Homework - Data Mining - Association Rule Mining

Interactive Supermarket Simulation with Association Rule Mining

Overview

In this assignment, you will design and implement an interactive application/web that simulates a supermarket shopping experience and performs association rule mining to discover purchasing patterns. This project combines data preprocessing, algorithm implementation, and user interface design to create a practical data mining tool. You are encouraged to leverage Generative AI tools (such as ChatGPT, Claude, GitHub Copilot, etc.) to assist with this assignment.

Appropriate Uses of Gen-AI:

- **Learning and Understanding**: Ask AI to explain concepts, algorithms, or debugging techniques
- **Code Generation**: Use AI to generate boilerplate code, UI components, or helper functions
- **Debugging Assistance**: Get help identifying and fixing bugs in your code
- **Documentation**: Assist with writing clear README files and comments
- **Optimization**: Ask for suggestions on improving code efficiency or structure
- **Design Ideas**: Brainstorm UI/UX design approaches

Project Requirements

1. Shopping System Application (25%)

Design an interactive supermarket shopping interface that supports:

Required Features:

- **Manual Transaction Creation**
  - Display at least 10 distinct products as clickable buttons/items
  - Allow users to select products to create shopping transactions
  - Display created transactions in a list or table
  - Allow users to create multiple transactions
- **CSV Data Import**
  - Implement functionality to import transaction data from CSV files
  - Parse and load the provided dataset (sample_transactions.csv)
  - Display import status (number of transactions loaded)
  - Handle file format errors gracefully

- **Data Display**
    - Show current transactions (before and after preprocessing)
    - Display basic statistics (transaction count, unique items, etc.)
    - Provide clear visual feedback for all user actions

User-friendly Environment:

- Clean, intuitive interface suitable for non-technical users
- Clear step-by-step workflow
- Appropriate error messages and user feedback

2. Data Preprocessing (25%)

Implement data cleaning functionality to handle the following types of dirty data:

Required Processing:

1. **Empty Transaction Detection and Removal**
    - Identify transactions with no items
    - Remove them from the dataset
2. **Single-Item Transaction Handling**
    - Detect transactions containing only one item
    - Remove or flag them (single items have no association value)
3. **Duplicate Item Removal**
    - Within each transaction, detect duplicate product entries
    - Remove duplicates (association rules use set-based representation)
4. **Product Name Standardization**
    - Handle case inconsistencies ("Milk" vs "milk" vs "MILK")
    - Remove extra whitespace ("Bread " vs "Bread")
    - Standardize all items to a consistent format (e.g., lowercase)
5. **Invalid Product ID Handling**
    - Detect items not in the valid product list
    - Remove invalid items or entire transactions as appropriate

Preprocessing Report:

Generate a simple report showing:

- Number of issues detected (before cleaning)
- Number of transactions removed

- Number of items cleaned
- Final dataset statistics (after cleaning)

Example:

```
Preprocessing Report:
--------------------
Before Cleaning:
- Total transactions: 85
- Empty transactions: 3
- Single-item transactions: 5
- Duplicate items found: 12 instances
- Invalid items found: 2 instances

After Cleaning:
- Valid transactions: 77
- Total items: 234
- Unique products: 18
```

3. Association Rule Mining (50%)

Implement **BOTH** of the following algorithms:

Required Algorithms:

1. **Apriori Algorithm**
    - Use horizontal data format
    - Generate candidate itemsets level by level
    - Prune using minimum support threshold
    - Extract association rules using minimum confidence threshold
2. **Eclat Algorithm**
    - Use vertical data format (TID-sets)
    - Implement depth-first search
    - Efficient intersection operations for support counting

Algorithm Parameters:

- Minimum Support: User-adjustable (suggested default: 0.2 or 20%)
- Minimum Confidence: User-adjustable (suggested default: 0.5 or 50%)

Performance Comparison:

Track and display for each algorithm:

- Execution time (milliseconds)

- Number of rules generated
- Memory usage (if feasible)

Present comparison in a clear format (table or simple chart).

