Market-basket analysis and prediction

Using Associative Data Mining and Apriori Algorithm

Gokul S Kumar Follow
Jun 18 · 12 min read ★

Photo by Brooke Cagle on Unsplash

Introduction

We all have been shopping more from online e-commerce sites recently, probably due to the lock-down imposed in most parts of the world. You must have noticed an up-sell feature named '***frequently bought together***' in most of these sites, for instance Amazon where it predicts what all items would go-along with the item that you have just added to the cart. Customers are given the choice of adding all the items shown under this feature to the cart or select the required ones. Amazon does it thorough what they call 'item-to-item collaborative filtering' where it runs recommendation algorithms based on the item-search history of the customer to improve the shopping experience. In the case of offline

retailers, it works pretty much the same. Let us consider the basic example of Bread and Jam. If the retailer finds that there is an increase in the sales of bread, he can further up-sell by giving a discount in the price of jam, so that more customers are bound to buy them together.

This entire process of analyzing the shopping trends of customers is called '***Market Basket Analysis***'. It is an analyzing technique based on the idea that if we buy an item then we are bound to buy or not-buy a group (or single) items. For example, if a customer is buying bread then the chances of him/her buying jam is more. This is represented by the following equation:

Association Mining Rule

This equation is called the Association Mining Rule. This can be thought of as an IF-THEN relationship. If item A is bought by a customer then the chances of item B being bought by the same user in the same transaction is found out. Here A is called the Antecedent and B the consequent. Antecedents are primary item that are found in the basket and consequents

are the items that are found with an antecedent/group of antecedents. The metrics for measuring association are:

Support: It tells us about the combination of items bought together frequently. It gives the part of transactions that contain both A and B.

We can filter out the less frequently occurring items-sets using support.

Confidence: It tells us how frequently the items A and B are bought together, for the no. of times A is bought.

Lift: It indicates the strength of a rule over the randomness of A and B being bought together. It basically measures the strength of any association rule(we will talk about association rules below).

More the lift more is the strength of the rule. If the lift is 3 for A -> B then it means if we buy A, the chances of buying B is 3 times.

So the process is to make rules for each item-set to figure out the metrics of its association so as to decide whether to include it in the analysis or not. But consider a large data-set having millions of user and transactions, thereby producing a large amount of item-sets. So to make rules for all of these would be a humongous task. This is where Apriori algorithm enters the scene.

Apriori algorithm uses frequently bought item-sets to generate association rules. It is built on the idea that the subset of a frequently bought items-set is also a frequently bought item-set. Frequently bought item-sets are decided if their support value is above a minimum threshold support value.

To demonstrate the working of this algorithm consider the following transactions:

Transaction_ID	Items bought
1	A C D
2	B C E

3	A B C E
4	B E
5	A C E

Transaction details

There are 5 items A,B,C,D and E in total which are bought together in different combinations in each transaction. Lets fix the minimum threshold support value of item-sets formed to be 2.

Iteration-1: Item-sets having 1 item is formed and their support is calculated.

			C1	
Transaction_ID	**Items bought**		**Item-Set**	**Support**
1	A C D	---→	{A}	3
2	B C E		{B}	3
3	A B C E		{C}	4
4	B E		{D}	1
5	A C E		{E}	4

Iteration-1

As seen from the support values of each item-set, D has a support of 1 which is less than the threshold value. So we are going to omit that item-set.

C1	
Item-Set	Support
{A}	3
{B}	3
{C}	4
{D}	1
{E}	4

---→

F1	
Item-Set	Support
{A}	3
{B}	3
{C}	4
{E}	4

Iteration-1 final

Iteration-2: Next we make all the possible item-sets having 2 items in them. All the items in the table F1 are used in this.

Transaction_ID	Items bought
1	A C D
2	B C E
3	A B C E
4	B E
5	A C E

---→

Only items in F1

C2	
Item-Set	Support
{A,B}	1
{A,C}	3
{A,E}	2
{B,C}	2
{B,E}	3
{C,E}	3

---→

F2	
Item-Set	Support
{A,C}	3
{A,E}	2
{B,C}	2
{B,E}	3
{C,E}	3

Iteration-2

Item-sets having support less than 2 are again omitted, {A,B} in this iteration.

Iteration-3: All the possible item-sets having 3 items in them are listed. Then we are going to divide these item-sets into their sub-sets and omit those having a support value less than the threshold i.e., 2. This process is called **Pruning**.

Transaction_ID	Items bought
1	A C D
2	B C E
3	A B C E
4	B E
5	A C E

---→

C3	
Item-Set	In F2 ?
{A,B,C}, {A,B}, {A,C}, {B,C}	NO
{A,B,E}, {A,B}, {A,E}, {B,E}	NO
{A,C,E}, {A,E}, {A,C}, {C,E}	YES
{B,C,E},{B,C},{B,E},{C,E}	YES

Iteration-3 and pruning

We will omit {A,B,C} and {A,B,E} as they both contain {A,B} which was omitted in Iteration-2. This pruning section is the key part of Apriori algorithm.

Transaction_ID	Items bought
1	A C D
2	B C E
3	A B C E
4	B E
5	A C E

---→

F3	
Item-Set	Support
{A,C,E}	2
{B,C,E}	2

Iteration-3 final

Iteration-4: Using F3 we will create item-sets having 4 items.

Transaction_ID	Items bought
1	A C D
2	B C E
3	A B C E
4	B E
5	A C E

---→

F3

Item-Set	Support
{A,C,E}	2
{B,C,E}	2

---→

Item-Set	Support
{A,B,C,E}	1

Iteration-4

As we can see, the support of the only item-set having 4 items is less than 2. So we stop the iterations here and the final item-set is F3.

If I={A,C,E} then the subsets are {A,C}, {A,E}, {C,E}, {A}, {C}, {E}.

If I={B,C,E} the the subsets are {B,C}, {B,E}, {C,E}, {B}, {C}, {E}.

Rules: For filtering out the relevant item-sets we will create certain rules and apply it to the subsets. Assume a minimum confidence value of 60%.

For every subset S of I we make the rule

S →(I-S) (which means if S then I minus S) if

support(I)/support(S) ≥ minimum confidence value i.e., 60%.

Consider {A,C,E}

Rule 1: {A,C} →({A,C,E}- {A,C}) which is {A,C} → {E}

Confidence = support{A,C,E}/support{A,C} = 2/3 = 66.6% > 60%

So rule 1 i.e., {A,C} → {E} is selected.

Rule 2: {A,E} →({A,C,E}- {A,E}) which is {A,E} → {C}

Confidence = support{A,C,E}/support{A,E} = 2/2 = 100% > 60%

So rule 2 i.e., {A,E} → {C} is selected.

Rule 3: {C,E} →({A,C,E}- {C,E}) which is {C,E} → {A}

Confidence = support{A,C,E}/support{C,E} = 2/3 = 66.6% > 60%

So rule 3 i.e., {C,E} → {A} is selected.

Rule 4: {A} →({A,C,E}- {A}) which is {A} → {C,E}

Confidence = support{A,C,E}/support{A} = 2/3 = 66.6% > 60%

So rule 4 i.e., {A} → {C,E} is selected.

Rule 5: {C} →({A,C,E}- {C}) which is {C} → {A,E}

Confidence = support{A,C,E}/support{C} = 2/4 = 50% < 60%

So rule 5 i.e., {C} → {A,E} is rejected.

Rule 6: {E} →({A,C,E}- {E}) which is {E} → {A,C}

Confidence = support{A,C,E}/support{E} = 2/4 = 50% < 60%

So rule 6 i.e., {E} → {A,C} is rejected.

Same steps can be done to {B,C,E}. Lift is not being used as the size of the input data-set is small and there is no need of further filtering. But in the case of larger data set further filtering is done by imposing a minimum lift value for the rules. The values of all three of the association metrics can be tweaked as per requirement.

Lets consider an actual data-set and see how the analysis is done.

The link to the data-set being used is given below

transaction_data.csv

Edit description

drive.google.com

We are going to sue python and pandas. We are also going to use Mlxtend lib as it contains inbuilt functions for Apriori algorithm and Associaction rules.

Load the necessary libraries:

```
import pandas as pd
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import matplotlib.pyplot as plt
from matplotlib import style
import numpy as np
```

Ingest the data into a pandas data-frame and try to know about the features.

```
read_df = pd.read_csv('transaction_data.csv')
df = read_df.copy()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1083818 entries, 0 to 1083817
Data columns (total 8 columns):
 #   Column                  Non-Null Count    Dtype
---  ------                  --------------    -----
 0   UserId                  1083818 non-null  int64
 1   TransactionId           1083818 non-null  int64
 2   TransactionTime         1083818 non-null  object
 3   ItemCode                1083818 non-null  int64
 4   ItemDescription         1080910 non-null  object
 5   NumberOfItemsPurchased  1083818 non-null  int64
 6   CostPerItem             1083818 non-null  float64
 7   Country                 1083818 non-null  object
dtypes: float64(1), int64(4), object(3)
memory usage: 66.2+ MB
```

Further description of the features are as follows:

UserId — Unique identifier of a user.

