In [1]:

Market Basket Analysis

Good news! If you keep record of the business' transactions you can take advantage of **Market Basket Analysis (MBA)**!

But what is Market Basket Analysis?

Market Basket Analysis is a powerful tool for translating vast amounts of customer transactions and viewing data into simple rules for product promotion and recommendation. It allows us, for instances, identifying products that are frequently bought together and from there building recommendations based on this information (e.g., bundles to offer, which items to present close to each other, how to improve inventory management, which items to upsell).

Market Basket Analysis is also a tool that helps people saving time and having fun. Yes! You read it right. This tool can also be used to build recommendation engines such as the ones used by Netflix and Spotify.

```
import pandas as pd
import numpy as np

# pd.set_option('display.max_rows', None)

#Import permutations from the itertools module
from itertools import permutations

#visualization packages
import matplotlib.pyplot as plt
import seaborn as sns
from pandas.plotting import parallel_coordinates

# MBA packages

# Import the transaction encoder function from mlxtend
from mlxtend.preprocessing import TransactionEncoder
# Import Apriori algorithm
from mlxtend.frequent_patterns import apriori
# Import the association rule function from mlxtend
from mlxtend.frequent_patterns import association_rules
```

In this notebook we start a series of tutorials introducing techniques that will help you making better data-driven marketing decisions.

In this first Marketing Analytics tutorial, we introduce you to **Market Basket Analysis**.

This tutorial is based on Python. In good part of it we make use of package mlxtend which is very useful for performing important tasks for Market Basket Analysis such as:

1. Pre-process data
2. Generate item sets and rules
3. Filter according to metrics

After completing this tutorial, you'll know:

- What Market Basket Analysis is

- How to prepare your data to apply it
- The metrics used in MBA
- Perform MBA using the Apriori algorithm
- Some simple visualizations used in MBA

Association Rules

MBA is based on the so called association rules. Association rules show us items that are associated with each other. For example, if we find out that buying croissant is associated with buying jam, then we state it as the following association rule:
{���������}→{���}

and we read it as "If croissant then jam".

In general, we have

{����������}→{����������}
A simple way to start Market Basket Analysis is by generating and analyzing simple association rules with one antecedent and one consequent. Further, we can consider both multiple antecedents and/or consequents. For instances, we can generate more complicated rules such as:

{���������,������}→{���}{����}→{���������,�h����}
Even if we use only simple rules (2 items) the number of association rules can be enormous. In addition, not all generated rules are useful. Bellow you can have an idea of the how the number of simple association rules grows with the number of items considered.

In [2]:

```
number_of_items = np.arange(1, 1000)
number_of_simple_rules = number_of_items*(number_of_items-1)

df_number_simple_association_rules = pd.DataFrame({'Number of
Items':number_of_items,'Number of Simple Rules':number_of_simple_rules})
df_number_simple_association_rules.head(10)
```

Out[2]:

	Number of Items	Number of Simple Rules
0	1	0
1	2	2
2	3	6
3	4	12
4	5	20

	Number of Items	Number of Simple Rules
5	6	30
6	7	42
7	8	56
8	9	72
9	10	90

In [3]:

```
plt.figure(figsize=(15,5))
plt.title("Number of Simple Association Rules per Number of Items")
sns.lineplot(data = df_number_simple_association_rules, x='Number of
Items', y='Number of Simple Rules')
```

Out[3]:

```
<AxesSubplot:title={'center':'Number of Simple Association Rules per Number
of Items'}, xlabel='Number of Items', ylabel='Number of Simple Rules'>
```

Therefore, an import issue considered by MBA is how to reduce an enormous set of potential association rules by selecting only those which are useful for a specific business application.

Later in this tutorial, we learn about the Apriori algorithm, a very important tool in MBA, especially when we are dealing with many items. But before that we need to understand some metrics used in MBA. These metrics are the ones that help selecting the useful association rules.

While association rules tell us that two or more items are related, the metrics presented here allow us quantifying the usefulness of those relationships.

