Mid-Term Project

Submitted By:

Pooja Pilla

Table of Contents

Overview

This project contains Python scripts that accomplish the following:

- **Transaction Data Generation:** Reads a list of items from a CSV file, creates multiple transaction datasets using a sliding window approach, and writes them into CSV files.
- **FP-Growth Implementation:** Uses an FP-Tree to mine frequent patterns from transaction data and then generates association rules based on a minimum support and confidence threshold.
- **Eclat Implementation:** Builds a vertical representation of transactions to mine frequent itemsets using the Eclat algorithm, followed by rule generation.

Project Description

The goal is to extract frequent itemsets and generate association rules from transactional data. The project is divided into two major mining algorithms:

- **FP-Growth:** Efficiently compresses the dataset using a tree structure (FP-Tree) and recursively mines the tree for frequent patterns.
- **Eclat:** Uses a vertical database format to intersect transaction ID (tid) sets and discover frequent itemsets.

Additionally, a helper module is provided for generating synthetic transaction datasets. This allows for testing and comparison of the mining algorithms on multiple, slightly shifted datasets.

Modules and Source Code

Transaction Data Generation

- **File:** transaction_generator.py
- **Key Functions:**
    - import_items(filepath): Reads a CSV file of items.
    - transaction_generator(items, start_offset, items_in_txn, txn_count): Uses a sliding window approach to generate transactions.
    - write_transactions_csv(transactions, filename): Writes the generated transactions to a CSV file with a header.
    - main(): Coordinates the generation of several transaction datasets with unique starting offsets.

Excerpt:

```
python
CopyEdit
def transaction_generator(items, start_offset, items_in_txn, txn_count):
    """
    Yields transactions using a sliding window approach.
    Wraps around the list if needed.
    """
    n = len(items)
    pointer = start_offset
    for _ in range(txn_count):
        txn = []
        for _ in range(items_in_txn):
            txn.append(items[pointer % n])
            pointer += 1
        yield txn
```

FP-Growth Implementation

- **File:** fp_growth.py
- **Key Classes and Functions:**
    - **FPNode & FPTree:** Classes that form the structure of the FP-Tree.
    - build_fp_tree(transactions, min_support): Constructs the FP-Tree after filtering infrequent items.
    - mine_fp_tree(tree, prefix, min_support, frequent_patterns): Recursively mines the FP-Tree for frequent patterns.
    - generate_association_rules(frequent_patterns, transactions, min_confidence): Generates rules based on the frequent patterns found.

Excerpt:

```
python
CopyEdit
class FPNode:
    def __init__(self, item, parent):
        self.item = item
        self.count = 1
        self.parent = parent
        self.children = {}
        self.link = None

    def increment(self, count=1):
        self.count += count
```

Eclat Implementation

- **File:** eclat.py
- **Key Functions:**
    - read_transactions(file_path): Reads transactions from a CSV file and converts them into a set representation.
    - build_vertical_representation(transactions): Converts the transactions into a vertical format (item-to-transaction indices mapping).
    - eclat(prefix, prefix_tids, items, vertical, min_support, freq_itemsets): Recursively mines for frequent itemsets using intersections.

- generate_association_rules(freq_itemsets, transactions, min_confidence): Generates association rules from the mined itemsets.

Excerpt:

python
CopyEdit

```
def eclat(prefix, prefix_tids, items, vertical, min_support, freq_itemsets):
    """
    Recursively mines frequent itemsets using the Eclat algorithm.
    """
    for i, item in enumerate(items):
        new_itemset = prefix | {item}
        new_tids = vertical[item] if prefix_tids is None else prefix_tids & vertical[item]
        if len(new_tids) >= min_support:
            freq_itemsets[frozenset(new_itemset)] = len(new_tids)
            remaining_items = items[i+1:]
            eclat(new_itemset, new_tids, remaining_items, vertical, min_support, freq_itemsets)
```

Usage Instructions

1. **Data Preparation:**
   - Create an items.csv file containing a list of items (one per row).
   - Ensure that the transaction CSV files (e.g., transactions_dataset_1.csv) are generated by running the transaction generator script.
2. **Running FP-Growth:**
   - Execute the FP-Growth module (e.g., run python fp_growth.py).
   - The program will process the transaction files, mine frequent patterns, and print association rules.
3. **Running Eclat:**
   - Run the Eclat module (e.g., execute python eclat.py).
   - Follow the prompts to input the path to your transaction CSV file and thresholds for support and confidence.

