DMW_p6.ipynb

File Edit View Insert Runtime Tools Help All changes saved

21162101012_CBA_Yash_Lakhtariya

DMW Practical 6

Question 1:

Assume a dataset of 10 transactions which has the list of the items that have been bought for each transactions. So the dataset will be having two kind of information in the dataset that is – (1) Transaction ID and (2) List of items. Find the support value of the each combination of the items.

```
[46] !pip install mlxtend
```

```
[47] import warnings
     warnings.filterwarnings("ignore", category=DeprecationWarning)
```

```
[48] data1 = [['Milk','Onion','Nutmug','Kidney Beans','Eggs','Yogurt'],
              ['Dill','Onion','Nutmug','Kidney Beans','Eggs','Yogurt'],
              ['Milk','Apple','Kidney Beans','Eggs'],
              ['Milk','Unicorn','Corn','Kidney Beans','Yogurt'],
              ['Corn','Onion','Kidney Beans','Icecream','Eggs']]
```

```
[49] data1

[['Milk', 'Onion', 'Nutmug', 'Kidney Beans', 'Eggs', 'Yogurt'],
 ['Dill', 'Onion', 'Nutmug', 'Kidney Beans', 'Eggs', 'Yogurt'],
 ['Milk', 'Apple', 'Kidney Beans', 'Eggs'],
 ['Milk', 'Unicorn', 'Corn', 'Kidney Beans', 'Yogurt'],
 ['Corn', 'Onion', 'Kidney Beans', 'Icecream', 'Eggs']]
```

```
[50] import pandas as pd
     from mlxtend.preprocessing import TransactionEncoder
```

```
[51] encoder = TransactionEncoder()
     encoded_data = encoder.fit(data1).transform(data1)
     encoded_data

array([[False, False, False,  True, False,  True,  True,  True,  True,
        False,  True],
       [False, False,  True,  True, False,  True, False,  True,  True,
        False,  True],
       [ True, False, False,  True, False,  True,  True, False, False,
        False, False],
       [False,  True, False, False, False,  True,  True, False, False,
         True,  True],
       [False,  True, False,  True,  True,  True, False, False,  True,
        False, False]])
```

```
df1 = pd.DataFrame(encoded_data,columns=encoder.columns_)
df1
```

	Apple	Corn	Dill	Eggs	Icecream	Kidney Beans	Milk	Nutmug	Onion	Unicorn	Yogurt
0	False	False	False	True	False	True	True	True	True	False	True
1	False	False	True	True	False	True	False	True	True	False	True
2	True	False	False	True	False	True	True	False	False	False	False
3	False	True	False	False	False	True	True	False	False	True	True
4	False	True	False	True	True	True	False	False	True	False	False

Next steps: View recommended plots

```
[53] test_data = encoded_data[:5]
     encoder.inverse_transform(test_data)
     encoded_data.astype('int')

array([[0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1],
       [0, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1],
       [1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0],
       [0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1],
       [0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0]])
```

```
[54] encoder.columns_

['Apple',
 'Corn',
 'Dill',
 'Eggs',
 'Icecream',
 'Kidney Beans',
 'Milk',
 'Nutmug',
 'Onion',
 'Unicorn',
 'Yogurt']
```

```
[55] from mlxtend.frequent_patterns import apriori
     apriori(df1,min_support =0.4)
```

	support	itemsets
0	0.4	(1)
1	0.8	(3)
2	1.0	(5)
3	0.6	(6)
4	0.4	(7)
5	0.6	(8)
6	0.6	(10)
7	0.4	(1, 5)
8	0.8	(3, 5)
9	0.4	(3, 6)
10	0.4	(3, 7)
11	0.6	(8, 3)
12	0.4	(10, 3)
13	0.6	(5, 6)
14	0.4	(5, 7)
15	0.6	(8, 5)
16	0.6	(10, 5)
17	0.4	(10, 6)
18	0.4	(8, 7)
19	0.4	(10, 7)
20	0.4	(8, 10)
21	0.4	(3, 5, 6)
22	0.4	(3, 5, 7)
23	0.6	(8, 3, 5)
24	0.4	(10, 3, 5)
25	0.4	(8, 3, 7)
26	0.4	(10, 3, 7)
27	0.4	(8, 10, 3)
28	0.4	(10, 5, 6)
29	0.4	(8, 5, 7)
30	0.4	(10, 5, 7)

```
[56] apriori(df1,min_support =0.1)
```

	support	itemsets
0	0.2	(0)
1	0.4	(1)
2	0.2	(2)
3	0.8	(3)
4	0.2	(4)
...	...	...
144	0.4	(3, 5, 7, 8, 10)
145	0.2	(3, 6, 7, 8, 10)
146	0.2	(5, 6, 7, 8, 10)
147	0.2	(2, 3, 5, 7, 8, 10)
148	0.2	(3, 5, 6, 7, 8, 10)

