Market Basket Analysis Project

PHASE-4

Submitted by

ATHIRALA GANESH

au723921243003

athiralaganesh004@gmail.com

Overview

This notebook is part of a project focused on market basket analysis. We will begin by loading and preprocessing the dataset.

Loading the Dataset

Let's start by loading the dataset into a DataFrame using pandas.

add Codeadd Markdown[ ]:

```
import pandas as pd

# Load the dataset
dataset_path = '/kaggle/input/market-basket-analysis/Assignment-1_Data.xlsx'
df = pd.read_excel(dataset_path)
```

add Codeadd Markdown

Phase 4 starts from here

Formatting the transaction data in a suitable format for analysis

Developing the preprocessed data into analysis. Split the 'Itemname' column in transaction_data into individual items using str.split(', ', expand=True).Concatenate the original DataFrame (transaction_data) with the items DataFrame (items_df) using pd.concat.Drop the original 'Itemname' column since individual items are now in separate columns.Display the resulting DataFrame.

add Codeadd Markdown[ ]:

```
# Split the 'Itemname' column into individual items
items_df = transaction_data['Itemname'].str.split(', ', expand=True)

# Concatenate the original DataFrame with the new items DataFrame
transaction_data = pd.concat([transaction_data, items_df], axis=1)

# Drop the original 'Itemname' column
transaction_data = transaction_data.drop('Itemname', axis=1)

# Display the resulting DataFrame
print(transaction_data.head())
```

add Codeadd Markdown

Association Rules - Data Mining

Converting Items to Boolean Columns

To prepare the data for association rule mining, we convert the items in the transaction_data DataFrame into boolean columns using one-hot encoding. This is achieved through the pd.get_dummies function, which creates a new DataFrame (df_encoded) with boolean columns representing the presence or absence of each item.

add Codeadd Markdown[ ]:

```
# Convert items to boolean columns
df_encoded = pd.get_dummies(transaction_data, prefix='', prefix_sep='').groupby(level=0, axis=1).max()

# Save the transaction data to a CSV file
df_encoded.to_csv('transaction_data_encoded.csv', index=False)
```

add Codeadd Markdown

Association Rule Mining

We apply the Apriori algorithm to perform association rule mining on the encoded transaction data. The min_support parameter is set to 0.007 to filter out infrequent itemsets. The resulting frequent itemsets are then used to generate association rules based on a minimum confidence threshold of 0.5.Finally, we print the generated association rules.

add Codeadd Markdown[ ]:

```
# Load transaction data into a DataFrame
df_encoded = pd.read_csv('transaction_data_encoded.csv')

from mlxtend.frequent_patterns import apriori, association_rules

# Association Rule Mining
frequent_itemsets = apriori(df_encoded, min_support=0.007, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
```

```
# Display information of the rules
print("Association Rules:")
print(rules.head())
```
add Codeadd Markdown

