



AoL Presentation

Group 3

2440076450 - Winson Allan Wijaya

2440069016 - Michael Christian

2440011135 - Christopher Reynard Julian

2440023103 - Nathanael Geoffrey Hanjaya

See our code!

<https://colab.research.google.com/drive/1jhv8oSzMtVOUMN9IxcOIftk0k8hIswp4?usp=sharing>

The background is a solid dark purple. On the right side, there are three large, organic, fluid shapes in shades of magenta and blue. One shape is at the top right, another is below it, and a third is at the bottom right. A smaller, similar shape is located near the bottom center.

01

Introduction

The background features a dark purple gradient with several large, organic, fluid shapes in lighter shades of purple and blue. A large, central, light blue circle is the most prominent element, containing the word "Background".

Background

Recommendation systems are systems that provide users with **suggestions** on items/objects to consume. These systems are already widely used in various industries, with the most prevalent applications being found in social medias such as YouTube, Twitch and TikTok. In these cases, the recommendation systems functions to assist both the customers and the content creators by helping the customers to not run out of content to enjoy, while promoting content creators to the larger audience.

These recommendation systems could also be used to **aid businesses**. Their nature of being a suggestion provider based on existing data means that they would be able to assist businesses in making decisions. The implementations of recommendation systems for businesses, however, is yet to be known to the extent of the ones being used in social services, such as the ones mentioned above. That's why, we aim to **socialize a way** on how recommendation systems could be used by businesses for **making decisions**.

Problem Definition

Promotions play a big role in businesses who sell products or services to customers. The ever-growing competition between businesses that are active in the same space makes the need for **successful marketing campaigns** becoming the highest they've ever been.

What we would like to achieve is to create the baseline of a system which could be used by businesses to conduct analysis on their transactions. Through the results of the conducted analysis, they could **formulate promotional campaigns** that would likely be successful.

The background features a dark purple gradient with several large, organic, fluid shapes in lighter shades of purple and blue. These shapes are positioned around the central text, creating a modern and artistic feel.

Solution

What we could do in this case is to conduct **Market Basket Analysis**. In Market Basket Analysis, we analyse for **patterns** that appear frequently in the transactions as through these patterns, we would be able to understand the **purchasing patterns** of our customers.

The background features a dark purple gradient with several large, organic, fluid shapes in lighter shades of purple and blue. These shapes are positioned around the central text, creating a modern and artistic feel.

Other Works

Apriori Algorithm

This method was proposed by Agrawal & Srikant back in 1994. In this method, frequent item sets are generated out of the original itemset. In this case, frequent item sets are item sets whose support value is more than the defined minimum support value. Below is how the Apriori Algorithm is done:

1. Initialize the **item set**, and variable **x** as **0**.
2. **Generate** all the item sets that contains **x+1 element(s)**.
3. Count the **support value** of the generated item sets.
4. **Remove** the item sets whose support value is less than the defined **minimum support value**.
5. If all the generated item sets are removed as their support values are not sufficient, proceed to the next step. Else, repeat the steps from step 2.
6. Generate the **association rules** based on the item sets and calculate their **confidence score**.
7. If the confidence score of the item sets are more than the set **threshold**, then those association-rules are the ones that we could use.

ECLAT Algorithm

The ECLAT (Equivalence Class Transformation) Algorithm proposed by Zaki, M. J in 2000 is based on the Apriori. In ECLAT, however, the format of the data that is used is **vertical**. This improves the **time complexity** of the algorithm as now, the **calculation of the support value** for each item set could be done **faster**. The algorithm itself is also based on using the **Depth First Search** algorithm, which is different from Apriori where Breath First Search algorithm served as the 'foundational' algorithm.

Transaction ID	Item List
001	T-shirt, pants, sweater
002	Pants, sweater

Table 1. Example of the horizontal data format of the dataset that is used in Apriori Algorithm.

Itemset	Item List (Transaction ID)
T-shirt	001
Pants	001, 002
Sweater	001, 002

Table 2. Example of the vertical data format of the dataset that is used in ECLAT Algorithm.

