Experiment 8 - Association Mining using Python

Roll No.	19
Name	Manav Jawrani
Class	D15A
Subject	Business Intelligence Lab
LO Mapped	LO2: Organize and prepare the data needed for data mining algorithms in terms of attributes and class inputs, training, validating, and testing files. LO4: Implement various data mining algorithms from scratch using languages like Python/ Java etc.

10

Page No.
Date

Experiment 8

Aim- To perform association mining using python.

Theory-

- Python Library functions used are-

1. apyori - Library for implementing Apriori algorithm
2. association_rules - A function from mlxtend that extracts rules from frequent itemsets
3. LabelEncoder - Preprocessing utility from sklearn that converts categorical values to numeric rules.
4. matplotlib.pyplot - Plotting library for creating visualizations.
5. mlxtend.frequent_patterns.apriori - A function for generating frequent itemsets from transactional data.
6. networkx - Library for creating and manipulating graphs and networks.
7. Pandas - Library for data manipulation and analysis.
8. preprocessing - module from sklearn containing function for data preprocessing.
9. train_test_split - A function from sklearn for splitting data into training and testing subsets.

Page No.
Date

Observation -

No.of rules generated = 10.

Conclusion -

We can see that the no.of rules generated by python is 10 which indicates that python is not able to process this complex dataset whereas when we performed the same on rapidminer it had generated 23 rules which shows rapidminer is able to process the given data set.

Implementation -

```
from google.colab import files
uploaded = files.upload()
```

Choose Files healthcare-…-data (1).csv

- **healthcare-dataset-stroke-data (1).csv**(text/csv) - 293610 bytes, last modified: 4/2/2023 - 100% done

```
Saving healthcare-dataset-stroke-data (1).csv to healthcare-dataset-stroke-data (1).csv
```

```
[10] import pandas as pd
     dataset = pd.read_csv("healthcare-dataset-stroke-data (1).csv")
     dataset.describe()
```

	age	hypertension	heart_disease	avg_glucose_level	bmi	stroke
count	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000	5109.000000
mean	43.229986	0.097475	0.054022	106.140399	28.894560	0.048738
std	22.613575	0.296633	0.226084	45.285004	7.698235	0.215340
min	0.080000	0.000000	0.000000	55.120000	10.300000	0.000000
25%	25.000000	0.000000	0.000000	77.240000	23.800000	0.000000
50%	45.000000	0.000000	0.000000	91.880000	28.400000	0.000000
75%	61.000000	0.000000	0.000000	114.090000	32.800000	0.000000
max	82.000000	1.000000	1.000000	271.740000	97.600000	1.000000

```
[11] # Split dataset into training set and test set
     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(dataset[['age', 'avg_glucose_level']], dataset[['bmi']], test_size=0.33, random_state=0)
```

```
[12] from sklearn import preprocessing

     X_train_norm = preprocessing.normalize(X_train)
     X_test_norm = preprocessing.normalize(X_test)
```

```
[38] !pip install apyori

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5976 sha256=7b8d036c3e614c20c9930ba4f3ab7761ef7bbcbd29
  Stored in directory: /root/.cache/pip/wheels/32/2a/54/10c595515f385f3726642b10c60bf788029e8f3a1323e3913a
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
[39] from apyori import apriori

     # Generate frequent itemsets using the Apriori algorithm
     frequent_itemsets = apriori(transactions, min_support=0.05, min_confidence=0.7, min_lift=1.2, min_length=2)
```

Encoding columns to 0 and 1

```
[55] from sklearn.preprocessing import LabelEncoder

     # create an instance of LabelEncoder
     le = LabelEncoder()

     # fit and transform the column 'my_col' to 0 and 1
     dataset['gender_encoded'] = le.fit_transform(dataset['gender'])

     # print the encoded values
     print(dataset['gender_encoded'])

0       1
1       0
2       1
3       0
4       0
       ..
5104    0
5105    0
5106    0
5107    1
5108    0
Name: gender_encoded, Length: 5109, dtype: int64
```

```
[56] from sklearn.preprocessing import LabelEncoder

     # create an instance of LabelEncoder
     le = LabelEncoder()

     # fit and transform the column 'my_col' to 0 and 1
     dataset['ever_married_encoded'] = le.fit_transform(dataset['ever_married'])

     # print the encoded values
     print(dataset['ever_married_encoded'])

0       1
1       1
2       1
3       1
4       1
       ..
5104    1
5105    1
5106    1
5107    1
5108    1
Name: ever_married_encoded, Length: 5109, dtype: int64
```

Generating association rules

```
[69] from mlxtend.frequent_patterns import apriori
     from mlxtend.frequent_patterns import association_rules
     # Find frequent itemsets with minimum support of 0.1
     frequent_itemsets = apriori(dataset, min_support=0.1, use_colnames=True)

     # Generate association rules with minimum confidence of 0.8
     rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.8)

     print(rules)

Empty DataFrame
Columns: [antecedents, consequents, antecedent support, consequent support, support, confidence, lift, leverage, conviction]
Index: []
```

```
[70] import pandas as pd
     from mlxtend.frequent_patterns import apriori
     from mlxtend.frequent_patterns import association_rules


     # Apply Apriori algorithm to find frequent itemsets
     frequent_itemsets = apriori(dataset, min_support=0.01, use_colnames=True)

     # Generate association rules
     rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)

     # Display top 10 rules
     print(rules.head(10))
```

```
           antecedents            consequents  antecedent support  \
0       (hypertension)        (heart_disease)            0.097475
1      (heart_disease)         (hypertension)            0.054022
2       (hypertension)               (stroke)            0.097475
3             (stroke)         (hypertension)            0.048738
4     (gender_encoded)         (hypertension)            0.413975
5       (hypertension)       (gender_encoded)            0.097475
6       (hypertension)  (ever_married_encoded)           0.097475
7 (ever_married_encoded)       (hypertension)            0.664905
8     (gender_encoded)        (heart_disease)            0.413975
9      (heart_disease)       (gender_encoded)            0.054022

   consequent support   support  confidence      lift  leverage  conviction
0            0.054022  0.012527    0.128514  2.378907  0.007261    1.085477
1            0.097475  0.012527    0.231884  2.378907  0.007261    1.174985
2            0.048738  0.012918    0.132530  2.719263  0.008168    1.096594
3            0.097475  0.012918    0.265060  2.719263  0.008168    1.228026
4            0.097475  0.043453    0.104965  1.076835  0.003100    1.008368
5            0.413975  0.043453    0.445783  1.076835  0.003100    1.057392
6            0.664905  0.089450    0.917671  1.380153  0.024638    4.070177
7            0.097475  0.089450    0.134530  1.380153  0.024638    1.042815
8            0.054022  0.031904    0.077069  1.426606  0.009541    1.024971
9            0.413975  0.031904    0.590580  1.426606  0.009541    1.431352
```

[71]
```python
import matplotlib.pyplot as plt
import networkx as nx

# Plot the association rules as a graph
G = nx.DiGraph()
edges = [(rule['antecedents'], rule['consequents'], {'label': round(rule['lift'], 2)}) for _, rule in rules.iterrows()]
G.add_edges_from(edges)

pos = nx.spring_layout(G, k=0.5)
nx.draw_networkx_nodes(G, pos, node_size=2000, node_color='lightblue')
nx.draw_networkx_edges(G, pos, edge_color='gray')
nx.draw_networkx_labels(G, pos, font_size=12, font_family='sans-serif')
nx.draw_networkx_edge_labels(G, pos, font_size=10, font_family='sans-serif')

plt.axis('off')
plt.show()
```