DIRE DAWA UNIVERSITY INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTING
DEPARTMENT OF COMPUTER SCIENCE
Post Graduate Program (Msc) in Computer Science
Course Title: Data Mining and Warehousing
Course Code: COSC 623
Assignment 3: Association Rule Mining
Course Assignment

By

Full Name	ID
Tadesse Shefera	DDU1400693

Submitted to: Gaddisa Olani. (PhD)

Contents

Overview of given dataset

The given data was 1000 instance with single column. It also in text format. Even a single transaction has multiple items, all items treated as a single data value in pandas. So, we must split every data item as single value in one transaction.

```
In [7]: import pandas as pd
        df = pd.read_table('supermarketdata.txt', header=None)
        df.shape

Out[7]: (1000, 1)
```

Preprocessing

Split every data item in a single transaction to evaluate individually

	0
0	5 15 32 61 78
1	11 12 21 64 87
2	16 45 55 64 66
3	20 51 55 68 74

For single transaction these 5 items considered as single value.

There is space between data items and we use it for splitting purpose

```
In [15]: # store all 1000 transaction in all_transaction
         all_transaction=[]
         for i in range (total_transaction):
             single_transaction=df.iloc[i][0]
             single_transaction_splited=single_transaction.split( )
             all_transaction.append(single_transaction_splited)
         all_transaction

Out[15]: [['5', '15', '32', '61', '78'],
          ['11', '12', '21', '64', '87'],
          ['16', '45', '55', '64', '66'],
          ['20', '51', '55', '68', '74'],
          ['8', '19', '24', '31', '95'],
          ['25', '40', '49', '58', '97'],
          ['16', '42', '56', '71', '73'],
          ['14', '38', '50', '68', '82'],
          ['7', '14', '37', '70'],
          ['17', '39', '50', '75', '79'],
          ['24', '28', '56', '60', '62'],
```

Now we have list of all transaction with list

Encoding

We try to encode list of transaction using "TransactionEncoder" and convert it to data frame

```
In [24]: #import necessary library and encode
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
transaction_encoder = TransactionEncoder()
transaction_array = transaction_encoder.fit(all_transaction).transform(all_transaction)
df = pd.DataFrame(transaction_array, columns=transaction_encoder.columns_)
```

Frequent itemset by support

Using apriori algorithm we try to find the association which satisfy the minimum support that accepted from user.

```
In [14]: #accepting minimum confidence and support
minimum_suport=float(input("Enter minimum support in %: "))/100
minimum_confidence=float(input("Enter minimum confidence in %: "))/100
print(f"You enter minimum_suport:{minimum_suport*100}% and minimum_confidence:{minimum_confidence*100}%")

Enter minimum support in %: 1
Enter minimum confidence in %: 0.1
You enter minimum_suport:1.0% and minimum_confidence:0.1%
```

```
In [25]: frequent_itemsets = apriori(df, min_support=minimum_suport, use_colnames=True)
frequent_itemsets
```

Out[25]:

	support	itemsets
0	0.037	(0)
1	0.074	(1)
2	0.073	(10)
3	0.040	(100)

Frequent itemset by support and confidence

By using the above support result we are going to find association that satisfy minimum confidence that accepted from user

```
In [27]: from mlxtend.frequent_patterns import association_rules
associationRules=association_rules(frequent_itemsets, metric="confidence", min_threshold=minimum_confidence)
associationRules
```

Out[27]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(11)	(1)	0.079	0.074	0.01	0.126582	1.710571	0.004154	1.060203
1	(1)	(11)	0.074	0.079	0.01	0.135135	1.710571	0.004154	1.064906
2	(1)	(22)	0.074	0.070	0.01	0.135135	1.930502	0.004820	1.075312
3	(22)	(1)	0.070	0.074	0.01	0.142857	1.930502	0.004820	1.080333

Formatting result based on Assignment instruction

For report purpose we need 4 attributes from above result (antecedents, consequents, support and confidence)

```
In [9]: """
        =>from above dataframe We need only 4 variables [antecedents, consequents, suppo
        =>Then we are going to filiter it
        """
        ass_rule=associationRules[['antecedents','consequents','support','confidence']]
        ass_rule.head(5)
```

Out[9]:

	antecedents	consequents	support	confidence
0	(72)	(1)	0.013	0.213115
1	(16)	(98)	0.015	0.217391
2	(26)	(41)	0.017	0.217949
3	(41)	(26)	0.017	0.202381

To access frozenset it shoud be convert to list and we try to display in the following format.

```
In [29]: #to access frozenset we try convert to list
         ass_rule=ass_rule.values.tolist()#run this line only one time
         print("######################################################################")
         print("Rule     \t\tConfidence\t\tSupport")
         print("######################################################################")
         for i in range (len(ass_rule)):
             print(f"{list(ass_rule[i][0])[0]} ==> {list(ass_rule[i][1])[0]}",
                   f"\t\t({round(ass_rule[i][3]*100,1)}%)",
                   f"\t\t({round(ass_rule[i][2]*100,1)}%)")
```

```
######################################################################
Rule                    Confidence              Support
######################################################################
72 ==> 1                (21.3%)                 (1.3%)
16 ==> 98               (21.7%)                 (1.5%)
26 ==> 41               (21.8%)                 (1.7%)
41 ==> 26               (20.2%)                 (1.7%)
```

Summery

In this assignment we apply different data preprocessing and we try to find association rule for given supermarket dataset by accepting minimum support and minimum confidence from user.

From given dataset we observe that the existence of data item in many transactions is less. And we decided the strong rule is rules is “26->41” with 1.7% support and 21.8% confidence.