Data Files and Formats

- **items.csv:**
  Contains a list of item names. Each line should have at least one item (the first column is used).
- **transactions_dataset_*.csv:**
  Generated CSV files include a header with two columns: "TransactionID" and "Items".
  The "Items" column lists items separated by semicolons.

Annotations and Code Explanation

- **Modular Design:**
  The project is separated into different modules for clarity: one for generating transactions, one for FP-Growth, and one for Eclat. Each module contains descriptive docstrings and inline comments to explain the purpose and functionality of functions and classes.

- **Sliding Window Transaction Generation:**
  Uses a continuous sliding window technique to ensure that each dataset has unique transactions by changing the starting offset.
- **Recursive Mining:**
  Both FP-Growth and Eclat use recursive techniques to explore potential frequent itemsets, making the algorithms efficient for the datasets provided.
- **Rule Generation:**
  Association rules are generated by iterating through all possible non-empty subsets of each frequent itemset and computing the confidence for each rule.

Running Code:

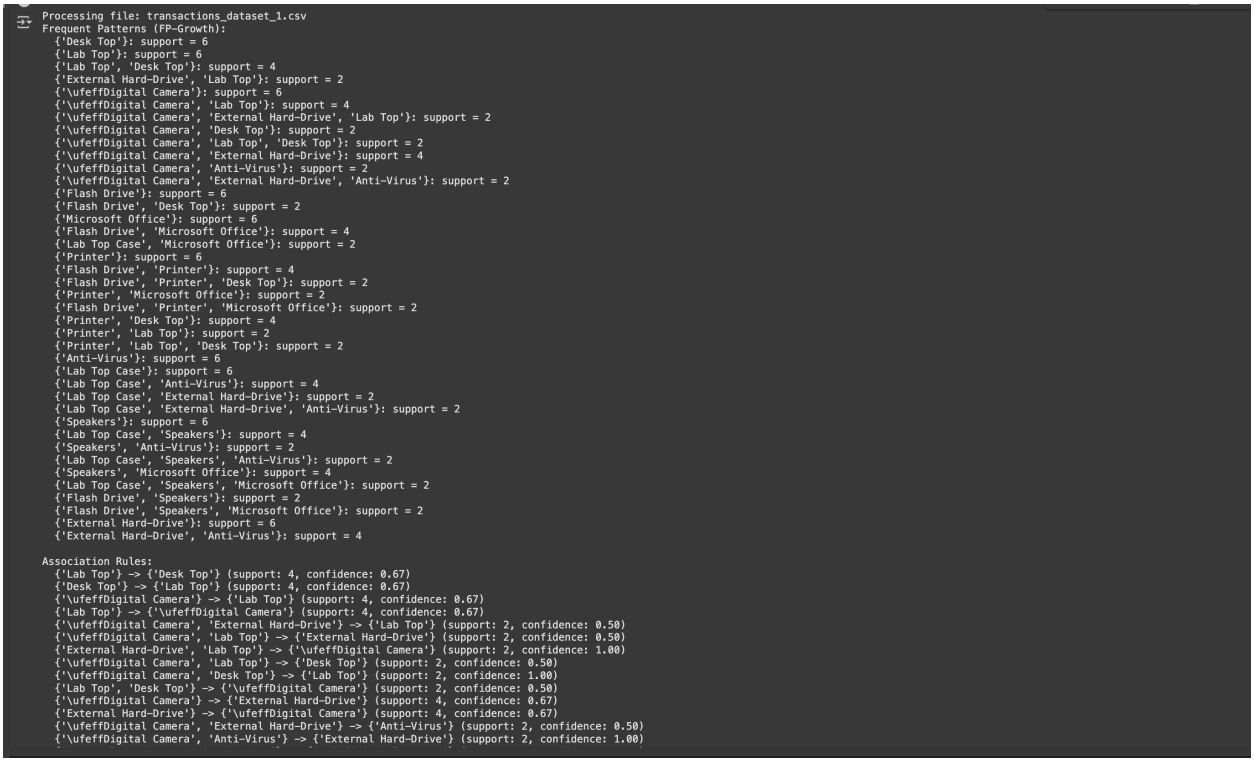
Please open the attached link in google collab upload the dataset file into the environment and run the cells cell by cell.

Source Code Link: [Data Mining Project]

Screenshots:

Transaction Datasets Creation:

Brute Force Method:



Comparative Analysis:

```
Enter the path to the transaction CSV file: transactions_dataset_1.csv
Loaded 20 transactions.
Enter minimum support as a fraction (e.g., 0.05): 0.2
Enter minimum confidence (e.g., 0.5): 0.2
Using minimum support count: 4
```

Enter the CSV Transaction files names in the box you want to process.