Visualization

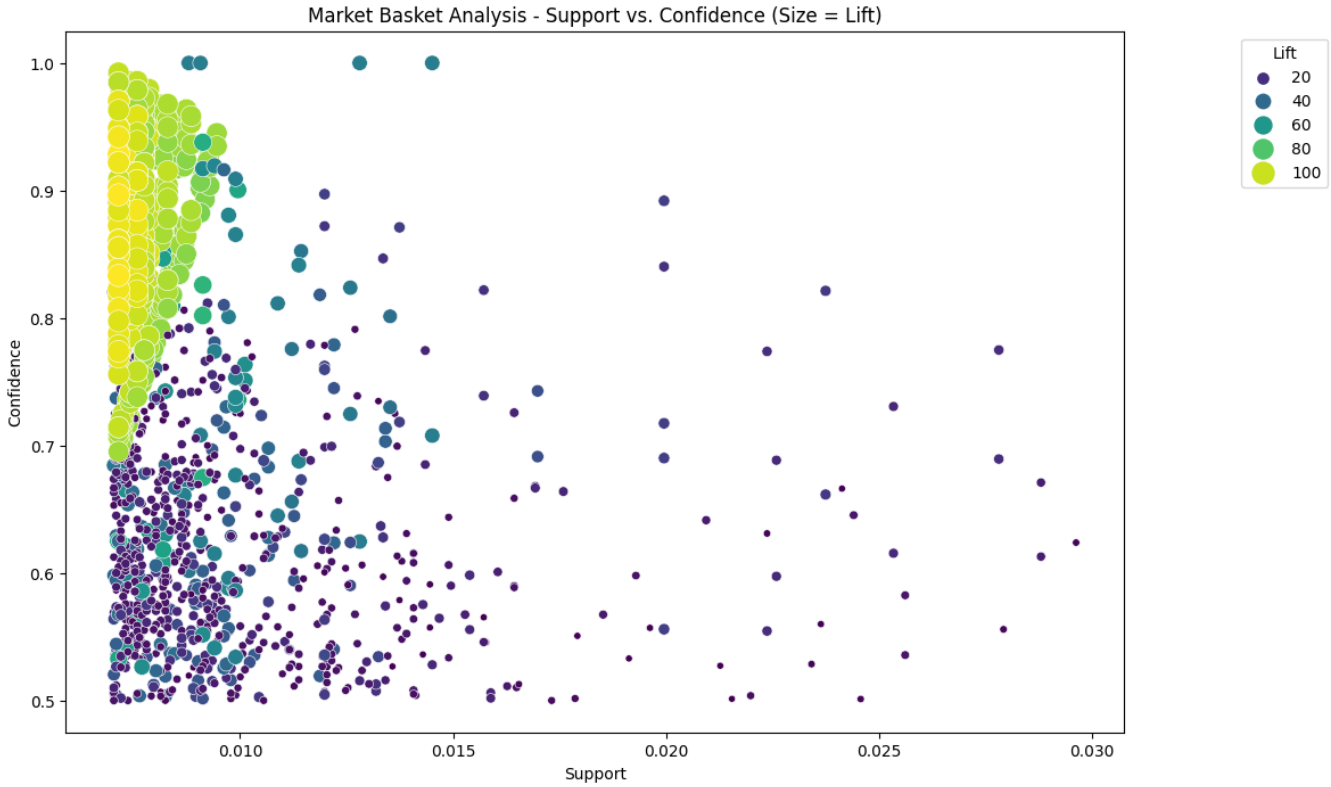
Visualizing Market Basket Analysis Results

We use matplotlib and seaborn libraries to create a scatterplot visualizing the results of the market basket analysis. The plot depicts the relationship between support, confidence, and lift for the generated association rules.

add Codeadd Markdown[ ]:

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

# Plot scatterplot for Support vs. Confidence
plt.figure(figsize=(12, 8))
sns.scatterplot(x="support", y="confidence", size="lift", data=rules, hue="lift",
palette="viridis", sizes=(20, 200))