TransactionId — Unique identifier of a transaction. If the same TransactionId is present in multiple rows, then all those products are bought together in the same transaction.

TransactionTime — Time at which the transaction is performed

ItemCode — Unique identifier of the product purchased

ItemDescription — Simple description of the product purchased

NumberOfItemsPurchased — Quantity of the product purchased in the transaction

CostPerItem — Price per each unit of the product

Country — Country from which the purchase is made.

EDA and data cleaning is done as follows:

```
df = df[df.UserId>0] # usedid <=0 : 25%
df = df[df.ItemCode>0]
df = df[df.NumberOfItemsPurchased>0]
df = df[df.CostPerItem>0]
df = df[df.ItemDescription.notna()]
df = df[df.TransactionTime.str[-4:] != '2028']
```

The following data-frame is obtained:

	UserId	TransactionId	TransactionTime	ItemCode	ItemDescription	NumberOfItemsPurchased	CostPerItem	Country
0	278166	6355745	Sat Feb 02 12:50:00 IST 2019	465549	FAMILY ALBUM WHITE PICTURE FRAME	6	11.73	United Kingdom
1	337701	6283376	Wed Dec 26 09:06:00 IST 2018	482370	LONDON BUS COFFEE MUG	3	3.52	United Kingdom
2	267099	6385599	Fri Feb 15 09:45:00 IST 2019	490728	SET 12 COLOUR PENCILS DOLLY GIRL	72	0.90	France
3	380478	6044973	Fri Jun 22 07:14:00 IST 2018	459186	UNION JACK FLAG LUGGAGE TAG	3	1.73	United Kingdom
5	285957	6307136	Fri Jan 11 09:50:00 IST 2019	1787247	CUT GLASS T-LIGHT HOLDER OCTAGON	12	3.52	United Kingdom

First five rows of the data-frame

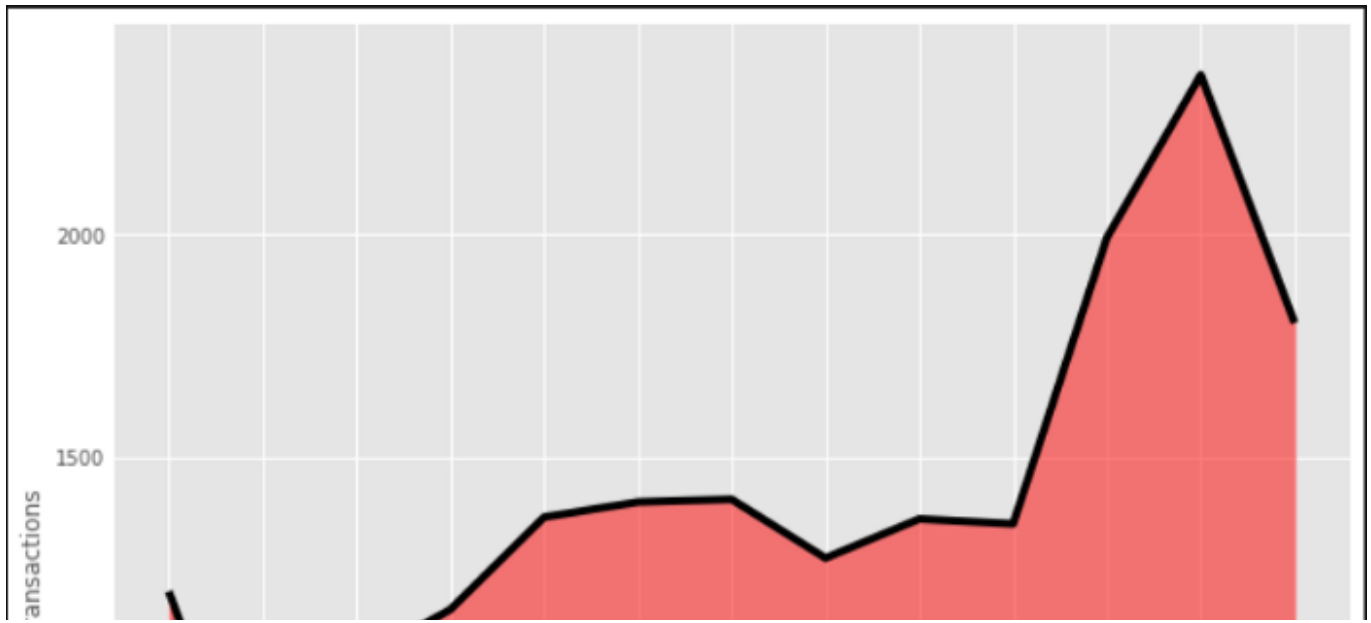
EDA:

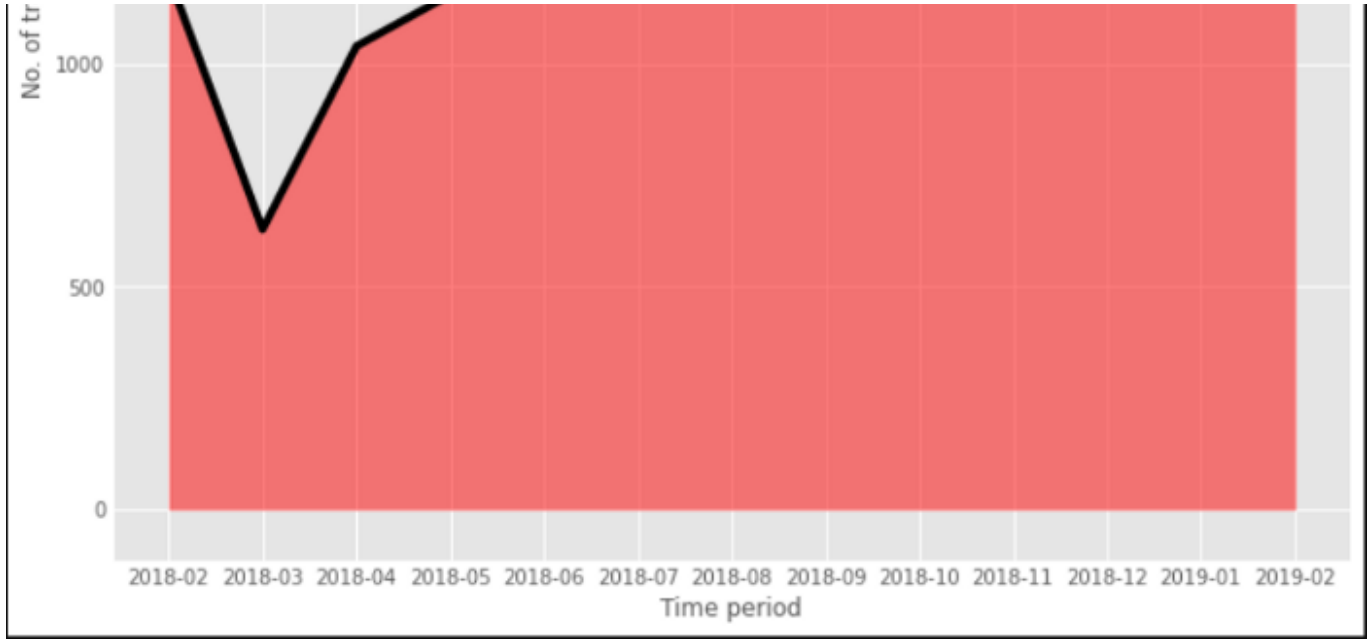
Lets do some exploratory data analysis now. Lets see the no. of transactions being done in each part of the year.

```
df.TransactionTime = pd.to_datetime(df.TransactionTime)
df['month_year']=
pd.to_datetime(df.TransactionTime).dt.to_period('M')
df.sort_values(by = ['month_year'], inplace = True)
```

```
Ser = df.groupby('month_year').TransactionId.nunique()
x = np.arange(0,len(Ser),1)
style.use('ggplot')
fig = plt.figure(figsize = (10,10))
ax1 = fig.add_subplot(111)
ax1.plot(x, Ser, color = 'k')
ax1.fill_between(x, Ser, color = 'r', alpha = 0.5)
ax1.set_xticks(x)
ax1.set_xticklabels(Ser.index)
plt.xlabel('Time period')
plt.ylabel('No. of transactions')
```

Basically we are creating a column named month_year to divide the datapoints into the month in which they occurred. Then we are taking the no. of unique transaction occurring in each month and plotting it using matplotlib.





