Product Requirements Document

Association Rules Mining with Custom Calculations

Author: Yair Levi
Date: October 15, 2025
Version: 1.0
Status: Final
Document Type: Technical Specification

---

1. Executive Summary

This document outlines the requirements for an association rules mining system that discovers patterns in binary datasets using custom implementations of support, confidence, and lift metrics. The system generates synthetic datasets with controlled probability distributions and identifies association rules without relying on external mining libraries.

1.1 Purpose

Develop a Python-based educational tool and research prototype for association rules mining that demonstrates fundamental data mining algorithms through manual metric calculations.

1.2 Business Goals

- Provide a transparent implementation of association rules mining for educational purposes
- Enable researchers to understand the mathematical foundations of rule discovery
- Create a baseline implementation for algorithm comparison and enhancement
- Support data mining education and training programs

1.3 Success Criteria

- Generate datasets matching specified probability distributions (within ±2%)
- Correctly calculate support, confidence, and lift using mathematical formulas
- Discover all qualifying rules meeting threshold criteria
- Execute within 5 seconds for 5000-row datasets
- Produce clear, interpretable output suitable for educational use

---

2. Product Overview

2.1 Target Users

User Type	Description	Primary Use Case
Data Science Students	Learning data mining concepts	Understanding association rules algorithms
Educators	Teaching data mining courses	Demonstrating rule mining mechanics
Researchers	Developing new algorithms	Baseline for performance comparison
Analysts	Exploring pattern discovery	Prototyping analysis workflows

2.2 Key Features

- Synthetic binary dataset generation with configurable distributions
- Manual implementation of support, confidence, and lift calculations

- Exhaustive rule discovery across all possible itemsets
- Configurable support and confidence thresholds
- Comprehensive statistical reporting and analysis
- Educational output with interpretable results

2.3 System Context

The system operates as a standalone Python script requiring only NumPy. It processes in-memory data and produces console output suitable for analysis and documentation.

---

3. Functional Requirements

3.1 Dataset Generation (FR-1)

Priority: Critical
Complexity: Medium

Requirements

The system shall generate a binary dataset with the following specifications:

FR-1.1: Dataset Dimensions

- Rows: 5,000 transactions (configurable)
- Columns: 6 binary features labeled 'a' through 'f' (configurable)
- Data type: Integer (0 or 1)
- Storage: NumPy 2D array

FR-1.2: Probability Distribution Each feature shall have a specified probability of containing value 1:

Feature	Target Probability	Expected Count (5000 rows)
a	80%	~4,000
b	60%	~3,000
c	40%	~2,000
d	30%	~1,500
e	20%	~1,000
f	10%	~500

FR-1.3: Random Generation

- Use NumPy's random.choice() function
- Support random seed configuration for reproducibility
- Generate each feature column independently

FR-1.4: Distribution Verification

- Calculate actual percentage of 1s in each column
- Display comparison between target and actual percentages
- Accept variation within ±2% of target values

Acceptance Criteria:

- Dataset created with exact dimensions (5000, 6)
- All values are either 0 or 1
- Actual distributions match targets within ±2%
- Generation completes in < 0.5 seconds

3.2 Support Calculation (FR-2)

Priority: Critical
Complexity: Medium

Requirements

FR-2.1: Support Formula Implementation Implement support calculation using the formula:

```
Support(X) = Count(transactions containing all items in X) / Total transactions
```

FR-2.2: Implementation Details

- Accept itemset as list of feature indices
- Use boolean masking for efficient counting
- Return support as decimal (0.0 to 1.0)
- Handle edge cases (empty itemsets, invalid indices)

FR-2.3: Efficiency Requirements

- Use vectorized NumPy operations (no Python loops over rows)
- Calculate support in O(n) time where n = number of rows
- Support itemsets of any size (1 to 6 features)

Acceptance Criteria:

- Correct support values for all itemset sizes
- Supports empty itemsets (return 0.0)
- Execution time < 0.001 seconds per itemset
- Results match manual calculations

Test Cases:

```
Support({a}) ≈ 0.80
Support({a, b}) ≈ 0.48
Support({a, b, c}) ≈ 0.19
```

