Association Rules

Example with the Online Retail dataset, from UCI

In [1]:
```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
```

Upload the file `'retail_france.csv'` .

Inspect its content. It is a transactional database where the role of transaction identifier is played by the column `InvoiceNo` and the items are in the column `Description` .

The database has some problems:

1. some descriptions represent the same item but have different leading or trailing spaces, therefore they must be made uniform with the Pandas' function `str.strip()`

In [2]:
```
url = 'retail_france.csv'
#-- to fill with reading and display
```

Out[2]:

	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
5	536365	22752	SET 7 BABUSHKA NESTING BOXES	2	2010-12-01 08:26:00	7.65	17850.0	United Kingdom
6	536365	21730	GLASS STAR FROSTED T-LIGHT HOLDER	6	2010-12-01 08:26:00	4.25	17850.0	United Kingdom
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
8	536366	22632	HAND WARMER RED POLKA DOT	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom
9	NaN	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom
10	536367	22745	POPPY'S PLAYHOUSE BEDROOM	6	2010-12-01 08:34:00	2.10	13047.0	United Kingdom
11	536367	22748	POPPY'S PLAYHOUSE KITCHEN	6	2010-12-01 08:34:00	2.10	13047.0	United Kingdom
12	536367	22749	FELTCRAFT PRINCESS CHARLOTTE DOLL	8	2010-12-01 08:34:00	3.75	13047.0	United Kingdom
13	536367	22310	IVORY KNITTED MUG COSY	6	2010-12-01 08:34:00	1.65	13047.0	United Kingdom
14	536367	84969	BOX OF 6 ASSORTED COLOUR TEASPOONS	6	2010-12-01 08:34:00	4.25	13047.0	United Kingdom
15	536367	22623	BOX OF VINTAGE JIGSAW BLOCKS	3	2010-12-01 08:34:00	4.95	13047.0	United Kingdom
16	536367	22622	BOX OF VINTAGE ALPHABET BLOCKS	2	2010-12-01 08:34:00	9.95	13047.0	United Kingdom
17	536367	21754	HOME BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	13047.0	United Kingdom
18	536367	21755	LOVE BUILDING BLOCK WORD	3	2010-12-01 08:34:00	5.95	13047.0	United Kingdom
19	536367	21777	RECIPE BOX WITH METAL HEART	4	2010-12-01 08:34:00	7.95	13047.0	United Kingdom

In [3]:
```
df0.head()
```

Out[3]:

	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

In [4]:
```

```

```
The number of unique Description values in the input file is 4224
```

In [5]:
```

```

In [6]:
```

```

```
After cleaning, the number of unique Description values in the input file is 4211
```

Some rows may not have an `InvoiceNo` and must be removed, because they cannot be used.

Check if there are such that rows and in case remove them. You can check with the Pandas' function `isna` and remove with `dropna` on `axis=0`, with the option `subset`

In [7]:
```

```

```
Rows with missing InvoiceNo before removing
```

Out[7]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
9	NaN	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom

In [8]:

```

```

In [9]:

```

```

```
Rows with missing InvoiceNo after removing
```

Out[9]:

InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country

Some InvoiceNo start with a `C`. They are "credit transactions" and must be removed.

Check the number of rows containing `C` in `InvoiceNo` and remove them. At the moment the column `InvoiceNo` is a generic object, in order to be able to use string functions, such as `contains`, it must be transformed into `str` with `astype`.

In [10]:

```

```

```
There are 9288 rows containing 'C' in 'InvoiceNo'
```

In [11]:

```

```

In [12]:

```

```

```
After removal, there are 0 rows containing 'C' in 'InvoiceNo'
```

Several transactions include the item `'POSTAGE'`, which represents the mailing expenses. In this analysis we are not interested in it, therefore the rows with `'POSTAGE'` will be removed.

