Warenkorbanalyse einer Bäckerei

Inhaltsverzeichnis

Datenbestand

Das Daten-Set liegt unter https://www.kaggle.com/xvivancos/market-basket-analysis/data (https://www.kaggle.com/xvivancos/market-basket-analysis/data)) und beschreibt die Tansaktionen einer Bäckerei namens "The Bread Basket" in der Altstadt von Edinburgh, Scotland. Diese Bäckerei bietet ein erfrischendes Angebot an argentinischen und spanischen Produkten.

Eigenschaften

Im Daten-Set gibt es über 9.000 Transaktionen aus der Bäckerei und es hat folgende Felder:

```
- Date:
    Kategoriale Variable beschreibt das Datum der Transaktionen im Format JJJJ-M
M-TT. Die Spalte enthält Daten vom 30.10.2016 bis zum 09.04.2017.
- Time:
    Kategoriale Variable beschreibt die Zeit der Transaktionen im Format HH:MM:S
S.
- Transactions:
    Quantitative Variable, mit der die Transaktionen unterscheiden können. Die Z
eilen, die in diesem Feld den gleichen Wert haben, gehören zu derselben Transakt
ion.
- Item:
    Kategoriale Variable mit den Produkten.
```

Idee

Jedes, wirklich jedes, Unternehmen, das etwas verkauft, besitzt automatisch die Daten, egal ob im elektronischen Format oder auf dem Papier, für eine Warenkorbanalyse. Und die Bäckereien sind in diesem Sinne keine Ausnahme. Das Daten-Set der Bäckerei aus Edinburgh wird genau für die Warenkorbanalyse-Modellierungstechnik (engl. Market Basket Analysis) verwendet. Die Idee dieser Technik ist ein Einkaufsmuster innerhalb von Artikeln vorhersagen zu können, was sie im Handel populär macht.

Anders gesagt, basierend am Daten-Set der Bäckerei werden wir versuchen, die Verbindung zwischen gekauften Artikeln zu finden. Zum Beispiel, wenn jemand 'Produkt_1' kauft, wie wahrscheinlich dann ist, dass auch 'Produkt_2' gekauft wird? Mehr dazu ist in der Wikipedia (https://de.wikipedia.org/wiki/Apriori-Algorithmus) zu finden.

Theorie

Die Warenkorbanalyse besteht aus einer Reihe von Assoziationsverfahren (die Suche nach starken Regeln). Eins davon ist der **Apriori-Algorithmus** (https://de.wikipedia.org/wiki/Apriori-Algorithmus) und seine zentralen Kennzahlen sind: Support, Konfidenz und Lift.

- Support

Der Support beantwortet die Frage, wie oft ein Produkt prozentual überhaupt gekauft wird:

Support(Produkt_1)(%) = [(Transaktonen mit dem Produkt_1) / (Gesamte Anzahl der Transaktonen)] * 100%

Support(Produkt_1 und Produkt_2) = [(Transakt. mit Produkt_1 und Produkt_2) / (Gesamte Anzahl der Transakt.)] * 100%

- Konfidenz

Die Konfidenz(Produkt_1 -> Produkt_2) sagt, wie oft das Produkt_2 gekauft wird, wenn das Produkt_1 gekauft wird.

Konfidenz(Produkt_1 -> Produkt_2)(%) = (Transakt. mit Produkt_1 und Produkt_2) / (Transakt. mit dem Produkt_1) * 100%
oder
Konfidenz(Produkt_1 -> Produkt_2)(%) = Support(Produkt_1 und Produkt_2) / Support(Produkt_1)

- Lift

Der Lift gibt die Antwort auf die Frage, um wie viel wahrscheinlicher mach das Produkt_1 den Kauf des Produkts_2

Lift(Produkt_1 -> Produkt_2)(%) = Konfidenz(Produkt_1 -> Produkt_2) / Support(Produkt_2)
oder
Lift(Produkt_1 -> Produkt_2)(%) = Support(Produkt_1 und Produkt_2) / (Support(Produkt_1) * Support(Produkt_2))

Ein Lift von 1 oder 100% bedeutet, dass es keine Verbindung zwischen dem Produkt_1 und dem Produkt_2 besteht. Ein Lift von mehr als 1 (mehr als 100%) bedeutet, dass Produkt_1 und Produkt_2 häufiger zusammen gekauft werden. Ein Lift von weniger als 1 (weniger als 100%) zeigt, dass es unwahrscheinlich ist, dass zwei Produkte zusammen gekauft werden.

Untersuchen der Daten

Zuerst werden die Python's Bibliotheken importiert, die wir für die Analyse der Daten verwenden werden.