ECLAT Algorithm

The rest of the algorithm is still the same as Apriori's as in ECLAT, we also need to **generate item sets** with the highest number of items possible, from which we then generate the **association-rules**, calculate their **confidence scores**, and determine whether they are valid or not based on the calculated scores compared to the set confidence score threshold. Despite the improvements on the time complexity aspect, ECLAT is **not suitable** for use with **large datasets** as it uses more **memory** for **higher-order item sets**.

FP-Growth Algorithm

The FP-Growth (Frequency Pattern Growth) Algorithm was proposed by Han, J., Pei, J., & Yin, Y back in 2000. It differs greatly from both Apriori and ECLAT. Unlike the two latter algorithms, the FP-Growth Algorithm uses the **Trie (sorted tree)** data structure, that is called as the **FP-Tree** to be used in generating the frequent item sets.

FP-Growth Algorithm

1. Create a **list of 1-item sets** with their **support count** and **sort** them in a **descending** order.
2. Create the **FP-Tree**, with the **root** of it represented as a **null** node.
3. Look at the **first transaction** in the dataset. See which items are in the transaction and sort them based on their **support count** in a **descending** order. Take the item with the **biggest support count** and make it the **child of the root node**. Take the **next items** and make them as the child of the '**max**' **support count node** based on the sorted order. For example, let's say that there is Transaction **T1** with items **I1**, **I2**, and **I3** with the support count of **3**, **2**, and **1** respectively. In this case, take **I1** as the child of the **root node**, **I2** as the child of **I1**, and **I3** as the child of **I2**. Set the **count variable** of each of the child nodes as **1**.

FP-Growth Algorithm

4. Look at the next transaction, order the items in a descending order. Do the same process as before, but if a branch already exists, **follow that branch** while **incrementing the count variable of traversed node** by 1. However, if a direct branch does not exist, make a **new** one with the count variable of 1. Redo this step until **all the transactions** have been inserted into the FP-Tree.

5. Construct the **conditional table**, which contains the **items, conditional pattern base**, and the **conditional frequent pattern tree**. This table is sorted based on the **item's support count** in an **ascending** manner.

6. To fill the conditional pattern base column, we just need to write the **path** on how to reach the node written in the **items' column**. If we took an example used in step no. 3, we will have a row in the table with the value in the 'items' column as **I3**, and its conditional pattern base of **{I1, I2: 1}**. The '1' in it corresponds to the count variable of the I3 node at the end of the path.

FP-Growth Algorithm

7. To fill in the conditional FP-Tree column, count the **number of occurrences** of the items in the path in total. We then check if the total number of occurrences is as big as the **minimum support count** or not. If it is, then we write it down.

8. To find out the generated frequent patterns, we need to **combine** the ones written in the 'items' column with the ones in the 'conditional FP-tree column' along with their **total count**. We also need to make sure that the **subsets** of them are also written. For example, if the item is 'I1' with the conditional FP-Tree of {I2 : 2, I3: 2}, the frequent patterns will be {I1, I2: 2}, {I1, I3: 2}, and {I1, I2, I3: 2}.

The FP-Algorithm improves on the **time complexity** when compared to Apriori and ECLAT. However, it is relatively **difficult to develop** and could potentially be a **memory-hungry** application when the dataset is **large**, or when there are a variety of **distinct items** in the dataset.

The Data

The background is a solid dark purple color. It features several large, organic, fluid shapes in lighter shades of purple and blue. One large, roughly circular shape is centered in the upper half of the frame. Another elongated, bean-like shape is in the top right corner. A third, more complex organic shape is in the bottom left corner. The text is centered within the large central circle.

**Let's take a look at the
dataset...**

So, what's the dataset like?

Name

"Transactions from a bakery"

Author

Sulman Sarwar

Attributes

Date:

Tells the date when a transaction occurred.

Time:

Tells the time of day when the transaction occurred.

Transaction:

Functions as the unique identifier for each distinct transaction (Transaction ID).

Item:

Tells the item that is bought in that transaction.

Data Exploration

WHAT DID WE FIND ?

Number of Transactions

The dataset contains **21293** rows of transactions.

Unique Values

We found that the **'item'** column contains **95** unique items.

Invalid Values

In the **'item'** column, we found that one of the unique values is **'none'**. We assume that this refers to the cancellation of transactions.

WHAT DID WE FIND ?

Best Selling Items

We took a look at the top 20 best selling items, coffee was at the top of it.

