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

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет по лабораторной работе №1
по курсу
”Проектирование Рекомендательных Систем”

Тема Сравнение алгоритмов поиска ассоциативных правил

Студент Якуба Д. В.

Группа ИУ7-33М

Оценка (баллы) _____

Преподаватели Быстрицкая А.Ю.

Москва — 2023 г.

Оглавление

ВВЕДЕНИЕ

Цель работы – сравнение алгоритмов поиска ассоциативных правил Apriori, ECLAT и FP-Growth.

Для достижения поставленной цели потребуется:

- привести описание алгоритмов Apriori, ECLAT и FP-Growth;
- привести описание используемых для исследования данных;
- провести сравнение алгоритмов по времени работы и затратам по памяти.

1. Аналитический раздел

1.1 Задача поиска ассоциативных правил

Правило ассоциации состоит из двух частей, предшествующей и последующей. Предшествующая задача – это элемент, находящийся в данных. А последующая – это элемент или множество элементов, которые встречаются в сочетании с предшествующей задачей. [1]

В интеллектуальном анализе данных правила ассоциации являются полезными и помогают спрогнозировать поведение клиента.

Для оценки качества полученных рекомендаций используются следующие метрики [1]:

- Поддержка – позволяет узнать, в какой части покупательских корзин содержатся все элементы того или иного ассоциативного правила. Определяется как $support(A \to B) = P(A \cup B)$
- Достоверность – показывает, насколько хорошим является правило для предсказания правой части,когда условие слева верно. Определяется как $confidence(A \to B) = \frac{P(A \cup B)}{P(A)}$
- Интерес – измеряет силу правила, сравнивая полное правило с предположенной правой частью и рассчитывается, как отношение достоверности правила к частоте появления следствия – $lift(A \to B) = \frac{P(A \cup B)}{P(A)P(B)}$

1.2 Apriori

Данный алгоритм основан на поиске в ширину, в котором свойство того, что с ростом набора поддержка монотонно убывает, позволяет уменьшить объем вычислений.

Принцип работы алгоритма [1]:

1. **Генерация кандидатов** – алгоритм начинается с создания набора всех возможных одиночных элементов и определения их частоты в данных. Данные элементы называются “кандидатами”;
2. **Поиск подмножеств** – далее следует генерация кандидатов более высокого уровня, используя информацию о частоте 1-элементных наборов. Создаются новые наборы элементов, добавляя один элемент к уже существующим кандидатам, которые являются кандидатами следующего уровня;
3. **Оценка поддержки** – для каждого кандидата подсчитывается частота

его появления в транзакциях. Если частота кандидата превышает заданный порог поддержки, то он считается частым и переходит на следующий уровень, иначе – отбрасывается;

4. **Сбор ассоциативных правил** – после завершения генерации кандидатов, с использованием частых наборов элементов создаются ассоциативные правила. Для каждого частого набора элементов создаются все возможные комбинации элементов внутри набора для нахождения ассоциативных правил;
5. **Оценка уверенности** – н данном этапе для каждого ассоциативного правила вычисляется уровень уверенности. Ассоциативные правила с уверенностью выше определенного порога считаются интересными.

1.3 ECLAT

Данный алгоритм, в отличие от Apriori, работает на основе более эффективного и компактного представления данных.

Принцип работы алгоритма [1]:

1. **Создание вертикальной структуры данных** – в отличие от Apriori, который работает с горизонтальной структурой данных, ECLAT использует вертикальную структуру данных. Это означает, что для каждого элемента данных создается список транзакций, в которых этот элемент присутствует;
2. **Рекурсивный поиск** – ход алгоритма начинается с 1-элементных наборов и проверяется, сколько транзакций содержит каждый элемент. Элементы, удовлетворяющие минимальному порогу поддержки считаются частыми наборами;
3. **Объединение наборов** – далее следует объединение частых 1-элементных наборов, чтобы создать более крупные наборы элементов. Это происходит путем пересечения вертикальных файлов элементов, которые входят в эти наборы. При этом также проверяется, удовлетворяют ли полученные наборы минимальному порогу поддержки;
4. **Рекурсивное продолжение** – затем рекурсивно продолжают создаваться все большие наборы элементов до тех пор, пока не будет достигнут максимальный размер набора или не будут удовлетворены пороги поддержки;
5. **Сбор ассоциативных правил** – после того, как все частые наборы эле-

ментов созданы, ECLAT может быть использован для извлечения ассоциативных правил, аналогично Apriori, причем ассоциативные правила определяются на основе уверенности.

1.4 FP-Growth

Данный алгоритм представляет собой эффективный и масштабируемый способ нахождения частых наборов элементов, используя структуру данных, называемую FP-деревом.