Transaction done in each month

As we can see the as the time advances the online retailer is getting more and more purchases which maxed in the month of Jan'19.

Lets explore the no. of items bought in each transaction.

```
Ser = df.groupby('TransactionId').ItemDescription.nunique()
Ser.describe()
```

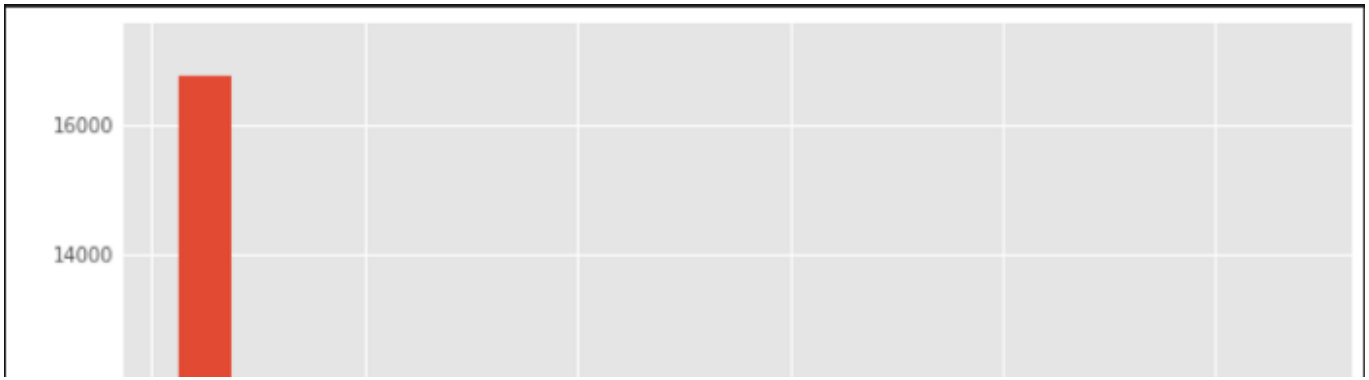
```
count    18334.000000
mean        21.014236
```

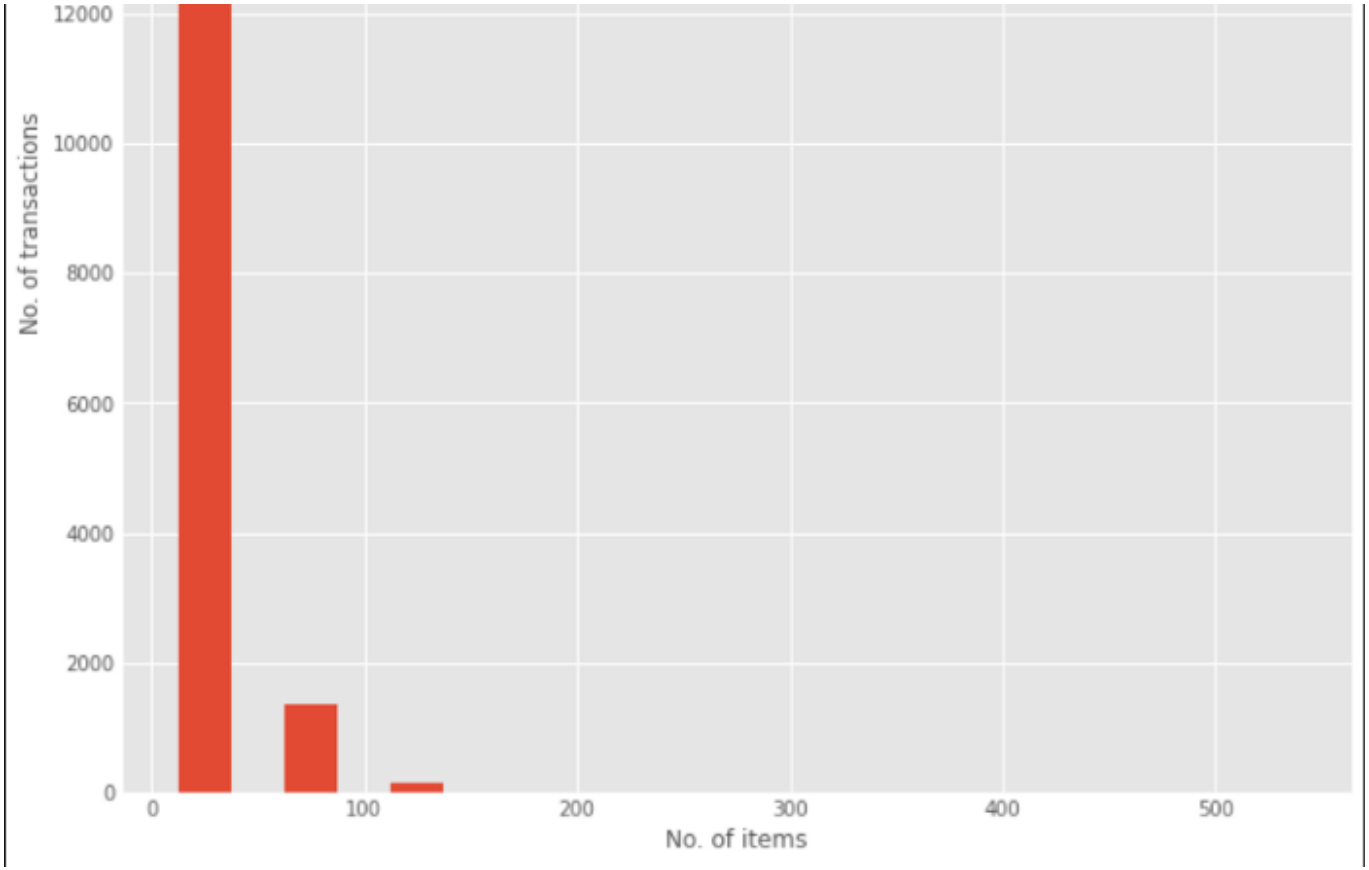
```
std          23.838977
min           1.000000
25%           7.000000
50%          15.000000
75%          27.000000
max         540.000000
Name: ItemDescription, dtype: float64
```

Description of Ser

As we can see the min no. of items is 1 and the max is 540. So we need to plot our histogram as shown below:

```
bins = [0,50,100,150,200,250,300,350,400,450,500,550]
fig = plt.figure(figsize = (10,10))
plt.hist(Ser, bins, histtype = 'bar', rwidth = 0.5)
plt.xlabel('No. of items')
plt.ylabel('No. of transactions')
plt.show()
```