Lowest Selling Items

We also took a look at the lowest selling ones. It turns out, there were some that was only sold once.

WHAT THE FINDINGS INDICATE

1. The dataset is large.
2. There are a lot of distinct items that are sold in the bakery shop.
3. Invalid values exist, and we should treat them like as we treat missing values.
4. We expect to find the majority of the best selling items (especially coffee) in one or more of the final, valid association-rules.
5. We shouldn't find the lowest selling items as a part of the generated association rules.

CODE SNIPPETS

Finding out the number of unique items sold in the bakery.

```
[ ] len(df['Item'].unique())
```

95

CODE SNIPPETS

Listing the unique items sold in the bakery.

```
▶ pd.set_option('display.max_rows', 100)  
   print(df['Item'].value_counts())  
   pd.reset_option('display.max_rows')
```

```
☞ coffee      5471  
   bread      3325  
   tea        1435  
   cake       1025  
   pastry      856  
   none        786  
   sandwich    771  
   medialuna   616  
   hot chocolate 590  
   cookies     540
```

CODE SNIPPETS

Seeing the number of transactions the dataset has.

```
print(df.value_counts)
```

	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
...
21288	2017-04-09	14:32:58	9682	coffee
21289	2017-04-09	14:32:58	9682	tea
21290	2017-04-09	14:57:06	9683	coffee
21291	2017-04-09	14:57:06	9683	pastry
21292	2017-04-09	15:04:24	9684	smoothies

[21293 rows x 4 columns]>

CODE SNIPPETS

Determining the top 20 best selling items and the top 20 least selling items

```
print("The 20 Best Selling Items:")
print(df['Item'].value_counts().sort_values(ascending=False).head(20))
print("=====")
print("The 20 Least Selling Items:")
print(df['Item'].value_counts().sort_values(ascending=True).head(20))
```

The 20 Best Selling Items:

coffee	5471
bread	3325
tea	1435
cake	1025
pastry	856
sandwich	771
medialuna	616

The 20 Least Selling Items:

polenta	1
chicken sand	1
the bart	1
adjustment	1
raw bars	1
gift voucher	1
olum & polenta	1
bacon	1
siblings	2
chimichurri oil	2

Data Pre-processing

So, what did we do to the dataset?

1

Handle the
invalid
values.

2

Remove 'Date'
and 'Time'
columns

3

Data
Aggregation

4

Data
Transformation
(Hot Encoding)

5

Data
Transformation
(Numerical ->
Boolean)

CODE SNIPPETS

Handling the invalid values

```
df = df.drop(df[df['Item'] == 'none'].index)
print(df.value_counts)
```

```
<bound method DataFrame.value_counts of
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
...      ...      ...      ...      ...
21288  2017-04-09  14:32:58    9682      coffee
21289  2017-04-09  14:32:58    9682      tea
21290  2017-04-09  14:57:06    9683      coffee
21291  2017-04-09  14:57:06    9683      pastry
21292  2017-04-09  15:04:24    9684  smoothies

[20507 rows x 4 columns]>
```

CODE SNIPPETS

Dropping the "Date" and "Time" Columns

```
[ ] df = df.drop('Date', axis = 1)
    df = df.drop('Time', axis = 1)
```

```
[ ] df.head()
```

	Transaction	Item
0	1	bread
1	2	scandinavian
2	2	scandinavian
3	3	hot chocolate
4	3	jam

CODE SNIPPETS

Data Aggregation

```
sum_df = df.groupby(['Transaction', 'Item']).size().reset_index(name='counter')  
sum_df.head()
```

	Transaction	Item	counter
0	1	bread	1
1	2	scandinavian	2
2	3	cookies	1
3	3	hot chocolate	1
4	3	jam	1

CODE SNIPPETS

Data Transformation (Hot Encoding)

```
[ ] final_df = sum_df.groupby(['Transaction', 'Item'])['counter'].sum().unstack().reset_index().fillna(0).set_index('Transaction')
    final_df.head()
```

[illegible]

CODE SNIPPETS

Data Transformation (Numerical -> Boolean)

```
def encoding_function(a):  
    if a <= 0:  
        return 0  
    else:  
        return 1  
  
final_df = final_df.applymap(encoding_function)  
final_df.head()
```