In [37]:

```
# Import der Data Science Pakete (pandas etc.)
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Warnings ignorieren
import warnings
warnings.filterwarnings('ignore')

# Style festlegen
sns.set(style='darkgrid')
plt.rcParams["patch.force_edgecolor"] = True

# Verzeichnis auflisten das Daten-Set 'BreadBasket_DMS.csv' sollte im Verzeicnis liegen

import os
print(os.listdir("./"))
```

```
['.ipynb_checkpoints', 'baeckerei_warenkorbanalyse.ipynb', 'BreadBasket_DM
S.csv']
```

Lesen wir das Datenbestandes aus der 'BreadBasket_DMS.csv'-Datei ins pandas's DataFrame-Objekt und auflisten die Daten mit den Eigenschaften.

In [5]:

```
# Lesen der 'BreadBasket_DMS.csv'-Datei ins pandas's DataFrame-Objekt
df = pd.read_csv('./BreadBasket_DMS.csv')
```

In [6]:

```
print('Data-Set Information: \n')
print(df.info())
```

```
Data-Set Information:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21293 entries, 0 to 21292
Data columns (total 4 columns):
Date           21293 non-null object
Time           21293 non-null object
Transaction    21293 non-null int64
Item           21293 non-null object
dtypes: int64(1), object(3)
memory usage: 665.5+ KB
None
```

In [10]:

```
print('Die ersten zehn Zeilen aus dem Datenbestand: \n')
print(df.head(10))
```

```
Die ersten zehn Zeilen aus dem Datenbestand:

         Date      Time  Transaction           Item
0  2016-10-30  09:58:11            1          Bread
1  2016-10-30  10:05:34            2   Scandinavian
2  2016-10-30  10:05:34            2   Scandinavian
3  2016-10-30  10:07:57            3  Hot chocolate
4  2016-10-30  10:07:57            3            Jam
5  2016-10-30  10:07:57            3        Cookies
6  2016-10-30  10:08:41            4         Muffin
7  2016-10-30  10:13:03            5         Coffee
8  2016-10-30  10:13:03            5         Pastry
9  2016-10-30  10:13:03            5          Bread
```

Da die Bäckerei aus Großbritannien ist und ein Angebot an argentinischen und spanischen Produkten hat, ist das ein wenig schwierig, die Benennung der Produkten auf eine "durchschnittliche" Bäckerei aus Deutschland zu portieren. Deshalb lassen wir die Produkte so benennen, wie sie im Daten-Set vorkommen.

In [11]:

```
print('Eindeutige verkaufte Produkte: ', df['Item'].nunique())
print( '\n', df['Item'].unique())
```

```
Eindeutige verkaufte Produkte:  95

 ['Bread' 'Scandinavian' 'Hot chocolate' 'Jam' 'Cookies' 'Muffin' 'Coffee'
 'Pastry' 'Medialuna' 'Tea' 'NONE' 'Tartine' 'Basket' 'Mineral water'
 'Farm House' 'Fudge' 'Juice' "Ella's Kitchen Pouches" 'Victorian Sponge'
 'Frittata' 'Hearty & Seasonal' 'Soup' 'Pick and Mix Bowls' 'Smoothies'
 'Cake' 'Mighty Protein' 'Chicken sand' 'Coke' 'My-5 Fruit Shoot'
 'Focaccia' 'Sandwich' 'Alfajores' 'Eggs' 'Brownie' 'Dulce de Leche'
 'Honey' 'The BART' 'Granola' 'Fairy Doors' 'Empanadas' 'Keeping It Local'
 'Art Tray' 'Bowl Nic Pitt' 'Bread Pudding' 'Adjustment' 'Truffles'
 'Chimichurri Oil' 'Bacon' 'Spread' 'Kids biscuit' 'Siblings'
 'Caramel bites' 'Jammie Dodgers' 'Tiffin' 'Olum & polenta' 'Polenta'
 'The Nomad' 'Hack the stack' 'Bakewell' 'Lemon and coconut' 'Toast'
 'Scone' 'Crepes' 'Vegan mincepie' 'Bare Popcorn' 'Muesli' 'Crisps'
 'Pintxos' 'Gingerbread syrup' 'Panatone' 'Brioche and salami'
 'Afternoon with the baker' 'Salad' 'Chicken Stew' 'Spanish Brunch'
 'Raspberry shortbread sandwich' 'Extra Salami or Feta' 'Duck egg'
 'Baguette' "Valentine's card" 'Tshirt' 'Vegan Feast' 'Postcard'
 'Nomad bag' 'Chocolates' 'Coffee granules ' 'Drinking chocolate spoons '
 'Christmas common' 'Argentina Night' 'Half slice Monster ' 'Gift voucher'
 'Cherry me Dried fruit' 'Mortimer' 'Raw bars' 'Tacos/Fajita']
```

Vorbereiten der Daten

Prüfen der Null Werte

Als erstes prüfen wir das Daten-Set auf die Null Werte und dann auf die Werte, die mit "NONE" (undefinierte Daten) gekennzeichnet sind.