FP-дерево – компактная и эффективная структура данных, представляющая собой древовидную структуру, где каждый узел представляет элемент данных, а ребра между узлами – это связи между элементами в транзакциях. Каждый путь от корня до листа в дереве представляет одну из транзакций из исходных данных, а счетчики на узлах отражают частоту встречаемости элементов. [2]

Принцип работы алгоритма [2]:

1. **Построение FP-дерева**:
   - Подсчет частоты встречаемости каждого элемента в транзакциях и сортировка элементов по убыванию частоты, таким образом более частые элементы находятся ближе к корню дерева;
   - Создание корневого узла дерева;
   - Для каждой транзакции создается путь в дереве, начиная с корневого узла и добавляя элементы транзакции по мере прохождения по дереву. Если элемент уже существует на пути, увеличивается счетчик этого элемента. Если элемент отсутствует – он добавляется как новый узел в дереве;
2. **Создание условных FP-деревьев**: для каждого элемента, начиная с самого частого, строится условное дерево. Данное дерево создается путем удаления всех путей в FP-дереве, которые не содержат данный элемент, а затем обновления счетчиков элементов на оставшихся путях;
3. **Рекурсивный поиск частых наборов**: для каждого условного дерева рекурсивно находятся все частые наборы элементов, начиная с элементов, которые находятся ближе к корню дерева. Это позволяет извлечь частые наборы элементов, учитывая их иерархию в FP-дереве;
4. **Сбор ассоциативных правил**: после того, как все частые наборы элементов найдены, ассоциативные правила определяются на основе уве-

ренности.

2. Конструкторский раздел

В данном разделе описаны данные, анализируемые в данной работе.

2.1 Market Basket Optimisation

В качестве источника данных был взят датасет, располагающийся в свободном доступе на веб-сайте Kaggle [3]. Набор данных включает в себя корзины потребителя некоторого продуктового магазина. В качестве предобработки была построена база данных транзакций, которая структурно изменялась по требованию входных данных используемых алгоритмов.

3. Технологический раздел

В данном разделе описываются средства разработки программного обеспечения.

3.1 Средства реализации

В качестве используемого был выбран язык программирования Python [4].

Данный выбор обусловлен следующими факторами:

- Большое количество исчерпывающей документации;
- Широкий выбор доступных библиотек для разработки;
- Простота синтаксиса языка и высокая скорость разработки.

При написании программного продукта использовалась среда разработки Visual Studio Code. Данный выбор обусловлен тем, что данная среда распространяется по свободной лицензии, поставляется для конечного пользователя с открытым исходным кодом, а также имеет большое число расширений, ускоряющих разработку.

3.2 Библиотеки

При анализе и обработке датасета, а также для решения поставленных задач использовались библиотеки:

- pandas;
- numpy;
- matplotlib;
- apyory [5];
- pyECLAT [6];
- fpgrowth-py [7].

Данные библиотеки позволили полностью покрыть спектр потребностей при выполнении работы.

4. Исследовательский раздел

4.1 Условия исследований

Исследование проводилось на персональном вычислительной машине со следующими характеристиками:

- процессор Apple M1 Pro,
- операционная система Ventura 13.5.2,
- 32 Гб оперативной памяти.

Временные затраты определялись с использованием библиотеки time. Затраты по памяти определялись с использованием библиотеки memory$_profiler$.

В данном исследовании значение параметра минимальной поддержки изменялось от значения 0.01 до 0.5 с шагом 0.01. Значение минимального и максимального количества элементов было равно 2.

4.2 Зависимость времени исполнения от параметра минимальной поддержки

На рисунке 4.1 представлен график зависимости времени исполнения алгоритмов от заданного параметра минимальной поддержки.