	Item	adjustment	afternoon with the baker	alfajores	argentina night	art tray	bacon	baguette	bakewell	bare popcorn	basket	...	the bart	tl nomi
Transaction														
1		0	0	0	0	0	0	0	0	0	0	...	0	
2		0	0	0	0	0	0	0	0	0	0	...	0	
3		0	0	0	0	0	0	0	0	0	0	...	0	
4		0	0	0	0	0	0	0	0	0	0	...	0	
5		0	0	0	0	0	0	0	0	0	0	...	0	

5 rows x 15 columns

The background features a dark purple gradient with several organic, flowing shapes in lighter shades of purple and blue. A small, solid blue circle is positioned in the upper left quadrant. A large, complex, multi-lobed shape in shades of purple and blue dominates the right side of the frame. Another smaller, elongated shape is visible in the bottom right corner.

03

Method

Here's our method, in steps...

1. Determine the rough **number of initial association rules** we would like to generate
2. Determine the minimum support value, which is done by:
 - a. **Set the initial value to 0.05** as the number of distinct items and the dataset itself is large.
 - b. Test the **rule generation**, if it still has not generated enough rules, decrease the **minimum support** by 0.005.
 - c. Redo **step b** until the desired output is reached.
3. Determine the **minimum confidence score**.
4. Filter the rules based on the **criterion on No. 3**, and sort them based on their **Lift score**.

So, here are the criteria we set.

1. We would like to generate around **100** initial rules, we then use this to determine the minimum support value.
2. We set the minimum confidence score to be **more than 50%**. This ensures that the rule will be more likely to happen than not.
3. We set the minimum lift score to be **1**.

Additional notes...

By using this method, we found that by using the minimum support value of **0.005**, the number of generated 'initial' association-rules to be **103**. Due to that, we decided to move forward with the minimum support value of 0.005.

CODE SNIPPETS

Adding minimum support value...

```
[ ] frequents = apriori(final_df, min_support = 0.005, use_colnames=True)
```

Generating the association rules

```
pd.set_option('display.max_rows', 1000)
mined_rules = association_rules(frequents, metric='lift', min_threshold=1)
mined_rules.head(1000)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(coffee)	(alfajores)	0.478394	0.036344	0.019651	0.041078	1.130235	0.002264	1.004936
1	(alfajores)	(coffee)	0.036344	0.478394	0.019651	0.540698	1.130235	0.002264	1.135648
2	(tea)	(alfajores)	0.142631	0.036344	0.006762	0.047407	1.304393	0.001578	1.011614
3	(alfajores)	(tea)	0.036344	0.142631	0.006762	0.186047	1.304393	0.001578	1.053339
4	(jam)	(bread)	0.015003	0.327205	0.005071	0.338028	1.033076	0.000162	1.016349
5	(bread)	(jam)	0.327205	0.015003	0.005071	0.015499	1.033076	0.000162	1.000504
6	(bread)	(pastry)	0.327205	0.086107	0.029160	0.089119	1.034977	0.000985	1.003306
7	(pastry)	(bread)	0.086107	0.327205	0.029160	0.338650	1.034977	0.000985	1.017305

CODE SNIPPETS

Filtering the rules with the additional confidence score constraint.