4. User-Friendly Output (Included in Mining 50%)

Primary Interface: Interactive Query (Required)

Design an intuitive recommendation system where:

- Users select a product from a dropdown/search box
- System displays associated products with that item
- Show association strength as percentages (not just support/confidence values)
- Include simple business recommendations

Example Output:

```
Query: Milk

Customers who bought Milk also bought:
- Bread:  78% of the time  ████████████████ (Strong)
- Eggs:   62% of the time  ████████████     (Moderate)
- Butter: 45% of the time  ████████         (Moderate)

💡 Recommendation: Consider placing Bread near Milk in store layout.
   Potential bundle: Milk + Bread + Eggs
```

Important: The default view should be user-friendly. Technical details should be hidden unless requested.

Technical Requirements

Programming Language

No specific required language, but the following options are recommended:

- Python (recommended: pandas, numpy for data handling)
- JavaScript/TypeScript (Node.js or browser-based)
- Java

Libraries and Frameworks

- You may use standard libraries for UI, file handling, and data structures
- **You must implement the three mining algorithms yourself** (no data mining libraries like mlxtend, scikit-learn for the core algorithms)

- You may use libraries for UI components, CSV parsing, and visualization

Testing

- Test your application thoroughly from a user's perspective
- Ensure all features work correctly with the provided dataset
- Handle edge cases and errors gracefully

Deliverables

Required Submissions:

1. **Source Code** (Complete and Runnable)
    - Well-organized project structure
    - Clear code comments
    - All necessary files to run your application
2. **README.md** (Use Provided Template)
    - Project overview
    - Installation and setup instructions
    - How to use the application
    - Algorithm implementation notes
    - Project structure explanation
3. **REPORT.pdf**
    - Introduction and system design
    - Data preprocessing approach
    - Algorithm implementation details (include pseudocode)
    - Performance analysis and comparison
    - User interface design
    - Testing and results
    - Conclusion and reflection
4. **Data Files** (If applicable)
    - Include any additional test data you created

Submission Format:

Option A: GitHub Repository (Preferred ⭐)

- Create a public (recommended) or private GitHub repository
- Submit the repository link via Canvas

- When using a private repository, remember to share the access.
- Well-organized GitHub submissions may receive bonus points (2-3 points)

Option B: ZIP Archive

- Compress your entire project folder
- Name format: `LastName_FirstName_StudentID_AssociationMining.zip`
- Upload to Canvas

Recommended Project Structure:

```
project-folder/
├── README.md
├── REPORT.pdf
├── src/
│   ├── algorithms/
│   ├── preprocessing/
│   └── main.py (or main entry point)
├── data/
│   ├── sample_transactions.csv
│   └── products.csv
├── requirements.txt (or package.json)
└── screenshots/ (optional but helpful)
```

Provided Materials

1. Dataset: sample_transactions.csv

- Contains 80-100 transactions
- Includes 15-25 different products
- **Intentionally contains dirty data** for preprocessing practice:
    - Empty transactions
    - Single-item transactions
    - Duplicate items within transactions
    - Case inconsistencies in product names
    - Invalid product IDs
    - Extra whitespace

2. Product List: products.csv

- List of valid product IDs and names
- Use this to validate items during preprocessing

3. README.md Template

- Template file to document your project
- Fill in all sections completely

Final Notes

- Using GitHub demonstrates professional development practices and is highly encouraged in the software industry
- Focus on making your application usable by non-technical users
- The quality of your documentation (README and report) is part of your grade
- Test your application on the provided dataset - we will use it for grading
- Ensure your code runs on a standard environment (provide clear setup instructions)
- Gen-AI tools can help you learn faster, but understanding is your responsibility
- Bonus Points (up to 20 points) based on rational add-on, e.g.,
    - Performance comparison visualization
    - Advanced visualization (network graphs, etc.)
    - Additional UI/UX optimization
    - Other innovative features
    - Well-organized GitHub submission
    - etc.
- Learning Philosophy: The goal is to learn data mining concepts and develop practical skills. Gen-AI is a tool that can accelerate your development and help you learn more effectively, but it's not a substitute for understanding. Think of AI as a highly knowledgeable teaching assistant available 24/7.
    - **Remember**: In professional software development, using AI tools is standard practice. Learning to use them effectively is a valuable skill.

Good luck with your assignment! This is an opportunity to build a complete data mining application from scratch while learning to leverage modern AI development tools.