***“*Heaven’s Light is Our Guide”**

A logo with text and symbols

AI-generated content may be incorrect.

**Rajshahi University of Engineering & Technology**

**Department of Computer Science & Engineering**

**Lab Report-3**

**Course Code: CSE 4120**

**Course Title: Data Mining Sessional**

|  |  |
| --- | --- |
| **Submitted By-**  **Name: Sajidur Rahman Tarafder**  **Department: CSE**  **Roll No: 2003154**  **Section: C**  **Session:2020-21** | **Submitted To-**  **Julia Rahman**  **Associate Professor**  **Department of CSE**  **RUET** |

**Module No:** **03**

**Experiment Name:** **Exploratory Data Analysis (EDA)**

**Objectives:**

Understanding basic data mining concepts and tools through visual analysis of the Titanic dataset using histograms for distribution analysis, boxplots for outlier identification, and correlation heatmaps for relationship analysis.

**Dataset:**

Titanic Dataset - Contains passenger information with 891 records and 12 attributes including Age, Sex, Pclass, Fare, etc.

**Task 1: Load and Display Dataset**

**Code:**

A screen shot of a computer code

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Task 2: Plot Distributions of Variables**

**Implementation:**

- Selected 4 key numerical features: from the Titanic dataset

- Created histogram plots to analyze the shape and spread of

numerical features

- Used 20 bins for adequate granularity in distribution

visualization

- Applied dropna() to handle missing values automatically

**Code (Distribution Analysis):**

A computer screen with text on it

AI-generated content may be incorrect.

**Output:**

**A group of blue and green bars

AI-generated content may be incorrect.**

**Discussion:**

The histogram analysis revealed important characteristics of our numerical variables:

**Age Distribution:** Shows a roughly normal distribution with a slight right skew, indicating more younger passengers were aboard the Titanic. The peak occurs around 20-30 years, reflecting the demographic composition of early 20th century travelers.

**Fare Distribution:** Exhibits extreme right skew, reflecting the wide range of ticket prices from third-class economy to luxury first-class accommodations. Most passengers paid low fares (under $50), while a small number paid exceptionally high amounts.

**SibSp & Parch Distributions:** Both shows highly right-skewed distributions, with most passengers traveling with few or no family members. This indicates that solo travelers or small family groups were more common than large families.

**Task 3: Identify Outliers Using Boxplots**

**Implementation:**

- Created boxplot visualizations for all 4 numerical variables

- Applied IQR (Interquartile Range) method for quantitative

outlier detection

- Calculated outlier boundaries using Q1 - 1.5×IQR and Q3 +

1.5×IQR formula

**Code (Outlier Detection):**

A screen shot of a computer code

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Discussion:**

Boxplot analysis using the IQR method identified several key patterns:

**Fare Outliers (116):** The highest number of outliers, representing passengers who paid exceptionally high prices, likely first-class luxury suites or special accommodations. These outliers are legitimate data points reflecting the economic disparity among passengers.

**Parch Outliers (213):** High number of outliers indicates many passengers traveled with larger numbers of parents/children than typical, suggesting some large family groups or special circumstances.

**SibSp Outliers (46):** Moderate outliers are representing the passengers with many siblings/spouses, indicating extended family travel arrangements.

**Age Outliers (11):** Fewest outliers, representing very young children and elderly passengers at the extremes of the age distribution.

**Task 4: Compute Pairwise Correlations**

**Implementation:**

- Calculated correlation matrix for all 4 numerical variables

- Created correlation heatmap using seaborn with color coding

- Applied 'cool warm' colormap for intuitive positive/negative

correlation visualization

- Displayed correlation values with 3 decimal precisions for

accuracy

**Code (Correlation Analysis):**

A computer screen with text on it

AI-generated content may be incorrect.

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Discussion:**

The correlation analysis revealed interesting relationships between passenger characteristics:

**Strongest Positive Correlation (0.415):** SibSp vs Parch indicates that passengers with siblings/spouses also tend to travel with parents/children, suggesting family group travel patterns.

**Moderate Negative Correlations:** Age shows negative correlations with both SibSp (-0.308) and Parch (-0.189), indicating that older passengers tend to travel with fewer family members, possibly reflecting life stage differences.

