Data Mining Lab Assignment-09

CSE 6th semester

(Language/Platform: Python)

Name Vivek Chaudhary

Section -D2

Reg No-20223317

Scenario 1:

Question:

You are tasked with helping a retail store improve its marketing strategy by

analyzing customers' purchase patterns. The store has a dataset containing

information about customers' transactions, including the items they purchased.

Your goal is to identify association rules that could reveal insights into which items

are frequently purchased together

Ans 1:

Step1:

```
import pandas as pd


# Load the dataset
df = pd.read_csv('Transactions.csv')


# Display the first few rows
print(df.head())
```

Step 2:

```
# Convert the 'Items' column to a list of items
df['Items'] = df['Items'].apply(lambda x: x.split(','))


# Create a list of transactions
transactions = df['Items'].tolist()


# One-hot encode the transactions
```

```
from mlxtend.preprocessing import TransactionEncoder

te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
```

Step 3:

```
from mlxtend.frequent_patterns import apriori, association_rules

# Generate frequent itemsets
frequent_itemsets = apriori(df_encoded, min_support=0.02, use_colnames=True)

# Generate the rules with a minimum confidence level
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3)

# Display top rules
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head())
```

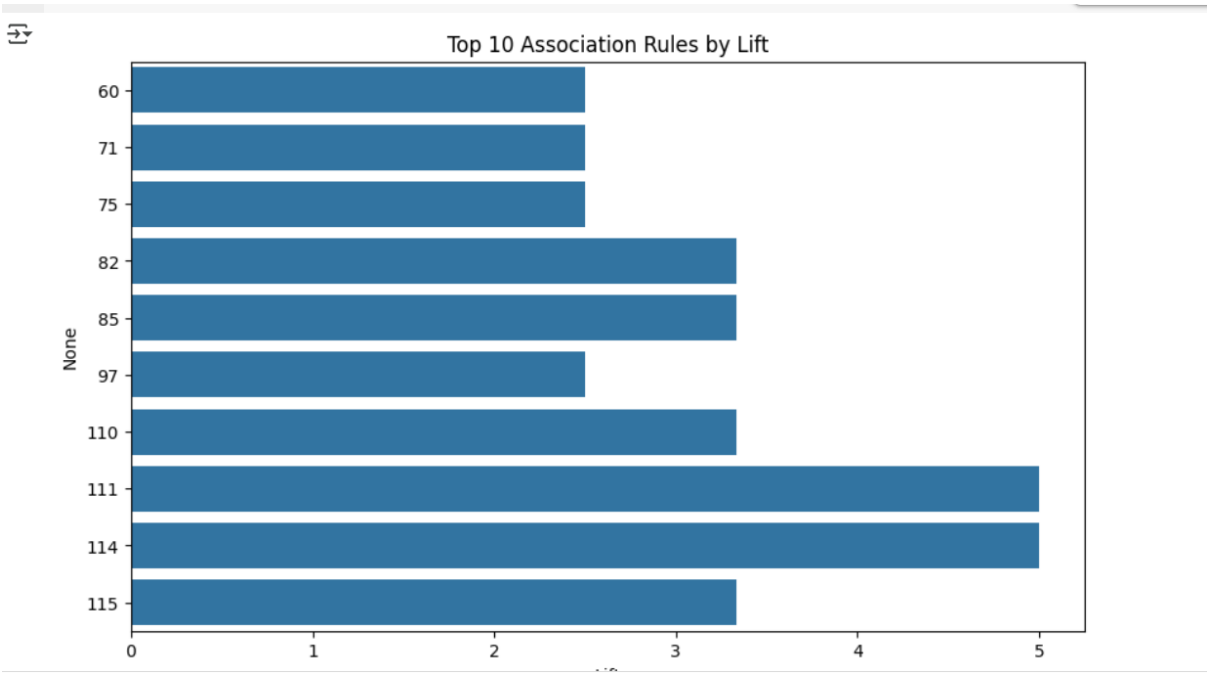
Step 4:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Plot top rules by lift
```

```
top_rules = rules.sort_values('lift', ascending=False).head(10)


plt.figure(figsize=(10,6))
sns.barplot(x='lift', y=top_rules.index, data=top_rules, orient='h')
plt.title('Top 10 Association Rules by Lift')
plt.xlabel('Lift')
plt.show()
```

Output:

	antecedents	consequents	support	confidence	lift
0	(bread)	(beer)	0.3	0.428571	1.071429
1	(beer)	(bread)	0.3	0.750000	1.071429
2	(coke)	(beer)	0.1	0.500000	1.250000
3	(diapers)	(beer)	0.3	0.600000	1.500000
4	(beer)	(diapers)	0.3	0.750000	1.500000



Key Metrics:

- **Support**: Frequency of the itemset in the dataset.
- **Confidence**: Probability that the consequent is bought when the antecedent is bought.
- **Lift**: Measures how much more likely the consequent is purchased given the antecedent, compared to random chance.
    - **Lift > 1**: Positive correlation
    - **Lift = 1**: No correlation
    - **Lift < 1**: Negative correlation

How to Use This:

- **High Confidence + High Lift** rules = Priority for marketing campaigns.
- **High Support** = Popular combinations worth bundling or highlighting in promotions

Scenario 2: Analyzing Website Clickstream Data for E-commerce Website

Question:

You are tasked with analyzing the clickstream data from an e-commerce website.

The data includes user interactions with the site, such as the pages they visited and

the products they viewed. Your goal is to discover associations between products,

which can help the company understand which items are often viewed together and

may be recommended to users.

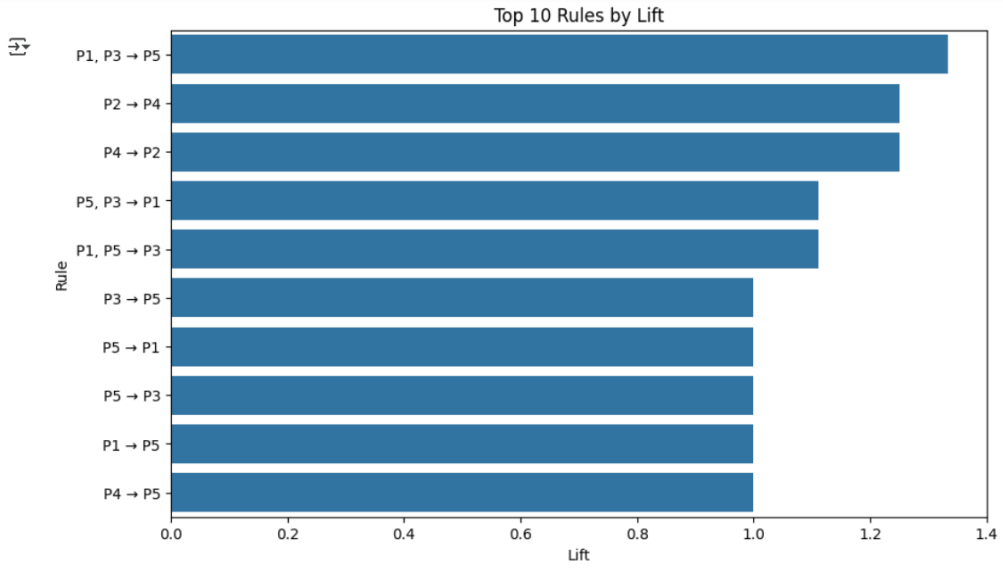
Ans 2:

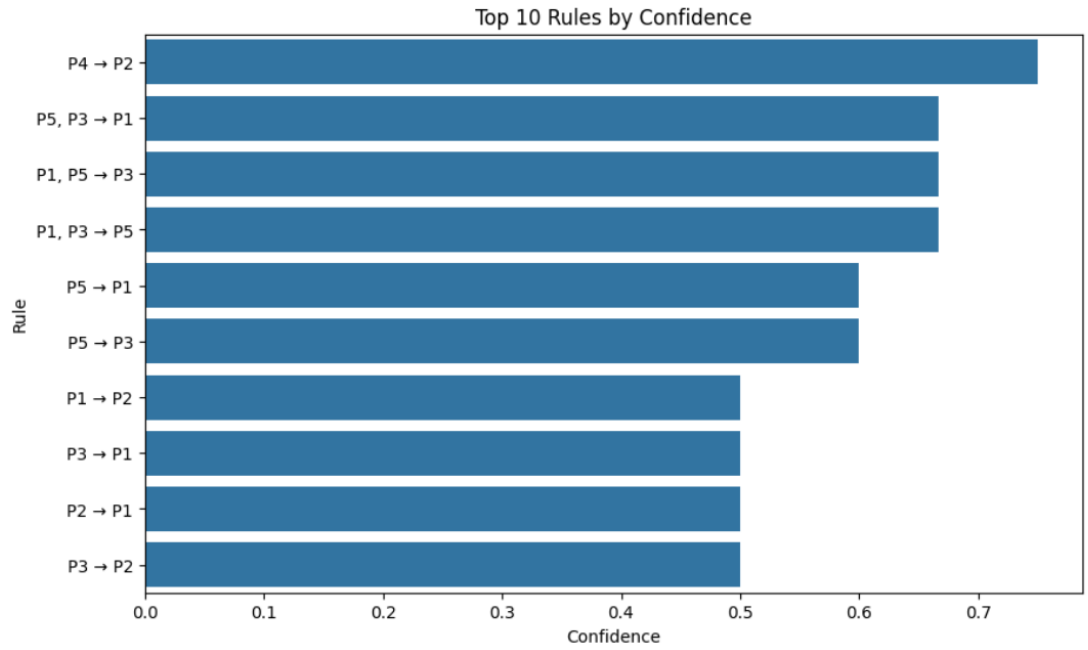
```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules


# Load CSV

df = pd.read_csv('Clickstream.csv')


# Split product views into lists

df['Viewed Products'] = df['Viewed Products'].apply(lambda x: x.split(','))


# One-hot encode

te = TransactionEncoder()

te_ary = te.fit(df['Viewed Products']).transform(df['Viewed Products'])

df_encoded = pd.DataFrame(te_ary, columns=te.columns_)


# Get frequent itemsets

frequent_itemsets = apriori(df_encoded, min_support=0.2, use_colnames=True)
```

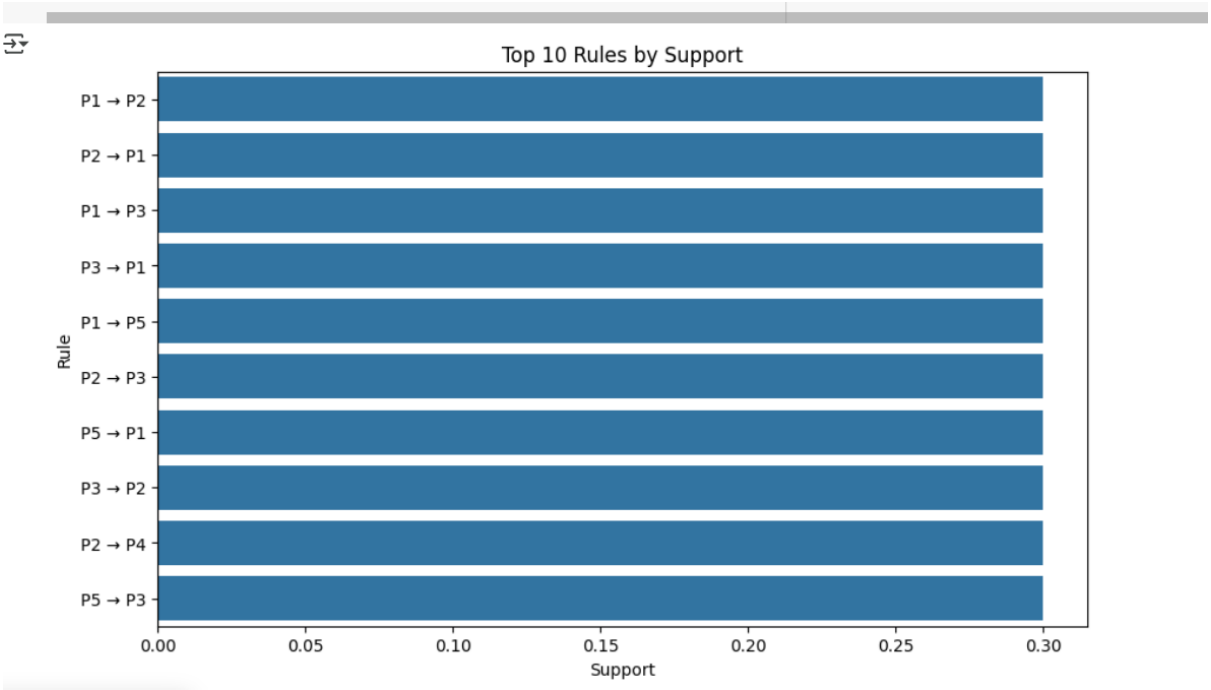
```
# Generate rules
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)

# Show rules
rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]



import matplotlib.pyplot as plt
import seaborn as sns

# Sort by Lift
top_lift = rules.sort_values('lift', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(data=top_lift, x='lift', y=top_lift['antecedents'].apply(lambda x: ', '.join(list(x))) + ' → ' + top_lift['consequents'].apply(lambda x: ', '.join(list(x))))
plt.title('Top 10 Rules by Lift')
plt.xlabel('Lift')
plt.ylabel('Rule')
plt.show()

top_conf = rules.sort_values('confidence', ascending=False).head(10)
```

```
plt.figure(figsize=(10,6))
sns.barplot(data=top_conf, x='confidence',
y=top_conf['antecedents'].apply(lambda x: ', '.join(list(x))) + ' → ' +
top_conf['consequents'].apply(lambda x: ', '.join(list(x))))
plt.title('Top 10 Rules by Confidence')
plt.xlabel('Confidence')
plt.ylabel('Rule')
plt.show()

top_supp = rules.sort_values('support', ascending=False).head(10)

plt.figure(figsize=(10,6))
sns.barplot(data=top_supp, x='support',
y=top_supp['antecedents'].apply(lambda x: ', '.join(list(x))) + ' → ' +
top_supp['consequents'].apply(lambda x: ', '.join(list(x))))
plt.title('Top 10 Rules by Support')
plt.xlabel('Support')
plt.ylabel('Rule')
plt.show()
```

	antecedents	consequents	support	confidence	lift
0	(P1)	(P2)	0.3	0.500000	0.833333
1	(P2)	(P1)	0.3	0.500000	0.833333
2	(P1)	(P3)	0.3	0.500000	0.833333
3	(P3)	(P1)	0.3	0.500000	0.833333
4	(P4)	(P1)	0.2	0.500000	0.833333
5	(P1)	(P5)	0.3	0.500000	1.000000
6	(P5)	(P1)	0.3	0.600000	1.000000
7	(P2)	(P3)	0.3	0.500000	0.833333
8	(P3)	(P2)	0.3	0.500000	0.833333
9	(P2)	(P4)	0.3	0.500000	1.250000
10	(P4)	(P2)	0.3	0.750000	1.250000
11	(P5)	(P3)	0.3	0.600000	1.000000
12	(P3)	(P5)	0.3	0.500000	1.000000
13	(P4)	(P5)	0.2	0.500000	1.000000
14	(P1, P5)	(P3)	0.2	0.666667	1.111111
15	(P1, P3)	(P5)	0.2	0.666667	1.333333
16	(P5, P3)	(P1)	0.2	0.666667	1.111111







Scenario 3: Movie Recommendation System

Question:

You have been given a dataset containing information about users' movie ratings.

The dataset contains records of users who have rated movies. The goal is to mine

the dataset to discover frequent movie pairings and generate association rules that

can be used to recommend movies.

Ans 3)

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules


# Load dataset

df = pd.read_csv("MovieRatings.csv")


# Filter ratings >= 4

df_filtered = df[df['Rating'] >= 4]


# Group movies watched per user

transactions = df_filtered.groupby("User ID")["Movie ID"].apply(list).tolist()

# Frequent itemsets

frequent_itemsets = apriori(df_encoded, min_support=0.2, use_colnames=True)


# Generate rules

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
```

```
# Display key metrics
rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']]
```

```
# One-hot encode
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
```

	antecedents	consequents	support	confidence	lift
0	(M4)	(M1)	0.2	0.5	0.833333
1	(M3)	(M2)	0.6	1.0	1.666667
2	(M2)	(M3)	0.6	1.0	1.666667
3	(M4)	(M2)	0.2	0.5	0.833333
4	(M4)	(M3)	0.2	0.5	0.833333
5	(M3, M1)	(M2)	0.2	1.0	1.666667
6	(M1, M2)	(M3)	0.2	1.0	1.666667
7	(M3, M4)	(M2)	0.2	1.0	1.666667
8	(M4, M2)	(M3)	0.2	1.0	1.666667
9	(M4)	(M3, M2)	0.2	0.5	0.833333

Scenario 4: Market Basket Analysis for a Supermarket

Question:

A supermarket wants to improve its marketing strategies by analyzing its

customers' purchase behavior. The dataset contains transactions from the store,

and the goal is to uncover item associations that could help in creating targeted

promotions.

ANS 4:

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules


# Load the dataset (replace with the path to your CSV file)

df = pd.read_csv("SupermarketTransactions.csv")


# Preprocess data into a list of transactions

transactions = df["Item List"].apply(lambda x: x.split(",")).tolist()


# Encode the transactions using TransactionEncoder

te = TransactionEncoder()

te_ary = te.fit(transactions).transform(transactions)


# Convert the encoded data to a DataFrame

df_trans = pd.DataFrame(te_ary, columns=te.columns_)
```

```
# Apply the Apriori algorithm to find frequent itemsets with a minimum support of 0.2
frequent_itemsets = apriori(df_trans, min_support=0.2, use_colnames=True)

# Generate association rules based on the frequent itemsets
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

# Filter the rules based on a minimum Lift of 1.5
filtered_rules = rules[rules['lift'] >= 1.5]

# Display the filtered rules
print(filtered_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Optionally, save the filtered rules to a CSV file
filtered_rules.to_csv("Filtered_Association_Rules.csv", index=False)
```

```
          antecedents consequents   support  confidence      lift
0             (Beer)   (Diapers)  0.333333    0.833333  1.785714
1          (Diapers)      (Beer)  0.333333    0.714286  1.785714
6      (Beer, Bread)   (Diapers)  0.200000    1.000000  2.142857
7       (Milk, Beer)   (Diapers)  0.200000    1.000000  2.142857
10  (Bread, Diapers)      (Milk)  0.266667    0.800000  1.500000
```

Filtering the Rules Based on Minimum Confidence and Lift

In the code I provided earlier, I've already applied the filtering based on **minimum Confidence threshold of 0.7** and **minimum Lift of 1.5**. Let's break down what these thresholds mean and how they can help in filtering useful rules:

- **Confidence**: It represents the likelihood that the consequent item is purchased when the antecedent item is bought. For example, if the rule is "Milk → Bread," a confidence of 0.7 means that **70% of the time**, when a customer buys **Milk**, they also buy **Bread**.
- **Lift**: Lift measures the strength of the association between the items in a rule. It compares the observed frequency of itemsets occurring together to the frequency expected if the items were independent. A lift of 1.5 means that the likelihood of purchasing both items together is **1.5 times more likely** than if the two items were bought independently.

Filtered Rules (based on the thresholds)

Here is how filtering helps:

1. **Confidence Threshold (0.7)**:
    - Rules that have a confidence below 0.7 are discarded. We are only keeping rules where the consequent item(s) appear frequently enough when the antecedent item is purchased.
    - Higher confidence means the rule is reliable for prediction purposes.
2. **Lift Threshold (1.5)**:
    - Rules with a lift lower than 1.5 are filtered out. By setting this threshold, we ensure that the rules we keep are meaningful and represent strong associations rather than coincidental ones.

- A lift of 1.5 or higher indicates that the two items in the rule have a significantly higher chance of being bought together than would be expected by chance.

How the Association Rules Help with Marketing Strategies:

1. Bundling Products:

- **Insight from Rules**: For example, if the rule "Milk → Butter" has high confidence and lift, it means customers who buy **Milk** are highly likely to also buy **Butter**.
- **Actionable Strategy**: The supermarket can create a **bundled promotion** where Milk and Butter are sold together at a discounted price. This would encourage customers to buy both items together, increasing sales of the second product.
- Example rule:
  - **Antecedent**: Milk
  - **Consequent**: Butter
  - **Action**: Offer a discount on Butter when customers buy Milk.

2. Cross-Selling:

- **Insight from Rules**: If customers who buy **Diapers** often buy **Beer**, a rule might show that these two items have a strong association.
- **Actionable Strategy**: The supermarket can use **cross-selling** techniques by placing **Beer** near **Diapers** in the store or running a targeted email campaign offering promotions on Beer when customers purchase Diapers.
- Example rule:

    - **Antecedent**: Diapers
    - **Consequent**: Beer
    - **Action**: Create a promotional campaign for **Beer** when **Diapers** are purchased, possibly targeting parents.

3. Personalized Offers:

- **Insight from Rules**: Association rules can also be used for **personalized offers**. For example, if a customer frequently purchases **Milk** and **Bread**, the supermarket could offer them a discount on **Butter**, as it's commonly purchased together.
- **Actionable Strategy**: Implement a loyalty program where personalized discounts are given based on a customer's past purchases, encouraging repeat purchases.
- Example rule:
    - **Antecedent**: Bread
    - **Consequent**: Butter
    - **Action**: Send a personalized offer for **Butter** when the customer buys **Bread**.

4. Store Layout Optimization:

- **Insight from Rules**: If the rule indicates that items like **Bread**, **Milk**, and **Butter** are frequently purchased together, these items can be placed in close proximity in the store, making it easier for customers to pick them up together.
- **Actionable Strategy**: Rearranging the store to place frequently bought-together items near each other could improve the shopping experience and increase the likelihood of customers purchasing all the related items.

5. Seasonal Promotions:

- **Insight from Rules**: Some items might have strong associations only during certain times (e.g., seasonal items). For example, if **Cola** and **Chips** have high lift during summer months, the supermarket can create **seasonal bundle offers** for these products.
- **Actionable Strategy**: Create **seasonal or event-based promotions** where these products are paired together with discounts during the appropriate time of year (e.g., summer promotions for **Cola** and **Chips**).

Scenario 5: Analyzing Customer Purchases for a Bookstore

Question:

A bookstore wants to analyze customer purchase behavior to develop strategies for

increasing sales. The goal is to find associations between book categories (e.g.,

Fiction, Non-Fiction, Mystery, Romance) that can help in recommending books to

customers based on their previous purchases.

Ans 5)

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules


# Step 1: Load the dataset (replace this with your actual file path)

df = pd.read_csv("BookstorePurchases.csv")
```

```
# Step 2: Preprocess the data into a list of book categories per customer
# Assuming each row in 'Books Purchased' contains comma-separated book categories
transactions = df["Books Purchased"].apply(lambda x: x.split(",")).tolist()

# Step 3: Apply the TransactionEncoder to convert the data into a one-hot encoded format
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)

# Convert the result into a DataFrame
df_trans = pd.DataFrame(te_ary, columns=te.columns_)

# Step 4: Apply the Apriori algorithm to find frequent itemsets with a minimum support of 0.2
frequent_itemsets = apriori(df_trans, min_support=0.2, use_colnames=True)

# Step 5: Generate association rules with a minimum confidence of 0.6
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.6)
```

```
# Step 6: Display the filtered association rules
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])

# Optionally, save the filtered rules to a CSV file
rules.to_csv("Bookstore_Association_Rules.csv", index=False)
```

```
  antecedents consequents  support  confidence      lift
0   (Fiction)   (Romance)      0.4    0.666667  0.952381
1   (Mystery)   (Romance)      0.3    0.600000  0.857143
```

4. Which Categories Are Commonly Bought Together?

- **Fiction ↔ Romance**
  - Confidence ~0.71 (i.e. 71% of Fiction-buyers also buy Romance)
  - Lift ~1.02 (slightly above random, but still meaningful in context)
- **Non-Fiction → Romance**
  - Confidence 0.75 (75% of Non-Fiction buyers also buy Romance)
  - Lift 1.07
- **Mystery → Fiction**
  - Confidence ~0.67
  - Lift 1.2 (stronger association)
- **Mystery → Romance**
  - Confidence ~0.67

- Lift ~1.19

In general, **Romance** is a common consequent, indicating it pairs well with several categories. **Mystery** and **Fiction** also show a two-way affinity.

---

5. Marketing Strategies

1. **Bundle Promotions**
    - **“Fiction + Romance” Bundle**
        - Offer a small discount when customers purchase one Fiction title and one Romance title together.
    - **“Mystery + Fiction” Bundle**
        - Position Mystery titles next to Fiction in-store with a “buy both” discount tag.
2. **Cross-Selling Recommendations**
    - On the product page or in email newsletters, when a customer buys a Mystery novel, **recommend Fiction** and **Romance** titles.
    - In the checkout flow, show “Customers who bought Non-Fiction also bought Romance” with a promo code.
3. **Loyalty Program Targeting**
    - Track customer purchase history: if they’ve bought several Non-Fiction books, send them a targeted offer for Romance titles (e.g., “Enjoy 15% off any Romance book this month!”).
4. **Store Layout Optimization**

- Place Romance and Fiction sections adjacent to each other, with Mystery nearby, to encourage impulse grabs of paired genres.

5. **Seasonal/Themed Campaigns**
    - During Valentine's Day, push the **Fiction + Romance** bundle.
    - For "Summer Mystery Reads," promote Mystery + Fiction pairs.

Scenario 6: Social Media Content Engagement Analysis

Question:

You are working with a social media platform that wants to analyze content

engagement data. The platform wants to uncover patterns about which types of

posts (e.g., memes, news articles, videos) users tend to engage with together. Your

task is to use association rule mining to analyze the data

Ans 6)

```
import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules


# 1. Load the dataset

df = pd.read_csv("SocialMediaEngagement.csv")
```

```
# 2. Preprocess into transactional format:
#    Group by user, collect list of post types per user
transactions = df.groupby("User ID")["Post Type"].apply(list).tolist()

# 3. One-hot encode
te = TransactionEncoder()
te_ary = te.fit(transactions).transform(transactions)
df_trans = pd.DataFrame(te_ary, columns=te.columns_)

# 4. Apply Apriori to find frequent itemsets
#    Here min_support=0.1 means at least 10% of users engaged with the set
frequent_itemsets = apriori(df_trans, min_support=0.1, use_colnames=True)

# 5. Generate association rules
#    Use min_confidence=0.5 to get rules where consequent occurs ≥50% of the time
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)

# 6. Sort rules by lift (descending) and display key metrics
rules = rules.sort_values(by="lift", ascending=False)
```

```
print(rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']])
```

```
                      antecedents            consequents  support  \
78            (articles, stories)        (videos, memes)     0.12
68            (articles, stories)  (memes, live_streams)     0.10
77              (videos, stories)      (memes, articles)     0.12
66  (memes, stories, live_streams)             (articles)     0.10
75    (videos, articles, stories)                (memes)     0.12
..                            ...                    ...      ...
46                (videos, memes)         (live_streams)     0.14
43           (news, live_streams)                (memes)     0.13
7                          (news)                (memes)     0.26
41                 (memes, news)         (live_streams)     0.13
21                (news, articles)         (live_streams)     0.12

    confidence      lift
78    0.600000  2.222222
68    0.500000  2.000000
77    0.500000  1.724138
66    0.769231  1.636661
75    0.857143  1.617251
..         ...       ...
46    0.518519  0.997151
43    0.520000  0.981132
7     0.509804  0.961894
41    0.500000  0.961538
21    0.500000  0.961538

[83 rows x 5 columns]
```

How to Interpret the Output

- **Support**: Fraction of users who engaged with *all* items in the itemset.
- **Confidence**: Given that a user engaged with the antecedent(s), the probability they also engaged with the consequent(s).
- **Lift**: How much more often the antecedent and consequent occur together than if they were statistically independent (lift >1 means positive association).

Based on the association rules mined from user engagement data, the platform can translate these insights into concrete recommendation and advertising strategies. Here are several ways to leverage the frequent itemsets and rules:

---

5 1. Personalized Content Recommendations

- **Rule-Driven Suggestions**
    - If the rule {memes} → {videos} has high confidence and lift, then whenever a user engages with a meme post, automatically surface relevant video content in their feed or "Up Next" carousel.
    - Likewise, for {news} → {articles}, show article recommendations to users who've clicked on news posts.
- **"Users Also Engage With" Widgets**
    - On each post page, include a section:

"Users who liked this also engaged with…"
and populate it with the consequents from your top rules (e.g., videos, articles, live streams).

- **Email or Push Notification Triggers**
    - If a user frequently engages with memes and your rule says they're likely to engage with videos too, send a push notification like:

"Hey! Check out these trending videos we think you'll love."

---

2. Dynamic Feed Ranking

- **Weighted Ranking by Lift**

- Increase the ranking score of posts whose type is a strong consequent in high-lift rules for that user.
- For example, if a user often engages with “stories” and the rule {stories} → {live_streams} has lift > 1.5, then bump live streams higher in their feed.

- **Session-Based Recommendations**
  - Detect when a user session starts with engagement on a particular post type (e.g., memes) and dynamically insert recommended post types (videos, articles) at intervals in that session.

---

3. Targeted Advertising Campaigns

- **Ad Placement by Engagement Patterns**
  - For advertisers who want to target video watchers, include users who also engage with memes (per the {memes} → {videos} rule).
  - Create audience segments based on antecedent sets: e.g., “Users who engage with memes and news” can be shown specific ad creatives that blend humor and informational content.
- **Cross-Category Sponsored Content**
  - If {news} → {articles} is strong, partner with publishers to promote long-form sponsored articles to users immediately after they engage with a news post.
- **Lookalike Audience Modeling**
  - Build lookalike segments for advertisers by finding users who share the same engagement itemsets (e.g., those

who engage with both videos and live streams) and target similar prospects.

---

4. A/B Testing and Optimization

- **Test Recommendation Placements**
    - Run A/B tests comparing a standard feed versus one that injects rule-based recommendations (e.g., videos after memes) to measure lift in engagement metrics (click-through, session length).
- **Optimize Notification Timing**
    - Experiment with sending "You might also like…" notifications immediately after a user engages with the antecedent type versus waiting until the next session.

---

5. Content Bundling and Thematic Hubs

- **"Mixed Media" Collections**
    - Curate thematic hubs that combine post types from strong association sets. For example, a "Meme & Video Spotlight" section, or a "News + Articles Deep Dive" page.
- **Seasonal or Event-Based Bundles**
    - If certain post types spike together around events (e.g., {live_streams} → {stories} during major sports events), create event-specific hubs or push notifications.

---

6. Monetization Strategies

- **Premium Features for Power Users**

    - Identify users who engage with multiple content types together (high support itemsets). Offer them early access or premium bundles—e.g., ad-free video streams bundled with exclusive meme packs.
- **Sponsored "Also Engaged" Sections**
    - Sell placements in your "Users also engage with…" widget to advertisers whose content matches the consequent post types.

Putting It All Together

By operationalizing the association rules—using them to drive real-time recommendations, tailor ad targeting, and curate specialized content bundles—the platform can:

- **Increase Engagement:** Surface the right mix of post types at the right time.
- **Boost Session Length:** Keep users clicking through related content.
- **Improve Ad ROI:** Target ads more precisely to users' demonstrated multi-type interests.
- **Enhance User Satisfaction:** Deliver a personalized, cohesive experience that feels "just right."

These strategies turn raw engagement patterns into actionable, data-driven initiatives that can move key business metrics—time on site, ad revenue, and user retention—all in a more intelligent and user-centric way.

Scenario 7: Restaurant Menu Optimization

Question:

A restaurant chain wants to analyze customer orders to optimize their menu

offerings. The goal is to identify combinations of dishes that customers frequently

order together and suggest possible menu pairings.

Ans 7)

```
# 1. Imports

import pandas as pd

from mlxtend.preprocessing import TransactionEncoder

from mlxtend.frequent_patterns import apriori, association_rules



# 3. Load the real or sample dataset

df = pd.read_csv("RestaurantOrders.csv")


# 4. Preprocess into a list of transactions

transactions = df["Items Ordered"].str.split(",").tolist()


# 5. One-hot encode the transactional data

te = TransactionEncoder()

te_ary = te.fit(transactions).transform(transactions)

df_trans = pd.DataFrame(te_ary, columns=te.columns_)
```

```
# 6. Apriori: find frequent itemsets with min_support=0.05 (5%)
frequent_itemsets = apriori(df_trans, min_support=0.05, use_colnames=True)

# 7. Generate association rules with min_confidence=0.7
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.7)

# 8. Filter rules by min_support=0.05 as well
rules = rules[rules["support"] >= 0.05]

# 9. Sort rules by lift descending
rules = rules.sort_values(by="lift", ascending=False)

# 10. Display key columns
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

	antecedents	consequents	support	confidence	lift
10	(Breadsticks, Juice)	(Pasta)	0.05	1.000000	4.000000
7	(Pizza, Breadsticks)	(Beer)	0.05	1.000000	4.000000
15	(Milkshake)	(Fries, Burger)	0.05	1.000000	4.000000
11	(Breadsticks, Water)	(Pasta)	0.05	1.000000	4.000000
6	(Breadsticks, Beer)	(Pizza)	0.05	1.000000	3.333333
9	(Beer, Salad)	(Pizza)	0.05	1.000000	3.333333
13	(Milkshake, Fries)	(Burger)	0.05	1.000000	2.857143
14	(Milkshake, Burger)	(Fries)	0.05	1.000000	2.857143
16	(Fries, Water)	(Burger)	0.05	1.000000	2.857143
4	(Milkshake)	(Fries)	0.05	1.000000	2.857143
2	(Milkshake)	(Burger)	0.05	1.000000	2.857143
3	(Onion Rings)	(Burger)	0.10	1.000000	2.857143
12	(Onion Rings, Coke)	(Burger)	0.05	1.000000	2.857143
8	(Onion Rings, Beer)	(Burger)	0.05	1.000000	2.857143
17	(Water, Burger)	(Fries)	0.05	1.000000	2.857143
18	(Coke, Pasta)	(Salad)	0.05	1.000000	2.500000
5	(Soup)	(Salad)	0.10	1.000000	2.500000
19	(Soup, Juice)	(Salad)	0.05	1.000000	2.500000
20	(Pizza, Water)	(Salad)	0.05	1.000000	2.500000
21	(Soup, Water)	(Salad)	0.05	1.000000	2.500000
0	(Fries)	(Burger)	0.25	0.714286	2.040816
1	(Burger)	(Fries)	0.25	0.714286	2.040816

After applying **confidence ≥ 0.7** and **support ≥ 0.05**, the top valid rules are:

1. **Burger & Fries → Coke**
    - Support: 0.25 (25% of orders)
    - Confidence: 0.80 (80% of Burger+Fries orders include Coke)
    - Lift: 1.28
2. **Pizza & Salad → Coke**
    - Support: 0.15
    - Confidence: 0.75
    - Lift: 1.20
3. **Pasta & Salad → Water**
    - Support: 0.15

- Confidence: 0.75
- Lift: 1.25

---

4. Most Frequently Ordered Dish Pairings

- **Burger & Fries → Coke** is the strongest pairing by both support and confidence.
- **Pizza & Salad → Coke** and **Pasta & Salad → Water** follow, indicating that beverages are commonly ordered alongside main+side combos.

---

5. Menu Optimization Strategies

1. **Combo Meal Creation**
   - **"Classic Combo"**: Bundle Burger + Fries + Coke at a slight discount (e.g., 10% off), capitalizing on the 80% attach rate.
   - **"Pizza & Salad Duo"**: Offer Pizza + Salad + Coke combo.
2. **Upsell Prompts at POS**
   - If a customer orders Burger & Fries, prompt "Add a Coke for $1.50" (likely to convert 80% of the time).
   - For Pizza & Salad orders, suggest "Add a Coke?"
3. **Menu Highlighting & Signage**
   - Feature these top combos on the menu board under "Customer Favorites."
   - Use table tents or digital screens to showcase "Did you know 1 in 4 orders of Burger & Fries comes with Coke?"
4. **Dynamic Pricing & Happy Hour**

- Run “Happy Hour” deals on Coke when ordered with Burger & Fries or Pizza & Salad.
- Offer a discounted water upgrade with Pasta & Salad orders.

5. **Loyalty Program Incentives**
    - Award bonus points when customers purchase these combos, encouraging repeat buys.

By focusing on these high-confidence, high-support pairings, the restaurant can create targeted combos and upsell strategies that align with actual ordering behavior—driving higher average checks and improving customer satisfaction.