AI_PHASE5

Market Basket Analysis

Final Submission

TEAM MEMBERS

A.ANAIKUTTY

M.ALEX PANDIAN

S.SURIYA

M.ARAVINTH

M.VIJAYARAJAN

IBM Naan Mudhalvan Project – Artificial Intelligence – Market Basket Analysis

Problem Statement

"The problem is to perform market basket analysis on a provided dataset to unveil hidden paterns and associations between products. The goal is to understand customer purchasing behavior and identify potential cross-selling opportunities for a retail business. This project involves using association analysis techniques, such as Apriori algorithm, to find frequently co-occurring products and generate insights for business optimization."

Problem Analysis and Inference [Phase 1]

From analyzing the given problem, the task is to perform market basket analysis with the given dataset to find hidden paterns and relations between products if they exist. This is to understand the purchasing behavior of customers and check for potential cross-selling opportunities for retailers. This problem requires the usage of association analysis techniques, to generate insights for business improvements.

Dataset Details and Description

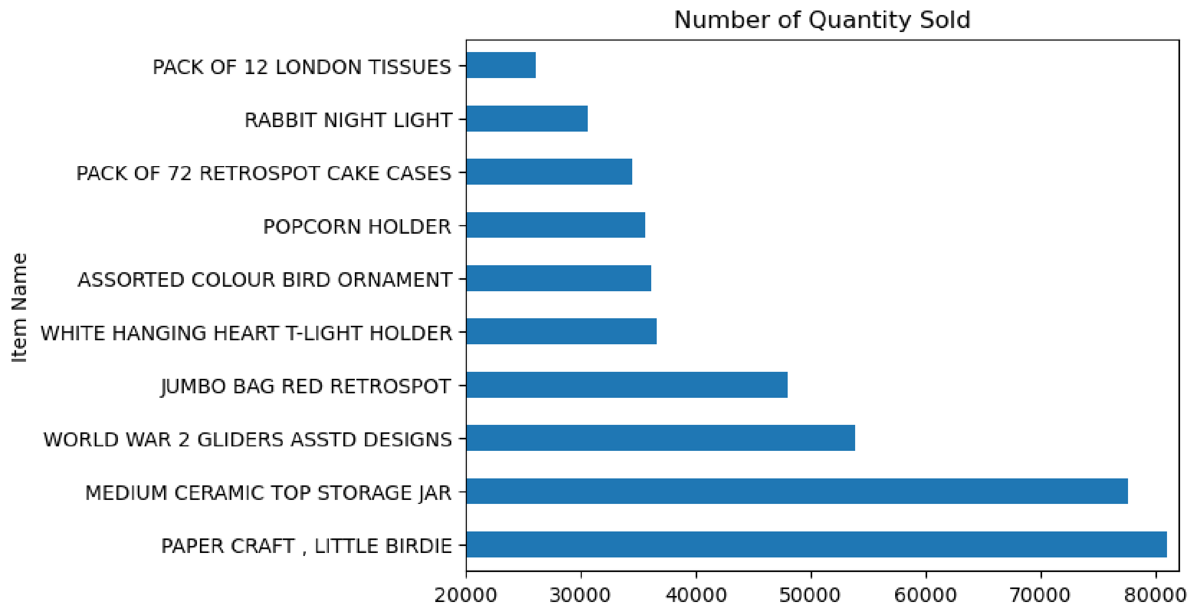
- **Link to the dataset:** *https://www.kaggle.com/datasets/aslanahmedov/market-basket-analysis*
- The dataset contains transactional data from a retail store over a specified time period.
- Each row represents a unique transaction, listing the items purchased by a customer.
- The dataset includes information such as transaction ID, customer ID, and a list of purchased products.
- Product details include product names or IDs, categories, and prices.
- It also contains additional metadata such as country of purchase, date, time and other information.