In [12]:

```
# Gibt es die Null Werte und wenn ja, wie viele?
print(df.isnull().sum().sort_values(ascending=False))
```

```
Item           0
Transaction    0
Time           0
Date           0
dtype: int64
```

Erstaunlich. Es sieht so aus, als ob es keine fehlenden Daten gebe. Und was ist mit den "NONE" Werten.

In [13]:

```
# Auflisten der "NONE" Werte
print(df[df['Item']=='NONE'])
```

	Date	Time	Transaction	Item
26	2016-10-30	10:27:21	11	NONE
38	2016-10-30	10:34:36	15	NONE
39	2016-10-30	10:34:36	15	NONE
66	2016-10-30	11:05:30	29	NONE
80	2016-10-30	11:37:10	37	NONE
85	2016-10-30	11:55:51	40	NONE
126	2016-10-30	13:02:04	59	NONE
140	2016-10-30	13:37:25	65	NONE
149	2016-10-30	13:46:48	67	NONE
167	2016-10-30	14:32:26	75	NONE
183	2016-10-31	08:47:05	82	NONE
201	2016-10-31	09:22:48	91	NONE
226	2016-10-31	10:07:40	103	NONE
235	2016-10-31	10:21:29	105	NONE
272	2016-10-31	11:42:05	123	NONE
282	2016-10-31	11:55:00	128	NONE
398	2016-11-01	09:26:03	184	NONE
413	2016-11-01	10:56:08	192	NONE
419	2016-11-01	11:06:09	195	NONE
431	2016-11-01	11:22:36	201	NONE
547	2016-11-02	08:07:05	257	NONE
560	2016-11-02	09:05:25	266	NONE
577	2016-11-02	09:52:58	274	NONE
587	2016-11-02	10:15:48	279	NONE
628	2016-11-02	12:11:56	298	NONE
718	2016-11-03	08:15:21	346	NONE
726	2016-11-03	08:49:23	348	NONE
788	2016-11-03	11:51:52	380	NONE
808	2016-11-03	12:16:15	387	NONE
810	2016-11-03	12:23:47	388	NONE
...	...	...	...	...
20232	2017-04-01	13:36:26	9211	NONE
20285	2017-04-02	09:49:32	9232	NONE
20289	2017-04-02	09:51:23	9234	NONE
20316	2017-04-02	10:50:11	9245	NONE
20332	2017-04-02	12:00:43	9254	NONE
20352	2017-04-02	13:19:35	9261	NONE
20376	2017-04-02	15:01:07	9270	NONE
20391	2017-04-02	15:22:05	9274	NONE
20412	2017-04-03	10:09:47	9286	NONE
20429	2017-04-03	10:45:41	9293	NONE
20460	2017-04-03	13:24:13	9309	NONE
20526	2017-04-04	07:58:54	9339	NONE
20538	2017-04-04	09:04:01	9346	NONE
20573	2017-04-04	12:18:12	9366	NONE
20574	2017-04-04	12:18:56	9367	NONE
20577	2017-04-04	12:19:48	9368	NONE
20678	2017-04-05	11:02:01	9406	NONE
20686	2017-04-05	11:05:00	9407	NONE
20799	2017-04-06	09:28:32	9457	NONE
20917	2017-04-07	08:47:29	9506	NONE
20919	2017-04-07	08:57:19	9507	NONE
20964	2017-04-07	13:06:01	9529	NONE
21010	2017-04-07	17:30:07	9550	NONE
21077	2017-04-08	10:44:44	9579	NONE
21080	2017-04-08	10:48:43	9580	NONE
21108	2017-04-08	11:54:22	9590	NONE
21122	2017-04-08	12:58:25	9599	NONE
21254	2017-04-09	12:01:07	9666	NONE
21255	2017-04-09	12:04:13	9667	NONE

```
21266  2017-04-09  12:31:28         9672  NONE

[786 rows x 4 columns]
```

Von den "NONE" Werte gibt es ein Haufen. Das passiert, wenn etwas verkauft wird und noch keine Bezeichnung im Bestand hat. Der einfachte Weg ist diese Daten nicht verwenden. So werden sie im Daten-Set gelöscht.