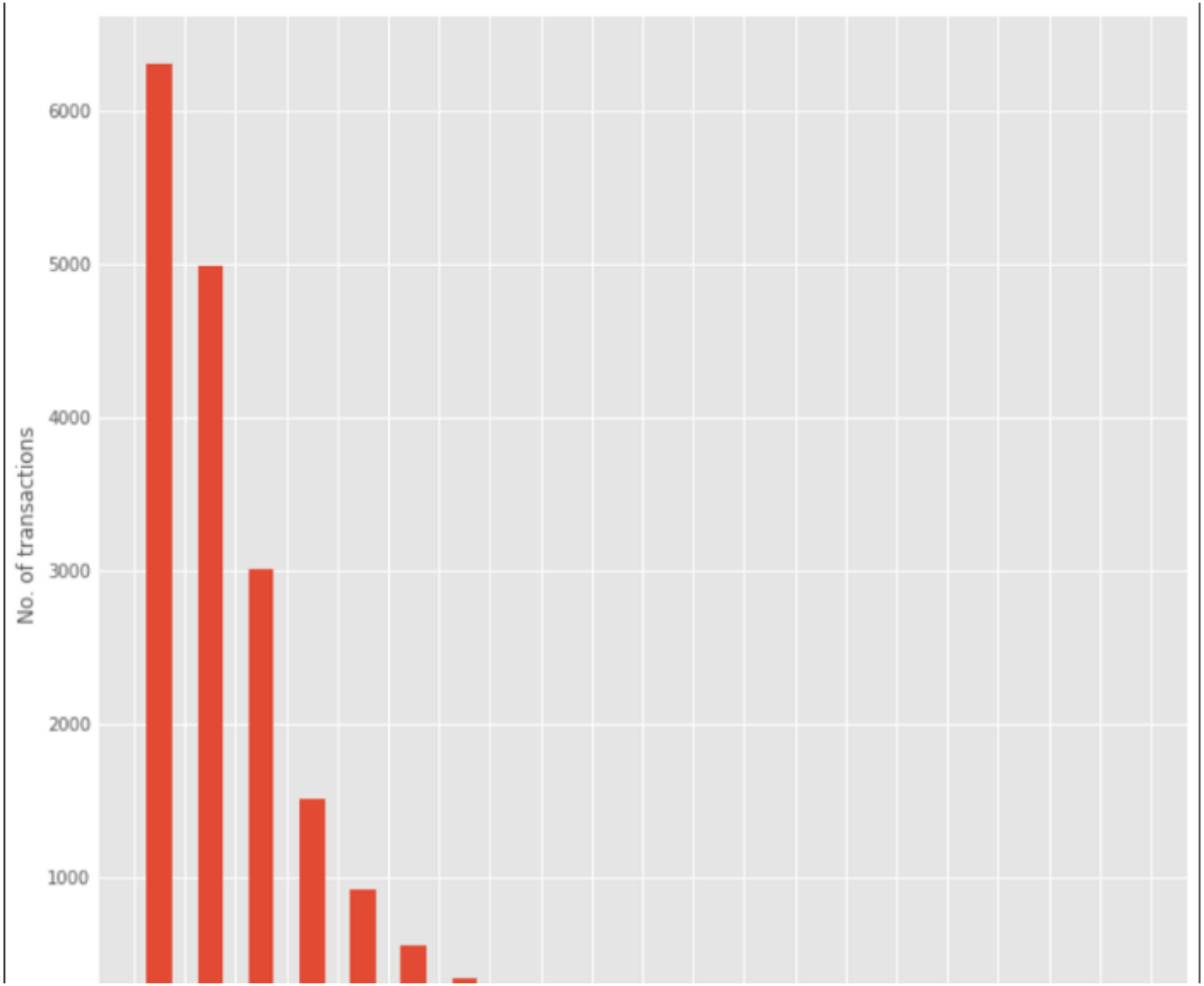


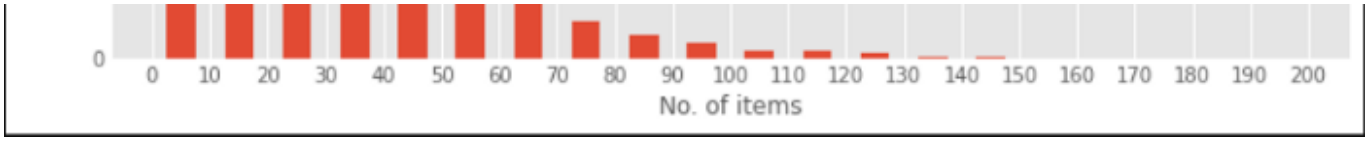


Oops! We can see that most of the transactions include items between 0–100 and some between 100–200. The max items transactions shown above probably is an outlier or is a customer who purchases in a large scale. So we need to re-scale our histogram.

```
bins =
[0,10,20,30,40,50,60,70,80,90,100,110,120,130,140,150,160,170,180,190
,200]
fig = plt.figure(figsize = (10,10))
```

```
ax1 = fig.add_subplot(111)
ax1.hist(Ser, bins, histtype = 'bar', rwidth = 0.5)
ax1.set_xticks(bins)
plt.xlabel('No. of items')
plt.ylabel('No. of transactions')
plt.show()
```

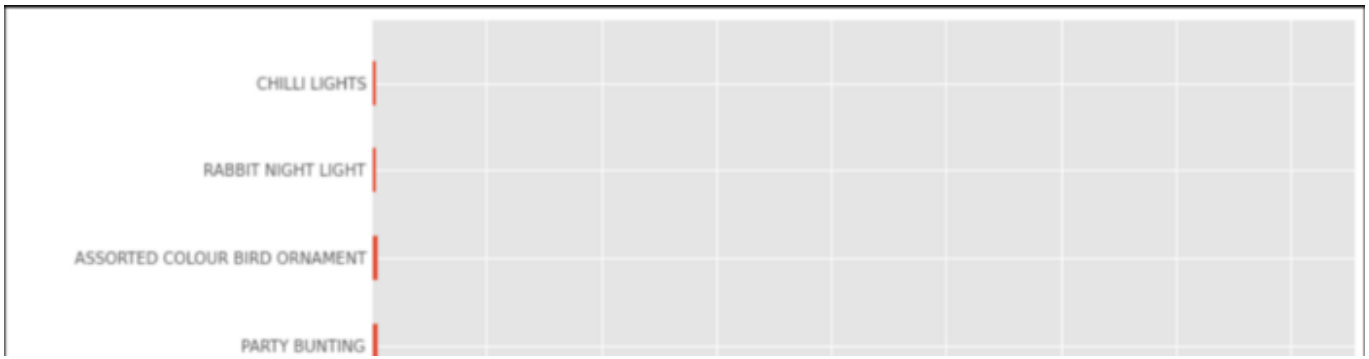


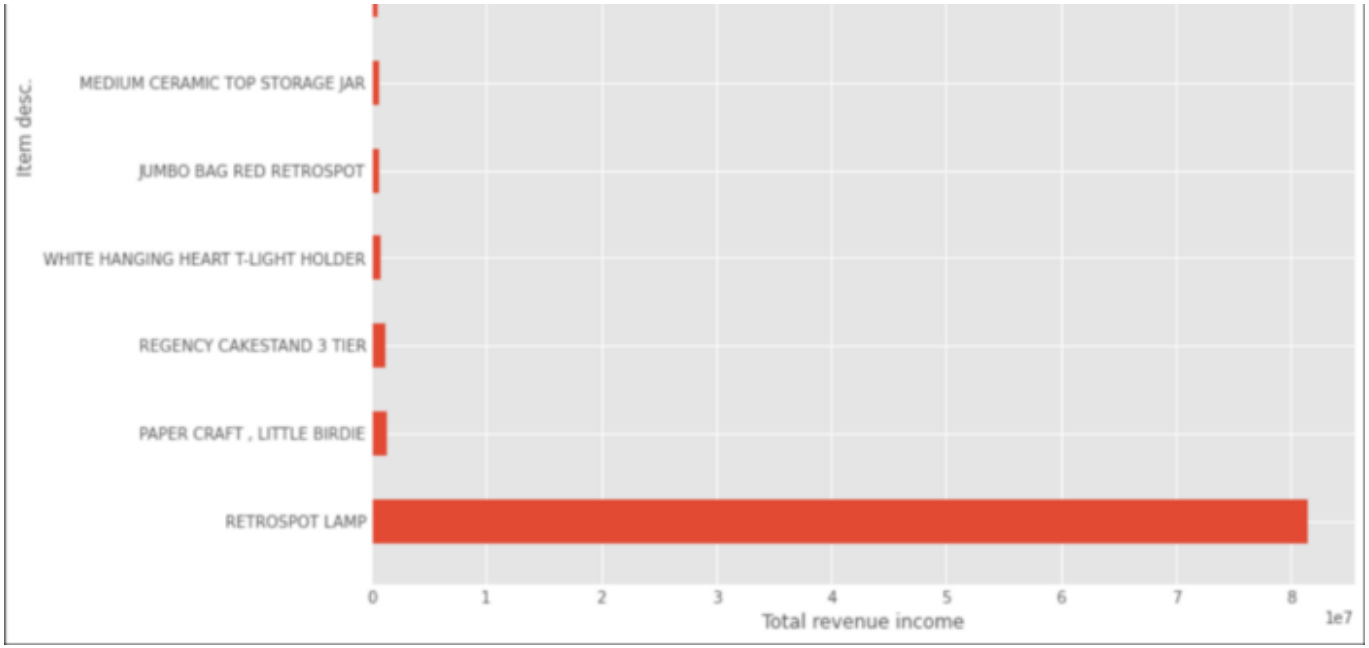


As we can see most transactions included less than 10 items.

Lets find out the most-selling item in the market. This can be interpreted in many way such as the item which brings in the maximum revenue, items which were found in maximum no. of transactions etc. We will be considering the items which bring in maximum revenue.

```
df['total_cost_item'] = df.NumberOfItemsPurchased*df.CostPerItem
Ser = df.groupby('ItemDescription').total_cost_item.sum()
Ser.sort_values(ascending = False, inplace = True)
Ser = Ser[:10]
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111)
ax.barh(Ser.index, Ser, height = 0.5)
```





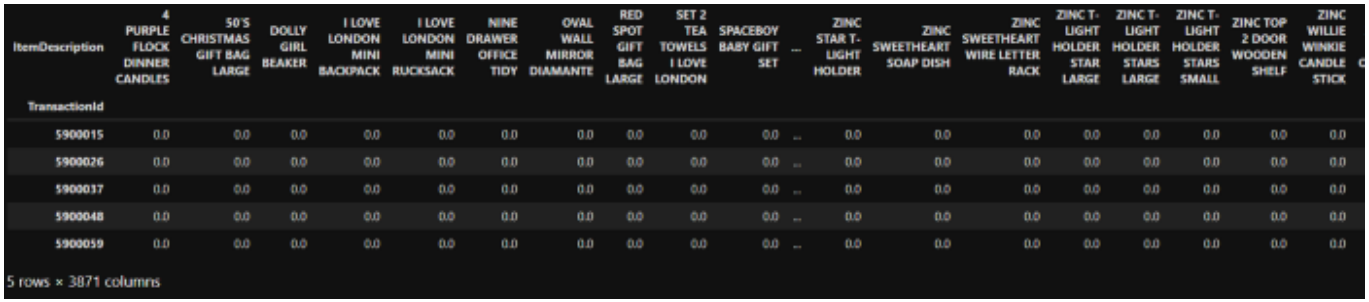
From the above plot showing the top 10 items, we can see that the item ‘retrospot lamp’ has maximum sales value.

