Assessment Report

On

"Market Basket Analysis"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

Name of discipline

By

YUVRAJ PATEL (20240110300290, CSE-AI D)

Under the supervision of

"MR.ABHISHEK SHUKLA"

KIET Group of Institutions, Ghaziabad

May, 2025

INTRODUCTION

Understanding customer purchasing behavior is a critical aspect of modern retail and e-commerce strategies. One of the most effective techniques for analyzing such behavior is **Market Basket Analysis (MBA)**. This method involves examining large sets of transaction data to discover associations and patterns among items that customers frequently purchase together.

In this report, we utilize **association rule mining**, specifically the **Apriori algorithm**, to classify purchasing patterns and uncover meaningful relationships between products. These relationships, or "association rules," help businesses identify product combinations that tend to occur in the same transactions.

The insights gained through this analysis are valuable for developing **targeted marketing strategies**, such as personalized recommendations, product bundling, and store layout optimization. By identifying which items are commonly bought together, businesses can enhance customer satisfaction, improve sales, and increase overall operational efficiency.

This report details the process of preparing and analyzing transactional data using Python, along with visualizations and interpretations of the resulting association rules.

(1)

Methodology

The methodology for this Market Basket Analysis involves several key stages, from data collection and preprocessing to applying the Apriori algorithm and interpreting the resulting association rules. The steps are outlined below:

1. Data Collection and Loading

The dataset used in this analysis contains transactional records, with each transaction representing a list of items purchased together by a customer. The dataset was uploaded into a Python environment using Pandas for data manipulation and analysis.

2. Data Preprocessing

To prepare the data for analysis, the following preprocessing steps were performed:

- Cleaning: Null values and irrelevant entries were removed to ensure data consistency.
- Transaction Formatting: Each transaction was converted into a list of items.
- One-Hot Encoding: Using the `TransactionEncoder` from the mlxtend library, the data was transformed into a binary matrix format, where each row represents a transaction and each column represents an item. A value of 1 indicates the presence of an item in a transaction, while 0 indicates

absence.

3. Frequent Itemset Generation

The Apriori algorithm was applied to identify frequent itemsets—combinations of items that appear together in transactions above a specified minimum support threshold. The `apriori()` function from the mlxtend.frequent_patterns module was used for this step.

4. Association Rule Mining

From the frequent itemsets, association rules were generated using the `association_rules()` function. Key metrics used to evaluate the rules included:

- Support: Frequency of the itemset in the dataset.
- Confidence: Likelihood that item Y is purchased when item X is purchased.
- Lift: Measure of how much more likely item Y is to be purchased when item X is purchased, compared to when item X is not purchased.

5. Analysis and Visualization

The resulting association rules were analyzed and visualized using Matplotlib and Seaborn to understand their significance and practical implications. Scatter plots and sorted tables were used to highlight the strongest and most relevant rules for targeted marketing strategies.