**Weak Positive Correlations:** Fare shows weak positive correlations with family size variables (SibSp: 0.160, Parch: 0.216), potentially indicating family ticket pricing or larger accommodations for families.

**Age-Fare Relationship (0.096**): Very weak correlation suggests that ticket price was not strongly related to passenger age, indicating that class selection was based on economic factors rather than age demographics.

**Conclusion:**

This exploratory data analysis successfully achieved its objective of understanding basic data mining concepts through practical application on the Titanic dataset. The analysis provided crucial insights:

**1.** **Distribution Patterns:** Identified demographic composition

and economic disparities among passengers

**2.** **Outlier Detection:** Revealed data quality and extreme cases

requiring special attention

**3.** **Variable Relationships:** Uncovered family travel patterns and

socioeconomic correlations

The EDA techniques demonstrated their effectiveness in revealing hidden patterns and relationships in historical data, providing a solid foundation for subsequent data mining and machine learning applications.

**Module No:** **04**

**Experiment Name: Mining Frequent Item sets and**

**Association Rules**

**Objectives:**

Implementing the Apriori algorithm manually to discover frequent itemsets and generate association rules from market basket data without using built-in libraries for frequent pattern mining.

**Dataset:**

[Lab4\_groceries.csv](vscode-file://vscode-app/Users/sajid/Downloads/Visual%20Studio%20Code.app/Contents/Resources/app/out/vs/code/electron-browser/workbench/workbench.html) - Contains weekly sales data with 52 records and 4 attributes including Date, ToothPaste, PeanutButter, and Biscuits sales figures representing grocery store transactions over one year.

**Task 1: Load and Display Dataset**

**Code:**

**A computer screen with text on it

AI-generated content may be incorrect.**

**Output:**

A screenshot of a computer screen

AI-generated content may be incorrect.

**Task 2: Generate Transaction Database**

**Implementation:**

1. Used median-based threshold approach to convert

continuous sales data into binary transactions

2. Applied median sales values as cutoff points to determine

high-sales weeks for each product

3. Generated transaction database where items appear only

when sales exceed their respective median values

**Code (Transaction Generation):**

A computer screen with text on it

AI-generated content may be incorrect.

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Discussion:**

The transaction generation process successfully converted continuous sales data into a format suitable for market basket analysis. Using median values as thresholds proved effective for binary conversion, as it naturally divided each product's sales into "high" and "low" periods. The median approach ensured balanced representation with approximately 50% of weeks showing above-median sales for each product.

ToothPaste showed a median of 229.5, PeanutButter 478.0, and Biscuits 394.5, reflecting the different sales scales across products. This preprocessing step was crucial for applying the Apriori algorithm, as it transformed the numerical sales data into the categorical transaction format required for frequent itemset mining.

**Task 3: Manual Apriori Algorithm Implementation**

**Implementation:**

1. Implemented complete Apriori algorithm from scratch

without using built-in functions

2. Used minimum support count of 13 transactions (25% of

total)

3. Generated candidate itemsets (C1, C2, C3) and filtered

frequent itemsets (L1, L2, L3)

