Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Южно-Уральский государственный университет»

(Национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

По лабораторной работе №1

|  |  |  |  |
| --- | --- | --- | --- |
| Поиск частых наборов | | | |
|  | | | |
|  |  | Руководитель: | |
|  |  | | \_\_\_\_\_\_\_\_\_\_\_\_\_М.Л. Цымблер  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.  Автор работы  студент группы КЭ-220  \_\_\_\_\_\_\_\_\_\_\_\_А.В.Витомсков  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г. |

Челябинск 2020

1. *Выполните поиск частых наборов объектов в трех различных наборах данных с помощью следующих алгоритмов (или их модификаций): Apriori, FP‑Growth, ECLAT. Наборы данных должны существенно отличаться друг от друга по количеству транзакций и/или типичной длине транзакции (количеству объектов). Варьируйте пороговое значение поддержки (например: 1%, 3%, 5%, 10%, 15%, 20%). Проверьте идентичность результатов, полученных с помощью различных алгоритмов.*

Для исследования мной были найдены 3 набора данных:

1) Набор данных о транзакциях UCI. Репозиторий машинного обучения UCI создал этот набор данных, содержащий фактические транзакции за 2010 и 2011 годы. Набор данных поддерживается на их сайте, где его можно найти под названием «Интернет-магазин». Ссылка на набор: <https://www.kaggle.com/carrie1/ecommerce-data> Набор данных содержит 25900 записей. Так он выглядит:

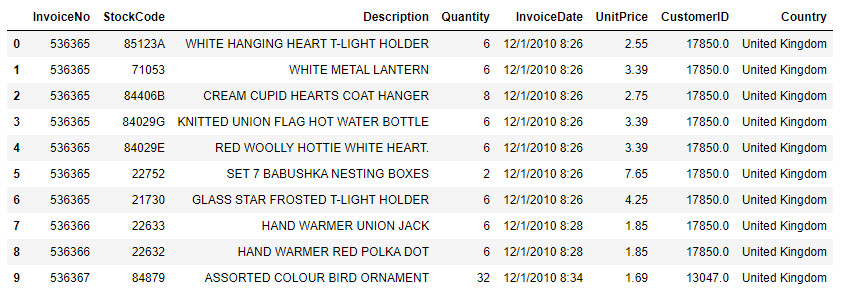


Рисунок 1 – Набор данных UCI

Нам понадобятся только данные столбцов InvoiceNO и StockCode.

2). Набор данных "market-basket-optimization". Набор данных рыночной корзины от roshansharma, широко используемый для проверки ARL алгоритмов. Ссылка на набор: <https://www.kaggle.com/roshansharma/market-basket-optimization> Набор данных содержит 9834 записи. Выглядит он так:

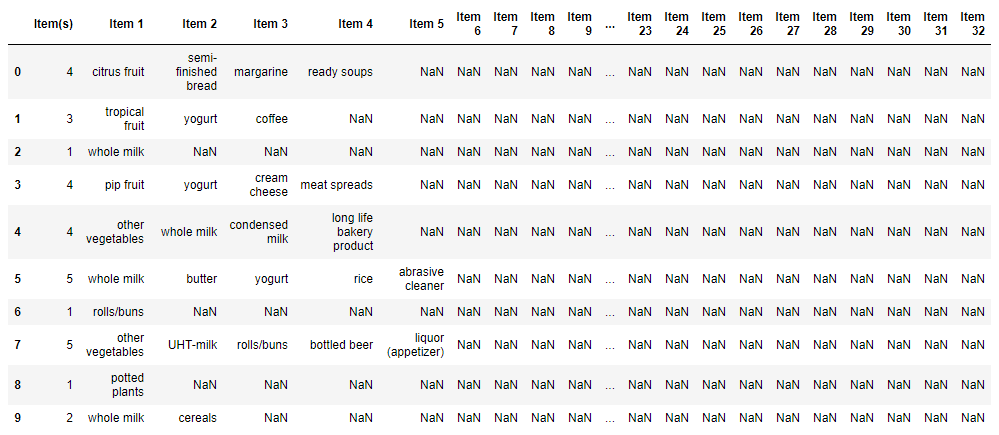


Рисунок 2 – Набор данных market-basket-optimization

3) Набор данных "Instacart Market Basket Analysis". Анализ корзины рынка Instacart. Ссылка на набор: <https://www.kaggle.com/psparks/instacart-market-basket-analysis/metadata> Анализировать мы будем только часть набора, которая содержит 131208 записей.

