|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра прикладной математики (ПМ)**

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ**

по дисциплине «Технологии и инструментарий анализа больших данных»

**Практическое занятие №8**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИНБО-07-21, Веригин Никита Алексеевич* | (подпись) | |
| Преподаватель | *Царёв Роман Юрьевич* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_2024 г. | |  | |

Москва 2024 г.

Введение

Цели работы:

1. Загрузить данные Market\_Basket\_Optimisation.csv.

2. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров).

3. Применить алгоритм Apriori, используя 3 разные библиотеки (apriori\_python, apyori, efficient\_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил.

4. Применить алгоритм FP-Growth из библиотеки fpgrowth\_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил.

5. Сравнить время выполнения всех алгоритмов и построить гистограмму.

6. Загрузить данные data.csv.

7. Визуализировать данные (отразить на гистограммах относительную и фактическую частоту встречаемости для 20 наиболее популярных товаров). 8. Применить алгоритм Apriori, используя 3 разные библиотеки (apriori\_python, apyori, efficient\_apriori). Подобрать гиперпараметры для алгоритмов так, чтобы выводилось порядка 10 наилучших правил.

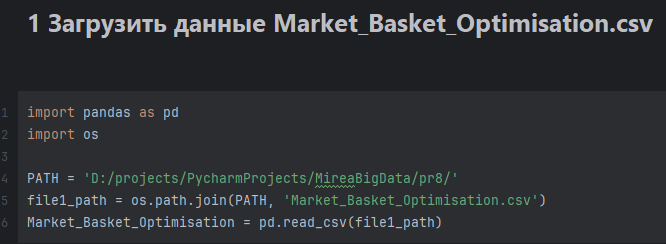
9. Применить алгоритм FP-Growth из библиотеки fpgrowth\_py. Подобрать гиперпараметры для алгоритма так, чтобы выводилось порядка 10 наилучших правил.

10.Сравнить время выполнения всех алгоритмов и построить гистограмму.

11.Сформулировать выводы и сделать отчет.

ХОД РАБОТЫ

Код программ и результаты их выполнения представлены на Рисунках 1-7.



**Рисунок 1 *—* Код решения задачи (ч.1)**

**Изображение выглядит как текст, программное обеспечение, Мультимедийное программное обеспечение, снимок экрана

Автоматически созданное описание**

**Рисунок 2 *—* Код решения задачи (ч.2)**

**Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание**

**Изображение выглядит как снимок экрана, текст, График, линия

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание**

**Рисунок 3 *—* Код решения задачи (ч.3)**

**Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание**

**Рисунок 4 *—* Код решения задачи (ч.4)**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, компьютер

Автоматически созданное описание**

**Рисунок 5 *—* Код решения задачи (ч.5)**

**Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание**

**Рисунок 6 *—* Код решения задачи (ч.6)**

**Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание**

**Изображение выглядит как текст, снимок экрана, График, диаграмма

Автоматически созданное описание**

**Рисунок 7 *—* Код решения задачи (ч.7)**

**Изображение выглядит как текст, снимок экрана, программное обеспечение

Автоматически созданное описание**

**Рисунок 8 *—* Код решения задачи (ч.8)**

**Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Автоматически созданное описание**

**Рисунок 9 *—* Код решения задачи (ч.9)**

**Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание**

**Рисунок 10 *—* Код решения задачи (ч.10)**

Вывод

1. apyori и efficient\_apriori:

- Оба алгоритма имеют минимальное время выполнения (около 0.01 секунд), что делает их наиболее быстрыми среди представленных библиотек.

- Это указывает на то, что эти библиотеки хорошо оптимизированы для представленных данных и настроенных параметров.

2. fpgrowth\_py:

- Выполнение заняло больше времени (0.21 секунд) по сравнению с apyori и efficient\_apriori.

- FP-Growth работает по-другому, используя префиксные деревья для сокращения числа проходов по данным, что иногда может быть более ресурсозатратным по сравнению с Apriori на меньших данных или при определенных конфигурациях.

3. apriori\_python:

- Самая медленная библиотека из всех (0.65 секунд).

- Эта задержка может свидетельствовать о менее оптимизированной реализации или способе обработки данных, особенно для больших наборов или менее эффективных параметров настройки.

**Сравнение алгоритмов**

- Apriori (в трех реализациях):

- Традиционный алгоритм, который повторно просматривает базу данных множества раз, и, как следствие, часто может быть менее эффективным в сравнении с более современным FP-Growth, особенно на больших и сложных наборах данных.

- FP-Growth:

- Использует другую стратегию поиска, уменьшая число проходов по данным и возможность сокращенного хранения инфоприходов с помощью деревьев. Может быть более эффективен на больших и сложных наборах данных, где Apriori столкнется с проблемами производительности.

**Выводы**

- Для переходов ассосациативных правил на большинстве типовых данных подходят все реализации Apriori, но apyori и efficient\_apriori показывают лучшую производительность.

- FP-Growth может быть предпочтительным выбором на больших или сложных наборах данных, если требуется работа с более сложными паттернами и большим числом товаров.

- При выборе библиотеки для работы с ассоциативными правилами важно учитывать размер данных, структуру и требования к производительности. Для больших наборов данных FP-Growth может быть более подходящей альтернативой.

- Предварительный анализ и эксперименты помогут подобрать оптимальные параметры и алгоритмы в зависимости от особенностей данных и поставленных задач.