```
▶ final_rules = mined_rules[(mined_rules['lift'] >= 1) & (mined_rules['confidence'] > 0.5)]  
final_rules.sort_values('confidence', ascending = False, inplace = True)  
final_rules.head(100).reset_index()
```

04

Result and Analysis

Here's what the raw result looks like

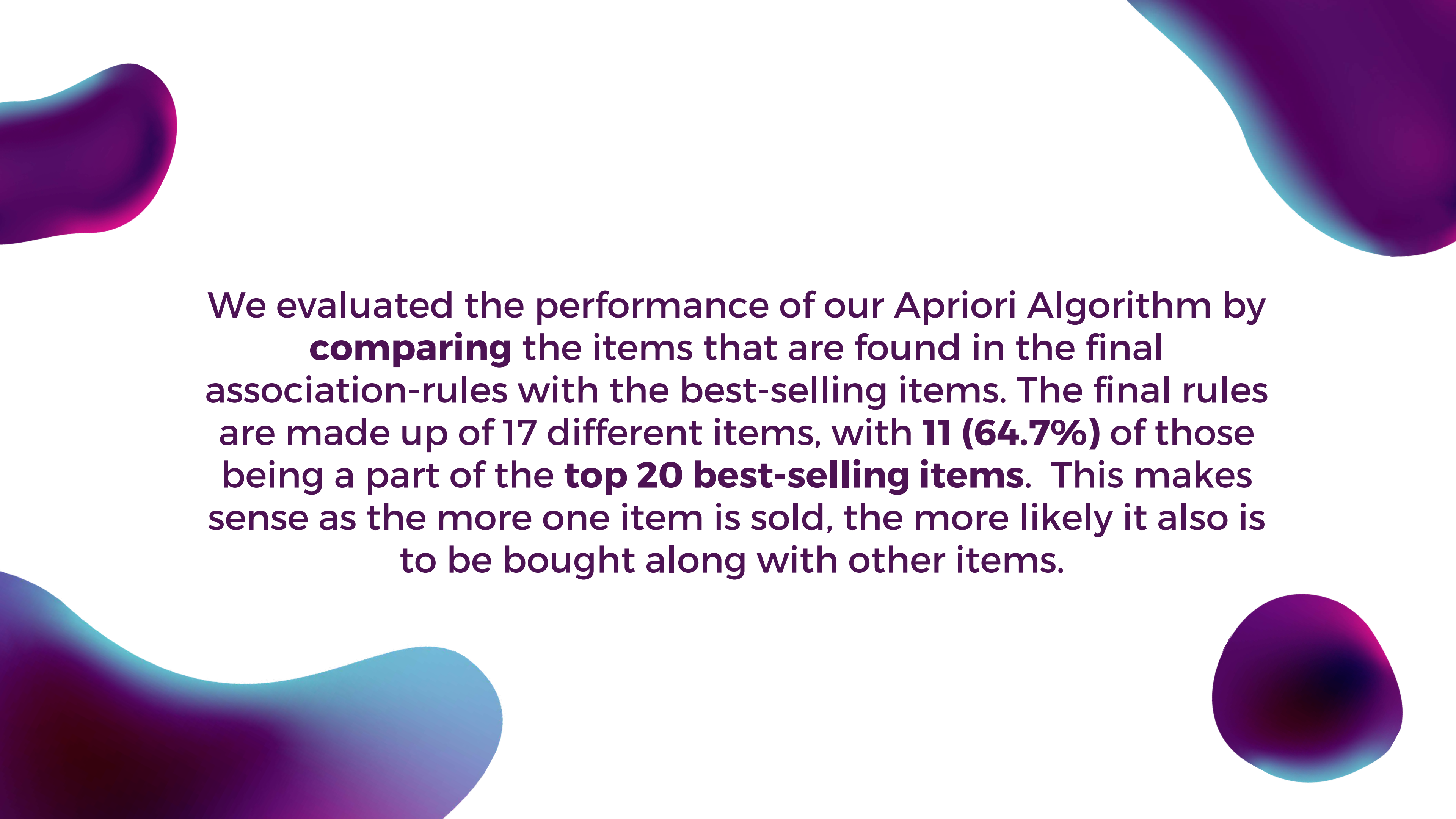
	index	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	33	(keeping it local)	(coffee)	0.006656	0.478394	0.005388	0.809524	1.692169	0.002204	2.738431
1	50	(toast)	(coffee)	0.033597	0.478394	0.023666	0.704403	1.472431	0.007593	1.764582
2	41	(salad)	(coffee)	0.010460	0.478394	0.006550	0.626263	1.309094	0.001547	1.395648
3	93	(hot chocolate, cake)	(coffee)	0.011410	0.478394	0.006867	0.601852	1.258067	0.001409	1.310080
4	47	(spanish brunch)	(coffee)	0.018172	0.478394	0.010882	0.598837	1.251766	0.002189	1.300235
5	35	(medialuna)	(coffee)	0.061807	0.478394	0.035182	0.569231	1.189878	0.005614	1.210871
6	39	(pastry)	(coffee)	0.086107	0.478394	0.047544	0.552147	1.154168	0.006351	1.164682
7	49	(tiffin)	(coffee)	0.015425	0.478394	0.008452	0.547945	1.145385	0.001073	1.153856
8	1	(alfajores)	(coffee)	0.036344	0.478394	0.019651	0.540698	1.130235	0.002264	1.135648