plt.title('Market Basket Analysis - Support vs. Confidence (Size = Lift)')
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.legend(title='Lift', loc='upper right', bbox_to_anchor=(1.2, 1))
plt.show()
```

add Codeadd Markdown



Interactive Market Basket Analysis Visualization

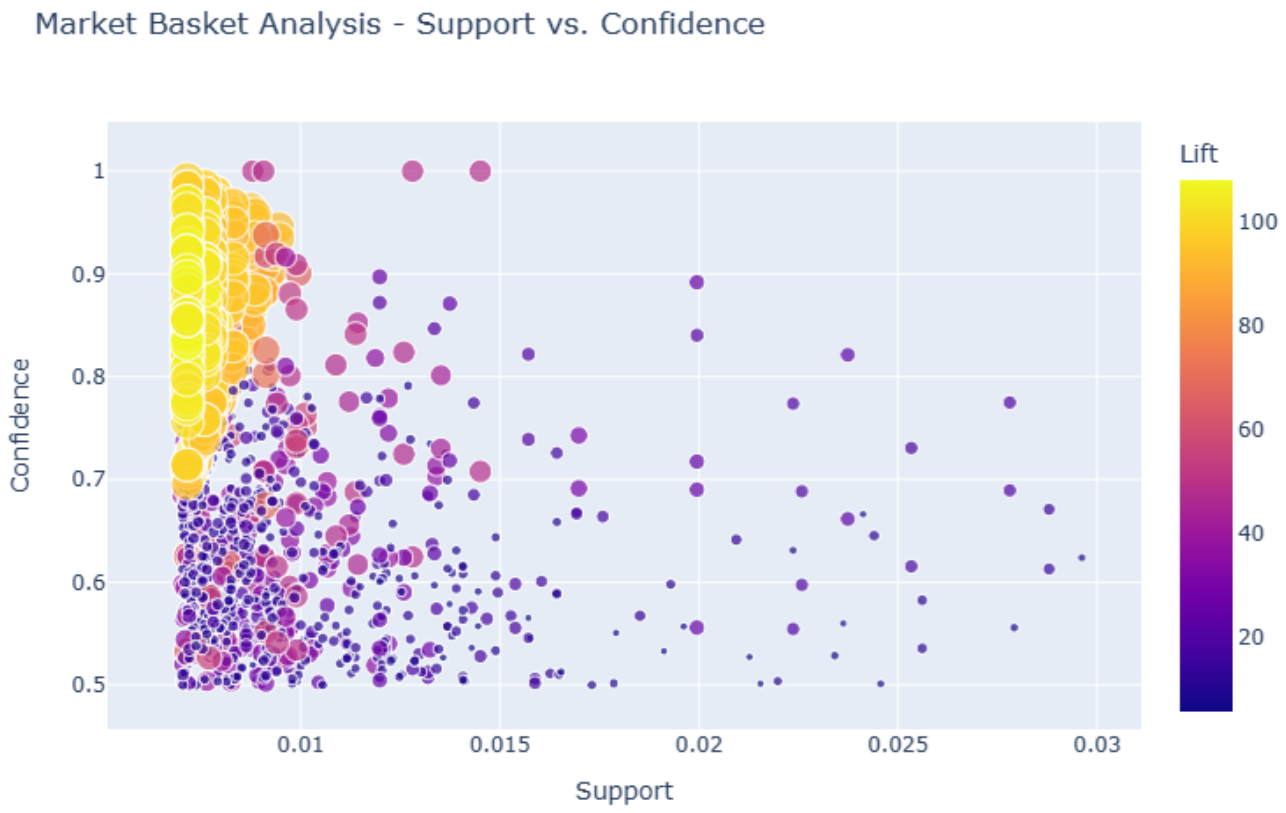
We leverage the Plotly Express library to create an interactive scatter plot visualizing the results of the market basket analysis. This plot provides an interactive exploration of the relationship between support, confidence, and lift for the generated association rules.

add Codeadd Markdown[ ]:

```
import plotly.express as px

# Convert frozensets to lists for serialization
rules['antecedents'] = rules['antecedents'].apply(list)
rules['consequents'] = rules['consequents'].apply(list)

# Create an interactive scatter plot using plotly express
fig = px.scatter(rules, x="support", y="confidence", size="lift",
                 color="lift", hover_name="consequents",
                 title='Market Basket Analysis - Support vs. Confidence',
                 labels={'support': 'Support', 'confidence': 'Confidence'})

# Customize the layout
```

```
fig.update_layout(
   xaxis_title='Support',
   yaxis_title='Confidence',
   coloraxis_colorbar_title='Lift',
   showlegend=True
)

# Show the interactive plot
fig.show()
```

add Codeadd Markdown



Interactive Network Visualization for Association Rules

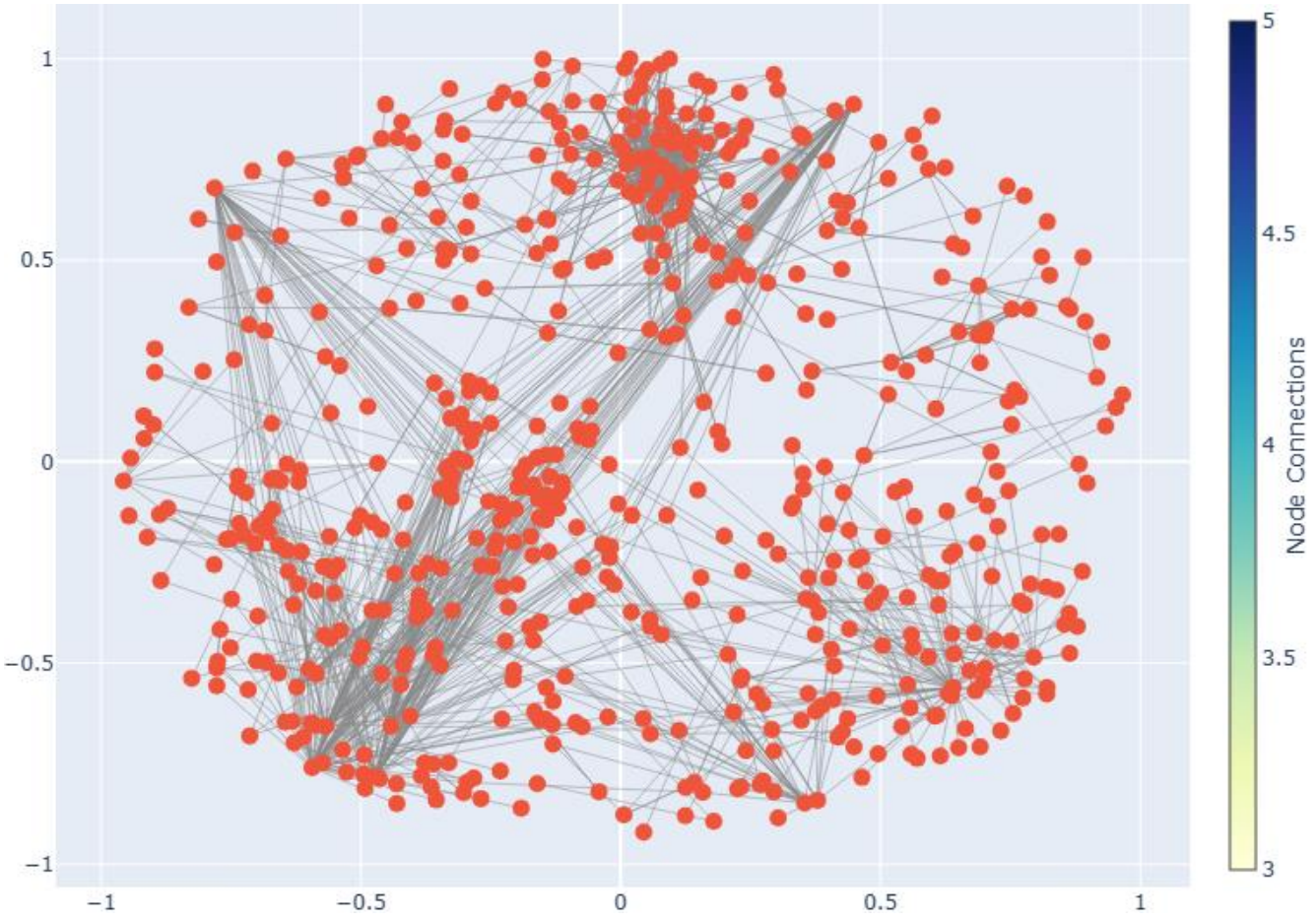
We utilize the NetworkX and Plotly libraries to create an interactive network graph visualizing the association rules. This graph represents relationships between antecedent and consequent items, showcasing support as edge weights.

add Codeadd Markdown[ ]:

```
import networkx as nx
import matplotlib.pyplot as plt
import plotly.graph_objects as go

# Create a directed graph
G = nx.DiGraph()

# Add nodes and edges from association rules
for idx, row in rules.iterrows():
    G.add_node(tuple(row['antecedents']), color='skyblue')
    G.add_node(tuple(row['consequents']), color='orange')
    G.add_edge(tuple(row['antecedents']), tuple(row['consequents']),
weight=row['support'])

# Set node positions using a spring layout
pos = nx.spring_layout(G)

# Create an interactive plot using plotly
edge_x = []
edge_y = []
for edge in G.edges(data=True):
    x0, y0 = pos[edge[0]]
    x1, y1 = pos[edge[1]]
    edge_x.append(x0)
    edge_x.append(x1)
    edge_x.append(None)
    edge_y.append(y0)
    edge_y.append(y1)
    edge_y.append(None)

edge_trace = go.Scatter(
    x=edge_x, y=edge_y,
    line=dict(width=0.5, color='#888'),
    hoverinfo='none',
    mode='lines')
```

```
node_x = []
node_y = []
for node in G.nodes():
    x, y = pos[node]
    node_x.append(x)
    node_y.append(y)

node_trace = go.Scatter(
    x=node_x, y=node_y,
    mode='markers',
    hoverinfo='text',
    marker=dict(
        showscale=True,
        colorscale='YlGnBu',
        size=10,
        colorbar=dict(
            thickness=15,
            title='Node Connections',
            xanchor='left',
            titleside='right'
        )
    )
)

# Customize the layout
layout = go.Layout(
    showlegend=False,
    hovermode='closest',
    margin=dict(b=0, l=0, r=0, t=0),
)

# Create the figure
fig = go.Figure(data=[edge_trace, node_trace], layout=layout)

# Show the interactive graph
fig.show()
```

add Codeadd Markdown


1
0.5
0
−0.5
−1
−1
−0.5
0
0.5
1
5
4.5
4
3.5
3
Node Connections

Interactive Sunburst Chart for Association Rules

We use Plotly Express to create an interactive sunburst chart visualizing association rules. This chart represents the relationships between antecedent and consequent items, showcasing lift as well as support through color intensity.

add Codeadd Markdown[ ]:

```
import plotly.express as px
```