149 rows × 2 columns

```
[57] apriori(df1, min_support = 0.4, use_colnames=True)
```

	support	itemsets
7	0.4	(Kidney Beans, Corn)
8	0.8	(Kidney Beans, Eggs)
9	0.4	(Milk, Eggs)
10	0.4	(Nutmug, Eggs)
11	0.6	(Onion, Eggs)
12	0.4	(Yogurt, Eggs)
13	0.6	(Kidney Beans, Milk)
14	0.4	(Kidney Beans, Nutmug)
15	0.6	(Kidney Beans, Onion)
16	0.6	(Kidney Beans, Yogurt)
17	0.4	(Milk, Yogurt)
18	0.4	(Onion, Nutmug)
19	0.4	(Nutmug, Yogurt)
20	0.4	(Onion, Yogurt)
21	0.4	(Kidney Beans, Milk, Eggs)
22	0.4	(Kidney Beans, Nutmug, Eggs)
23	0.6	(Kidney Beans, Onion, Eggs)
24	0.4	(Kidney Beans, Yogurt, Eggs)
25	0.4	(Onion, Nutmug, Eggs)
26	0.4	(Nutmug, Yogurt, Eggs)
27	0.4	(Onion, Yogurt, Eggs)
28	0.4	(Kidney Beans, Milk, Yogurt)
29	0.4	(Kidney Beans, Onion, Nutmug)
30	0.4	(Kidney Beans, Nutmug, Yogurt)
31	0.4	(Kidney Beans, Onion, Yogurt)
32	0.4	(Onion, Yogurt, Nutmug)
33	0.4	(Kidney Beans, Onion, Nutmug, Eggs)
34	0.4	(Kidney Beans, Nutmug, Yogurt, Eggs)
35	0.4	(Kidney Beans, Onion, Yogurt, Eggs)
36	0.4	(Onion, Yogurt, Eggs, Nutmug)
37	0.4	(Kidney Beans, Onion, Yogurt, Nutmug)
38	0.4	(Onion, Nutmug, Eggs, Yogurt, Kidney Beans)

```
[58] frequent_items = apriori(df1, min_support=0.4, use_colnames=True)
     frequent_items['length'] = frequent_items['itemsets'].apply(lambda x: len(x))
```

```
[59] frequent_items
```

	support	itemsets	length
0	0.4	(Corn)	1
1	0.8	(Eggs)	1
2	1.0	(Kidney Beans)	1
3	0.6	(Milk)	1
4	0.4	(Nutmug)	1
5	0.6	(Onion)	1
6	0.6	(Yogurt)	1
7	0.4	(Kidney Beans, Corn)	2
8	0.8	(Kidney Beans, Eggs)	2
9	0.4	(Milk, Eggs)	2
10	0.4	(Nutmug, Eggs)	2
11	0.6	(Onion, Eggs)	2
12	0.4	(Yogurt, Eggs)	2
13	0.6	(Kidney Beans, Milk)	2
14	0.4	(Kidney Beans, Nutmug)	2
15	0.6	(Kidney Beans, Onion)	2
16	0.6	(Kidney Beans, Yogurt)	2
17	0.4	(Milk, Yogurt)	2
18	0.4	(Onion, Nutmug)	2
19	0.4	(Nutmug, Yogurt)	2
20	0.4	(Onion, Yogurt)	2
21	0.4	(Kidney Beans, Milk, Eggs)	3
22	0.4	(Kidney Beans, Nutmug, Eggs)	3
23	0.6	(Kidney Beans, Onion, Eggs)	3
24	0.4	(Kidney Beans, Yogurt, Eggs)	3
25	0.4	(Onion, Nutmug, Eggs)	3
26	0.4	(Nutmug, Yogurt, Eggs)	3
27	0.4	(Onion, Yogurt, Eggs)	3
28	0.4	(Kidney Beans, Milk, Yogurt)	3
29	0.4	(Kidney Beans, Onion, Nutmug)	3
30	0.4	(Kidney Beans, Nutmug, Yogurt)	3