9	24	(hearty & seasonal)	(coffee)	0.010565	0.478394	0.005705	0.540000	1.128777	0.000651	1.133926
10	31	(juice)	(coffee)	0.038563	0.478394	0.020602	0.534247	1.116750	0.002154	1.119919
11	43	(sandwich)	(coffee)	0.071844	0.478394	0.038246	0.532353	1.112792	0.003877	1.115384
12	13	(cake)	(coffee)	0.103856	0.478394	0.054728	0.526958	1.101515	0.005044	1.102664
13	45	(scone)	(coffee)	0.034548	0.478394	0.018067	0.522936	1.093107	0.001539	1.093366
14	22	(cookies)	(coffee)	0.054411	0.478394	0.028209	0.518447	1.083723	0.002179	1.083174
15	26	(hot chocolate)	(coffee)	0.058320	0.478394	0.029583	0.507246	1.060311	0.001683	1.058553
16	29	(jammie dodgers)	(coffee)	0.013207	0.478394	0.006656	0.504000	1.053525	0.000338	1.051625

And here's our analysis of it...

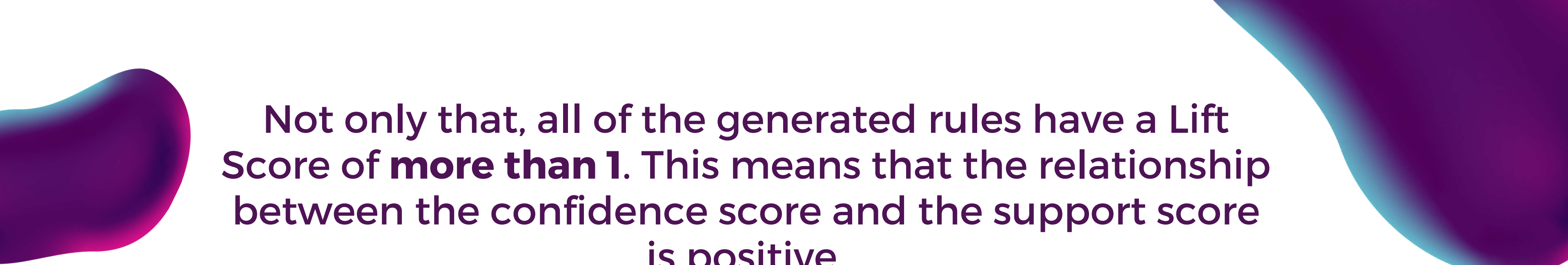
For the result, we managed to generate **17 rules**. These 17 rules are the rules that have met the criteria that we have set. The result shows that all the criteria-fulfilling rules have the item **'coffee'** as it's consequent, meaning that it might be a good idea to create promotional events that includes the item 'coffee' in it.

We felt that this makes sense as we generally need an **accompanying beverage** along with our food, with coffee being the go-to-choice for accompanying the sold items in this case.