The no. of unique TransactionId’s is found to be 18334 and the no. of unique ItemDescription’s is found to be 3871.

Now that we have our data after all the pre-processing, lets arrange it with the TransactionId’s as the index and the ItemDescriptions as the columns with the total no. of items bought in each transaction of each item as the data-point.

```
df_set = df.groupby(['TransactionId',
'ItemDescription']).NumberOfItemsPurchased.sum().unstack().reset_inde
x().fillna(0).set_index('TransactionId')
```

The following data-frame is obtained:



ItemDescription	4 PURPLE FLOCK DINNER CANDLES	50'S CHRISTMAS GIFT BAG LARGE	DOLLY GIRL BEAKER	I LOVE LONDON MINI BACKPACK	I LOVE LONDON MINI RUCKSACK	NINE DRAWER OFFICE TIDY	OVAL WALL MIRROR DIAMANTE	RED SPOT GIFT BAG LARGE	SET 2 TEA TOWELS I LOVE LONDON	SPACEBOY BABY GIFT SET	...	ZINC STAR T-LIGHT HOLDER	ZINC SWEETHEART SOAP DISH	ZINC SWEETHEART WIRE LETTER RACK	ZINC T-LIGHT HOLDER STAR LARGE	ZINC T-LIGHT HOLDER STARS LARGE	ZINC T-LIGHT HOLDER STARS SMALL	ZINC TOP 2 DOOR WOODEN SHELF	ZINC WILLIE WINKIE CANDLE STICK
TransactionId																			
5900015	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5900026	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5900037	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5900048	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5900059	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

5 rows × 3871 columns

We need to make sure that any positive values are encoded to one and all the negative values(if any) to zero.

```
def encode(x):
 if x <= 0:
 return 0
 else:
 return 1
df_set = df_set.applymap(encode)
df_set
```

As the data-frame is ready we can apply Apriori algorithm to get the frequently bought item-sets.

```
frequent_itemsets = apriori(df_set, min_support = 0.015, use_colnames
= True)
```

Here the minimum threshold support value is set as 1.5%. We get the following item-sets.

	support	itemsets
0	0.016690	(12 PENCILS SMALL TUBE RED RETROSPOT)
1	0.015872	(12 PENCILS SMALL TUBE SKULL)
2	0.016581	(3 HOOK PHOTO SHELF ANTIQUE WHITE)
3	0.016145	(3 PIECE SPACEBOY COOKIE CUTTER SET)
4	0.021926	(3 STRIPEY MICE FELTCRAFT)
...	...	...
447	0.017345	(WOODEN HEART CHRISTMAS SCANDINAVIAN, WOODEN S...
448	0.021272	(PINK REGENCY TEACUP AND SAUCER, ROSES REGENCY...
449	0.017018	(ROSES REGENCY TEACUP AND SAUCER , REGENCY CAK...
450	0.016799	(LUNCH BAG BLACK SKULL., LUNCH BAG PINK POLKA...
451	0.015218	(LUNCH BAG PINK POLKADOT, LUNCH BAG CARS BLUE,...

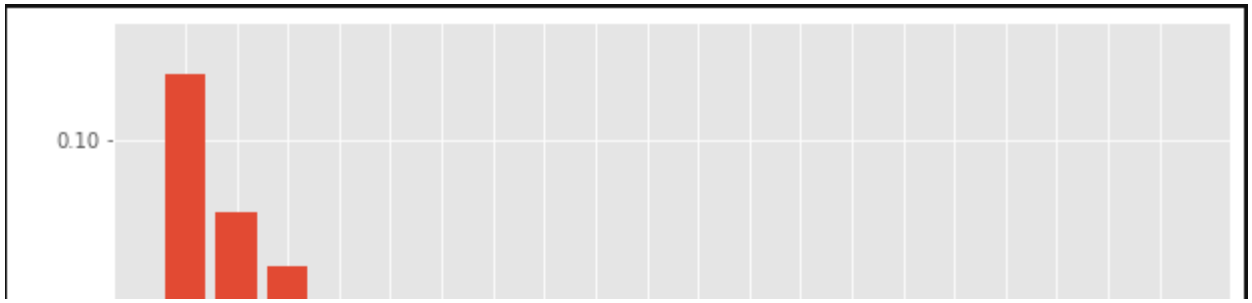
452 rows × 2 columns

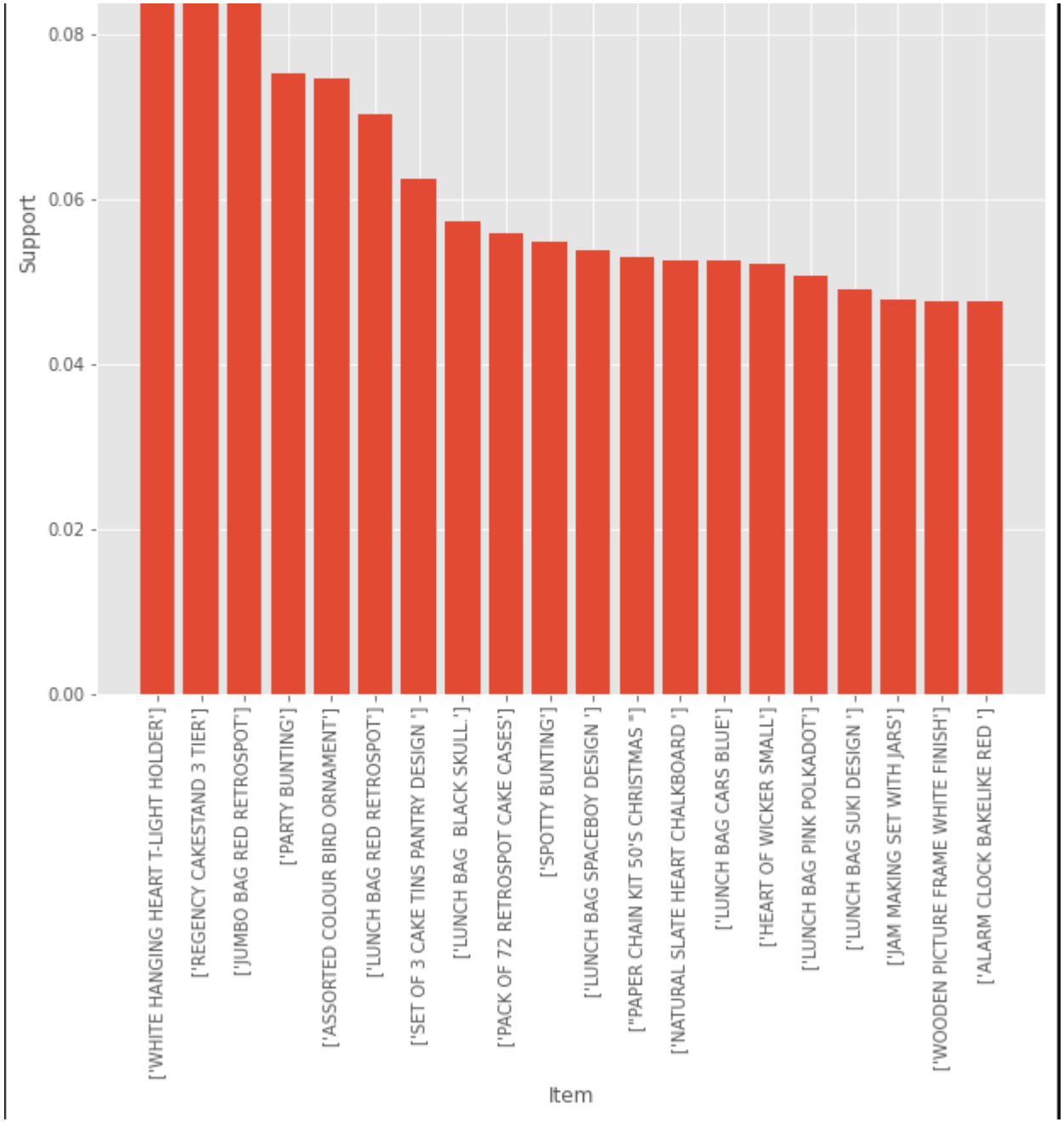
We then arrange the itemsets in descending order of their support values which gives us

```
frequent_itemsets = apriori(df_set, min_support = 0.015, use_colnames
= True)
top_items = frequent_itemsets.sort_values('support', ascending =
False)[:20]
for i in range(len(top_items.itemsets)):
    top_items.itemsets.iloc[i] =
str(list(top_items.itemsets.iloc[i]))
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111)
ax.bar(top_items.itemsets, top_items.support)
for label in ax.xaxis.get_ticklabels():
    label.set_rotation(90)
plt.xlabel('Item')
plt.ylabel('Support')
```

	support	itemsets
339	0.107	['WHITE HANGING HEART T-LIGHT HOLDER']
250	0.093	['REGENCY CAKESTAND 3 TIER']
140	0.087	['JUMBO BAG RED RETROSPOT']
205	0.075	['PARTY BUNTING']

21	0.075	['ASSORTED COLOUR BIRD ORNAMENT']
166	0.070	['LUNCH BAG RED RETROSPOT']
278	0.063	['SET OF 3 CAKE TINS PANTRY DESIGN ']
158	0.057	['LUNCH BAG BLACK SKULL.']
195	0.056	['PACK OF 72 RETROSPOT CAKE CASES']
312	0.055	['SPOTTY BUNTING']
167	0.054	['LUNCH BAG SPACEBOY DESIGN ']
201	0.053	["PAPER CHAIN KIT 50'S CHRISTMAS "]
181	0.053	['NATURAL SLATE HEART CHALKBOARD ']
161	0.053	['LUNCH BAG CARS BLUE']
115	0.052	['HEART OF WICKER SMALL']
165	0.051	['LUNCH BAG PINK POLKADOT']
168	0.049	['LUNCH BAG SUKI DESIGN ']
127	0.048	['JAM MAKING SET WITH JARS']
351	0.048	['WOODEN PICTURE FRAME WHITE FINISH']
17	0.048	['ALARM CLOCK BAKELIKE RED ']




Support
0.08
0.06
0.04
0.02
0.00
['WHITE HANGING HEART T-LIGHT HOLDER'] -
['REGENCY CAKESTAND 3 TIER'] -
['JUMBO BAG RED RETROSPOT'] -
['PARTY BUNTING'] -
['ASSORTED COLOUR BIRD ORNAMENT'] -
['LUNCH BAG RED RETROSPOT'] -
['SET OF 3 CAKE TINS PANTRY DESIGN '] -
['LUNCH BAG BLACK SKULL '] -
['PACK OF 72 RETROSPOT CAKE CASES'] -
['SPOTTY BUNTING'] -
['LUNCH BAG SPACEBOY DESIGN '] -
["PAPER CHAIN KIT 50'S CHRISTMAS "] -
['NATURAL SLATE HEART CHALKBOARD '] -
['LUNCH BAG CARS BLUE'] -
['HEART OF WICKER SMALL'] -
['LUNCH BAG PINK POLKADOT'] -
['LUNCH BAG SUKI DESIGN '] -
['JAM MAKING SET WITH JARS'] -
['WOODEN PICTURE FRAME WHITE FINISH'] -
['ALARM CLOCK BAKELIKE RED '] -
Item

We then apply the association rules to these item-sets formed by Apriori algorithm.

```
rules = association_rules(frequent_itemsets, metric = 'confidence',
min_threshold = 0.2)
```

The metric used here is confidence and its minimum threshold value is set to be 0.2.

The following data-frame is obtained

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(60 TEATIME FAIRY CAKE CASES)	(PACK OF 60 PINK PAISLEY CAKE CASES)	0.036	0.036	0.015	0.427	11.715	0.014	1.683
1	(PACK OF 60 PINK PAISLEY CAKE CASES)	(60 TEATIME FAIRY CAKE CASES)	0.036	0.036	0.015	0.419	11.715	0.014	1.658
2	(PACK OF 72 RETROSPOT CAKE CASES)	(60 TEATIME FAIRY CAKE CASES)	0.056	0.036	0.018	0.322	9.002	0.016	1.421
3	(60 TEATIME FAIRY CAKE CASES)	(PACK OF 72 RETROSPOT CAKE CASES)	0.036	0.056	0.018	0.502	9.002	0.016	1.897
4	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.043	0.033	0.019	0.433	13.018	0.017	1.705
...	...	...	...	...	...	...	...	...	...
182	(LUNCH BAG CARS BLUE, LUNCH BAG RED RETROSPOT)	(LUNCH BAG PINK POLKADOT)	0.025	0.051	0.015	0.609	11.996	0.014	2.429
183	(LUNCH BAG PINK POLKADOT, LUNCH BAG CARS BLUE)	(LUNCH BAG RED RETROSPOT)	0.023	0.070	0.015	0.653	9.308	0.014	2.683
184	(LUNCH BAG RED RETROSPOT)	(LUNCH BAG PINK POLKADOT, LUNCH BAG CARS BLUE)	0.070	0.023	0.015	0.217	9.308	0.014	1.247
185	(LUNCH BAG PINK POLKADOT)	(LUNCH BAG CARS BLUE, LUNCH BAG RED RETROSPOT)	0.051	0.025	0.015	0.300	11.996	0.014	1.392
186	(LUNCH BAG CARS BLUE)	(LUNCH BAG PINK POLKADOT, LUNCH BAG RED RETROS...	0.053	0.029	0.015	0.289	10.135	0.014	1.367

187 rows × 9 columns

All the antecedents with the corresponding consequents are listed with their individual supports, the total support of the item-set and all other

metrics.

	antecedent support	consequent support	support	confidence	lift	leverage	conviction
count	187.000	187.000	187.000	187.000	187.000	187.000	187.000
mean	0.047	0.049	0.020	0.456	10.436	0.017	1.957
std	0.016	0.019	0.004	0.144	5.143	0.004	0.942
min	0.020	0.020	0.015	0.202	1.894	0.007	1.120
25%	0.036	0.037	0.017	0.356	7.076	0.015	1.472
50%	0.045	0.047	0.019	0.423	8.711	0.016	1.655
75%	0.053	0.054	0.023	0.551	13.018	0.020	2.062
max	0.107	0.107	0.030	0.894	29.212	0.028	9.121

The summary of the rules data-frames gives us the following insights:

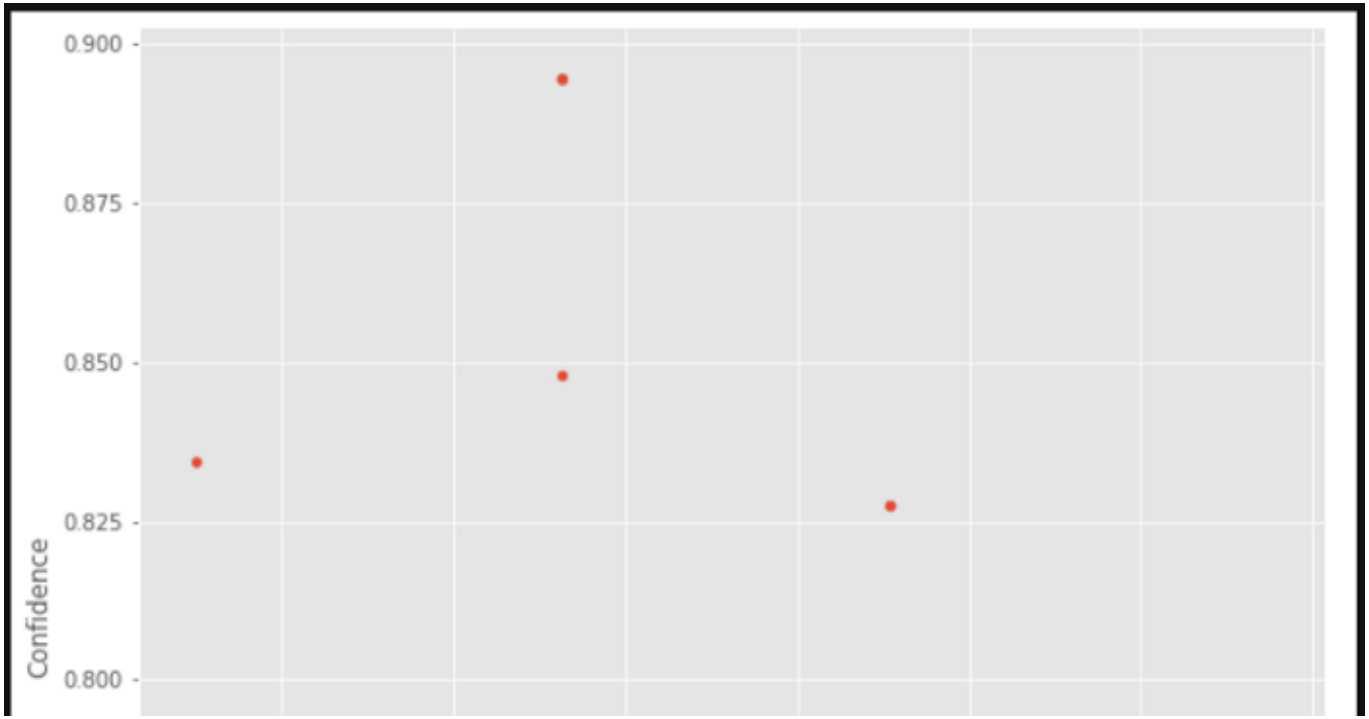
1. There are a total of 187 rules.
2. The summary of various metrics.