In [13]:

```

```

```
There are 1961 rows containing POSTAGE in Description
```

In [14]:

In [15]:
```
df.describe
```

Out[15]:
```
<bound method NDFrame.describe of        InvoiceNo StockCode                          Description  Quantity  \
0         536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1         536365     71053                  WHITE METAL LANTERN         6
2         536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3         536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4         536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6
...          ...       ...                                  ...       ...
541904    581587     22613          PACK OF 20 SPACEBOY NAPKINS        12
541905    581587     22899          CHILDREN'S APRON DOLLY GIRL         6
541906    581587     23254         CHILDRENS CUTLERY DOLLY GIRL         4
541907    581587     23255      CHILDRENS CUTLERY CIRCUS PARADE         4
541908    581587     22138         BAKING SET 9 PIECE RETROSPOT         3

               InvoiceDate  UnitPrice  CustomerID         Country
0      2010-12-01 08:26:00       2.55     17850.0  United Kingdom
1      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
2      2010-12-01 08:26:00       2.75     17850.0  United Kingdom
3      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
4      2010-12-01 08:26:00       3.39     17850.0  United Kingdom
...                    ...        ...         ...             ...
541904 2011-12-09 12:50:00       0.85     12680.0          France
541905 2011-12-09 12:50:00       2.10     12680.0          France
541906 2011-12-09 12:50:00       4.15     12680.0          France
541907 2011-12-09 12:50:00       4.15     12680.0          France
541908 2011-12-09 12:50:00       4.95     12680.0          France

[530786 rows x 8 columns]>
```

After the cleanup, we need to consolidate the items into 1 transaction per row with each product 1 hot encoded. For the sake of keeping the data set small, we are only looking at sales for France. However, in additional code below, we will compare these results to sales from Germany. Further country comparisons would be interesting to investigate.

Actions:

1. filter the rows `` `Country` ='France'` ``
2. group by `['InvoiceNo', 'Description']` computing a sum on `['Quantity']`
3. use the `unstack` function to move the items from rows to columns
4. reset the index
5. fill the missing with zero (`fillna(0)`)
6. store the result in the new dataframe `basket` and inspect it

In [16]:

Out[16]:

Description	10 COLOUR SPACEBOY PEN	12 COLOURED PARTY BALLOONS	12 EGG HOUSE PAINTED WOOD	12 MESSAGE CARDS WITH ENVELOPES	12 PENCIL SMALL TUBE WOODLAND	12 PENCILS SMALL TUBE RED RETROSPOT	12 PENCILS SMALL TUBE SKULL	12 PENCILS TALL TUBE POSY	12 PENCILS TALL TUBE RED RETROSPOT	12 PENCILS TALL TUBE WOODLAND	...	V
InvoiceNo												
536370	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
536852	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
536974	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
537065	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	
537463	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	

5 rows × 1562 columns

There are a lot of zeros in the data but we also need to make sure any positive values are converted to a 1 and anything less than 0 is set to 0.

You can define a function `encode_units` which takes a number and returns 0 if the number is 0 or less, 1 if the number is 1 or more. The function can be applied to `basket` with the Pandas' function `applymap`, the result is stored in the variable `basket_sets`

This step will complete the one hot encoding of the data.

In [17]:

Now that the data is structured properly, we can generate frequent item sets that have a support of at least 7% (this number was chosen so that we can get enough useful examples):

- generate the `frequent_itemsets` with `apriori`, setting `min_support=0.07` and `use_colnames=True`
- generate the `rules` with `association_rules` using `metric="lift"` and `min_threshold=1`
- show the rules

In [18]:

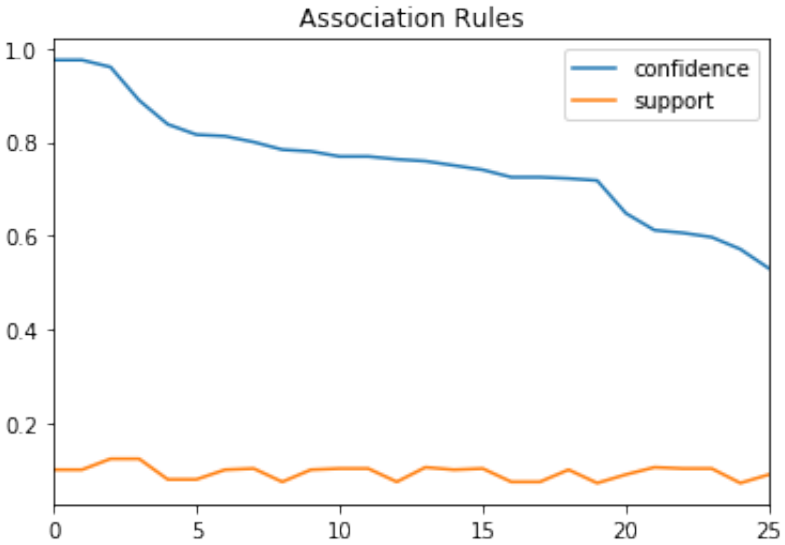
Out[18]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN)	0.103359	0.098191	0.074935	0.725000	7.383553	0.064786	3.279305
1	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.098191	0.103359	0.074935	0.763158	7.383553	0.064786	3.785817
2	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.095607	0.098191	0.080103	0.837838	8.532717	0.070716	5.561154
3	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.098191	0.095607	0.080103	0.815789	8.532717	0.070716	4.909561
4	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE PINK)	0.095607	0.103359	0.074935	0.783784	7.583108	0.065054	4.146964

In order to plot the rules, it is better to sort them according to some metrics. We will sort on descending confidence and support and plot `'confidence'` and `'support'`.

In [19]:

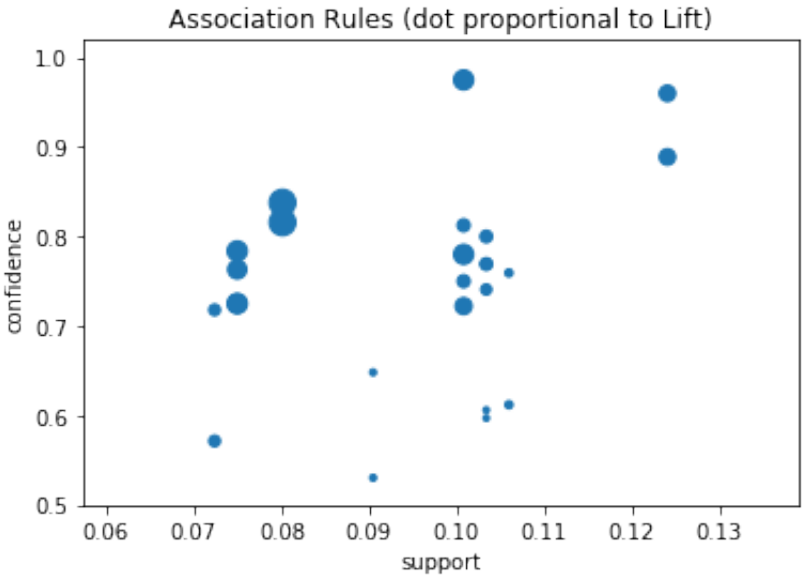
Out[19]: `<matplotlib.axes._subplots.AxesSubplot at 0x13aa9e290>`



You find below a three dimensional plot, where the dot size is proportional to the lift, obtained using `plot.scatter`.

In [20]:

Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x12070f7d0>



Finally, we draw a plot of a subset of the rules using the function `draw_graph` , provided in this package.