3.3 Confidence Calculation (FR-3)

Priority: Critical
Complexity: Medium

Requirements

FR-3.1: Confidence Formula Implementation Implement confidence calculation using the formula:

$$\text{Confidence}(X \rightarrow Y) = \text{Support}(X \cup Y) / \text{Support}(X)$$

FR-3.2: Implementation Details

- Accept antecedent and consequent as separate lists
- Calculate union support (antecedent + consequent)
- Handle division by zero (return 0.0 if antecedent support is 0)
- Return confidence as decimal (0.0 to 1.0)

FR-3.3: Input Validation

- Ensure antecedent and consequent are non-empty
- Verify no overlap between antecedent and consequent
- Handle invalid feature indices gracefully

Acceptance Criteria:

- Correct confidence values for all valid rules
- Proper handling of edge cases (zero support, empty sets)
- Results interpretable as percentages
- Confidence never exceeds 1.0

Test Cases:

$$\text{Confidence}(\{b\} \rightarrow \{a\}) \approx 0.80$$
$$\text{Confidence}(\{c\} \rightarrow \{a\}) \approx 0.78$$

3.4 Lift Calculation (FR-4)

Priority: High
Complexity: Low

Requirements

FR-4.1: Lift Formula Implementation Implement lift calculation using the formula:

$$\text{Lift}(X \rightarrow Y) = \text{Support}(X \cup Y) / (\text{Support}(X) \times \text{Support}(Y))$$

FR-4.2: Implementation Details

- Calculate using support values only
- Handle division by zero (return 0.0)

- Support lift interpretation (>1, =1, <1)
- Return lift as decimal (typically 0.0 to 3.0)

FR-4.3: Lift Interpretation System should support interpretation of lift values:

- Lift > 1.0: Positive correlation
- Lift = 1.0: Independence
- Lift < 1.0: Negative correlation

Acceptance Criteria:

- Correct lift calculations for all rules
- Proper handling of zero support scenarios
- Lift values mathematically consistent with support/confidence
- Results enable correlation interpretation

3.5 Rule Discovery (FR-5)

Priority: Critical
Complexity: High

Requirements

FR-5.1: Itemset Generation Generate all possible itemsets from 2 to 6 features:

- Use itertools.combinations for enumeration
- Process itemsets in order of increasing size
- Calculate support for each itemset

FR-5.2: Frequent Itemset Filtering

- Apply minimum support threshold (default: 30%)
- Only generate rules from frequent itemsets
- Prune infrequent itemsets early

FR-5.3: Rule Generation For each frequent itemset:

- Generate all possible antecedent-consequent splits
- Calculate confidence for each rule
- Filter rules by minimum confidence threshold (default: 70%)

FR-5.4: Rule Storage Store qualifying rules with:

- Antecedent (list of feature indices)
- Consequent (list of feature indices)
- Support (decimal)
- Confidence (decimal)
- Lift (decimal)

FR-5.5: Thresholds

- Minimum Support: 30% (configurable)
- Minimum Confidence: 70% (configurable)
- Both thresholds must be satisfied

Acceptance Criteria:

- All rules meeting both thresholds discovered

- No duplicate rules in output
- Rules sorted by confidence (descending)
- Complete enumeration of rule space
- Execution completes in < 5 seconds

3.6 Output and Reporting (FR-6)

Priority: High
Complexity: Medium

Requirements

FR-6.1: Console Output Structure Display the following sections in order:

1. System header with program title
2. Dataset configuration parameters
3. Threshold settings
4. Dataset creation confirmation
5. Actual vs. target distributions
6. Rule discovery progress
7. Discovered rules table
8. Summary statistics
9. Top rules by confidence

FR-6.2: Rules Display Format For each rule, display:

- Rule notation: {antecedent} → {consequent}
- Support: decimal and percentage
- Confidence: decimal and percentage
- Lift: decimal (4 decimal places)

FR-6.3: Summary Statistics Calculate and display:

- Total number of rules found
- Average support, confidence, lift
- Maximum support, confidence, lift
- Minimum support, confidence, lift (if useful)

FR-6.4: Top Rules Section Display top 5 rules by confidence with:

- Rule notation with feature names
- All three metrics with percentages
- Clear numbering and formatting

FR-6.5: Formatting Requirements

- Use consistent column widths
- Align numbers properly
- Include visual separators (borders, lines)
- Use clear section headers
- Display percentages alongside decimals

Acceptance Criteria:

- Output is clearly structured and readable
- All metrics displayed with appropriate precision
- Rules sortable and filterable
- Statistics accurately calculated

- Output suitable for documentation

---

4. Non-Functional Requirements

4.1 Performance (NFR-1)

Priority: High

Metric	Requirement	Target
Dataset Generation	< 1 second	0.5 seconds
Rule Mining	< 10 seconds	5 seconds
Total Execution	< 15 seconds	8 seconds
Memory Usage	< 500 MB	100 MB
Scalability	Up to 50,000 rows	10,000 rows

Constraints:

- Must work on standard desktop/laptop hardware
- Single-threaded execution acceptable
- No GPU acceleration required

4.2 Usability (NFR-2)

Priority: High

Ease of Use:

- Zero configuration for default operation
- Single file implementation
- Clear error messages
- Self-documenting output

Modifiability:

- Parameters clearly defined at top of script
- Well-commented code
- Modular function structure
- Easy threshold adjustment

Educational Value:

- Code demonstrates algorithm steps clearly
- Output explains metrics and interpretations
- Suitable for classroom demonstrations
- Manual calculations visible in code

4.3 Maintainability (NFR-3)

Priority: Medium

Code Quality:

- Clear variable naming
- Comprehensive comments
- Docstrings for all functions
- Consistent coding style

Documentation:

- Inline code comments explaining logic
- Function docstrings with parameters and return values
- README with usage examples
- Mathematical formulas documented

Extensibility:

- Easy to add new metrics
- Configurable feature count
- Pluggable threshold strategies
- Support for custom probability distributions

4.4 Reliability (NFR-4)

Priority: High

Correctness:

- Mathematically accurate calculations
- Handles edge cases properly
- No silent failures
- Validates input data

Robustness:

- Graceful handling of invalid inputs
- Prevents division by zero
- Manages memory efficiently
- No crashes on edge cases

4.5 Portability (NFR-5)

Priority: Medium

Cross-Platform:

- Works on Windows, macOS, Linux
- Python 3.7+ compatibility
- No platform-specific code
- Standard library dependencies only (+ NumPy)

---

5. Technical Specifications

5.1 Technology Stack

Component	Technology	Version	Purpose
Language	Python	3.7+	Implementation
Array Processing	NumPy	1.19+	Dataset and calculations
Standard Library	itertools	Built-in	Combinations generation
Standard Library	random	Built-in	Optional seed setting

5.2 Data Structures

Dataset:

python

```
Type: numpy.ndarray
Shape: (5000, 6)
dtype: int (0 or 1)
```

Rule:

python

```
{
  'antecedent': [int, ...],      # Feature indices
  'consequent': [int, ...],      # Feature indices
  'support': float,              # 0.0 to 1.0
  'confidence': float,           # 0.0 to 1.0
  'lift': float                  # Typically 0.0 to 3.0
}
```

5.3 Algorithm Complexity

Time Complexity:

- Dataset Generation: O(n × m) where n=rows, m=features
- Support Calculation: O(n) per itemset
- Itemset Enumeration: O(2^m) possible itemsets
- Rule Generation: O(2^k) per k-itemset
- Overall: O(n × 2^m)

Space Complexity:

- Dataset: O(n × m)
- Rules Storage: O(r) where r=number of rules
- Overall: O(n × m + r)

5.4 Configuration Parameters

python

```
# Dataset Configuration
n_rows = 5000                # Number of transactions
n_features = 6               # Number of features
feature_names = ['a', 'b', 'c', 'd', 'e', 'f']
probabilities = [0.80, 0.60, 0.40, 0.30, 0.20, 0.10]

# Mining Configuration
min_support = 0.30           # 30% minimum support
min_confidence = 0.70        # 70% minimum confidence

# Reproducibility
random_seed = 42             # Optional, None for random
```

