ARITIFICAL INTELLIGENCE

PHASE-5 SUBMISSION

Market Basket Insights

Data set link: https://www.kaggle.com/datasets/aslanahmedov/market basket analysis

Data Preprocessing Steps for Market Basket Insights:

1. Loading the Dataset:

    Load the dataset into a data analysis tool like Pandas.

    Ensure correct file path and character encoding for accurate data reading.

2. Data Cleaning:

    Remove irrelevant columns to reduce dimensionality and improve computational efficiency.

3. Handling Missing Values:

    Manage missing data, especially transactions without items.

    Decide whether to drop, impute, or address missing values based on dataset and analysis goals.

4. One Hot Encoding:

    Convert categorical data into a numerical format suitable for analysis.

    Utilize one hot encoding to represent each unique category as a binary column indicating its presence or absence.

5. Splitting the Dataset:

    Divide the dataset into training and test sets.

    The training set is used for model development, while the test set is reserved for evaluating model performance.

Data preprocessing is an iterative process that may require adjustments based on the specific dataset and analysis needs.

Step 1: Load Dataset
Code for the loading the dataset.

```python
import pandas as pd
df=pd.read_excel(r'D:\New folder\Assignment-1_Data.xlsx')
print(df.info())
```

Output for the above code:

```
<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
```

Step 2: Data Cleansing:

```python
import pandas as pd
df = pd.read_excel(r'D:\New folder\Assignment-1_Data.xlsx')
df.dropna(inplace=True)
print(df.info())
```

Code for the cleansing the dataset.

```
<class 'pandas.core.frame.DataFrame'>
Index: 388023 entries, 0 to 522063
Data columns (total 7 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   BillNo      388023 non-null  object
 1   Itemname    388023 non-null  object
 2   Quantity    388023 non-null  int64
 3   Date        388023 non-null  datetime64[ns]
 4   Price       388023 non-null  float64
 5   CustomerID  388023 non-null  float64
 6   Country     388023 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(3)
memory usage: 23.7+ MB
None
```

Output for the above code.

Step 3: Handling Missing Values:
Code for handling missing values:

```
import pandas as pd
df = pd.read_excel(r'D:\New folder\Assignment-1_Data.xlsx')
missing_values=df.isnull().sum()
print(missing_values)
```

Output:

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

Step 4: One Hot Encoding:
Code for one hot encoding.

```
import pandas as pd
df = pd.read_excel(r'D:\New folder\Assignment-1_Data.xlsx')
transaction_item_matrix = pd.get_dummies(df['Itemname']).groupby(df['BillNo']).max()
transaction_item_matrix.fillna(0, inplace=True)
print(transaction_item_matrix.head())
```

Output:

```
        *Boombox Ipod Classic  ...  wrongly sold sets
BillNo                         ...
536365                  False  ...              False
536366                  False  ...              False
536367                  False  ...              False
536368                  False  ...              False
536369                  False  ...              False

[5 rows x 4185 columns]
```

Step 5 :

```
import pandas as pd
df = pd.read_excel(r'D:\New folder\Assignment-1_Data.xlsx')
import pandas as pd
data = {
    'BillNo': [536365, 536365, 536365, 536366, 536366, 536367, 536367, 536367],
    'Itemname': [
        'WHITE HANGING HEART T-LIGHT HOLDER',
        'WHITE METAL LANTERN',
        'CREAM CUPID HEARTS COAT HANGER',
        'HAND WARMER UNION JACK',
        'HAND WARMER RED POLKA DOT',
        'ASSORTED COLOUR BIRD ORNAMENT',
        'POPPY\'S PLAYHOUSE BEDROOM',
        'POPPY\'S PLAYHOUSE KITCHEN'
    ]
}
df = pd.DataFrame(data)
dummy_df = pd.get_dummies(df, columns=['Itemname'])
print(dummy_df)
```

```
------------------------------------------------------------
   BillNo  ...  Itemname_WHITE METAL LANTERN
0  536365  ...                         False
1  536365  ...                          True
2  536365  ...                         False
3  536366  ...                         False
4  536366  ...                         False
5  536367  ...                         False
6  536367  ...                         False
7  536367  ...                         False

[8 rows x 9 columns]
```

Program:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori from
mlxtend.frequent_patterns import association_rules

# Load your dataset (replace 'your_dataset.csv' with the actual dataset file path)
data = pd.read_csv('Assignment-1_Data.csv', sep=';')

# Select relevant columns ('BillNo' and 'Itemname') data
= data[['BillNo', 'Itemname']]

# Convert the data into a one-hot encoded format basket =
(data.groupby(['BillNo', 'Itemname'])['Itemname']
        .count().unstack().reset_index().fillna(0)
        .set_index('BillNo'))

# Convert item counts to 1 or 0
basket_sets = basket.applymap(lambda x: 1 if x > 0 else 0) #
Convert item counts to boolean (True/False) values
basket_sets = basket.applymap(lambda x: x > 0).astype(bool)

frequent_itemsets = apriori(basket_sets, min_support=0.01, use_colnames=True)

# Generate association rules with a lower confidence threshold association_rules
= association_rules(frequent_itemsets, metric="lift", min_threshold=0.01)

# Filter and interpret the rules based on your specific requirements

# Print the frequent item sets
```

```python
print("Frequent Item Sets:")
print(frequent_itemsets)

# Print the association rules print("\nAssociation
Rules:")
print(association_rules)
```

Output:

Frequent Item Sets:

```
      support                                item sets
0     0.017248                   (10 COLOUR SPACEBOY PEN)
1     0.012094              (12 IVORY ROSE PEG PLACE SETTINGS)
2     0.013085              (12 MESSAGE CARDS WITH ENVELOPES)
3     0.017645                (12 PENCIL SMALL TUBE WOODLAND) 4     0.027359
      (12 PENCILS SMALL TUBE RED RETROSPOT)
...       ...                                   ...
2859  0.010508  (JUMBO BAG RED RETROSPOT, JUMBO  BAG BAROQUE B...
2860  0.010111  (JUMBO BAG OWLS, JUMBO BAG RED RETROSPOT, JUMB...
2861  0.010508  (JUMBO BAG WOODLAND ANIMALS, JUMBO BAG OWLS, J...
2862  0.010508  (JUMBO BAG OWLS, JUMBO BAG RED RETROSPOT, JUMB...
2863  0.010309  (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ...

[2864 rows x 2 columns]
```

Association Rules:

```
                              antecedents  \
0                 (12 PENCILS SMALL TUBE RED RETROSPOT)
1                 (12 PENCILS SMALL TUBE SKULL)
2                 (3 PIECE SPACEBOY COOKIE CUTTER SET)
3                 (GINGERBREAD MAN COOKIE CUTTER)
4                 (FELTCRAFT 6 FLOWER FRIENDS)
```

... ...

5935 (POPPY'S PLAYHOUSE BATHROOM, POPPY'S PLAYHOUSE...

5936 (POPPY'S PLAYHOUSE KITCHEN)

5937 (POPPY'S PLAYHOUSE LIVINGROOM)

5938 (POPPY'S PLAYHOUSE BATHROOM)

5939 (POPPY'S PLAYHOUSE BEDROOM)

**consequents antecedent support **

0 (12 PENCILS SMALL TUBE SKULL) 0.027359

1 (12 PENCILS SMALL TUBE RED RETROSPOT) 0.022205

2 (GINGERBREAD MAN COOKIE CUTTER) 0.025575

3 (3 PIECE SPACEBOY COOKIE CUTTER SET) 0.045202

4 (3 STRIPEY MICE FELTCRAFT) 0.039056

... ...

5935 (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ... 0.012292

5936 (POPPY'S PLAYHOUSE LIVINGROOM, POPPY'S PLAYHOU... 0.028945

5937 (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ... 0.022403

5938 (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ... 0.014869

5939 (POPPY'S PLAYHOUSE KITCHEN, POPPY'S PLAYHOUSE ...

0.028549

consequent support support confidence lift leverage \ 0

0.022205 0.015067 0.550725 24.802277 0.014460

1 0.027359 0.015067 0.678571 24.802277 0.014460

2 0.045202 0.010111 0.395349 8.746226 0.008955

3 0.025575 0.010111 0.223684 8.746226 0.008955 4 0.023394

0.010309 0.263959 11.283145 0.009396

... ...

5935 0.016257 0.010309 0.838710 51.590873 0.010109

5936 0.010904 0.010309 0.356164 32.663512 0.009994

5937 0.011499 0.010309 0.460177 40.019530 0.010052

5938 0.014869 0.010309 0.693333 46.628978 0.010088 5939 0.010309 0.010309 0.361111 35.027778 0.010015

conviction zhangs_metric 0

2.176383 0.986676

1 3.025993 0.981474

2 1.579089 0.908910

3 1.255192 0.927594

4 1.326837 0.948414

... ...

5935 6.099207 0.992820

5936 1.536255 0.998280

5937 1.831158 0.997356

5938 3.212383 0.993324

5939 1.549081 1.000000 [5940 rows x 10 columns]

Frequent Item Sets:

Support: Proportion of transactions containing the item set.

Itemsets: Lists frequent itemsets along with their support.

Association Rules:

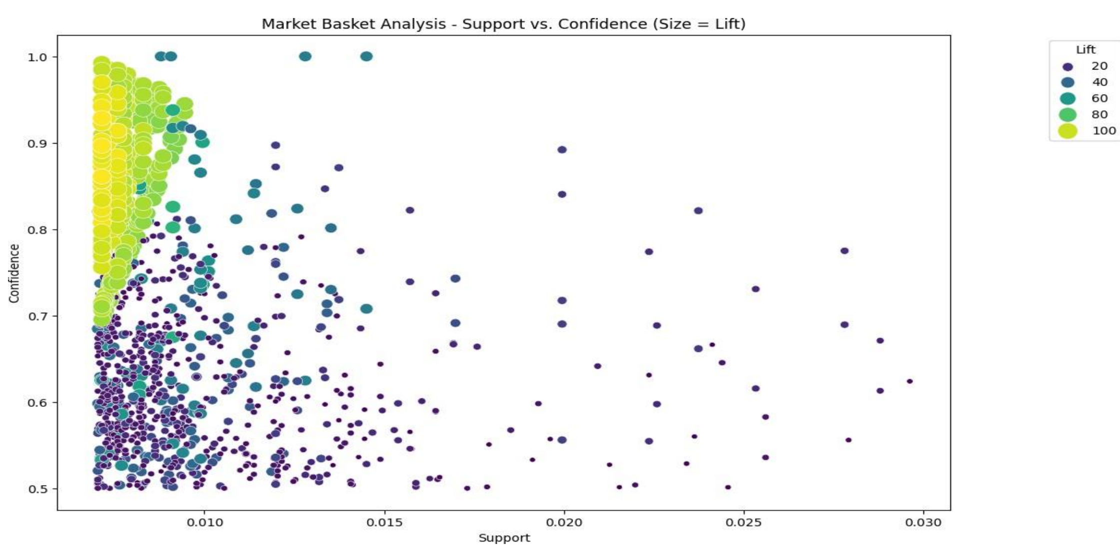
- **Antecedents**: Items on the left side of the rule.
- **Consequents**: Items on the right side of the rule.
- **Antecedent Support**: How often antecedent items appear in transactions.
- **Consequent Support**: How often consequent items appear in transactions.
- **Support**: Frequency of the entire rule in transactions.
- **Confidence**: Reliability of the rule (e.g., 55.07% means it's correct in 55.07% of cases).

- **Lift**: Indicates how much more likely consequent items are purchased when antecedent items are purchased, compared to when they're independent (lift > 1 suggests a positive association).
- **Leverage**: Measures the difference in support between antecedent and consequent appearing together and what's expected if they were independent (positive values show they're purchased together more often).
- **Conviction**: Indicates how much more likely the consequent is bought when the antecedent is true compared to when it's false (higher values suggest stronger associations).
- **Zhang's Metric**: Another measure of rule interest, with higher values indicating stronger associations.

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.

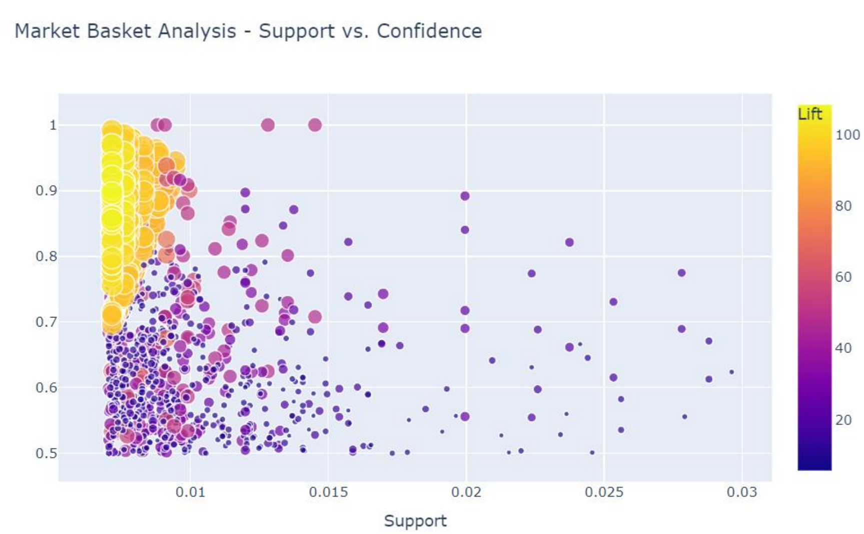
```
import matplotlib.pyplot as plt
import seaborn as sns


# 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")
```



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


# 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()
```



TEAM MEMBERS:

1. J. ARUN KUMAR-813821205002
2. C. MURUGANANTHAN-813821205033
3. T. SRIHARIHARAN-813821205049
4. S. VENGADASHAN-813821205055
5. S. VISHWA-813821205059