Next steps: View recommended plots

```
[60] frequent_items[frequent_items['itemsets'].apply(lambda x: 'Milk' in x and 'Eggs' in x)]
```

	support	itemsets	length
9	0.4	(Milk, Eggs)	2
21	0.4	(Kidney Beans, Milk, Eggs)	3

```
[61] frequent_items[frequent_items['itemsets'] == {'Milk','Eggs'}]
```

	support	itemsets	length
9	0.4	(Milk, Eggs)	2

```
[62] frequent_items[(frequent_items['length'] == 2) & (frequent_items['support'] >= 0.6)]
```

	support	itemsets	length
8	0.8	(Kidney Beans, Eggs)	2
11	0.6	(Onion, Eggs)	2
13	0.6	(Kidney Beans, Milk)	2
15	0.6	(Kidney Beans, Onion)	2
16	0.6	(Kidney Beans, Yogurt)	2

Question 2:

Suppose a superstore management wants to analyze buying patterns of the customers and come up with a strategy to increase the profit.
The dataset contains details related to transactions.

```
[63] import io
     import pandas as pd
```

```
[64] data2 = pd.read_excel('OnlineRetail.xlsx')
     data2.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

1. As a data analyst, analyze the data and try to find the correlation between the items.

```
[65] from mlxtend.frequent_patterns import apriori
     from mlxtend.frequent_patterns import association_rules
```

```
[66] basket = (data2.groupby(['InvoiceNo', 'Description'])['Quantity']
               .sum().unstack().reset_index().fillna(0)
               .set_index('InvoiceNo'))
```

```
[67] def encode_units(x):
         if x <= 0:
             return 0
         if x >= 1:
             return 1

     basket_sets = basket.applymap(encode_units)

     frequent_itemsets = apriori(basket_sets, min_support=0.05, use_colnames=True)
```

```
[68] print(frequent_itemsets)

    support                             itemsets
0  0.059519       (ASSORTED COLOUR BIRD ORNAMENT)
1  0.085576             (JUMBO BAG RED RETROSPOT)
2  0.052074             (LUNCH BAG  BLACK SKULL.)
3  0.063978             (LUNCH BAG RED RETROSPOT)
4  0.051092     (NATURAL SLATE HEART CHALKBOARD )
5  0.053997     (PACK OF 72 RETROSPOT CAKE CASES)
6  0.068968                       (PARTY BUNTING)
7  0.081363            (REGENCY CAKESTAND 3 TIER)
8  0.056655   (SET OF 3 CAKE TINS PANTRY DESIGN )
9  0.092449  (WHITE HANGING HEART T-LIGHT HOLDER)
```

```
[69] print(data2.dtypes)
     numeric_data = data2.select_dtypes(include=['int64', 'float64'])
     item_correlation = numeric_data.corr()
     item_correlation

InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
dtype: object
```

	Quantity	UnitPrice	CustomerID
Quantity	1.000000	-0.001235	-0.00360
UnitPrice	-0.001235	1.000000	-0.00456
CustomerID	-0.003600	-0.004560	1.00000

Next steps: View recommended plots

2. Find the most popular items among the customers.

```
[70] popular_items = data2['Description'].value_counts().head(10)
     popular_items

Description
WHITE HANGING HEART T-LIGHT HOLDER    2369
REGENCY CAKESTAND 3 TIER              2200
JUMBO BAG RED RETROSPOT               2159
PARTY BUNTING                         1727
LUNCH BAG RED RETROSPOT               1638
ASSORTED COLOUR BIRD ORNAMENT         1501
SET OF 3 CAKE TINS PANTRY DESIGN      1473
PACK OF 72 RETROSPOT CAKE CASES       1385
LUNCH BAG  BLACK SKULL.               1350
NATURAL SLATE HEART CHALKBOARD        1280
Name: count, dtype: int64
```

3. Which items' combinations are bought on most frequent basis?

```
[71] item_combinations = data2.groupby('InvoiceNo')['Description'].apply(list)
     item_combinations_counts = item_combinations.value_counts().head(10)
     item_combinations
     item_combinations_counts