Enter Minimum Support value.

Enter Minimum confidence value.

```
Frequent Itemsets Discovered (Eclat):
  {'Anti-Virus'}: support = 6
  {'External Hard-Drive', 'Anti-Virus'}: support = 4
  {'Lab Top Case', 'Anti-Virus'}: support = 4
  {'Desk Top'}: support = 6
  {'Lab Top', 'Desk Top'}: support = 4
  {'Printer', 'Desk Top'}: support = 4
  {'External Hard-Drive'}: support = 6
  {'\ufeffDigital Camera', 'External Hard-Drive'}: support = 4
  {'Flash Drive'}: support = 6
  {'Flash Drive', 'Microsoft Office'}: support = 4
  {'Flash Drive', 'Printer'}: support = 4
  {'Lab Top'}: support = 6
  {'\ufeffDigital Camera', 'Lab Top'}: support = 4
  {'Lab Top Case'}: support = 6
  {'Lab Top Case', 'Speakers'}: support = 4
  {'Microsoft Office'}: support = 6
  {'Speakers', 'Microsoft Office'}: support = 4
  {'Printer'}: support = 6
  {'Speakers'}: support = 6
  {'\ufeffDigital Camera'}: support = 6

Association Rules Derived:
  {'External Hard-Drive'} -> {'Anti-Virus'} (support: 4, confidence: 0.67)
  {'Anti-Virus'} -> {'External Hard-Drive'} (support: 4, confidence: 0.67)
  {'Lab Top Case'} -> {'Anti-Virus'} (support: 4, confidence: 0.67)
  {'Anti-Virus'} -> {'Lab Top Case'} (support: 4, confidence: 0.67)
  {'Lab Top'} -> {'Desk Top'} (support: 4, confidence: 0.67)
  {'Desk Top'} -> {'Lab Top'} (support: 4, confidence: 0.67)
  {'Printer'} -> {'Desk Top'} (support: 4, confidence: 0.67)
  {'Desk Top'} -> {'Printer'} (support: 4, confidence: 0.67)
  {'\ufeffDigital Camera'} -> {'External Hard-Drive'} (support: 4, confidence: 0.67)
  {'External Hard-Drive'} -> {'\ufeffDigital Camera'} (support: 4, confidence: 0.67)
  {'Flash Drive'} -> {'Microsoft Office'} (support: 4, confidence: 0.67)
  {'Microsoft Office'} -> {'Flash Drive'} (support: 4, confidence: 0.67)
  {'Flash Drive'} -> {'Printer'} (support: 4, confidence: 0.67)
  {'Printer'} -> {'Flash Drive'} (support: 4, confidence: 0.67)
  {'\ufeffDigital Camera'} -> {'Lab Top'} (support: 4, confidence: 0.67)
  {'Lab Top'} -> {'\ufeffDigital Camera'} (support: 4, confidence: 0.67)
  {'Lab Top Case'} -> {'Speakers'} (support: 4, confidence: 0.67)
  {'Speakers'} -> {'Lab Top Case'} (support: 4, confidence: 0.67)
  {'Speakers'} -> {'Microsoft Office'} (support: 4, confidence: 0.67)
  {'Microsoft Office'} -> {'Speakers'} (support: 4, confidence: 0.67)

Eclat mining and rule generation completed in 0.0002 seconds.
```

Conclusion

This project successfully demonstrates two robust methodologies—FP-Growth and Eclat—for mining association rules from transactional data. The implementation not only shows how to generate synthetic transaction datasets using a sliding window approach but also highlights the efficiency of each algorithm in identifying frequent itemsets and generating actionable rules. Key insights include:

- **Algorithmic Efficiency:**
  Both FP-Growth and Eclat offer distinct advantages. FP-Growth leverages a compressed tree structure (FP-Tree) to rapidly mine frequent patterns, while Eclat's vertical data representation simplifies the intersection process for support counting.
- **Practical Applicability:**
  The modular design and clear documentation make the project accessible for further extensions and integration into larger data analysis pipelines. This flexibility is critical for adapting the approach to various real-world datasets.
- **Foundational Understanding:**
  By comparing two different mining techniques, the project reinforces foundational concepts in association rule mining, enabling a deeper understanding of how support and confidence thresholds influence rule generation.
- **Future Directions:**
  Potential enhancements include incorporating scalability improvements, exploring additional optimization techniques, and applying the methods to diverse datasets to evaluate performance under varying conditions.

Overall, the project not only provides a comprehensive toolkit for association rule mining but also serves as an educational resource for those looking to understand the intricacies of pattern discovery in large transactional datasets.