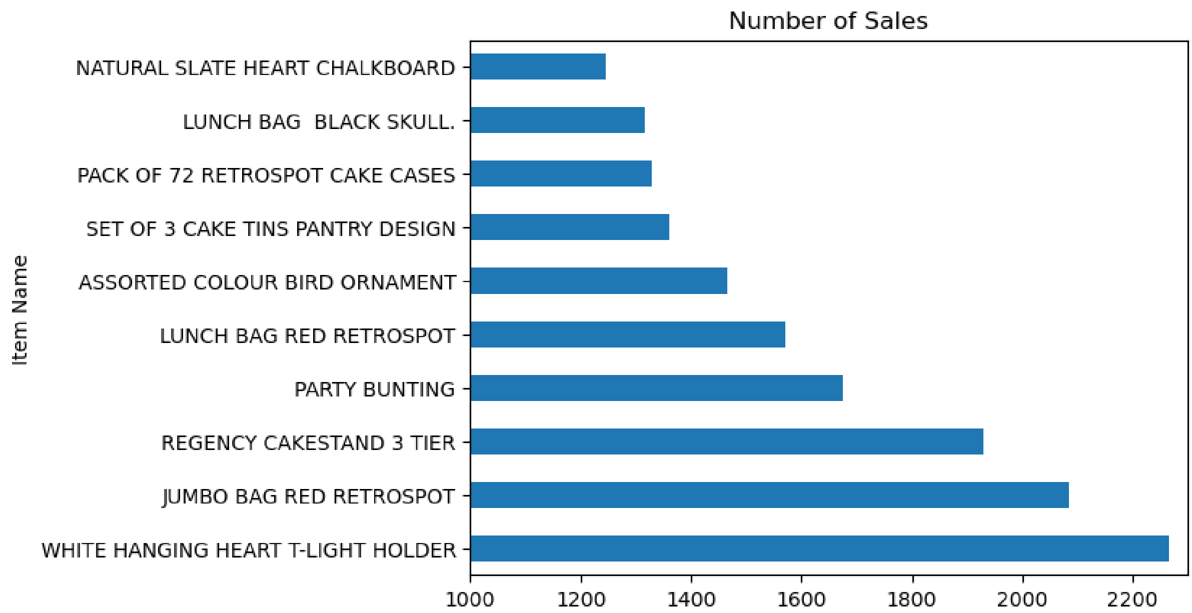
Design Thinking

- Discover frequent itemsets: Apply the Apriori algorithm to identify which products are frequently purchased together in customer transactions.
- Calculate association rules: Establish association rules, including support, confidence, and lift, to quantify the relationships between products.
- Uncover cross-selling opportunities: Identify product pairs or sets that exhibit strong associations, enabling the retail business to strategically promote and bundle related products.
- Understand customer purchasing behavior: Gain insights into customer preferences and behaviors based on the discovered paterns.
- Optimize business strategies: Utilize the analysis findings to enhance product placement, marketing campaigns, and overall business operations.

- ## Phases of Development

1. **Phase 1 (Problem Definition and Design Thinking)** – The given problem is read and understood to find possible solutions. An outline of the solution is thought out using Design Thinking. The dataset is downloaded and prepared for analysis.
2. **Phase 2 (Innovation)** – The dataset is preprocessed and imported in the program. Then the sales data is taken as variables and the products sold the most and highest quantity products are taken and a graph is ploted to visualize the data and to be able to interpret it easily





3. **Phase 3 (AI Development Part 1)** – The apriori algorithm is imported and the association rules are created with a minimum threshold of 0.5% using the frequent itemsets sold.

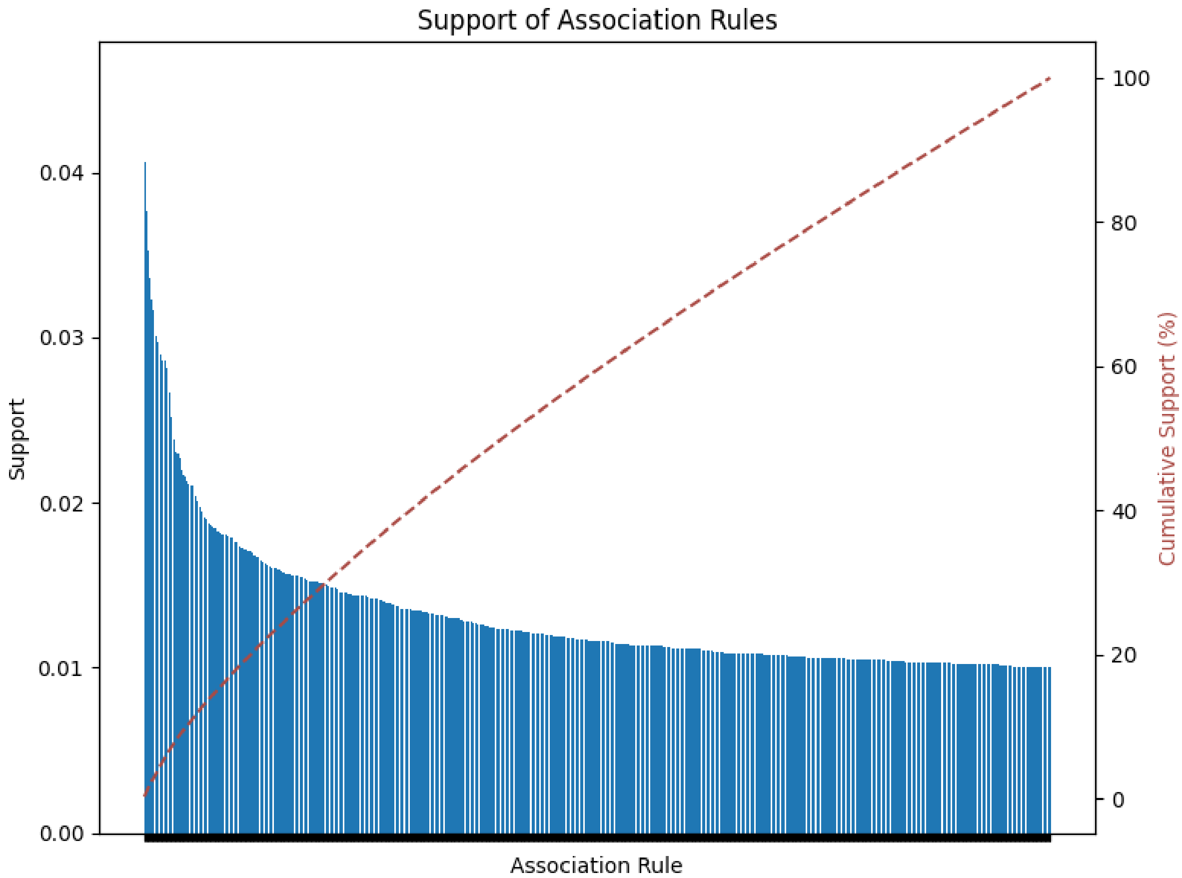
```
Frequent Itemsets:
       support                                          itemsets
0     0.017370                         (10 COLOUR SPACEBOY PEN)
1     0.013751                   (12 MESSAGE CARDS WITH ENVELOPES)
2     0.019653                     (12 PENCIL SMALL TUBE WOODLAND)
3     0.019820               (12  PENCILS SMALL TUBE RED RETROSPOT)
4     0.019597                      (12 PENCILS SMALL TUBE SKULL)
...        ...                                               ...
2467  0.010355  (LUNCH BAG RED  RETROSPOT, LUNCH BAG SUKI DESIG...
2468  0.010188  (LUNCH BAG RED  RETROSPOT, LUNCH BAG SUKI DESIG...
2469  0.010300  (LUNCH BAG RED  RETROSPOT, LUNCH BAG SPACEBOY D...
2470  0.010467  (LUNCH BAG RED RETROSPOT, LUNCH BAG PINK POLKA...
2471  0.011302  (CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...

[2472 rows x 2 columns]

Association Rules:
```

