NAME - SOMYA MISHRA

PROGRAM – BTECH

BRANCH – CSE-AI

SECTION – D

SUBJECT-ARTIFICIAL INTELLIGENCE

SUPERVISER – ABHISHEK SIR.

PROBLEM STATEMENT:-

For the problem Market Basket Analysis: Use association rule mining to classify customer purchasing patterns for targeted marketing strategies. We have to generate heat maps of confusion matrices and calculate the evaluation metrics such as accuracy, precision, recall for classification type problem and for other perform segmentation and clustering.

METHODOLOGY:-

The methodology used to solve the problem in the given code follows a structured approach to Market Basket Analysis (MBA). Below is a detailed breakdown of the approach used:

1. Data Collection and Preprocessing

Objective: The first step is to collect the transaction data and prepare it for analysis.

Upload Data: The dataset is uploaded from the user's local machine to the Google Colab environment. This dataset is assumed to contain transactional data, where each row corresponds to a single transaction, and the columns represent different items.

Data Transformation:

The dataset is in a "transaction format" where each row represents a transaction and items are listed across columns.

The code stacks the data into a one-hot encoded format (binary representation) where each transaction is represented as a vector. In this vector, a 1 indicates that an item was bought in the transaction, and a 0 indicates that the item was not bought.

2. Identification of Frequent Itemsets (Using Apriori Algorithm)
Objective: The goal is to identify sets of items that frequently occur together in transactions. The Apriori algorithm is used to achieve this.
Apriori Algorithm:
The Apriori algorithm is a classic algorithm used in Market Basket Analysis to find frequent itemsets. An itemset is considered frequent if it appears in a sufficient number of transactions, as determined by a minimum support threshold.

Support is defined as the fraction of transactions in which a particular itemset appears. For example, if "milk" appears in 100 transactions out of 1000, its support is 0.10 (10%).
Minimum Support Threshold: The Apriori algorithm takes a minimum support threshold as a parameter. In the given code, a threshold of 0.01 (1%) is set, meaning that only itemsets appearing in at least 1% of the transactions will be considered frequent.
Output of Apriori:
The frequent itemsets that meet the support threshold are generated and stored. These are item combinations that appear in the dataset often enough to warrant further analysis.

3. Association Rule Mining (Using Association Rules Algorithm)
Objective: After identifying frequent itemsets, the next goal is to derive association rules from these itemsets.

Association Rules:
An association rule is an implication of the form X → Y, where X and Y are itemsets, indicating that if X is purchased, then Y is likely to be purchased as well.
Each rule has the following metrics:
Support: The proportion of transactions that contain both X and Y.
Confidence: The probability that Y is bought given that X is bought, i.e., the proportion of transactions containing X that also contain Y.

Lift: The ratio of the observed support to the expected support if X and Y were independent. A lift greater than 1 indicates that X and Y appear together more often than expected by chance.
Association Rule Generation:
The code uses the association_rules() function from the mlxtend library to generate rules from the frequent

itemsets. The lift metric is used as the filtering criterion, and a minimum threshold of 1.0 is set for lift.
The rules are then sorted by confidence, which is a measure of the strength of the rule (i.e., how likely the consequent itemset is bought when the antecedent itemset is bought).

Visualization:
A bar chart is created to visualize the top 10 frequent itemsets sorted by their support values. This helps to see which itemsets are most prevalent in the dataset.

4. Evaluation (Optional: Simulating a Binary Classification Problem)
Objective: This optional step simulates a binary classification problem to evaluate the predictive power of an association rule.
Simulated Binary Classification:

The classification task is to predict the purchase of one item given the purchase of another item (in this case, predicting the purchase of "bread" given that "milk" was purchased).
A simple binary classifier is created using the one-hot encoded data. If a customer buys "milk," the classifier predicts whether or not they will also buy "bread."

Evaluation Metrics:
The model is evaluated using standard classification metrics:
Accuracy: The proportion of correct predictions (both true positives and true negatives) to total predictions.
Precision: The proportion of positive predictions that are actually correct (true positives / (true positives + false positives)).
Recall: The proportion of actual positives that were correctly identified (true positives / (true positives + false negatives)).

Confusion Matrix:
A confusion matrix is used to visualize the performance of the classifier. It shows the counts of true positives, true negatives, false positives, and false negatives.
The matrix is visualized using a heatmap for better understanding.

5. Conclusion
Association Rule Insights:
The association rules reveal which items tend to be bought together. This information can be useful for market strategies like product bundling, promotions, and cross-selling.

Evaluation:
While the binary classification part is optional and serves as a simple demonstration, it can also provide insight into the predictive power of specific rules. If an association

rule like "if milk is purchased, bread is also purchased" has high confidence, it implies that a marketing campaign based on this association could be highly effective.