---

6. Use Cases

6.1 Educational Demonstration (UC-1)

Actor: University Professor
Goal: Teach association rules mining in data mining course
Preconditions: Python and NumPy installed, students have basic Python knowledge

Main Flow:

1. Professor explains association rules concepts
2. Runs script to generate dataset
3. Shows actual probability distributions vs. targets
4. Explains support calculation with examples
5. Demonstrates confidence and lift interpretations
6. Discusses discovered rules and their meanings
7. Adjusts thresholds to show impact on results
8. Students modify parameters and observe changes

Postconditions: Students understand support, confidence, lift, and rule mining process

Alternative Flows:

- Modify probabilities to create different scenarios
- Compare with library implementations (mlxtend)
- Export results for homework analysis

6.2 Research Baseline (UC-2)

Actor: Data Mining Researcher
Goal: Establish baseline performance for new algorithm
Preconditions: Understanding of association rules mining

Main Flow:

1. Researcher generates test dataset with known distributions
2. Runs baseline implementation to get ground truth
3. Records execution time and memory usage

4. Documents discovered rules
5. Implements new algorithm
6. Compares results with baseline
7. Measures performance improvements
8. Publishes comparative analysis

Postconditions: Performance comparison documented, results validated

6.3 Market Basket Analysis Prototype (UC-3)

Actor: Business Analyst
Goal: Prototype product association analysis
Preconditions: Understanding of business data structure

Main Flow:

1. Analyst maps products to binary features
2. Adjusts probabilities to match product frequencies
3. Configures support/confidence thresholds for business relevance
4. Runs mining algorithm
5. Interprets discovered rules for marketing insights
6. Identifies product bundling opportunities
7. Presents findings to stakeholders
8. Decides whether to implement full-scale solution

Postconditions: Business insights discovered, decision made on full implementation

6.4 Algorithm Learning (UC-4)

Actor: Self-Learning Data Scientist
Goal: Understand association rules mining internals
Preconditions: Basic Python and math knowledge

Main Flow:

1. Downloads and reads script
2. Reviews mathematical formulas in comments
3. Runs with default parameters
4. Manually verifies calculations with calculator
5. Modifies thresholds to see effect
6. Adds print statements to trace execution
7. Implements improvements or optimizations
8. Compares with textbook algorithms

Postconditions: Deep understanding of algorithm mechanics

---

7. Constraints and Assumptions

7.1 Constraints

Technical Constraints:

- Python 3.7+ required (f-string support, type hints compatible)
- NumPy library must be installed
- Single-threaded execution only
- In-memory processing (no database)

- Command-line interface only (no GUI)

Performance Constraints:

- Practical limit of 10-15 features (exponential complexity)
- Maximum dataset size: 100,000 rows
- No real-time processing requirements
- Batch processing only

Business Constraints:

- Educational use case (not production system)
- No warranty or support requirements
- Open-source distribution
- No monetization

7.2 Assumptions

User Assumptions:

- Basic Python programming knowledge
- Understanding of basic statistics
- Familiarity with command-line interfaces
- Access to Python development environment

Data Assumptions:

- Binary features only (0 or 1)
- Independent feature generation
- No missing values
- Synthetic data (not real-world)

Environment Assumptions:

- Desktop/laptop environment
- Sufficient RAM (>1GB available)
- No network connectivity required
- Standard operating system (Windows/macOS/Linux)

---

8. Dependencies and Integrations

8.1 External Dependencies

Dependency	Version	Purpose	Critical
NumPy	≥1.19.0	Array operations, random generation	Yes
Python	≥3.7	Runtime environment	Yes

8.2 Standard Library Dependencies

- `itertools`: Combination generation
- `random` (via NumPy): Random number generation

8.3 Future Integrations

Planned:

- Export to CSV/JSON for analysis tools
- Visualization with matplotlib (optional)
- Integration with pandas DataFrames
- API for programmatic access

Potential:

- Web interface for parameter adjustment
- Database connectivity for real data
- Parallel processing with multiprocessing
- GPU acceleration with CuPy