4. Applied support counting and pruning at each level

**Code (Apriori Algorithm):**

|  |
| --- |
| **def *calculate\_support*(itemset, transactions):**  **count = 0**  **for transaction in transactions:**  **if *all*(item in transaction for item in itemset):**  **count += 1**  **return count / *len*(transactions)**  **def *generate\_candidates*(frequent\_itemsets, k):**  **candidates = []**  **n = *len*(frequent\_itemsets)**  **for i in *range*(n):**  **for j in *range*(i + 1, n):**  **union = frequent\_itemsets[i] | frequent\_itemsets[j]**  **if *len*(union) == k:**  **candidates.append(union)**  **return list(set(frozenset(c) for c in candidates))**  **min\_support\_count = 13**  **items = ['ToothPaste', 'PeanutButter', 'Biscuits']**  ***print*("Database D:")**  **for i, transaction in *enumerate*(transactions[:5]):**  ***print*(f"T{i+1}: {transaction}")**  ***print*(f"... ({*len*(transactions)} total transactions)")**  ***print*(f"Min support count = {min\_support\_count}")**  ***# C1***  ***print*(f"\nC1:")**  **C1 = [{item} for item in items]**  **for itemset in C1:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  ***print*(f" {{{list(itemset)[0]}}}: {support\_count}")**  ***# L1***  ***print*(f"\nL1:")**  **L1 = []**  **for itemset in C1:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  **if support\_count >= min\_support\_count:**  **L1.append(frozenset(itemset))**  ***print*(f" {{{list(itemset)[0]}}}: {support\_count}")**  **frequent\_itemsets = {1: L1}**  ***# C2***  ***print*(f"\nC2:")**  **C2 = [frozenset(combo) for combo in combinations(items, 2)]**  **for itemset in C2:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  **item\_list = *sorted*(list(itemset))**  ***print*(f" {{{' '.join(item\_list)}}}: {support\_count}")**  ***# L2***  ***print*(f"\nL2:")**  **L2 = []**  **for itemset in C2:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  **if support\_count >= min\_support\_count:**  **L2.append(itemset)**  **item\_list = *sorted*(list(itemset))**  ***print*(f" {{{' '.join(item\_list)}}}: {support\_count}")**  **if not L2:**  ***print*(" (empty)")**  **frequent\_itemsets[2] = L2**  ***# C3***  ***print*(f"\nC3:")**  **if *len*(L2) >= 2:**  **C3 = generate\_candidates(L2, 3)**  **else:**  **C3 = [frozenset(items)]**  **for itemset in C3:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  **item\_list = *sorted*(list(itemset))**  ***print*(f" {{{' '.join(item\_list)}}}: {support\_count}")**  ***# L3***  ***print*(f"\nL3:")**  **L3 = []**  **for itemset in C3:**  **support\_count = *sum*(1 for t in transactions if *all*(item in t for item in itemset))**  **if support\_count >= min\_support\_count:**  **L3.append(itemset)**  **item\_list = *sorted*(list(itemset))**  ***print*(f" {{{' '.join(item\_list)}}}: {support\_count}")**  **if not L3:**  ***print*(" (empty)")**  **frequent\_itemsets[3] = L3** |

**Output:**

A screenshot of a computer program

AI-generated content may be incorrect.

**Discussion:**

The Apriori algorithm execution revealed interesting patterns in the grocery sales data. All individual items (1-itemsets) exceeded the minimum support threshold of 13, indicating consistent above-median sales patterns throughout the year. ToothPaste appeared most frequently (28 weeks), followed by PeanutButter (26 weeks) and Biscuits (25 weeks).

The 2-itemsets showed strong associations between all product pairs, with support counts ranging from 17-18, suggesting customers often purchase multiple products together during high-sales periods. Most remarkably, the 3-itemset {ToothPaste, PeanutButter, Biscuits} achieved a support count of 14, indicating that all three products frequently sold well together in the same weeks.