So, we start this tutorial presenting some useful metrics in MBA by building recommendations for a small bakery (smaller dataset). Later we present the Apriori algorithm building recommendations for products of an online retailer (larger dataset).

Datasets

In this tutorial we use two datasets.

In order to understand each metric and its use we will make use of a fictional, artificially generated dataset, i.e., Our little bakery transaction dataset. This dataset consists of 298 transactions.

After that we use a bigger dataset: A transactional dataset which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. This one is available at the UCI Machine Learning repository. In particular, we used a part of the dataset; only transactions of customers in The Netherlands.

□ Our Little Bakery at the Train Station □

Imagine we have a little bakery in a local train station and we would like to offer an interesting bundle to our rushed morning customers. To find out the most interesting options using MBA, we have collected transactions containing 7 products.

The basic steps we need to apply for our MBA are:

- **Step 1**. Prepare data
- **Step 2**. Generate association rules
- **Step 3**. Use some metric to choose the most interesting rule(s) for the business case.

In [4]:

```
# importing transaction data

df_bakery =
pd.read_csv('../data/raw/bakery_transaction_list_without_filter_4_B.csv')
df_bakery.head()
```

Out[4]:

	TransactionId	Transaction
0	0	croissant,coffee
1	1	croissant,brownie
2	2	brownie,croissant
3	3	sausage bread,coffee
4	4	orange juice,croissant

In [5]:

```
df_bakery.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 298 entries, 0 to 297
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   TransactionId  298 non-null    int64 
 1   Transaction    298 non-null    object
dtypes: int64(1), object(1)
memory usage: 4.8+ KB
```

Our data is already presented in a pretty good way. We have 298 transactions, each one with unique products from our set of 7 products: sausage bread,croissant,pain au chocolat,orange juice,coffee, cookie, and brownie.

In [6]:

```
# Obtain unique items in transactions dataset
```

```
transactions = df_bakery['Transaction'].apply(lambda t: t.split(','))
transactions = list(transactions)
items_list = [item for transaction in transactions for item in transaction]
items_list = list(set(items_list))
items_list
```

Out[6]:

```
['croissant',
 'orange juice',
 'brownie',
 'coffee',
 'pain au chocolat',
 'cookie',
 'sausage bread']
```

The Apriori Algorithm

The Apriori algorithm helps reducing the complexity of the MBA problem by eliminating low support item sets before generating the association rules.

But how can we remove an itemset without knowing that we are not eliminating interesting association rules? Previously, in our bakery example we performed pruning when we've only considered item sets which contained coffee or orange juice. As a result, we didn't come out with viable bundles because, without knowing, we were eliminating item sets that were indeed the best ones for promotion.

The Apriori algorithm offers an alternative approach that does not require enumerating of all item sets. It is based on the Apriori principle that states : subsets of frequent sets must also be frequent. By applying this principle, the algorithm retains frequent sets, i.e., item sets that exceed some minimal level of support, and prune those that cannot be said to be frequent.

For example, if cookie is considered infrequent item because it falls bellow the minimum support then the itemset {cookie, coffee} as well as the itemset {cookie, coffee, croissant} are eliminated. In this way, computing support just once for cookie make possible to eliminate many other rules without having to enumerate them.

To apply the Apriori Algorithm we make use of the Mlxtend (machine learning extensions) Python package.

The steps performed to apply the Apriori algorithm consist of:

- **Step 1:** Pre-process the transaction dataset in order to obtain a one-hot encoded dataframe (See section about metrics)

- **Step 2:** Apply the Apriori algorithm to generate frequent item sets setting minimum support (min_support) and number of items (max_len) for the pruning process.

  Notice that the version of the Apriori algorithm used by apriori from mlxtend.frequent_patterns allows you also pruning by the number of items in the item sets.

- **Step 3:** Generate association rules using the frequent item sets obtained in Step 1 and defining metric and its minimum threshold.

  association_rules from mlxtend.frequent_patterns supports all metrics that were presented previously with exception of the Zhang's metric. So, you can perform pruning using the metric that seems more appropriate for your user case.

