# Association Rules Homework

Group Name: 5

Group Members:

* Ali Doroodchi
* Bhanu Hithesh Chaluvadhi
* Revant Reddy Dondeti
* Nikshitha Reddy Aella

1. **How can you apply association rules to your choice of data for your project?  If the data doesn't lend itself to association rules, describe a covid-19 data source that you can use with association rules (with some data preparation of course).**

**1. Choosing the Data**

Select a dataset that includes transactions or events with multiple items or occurrences recorded together. Examples include retail transaction data, healthcare patient records, e-commerce clickstream data, or survey responses.

**2. Data Preparation**

Prepare the dataset for the Apriori algorithm:

* **Clean the Data**: Remove any irrelevant or missing data.
* **Transform the Data**: Convert the dataset into a binary matrix format using One-Hot Encoding, where each row represents a transaction and each column represents an item, with cell values indicating the presence (1) or absence (0) of the item.

**3. Applying Apriori and Association Rules**

Use the mlxtend library in Python to apply the Apriori algorithm and extract association rules:

1. **Load the Data**:

python

Copy code

import pandas as pd

df = pd.read\_csv('your\_dataset.csv')

1. **One-Hot Encoding**:

python

Copy code

items = set()

for col in df:

items.update(df[col].unique())

encoded\_vals = []

for index, row in df.iterrows():

rowset = set(row)

labels = {item: 1 if item in rowset else 0 for item in items}

encoded\_vals.append(labels)

ohe\_df = pd.DataFrame(encoded\_vals)

1. **Apply Apriori Algorithm**:

python

Copy code

from mlxtend.frequent\_patterns import apriori

freq\_items = apriori(ohe\_df, min\_support=0.2, use\_colnames=True)

1. **Generate Association Rules**:

python

Copy code

from mlxtend.frequent\_patterns import association\_rules

rules = association\_rules(freq\_items, metric="confidence", min\_threshold=0.6)

1. **Visualize Results**:

python

Copy code

import matplotlib.pyplot as plt

plt.scatter(rules['support'], rules['confidence'], alpha=0.5)

plt.xlabel('Support')

plt.ylabel('Confidence')

plt.title('Support vs Confidence')

plt.show()

**COVID-19 Data Application**

For COVID-19 data, use a dataset with patient symptoms, test results, or contact tracing. Clean and transform the data similarly into a binary matrix format. Apply the Apriori algorithm to identify common symptom patterns or demographic associations. This can help uncover insights into symptom co-occurrence, testing results patterns, or other relevant public health trends

1. **What types of data preparation are necessary to ready your choice of data for association rules?**

To prepare your data for association rules, several crucial steps are necessary to ensure the dataset is suitable for the Apriori algorithm. Here are the key steps:

**1. Data Cleaning**

* **Remove Irrelevant Data**: Exclude columns or records that don't contribute to the analysis.
* **Handle Missing Values**: Decide on a strategy for missing data, such as removing records, imputing values, or filling with zeros for binary data.

**2. Data Transformation**

* **Convert to Transaction Format**: Structure your data so that each row represents a transaction and each column represents an item. For instance, in retail data, each row should list the items bought in a single transaction.
* **One-Hot Encoding**: Convert categorical data into a binary matrix where each column represents an item, and each cell indicates the presence (1) or absence (0) of that item in the transaction. This transformation is essential for the Apriori algorithm.

python

Copy code

import pandas as pd

# Example One-Hot Encoding

items = set()

for col in df:

items.update(df[col].unique())

encoded\_vals = []

for index, row in df.iterrows():

rowset = set(row)

labels = {item: (1 if item in rowset else 0) for item in items}

encoded\_vals.append(labels)

ohe\_df = pd.DataFrame(encoded\_vals)

**3. Consistent Data Types**

* **Binary Values**: Ensure the transformed data has only binary values (0 and 1) to comply with the Apriori algorithm's requirements.
* **Meaningful Column Names**: Use descriptive column names to represent items, facilitating easier interpretation of the results.

**4. Data Reduction (Optional)**

* **Filter Infrequent Items**: Depending on the dataset size and minimum support threshold, filter out items that appear very infrequently to focus on significant patterns and reduce computational complexity.

**Example Implementation**

Here’s a concise example using Python:

python

Copy code

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

# Load and encode dataset

df = pd.read\_csv('retail\_dataset.csv')

items = set(df.values.flatten()) - {None}

encoded\_vals = [{item: (1 if item in row else 0) for item in items} for row in df.values]

ohe\_df = pd.DataFrame(encoded\_vals)

# Apply Apriori algorithm

freq\_items = apriori(ohe\_df, min\_support=0.2, use\_colnames=True)

# Generate association rules

rules = association\_rules(freq\_items, metric="confidence", min\_threshold=0.6)

print(rules.head())

Data preparation for association rules involves cleaning the data, transforming it into a binary matrix format, ensuring consistent data types, and optionally filtering out infrequent items. These steps are crucial for applying the Apriori algorithm effectively and deriving meaningful association rules.

**c)How would you plan to limit the consequent to get the types of rules that can be useful?**

To limit the consequent and obtain useful association rules, you can use several strategies:

**1. Specify Target Consequents**

Identify specific items or categories that are of particular interest. For example, in retail, focus on high-margin or promotional products. This ensures the rules generated are relevant to your business objectives.

**2. Post-Processing Filtering**

Generate all possible rules and filter them based on the consequent. This approach provides flexibility and allows you to refine your rules based on specific needs.

**3. Domain Knowledge**

Leverage your understanding of the domain to pre-select items likely to be useful as consequents. For instance, in healthcare, focus on rules where the consequent is a critical diagnosis or high-cost treatment.

**4. Rule Generation with Constraints**

Directly apply constraints during rule generation to limit the consequents. This can be done using libraries like mlxtend in Python.

**Example Using Python and mlxtend**

**Step-by-Step Implementation**

1. **Load and Prepare Data**:

python

Copy code

import pandas as pd

from mlxtend.frequent\_patterns import apriori, association\_rules

# Load dataset

df = pd.read\_csv('retail\_dataset.csv')

# One-Hot Encoding

items = set(df.values.flatten()) - {None}

encoded\_vals = [{item: (1 if item in row else 0) for item in items} for row in df.values]

ohe\_df = pd.DataFrame(encoded\_vals)

1. **Generate Frequent Itemsets**:

python

Copy code

freq\_items = apriori(ohe\_df, min\_support=0.2, use\_colnames=True)

1. **Generate Association Rules**:

python

Copy code

rules = association\_rules(freq\_items, metric="confidence", min\_threshold=0.6)

1. **Filter Rules by Consequent**:

python

Copy code

# Define target consequents

target\_consequents = ['Milk', 'Bread']

# Filter rules

filtered\_rules = rules[rules['consequents'].apply(lambda x: any(item in x for item in target\_consequents))]

print(filtered\_rules)

1. **Further Filter Based on Domain Knowledge**:

python

Copy code

# Example: Filtering by additional metrics such as lift

useful\_rules = filtered\_rules[filtered\_rules['lift'] > 1.2]

print(useful\_rules)

**Summary**

To limit the consequent for useful association rules:

* Specify target items or categories.
* Filter rules post-generation based on the consequent.
* Use domain knowledge to select meaningful consequents.
* Apply constraints during rule generation to limit the consequents directly.

These strategies ensure the resulting rules are relevant, actionable, and aligned with your analytical goals.

**d) What support and confidence will be used for good results?**

Choosing appropriate support and confidence thresholds is crucial for obtaining meaningful association rules through the Apriori algorithm. These thresholds directly impact the quality and relevance of the rules generated. Here’s a breakdown of considerations and guidelines for setting these parameters effectively:

**Support:**

**Support** measures the frequency of occurrence of an itemset in the dataset. It helps in identifying itemsets that are sufficiently frequent to be considered for association rule mining.

* **High Support**: Rules with high support are derived from itemsets that appear frequently in transactions. This ensures that the rules are based on solid patterns in the data, making them more reliable and applicable.
* **Low Support**: Allows for the discovery of less frequent but potentially interesting associations. However, setting support too low may result in an overwhelming number of rules or include associations that are not significant.

A typical starting point for support is between **0.1 (10%) to 0.5 (50%)**. This range balances between capturing frequent patterns and avoiding too many trivial rules.

**Confidence:**

**Confidence** measures the conditional probability that a rule is true, given the antecedent. It indicates the strength of the association between antecedent and consequent.

* **High Confidence**: Rules with high confidence are more likely to be true and actionable. They represent strong associations where the presence of the antecedent implies a high likelihood of the consequent.
* **Low Confidence**: Rules with lower confidence may still be interesting but might not be as reliably predictive. Adjusting confidence helps in filtering out weaker associations.

A common starting point for confidence is between **0.6 (60%) to 0.8 (80%)**. This ensures that the rules generated are sufficiently strong while allowing for flexibility in the relationships between items.

**Practical Application:**

In Python using mlxtend, you can apply these thresholds as follows:

python

Copy code

from mlxtend.frequent\_patterns import apriori, association\_rules

# Load dataset and perform one-hot encoding

# Apply Apriori algorithm with specified support and confidence thresholds

freq\_items = apriori(ohe\_df, min\_support=0.2, use\_colnames=True)

rules = association\_rules(freq\_items, metric="confidence", min\_threshold=0.7)

# Analyze and interpret the generated association rules

print(rules.head())

**Iterative Adjustment:**

* Start with moderate thresholds (e.g., support=0.2, confidence=0.7) to explore initial patterns.
* Adjust thresholds based on the number and relevance of the rules obtained:
  + Increase support to focus on more frequent itemsets.
  + Increase confidence to filter out weaker associations.

By iteratively adjusting these thresholds based on the specific characteristics of your dataset and the insights you seek, you can derive meaningful association rules that are relevant and actionable for your analysis.

**e) How does the concept of lift apply to the evaluation of your rules?**

Lift is a crucial metric for evaluating association rules in data mining, providing insight into the strength and significance of the relationships between items. It helps to determine how much more likely the consequent of a rule is to occur given the antecedent, compared to its baseline occurrence. Here's how lift applies to rule evaluation:

**Definition of Lift**

**Lift** is calculated using the formula: Lift(A→B)=Confidence(A→B)Support(B)\text{Lift}(A \rightarrow B) = \frac{\text{Confidence}(A \rightarrow B)}{\text{Support}(B)}Lift(A→B)=Support(B)Confidence(A→B)​ where:

* **Confidence(A → B)** is the probability that item B is purchased given that item A is purchased.
* **Support(B)** is the probability that item B is purchased overall.

**Interpretation of Lift**

* **Lift > 1**: Indicates a positive association between A and B. The occurrence of A increases the likelihood of B occurring, suggesting a strong, positive relationship.
* **Lift = 1**: Implies that A and B are independent. The occurrence of A does not affect the probability of B.
* **Lift < 1**: Suggests a negative association, meaning A and B are less likely to be purchased together than independently. This could indicate that the presence of A actually reduces the likelihood of B.

**Practical Use**

In practice, lift helps to filter and prioritize rules based on their strength. Rules with high lift values are particularly valuable because they reveal stronger and more significant relationships between items. For instance, in market basket analysis, a rule with high lift might suggest that when customers buy bread, they are much more likely to also buy butter, indicating a strong relationship between these products.

**Example in Python**

Using mlxtend to compute and evaluate lift:

python

Copy code

from mlxtend.frequent\_patterns import apriori, association\_rules

# Generate frequent itemsets

freq\_items = apriori(ohe\_df, min\_support=0.2, use\_colnames=True)

# Generate association rules

rules = association\_rules(freq\_items, metric="lift", min\_threshold=1.0)

# Display rules with high lift

print(rules[rules['lift'] > 1.5])

**Summary**

Lift is an essential metric for assessing the strength of association rules, highlighting significant and actionable relationships between items. By focusing on rules with high lift values, you can identify meaningful patterns and relationships in your data, enhancing the relevance and utility of the insights derived from association rule mining.