In [14]:

```
df.drop(df[df['Item']=='NONE'].index, inplace=True)
```

In [15]:

```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20507 entries, 0 to 21292
Data columns (total 4 columns):
Date           20507 non-null object
Time           20507 non-null object
Transaction    20507 non-null int64
Item           20507 non-null object
dtypes: int64(1), object(3)
memory usage: 801.1+ KB
None
```

Formatieren der Daten

Das Datum und die Zeit sind nicht die numerischen Werte im Datenbestand. Zur besseren Visualisierung und Analyse der Daten wandeln wir sie in die numerischen Werte um.

In [16]:

```
# Year
df['Year'] = df['Date'].apply(lambda x: x.split("-")[0])
# Month
df['Month'] = df['Date'].apply(lambda x: x.split("-")[1])
# Day
df['Day'] = df['Date'].apply(lambda x: x.split("-")[2])
```

In [17]:

```
print(df.info())
print(df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20507 entries, 0 to 21292
Data columns (total 7 columns):
Date           20507 non-null object
Time           20507 non-null object
Transaction    20507 non-null int64
Item           20507 non-null object
Year           20507 non-null object
Month          20507 non-null object
Day            20507 non-null object
dtypes: int64(1), object(6)
memory usage: 1.3+ MB
None
         Date      Time  Transaction           Item  Year Month Day
0  2016-10-30  09:58:11            1          Bread  2016    10  30
1  2016-10-30  10:05:34            2   Scandinavian  2016    10  30
2  2016-10-30  10:05:34            2   Scandinavian  2016    10  30
3  2016-10-30  10:07:57            3  Hot chocolate  2016    10  30
4  2016-10-30  10:07:57            3            Jam  2016    10  30
```

Visualizing and Understanding the Data

Unser Datenbestand beinhaltet die Transaktionen vom 30.10.2016 bis zum 09.04.2017. Logischerweise sind folgene Fragen für die Analyse interessant: Welche Artikel kaufen Kunden am meisten? Welche Monate waren erfolgreicher? Lassen wird die Fragen visuelle darzustellen.

In [18]:

```
# Die ersten 15 meistverkauften Produkte
most_sold = df['Item'].value_counts().head(15)

print('Meistverkaufte Produkte: \n')
print(most_sold)
```

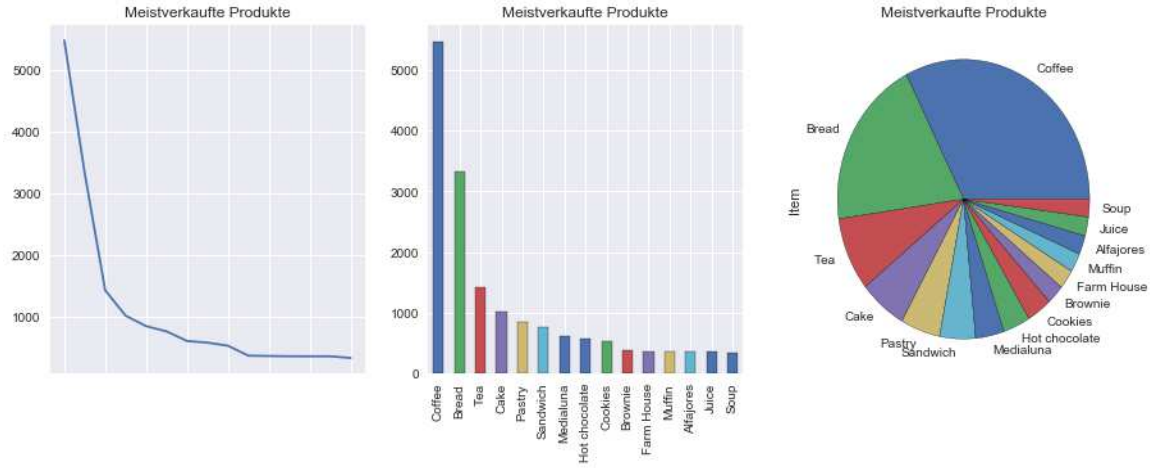
```
Meistverkaufte Produkte:

Coffee           5471
Bread            3325
Tea              1435
Cake             1025
Pastry            856
Sandwich          771
Medialuna         616
Hot chocolate     590
Cookies           540
Brownie           379
Farm House        374
Muffin            370
Alfajores         369
Juice             369
Soup              342
Name: Item, dtype: int64
```

In [19]:

```
plt.figure(figsize=(15,5))

# Meistverkaufte Produkte als Linie
plt.subplot(1,3,1)
most_sold.plot(kind='line')
plt.title('Meistverkaufte Produkte')

# Meistverkaufte Produkte als Balkendiagramm
plt.subplot(1,3,2)
most_sold.plot(kind='bar')
plt.title('Meistverkaufte Produkte')

# Meistverkaufte Produkte als Kreisdiagramm
plt.subplot(1,3,3)
most_sold.plot(kind='pie')
plt.title('Meistverkaufte Produkte')
```