CODE

```python
# Install required packages
!pip install mlxtend plotly --upgrade

import pandas as pd
import numpy as np
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.graph_objects as go
import plotly.express as px  # Import plotly.express for scatter plot
from itertools import combinations

# ======================
# 1. DATA PREPARATION
# ======================

# Load your aisle data
aisles = pd.read_csv('10. Market Basket Analysis.csv')

# Generate realistic sample transaction data based on common shopping
patterns
np.random.seed(42)
num_transactions = 500

# Create common product groupings that make sense together
common_combinations = [
[24, 83, 84],        # fruits + vegetables + milk
[26, 121],           # coffee + cereal
[112, 36],           # bread + butter
[120, 57],           # yogurt + granola
[45, 46],            # candy + gum
[84, 86],            # milk + eggs
[24, 83, 112],       # fruits + vegetables + bread
[26, 94],            # coffee + tea
[37, 103],           # ice cream + toppings
[72, 89]             # condiments + salad dressing
]
```

```python
transactions = []
for i in range(1, num_transactions + 1):
    # Start with a random common combination
    combo = common_combinations[np.random.randint(0,
    len(common_combinations))]

    # Add 1-3 random additional items
    extra_items = np.random.choice(aisles['aisle_id'],
                                   size=np.random.randint(1, 4),
                                   replace=False)
    basket = list(combo) + list(extra_items)

    for item in basket:
        transactions.append({'transaction_id': i, 'aisle_id': item})

transactions_df = pd.DataFrame(transactions)
transactions_df = transactions_df.merge(aisles, on='aisle_id')

# =======================
# 2. MARKET BASKET ANALYSIS
# =======================

# Prepare basket data
basket = (transactions_df.groupby(['transaction_id', 'aisle'])['aisle']
          .count().unstack().fillna(0)
          .applymap(lambda x: 1 if x > 0 else 0))

# Generate rules
frequent_itemsets = apriori(basket, min_support=0.05, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="lift",
min_threshold=1)
strong_rules = rules[(rules['lift'] >= 1.5) & (rules['confidence'] >=
0.6)]

# =======================
# 3. IMPROVED VISUALIZATIONS
# =======================

plt.figure(figsize=(15, 10))
```

```
# 1. Top Rules Bar Chart
plt.subplot(2, 2, 1)
top_rules = strong_rules.sort_values('lift', ascending=False).head(5)
top_rules['rule'] = top_rules.apply(
lambda x: f"{list(x['antecedents'])[0]} → {list(x['consequents'])[0]}",
axis=1)
sns.barplot(x='lift', y='rule', data=top_rules, palette='viridis')
plt.title('Top 5 Association Rules by Lift')
plt.xlabel('Lift Score')
plt.ylabel('')

# 2. Support vs Confidence Scatter Plot
plt.subplot(2, 2, 2)
scatter = sns.scatterplot(
x='support', y='confidence',
size='lift', hue='lift',
sizes=(50, 300), palette='coolwarm',
data=strong_rules)
plt.title('Rule Strength Metrics')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

# 3. Antecedent-Consequent Heatmap
plt.subplot(2, 2, 3)
# Prepare data for heatmap
heatmap_data = strong_rules.copy()
heatmap_data['antecedent'] = heatmap_data['antecedents'].apply(lambda x:
list(x)[0])
heatmap_data['consequent'] = heatmap_data['consequents'].apply(lambda x:
list(x)[0])
heatmap_pivot = heatmap_data.pivot_table(index='antecedent',
                                         columns='consequent',
                                         values='lift',
                                         aggfunc='mean')

sns.heatmap(heatmap_pivot, cmap='YlGnBu', annot=True, fmt='.1f')
plt.title('Product Association Strength (Lift)')
plt.xticks(rotation=45)
plt.yticks(rotation=0)
```

```python
# 4. Rule Metrics Parallel Plot
plt.subplot(2, 2, 4)
parallel_data = strong_rules[['antecedents', 'consequents', 'support',
'confidence', 'lift']].head(5)
parallel_data['rule'] = parallel_data.apply(
lambda x: f"{list(x['antecedents'])[0]} → {list(x['consequents'])[0]}",
axis=1)
parallel_data = parallel_data.melt(id_vars='rule',
value_vars=['support', 'confidence',
'lift'],
var_name='metric')

sns.lineplot(x='metric', y='value', hue='rule',
data=parallel_data, marker='o', linewidth=2.5)
plt.title('Comparison of Rule Metrics')
plt.ylabel('Score')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')

plt.tight_layout()
plt.show()

# ========================
# 4. INTERACTIVE PLOTLY VISUALIZATIONS
# ========================

# Interactive Sankey Diagram using plotly.graph_objects
if len(strong_rules) > 0:
# Prepare data
sankey_rules = strong_rules.sort_values('lift',
ascending=False).head(5)
all_labels = []
for _, rule in sankey_rules.iterrows():
all_labels.append(list(rule['antecedents'])[0])
all_labels.append(list(rule['consequents'])[0])
unique_labels = list(dict.fromkeys(all_labels))

source = []
target = []
value = []
for _, rule in sankey_rules.iterrows():
```

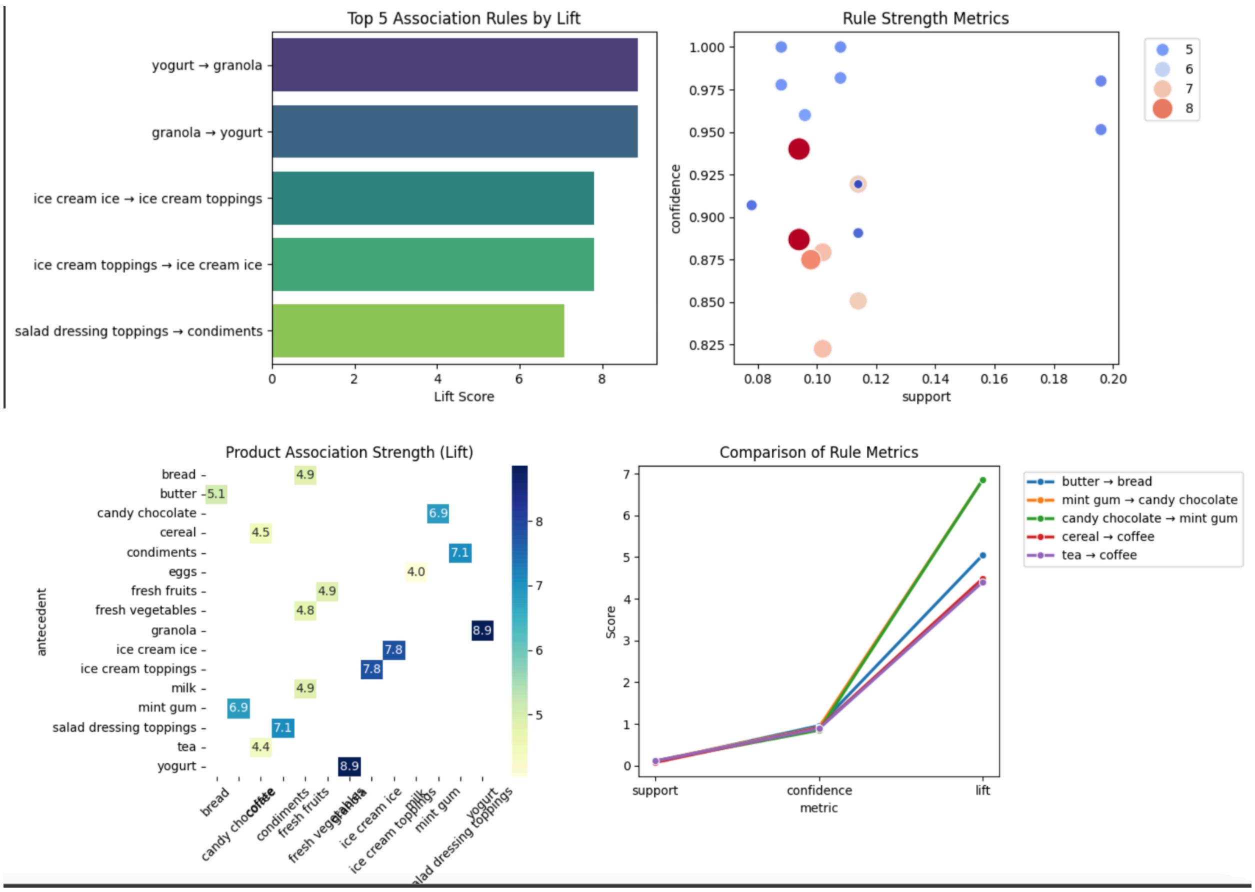
```
source.append(unique_labels.index(list(rule['antecedents'])[0]))
target.append(unique_labels.index(list(rule['consequents'])[0]))
value.append(rule['lift'])

# Sankey Diagram using plotly.graph_objects
fig = go.Figure(go.Sankey(
node=dict(
pad=15,
thickness=20,
line=dict(color="black", width=0.5),
label=unique_labels
),
link=dict(
source=source,
target=target,
value=value
)
))

fig.update_layout(title="Product Association Flow", font_size=10)
fig.show()

# Interactive Bubble Chart using plotly.express
fig = px.scatter(
strong_rules, x='support', y='confidence',
size='lift', color='lift',
hover_name=strong_rules.apply(
lambda x: f"{list(x['antecedents'])[0]} →
{list(x['consequents'])[0]}", axis=1),
log_x=True, size_max=60,
title="Association Rules Explorer"
)
fig.update_layout(showlegend=False)
fig.show()
```