Let's apply the steps above and use the Apriori algorithm to our dataset considering three use cases:

1. **Bundle of products:** Which products could we offer together?
2. **Cross-promotion:** Which product should we use to promote another product?
3. **Cross-promotion to sell a target product:** If we want to promote a specific product which product(s) should we use to promote it?

In [36]:

```
df_NL = pd.read_csv(r"C:\MKB_datalab\Tutorials\marketing-analysis-market-basket-analysis\data\raw\retailer_nl.csv")
df_NL.head()
```

Out[36]:

	TransactionId	Transaction
0	536403	hand warmer bird design,postage
1	539491	pack of 12 woodland tissues,pack of 12 pink po...
2	539731	pack of 72 retrospot cake cases,easter tin kee...
3	541206	pack of 12 pink polkadot tissues,rose cottage ...
4	541570	strawberry lunch box with cutlery,dinosaur lun...

In [37]:

```
# One-hot encoded dataframe
onehot_NL = onehot_encode_transactions(df_NL["Transaction"])

onehot_NL
```

Out[37]:

In [38]:

```
# Apply the Apriori algorithm with a support value of 0.005
frequent_itemsets = apriori(onehot_NL,
                            min_support = 0.005,
                            use_colnames = True,
                            max_len = 2)

frequent_itemsets['length'] = frequent_itemsets['itemsets'].apply(lambda x
: len(list(x)))

frequent_itemsets.head()
```

Out[38]:

	support	itemsets	length
0	0.049505	()	1

	support	itemsets	length
1	0.039604	(1 hanger)	1
2	0.039604	(birthday card)	1
3	0.009901	(pink spots)	1
4	0.019802	(retro spot)	1

In [39]:

```
len(frequent_itemsets)
```

Out[39]:

```
52676
```

But how we choose the minimum support value?

We can try different values, check the number of frequent item sets and pick the value of support that provides us a convenient number of item sets. Is 52676 too much? Then pick a higher min_support. Tweaking the number of items is also a way to reduce, allowing less items will produce less frequent item sets.

A good way to have an idea on the range of values you can use for support is to plot a scatterplot using support x confidence. This because Bayardo and Agrawal showed in their [1999 paper] that the best-performing rules along a variety of common metrics, including the ones we explored here, must be located on the confidence-support border.

The scatterplot below includes also a third dimension, i.e., another metric: lift. Observe that the values support goes until around 0.2, and most of the point are below support 0.125

In [40]:

```
# Generate association rules without performing additional pruning
rules = association_rules(frequent_itemsets,
                          metric = 'support',
                          min_threshold = 0.00)

# Generate scatterplot using support and confidence
plt.figure(figsize=(8,6))
plt.title("Scaterplot support x confidence x lift")
sns.scatterplot(x = "support",
                y = "confidence",
                size = "lift",
                data = rules)
plt.show()
```

Let's reduce the number of items by choosing a larger minimum support value and generate the association rules for those frequent item sets.

In [41]:

```
# Compute frequent itemsets using a minimum support of 0.1
frequent_itemsets_1 = apriori(onehot_NL,
                              min_support = 0.1,
```

```
                                max_len = 2,
                                use_colnames = True )

frequent_itemsets_1['length'] =
frequent_itemsets_1['itemsets'].apply(lambda x : len(list(x)))

frequent_itemsets_1.head()
```

Out[41]:

	support	itemsets	length
0	0.118812	(card dolly girl)	1
1	0.108911	(charlotte bag pink polkadot)	1
2	0.108911	(charlotte bag suki design)	1
3	0.128713	(childrens apron spaceboy design)	1
4	0.108911	(circus parade lunch box)	1

In [42]:

```
len(frequent_itemsets_1)
```

Out[42]:

```
36
```

First, let's generate association rules without pruning, i.e., min_threshold = 0.0.

In [43]:

```
# Compute the association rules for frequent_itemsets_1 without pruning
rules_1 = association_rules(frequent_itemsets_1,
                              metric = 'support',
                              min_threshold = 0.0)
rules_1
```

Out[43]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(dolly girl lunch box)	(round snack boxes set of 4 fruits)	0.217822	0.168317	0.108911	0.500000	2.970588	0.072248	1.663366

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
1	(round snack boxes set of 4 fruits)	(dolly girl lunch box)	0.168317	0.217822	0.108911	0.647059	2.970588	0.072248	2.216172
2	(dolly girl lunch box)	(round snack boxes set of4 woodland)	0.217822	0.247525	0.158416	0.727273	2.938182	0.104500	2.759076
3	(round snack boxes set of4 woodland)	(dolly girl lunch box)	0.247525	0.217822	0.158416	0.640000	2.938182	0.104500	2.172717
4	(spaceboy lunch box)	(dolly girl lunch box)	0.277228	0.217822	0.207921	0.750000	3.443182	0.147535	3.128713
5	(dolly girl lunch box)	(spaceboy lunch box)	0.217822	0.277228	0.207921	0.954545	3.443182	0.147535	15.900990
6	(round snack boxes set of4 woodland)	(plasters in tin spaceboy)	0.247525	0.118812	0.118812	0.480000	4.040000	0.089403	1.694593
7	(plasters in tin	(round snack	0.1188	0.24752	0.118	1.0000	4.040	0.089	inf

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
	spaceboy)	boxes set of4 woodland)	12	5	812	00	000	403	
8	(round snack boxes set of4 woodland)	(red retrospot charlotte bag)	0.247525	0.118812	0.108911	0.440000	3.703333	0.079502	1.573550
9	(red retrospot charlotte bag)	(round snack boxes set of4 woodland)	0.118812	0.247525	0.108911	0.916667	3.703333	0.079502	9.029703
10	(red toadstool led night light)	(round snack boxes set of4 woodland)	0.138614	0.247525	0.118812	0.857143	3.462857	0.084502	5.267327
11	(round snack boxes set of4 woodland)	(red toadstool led night light)	0.247525	0.138614	0.118812	0.480000	3.462857	0.084502	1.656512
12	(round snack boxes set of4 woodland)	(round snack boxes set of 4 fruits)	0.247525	0.168317	0.148515	0.600000	3.564706	0.106852	2.079208

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
13	(round snack boxes set of 4 fruits)	(round snack boxes set of4 woodland)	0.168317	0.247525	0.148515	0.882353	3.564706	0.106852	6.396040
14	(spaceboy lunch box)	(round snack boxes set of 4 fruits)	0.277228	0.168317	0.118812	0.428571	2.546218	0.072150	1.455446
15	(round snack boxes set of 4 fruits)	(spaceboy lunch box)	0.168317	0.277228	0.118812	0.705882	2.546218	0.072150	2.457426
16	(spaceboy birthday card)	(round snack boxes set of4 woodland)	0.168317	0.247525	0.128713	0.764706	3.089412	0.087050	3.198020
17	(round snack boxes set of4 woodland)	(spaceboy birthday card)	0.247525	0.168317	0.128713	0.520000	3.089412	0.087050	1.732673
18	(spaceboy lunch box)	(round snack boxes set of4 woodland)	0.277228	0.247525	0.178218	0.642857	2.597143	0.109597	2.106931

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
19	(round snack boxes set of4 woodland)	(spaceboy lunch box)	0.247525	0.277228	0.178218	0.720000	2.597143	0.109597	2.581330
20	(round snack boxes set of4 woodland)	(woodland charlotte bag)	0.247525	0.158416	0.118812	0.480000	3.030000	0.079600	1.618431
21	(woodland charlotte bag)	(round snack boxes set of4 woodland)	0.158416	0.247525	0.118812	0.750000	3.030000	0.079600	3.009901
22	(spaceboy birthday card)	(spaceboy lunch box)	0.168317	0.277228	0.128713	0.764706	2.758403	0.082051	3.071782
23	(spaceboy lunch box)	(spaceboy birthday card)	0.277228	0.168317	0.128713	0.464286	2.758403	0.082051	1.552475
24	(spaceboy lunch box)	(woodland charlotte bag)	0.277228	0.158416	0.118812	0.428571	2.705357	0.074895	1.472772
25	(woodland charlotte	(spaceboy lunch	0.158416	0.277228	0.118812	0.750000	2.705357	0.074895	2.891089