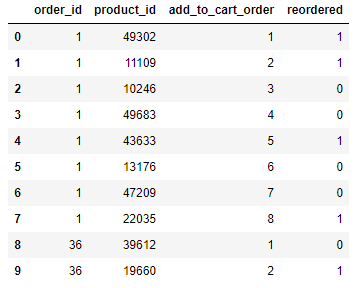


Рисунок 3 – Набор данных market-basket-optimization

Нам понадобятся только данные столбцов order\_id и product\_id.

Для начала приведем все наборы данных к единому виду: идентификаторы товаров заменим на натуральные числа; сгруппируем набор по номеру заказа. Ограничим количество товара в одном заказе числом 20. Все операции представлены в файле ARL\_Vitomskov\_1.ipynb. В итоге получим файлы в следующем виде:

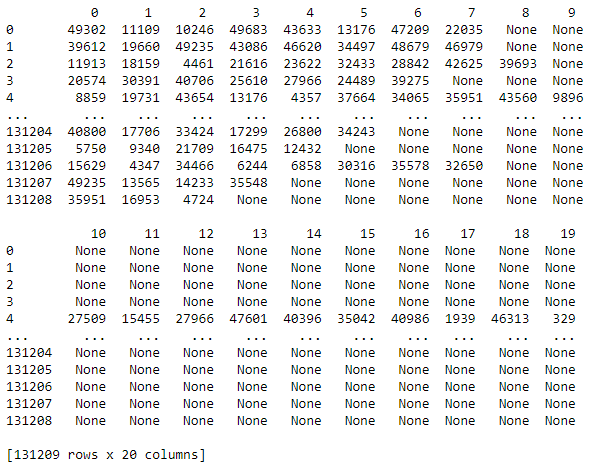


Рисунок 4 – Данные на входе алгоритмов

Для анализа я выбрал следующие реализации алгоритмов на языке Python:

|  |  |  |
| --- | --- | --- |
| Алгоритм | Библиотека | Функция |
| apriori | mlxtend | apriori\_execute |
| apyori | apyori\_execute |
| eclat | https://habr.com/ru/company/ods/blog/353502/ | eclat\_execute |
| fp-growth | mlxtend | fpgrowth\_execute |

Таблица 1 – Анализируемые алгоритмы

На вход все алгоритмы (за исключением алгоритмов из mlxtend) должны получать данные в различных форматах, оговоренных в документации на них. Поэтому на первом этапе в файле ARL\_Vitomskov\_2.ipynb формируются необходимые входные структуры. Сам запуск алгоритмов вынесен в отдельные функции (см. столбец Функция в таблице 1).

Для вычисления среднего времени выполнения, каждый алгоритм запускается 10 раз с одним и тем же значением min\_support. Значения min\_support берутся из списка [0.20, 0.15, 0.10, 0.05, 0.03, 0.01]. Время работы алгоритмов и прочая статистика сохраняется в списке tables. Выходные данные алгоритмов приводятся к единообразному виду и сохраняются в виде dataframe в списке reports.

|  |
| --- |
| Набор данных UCI\_dataset: сводная таблица времени выполнения  name support time rows\_count max\_len count\_different\_len  0 apriori\_execute 0.20 1.793529 0 NaN {}  1 apyori\_execute 0.20 0.000016 0 NaN {}  2 fpgrowth\_execute 0.20 4.658317 0 NaN {}  3 eclat\_execute 0.20 35.253209 0 NaN {}  4 apriori\_execute 0.15 1.816421 0 NaN {}  5 apyori\_execute 0.15 0.000016 0 NaN {}  6 fpgrowth\_execute 0.15 4.759325 0 NaN {}  7 eclat\_execute 0.15 34.846475 0 NaN {}  8 apriori\_execute 0.10 1.622675 0 NaN {}  9 apyori\_execute 0.10 0.000013 0 NaN {}  10 fpgrowth\_execute 0.10 4.505783 0 NaN {}  11 eclat\_execute 0.10 34.751094 3 2.0 {2: 3}  12 apriori\_execute 0.05 1.661577 2 1.0 {1: 2}  13 apyori\_execute 0.05 0.000014 2 1.0 {1: 2}  14 fpgrowth\_execute 0.05 4.614114 2 1.0 {1: 2}  15 eclat\_execute 0.05 33.205877 22 2.0 {2: 22}  16 apriori\_execute 0.03 2.174578 10 1.0 {1: 10}  17 apyori\_execute 0.03 0.000014 10 1.0 {1: 10}  18 fpgrowth\_execute 0.03 5.036532 10 1.0 {1: 10}  19 eclat\_execute 0.03 34.572456 45 2.0 {2: 45}  20 apriori\_execute 0.01 231.134767 236 3.0 {1: 215, 2: 20, 3: 1}  21 apyori\_execute 0.01 0.000017 236 3.0 {1: 215, 2: 20, 3: 1}  22 fpgrowth\_execute 0.01 10.434879 236 3.0 {1: 215, 2: 20, 3: 1}  23 eclat\_execute 0.01 36.465714 87 3.0 {2: 45, 3: 42} |

Таблица 2 – Результаты тестирования алгоритмов для UCI\_dataset

Здесь:

name – наименование алгоритма,

support – минимальное значение поддержки,

time – время выполнения 10 итераций,

rows\_count – количество частых наборов,

max\_len – наибольшее число элементов в наборе,

count\_different\_len – расписано количество частых наборов при разном числе элементов в наборе. В сумме даёт rows\_count.

|  |
| --- |
| Набор данных market-basket-optimization: сводная таблица времени выполнения  name support time rows\_count max\_len count\_different\_len  0 apriori\_execute 0.20 0.069779 1 1.0 {1: 1}  1 apyori\_execute 0.20 0.000014 1 1.0 {1: 1}  2 fpgrowth\_execute 0.20 0.904365 1 1.0 {1: 1}  3 eclat\_execute 0.20 11.331937 0 NaN {}  4 apriori\_execute 0.15 0.072403 4 1.0 {1: 4}  5 apyori\_execute 0.15 0.000014 4 1.0 {1: 4}  6 fpgrowth\_execute 0.15 0.925393 4 1.0 {1: 4}  7 eclat\_execute 0.15 11.528947 0 NaN {}  8 apriori\_execute 0.10 0.093123 8 1.0 {1: 8}  9 apyori\_execute 0.10 0.000014 8 1.0 {1: 8}  10 fpgrowth\_execute 0.10 0.997608 8 1.0 {1: 8}  11 eclat\_execute 0.10 11.658735 1 2.0 {2: 1}  12 apriori\_execute 0.05 0.391499 31 2.0 {1: 28, 2: 3}  13 apyori\_execute 0.05 0.000016 31 2.0 {1: 28, 2: 3}  14 fpgrowth\_execute 0.05 1.768760 31 2.0 {1: 28, 2: 3}  15 eclat\_execute 0.05 13.225267 4 2.0 {2: 4}  16 apriori\_execute 0.03 1.019785 63 2.0 {1: 44, 2: 19}  17 apyori\_execute 0.03 0.000017 63 2.0 {1: 44, 2: 19}  18 fpgrowth\_execute 0.03 2.080632 63 2.0 {1: 44, 2: 19}  19 eclat\_execute 0.03 11.739406 7 2.0 {2: 7}  20 apriori\_execute 0.01 5.729199 329 3.0 {2: 209, 1: 88, 3: 32}  21 apyori\_execute 0.01 0.000015 329 3.0 {2: 209, 1: 88, 3: 32}  22 fpgrowth\_execute 0.01 3.087898 329 3.0 {2: 209, 1: 88, 3: 32}  23 eclat\_execute 0.01 11.920142 16 2.0 {2: 16} |

