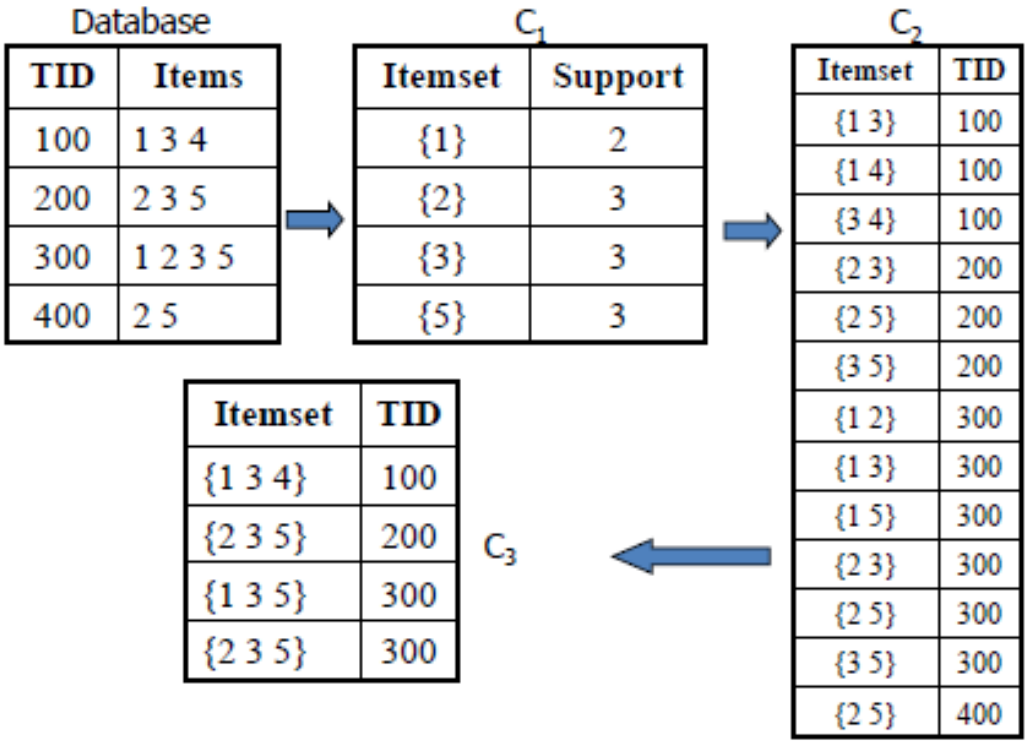

Рисунок 5 – Пример работы алгоритма SETM

«Алгоритм Apriori используется для добычи частых наборов элементов и изучения правил ассоциации. Алгоритм использует поиск по уровням, где k-наборы элементов используются для исследования (k+1)-наборов элементов, для добычи частых наборов элементов из транзакционной базы данных для булевых правил ассоциации. В этом алгоритме частые подмножества расширяются по одному элементу за раз, и этот шаг известен как процесс генерации кандидатов. Затем группы кандидатов проверяются по данным. Для эффективного подсчета наборов элементов-кандидатов Apriori использует метод поиска в ширину и структуру хеш-дерева.

Он определяет частые отдельные элементы в базе данных и расширяет их до все больших и больших наборов элементов, пока эти наборы элементов появляются в базе данных достаточно часто.

Алгоритм Apriori определяет частые наборы элементов, которые можно использовать для определения правил ассоциации, которые выделяют общие тенденции в базе данных» [14].

На рисунке 6 показан псевдокод классического алгоритма Apriori.

F_1 = {frequent items of size 1};
for (k = 1; F_k !=ϕ; k++) do begin
  C_{k+1} = apriori-gen(F_k); // New candidates generated from F_k
  for all transactions t in database do begin
    C'_t = subset (C_{k+1},t); // Candidates contained in t
    for all candidate $c \in C'_t$ do
      $c.count$++; // Increment the count of all candidates
      in C_{k+1} that are contained in t
    end
    F_{k+1} = {$C \in C_{k+1} | c.count \geq$ minimum suport}
    //Candidates in C_{k+1} with minimum support
  end
end
Answer $\bigcup_k F_k$;

Рисунок 6 – Псевдокод классического алгоритма Apriori

Пример работы алгоритма Apriori показан на рисунке 7.

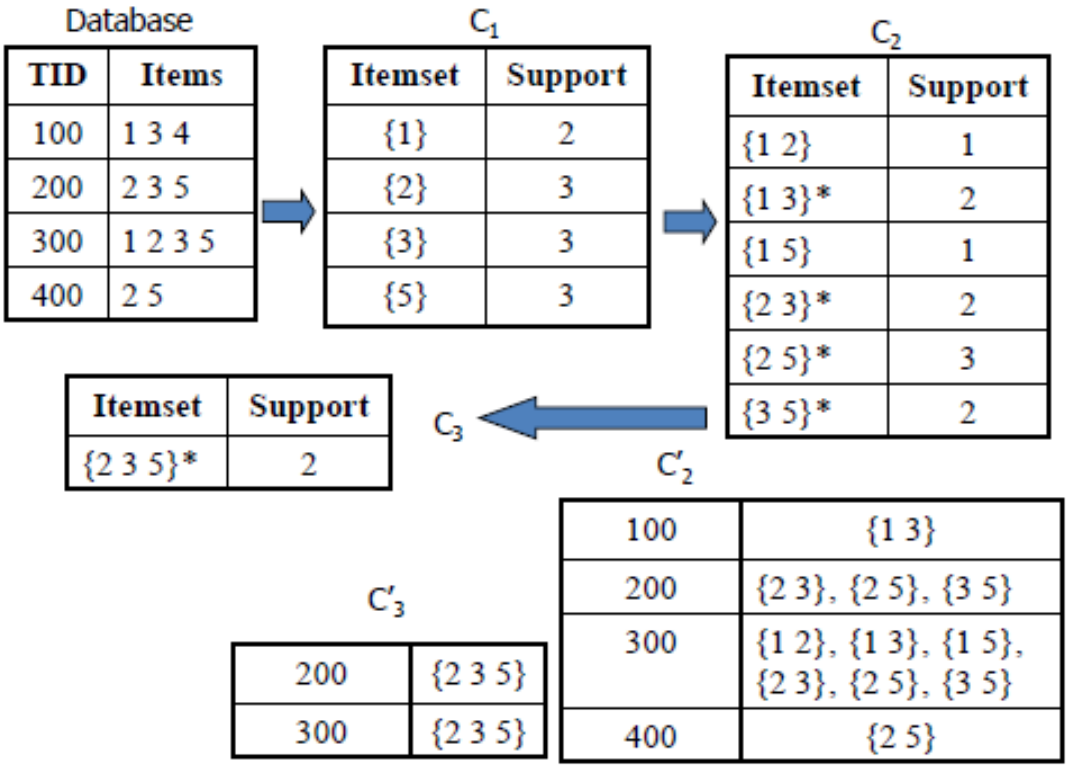

Рисунок 7 – Пример работы алгоритма Apriori

Для сравнения характеристик алгоритмов поиска ассоциативных правил построена таблица 1.

Таблица 1 – Характеристики алгоритмов поиска ассоциативных правил

Алгоритм	Преимущества	Недостатки
AIS	«Алгоритм использовался для определения наличия связи между отделами в поведении покупателей» [14]	Приводит к ненужной генерации и подсчету слишком большого количества наборов элементов-кандидатов, которые оказываются небольшими.

Продолжение таблицы 1

Алгоритм	Преимущества	Недостатки
SETM	Генерируя наборы кандидатов, алгоритм SETM последовательно сохраняет копию наборов кандидатов вместе с TID генерирующей транзакции	Имеет те же недостатки, что и алгоритм AIS.
Apriori	Алгоритм Apriori использует тот факт, что любое подмножество часто встречающегося набора элементов также является частым набором элементов. Таким образом, алгоритм может сократить количество рассматриваемых кандидатов, исследуя только те наборы элементов, количество поддержки которых больше минимального количества поддержки. Все нечастые наборы элементов могут быть отсечены, если у них есть нечастое подмножество. Имеются эффективные модификации алгоритма.	Сложный процесс генерации кандидатов, который использует большую часть времени, пространства и памяти. Требует многократного сканирования базы данных

По результатам анализа выбираем алгоритм Apriori.

«Как отмечено выше, для повышения эффективности алгоритма Apriori разработаны его модификации AprioriTid и AprioriHybrid.

В алгоритме AprioriTid база данных не используется для подсчета поддержки наборов элементов-кандидатов после первого прохода. Процесс генерации наборов элементов-кандидатов такой же, как и в алгоритме Apriori.

Генерируется другой набор C’, в котором каждый участник имеет TID каждой транзакции и больших наборов элементов, присутствующих в этой транзакции. Сгенерированный набор, т. е. C’ используется для подсчета поддержки каждого кандидата на набор элементов (рисунок 8)» [23].

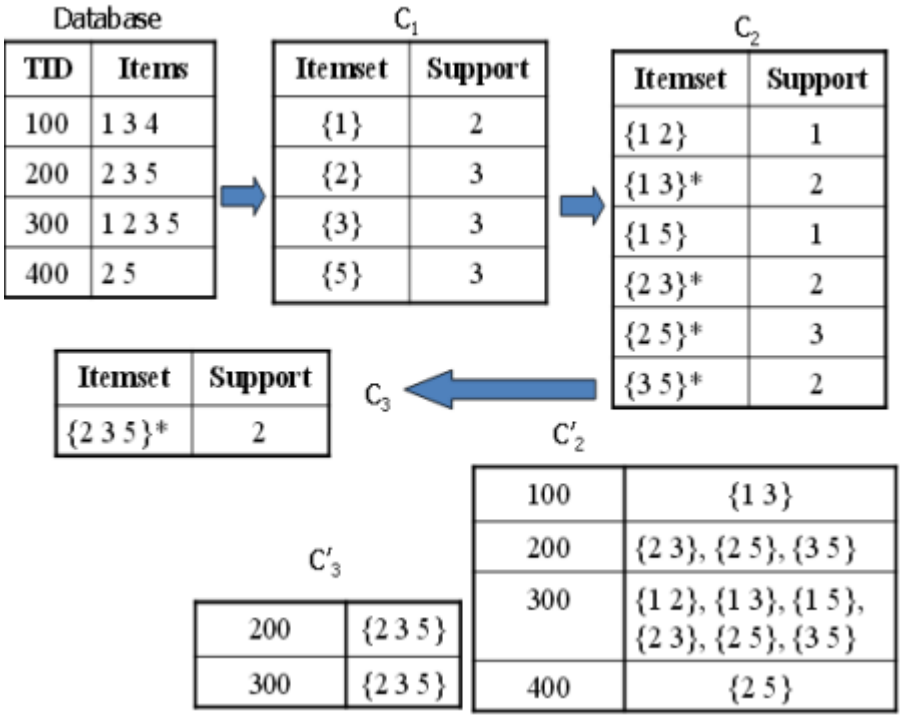

Рисунок 8 – Пример работы алгоритма AprioriTid

Преимущество этого алгоритма в том, что на более поздних проходах производительность AprioriTid лучше, чем Apriori.

Также разработан алгоритм AprioriHybrid, который использует достоинства обоих вышеописанных версий Apriori [23].

Выводы к главе 2

Для повышения эффективности алгоритма Apriori разработаны его модификации AprioriTid и AprioriHybrid.

Глава 3 Реализация и тестирование алгоритма поиска ассоциативных правил в большом наборе данных

3.1 Выбор средства разработки программы

«Для реализации алгоритма поиска ассоциативных правил в большом наборе данных используем язык программирования Python» [3].

«Для выбора среды разработки сравним характеристики двух сред разработки: Jupyter Notebook и Replit.com» [9].

Рассмотрим возможности среды Jupyter Notebook.

«Jupyter Notebook – это веб-приложение с открытым исходным кодом, которое позволяет пользователям создавать и обмениваться документами, содержащими живой код, уравнения, визуализации и повествовательный текст. Оно широко используется в науке о данных, машинном обучении, научных вычислениях и академических исследованиях благодаря своей способности сочетать выполнение кода с расширенным форматированием текста (рисунок 9)» [13].

Рисунок 9 – Главная страница среды Jupyter Notebook

Вот некоторые ключевые особенности и аспекты Jupyter Notebook:

- интерактивное кодирование: пользователи могут писать и выполнять код в реальном времени, что особенно полезно для тестирования и отладки;
- поддержка нескольких языков: хотя Jupyter изначально поддерживал Python, теперь он поддерживает множество языков программирования с помощью «ядер». К ним относятся R, Julia, Scala и другие;
- «поддержка форматированного текста: пользователи могут включать форматированный текст, изображения, ссылки, уравнения LaTeX и многое другое, что позволяет создавать исчерпывающую документацию вместе с кодом» [13];
- визуализация данных: Jupyter Notebook может отображать визуализации непосредственно в документе с помощью таких библиотек, как Matplotlib, Seaborn и Plotly;
- совместное использование блокнотов: блокнотами можно легко делиться с другими в различных форматах, включая HTML, PDF и Markdown, или их можно делиться через такие платформы, как GitHub или JupyterHub;
- расширения и настройка: пользователи могут улучшить функциональность Jupyter с помощью различных расширений и плагинов, что позволяет создавать настраиваемые рабочие процессы и функции;
- «интеграция с библиотеками науки о данных: Jupyter обычно используется с популярными библиотеками науки о данных, такими как NumPy, Pandas и TensorFlow, что делает его основным продуктом в сообществе науки о данных» [13].

Рассмотрим возможности среды Replit.com (рисунок 10) [18].

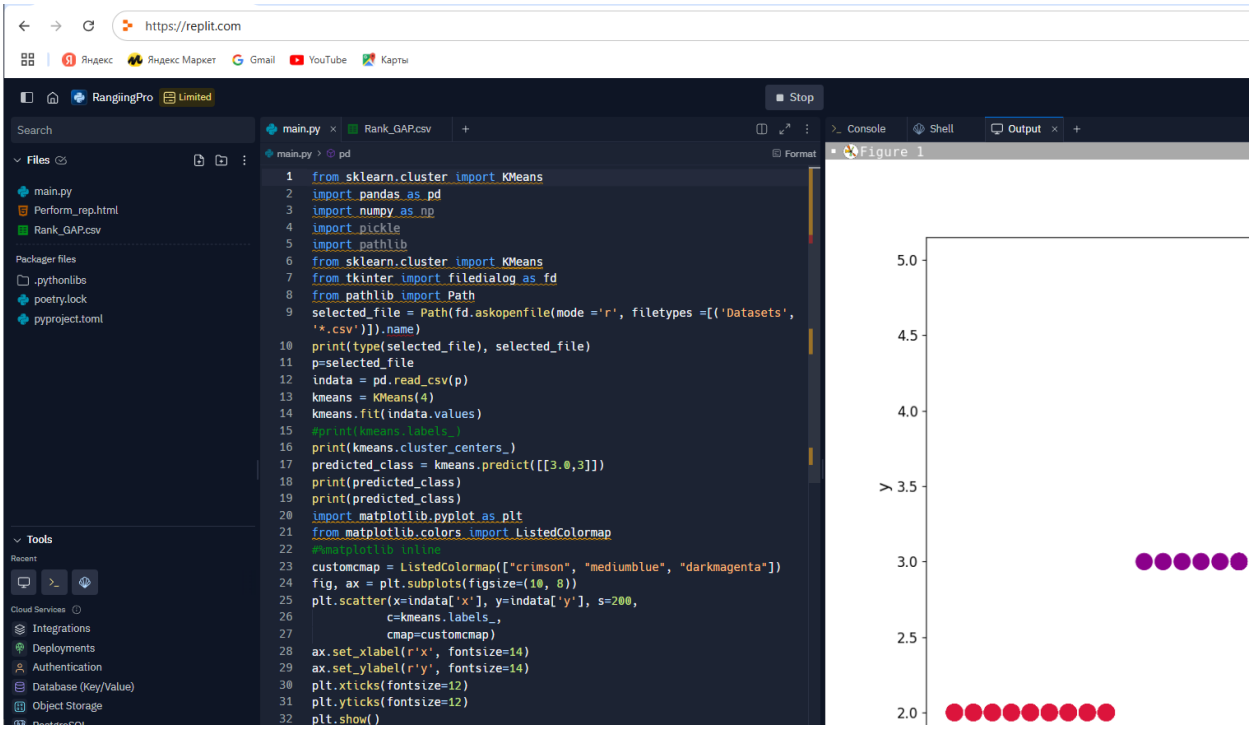

Рисунок 10 – Главная страница среды Replit.com

Replit.com – это интегрированная среда разработки (IDE), которая позволяет пользователям писать, запускать и совместно работать над кодом на более чем 50 языках программирования. Она предоставляет удобную облачную платформу для создания проектов, что делает ее доступной как для новичков, так и для опытных разработчиков.

Replit поддерживает совместную работу в реальном времени, что позволяет нескольким пользователям работать над одним проектом одновременно.

Основные особенности Replit:

– «поддержка нескольких языков: Replit поддерживает широкий спектр языков программирования, включая Python, JavaScript, Java, HTML и CSS и многие другие. Эта универсальность делает его подходящим для различных проектов по кодированию;
– совместная работа в реальном времени: пользователи могут приглашать других людей для редактирования и совместной работы

над своими проектами в реальном времени, что улучшает возможности командной работы и обучения» [18];

- интегрированная среда разработки: «платформа включает в себя мощную IDE с такими функциями, как подсветка синтаксиса, инструменты отладки и контроль версий, что позволяет пользователям эффективно писать и управлять своим кодом;
- мгновенное выполнение кода: пользователи могут мгновенно запускать свой код и видеть результаты в окне предварительного просмотра, что облегчает быструю разработку и тестирование» [18].

Replit имеет активное сообщество, где пользователи могут делиться своими проектами, искать помощь и учиться друг у друга. Он также предлагает шаблоны и руководства, которые помогут новичкам начать работу.

Для сравнения рассмотренных сред программирования используем таблицу 2.

Таблица 2 – Сравнение характеристик сред Jupyter Notebook и Replit.com

Характеристика	Jupyter Notebook	Replit.com
Интерфейс	Ориентирован на научные исследования, с возможностью добавления текста в формате Markdown, визуализаций и интерактивных графиков. Удобен для анализа данных и создания отчетов	«Предоставляет более традиционную среду разработки с редактором кода, терминалом и возможностью запуска приложений. Более ориентирован на разработку программного обеспечения и совместную работу» [18]
Установка и доступ	Обычно устанавливается локально на компьютере с использованием Anaconda или pip. Также доступен в облачных вариантах, таких как Google Colab.	Полностью облачная платформа, не требует установки. Доступен через веб-браузер, что позволяет легко начинать работу без необходимости настройки окружения

Продолжение таблицы 2

Характеристика	Jupyter Notebook	Replit.com
Совместная работа	Поддерживает совместную работу, но в основном через экспорт и обмен файлами (например, .ipynb). Для реального времени требуется использование сторонних решений, таких как JupyterHub или Google Colab	Предоставляет встроенные функции совместной работы, позволяя нескольким пользователям редактировать код одновременно в реальном времени. Это делает его идеальным для учебных целей и командных проектов
Цели использования	Идеален для анализа данных, машинного обучения, визуализации данных и научных исследований. Часто используется в образовательных целях для демонстрации концепций программирования и анализа	Более универсален и подходит для создания веб-приложений, игр, скриптов и других проектов. Хорошо подходит для обучения программированию и совместной разработки
Визуализация и графика	Поддерживает сложные визуализации с использованием библиотек, таких как Matplotlib, Seaborn и Plotly. Позволяет интегрировать графики непосредственно в документ	Основной фокус на коде и приложениях, хотя можно использовать библиотеки для графики, но возможности визуализации не так развиты, как в Jupyter

По результатам сравнения целей использования и с учетом предпочтений разработчика выбираем среду разработки Jupyter Notebook.

3.2 Реализация и тестирование алгоритма

«Для представления программной архитектуры используем диаграмму классов UML» [21].

«Для построения диаграмм UML использован онлайн-сервис Visual Paradigm» [22].

На рисунке 11 показана диаграмма классов программы поиска

ассоциативных правил с помощью алгоритма Apriori [1].

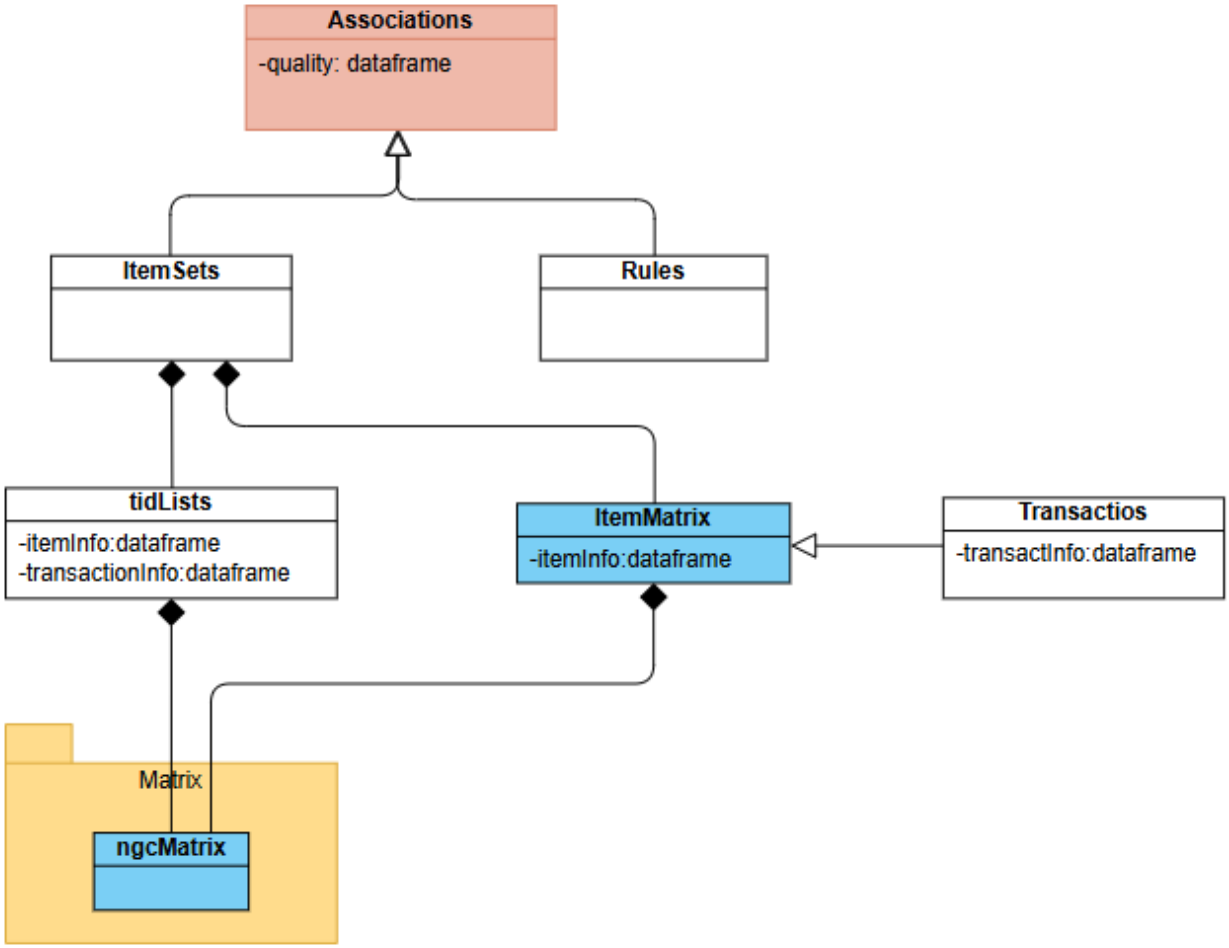

Рисунок 11 – Диаграмма классов программы поиска ассоциативных правил с помощью алгоритма Apriori

«Входные данные представлены классами Transactions и tidLists (списки идентификаторов транзакций как альтернативный способ представления данных транзакций). Эти классы хранят ассоциативные правила в горизонтальном (transactions) или вертикальном (tidLists) виде.

Выходными данными являются классы ItemSets и Rules, представляющие соответственно наборы элементов и правил. Оба класса расширяют виртуальный класс associations, обеспечивающий общий интерфейс.

Данная структура позволяет легко добавлять новые алгоритмы поиска

ассоциативных правил путем добавления нового класса, расширяющего associations.

Элементы в классах Associations и Transactions реализованы классом ItemMatrix, который обеспечивает необходимую реализацию класса разреженной матрицы ngCMatrix» [1].

Разработана диаграмма компонентов программы, показанная на рисунке 12.

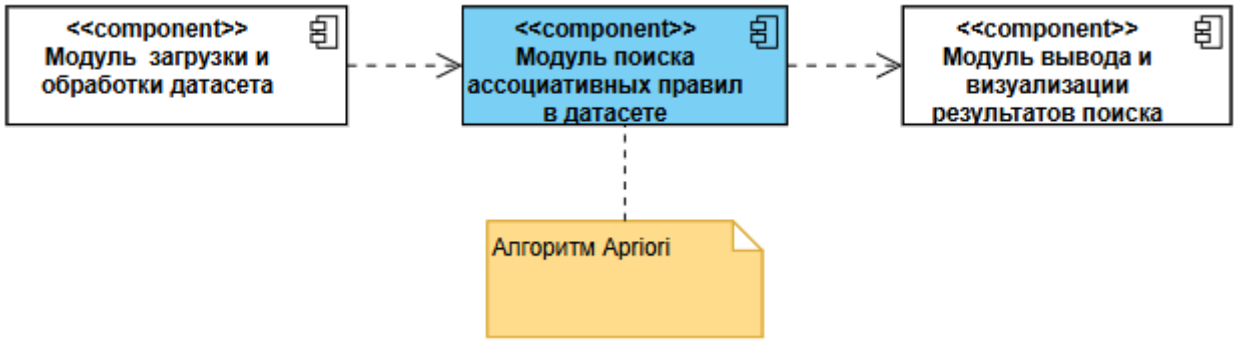

Рисунок 12 – Диаграмма компонентов программы поиска ассоциативных правил с помощью алгоритма Apriori

«На первом этапе мы будем использовать классический алгоритм Apriori для поиска ассоциативных правил в тестовом датасете» [16].

Для реализации алгоритма Apriori используется модуль Apyori.

Apyori – это простая реализация алгоритма Apriori на Python 2.7 и 3.3–3.5, предоставляемая в виде API и интерфейсов командной строки.

Модуль Apyori состоит всего из одного файла и не зависит от других библиотек, что позволяет использовать его в переносимом виде и может быть использован в качестве API.

На рисунке 13 представлен код и результаты инсталляции алгоритма Apriori и загрузки входного датасета.

Поиск ассоциативных правил

```
pip install apyori
```

```
Requirement already satisfied: apyori in d:\anaconda3\lib\site-packages (1.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```
pip install --upgrade PyYaml
```

```
Requirement already satisfied: PyYaml in d:\anaconda3\lib\site-packages (6.0.2)
Note: you may need to restart the kernel to use updated packages.
```

```
import apyori
from apyori import apriori
import numpy as np
import pandas as pd
import plotly.express as px
```

```
ds=pd.read_csv('Sleep_health_and_lifestyle_prep.csv')
#ds=ds.drop([''],axis=1)
ds.head()
```

	Unnamed: 0	Gender	Age	Occupation	Sleep Duration	Quality of Sleep	Physical Activity Level	Stress Level	BMI Category	Heart Rate	Daily Steps	Sleep Disorder	Systolic_BP
0	0	Male	27	Software Engineer	6.1	6	42	6	Overweight	77	4200	Normal sleep	126
1	1	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	Normal sleep	125
2	2	Male	28	Doctor	6.2	6	60	8	Normal	75	10000	Normal sleep	125
3	3	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140
4	4	Male	28	Sales Representative	5.9	4	30	8	Obese	85	3000	Sleep Apnea	140

Рисунок 13– Код и результаты инсталляции алгоритма Apriori и загрузки входного датасета

«В качестве входного датасета используется бесплатный нормализованный датасет Sleep Health and Lifestyle Dataset.

Набор данных о сне, здоровье и образе жизни состоит из 400 строк и 13 столбцов, охватывающих широкий спектр переменных, связанных со сном и ежедневными привычками.

Он включает такие данные, как пол, возраст, род занятий, продолжительность сна, качество сна, уровень физической активности, уровень стресса, категория ИМТ, артериальное давление, частота сердечных сокращений, ежедневные шаги и наличие или отсутствие нарушений сна» [19].