In [23]:
```
from draw_rules_graph import draw_graph
help(draw_graph)
```

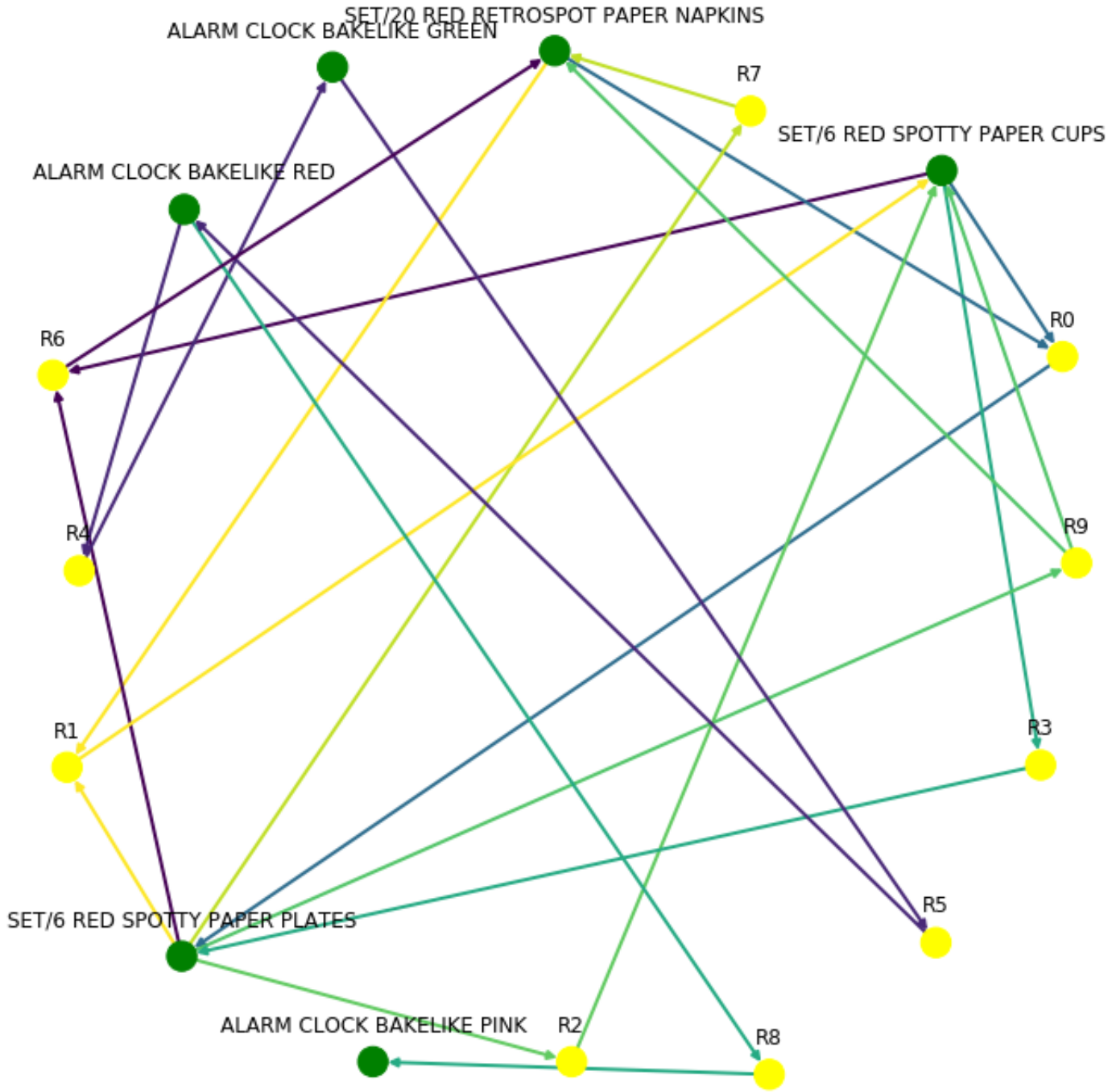
```
Help on function draw_graph in module draw_rules_graph:

draw_graph(rules, rules_to_show=5)
    draws the rules as a graph linking antecedents and consequents
    "rule nodes" are yellow, with name "R<n>", "item nodes" are green
    arrows colors are different for each rule, and go from the antecedent(s)
    to the rule node and to the consequent(s)
    the "rules_to_show" parameter limits the rules to show to the initial
    part of the "rules" dataframe
```

In [24]:
```

```

```
<Figure size 432x288 with 0 Axes>
```


SET/20 RED RETROSPOT PAPER NAPKINS
ALARM CLOCK BAKELIKE GREEN
R7
SET/6 RED SPOTTY PAPER CUPS
ALARM CLOCK BAKELIKE RED
R6
R0
R4
R9
R1
R3
R5
SET/6 RED SPOTTY PAPER PLATES
ALARM CLOCK BAKELIKE PINK
R2
R8

In [ ]: