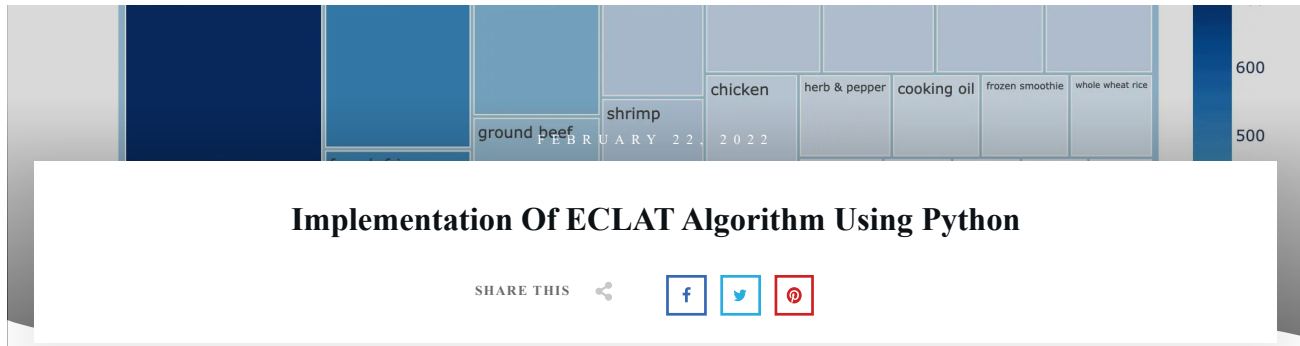




## Sponsored Links



By Bashir Alam

February 22, 2022

algorithm, machine-learning, python



Enjoy what I do? Consider [buying me a coffee](#) ☕

The ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) algorithm is a data mining algorithm for association rule mining designed to solve customer bucket analysis problems. The goal is to understand which products from the bucket are commonly bought together. This article will describe the ECAT algorithm implementation and association rules generation using Python, Jupyter Notebook, and Amazon SageMaker.

## Table Of Contents

- Overview of the ECLAT algorithm
- Implementation of ECLAT using Python
  - Importing and exploring the dataset
  - Visualizing the frequent items
  - Generating association rules
- Additional Learning Materials
- Summary

There are two ways to organize data in relational databases:

- **Row-oriented** – the traditional way of storing data that stores data records in rows and splits it by one or several columns
- **Column-oriented** (also known as **columnar** or **C-store**) – stores data by field, keeping all of the data associated with a field next to each other

Check out the [Row vs. Column Oriented Databases](#) article for more information on the topic.

The [Apriori](#) and [FP-growth](#) algorithms require data to be in the row format (sometimes called “horizontal format” in other blogs on the internet). In contrast, **the ECLAT algorithm is designed to deal with the data stored in the column-oriented format** (sometimes called “vertical format”).

Here’s an example of transactions datasets stored in both formats:

Horizontal format		Vertical format	
ID	Items bought	Items	ID_set
100	{f, a, c,}	a	{100,200}
200	{a,c}	b	{300, 400}
300	{b, f}	c	{100, 200, 400}
400	{b, c}	f	{100, 300}

## Overview Of The ECLAT Algorithm

ECLAT (Equivalence Class Clustering and bottom-up Lattice Traversal) algorithm is a data mining algorithm for association rule mining while solving customer’s basket analysis problem: the goal is to understand which products are often bought together. The ECLAT algorithm can’t be applied to the data represented in horizontal format, and you have to convert it into the vertical format before using it.

This vertical approach of the ECLAT algorithm makes it faster than the [Apriori](#) and [FP-growth](#) algorithms as it scans the database only once. The Apriori algorithm scans the database every single iteration, and the FP-growth algorithm does it two times two times.

Now, we can take sample data and review the ECLAT algorithm steps.

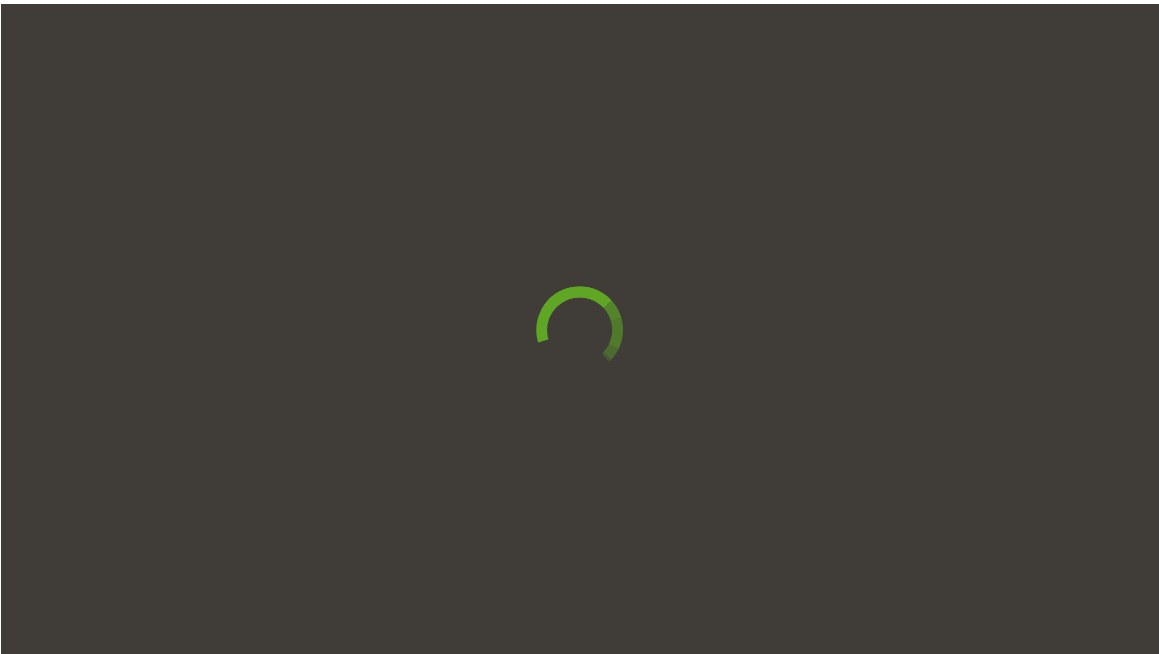
Let’s assume that we have the following database, and we set up the minimum support value (minimum number of item occurrences in the transactions list) to 2.

ID	Items bought
100	A, B, E
200	B, D
300	B, C
400	A, B, D
500	A, C
600	A, B, C

As we can see, the data is in horizontal format. So, we need to transform it:

Items	ID list
A	100, 400, 500, 600
B	100, 200, 300, 400, 600
C	300, 500, 600
D	200, 400,
E	100

As our minimum support value is 2, all items that appeared in only one transaction will be excluded from the dataset.



Working with AWS IAM in Python using Boto3

Items	ID list
A	100, 400, 500, 600
B	100, 200, 300, 400, 600
C	300, 500, 600
D	200, 400,
E	100

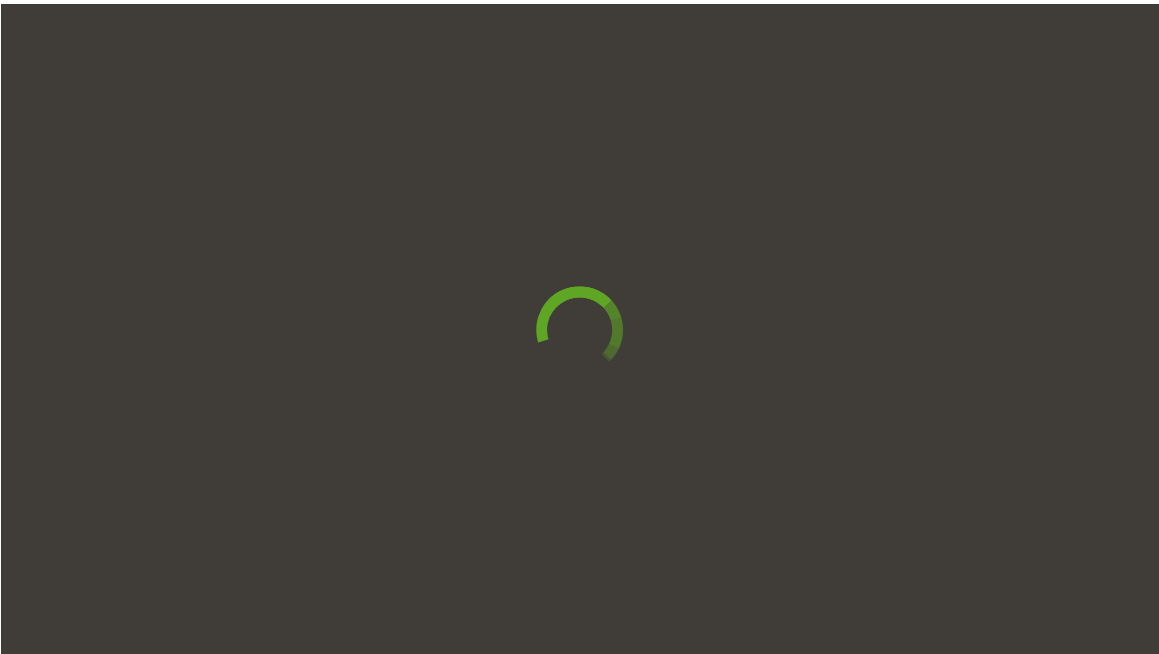
Items	ID list
A	100, 400, 500, 600
B	100, 200, 300, 400, 600
C	300, 500, 600
D	200, 400,

The next step is to create a list containing different sets of items’ combinations with the total set length equal to 2.

All possible combinations are the following:

Itemlist
A, B
A, C
A, D
B, C
B, D
C, D

Now, we need to associate all item combinations with corresponding transaction IDs. In our example, we’ll get the following table:



How to setup AWS Cloud9 IDE for Python

Sponsored Links

Itemlist	ID list
A, B	100, 400, 600
A, C	500, 600
A, D	400
B, C	300, 600
B, D	200, 400
C, D	--

We need to remove items combinations having support value less than the minimum support:

Itemlist	ID list
A, B	100, 400, 600
A, C	500, 600
A, D	400
B, C	300, 600
B, D	200, 400
C, D	--

Itemlist	ID list
A, B	100, 400, 600
A, C	500, 600
B, C	300, 600
B, D	200, 400

We repeat these steps as many times as needed to analyze itemsets of the required length. In our current example, when we create the product pairs of three products, we can find that only one group of items appeared in a single transaction.

Items	ID list
A, B, C	600
B, C, D	--

But one transaction is less than the minimum support value (two transactions), so we will generate association rules based on the previous step output.

Here's the recommended items list generated by the ECLAT algorithm based on our conditions and dataset:

Itemlist
A, B
A, C
B, C
B, D

## Implementation Of ECLAT Using Python

Let's implement the ECLAT algorithm using Python. We will use the [pyECLAT](#) module to implement the ECLAT algorithm. You can install the pyECLAT by running the following command in the cell of your Jupyter notebook:

### Installing requirements

```
%pip install pyECLAT
%pip install numpy
%pip install pandas
%pip install plotly
```

## Importing And Exploring The Dataset

We will use the built-in dataset available in the pyECLAT module. Let us first import the pyECLAT module and the built-in dataset.

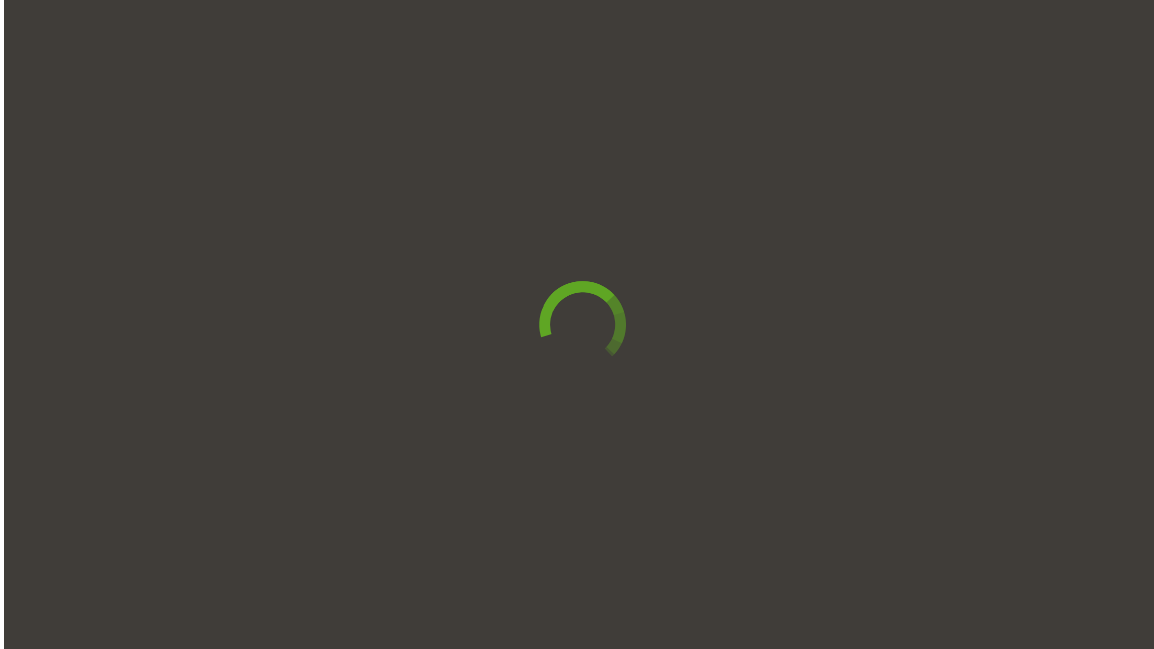
### Importing dataset

```
# importing dataset ( example 1 and example 2 are datasets in pyECLAT)
from pyECLAT import Example2
# storing the dataset in a variable
dataset = Example2().get()
# printing the dataset
dataset.head()
```

Output:

	0	1	2	3	4	5	6
0	shrimp	almonds	avocado	vegetables mix	green grapes	whole weat flour	yams
1	burgers	meatballs	eggs	NaN	NaN	NaN	NaN
2	chutney	NaN	NaN	NaN	NaN	NaN	NaN
3	turkey	avocado	NaN	NaN	NaN	NaN	NaN
4	mineral water	milk	energy bar	whole wheat rice	green tea	NaN	NaN

Each row represents a customer's purchase at a supermarket in this dataset. For example, in row 1, the customer purchased only burgers, meatballs, and eggs.



Python Programming - From Algorithm to Program Statements

Let's get more information about the dataset by printing more details.

#### Info method

```
# printing the info  
dataset.info()
```

Output:

The output shows that the dataset contains 3001 rows and 7 columns.

## Visualizing The Frequent Items

To visualize the frequent items, let's load the dataset to the ECLAT class and generate binary DataFrame:

### Numpy array of items

```
# importing the ECLAT module
from pyECLAT import ECLAT
# loading transactions DataFrame to ECLAT class
eclat = ECLAT(data=dataset)
# DataFrame of binary values
eclat.df_bin
```

Output:

In this binary dataset, every row represents a transaction. Columns are possible products that might appear in every transaction. Every cell contains one of two possible values:



#### Working with EC2 instances in Python using Boto3

- 0 – the product was not included in the transaction
- 1 – the transaction contains the product

Now, we need to count items for every column in the DataFrame:

#### Count items in each column

```
# count items in each column
items_total = eclat.df_bin.astype(int).sum(axis=0)
items_total
```

And it would be helpful to get the count of the items for every row in the DataFrame:

#### Count items in each row

```
# count items in each row
items_per_transaction = eclat.df_bin.astype(int).sum(axis=1)
items_per_transaction
```



Now, we can use these Series to visualize items distribution:

#### Frequent item list

```
import pandas as pd
# Loading items per column stats to the DataFrame
df = pd.DataFrame({'items': items_total.index, 'transactions': items_total.values})
# cloning pandas DataFrame for visualization purpose
df_table = df.sort_values("transactions", ascending=False)
# Top 5 most popular products/items
df_table.head(5).style.background_gradient(cmap='Blues')
```

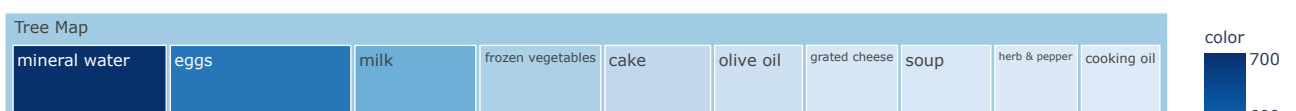
Output:

We can also visualize the frequently occurring items using TreeMap:

#### Frequent item list - Tree Map

```
# importing required module
import plotly.express as px
# to have a same origin
df_table["all"] = "Tree Map"
# creating tree map using plotly
fig = px.treemap(df_table.head(50), path=['all', "items"], values='transactions',
                 color=df_table["transactions"].head(50), hover_data=['items'],
                 color_continuous_scale='Blues',
                 )
# plotting the treemap
fig.show()
```

Output:





## Generating Association Rules

To generate association rules, we need to define:

- **Minimum support** – should be provided as a percentage of the overall items from the dataset
- **Minimum combinations** – the minimum amount of items in the transaction
- **Maximum combinations** – the maximum amount of items in the transaction

**Note:** the higher the value of the maximum combinations the longer the calculation will take.

### ECLAT algorithm

```
# the item should appear at least at 5% of transactions
min_support = 5/100
# start from transactions containing at least 2 items
min_combination = 2
# up to maximum items per transaction
max_combination = max(items_per_transaction)
rule_indices, rule_supports = eclat.fit(min_support=min_support,
                                       min_combination=min_combination,
                                       max_combination=max_combination,
                                       separator=' & ',
                                       verbose=True)
```

Output:

The `fit()` method of the `ECLAT` class returns:

- association rule indices
- association rule support values

We're interested only in association rule support values (the `rule_supports` dictionary):

#### Association rules DataFrame

```
import pandas as pd
result = pd.DataFrame(rule_supports.items(), columns=['Item', 'Support'])
result.sort_values(by=['Support'], ascending=False)
```

We found that mineral water and spaghetti are commonly purchased by customers based on the transaction data in our dataset and the minimum support value we've provided.

## Additional Learning Materials

From our point of view, these are the best hands-on online courses related to Machine Learning and Deep Learning available on the market right now:

Related Posts



Summary


Andrei Maksimov | 02/16/2023

**How To Get A List Of All AWS Services**  
Apriori, FP-tree, and ECLAT are the most famous association rules-generating algorithms. This article covers the ECLAT algorithm implementation and association rules generation using Python, Jupyter Notebook, and Amazon SageMaker.



Andrei Maksimov | 02/08/2023

**AWS Shield – The Most Important Information**



**Bashir Alam**

[f](#) [t](#) [in](#)


Bashir Alam, majoring in Computer Science and having extensive knowledge of Python, Machine learning, and Data Science. I have been working with different organizations and companies along with my studies. I have solid knowledge and experience of working offline and online, in fact, I am more comfortable in working online.

I love to learn new technologies and skills and I believe I am smart enough to learn new technologies in a short period of time.



Andrei Maksimov | 02/07/2023

**How To Install AWS CLI – Windows, Linux, OS X**



Subscribe Now To Get The **Latest Updates!**

ABOUT HANDS-ON.CLOUD LLC

We're helping 65,000+ IT professionals worldwide monthly to overcome their daily challenges.




LEGAL

- Contact Us
- Terms and Conditions
- Cookie Policy
- Privacy Policy
- Disclaimer
- Sitemap

PAGES

Python Boto3 Tutorials

CONTACT

-  600 Broadway, Ste 200 #6771, Albany, New York, 12207, US
-  929-990-3387
-  [info@hands-on.cloud](mailto:info@hands-on.cloud)