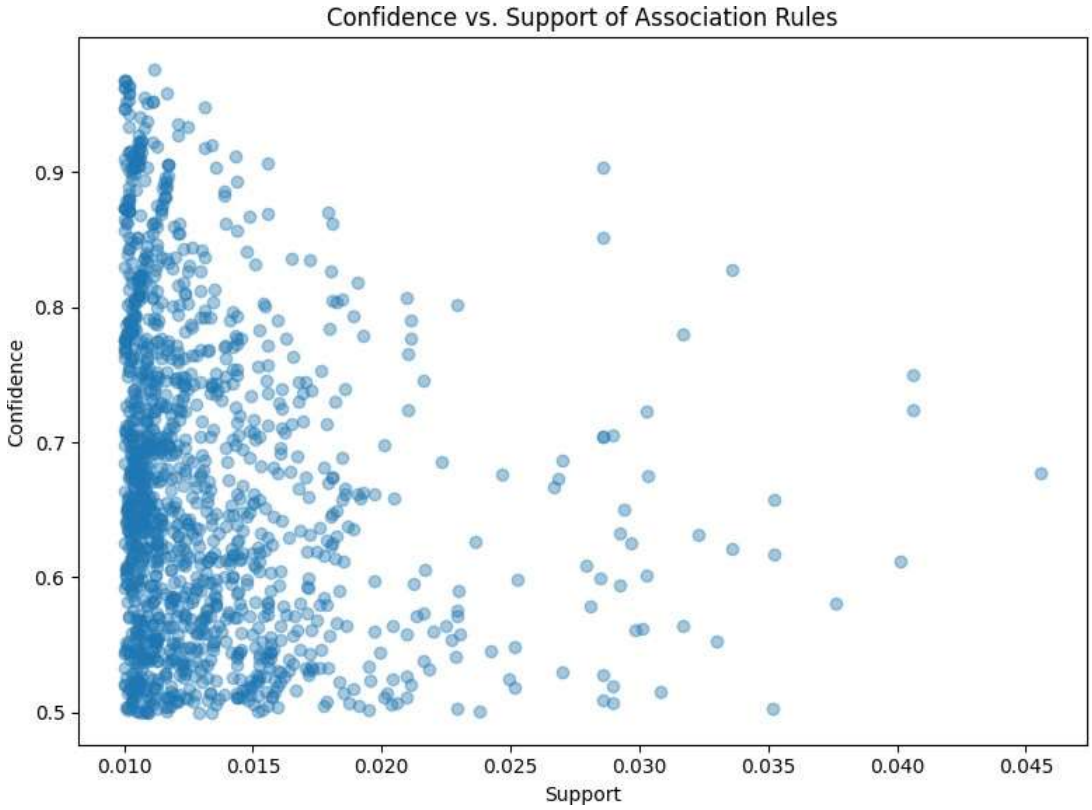
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(60 CAKE CASES DOLLY GIRL DESIGN)	(PACK OF 72 RETROSPOT CAKE CASES)	0.023160	0.071206	0.013028	0.562500	7.899629	0.011378	2.122958	0.894120
1	(60 TEATIME FAIRY CAKE CASES)	(PACK OF 72 RETROSPOT CAKE CASES)	0.044427	0.071206	0.024218	0.545113	7.655446	0.021054	2.041812	0.909794
2	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE GREEN)	0.023216	0.053558	0.015254	0.657074	12.268575	0.014011	2.759906	0.940321
3	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE PINK)	0.023216	0.042256	0.011691	0.503597	11.917802	0.010710	1.929369	0.937865
4	(ALARM CLOCK BAKELIKE CHOCOLATE)	(ALARM CLOCK BAKELIKE RED)	0.023216	0.057121	0.015811	0.681055	11.923112	0.014485	2.956246	0.937903
...	...	...	...	...	...	...	...	...	...	...
1392	(CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT...	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	0.018483	0.021824	0.011302	0.611446	28.017319	0.010898	2.517477	0.982467
1393	(CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE...	(CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...	0.018595	0.020989	0.011302	0.607784	28.957623	0.010911	2.496105	0.983760
1394	(CHARLOTTE BAG PINK POLKADOT, STRAWBERRY CHARL...	(CHARLOTTE BAG SUKI DESIGN, WOODLAND CHARLOTTE...	0.020989	0.018595	0.011302	0.538462	28.957623	0.010911	2.126378	0.986165
1395	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	(CHARLOTTE BAG SUKI DESIGN, STRAWBERRY CHARLOT...	0.021824	0.018483	0.011302	0.517857	28.017319	0.010898	2.035738	0.985822
1396	(WOODLAND CHARLOTTE BAG, STRAWBERRY CHARLOTTE ...	(CHARLOTTE BAG PINK POLKADOT, CHARLOTTE BAG SU...	0.022492	0.018261	0.011302	0.502475	27.516648	0.010891	1.973247	0.985832

```
Frequent Itemsets:
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
17	(BEADED CRYSTAL HEART PINK ON STICK)	(DOTCOM POSTAGE)	0.011469	0.039305	0.011190	0.975728	24.824404	0.010740	39.580626	0.970851
614	(HERB MARKER CHIVES, HERB MARKER THYME)	(HERB MARKER PARSLEY)	0.010411	0.012916	0.010077	0.967914	74.938272	0.009942	30.764113	0.997036
607	(HERB MARKER CHIVES, HERB MARKER ROSEMARY)	(HERB MARKER PARSLEY)	0.010355	0.012916	0.010021	0.967742	74.924917	0.009887	30.599599	0.996977
619	(HERB MARKER CHIVES, HERB MARKER ROSEMARY)	(HERB MARKER THYME)	0.010355	0.012916	0.010021	0.967742	74.924917	0.009887	30.599599	0.996977
1217	(HERB MARKER BASIL, HERB MARKER ROSEMARY, HERB...	(HERB MARKER THYME)	0.010578	0.012916	0.010188	0.963158	74.570009	0.010052	26.792276	0.997137
...	...	...	...	...	...	...	...	...	...	...
25	(RED RETROSPOT CUP)	(BLUE POLKADOT CUP)	0.021378	0.018038	0.010609	0.500000	27.719136	0.010304	1.963924	0.984981
1159	(STRAWBERRY CHARLOTTE BAG, RED RETROSPOT CHARL...	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	0.026834	0.021824	0.013417	0.500000	22.910714	0.012832	1.956352	0.982723
113	(HAND WARMER RED LOVE HEART)	(HAND WARMER SCOTTY DOG DESIGN)	0.021935	0.030286	0.010968	0.500000	16.509191	0.010303	1.939428	0.960496
147	(LOVE HOT WATER BOTTLE)	(HOT WATER BOTTLE KEEP CALM)	0.025832	0.042701	0.012916	0.500000	11.709257	0.011813	1.914597	0.938850
370	(CHARLOTTE BAG PINK POLKADOT, WOODLAND CHARLOT...	(PACK OF 72 RETROSPOT CAKE CASES)	0.021824	0.071206	0.010912	0.500000	7.021892	0.009358	1.857588	0.876722

4. **Phase 4 (AI Development Part 2)** – The association rules, i.e. confidence and support, are calculated and the results are ploted in a bar graph and scater plot respectively.





5. **Phase 5 (Project Documentation and Submission)** – Further cross-selling and upselling opportunities are explored and a conclusion is drawn. Now this program is ready to be used.