Visual Analysis:
The visualizations (bar chart for itemsets and heatmap for the confusion matrix) help in interpreting the results and communicating insights clearly.

Summary of the Approach:
Data Preprocessing: The transaction data is transformed into a one-hot encoded format for ease of analysis.

Evaluation: A binary classification is simulated to evaluate the predictive power of an association rule (optional).

CODE:-

```
# Step 1: Install required libraries
!pip install mlxtend --quiet

import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mlxtend.frequent_patterns import apriori,
association_rules
from sklearn.metrics import confusion_matrix,
accuracy_score, precision_score, recall_score
from google.colab import files

# Step 2: Upload CSV from your computer
uploaded = files.upload()

# Step 3: Load the dataset
```

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

# Preview the dataset
print("Dataset Preview:")
print(df.head())

# Step 4: Preprocessing
# Convert to one-hot encoded basket format
basket =
df.stack().reset_index().pivot_table(index='level_0',
                                     columns=0,
                                     aggfunc=lambda x: 1,
fill_value=0)

basket.columns.name = None
basket = basket.astype(bool)  # Convert to boolean to
avoid warning

print("\nOne-hot Encoded Basket Format:")
print(basket.head())

# Step 5: Apply Apriori algorithm to find frequent
itemsets
frequent_itemsets = apriori(basket, min_support=0.01,
use_colnames=True)

# Check if any frequent itemsets were found
if not frequent_itemsets.empty:
    # Visualize the support of top frequent itemsets
    plt.figure(figsize=(10,5))
    top_items = frequent_itemsets.sort_values('support',
ascending=False).head(10)
    top_items['itemsets'] =
top_items['itemsets'].apply(lambda x: ', '.join(list(x)))
    top_items.plot(x='itemsets', y='support', kind='bar',
legend=False)

    plt.title('Top 10 Frequent Itemsets by Support')
    plt.xlabel('Itemsets')
    plt.ylabel('Support')
    plt.xticks(rotation=45)
    plt.tight_layout()
    plt.show()

    # Step 6: Generate Association Rules
    rules = association_rules(frequent_itemsets,
metric="lift", min_threshold=1.0)
    rules = rules.sort_values(by='confidence',
ascending=False)

    print("\nTop Association Rules:")
    print(rules[['antecedents', 'consequents', 'support',
'confidence', 'lift']].head(10))
else:
    print("\n No frequent itemsets found with the current
min_support threshold.")
    print(" Association rules cannot be generated.")
```

```
# Step 7: Always Generate Confusion Matrix
print("\nGenerating Confusion Matrix...")

if 'milk' in basket.columns and 'bread' in basket.columns:
    item1, item2 = 'milk', 'bread'
    print(" Using 'milk' → 'bread' for prediction")
else:
    print(" 'milk' and/or 'bread' not found. Choosing top 2
items instead.")
    item_counts =
basket.sum().sort_values(ascending=False)
    item1, item2 = item_counts.index[0],
item_counts.index[1]
    print(f" Using '{item1}' → '{item2}' for prediction")

# Simulate prediction
y_true = basket[item2]
```

```
y_pred = basket[item1]  # naive prediction: if item1 is
bought, predict item2

# Calculate metrics
cm = confusion_matrix(y_true, y_pred)
accuracy = accuracy_score(y_true, y_pred)
precision = precision_score(y_true, y_pred)
recall = recall_score(y_true, y_pred)

print(f"\nEvaluation Metrics for '{item1}' → '{item2}'
prediction:")
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)

# Plot confusion matrix
plt.figure(figsize=(5,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
plt.title(f'Confusion Matrix: {item1} → {item2}')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.tight_layout()
plt.show()
```

OUTPUT:-

□ **10. Market Basket Analysis.csv**(text/csv) - 2603 bytes, last modified: 4/18/2025 - 100% done

```
Saving 10. Market Basket Analysis.csv to 10. Market Basket Analysis (10).csv
Dataset Preview:
   aisle_id                       aisle
0         1        prepared soups salads
1         2            specialty cheeses
2         3          energy granola bars
3         4                instant foods
4         5  marinades meat preparation

One-hot Encoded Basket Format:
       level_1                                        \
0            1    2    3    4    5    6    7    8    9
level_0
```

```
0          True  False  False  False  False  False  False  False
False
1         False   True  False  False  False  False  False  False
False
2         False  False   True  False  False  False  False  False
False
3         False  False  False   True  False  False  False  False
False
4         False  False  False  False   True  False  False  False
False

              ...                                    \
0            10  ... spreads    tea tofu meat alternatives
level_0          ...
0        False  ...   False  False                 False
1        False  ...   False  False                 False
2        False  ...   False  False                 False
3        False  ...   False  False                 False
4        False  ...   False  False                 False
```