Out[19]:

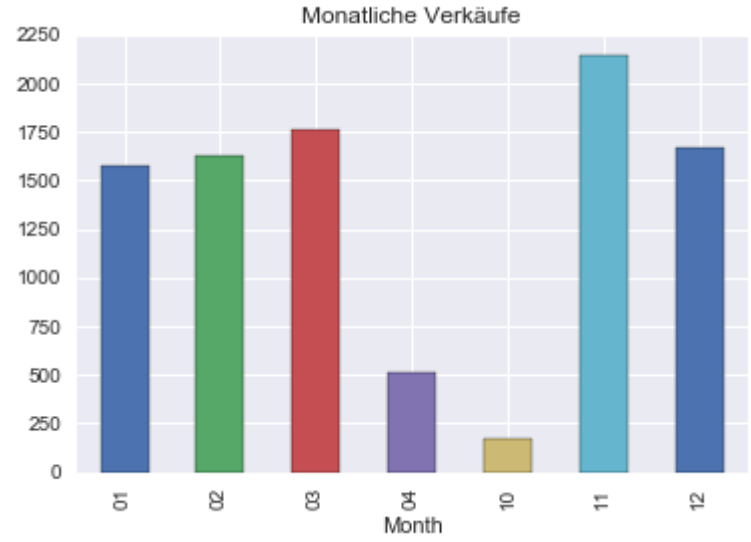
```
<matplotlib.text.Text at 0xa7b40f0>
```



Das meistverkaufte Produkt ist Kaffe gefolgt von Brot, Tee, Kuchen und Gebäck. Das ist völlig normal und verständlich für eine Bäckerei. Nachdem wir wissen, welche Artikel am beliebtesten sind, schauen wir uns an, in welchen Monaten die meisten Verkäufe erzielt werden.

In [20]:

```
df.groupby('Month')['Transaction'].nunique().plot(kind='bar', title='Monatliche Verkäufe')
plt.show()
```



Sehr interessant. Im Oktober und April gibt es die drastischen Umsatzunterschiede. Sind das die Ausreißermonate im Datenbestand? Wir prüfen die Anzahl der Transaktionen für diese Monate im Vergleich zu den anderen.

In [21]:

```
print(df.groupby('Month')['Day'].nunique())
```

```
Month
01    30
02    28
03    31
04     9
10     2
11    30
12    29
Name: Day, dtype: int64
```

Tatsächlich. Wie bereits aus der Beschreibung des Datenbestandes bekannt war, wurden nur 9 Tagen für April und 2 Tage für Oktober erfasst.

Warenkorbanalyse

Die Information zur Bibliothek liegt unter: https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/ (https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/apriori/)

In [22]:

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import association_rules, apriori
```

Wir erstellen erst die Liste der Transaktionen, damit wir sie später für den Apriori Algorithmus im TransactionEncoder formatieren können.

In [23]:

```
transaction_list = []

# FOR-Schleife zum Erstellen einer Liste der eindeutigen Transaktionen im Data-Set:
for i in df['Transaction'].unique():
    tlist = list(set(df[df['Transaction']==i]['Item']))
    if len(tlist)>0:
        transaction_list.append(tlist)
print(len(transaction_list))
```

```
9465
```

In [24]:

```
te = TransactionEncoder()
te_ary = te.fit(transaction_list).transform(transaction_list)
df_ary = pd.DataFrame(te_ary, columns=te.columns_)
df_ary
```

Out[24]:

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
5	False	False	False	False	False	False	False	False
6	False	False	False	False	False	False	False	False
7	False	False	False	False	False	False	False	False
8	False	False	False	False	False	False	False	False
9	False	False	False	False	False	False	False	False
10	False	False	False	False	False	False	False	False
11	False	False	False	False	False	False	False	False
12	False	False	False	False	False	False	False	False
13	False	False	False	False	False	False	False	False
14	False	False	False	False	False	False	False	False
15	False	False	False	False	False	False	False	False
16	False	False	False	False	False	False	False	False
17	False	False	False	False	False	False	False	False
18	False	False	False	False	False	False	False	False
19	False	False	False	False	False	False	False	False
20	False	False	False	False	False	False	False	False
21	False	False	False	False	False	False	False	False
22	False	False	False	False	False	False	False	False
23	False	False	False	False	False	False	False	False
24	False	False	False	False	False	False	False	False
25	False	False	False	False	False	False	False	False
26	False	False	False	False	False	False	False	False
27	False	False	False	False	False	False	False	False
28	False	False	False	False	False	False	False	False
29	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
9435	False	False	False	False	False	False	False	False

	Adjustment	Afternoon with the baker	Alfajores	Argentina Night	Art Tray	Bacon	Baguette	Bakewell
9436	False	False	False	False	False	False	True	False
9437	False	False	False	False	False	False	False	False
9438	False	False	False	False	False	False	False	False
9439	False	False	False	False	False	False	False	False
9440	False	False	False	False	False	False	False	False
9441	False	False	False	False	False	False	False	False
9442	False	False	False	False	False	False	False	False
9443	False	False	False	False	False	False	False	False
9444	False	False	False	False	False	False	False	False
9445	False	False	False	False	False	False	False	False
9446	False	False	False	False	False	False	False	False
9447	False	False	False	False	False	False	False	False
9448	False	False	False	False	False	False	False	False
9449	False	False	False	False	False	False	False	False
9450	False	False	False	False	False	False	False	False
9451	False	False	False	False	False	False	False	False
9452	False	False	False	False	False	False	False	False
9453	False	False	False	False	False	False	False	False
9454	False	False	False	False	False	False	False	False
9455	False	False	False	False	False	False	False	False
9456	False	False	False	False	False	False	False	False
9457	False	False	False	False	False	False	False	False
9458	False	False	False	False	False	False	False	False
9459	False	False	False	False	False	False	False	False
9460	False	False	False	False	False	False	False	False
9461	False	False	False	False	False	False	False	False
9462	False	False	False	False	False	False	False	False
9463	False	False	False	False	False	False	False	False
9464	False	False	False	False	False	False	False	False

9465 rows × 94 columns

In [25]:

```
print('Data-Set Information nach der Pivot-Transformation: \n')
print(df_ary.info())
```

```
Data-Set Information nach der Pivot-Transformation:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9465 entries, 0 to 9464
Data columns (total 94 columns):
Adjustment                       9465 non-null bool
Afternoon with the baker         9465 non-null bool
Alfajores                        9465 non-null bool
Argentina Night                  9465 non-null bool
Art Tray                         9465 non-null bool
Bacon                            9465 non-null bool
Baguette                         9465 non-null bool
Bakewell                         9465 non-null bool
Bare Popcorn                     9465 non-null bool
Basket                           9465 non-null bool
Bowl Nic Pitt                    9465 non-null bool
Bread                            9465 non-null bool
Bread Pudding                    9465 non-null bool
Brioche and salami               9465 non-null bool
Brownie                          9465 non-null bool
Cake                             9465 non-null bool
Caramel bites                    9465 non-null bool
Cherry me Dried fruit            9465 non-null bool
Chicken Stew                     9465 non-null bool
Chicken sand                     9465 non-null bool
Chimichurri Oil                  9465 non-null bool
Chocolates                       9465 non-null bool
Christmas common                 9465 non-null bool
Coffee                           9465 non-null bool
Coffee granules                  9465 non-null bool
Coke                             9465 non-null bool
Cookies                          9465 non-null bool
Crepes                           9465 non-null bool
Crisps                           9465 non-null bool
Drinking chocolate spoons        9465 non-null bool
Duck egg                         9465 non-null bool
Dulce de Leche                   9465 non-null bool
Eggs                             9465 non-null bool
Ella's Kitchen Pouches           9465 non-null bool
Empanadas                        9465 non-null bool
Extra Salami or Feta             9465 non-null bool
Fairy Doors                      9465 non-null bool
Farm House                       9465 non-null bool
Focaccia                         9465 non-null bool
Frittata                         9465 non-null bool
Fudge                            9465 non-null bool
Gift voucher                     9465 non-null bool
Gingerbread syrup                9465 non-null bool
Granola                          9465 non-null bool
Hack the stack                   9465 non-null bool
Half slice Monster               9465 non-null bool
Hearty & Seasonal                9465 non-null bool
Honey                            9465 non-null bool
Hot chocolate                    9465 non-null bool
Jam                              9465 non-null bool
Jammie Dodgers                   9465 non-null bool
Juice                            9465 non-null bool
Keeping It Local                 9465 non-null bool
Kids biscuit                     9465 non-null bool
Lemon and coconut                9465 non-null bool
Medialuna                        9465 non-null bool
```