```
Cross-Selling Recommendations:
Customers who bought 'BEADED CRYSTAL HEART PINK ON STICK' also bought
'DOTCOM POSTAGE'.
Customers who bought 'HERB MARKER THYME' also bought 'HERB MARKER
ROSEMARY'.
Customers who bought 'HERB MARKER ROSEMARY' also bought 'HERB MARKER
THYME'.
Customers who bought 'HERB MARKER CHIVES' also bought 'HERB MARKER
PARSLEY'.
Customers who bought 'REGENCY TEA PLATE PINK' also bought 'REGENCY TEA
PLATE GREEN'.

Upselling Recommendations:
For customers who bought 'HERB MARKER CHIVES', recommend the following
upgrades: HERB MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'HERB MARKER CHIVES', recommend the following
upgrades: HERB MARKER PARSLEY, HERB MARKER MINT.
For customers who bought 'HERB MARKER CHIVES', recommend the following
upgrades: HERB MARKER ROSEMARY, HERB MARKER PARSLEY.
For customers who bought 'HERB MARKER CHIVES', recommend the following
upgrades: HERB MARKER ROSEMARY, HERB MARKER THYME.
For customers who bought 'HERB MARKER THYME', recommend the following
upgrades: HERB MARKER ROSEMARY, HERB MARKER PARSLEY.
```

For customers who bought 'HERB MARKER CHIVES', recommend the following upgrades: HERB MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'HERB MARKER THYME', recommend the following upgrades: HERB MARKER ROSEMARY, HERB MARKER PARSLEY.
For customers who bought 'HERB MARKER PARSLEY', recommend the following upgrades: HERB MARKER ROSEMARY, HERB MARKER THYME.
For customers who bought 'HERB MARKER ROSEMARY', recommend the following upgrades: HERB MARKER PARSLEY, HERB MARKER THYME.
For customers who bought 'REGENCY TEA PLATE PINK', recommend the following upgrades: REGENCY TEA PLATE GREEN, REGENCY TEA PLATE ROSES.

Loading the Dataset

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

```
import pandas as pd
dataset_path = “Assignment-1_Data.xlsx”
df = pd.read_excel(dataset_path)
```

Initial Exploration

We'll perform an initial exploration of the dataset to understand its structure and characteristics.

```
print("Number of rows and columns:", df.shape)
print("\nData Types and Missing Values:")
print(df.info())
print("\nFirst few rows of the dataset:")
print(df.head())
```

Preprocessing

We'll preprocess the data to ensure it's ready for analysis.

```
print("Missing Values:")
print(df.isnull().sum())
df.dropna(inplace=True)
```

```
transaction_data = df.groupby(['BillNo', 'Date'])['Itemname'].apply(lambda x: ', '.join(x)).reset_index()
columns_to_drop = ['BillNo', 'Date']
transaction_data.drop(columns=columns_to_drop, inplace=True)
transaction_data_path = '/kaggle/working/transaction_data.csv'
transaction_data.to_csv(transaction_data_path, index=False)
```

```
print("\nTransaction Data for Association Rule Mining:")
print(transaction_data.head())
transaction_data.shape
```

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.

```
items_df = transaction_data['Itemname'].str.split(', ', expand=True)
transaction_data = pd.concat([transaction_data, items_df], axis=1)
transaction_data = transaction_data.drop('Itemname', axis=1)
print(transaction_data.head())
```

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.

```
df_encoded = pd.get_dummies(transaction_data, prefix='', prefix_sep='').groupby(level=0,
axis=1).max()
df_encoded.to_csv('transaction_data_encoded.csv', index=False)
```

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.

```
df_encoded = pd.read_csv('transaction_data_encoded.csv')
from mlxtend.frequent_patterns import apriori, association_rules
frequent_itemsets = apriori(df_encoded, min_support=0.007, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence",
min_threshold=0.5)
print("Association Rules:")
print(rules.head())
```

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.

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

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.

```
import plotly.express as px
rules['antecedents'] = rules['antecedents'].apply(list)
rules['consequents'] = rules['consequents'].apply(list)
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'})
fig.update_layout(
    xaxis_title='Support',
    yaxis_title='Confidence',
    coloraxis_colorbar_title='Lift',
    showlegend=True
)
fig.show()
```

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

```
import networkx as nx
import matplotlib.pyplot as plt
import plotly.graph_objects as go
G = nx.DiGraph()
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'])
pos = nx.spring_layout(G)
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()
```

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.

```
import plotly.express as px
rules['rule'] = rules['antecedents'].astype(str) + ' -> ' + rules['consequents'].astype(str)

fig = px.sunburst(rules, path=['rule'], values='lift',
                 title='Market Basket Analysis - Sunburst Chart',
                 color='support', color_continuous_scale='YlGnBu')

fig.update_layout(
    margin=dict(l=0, r=0, b=0, t=40),
)

fig.show()
```

Complete Source Program :

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import networkx as nx
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
df = pd.read_excel("Assignment-1_Data.xlsx")
print("Number of rows and columns:", df.shape)
print("\nData Types and Missing Values:")
print(df.info())
print("\nFirst few rows of the dataset:")
print(df.head())
print("Missing Values:")
print(df.isnull().sum())
df.dropna(inplace=True)
df
transaction_data = df.groupby(['BillNo', 'Date'])['Itemname'].apply(lambda x: ', '.join(x)).reset_index()
columns_to_drop = ['BillNo', 'Date']
transaction_data.drop(columns=columns_to_drop, inplace=True)
transaction_data_path = 'C:/Users/akfla/Documents/Programs/transaction_data.csv'
```

```
transaction_data.to_csv(transaction_data_path, index=False)
print("\nTransaction Data for Association Rule Mining:")
print(transaction_data.head())
transaction_data.shape
items_df = transaction_data['Itemname'].str.split(', ', expand=True)
transaction_data = pd.concat([transaction_data, items_df], axis=1)
transaction_data = transaction_data.drop('Itemname', axis=1)
print(transaction_data.head())
df_encoded = pd.get_dummies(transaction_data, prefix='', prefix_sep='').groupby(level=0, axis=1).max()
df_encoded.to_csv('transaction_data_encoded.csv', index=False)
df_encoded = pd.read_csv('transaction_data_encoded.csv')
from mlxtend.frequent_patterns import apriori, association_rules
frequent_itemsets = apriori(df_encoded, min_support=0.007, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
print("Association Rules:")
print(rules.head())
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()
```

```
rules['antecedents'] = rules['antecedents'].apply(list)
rules['consequents'] = rules['consequents'].apply(list)
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'})
fig.update_layout(
    xaxis_title='Support',
    yaxis_title='Confidence',
    coloraxis_colorbar_title='Lift',
    showlegend=True
)
fig.show()
G = nx.DiGraph()
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'])
pos = nx.spring_layout(G)
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'
        )
    )
)
layout = go.Layout(
    showlegend=False,
    hovermode='closest',
    margin=dict(b=0, l=0, r=0, t=0),
)
```

```
fig = go.Figure(data=[edge_trace, node_trace], layout=layout)
fig.show()
rules['rule'] = rules['antecedents'].astype(str) + ' -> ' + rules['consequents'].astype(str)
fig = px.sunburst(rules, path=['rule'], values='lift',
                  title='Market Basket Analysis - Sunburst Chart',
                  color='support', color_continuous_scale='YlGnBu')
fig.update_layout(
    margin=dict(l=0, r=0, b=0, t=40),
)
fig.show()
```

Complete Program :

[35]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import networkx as nx
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
df = pd.read_excel("Assignment-1_Data.xlsx")
```

[16]:

[16]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...
522059	581587	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
522060	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
522061	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
522062	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
522063	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

522064 rows × 7 columns

[36]:

```python
print("Number of rows and columns:", df.shape)
print("\nData Types and Missing Values:")
print(df.info())
print("\nFirst few rows of the dataset:")
print(df.head())
```

```
Number of rows and columns: (522064, 7)

Data Types and Missing Values:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 522064 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      522064 non-null  object
 1   Itemname    520609 non-null  object
 2   Quantity    522064 non-null  int64
 3   Date        522064 non-null  datetime64[ns]
 4   Price       522064 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     522064 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 27.9+ MB
None

First few rows of the dataset:
   BillNo                             Itemname  Quantity                Date  \
0  536365   WHITE HANGING HEART T-LIGHT HOLDER         6 2010-12-01 08:26:00
1  536365                  WHITE METAL LANTERN         6 2010-12-01 08:26:00
2  536365       CREAM CUPID HEARTS COAT HANGER         8 2010-12-01 08:26:00
3  536365  KNITTED UNION FLAG HOT WATER BOTTLE         6 2010-12-01 08:26:00
4  536365       RED WOOLLY HOTTIE WHITE HEART.         6 2010-12-01 08:26:00

   Price  CustomerID         Country
0   2.55     17850.0  United Kingdom
1   3.39     17850.0  United Kingdom
2   2.75     17850.0  United Kingdom
3   3.39     17850.0  United Kingdom
4   3.39     17850.0  United Kingdom
```

[37]:

```python
print("Missing Values:")
print(df.isnull().sum())
df.dropna(inplace=True)
df
```

```
Missing Values:
BillNo             0
Itemname        1455
Quantity           0
Date               0
Price              0
CustomerID    134041
Country            0
dtype: int64
```

[37]:

	BillNo	Itemname	Quantity	Date	Price	CustomerID	Country
0	536365	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
...	...	...	...	...	...	...	...
522059	581587	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
522060	581587	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
522061	581587	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
522062	581587	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
522063	581587	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

388023 rows × 7 columns

[38]:
```
transaction_data = df.groupby(['BillNo', 'Date'])['Itemname'].apply(lambda x: ', '.join(x)).reset_index()
columns_to_drop = ['BillNo', 'Date']
transaction_data.drop(columns=columns_to_drop, inplace=True)
transaction_data_path = 'C:/Users/akfla/Documents/Programs/transaction_data.csv'
transaction_data.to_csv(transaction_data_path, index=False)
```

[39]:
```
print("\nTransaction Data for Association Rule Mining:")
print(transaction_data.head())
transaction_data.shape
```

```
Transaction Data for Association Rule Mining:
                                            Itemname
0  WHITE HANGING HEART T-LIGHT HOLDER, WHITE META...
1  HAND WARMER UNION JACK, HAND WARMER RED POLKA DOT
2  ASSORTED COLOUR BIRD ORNAMENT, POPPY'S PLAYHOU...
3  JAM MAKING SET WITH JARS, RED COAT RACK PARIS ...
4                            BATH BUILDING BLOCK WORD
```

[39]: (18192, 1)

[40]:
```
items_df = transaction_data['Itemname'].str.split(', ', expand=True)
transaction_data = pd.concat([transaction_data, items_df], axis=1)
transaction_data = transaction_data.drop('Itemname', axis=1)
print(transaction_data.head())
```

```
                                    0                             1    \
0  WHITE HANGING HEART T-LIGHT HOLDER           WHITE METAL LANTERN
1              HAND WARMER UNION JACK     HAND WARMER RED POLKA DOT
2       ASSORTED COLOUR BIRD ORNAMENT     POPPY'S PLAYHOUSE BEDROOM
3            JAM MAKING SET WITH JARS  RED COAT RACK PARIS FASHION
4            BATH BUILDING BLOCK WORD                          None

                              2                                    3    \
0  CREAM CUPID HEARTS COAT HANGER  KNITTED UNION FLAG HOT WATER BOTTLE
1                            None                                 None
2       POPPY'S PLAYHOUSE KITCHEN    FELTCRAFT PRINCESS CHARLOTTE DOLL
3  YELLOW COAT RACK PARIS FASHION          BLUE COAT RACK PARIS FASHION
4                            None                                 None
```

```
                               4                                    5    \
0  RED WOOLLY HOTTIE WHITE HEART.         SET 7 BABUSHKA NESTING BOXES
1                             None                                 None
2          IVORY KNITTED MUG COSY  BOX OF 6 ASSORTED COLOUR TEASPOONS
3                             None                                 None
4                             None                                 None

                                  6                              7    \
0  GLASS STAR FROSTED T-LIGHT HOLDER                            None
1                                None                            None
2        BOX OF VINTAGE JIGSAW BLOCKS  BOX OF VINTAGE ALPHABET BLOCKS
3                                None                            None
4                                None                            None

                         8                          9  ...   534   535   536  \
0                     None                       None  ...  None  None  None
1                     None                       None  ...  None  None  None
2  HOME BUILDING BLOCK WORD  LOVE BUILDING BLOCK WORD  ...  None  None  None
3                     None                       None  ...  None  None  None
4                     None                       None  ...  None  None  None

    537   538   539   540   541   542   543
0  None  None  None  None  None  None  None
1  None  None  None  None  None  None  None
2  None  None  None  None  None  None  None
3  None  None  None  None  None  None  None
4  None  None  None  None  None  None  None

[5 rows x 544 columns]
```

[41]:

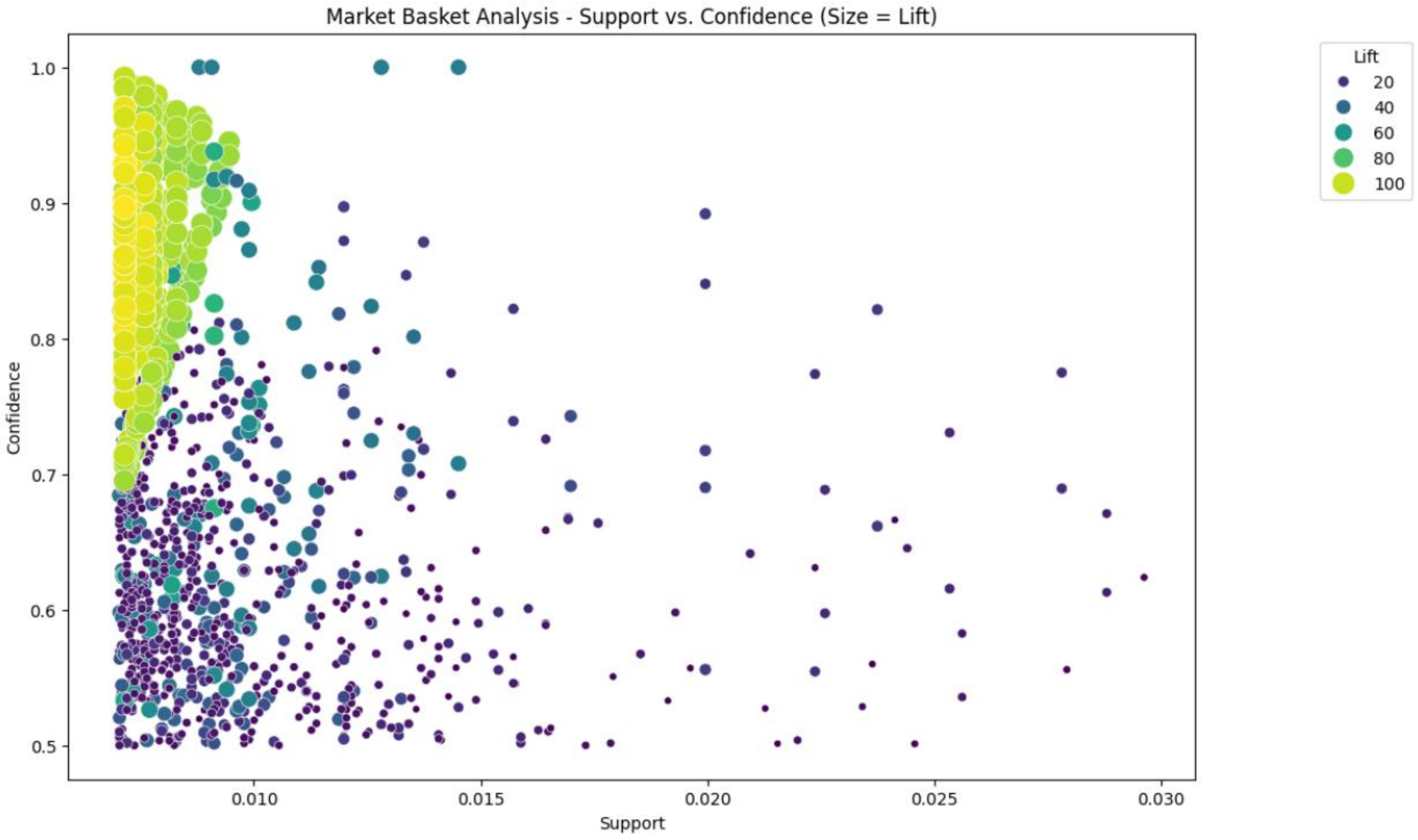
```python
df_encoded = pd.get_dummies(transaction_data, prefix='', prefix_sep='').groupby(level=0, axis=1).max()
df_encoded.to_csv('transaction_data_encoded.csv', index=False)
```

```
C:\Users\akfla\AppData\Local\Temp\ipykernel_16312\344312066.py:1: FutureWarning:

DataFrame.groupby with axis=1 is deprecated. Do `frame.T.groupby(...)` without axis instead.
```

```
[42]: df_encoded = pd.read_csv('transaction_data_encoded.csv')
from mlxtend.frequent_patterns import apriori, association_rules
frequent_itemsets = apriori(df_encoded, min_support=0.007, use_colnames=True)
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.5)
print("Association Rules:")
print(rules.head())
```

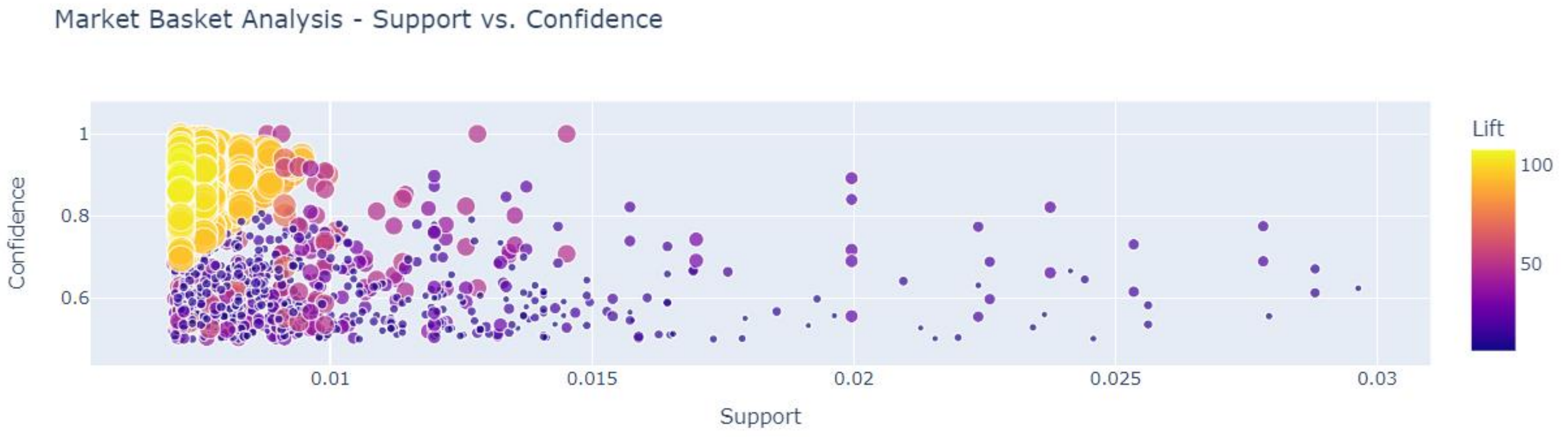
```
Association Rules:
                          antecedents                          consequents  \
0              (CHOCOLATE BOX RIBBONS)              (6 RIBBONS RUSTIC CHARM)
1    (60 CAKE CASES DOLLY GIRL DESIGN)  (PACK OF 72 RETROSPOT CAKE CASES)
2        (60 TEATIME FAIRY CAKE CASES)  (PACK OF 72 RETROSPOT CAKE CASES)
3     (ALARM CLOCK BAKELIKE CHOCOLATE)        (ALARM CLOCK BAKELIKE GREEN)
4     (ALARM CLOCK BAKELIKE CHOCOLATE)         (ALARM CLOCK BAKELIKE PINK)