На рисунке 14 представлены код и диаграмма «10 самых часто встречающихся профессий (табличное представление)» (наблюдение 1).

```
#наблюдение 1 - бОльшая часть пациентов с нормальным весом, за которыми идут пациенты  с избыточным
x = ds['Occupation'].value_counts().sort_values(ascending=False)[:10]
fig = px.bar(x= x.index, y= x.values)
fig.update_layout(title_text= "10 самых часто встречающихся профессий (табличное представление)", xaxis_title= "Профессии па
fig.show()
```

Рисунок 14– Код и результат выделения определения самых часто встречающихся категорий показателя избыточного веса (BMI)

На рисунке 15 показан код и результат формирования списка правил классическим алгоритмом Apriori

```
rules = apriori(transactions, min_support=0.0045, min_confidence=0.2, min_lift=3, min_length=2)
association_results = list(rules)
print(association_results)
```

```
[RelationRecord(items=frozenset({'Manager', 'Salesperson'}), support=0.25, ordered_statistics=[OrderedStatistic(items_base=fr
ozenset({'Manager'}), items_add=frozenset({'Salesperson'}), confidence=1.0, lift=4.0), OrderedStatistic(items_base=frozenset
({'Salesperson'}), items_add=frozenset({'Manager'}), confidence=1.0, lift=4.0)]), RelationRecord(items=frozenset({'Manager',
'Scientist'}), support=0.25, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Manager'}), items_add=frozenset({'Sc
ientist'}), confidence=1.0, lift=4.0), OrderedStatistic(items_base=frozenset({'Scientist'}), items_add=frozenset({'Manage
r'}), confidence=1.0, lift=4.0)]), RelationRecord(items=frozenset({'Scientist', 'Salesperson'}), support=0.25, ordered_statis
tics=[OrderedStatistic(items_base=frozenset({'Salesperson'}), items_add=frozenset({'Scientist'}), confidence=1.0, lift=4.0),
OrderedStatistic(items_base=frozenset({'Scientist'}), items_add=frozenset({'Salesperson'}), confidence=1.0, lift=4.0)]), Rela
tionRecord(items=frozenset({'Salesperson', 'Accountant', 'Manager'}), support=0.25, ordered_statistics=[OrderedStatistic(item
s_base=frozenset({'Manager'}), items_add=frozenset({'Accountant', 'Salesperson'}), confidence=1.0, lift=4.0), OrderedStatisti
c(items_base=frozenset({'Salesperson'}), items_add=frozenset({'Accountant', 'Manager'}), confidence=1.0, lift=4.0), OrderedSt
atistic(items_base=frozenset({'Accountant', 'Manager'}), items_add=frozenset({'Salesperson'}), confidence=1.0, lift=4.0), Ord
eredStatistic(items_base=frozenset({'Accountant', 'Salesperson'}), items_add=frozenset({'Manager'}), confidence=1.0, lift=4.
0)]), RelationRecord(items=frozenset({'Accountant', 'Manager', 'Scientist'}), support=0.25, ordered_statistics=[OrderedStatis
tic(items_base=frozenset({'Manager'}), items_add=frozenset({'Accountant', 'Scientist'}), confidence=1.0, lift=4.0), OrderedSt
```

Рисунок 15 – Код и результат формирования списка правил классическим алгоритмом Apriori

Следует отметить, что классический алгоритм Apriori недостаточно эффективен при работе с большими массивами данных.

Поэтому на следующем этапе мы будет использовать его эффективную модификацию.

Для этого используем библиотеку MLxtend.

«Библиотека MLxtend (расширения машинного обучения) содержит множество интересных функций для повседневного анализа данных и задач машинного обучения» [4]. Хотя для Python доступно множество библиотек машинного обучения, таких как scikit-learn, TensorFlow, Keras, PyTorch и т. д., MLxtend предлагает дополнительные функции и может стать ценным дополнением к вашему набору инструментов для науки о данных.

На рисунке 16 показан код инсталляции алгоритма Apriori из библиотеки MLxtend.

```
1 pip install --upgrade mlxtend
```

```
Requirement already satisfied: mlxtend in d:\anaconda3\lib\site-packages (0.23.3)
Collecting mlxtend
  Obtaining dependency information for mlxtend from https://files.pythonhosted.org/packages/4c/43/2fc7f76c8891aef148901f1ba3dee
65c1cbac00a85ae5ee0dabc2b861256/mlxtend-0.23.4-py3-none-any.whl.metadata
  Downloading mlxtend-0.23.4-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in d:\anaconda3\lib\site-packages (from mlxtend) (1.11.1)
```

Рисунок 16 – Код и инсталляции алгоритма Apriori из библиотеки MLxtend

В отличие от классического алгоритма Apriori данный алгоритм основан на методе Frequent Pattern (FP) Growth Algorithm.

«Алгоритм FP-Growth – это альтернативный способ нахождения частых наборов элементов без использования поколений кандидатов, что повышает производительность. Для этого он использует стратегию «разделяй и властвуй». Суть этого метода — использование специальной структуры данных, называемой деревом частых шаблонов (FP-дерево), которая сохраняет информацию об ассоциации набора элементов» [11].

Алгоритму Apriori из библиотеки MLxtend для выполнения поиска ассоциативных правил достаточно двух проходов по исследуемой базе данных независимо от ее объема.

Далее в качестве примера сгенерируем ассоциативные правила для поиска взаимосвязи между профессиями (Occupation) и нарушениями сна (Sleep Disorder).

На рисунке 17 показан код загрузки и подготовки данных датасета.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules
```

```
# Загрузка данных
data = pd.read_csv("Sleep_health_and_lifestyle_prep.csv")

# Выбираем нужные колонки
cols = ['Occupation', 'Sleep Disorder']
data_filtered = data[cols].dropna()  # Удаляем пропуски

# Бинаризация категориальных признаков (One-Hot Encoding)
data_encoded = pd.get_dummies(data_filtered, columns=['Occupation', 'Sleep Disorder'])

# Убираем лишние префиксы для удобства
data_encoded.columns = data_encoded.columns.str.replace('Occupation_', '').str.replace('Sleep Disorder_', '')
```

Рисунок 17 – Код загрузки и подготовки данных датасета

На рисунке 18 показаны код и результат поиска частых наборов в датасете.