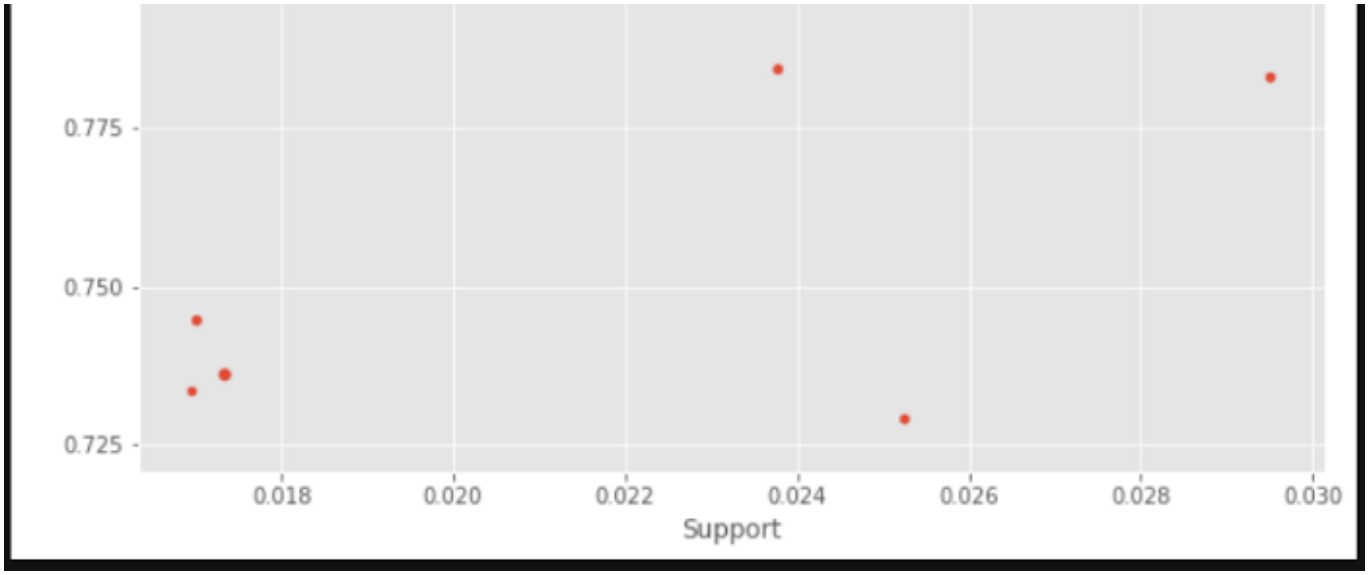
Lets take the top rules having the highest confidence

```
top_rules = rules.sort_values('confidence', ascending = False)[:10]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
164	(ROSES REGENCY TEACUP AND SAUCER , PINK REGENC...	(GREEN REGENCY TEACUP AND SAUCER)	0.024	0.038	0.021	0.894	23.733	0.020	9.121
166	(PINK REGENCY TEACUP AND SAUCER, GREEN REGENCY...	(ROSES REGENCY TEACUP AND SAUCER)	0.025	0.043	0.021	0.848	19.877	0.020	6.291
172	(REGENCY CAKESTAND 3 TIER, GREEN REGENCY TEACU...	(ROSES REGENCY TEACUP AND SAUCER)	0.020	0.043	0.017	0.834	19.558	0.016	5.775
24	(PINK REGENCY TEACUP AND SAUCER)	(GREEN REGENCY TEACUP AND SAUCER)	0.030	0.038	0.025	0.827	21.951	0.024	5.573
146	(PINK REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.030	0.043	0.024	0.784	18.385	0.022	4.436
29	(GREEN REGENCY TEACUP AND SAUCER)	(ROSES REGENCY TEACUP AND SAUCER)	0.038	0.043	0.030	0.783	18.356	0.028	4.410
170	(ROSES REGENCY TEACUP AND SAUCER , REGENCY CAK...	(GREEN REGENCY TEACUP AND SAUCER)	0.023	0.038	0.017	0.745	19.757	0.016	3.768
163	(WOODEN STAR CHRISTMAS SCANDINAVIAN)	(WOODEN HEART CHRISTMAS SCANDINAVIAN)	0.024	0.025	0.017	0.736	29.212	0.017	3.694
13	(BAKING SET SPACEBOY DESIGN)	(BAKING SET 9 PIECE RETROSPOT)	0.023	0.047	0.017	0.733	15.747	0.016	3.577
23	(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.035	0.041	0.025	0.729	17.682	0.024	3.540

```
fig = plt.figure(figsize = (10,10))
ax = fig.add_subplot(111)
ax.scatter(top_rules.support, top_rules.confidence, top_rules.lift)
```

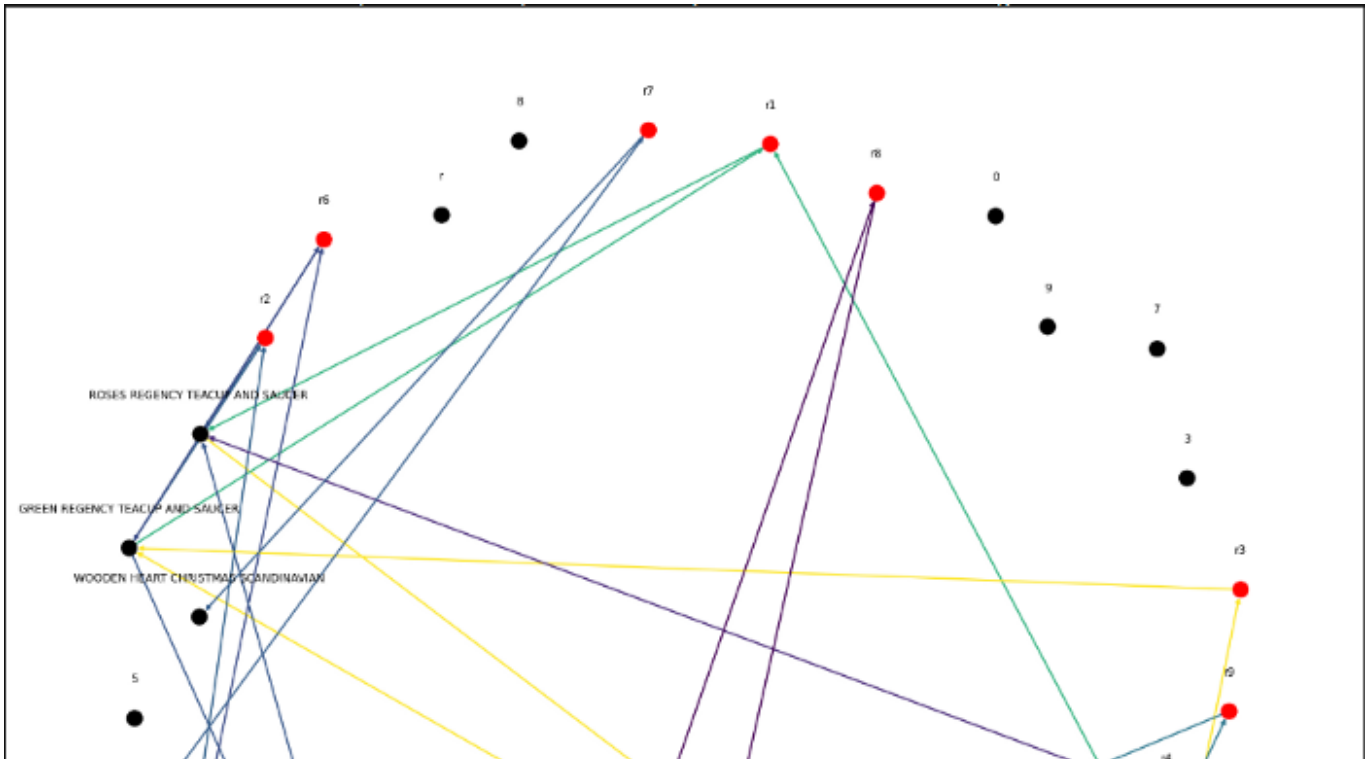


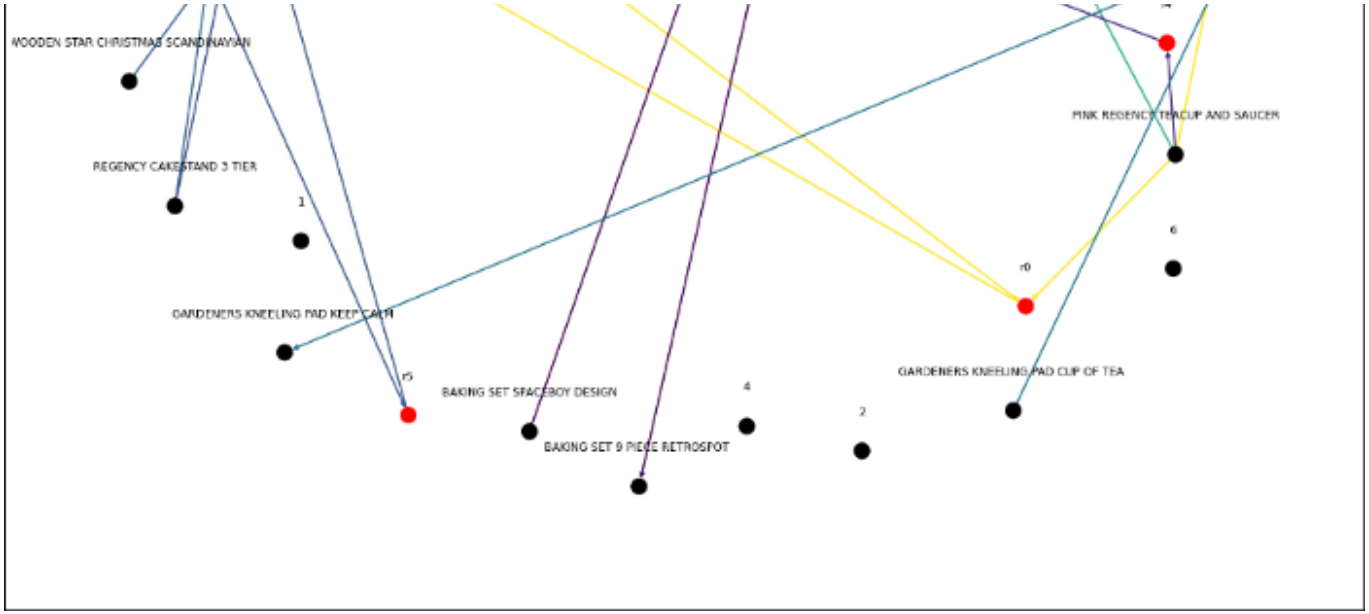


Plotting the rules in a graph:

```
import networkx as nx
G1 = nx.DiGraph()
color_map = []
N = 50
colors = np.random.rand(N)
strs = ['r0', 'r1', 'r2', 'r3', 'r4', 'r5', 'r6', 'r7', 'r8', 'r9']
for i in range(10):
    G1.add_nodes_from('r'+str(i))
    for a in top_rules.iloc[i]['antecedents']:
        G1.add_nodes_from([a])
        G1.add_edge(a, 'r'+str(i), color = colors[i], weight = 2)
    for c in top_rules.iloc[i]['consequents']:
        G1.add_nodes_from([c])
        G1.add_edge('r'+str(i), c, color = colors[i], weight = 2)
for node in G1:
    found_a_string = False
    for item in strs:
```

```
            if node == item:
                found_a_string = True
        if found_a_string:
            color_map.append('red')
        else:
            color_map.append('black')
edges = G1.edges()
colors = [G1[u][v]['color'] for u,v in edges]
weights = [G1[u][v]['weight'] for u,v in edges]
pos = nx.spring_layout(G1, k = 16, scale = 1)
fig = plt.figure(figsize = (20,20))
nx.draw(G1, pos, edges = edges, node_color = color_map, edge_color =
colors, width = weights, font_size = 16, with_labels = False)
for p in pos:
    pos[p][1] += 0.07
nx.draw_networkx_labels(G1, pos)
```





All these plotting and and further analysis can be done more easily in R rather than python as there are more compatible libraries for data-mining and association rules in R.

Further filtering of the obtained item-sets can be done using lift and confidence values (as they measure association rule strength).

Here the mean value of lift and confidence are found to be 9.388 and 0.429 respectively.

```
rules[(rules.lift >= 9.388) & (rules.confidence >= 0.429)]
```

The item-sets are filtered such that only those having the lift and confidence values above the mean values are included.

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
4	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE GREEN)	0.033272	0.042817	0.018545	0.557377	13.017772	0.017120	2.162525
5	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE PINK)	0.042817	0.033272	0.018545	0.433121	13.017772	0.017120	1.705352
6	(ALARM CLOCK BAKELIKE GREEN)	(ALARM CLOCK BAKELIKE RED)	0.042817	0.047562	0.028744	0.671338	14.115027	0.026708	2.897922
7	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE GREEN)	0.047562	0.042817	0.028744	0.604358	14.115027	0.026708	2.419316
8	(ALARM CLOCK BAKELIKE IVORY)	(ALARM CLOCK BAKELIKE RED)	0.025363	0.047562	0.017018	0.670968	14.107251	0.015811	2.894665
10	(ALARM CLOCK BAKELIKE PINK)	(ALARM CLOCK BAKELIKE RED)	0.033272	0.047562	0.021436	0.644262	13.545763	0.019853	2.677361
11	(ALARM CLOCK BAKELIKE RED)	(ALARM CLOCK BAKELIKE PINK)	0.047562	0.033272	0.021436	0.450688	13.545763	0.019853	1.759890
12	(BAKING SET SPACEBOY DESIGN)	(BAKING SET 9 PIECE RETROSPOT)	0.023126	0.046580	0.016963	0.733491	15.746857	0.015886	3.577434
14	(CHARLOTTE BAG PINK POLKADOT)	(RED RETROSPOT CHARLOTTE BAG)	0.026508	0.040253	0.017508	0.660494	16.408528	0.016441	2.826891
15	(RED RETROSPOT CHARLOTTE BAG)	(CHARLOTTE BAG PINK POLKADOT)	0.040253	0.026508	0.017508	0.434959	16.408528	0.016441	1.722871
16	(CHARLOTTE BAG SUKI DESIGN)	(RED RETROSPOT CHARLOTTE BAG)	0.031853	0.040253	0.016199	0.508562	12.634105	0.014917	1.952934
19	(HOT WATER BOTTLE I AM SO POORLY)	(CHOCOLATE HOT WATER BOTTLE)	0.027272	0.037744	0.015545	0.570000	15.101705	0.014516	2.237804
20	(SPACEBOY LUNCH BOX)	(DOLLY GIRL LUNCH BOX)	0.038344	0.033599	0.023126	0.603129	17.950934	0.021838	2.435054
21	(DOLLY GIRL LUNCH BOX)	(SPACEBOY LUNCH BOX)	0.033599	0.038344	0.023126	0.688312	17.950934	0.021838	3.085313
22	(GARDENERS KNEELING PAD KEEP CALM)	(GARDENERS KNEELING PAD CUP OF TEA)	0.041235	0.034635	0.025254	0.612434	17.682461	0.023825	2.490839
23	(GARDENERS KNEELING PAD CUP OF TEA)	(GARDENERS KNEELING PAD KEEP CALM)	0.034635	0.041235	0.025254	0.729134	17.682461	0.023825	3.539627

In this way all the required item-sets can be found. Further fine-tuning of the item-sets to find the more probable ones can be done by increasing the threshold values of support and lift. We can store the found item-sets in a csv file for furtherance.

Summary:

We looked at the basic Association Mining Rule and its applications. Then we solved the problem of generating frequently bought item-sets by using Apriori algorithm. The process was applying Association Rules to these

generated item-set was looked upon. We also learned how to do this entire process in Python using Pandas and Mlxtend on a comparatively large-scale dataset.

References:

Apriori Algorithm : Know How to Find Frequent Itemsets | Edureka

Has it ever happened that you're out to buy something, and you end up buying a lot more than you planned? It's a...

www.edureka.co

Introduction to Market Basket Analysis in Python

There are many data analysis tools available to the python analyst and it can be challenging to know which ones to use...

pbpython.com

Machine Learning Market Basket Analysis Data Mining Apriori Data Science