   antecedent support  consequent support   support  confidence       lift  \
0            0.012368            0.039193  0.007036    0.568889  14.515044
1            0.018525            0.054529  0.010059    0.543027   9.958409
2            0.034631            0.054529  0.017315    0.500000   9.169355
3            0.017150            0.042931  0.011379    0.663462  15.454151
4            0.017150            0.032652  0.009125    0.532051  16.294742

   leverage  conviction  zhangs_metric
0  0.006551    2.228676       0.942766
1  0.009049    2.068984       0.916561
2  0.015427    1.890941       0.922902
3  0.010642    2.843862       0.951613
4  0.008565    2.067210       0.955009
```

```
[43]: 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()
```


Market Basket Analysis - Support vs. Confidence (Size = Lift)
Lift
20
40
60
80
100
Confidence
1.0
0.9
0.8
0.7
0.6
0.5
Support
0.010
0.015
0.020
0.025
0.030

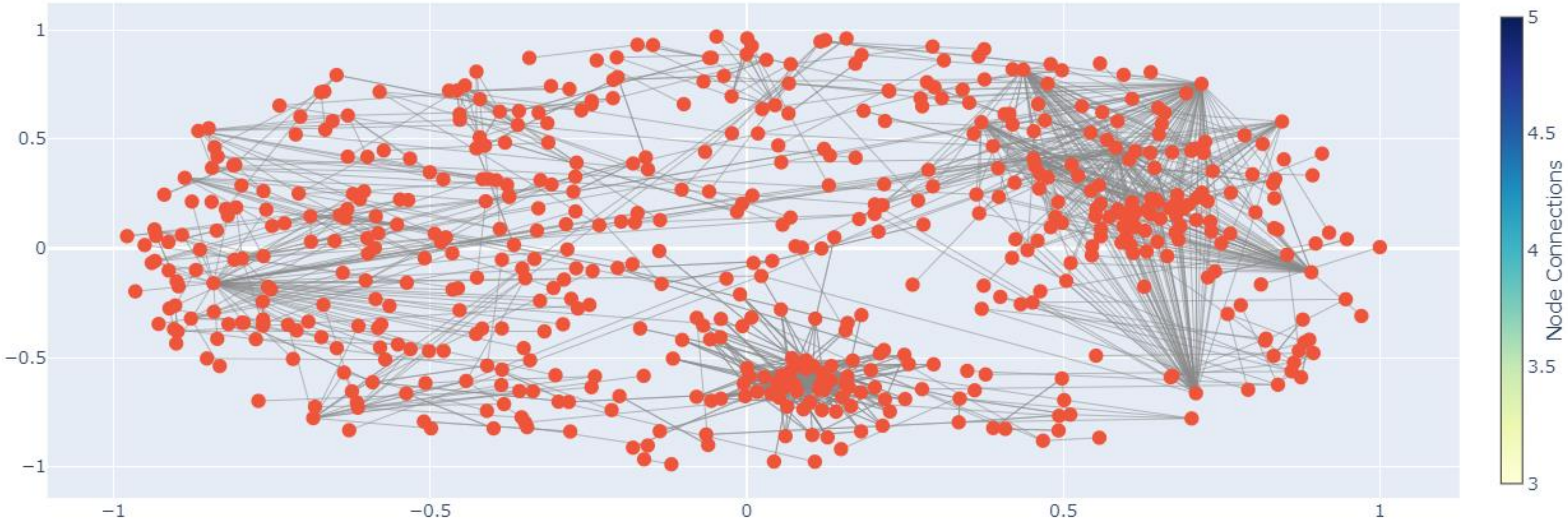
[44]:

```
rules['antecedents'] = rules['antecedents'].apply(list)
rules['consequents'] = rules['consequents'].apply(list)
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'})
fig.update_layout(
    xaxis_title='Support',
    yaxis_title='Confidence',
    coloraxis_colorbar_title='Lift',
    showlegend=True
)
fig.show()
```



```
[45]: G = nx.DiGraph()
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'])
pos = nx.spring_layout(G)
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,
```

```
            thickness=15,
            title='Node Connections',
            xanchor='left',
            titleside='right'
        )
    )
)
layout = go.Layout(
    showlegend=False,
    hovermode='closest',
    margin=dict(b=0, l=0, r=0, t=0),
)
fig = go.Figure(data=[edge_trace, node_trace], layout=layout)
fig.show()
```



[46]:
```
rules['rule'] = rules['antecedents'].astype(str) + ' -> ' + rules['consequents'].astype(str)
fig = px.sunburst(rules, path=['rule'], values='lift',
                  title='Market Basket Analysis - Sunburst Chart',
                  color='support', color_continuous_scale='YlGnBu')
fig.update_layout(
    margin=dict(l=0, r=0, b=0, t=40),
)
fig.show()
```