```
Mighty Protein                  9465 non-null bool
Mineral water                   9465 non-null bool
Mortimer                        9465 non-null bool
Muesli                          9465 non-null bool
Muffin                          9465 non-null bool
My-5 Fruit Shoot                9465 non-null bool
Nomad bag                       9465 non-null bool
Olum & polenta                  9465 non-null bool
Panatone                        9465 non-null bool
Pastry                          9465 non-null bool
Pick and Mix Bowls              9465 non-null bool
Pintxos                         9465 non-null bool
Polenta                         9465 non-null bool
Postcard                        9465 non-null bool
Raspberry shortbread sandwich   9465 non-null bool
Raw bars                        9465 non-null bool
Salad                           9465 non-null bool
Sandwich                        9465 non-null bool
Scandinavian                    9465 non-null bool
Scone                           9465 non-null bool
Siblings                        9465 non-null bool
Smoothies                       9465 non-null bool
Soup                            9465 non-null bool
Spanish Brunch                  9465 non-null bool
Spread                          9465 non-null bool
Tacos/Fajita                    9465 non-null bool
Tartine                         9465 non-null bool
Tea                             9465 non-null bool
The BART                        9465 non-null bool
The Nomad                       9465 non-null bool
Tiffin                          9465 non-null bool
Toast                           9465 non-null bool
Truffles                        9465 non-null bool
Tshirt                          9465 non-null bool
Valentine's card                9465 non-null bool
Vegan Feast                     9465 non-null bool
Vegan mincepie                  9465 non-null bool
Victorian Sponge                9465 non-null bool
dtypes: bool(94)
memory usage: 868.9 KB
None
```

Nachdem das Daten-Set formatiert wurde, werden wir den Apriori-Algorithmus mit den assoziativen Regeln verwenden. Dabei wird für den Lift min_threshold = 1,0 festgelegt. Wenn er weniger als 1,0 ist, werden die beiden Produkte wahrscheinlich nicht zusammen gekauft (siehe Theorie oben). Die Ergebnisse werden nach der Konfidenz absteigend sortiert, um die Wahrscheinlichkeit zu sehen, dass ein Produkt gekauft wird, wenn sein "Vorgänger" gekauft wird.

In [29]:

```
frequent_itemsets = apriori(df_ary, min_support=0.01, use_colnames=True)
rules = association_rules(frequent_itemsets, metric='lift', min_threshold=1.0)
rules.sort_values('confidence', ascending=False)
```

Out[29]:

	antecedents	consequents	antecedent support	consequent support	support	confidence	
35	(Toast)	(Coffee)	0.033597	0.478394	0.023666	0.704403	1.4724
27	(Spanish Brunch)	(Coffee)	0.018172	0.478394	0.010882	0.598837	1.2517
37	(Medialuna)	(Coffee)	0.061807	0.478394	0.035182	0.569231	1.1898
0	(Pastry)	(Coffee)	0.086107	0.478394	0.047544	0.552147	1.1541
10	(Alfajores)	(Coffee)	0.036344	0.478394	0.019651	0.540698	1.1302
38	(Juice)	(Coffee)	0.038563	0.478394	0.020602	0.534247	1.1167
16	(Sandwich)	(Coffee)	0.071844	0.478394	0.038246	0.532353	1.1127
25	(Cake)	(Coffee)	0.103856	0.478394	0.054728	0.526958	1.1015
23	(Scone)	(Coffee)	0.034548	0.478394	0.018067	0.522936	1.0931
9	(Cookies)	(Coffee)	0.054411	0.478394	0.028209	0.518447	1.0837
19	(Hot chocolate)	(Coffee)	0.058320	0.478394	0.029583	0.507246	1.0603
14	(Brownie)	(Coffee)	0.040042	0.478394	0.019651	0.490765	1.0258
33	(Muffin)	(Coffee)	0.038457	0.478394	0.018806	0.489011	1.0221
7	(Pastry)	(Bread)	0.086107	0.327205	0.029160	0.338650	1.0349
3	(Cake)	(Tea)	0.103856	0.142631	0.023772	0.228891	1.6047
28	(Coffee, Tea)	(Cake)	0.049868	0.103856	0.010037	0.201271	1.9379
40	(Sandwich)	(Tea)	0.071844	0.142631	0.014369	0.200000	1.4022
21	(Hot chocolate)	(Cake)	0.058320	0.103856	0.011410	0.195652	1.8838
29	(Coffee, Cake)	(Tea)	0.054728	0.142631	0.010037	0.183398	1.2858
2	(Tea)	(Cake)	0.142631	0.103856	0.023772	0.166667	1.6047
5	(Pastry)	(Bread, Coffee)	0.086107	0.090016	0.011199	0.130061	1.4448
4	(Bread, Coffee)	(Pastry)	0.090016	0.086107	0.011199	0.124413	1.4448
24	(Coffee)	(Cake)	0.478394	0.103856	0.054728	0.114399	1.1015
12	(Bread, Coffee)	(Cake)	0.090016	0.103856	0.010037	0.111502	1.0736
20	(Cake)	(Hot chocolate)	0.103856	0.058320	0.011410	0.109868	1.8838
41	(Tea)	(Sandwich)	0.142631	0.071844	0.014369	0.100741	1.4022
1	(Coffee)	(Pastry)	0.478394	0.086107	0.047544	0.099382	1.1541

	antecedents	consequents	antecedent support	consequent support	support	confidence	
13	(Cake)	(Bread, Coffee)	0.103856	0.090016	0.010037	0.096643	1.0736
31	(Cake)	(Coffee, Tea)	0.103856	0.049868	0.010037	0.096643	1.9379
6	(Bread)	(Pastry)	0.327205	0.086107	0.029160	0.089119	1.0349
17	(Coffee)	(Sandwich)	0.478394	0.071844	0.038246	0.079947	1.1127
36	(Coffee)	(Medialuna)	0.478394	0.061807	0.035182	0.073542	1.1898
30	(Tea)	(Coffee, Cake)	0.142631	0.054728	0.010037	0.070370	1.2858
18	(Coffee)	(Hot chocolate)	0.478394	0.058320	0.029583	0.061837	1.0603
8	(Coffee)	(Cookies)	0.478394	0.054411	0.028209	0.058966	1.0837
34	(Coffee)	(Toast)	0.478394	0.033597	0.023666	0.049470	1.4724
39	(Coffee)	(Juice)	0.478394	0.038563	0.020602	0.043065	1.1167
11	(Coffee)	(Alfajores)	0.478394	0.036344	0.019651	0.041078	1.1302
15	(Coffee)	(Brownie)	0.478394	0.040042	0.019651	0.041078	1.0258
32	(Coffee)	(Muffin)	0.478394	0.038457	0.018806	0.039311	1.0221
22	(Coffee)	(Scone)	0.478394	0.034548	0.018067	0.037765	1.0931
26	(Coffee)	(Spanish Brunch)	0.478394	0.018172	0.010882	0.022747	1.2517

Fazit

Allgemein gesehen (s. Meistverkaufte Produkte oben) ist der Kaffee das beliebteste Produkt. Das ist logisch und selbsterklärend, weil es eine Bäckerei ist.

Außerdem in der ersten Zeile in der Tabelle oben sehen wir, wenn ein Toast gekauft wird, wird zu 70% (s. Konfidenz) auch ein Kaffee gekauft. Oder der Kauf eines Toastes macht den Kauf eines Kaffees um 47% (s. Lift 147%) wahrscheinlicher. Anders gesagt, je höher der Lift, desto stärke die Korrelation zwischen den Produkten.

Aus diesem Grund sind alle Beziehungen zwischen Produkten interessant, wo der Lift hoch ist, auch wenn die Konfidenz niedrig ist z.B. (Format: Vorgänger (antecedents) -> Folger (consequents)):

- Cake -> Tea (Lift 160%)
- Tea -> Cake (Lift 160%)
- (Coffee + Tea) -> Cake (Lift 193% -> ca. 200% -> Wahrscheinlichkeit für das zusätzliche Produkt sich verdoppelt)
- Cake -> (Coffee + Tea) (Lift 193% -> ca. 200% -> Wahrscheinlichkeit für das zusätzliche Produkt sich verdoppelt)
- Sandwhich -> Tea (Lift 140%)
- Tea -> Sandwich (Lift 140%)
- Hot Chocolate -> Cake (Lift 188%)
- Cake -> Hot Chocolate (Lift 188%)

Die Spalten 'leverage' and 'conviction' beinhalten zusätzliche Metriken. Ihre Beschreibung finden Sie unter https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/ (https://rasbt.github.io/mlxtend/user_guide/frequent_patterns/association_rules/)

Was kann eine Bäckerei mit dieser Analyse machen bzw. was für ein Nutzen in dieser Analyse für eine Bäckerei? Unternehmen sind immer bestrebt, ihre Einrichtung zu optimieren und ihren Umsatz zu steigern. Bäckereien sind nicht anders und Warenkorbanalyse kann für jede Art von Ladengeschäft oder Markt durchgeführt werden. Auf Grund der höhen Konfidenz kann man die Produkte näher plazieren und auf Grund des höhen Liftes kann man die Preisgestaltung besser nutzen. Dadurch können bestimmt die neuen Kunden gewinnen werden.

Excel Bonus

Fall die Produktpalette überschaubar ist und die Anzahl der Transaktionen relativ klein ist, kann man diese Analyse in Excel machen. Wenn das Interesse besteht, kann ich ein kleines Beispiel machen. Sonst fragen Sie den Uncle Google. Die Stichworte sind: Warenkorbanalyse, Excel etc. Viel Spaß und Erfolg. Ich freue mich über ein Feedback unter (v (punkt) poliakov (at) gmx (dot) net :-)