---

9. Success Metrics and KPIs

9.1 Functional Metrics

Metric	Target	Measurement Method
Calculation Accuracy	100%	Compare with manual calculations
Rule Discovery Completeness	100%	Verify all qualifying rules found
Distribution Accuracy	±2%	Compare actual vs. target distributions
Execution Success Rate	100%	No crashes or errors

9.2 Performance Metrics

Metric	Target	Measurement Method
Execution Time (5K rows)	< 5s	Time measurement
Memory Usage	< 100 MB	Memory profiler
Support Calc Time	< 1ms per itemset	Profiling
Scalability Factor	Linear in rows	Benchmark tests

9.3 Quality Metrics

Metric	Target	Measurement Method
Code Coverage	> 80%	pytest-cov
Documentation Coverage	100%	Manual review
Comment Ratio	> 25%	Code analysis
Cyclomatic Complexity	< 10 per function	pylint/radon

9.4 User Experience Metrics

Metric	Target	Measurement Method
Setup Time	< 5 minutes	User testing
First-Run Success	> 95%	User feedback
Output Clarity	> 90% satisfaction	Survey
Educational Effectiveness	> 85% comprehension	Quiz/assessment

---

10. Testing Requirements

10.1 Unit Tests

Required Test Cases:

Dataset Generation:

- Test correct dimensions (5000, 6)
- Test value range (only 0 and 1)
- Test distribution accuracy (within ±2%)
- Test reproducibility with seed

Support Calculation:

- Test single-item support
- Test multi-item support
- Test edge cases (empty, full)
- Test accuracy vs. manual calculation

Confidence Calculation:

- Test various antecedent-consequent pairs
- Test division by zero handling
- Test range validation (0.0 to 1.0)
- Test accuracy vs. formula

Lift Calculation:

- Test positive correlation (lift > 1)
- Test independence (lift ≈ 1)
- Test negative correlation (lift < 1)
- Test zero support handling

Rule Discovery:

- Test threshold filtering
- Test complete enumeration
- Test sorting correctness
- Test no duplicates

10.2 Integration Tests

- End-to-end execution with default parameters
- Parameter modification scenarios
- Output format verification
- Performance benchmarks

10.3 Validation Tests

- Compare with known association rules libraries (mlxtend)
- Verify mathematical correctness with hand calculations
- Cross-validate with different random seeds
- Stress test with varying dataset sizes

---

11. Risks and Mitigation

Risk	Impact	Probability	Mitigation Strategy
NumPy version incompatibility	High	Low	Document minimum version, test on multiple versions
Memory exhaustion with large datasets	High	Medium	Document limits, add memory checks, implement early stopping
Incorrect calculations	Critical	Low	Extensive unit tests, manual verification, peer review
Poor performance with many features	Medium	High	Document complexity, warn users, suggest feature limits
Confusing output format	Medium	Medium	User testing, clear formatting, examples in documentation
Difficulty understanding metrics	Medium	Medium	Add interpretations, examples, educational content
Platform-specific issues	Low	Low	Test on all major platforms, use standard libraries only
User mistakes in configuration	Low	Medium	Clear parameter names, validation, helpful error messages

---

12. Future Enhancements

12.1 Phase 2 Features (Post-MVP)

Priority: High

- [ ] Export results to CSV/JSON formats
- [ ] Add visualization of rules (network graphs)
- [ ] Implement Apriori algorithm for comparison
- [ ] Add progress bars for long-running operations
- [ ] Create interactive parameter tuning

Priority: Medium

- [ ] Support for multi-valued features (not just binary)
- [ ] Parallel processing for large datasets
- [ ] GUI application using Tkinter or PyQt
- [ ] Real-time dataset loading from CSV
- [ ] Statistical significance testing (chi-square)

Priority: Low

- [ ] Web-based interface with Flask/Django
- [ ] Database integration (SQLite, PostgreSQL)
- [ ] Cloud deployment options
- [ ] Mobile app for result viewing
- [ ] Integration with BI tools (Tableau, Power BI)

12.2 Advanced Features (Version 2.0)

- FP-Growth algorithm implementation
- Sequential pattern mining
- Hierarchical association rules
- Fuzzy association rules
- Temporal association rules
- Multi-level association rules
- GPU acceleration
- Distributed processing (Spark integration)