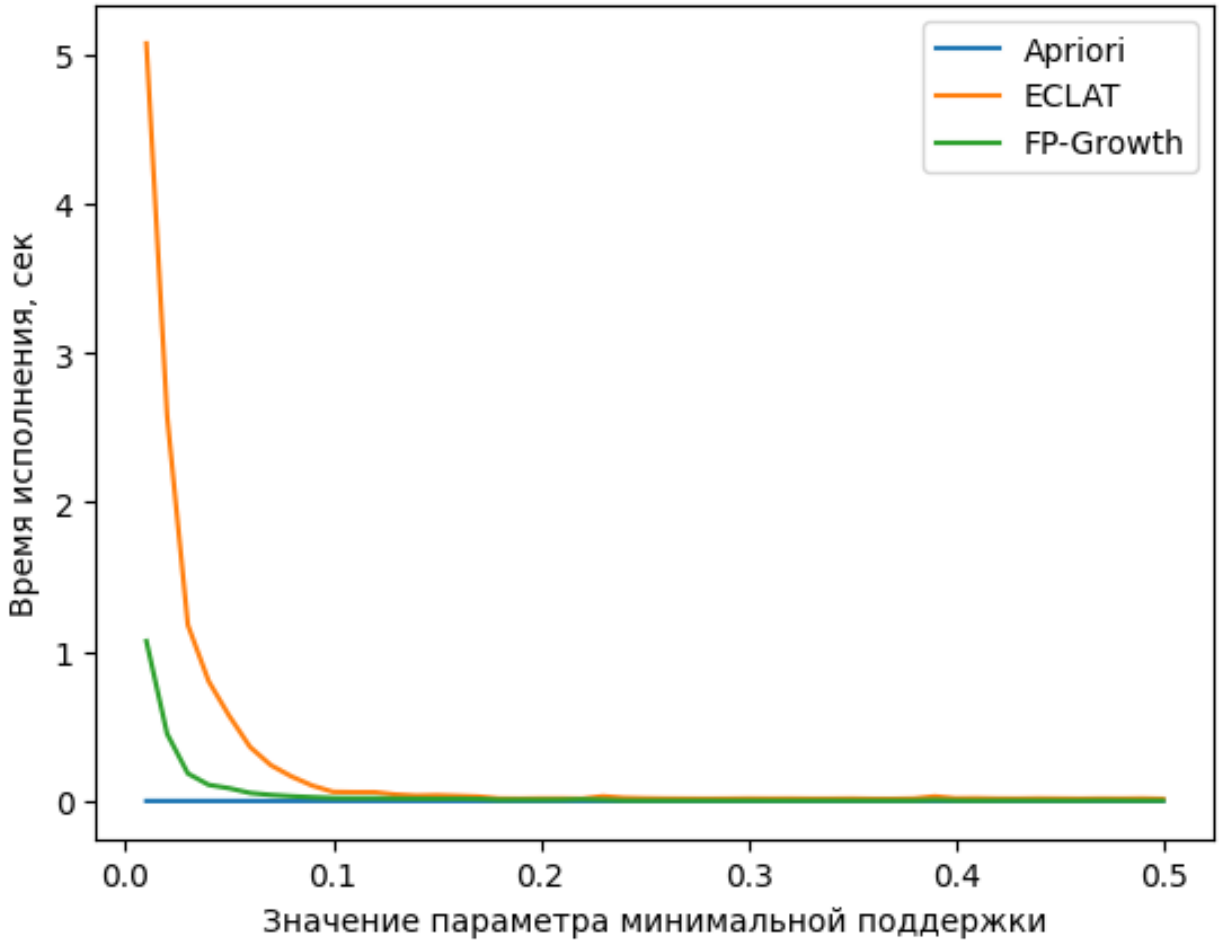

Рис. 4.1: График зависимости времени исполнения от заданного параметра минимальной поддержки.

4.3 Зависимость количества занятой памяти процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки

На рисунке 4.2 представлен график зависимости количества занятой памяти процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки.