Таблица 3 – Результаты тестирования алгоритмов для market-basket-optimization

|  |
| --- |
| Набор данных Instacart Market Basket Analysis: сводная таблица времени выполнения  name support time rows\_count max\_len count\_different\_len  0 apriori\_execute 0.20 76.873656 0 NaN {}  1 apyori\_execute 0.20 0.000016 0 NaN {}  2 fpgrowth\_execute 0.20 119.590685 0 NaN {}  3 eclat\_execute 0.20 166.955985 2 2.0 {2: 2}  4 apriori\_execute 0.15 75.755357 0 NaN {}  5 apyori\_execute 0.15 0.000020 0 NaN {}  6 fpgrowth\_execute 0.15 120.404326 0 NaN {}  7 eclat\_execute 0.15 171.520954 17 2.0 {2: 17}  8 apriori\_execute 0.10 79.861944 2 1.0 {1: 2}  9 apyori\_execute 0.10 0.000014 2 1.0 {1: 2}  10 fpgrowth\_execute 0.10 121.134198 2 1.0 {1: 2}  11 eclat\_execute 0.10 178.560065 34 2.0 {2: 34}  12 apriori\_execute 0.05 76.781623 7 1.0 {1: 7}  13 apyori\_execute 0.05 0.000014 7 1.0 {1: 7}  14 fpgrowth\_execute 0.05 120.728434 7 1.0 {1: 7}  15 eclat\_execute 0.05 186.119948 68 4.0 {2: 45, 3: 22, 4: 1}  16 apriori\_execute 0.03 85.561448 15 1.0 {1: 15}  17 apyori\_execute 0.03 0.000014 15 1.0 {1: 15}  18 fpgrowth\_execute 0.03 124.134246 15 1.0 {1: 15}  19 eclat\_execute 0.03 196.525663 131 4.0 {3: 84, 2: 45, 4: 2}  20 apriori\_execute 0.01 515.483933 109 2.0 {1: 96, 2: 13}  21 apyori\_execute 0.01 0.000014 109 2.0 {1: 96, 2: 13}  22 fpgrowth\_execute 0.01 138.177787 109 2.0 {1: 96, 2: 13}  23 eclat\_execute 0.01 202.242638 219 4.0 {3: 120, 4: 54, 2: 45} |

Таблица 3 – Результаты тестирования алгоритмов для   
Instacart Market Basket Analysis

Уже из данных таблиц видно, что алгоритм ECLAT (https://habr.com/ru/company/ods/blog/353502/) не только является самым медленным алгоритмом, но и выдает результат, отличный от результатов остальных трех алгоритмов.

2. *Подготовьте список частых наборов, в которых не более семи объектов (разумное количество). Проанализируйте и изложите содержательный смысл полученного результата.*

Максимальным значением поддержки обладают наборы из 1 элемента, так как они встречаются наиболее часто. Наборы из 2-х и 3-х элементов встречаются значительно реже, но именно они представляют на набольший интерес, как кандидаты на выявление ассоциативных правил.

|  |
| --- |
| Алгоритм apyori\_execute, поддержка 0.01, количество строк результата - 236  Первые 5 строк результата c наибольшей поддержкой:  support itemsets len  0 0.057915 (1,) 1  111 0.057181 (2,) 1  138 0.048185 (3,) 1  149 0.041699 (4,) 1  160 0.039537 (5,) 1 |

Таблица 4 – Первые 5 строк результата работы алгоритма apriori с наибольшей поддержкой

*3. Выполните визуализацию полученных результатов в виде следующих диаграмм:*

*- сравнение быстродействия алгоритмов на фиксированном наборе данных при изменяемом пороге поддержки.*

Результаты для UCI\_dataset:

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 5 - Результаты для UCI\_dataset

Результаты для market-basket-optimization:

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 6 – Результаты для market-basket-optimization

Результаты для Instacart:

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 7 – Результаты для Instacart

Таким образом, наименьшее время работы показал алгоритм apriori из библиотеки apyori. Наибольшее время работы показал алгоритм eclat, однако при малой поддержке (около 1%) eclat показывает время лучшее, чем априори из библиотеки mlxtend, для которого в данном случае время начинает возрастать экспоненциально.

*- общее количество частых наборов объектов на фиксированном наборе данных при изменяемом пороге поддержки;*

Результаты для UCI\_dataset:

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 8 - Результаты для UCI\_dataset

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 9 – Результаты для market-basket-optimization

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 10 – Результаты для Instacart

Таким образом, количество частых наборов экспоненциально возрастает при уменьшении порога поддержки

*- максимальная длина частого набора объектов на фиксированном наборе данных при изменяемом пороге поддержки;*

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 11 - Результаты для UCI\_dataset

|  |  |
| --- | --- |
|  |  |

Таблица 12 - Результаты для market-basket-optimization

|  |  |
| --- | --- |
|  |  |

Таблица 13 - Результаты для Instacart

*- количество частых наборов объектов различной длины на фиксированном наборе данных при изменяемом пороге поддержки.*

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 14 - Результаты для UCI\_dataset

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 15 - Результаты для market-basket-optimization

|  |  |
| --- | --- |
|  |  |
|  |  |

Таблица 16 - Результаты для Instacart