---

13. Documentation Requirements

13.1 User Documentation

Required Documents:

- README.md with quick start guide
- Installation instructions
- Usage examples with screenshots
- Parameter reference guide
- Troubleshooting guide
- FAQ section

13.2 Technical Documentation

Required Documents:

- Algorithm explanation with formulas
- Code architecture overview
- Function reference (auto-generated from docstrings)
- Performance characteristics
- Complexity analysis
- Testing guide

13.3 Educational Materials

Required Documents:

- Tutorial for beginners
- Classroom exercises
- Assignment ideas for instructors
- Comparison with commercial tools
- Real-world use case examples
- Video demonstrations (optional)

---

14. Acceptance Criteria

14.1 Functional Acceptance

The system is accepted if:

- ✅ Generates 5000×6 binary dataset with specified distributions (±2%)
- ✅ Calculates support, confidence, and lift accurately (verified manually)
- ✅ Discovers all rules meeting support > 30% and confidence > 70%
- ✅ Produces clearly formatted output with all required sections
- ✅ Executes successfully on Windows, macOS, and Linux
- ✅ Completes execution in < 5 seconds for default parameters

14.2 Quality Acceptance

The system is accepted if:

- ✅ Code is well-commented with docstrings
- ✅ No critical or high-severity bugs

- ✅ Documentation is complete and accurate
- ✅ Unit tests achieve > 80% coverage
- ✅ Manual testing validates all features
- ✅ Peer review completed and approved

14.3 Performance Acceptance

The system is accepted if:

- ✅ Execution time < 5 seconds (5000 rows, 6 features)
- ✅ Memory usage < 100 MB
- ✅ Scales to 10,000 rows without issues
- ✅ No memory leaks detected

---

15. Approval and Sign-off

Role	Name	Signature	Date
Author	Yair Levi	_____	Oct 15, 2025
Technical Reviewer	_____	_____	_____
Product Owner	_____	_____	_____
Stakeholder	_____	_____	_____

---

16. Revision History

Version	Date	Author	Changes
0.1	Oct 8, 2025	Yair Levi	Initial draft
0.5	Oct 12, 2025	Yair Levi	Added detailed requirements
1.0	Oct 15, 2025	Yair Levi	Final version after implementation

---

Appendix A: Mathematical Formulas Reference

Support Formula

$$\text{Support}(X) = |\{t \in T \mid X \subseteq t\}| / |T|$$

Where:
- T = set of all transactions
- X = itemset
- $|\cdot|$ = cardinality (count)

Confidence Formula

$$\text{Confidence}(X \rightarrow Y) = \text{Support}(X \cup Y) / \text{Support}(X)$$
$$= P(Y \mid X)$$

Where:
- X = antecedent (left-hand side)
- Y = consequent (right-hand side)
- P(Y | X) = conditional probability

Lift Formula

$$\text{Lift}(X \rightarrow Y) = \text{Support}(X \cup Y) / (\text{Support}(X) \times \text{Support}(Y))$$
$$= \text{Confidence}(X \rightarrow Y) / \text{Support}(Y)$$
$$= P(X \cup Y) / (P(X) \times P(Y))$$

Interpretation:
- Lift > 1: Positive correlation
- Lift = 1: Independence
- Lift < 1: Negative correlation

Appendix B: Sample Output Format

```
================================================================================
ASSOCIATION RULES MINING
================================================================================

Dataset: 5000 rows, 6 features
Features: ['a', 'b', 'c', 'd', 'e', 'f']
Probabilities: [0.8, 0.6, 0.4, 0.3, 0.2, 0.1]

Thresholds:
  Minimum Support: 30.0%
  Minimum Confidence: 70.0%
================================================================================

[... Dataset generation and verification ...]

Rule                    Support      Confidence   Lift
--------------------------------------------------------------------------------
{b} -> {a}               0.4748 (47.48%)  0.7980 (79.80%)  1.0048
{c} -> {a}               0.3120 (31.20%)  0.7816 (78.16%)  0.9841
```

Document Control:

- Classification: Internal Use
- Distribution: Development Team, Stakeholders
- Review Cycle: Quarterly
- Next Review: January 15, 2026