Description
[nan]                          1454
[Manual]                        233
[check]                         159
[POSTAGE]                       141
[REGENCY CAKESTAND 3 TIER]       73
[Discount]                       57
[?]                              47
[damages]                        45
[damaged]                        43
[Bank Charges]                   35
Name: count, dtype: int64
```

4. Mention the items and combinations of the items whose sale should be more focused?

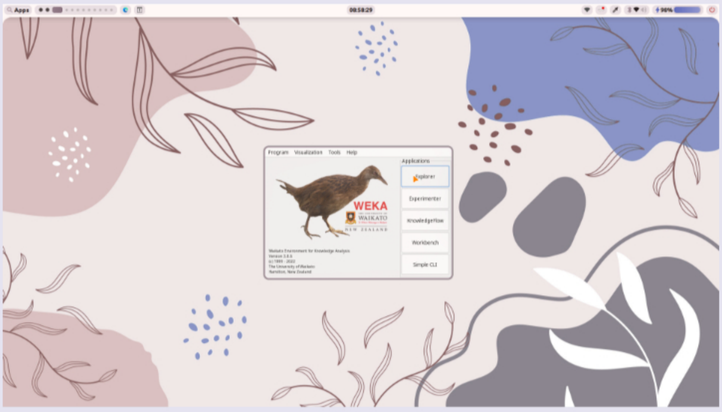
```
[72] high_price_items = data2[data2['UnitPrice'] > data2['UnitPrice'].quantile(0.95)]['Description'].value_counts().head(5)
     high_price_items

Description
REGENCY CAKESTAND 3 TIER       2194
POSTAGE                        1163
DOTCOM POSTAGE                  696
CREAM SWEETHEART MINI CHEST     553
RED RETROSPOT CAKE STAND        542
Name: count, dtype: int64
```


✓ 0s completed at 12:29 PM

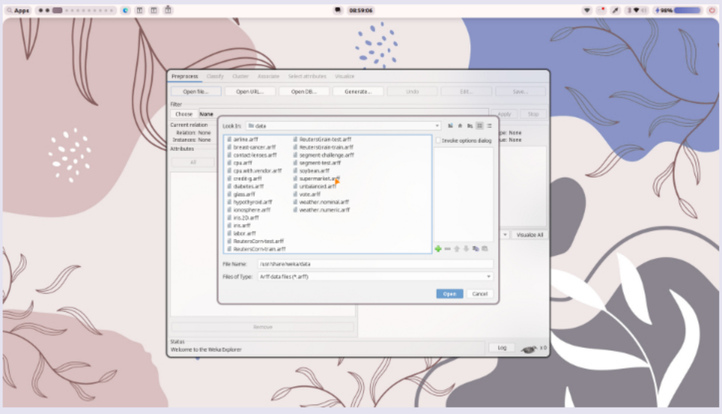
Name - Yash Lakhtariya
Enrollment number - 21162101012
Branch - CBA Batch - 61
DMW Practical 6

Apriori testing using Weka tool :

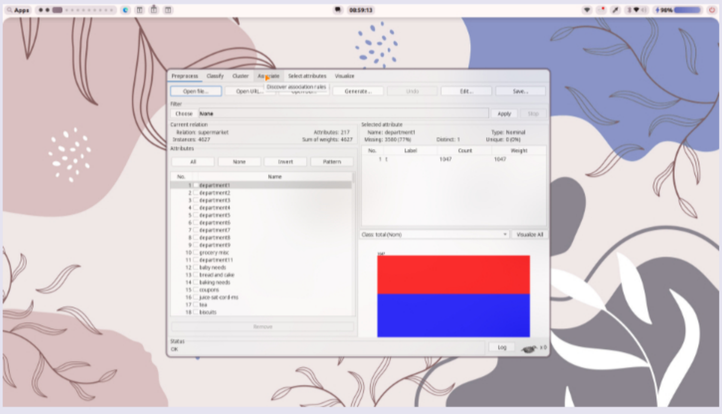
1. Open Explorer in Weka tool



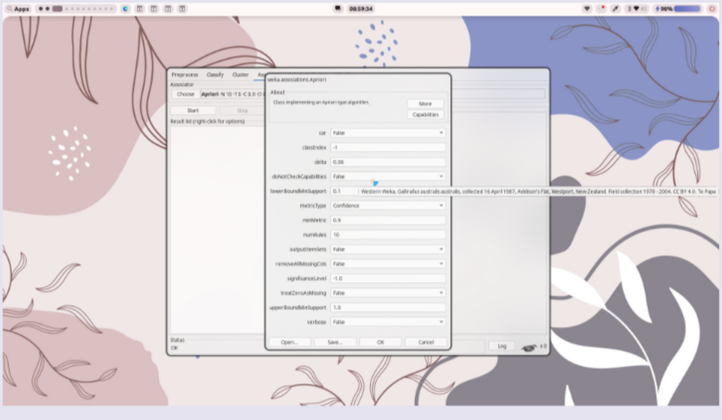
2. Open any file say supermarket.arff here from given data in Weka folder, here for linux system, the path is */usr/share/weka/data/*



3. Now, visit Associate tab in explorer



4. Click Apriori option and change some options if needed



5. Final output is obtained by using Apriori algorithm containing min support, no. of cycles. sets of large items, best rules obtained