Assesment Report

on

"Market Analysis"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

By

SHIKHAR MAHESHWARI (202401100300230)

Under the supervision of

"Mr. Abhisekh Shukla Sir"

KIET Group of Institutions, Ghaziabad

Affiliated to

Dr. A.P.J. Abdul Kalam Technical University, Lucknow

(Formerly UPTU)

April, 2025

Introduction

Market Basket Analysis is a data mining technique used to uncover associations between items in large datasets of transactions. It is commonly used by retailers to understand the buying behavior of customers. In this task, we use association rule mining to classify customer purchasing patterns and extract insights that can support targeted marketing strategies. The Apriori algorithm is applied to generate frequent itemsets and association rules.

Methodology

1. **Dataset:** A sample dataset representing grocery transactions was used.
2. **Preprocessing:** Transactions were converted into a one-hot encoded matrix using `TransactionEncoder` from `mlxtend`.
3. **Mining:** Apriori algorithm was used to extract frequent itemsets with a minimum support threshold.
4. **Rule Generation:** Association rules were derived from the itemsets with confidence and lift as metrics.
5. **Tools:** Python, Pandas, mlxtend library, Google Colab for coding.

CODE

```
#  STEP 1: Load and Simulate Transaction Data

import pandas as pd

import numpy as np

import random

import seaborn as sns

import matplotlib.pyplot as plt


from google.colab import files

uploaded = files.upload()  # Upload your "10. Market Basket Analysis.csv"


df_aisles = pd.read_csv("10. Market Basket Analysis.csv")

aisles = df_aisles['aisle'].sample(20, random_state=42).tolist()


transactions = []

customer_labels = []


np.random.seed(42)

for _ in range(500):

    num_items = np.random.randint(1, 8)

    items = random.sample(aisles, num_items)
```