Evaluation



We evaluated the performance of our Apriori Algorithm by **comparing** the items that are found in the final association-rules with the best-selling items. The final rules are made up of 17 different items, with **11 (64.7%)** of those being a part of the **top 20 best-selling items**. This makes sense as the more one item is sold, the more likely it also is to be bought along with other items.



Not only that, all of the generated rules have a Lift Score of **more than 1**. This means that the relationship between the confidence score and the support score is positive.

Additional notes...

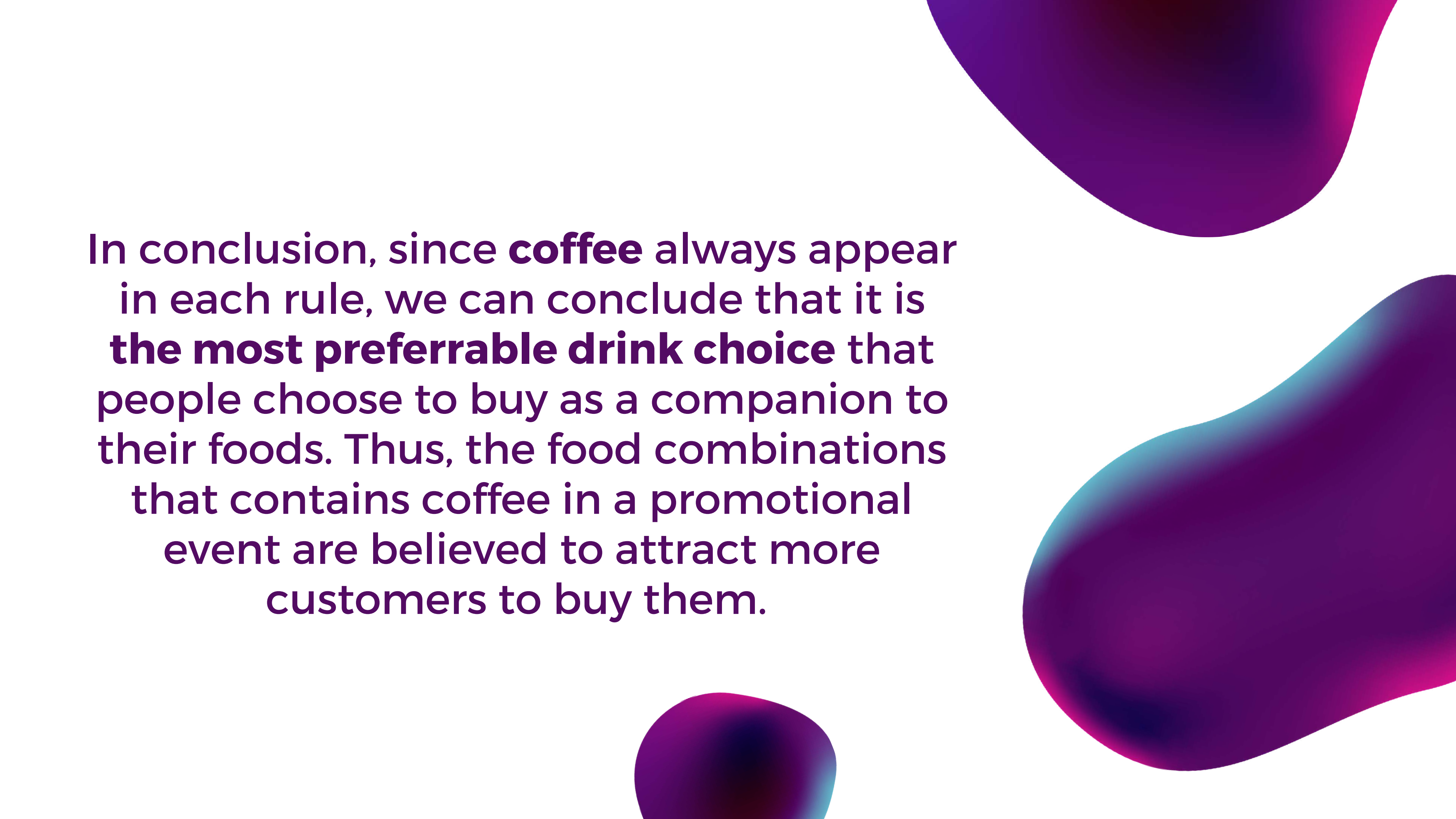
Out of the generated rules, the rules that have the smallest lift score has a lift score of around 1.05, and the rule that has the highest lift score has a lift score of around 1.69.



The background features a dark purple gradient with several large, organic, fluid shapes in shades of blue and light purple. One large shape is on the right side, and a smaller circular one is in the upper left.

06

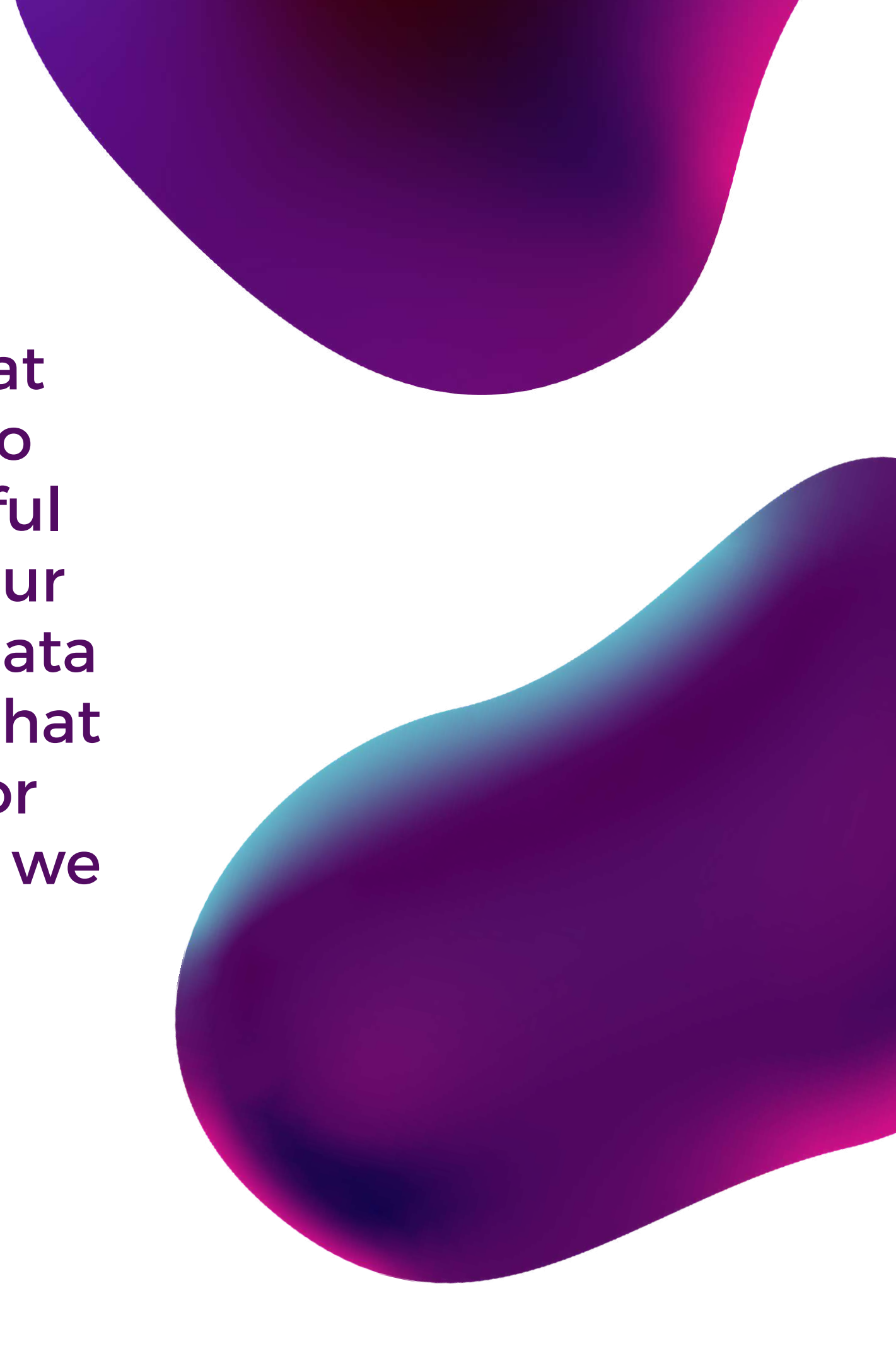
Conclusion

The background features three large, abstract, organic shapes in shades of purple and blue. One shape is in the top right corner, another is in the middle right, and a smaller one is at the bottom center. They have a soft, blurred appearance with some internal gradients.

In conclusion, since **coffee** always appear in each rule, we can conclude that it is **the most preferable drink choice** that people choose to buy as a companion to their foods. Thus, the food combinations that contains coffee in a promotional event are believed to attract more customers to buy them.

07

Implication



Based on our findings, we are confident that our method could be used by businesses to assist them in creating likely-to-be-successful promotional campaigns. This is proven by our hypothesis made when we conducted the ‘data exploration’ phase, where we hypothesized that the item ‘**coffee**’ would be present in one or more association rules that fulfils the criteria we have set, either as its antecedent or as its consequent.

The background is a deep purple color with several large, organic, wavy shapes in shades of blue and magenta. These shapes are positioned around the edges of the frame, creating a modern, abstract aesthetic.

THANK YOU !