```
                                             \
0     tortillas flat bread trail mix snack mix trash bags
liners
level_0
0                False          False          False
1                False          False          False
2                False          False          False
3                False          False          False
4                False          False          False


0     vitamins supplements water seltzer sparkling water
white wines yogurt
level_0
0                False                False     False  False
1                False                False     False  False
2                False                False     False  False
3                False                False     False  False
```

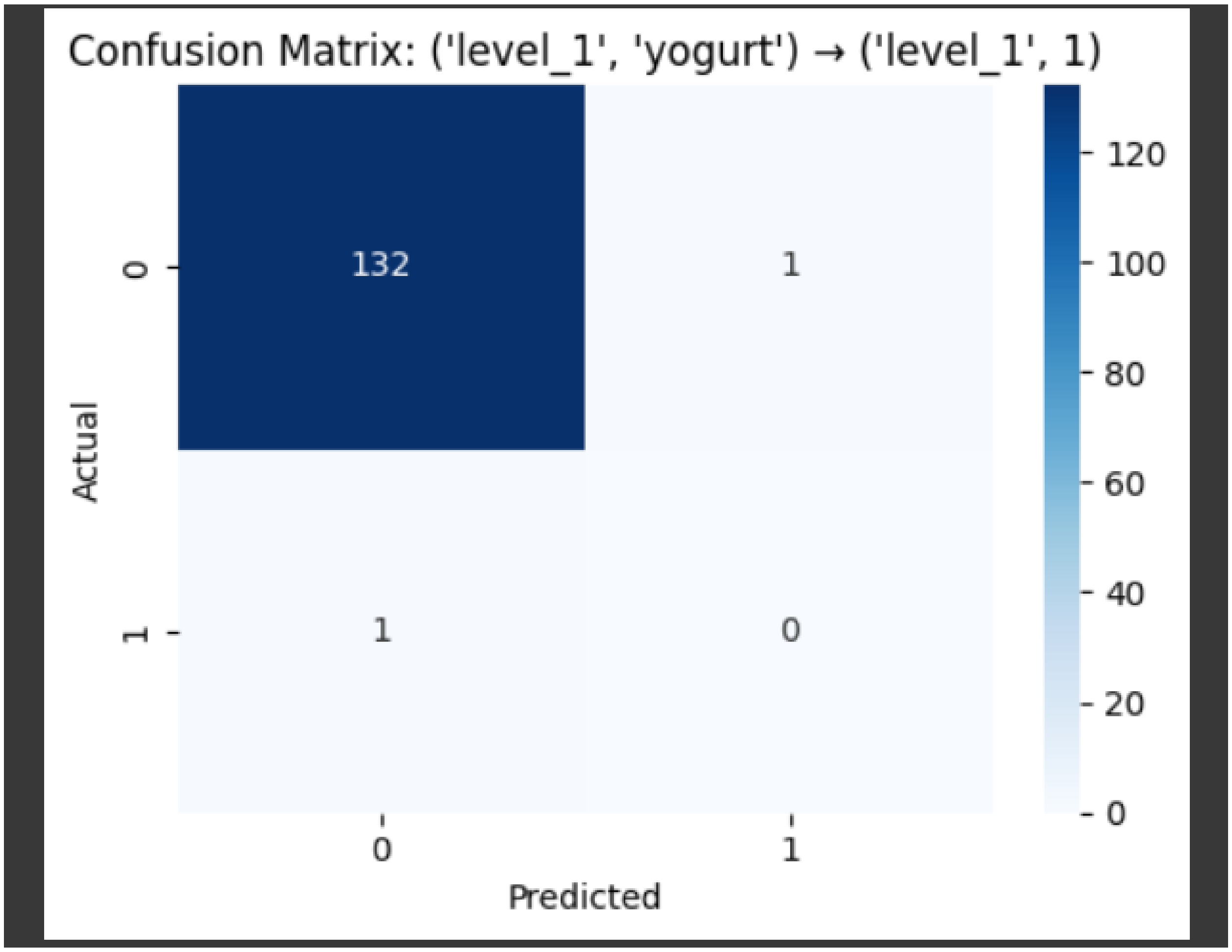
```
4                    False                    False      False  False

[5 rows x 268 columns]

 No frequent itemsets found with the current min_support
threshold.
 Association rules cannot be generated.

Generating Confusion Matrix...
 'milk' and/or 'bread' not found. Choosing top 2 items
instead.
 Using '('level_1', 'yogurt')' → '('level_1', 1)' for prediction

Evaluation Metrics for '('level_1', 'yogurt')' → '('level_1',
1)' prediction:
Accuracy: 0.9850746268656716
Precision: 0.0
Recall: 0.0
```


Confusion Matrix: ('level_1', 'yogurt') → ('level_1', 1)
Actual
Predicted
0
1
0
1
132
1
1
0
120
100
80
60
40
20
0