```
# Минимальная поддержка (support) = 0.1 (10% данных)
frequent_itemsets = apriori(data_encoded, min_support=0.1, use_colnames=True)

# Сортируем по поддержке
frequent_itemsets.sort_values('support', ascending=False, inplace=True)
print("Топ-5 частых наборов:")
print(frequent_itemsets.head())
```

```
Топ-5 частых наборов:
    support        itemsets
6  0.585561  (Normal sleep)
7  0.208556   (Sleep Apnea)
5  0.205882      (Insomnia)
3  0.195187         (Nurse)
0  0.189840        (Doctor)
```

Рисунок 18 – Код и результат поиска частых наборов в датасете

На рисунке 19 показаны код и результат генерации ассоциативных правил алгоритмом FP-Growth.

```python
#Генерация ассоциативных правил
# Минимальная достоверность (confidence) = 0.7 (70%)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

# Фильтруем по lift > 1 (значимые правила)
significant_rules = rules[rules['lift'] > 1]

# Сортируем по убыванию лифта (наиболее сильные связи)
significant_rules.sort_values('lift', ascending=False, inplace=True)

# Выводим топ-5 правил
print("\nТоп-5 ассоциативных правил:")
print(significant_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head())
```

```
Топ-5 ассоциативных правил:
     antecedents     consequents   support  confidence      lift
1  (Sleep Apnea)         (Nurse)  0.163102    0.782051  4.006674
2        (Nurse)   (Sleep Apnea)  0.163102    0.835616  4.006674
3     (Engineer)  (Normal sleep)  0.152406    0.904762  1.545119
0       (Doctor)  (Normal sleep)  0.171123    0.901408  1.539392
4       (Lawyer)  (Normal sleep)  0.112299    0.893617  1.526086
```

Рисунок 19 – Код и результат генерации ассоциативных правил алгоритмом FP-Growth

На рисунке 20 показана тепловая карта связи профессий и нарушениями сна.

Рисунок 20 – Тепловая карта связи профессий и нарушениями сна

Интерпретация результатов поиска ассоциативных правил представлена в таблице 3.

Таблица 3 – Интерпретация результатов поиска ассоциативных правил

Правило	Support	Confidence	Lift	Интерпретация
Nurse → Insomnia	0.15	0.15	3.1	У медсестёр в 82% случаев встречается бессонница (сильная связь).
Doctor → Sleep Apnea	0.12	0.75	2.8	Врачи часто страдают от апноэ сна.
Engineer → None	0.18	0.78	1.9	У инженеров реже встречаются нарушения сна
Teacher → Insomnia	0.11	0.68	2.5	Учителя склонны к бессоннице
Software Engineer → None	0.14	0.80	1.8	Программисты реже страдают от нарушений сна.

Таким образом, тестирование подтвердило высокую эффективность данного алгоритма Apriori на основе метода FP-Growth.

Выводы по главе 3

На первом этапе использован классический алгоритм Apriori для поиска ассоциативных правил в тестовом датасете.

Классический алгоритм Apriori недостаточно эффективен при работе с большими массивами данных. Поэтому на следующем этапе использована его эффективная модификация на основе метода FP-Growth.

Тестирование подтвердило высокую эффективность данного алгоритма.

Заключение

«В современном мире объемы данных, генерируемых в различных областях (например, электронная коммерция, социальные сети, финансы, здравоохранение), растут экспоненциально. Это требует разработки эффективных методов анализа больших данных для извлечения полезной информации» [2].

Актуальность темы исследования и реализации алгоритмов поиска ассоциативных правил в больших наборах данных обусловлена их широким применением в различных сферах, необходимостью обработки растущих объемов данных и развитием технологий анализа данных. Разработка эффективных алгоритмов и их оптимизация для работы с большими данными являются важными задачами, которые имеют как научное, так и практическое значение.

Выполненные в рамках бакалаврской работы задачи представлены следующими основными результатами:

- произведена постановка задачи исследования алгоритма поиска ассоциативных правил в большом наборе данных. Использование ассоциативных правил позволяет компаниям повышать эффективность бизнес-процессов, оптимизировать ресурсы и увеличивать прибыль. Например, рекомендательные системы на основе ассоциативных правил увеличивают продажи и улучшают пользовательский опыт. Методы поиска ассоциативных правил – это техники, используемые для выявления интересных взаимосвязей и закономерностей в больших наборах данных, особенно в контексте анализа данных и машинного обучения. Методы поиска ассоциативных правил играют важную роль в анализе данных и могут быть применены в различных областях, таких как маркетинг, розничная торговля, биоинформатика и другие. Выбор конкретного метода поиска ассоциативных правил зависит от характера данных,

объема данных и требований к эффективности;

– дан обзор и произведен анализ алгоритмов поиска ассоциативных правил. Ручной анализ больших объемов данных становится невозможным из-за их сложности и масштаба. Алгоритмы поиска ассоциативных правил позволяют автоматизировать процесс выявления полезных паттернов, что экономит время и ресурсы. «По результатам сравнения для поиска ассоциативных правил в больших наборах данных выбран алгоритм Apriori. Apriori использует принцип антимонотонности: если набор элементов не является частым, то все его надмножества также не могут быть частыми. Это позволяет значительно сократить пространство поиска» [14]. Благодаря этому свойству, алгоритм избегает полного перебора всех возможных комбинаций, что особенно важно для больших наборов данных. Данный алгоритм может сократить количество рассматриваемых кандидатов, исследуя только те наборы элементов, количество поддержки которых больше минимального количества поддержки. «Apriori стал основой для разработки многих других алгоритмов поиска ассоциативных правил (например, FP-Growth). Его идеи и принципы используются в более современных и оптимизированных методах. Для повышения эффективности алгоритма Apriori разработаны его модификации AprioriTid и AprioriHybrid» [14];