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
bag)	box)							

Now that we have our association rules let's consider some possible use cases to illustrate the use of MBA and what we've learnt so far.

Use Case 1 - Bundle

Let's say we would like to find two items to be sold together.

We can start by inspecting the table above for high support values. But even better, we can use a heatmap to spot faster which values are interesting for us.

A heatmap as below is useful to help us during the pruning process because it helps us visualizing the intensity of the relationships between pairs of objects, i.e., the value of a metric of association rules between antecedent and consequent.

In [44]:

```
# Replace frozen sets with strings
rules_1['antecedents'] = rules_1['antecedents'].apply(lambda x:
','.join(list(x)))
rules_1['consequents'] = rules_1['consequents'].apply(lambda x:
','.join(list(x)))
```

In [45]:

```
# Transform data to matrix format and generate heatmap
pivot = rules_1.pivot(index='consequents', columns='antecedents',
values='support')
# sns.heatmap(pivot)

# Generate a heatmap with annotations on and the colorbar off
plt.figure(figsize=(22,12))
plt.title("Heatmap showing support values for one item sets {antecendent} -
> {consequent}\n", size = 20)
sns.heatmap(pivot, annot = True, cbar = True, cmap='GnBu')

# Format and display plot
plt.yticks(rotation=0, size = 16)
plt.xticks(rotation=60, size = 16)
plt.show()
```

The heatmap above shows values of support for the association rules. Lighter colors mean lower intensity, darker color higher intensity, and white means that no association rule was identified for that itemset. We can easily see high support for the association rule {����� ���� ����*h* ���}→{�������� ����*h* ���}, and vice-versa.

In [46]:

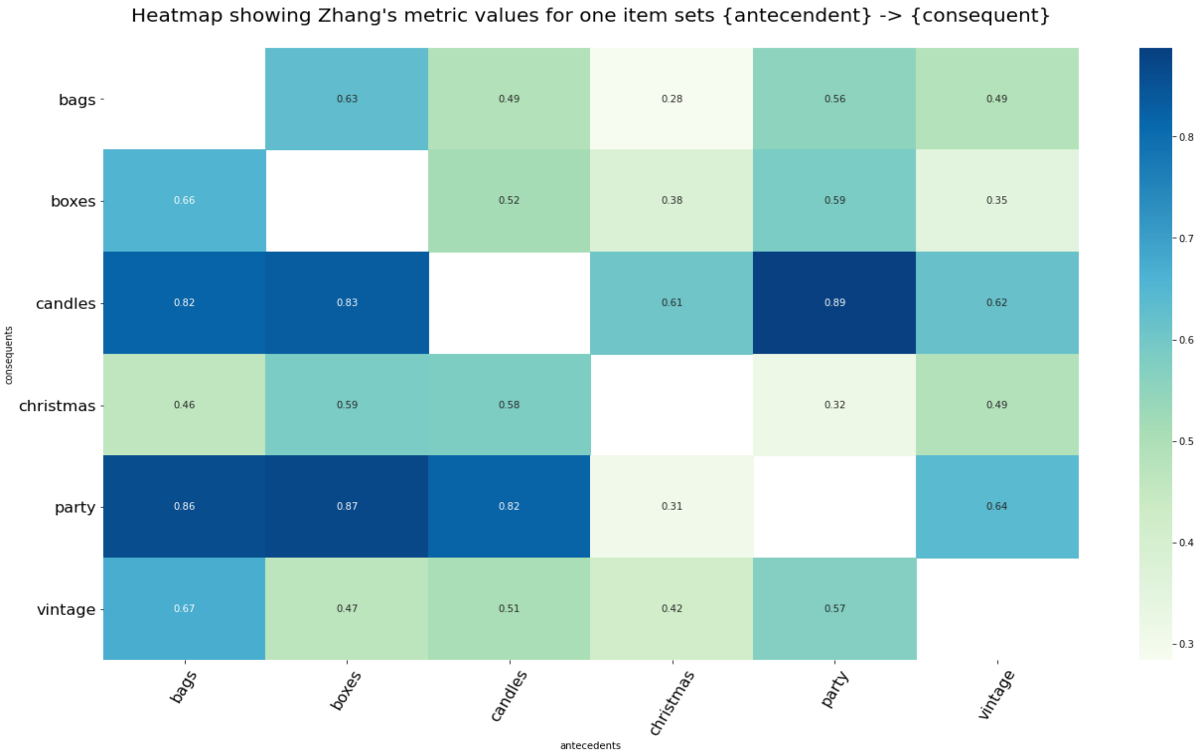
```
# Apply a 0.2 antecedent support threshold, 0.60 confidence threshold, and
2.50 lift threshold
```

```
filtered_rules_1 = rules_1[(rules_1['support'] > 0.2) &
                          (rules_1['confidence'] > 0.6) &
                          (rules_1['lift'] > 2.50)]
```