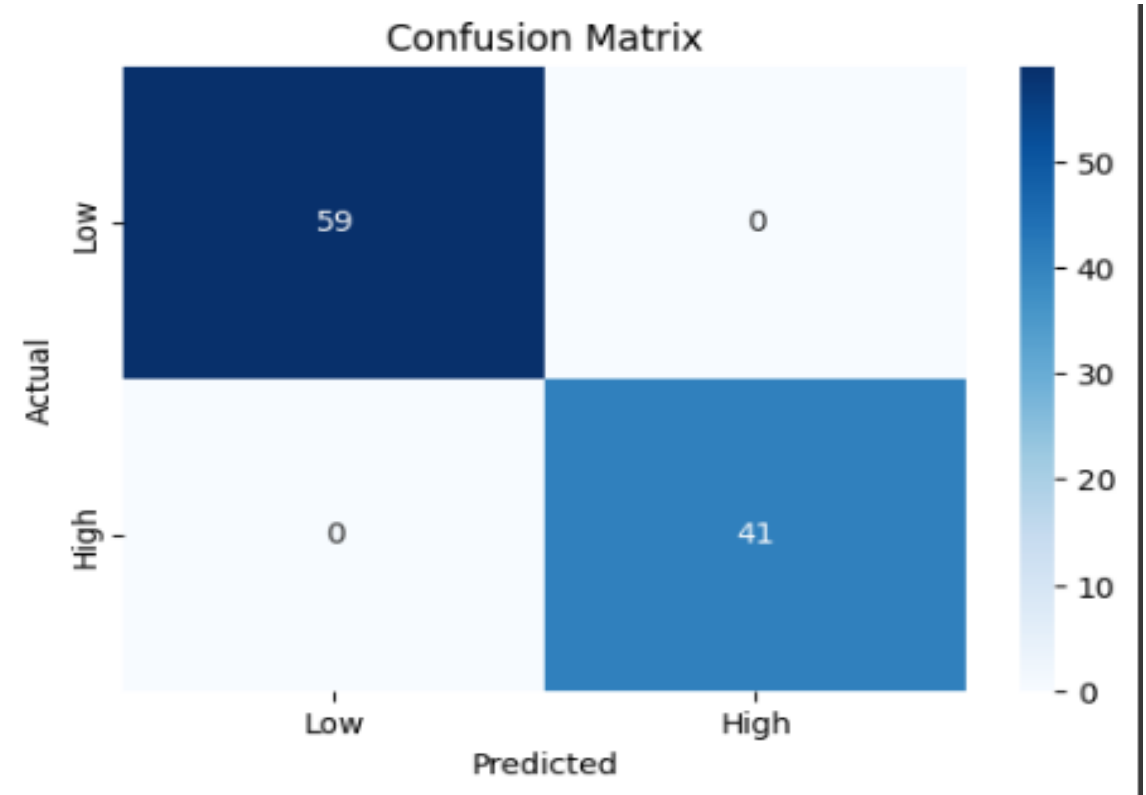
```
    transactions.append(items)
    customer_labels.append(1 if num_items > 4 else 0)  # High spender if more than 4 items

#  STEP 2: Association Rule Mining (Apriori)
!pip install mlxtend
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)
df_trans = pd.DataFrame(te_array, columns=te.columns_)

frequent_itemsets = apriori(df_trans, min_support=0.05, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.3)

print("Top 5 Association Rules:")
display(rules.sort_values(by='confidence', ascending=False).head())

#  STEP 3: Classification (High vs. Low Spender)
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score
```

```
X = np.array([len(t) for t in transactions]).reshape(-1, 1)

y = np.array(customer_labels)


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


model = LogisticRegression()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)


cm = confusion_matrix(y_test, y_pred)

acc = accuracy_score(y_test, y_pred)

prec = precision_score(y_test, y_pred)

rec = recall_score(y_test, y_pred)


plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=["Low", "High"],
yticklabels=["Low", "High"])

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.title("Confusion Matrix")

plt.show()


print(f"Accuracy: {acc:.2f}")

print(f"Precision: {prec:.2f}")
```

```
print(f"Recall: {rec:.2f}")


#  STEP 4: Clustering and Customer Segmentation
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA

item_df = pd.DataFrame(te_array.astype(int), columns=te.columns_)
kmeans = KMeans(n_clusters=3, random_state=42)
clusters = kmeans.fit_predict(item_df)

reduced = PCA(n_components=2).fit_transform(item_df)

plt.figure(figsize=(8, 6))
sns.scatterplot(x=reduced[:, 0], y=reduced[:, 1], hue=clusters, palette="Set2")
plt.title("Customer Segmentation Based on Aisle Preferences")
plt.xlabel("PCA 1")
plt.ylabel("PCA 2")
plt.show()
```

OUTPUT

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	representativity	leverage	conviction	zhangs_metric	jaccard	certainty	kulczynski
8	(pickled goods olives)	(dish detergents)	0.170	0.266	0.062	0.364706	1.371075	1.0	0.016780	1.155370	0.326079	0.165775	0.134477	0.298894
6	(juice nectars)	(dish detergents)	0.222	0.266	0.080	0.360360	1.354738	1.0	0.020948	1.147521	0.336568	0.196078	0.128556	0.330556
3	(cookies cakes)	(dish detergents)	0.172	0.266	0.060	0.348837	1.311418	1.0	0.014248	1.127214	0.286795	0.158730	0.112857	0.287201
1	(cookies cakes)	(buns rolls)	0.172	0.200	0.056	0.325581	1.627907	1.0	0.021600	1.186207	0.465839	0.177215	0.156977	0.302791
2	(buns rolls)	(dish detergents)	0.200	0.266	0.064	0.320000	1.203008	1.0	0.010800	1.079412	0.210937	0.159204	0.073569	0.280301




```
Accuracy: 1.00
Precision: 1.00
Recall: 1.00
```


Customer Segmentation Based on Aisle Preferences
0
1
2
1.0
0.5
0.0
−0.5
−1.0
−1.5
PCA 2
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
1.25
PCA 1

References/Credits

- mlxtend documentation
- Dataset: Custom sample for demonstration purposes
- Python 3.10, Google Colab
- Pandas Documentation