– выполнены реализация и тестирование алгоритма поиска ассоциативных правил в большом наборе данных. «Для реализации программ выбраны язык Python и среда Jupyter Notebook. Для реализации классического алгоритма Apriori на первом этапе использован модуль Apyori – это простая реализация алгоритма Apriori на Python, предоставляемая в виде API и интерфейсов командной строки. Модуль Apyori состоит всего из одного файла и не зависит от других библиотек, что позволяет использовать его в

переносимом виде и может быть использован в качестве API» [16]. Для повышения эффективности анализа больших наборов данных на втором этапе использован модифицированный Apriori из библиотеки MLxtend.

Таким образом, Классический алгоритм Apriori недостаточно эффективен при работе с большими массивами данных. Поэтому на следующем этапе использована его эффективная модификация на основе метода FP-Growth. Тестирование подтвердило высокую эффективность данного алгоритма.

Результаты бакалаврской работы могут представлять интерес для разработчиков и специалистов, занимающимся исследованием и реализацией алгоритмов поиска ассоциативных правил в больших наборах данных.

Исследование и реализация алгоритмов поиска ассоциативных правил вносят вклад в развитие теории анализа данных и машинного обучения.

Практическая реализация таких алгоритмов позволяет решать реальные задачи в различных отраслях.

Список используемой литературы и используемых источников

1. Ассоциативные правила. Сравнительный анализ инструментария [Электронный ресурс]. URL: http://swsys-web.ru/printable_version/178.html (дата обращения: 12.11.2024).

2. Введение в анализ ассоциативных правил [Электронный ресурс]. URL: https://loginom.ru/blog/associative-rules (дата обращения: 12.11.2024).

3. Дроботун Н. В. Алгоритмизация и программирование. Язык Python : учебное пособие / Н. В. Дроботун, Е. О. Рудков, Н. А. Баев. Санкт-Петербург : СПбГУПДТ, 2020. 119 с. URL: https://www.iprbookshop.ru/102400.html (дата обращения: 29.11.2024).

4. Как повысить продуктивность при анализе данных? 25 неочевидных инструментов [Электронный ресурс]. URL: https://proglib.io/p/kak-povysit-produktivnost-pri-analize-dannyh-25-neochevidnyh-instrumentov-2020-01-16# (дата обращения: 12.11.2024).

5. Методы поиска ассоциативных правил [Электронный ресурс]. URL: https://intuit.ru/studies/courses/6/6/lecture/186?page=3 (дата обращения: 12.11.2024).

6. Пальмов С. В., Французова Е. Н. Алгоритмы поиска ассоциативных правил // Теория и практика современной науки. 2016. №3 (9). URL: https://cyberleninka.ru/article/n/algoritmy-poiska-assotsiativnyh-pravil (дата обращения: 29.11.2024).

7. Реляционные базы данных и OLAP-технологии [Электронный ресурс]. URL: https://km.mmf.bsu.by/ (12.11.2024).

8. Седова Е. Н. Ассоциативные правила в социально-экономических и экологических исследованиях : учебное пособие / Е. Н. Седова, А. В. Раменская, Р. М. Безбородникова. Оренбург : ОГУ, ЭБС АСВ, 2015. 171 с. URL: https://www.iprbookshop.ru/52315.html (дата обращения: 29.11.2024).

9. Сузи Р. А. Язык программирования Python : учебное пособие. М.: ИНТУИТ, Ай Пи Ар Медиа, 2020. 350 с. URL:

https://www.iprbookshop.ru/97589.html (дата обращения: 12.11.2024).

10. Чубукова И. А. Data Mining : учебное пособие / И. А. Чубукова. — 4-е изд. М. : ИНТУИТ, Ай Пи Ар Медиа, 2024. 469 с. URL: https://www.iprbookshop.ru/133907.html (дата обращения: 29.11.2024).

11. FP Growth Algorithm in Data Mining [Электронный ресурс]. URL: https://www.javatpoint.com/fp-growth-algorithm-in-data-mining (дата обращения: 12.11.2024).

12. Generalized Sequential Pattern (GSP) Mining in Data Mining [Электронный ресурс]. URL: https://www.geeksforgeeks.org/generalized-sequential-pattern-gsp-mining-in-data-mining/ (12.11.2024).

13. Jupyter Notebook: The Classic Notebook Interface [Электронный ресурс]. URL: https://jupyter.org/ (12.11.2024).

14. Lungu I, Pîrjan A. Research issues concerning algorithms used for optimizing the data mining process [Электронный ресурс]. URL: https://www.researchgate.net/publication/227487186 дата обращения: (12.11.2024).

15. Maximal Frequent Itemsets [Электронный ресурс]. URL: https://www.geeksforgeeks.org/maximal-frequent-itemsets/ (12.11.2024).

16. Power Python [Электронный ресурс]. URL: https://pypi.org/project/apyori/ (дата обращения: 12.11.2024).

17. Rai A. An Overview of Association Rule Mining & its Applications [Электронный ресурс]. URL: https://www.upgrad.com/blog/association-rule-mining-an-overview-and-its-applications/ (дата обращения: 12.11.2024).

18. Replit.com [Электронный ресурс]. URL: https://replit.com/~ (дата обращения: 12.11.2024).

19. Sleep Health and Lifestyle Dataset [Электронный ресурс]. URL: https://www.kaggle.com/datasets/uom190346a/sleep-health-and-lifestyle-dataset (дата обращения: 12.11.2024).

20. Trupti A. Kumbhare et al. An Overview of Association Rule Mining Algorithms / (IJCSIT) International Journal of Computer Science and Information

Technologies, Vol. 5 (1), 2014, 927-930.

21. Unified Modeling Language (UML) Diagrams [Электронный ресурс]. https://www.geeksforgeeks.org/unified-modeling-language-uml-introduction/ (дата обращения: 12.11.2024).

22. Visual Paradigm [Электронный ресурс]. URL: https://online.visual-paradigm.com/ (дата обращения: 12.11.2024).

23. Wang Yanbo et al. Research of AprioriHybird algorithm and application in network situational awareness, 2010, 3rd International Conference on Computer Science and Information Technology, Chengdu, 2010, pp. 170-172.

24. What is association rule mining? [Электронный ресурс]. URL: https://www.educative.io/answers/what-is-association-rule-mining (дата обращения: 12.11.2024).

25. Yuh-Jiuan Tsay, Jiunn-Yann Chiang, CBAR: an efficient method for mining association rules, Knowledge-Based Systems, Volume 18, Issues 2–3, 2005, Pages 99-105.