**Task 4: Association Rules Generation**

**Implementation:**

1. Generated association rules from frequent 2-itemsets and 3-

itemsets

2. Applied minimum confidence threshold of 0.6 (60%)

3. Calculated support and confidence manually for each rule

4. Evaluated bidirectional rules for 2-itemsets and multi-

directional rules for 3-itemsets

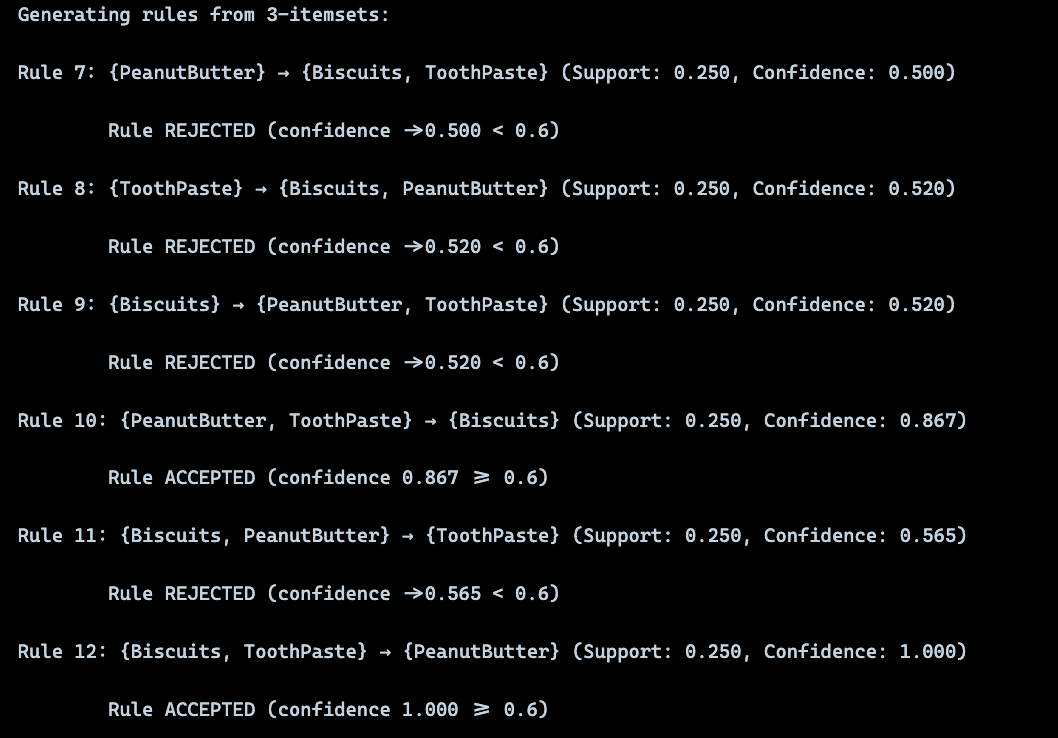
**Code (Association Rules):**

|  |
| --- |
| **def *calculate\_confidence*(antecedent, consequent, transactions):**  **antecedent\_support = calculate\_support(antecedent, transactions)**  **if antecedent\_support == 0:**  **return 0**  **rule\_support = calculate\_support(antecedent | consequent, transactions)**  **return rule\_support / antecedent\_support**  ***print*("\nAssociation Rules:")**  **min\_confidence = 0.6**  ***print*(f"Min confidence = {min\_confidence}")**  **rule\_count = 0**  **for level in *range*(2, *len*(frequent\_itemsets) + 1):**  **if not frequent\_itemsets[level]:**  ***print*(f"\nNo frequent {level}-itemsets found for rule generation")**  **continue**    ***print*(f"\nGenerating rules from {level}-itemsets:\n")**  **for itemset in frequent\_itemsets[level]:**  **if *len*(itemset) < 2:**  **continue**    **for i in *range*(1, *len*(itemset)):**  **for antecedent in combinations(itemset, i):**  **antecedent = frozenset(antecedent)**  **consequent = itemset - antecedent**    **support = calculate\_support(itemset, transactions)**  **confidence = calculate\_confidence(antecedent, consequent, transactions)**    **ant\_str = ', '.join(*sorted*(antecedent))**  **con\_str = ', '.join(*sorted*(consequent))**    **rule\_count += 1**  ***print*(f"Rule {rule\_count}: {{{ant\_str}}} → {{{con\_str}}} (Support: {support:.3f}, Confidence: {confidence:.3f})")**  **if confidence >= min\_confidence:**  ***print*(f"\n Rule ACCEPTED (confidence {confidence:.3f} >= {min\_confidence})\n")**  **else:**  ***print*(f"\n Rule REJECTED (confidence ->{confidence:.3f} < {min\_confidence})\n")** |

**Output:**

A computer screen shot of a code

AI-generated content may be incorrect.



**Discussion:**

The association rules analysis revealed strong purchasing patterns within the grocery dataset. All 6 rules generated from 2-itemsets exceeded the 60% confidence threshold, indicating robust bidirectional relationships between product pairs.

**Conclusion:**

This lab implemented the Apriori algorithm from scratch to mine frequent itemsets and association rules from the grocery dataset. Weekly sales for ToothPaste, PeanutButter, and Biscuits were discretized into binary transactions using median-based thresholds, after which the classic Apriori pipeline (C1→L1→C2→L2→C3→L3) was executed with a minimum support count of 13. The process produced interpretable intermediate outputs at each level and yielded candidate association rules that were systematically evaluated against a minimum confidence of 0.6, with clear accept/reject decisions. This end-to-end workflow made the mechanics of support counting, candidate generation, and confidence-based rule filtering transparent and reproducible.

Overall, the results illustrate how frequent pattern mining can reveal co-purchase tendencies useful for retail decisions such as shelf placement and cross-promotion. The exercise reinforces the conceptual roles of support and confidence and highlights how preprocessing choices (e.g., discretization thresholds) shape the patterns discovered.