In [47]:

```
filtered_rules_1
```

Out[47]:



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
4	spaceboy lunch box	dolly girl lunch box	0.277228	0.217822	0.207921	0.750000	3.443182	0.147535	3.128713
5	dolly girl lunch box	spaceboy lunch box	0.217822	0.277228	0.207921	0.954545	3.443182	0.147535	15.900990

After filtering with multiple metrics, we confirm that selling these products together is a good option. We have high lift that tells us that this bundle is viable because it does not happen by chance.

Use Case 2: Cross-promotion

What if we decide we want to use one of these products to promote the other?

From the filtered rules above we can see that the best choice is to use dolly girl lunch box to promote spaceboy lunch box. Confidence tells us that we can be 95% confident that someone that buys dolly girl lunch box will buy spaceboy lunch box.

The heatmap using confidence values confirms it.

In [48]:

```
# Transform data to matrix format and generate heatmap
pivot = rules_1.pivot(index='consequents', columns='antecedents',
values='confidence')
# sns.heatmap(pivot)

# Generate a heatmap with annotations on and the colorbar off
plt.figure(figsize=(22,12))
plt.title("Heatmap showing confidence values for one item sets
{antecendent} -> {consequent}\n", size = 20)
sns.heatmap(pivot, annot = True, cbar = True, cmap='GnBu',linewidths=.5)

# Format and display plot
plt.yticks(rotation=0, size = 16)
plt.xticks(rotation=60, size = 16)
plt.show()
```

Conclusions

- Market Basket Analysis (MBA) is a powerful marketing tool that helps getting insights for product promotion and recommendations.
- A simple transactional dataset consisting of transaction ids (e.g., invoice numbers) and items of these transactions is enough to start your MBA.
- MBA helps identifying items frequently bought together and from there building recommendations based on this information (e.g., which bundles to offer, which items to present close to each other, how to improve inventory management, which items to upsell).
- MBA is based on the so called association rules. Association rules show us items that are associated with each other. The number of association rules grows very fast with the number of items.
- The Apriori algorithm is a fundamental tool to simplify MBA without eliminating useful association rules.
- A good knowledge of the metrics presented here is important so you can keep the association rules that are useful for your business application.
- Simple visualizations like scatterplots, heatmap, and parallel coordinates plots are great tools to guide the prunning process as well as to summarize final results.
- Scatterplots help visualizing the boundaries of values of metrics such support. This helps identifying correct pruning thresholds.
- Instead of inspecting the association rules table, we can make use of a heatmap and easily spot interesting association rules.