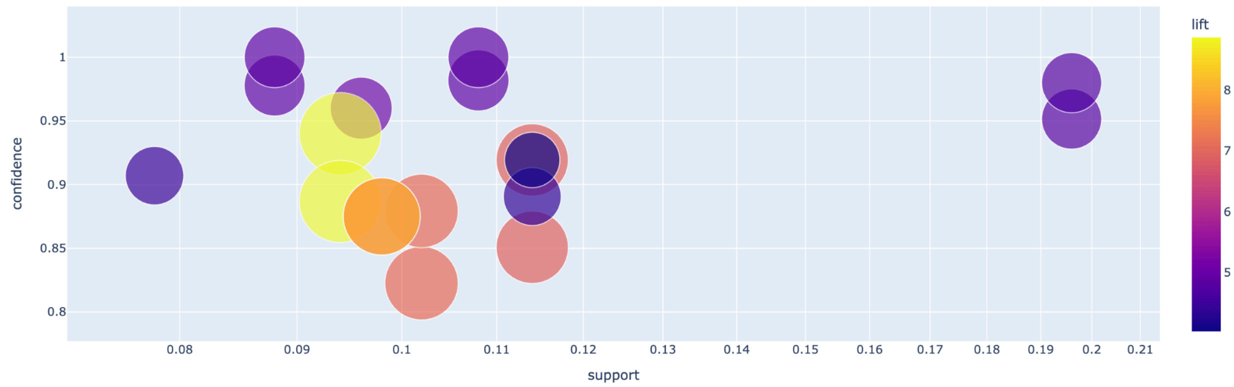
OUTPUT



Top 5 Association Rules by Lift
yogurt → granola
granola → yogurt
ice cream ice → ice cream toppings
ice cream toppings → ice cream ice
salad dressing toppings → condiments
0 2 4 6 8
Lift Score
Rule Strength Metrics
1.000 0.975 0.950 0.925 0.900 0.875 0.850 0.825
confidence
0.08 0.10 0.12 0.14 0.16 0.18 0.20
support
5
6
7
8
Product Association Strength (Lift)
bread
butter
candy chocolate
cereal
condiments
eggs
fresh fruits
fresh vegetables
granola
ice cream ice
ice cream toppings
milk
mint gum
salad dressing toppings
tea
yogurt
antecedent
4.9 5.1 6.9 4.5 7.1 4.0 4.9 4.8 8.9 7.8 7.8 4.9 6.9 7.1 4.4 8.9
8 7 6 5
Comparison of Rule Metrics
butter → bread
mint gum → candy chocolate
candy chocolate → mint gum
cereal → coffee
tea → coffee
0 1 2 3 4 5 6 7
Score
support
confidence
lift
metric

Product Association Flow

Association Rules Explorer



REFERNCE

.365 data science

• Python Libraries: pandas, seaborn, matplotlib, scikit-learn