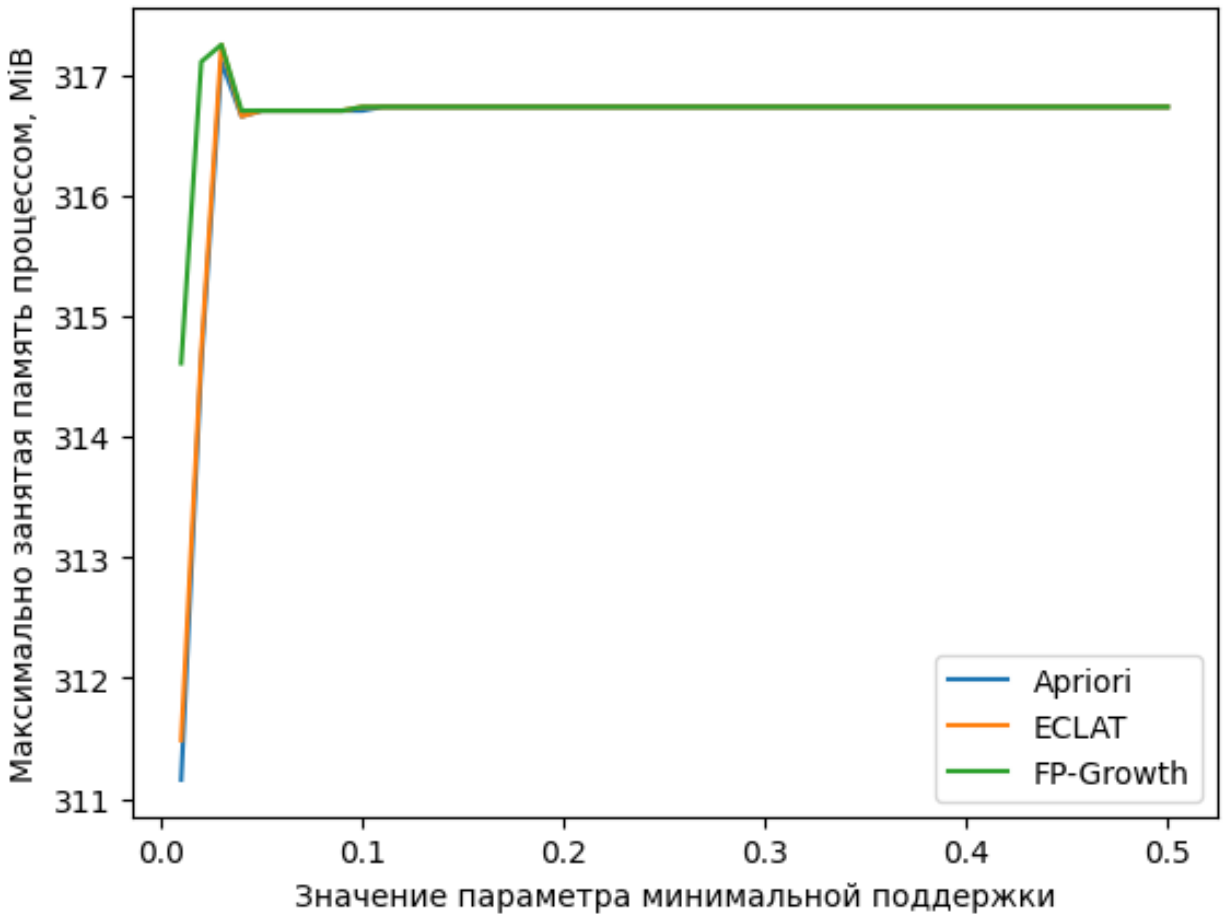

Рис. 4.2: График зависимости количества занятой процессом во время исполнения алгоритмов от заданного параметра минимальной поддержки.

Заключение

В результате проведенных исследований заметно, что по времени исполнения на меньших значениях параметра минимальной поддержки самое долгое время исполнения у алгоритма ECLAT. Самым быстрым временем исполнения обладает алгоритм Apriori. В дальнейшем, при увеличении параметра, разница во времени исполнения у трех алгоритмов становится уже не такой заметной и при значении параметра минимальной поддержки равным 0.15 уже практически отсутствует.

Также стоит отметить, что размер задействованной памяти, потребовавшейся для исполнения каждого из алгоритмов, практически одинаков. Отличия присутствуют лишь при начальных значениях величины параметра минимальной поддержки – в данном случае однозначно можно считать, что Apriori занимает на ≈ 4 мебибайта меньше алгоритма FP-Growth.

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы было проведено сравнение алгоритмов поиска ассоциативных правил Apriori, ECLAT и FP-Growth.

Исследования показали, что при заданных условиях отличия во времени исполнения между алгоритмами не разительны, однако на меньших значениях параметра минимальной поддержки наилучшим образом себя показал именно алгоритм Apriori.

Были решены следующие задачи:

- приведено описание алгоритмов Apriori, ECLAT и FP-Growth;
- приведено описание используемых для исследования данных;
- проведено сравнение алгоритмов по времени работы и затратам по памяти.

Список литературы

1. Анатольевич Олянич Игорь. Сравнение алгоритмов построения ассоциативных правил на основе набора данных покупательских транзакций // Известия Самарского научного центра РАН. 2018. № 6-2.

2. Jiawei Han Hong Cheng Dong Xi. Frequent pattern mining: current status and future directions // Режим доступа: https://sites.cs.ucsb.edu/~xyan/papers/dmkd07_frequentpattern.pdf (дата обращения 20.09.2023). 2006.

3. Kaggle Market Basket Optimisation Dataset [Электронный ресурс]. Режим доступа: https://www.kaggle.com/datasets/devchauhan1/market-basket-optimisationcsv (дата обращения 16.09.2023).

4. Python official page [Электронный ресурс]. Режим доступа: https://www.python.org/ (дата обращения 10.05.2023).

5. Apyory: official PyPl project page [Электронный ресурс]. Режим доступа: https://pypi.org/project/apyori/ (дата обращения 16.09.2023).

6. pyECLAT: official PyPl project page [Электронный ресурс]. Режим доступа: https://pypi.org/project/pyECLAT/ (дата обращения 16.09.2023).

7. fpgrowth-py: official PyPl project page [Электронный ресурс]. Режим доступа: https://pypi.org/project/fpgrowth-py/ (дата обращения 16.09.2023).