Eclat

January 15, 2025

[ ]:
```
'''
   Eclat -->

   The ECLAT (Equivalence Class Clustering and Bottom-Up Lattice Traversal) algorithm
   is a frequent itemset mining algorithm widely used in association rule learning.
   Unlike the Apriori algorithm, which generates candidate itemsets level by level,
   ECLAT operates using a vertical data format and is typically faster in scenarios
   with dense datasets.
'''
```

[ ]:
```
'''
   Steps in ECLAT Algorithm -->

   Transform Dataset :
   Convert the dataset into a vertical format (item-TID).

   Generate Frequent Items :
   Identify items that satisfy the minimum support threshold.

   Combine Items :
   Use the intersection of TID lists to generate k-itemsets.

   Prune :
   Discard itemsets that do not meet the minimum support threshold.

   Repeat :
   Continue until no more frequent itemsets can be generated.

'''
```

[ ]:
```
'''
   Apriori vs Eclat -->
```

```
   Feature                          Apriori                                  ␣
↪ECLAT

   Data Format                |     Horizontal                          |    ␣
↪Vertical
   Search Strategy            |     BFS                                 |     DFS
   Efficiency                 |     Slower for dense data               |    ␣
↪Faster for dense data
   Pruning                    |     Apriori property                    |    ␣
↪TID list intersection
   Memory Use                 |     Scans dataset multiple times        |    ␣
↪Stores TID lists
'''
```

[ ]:
```
#   Eclat uses support so confidence and lift is removed in Apriori approach !
```

[1]:
```
#   Importing Libraries -->

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from apyori import apriori
```

[3]:
```
#   Importing Dataset -->

data = pd.read_csv('Data/Market_Basket_Optimisation.csv', header=None)
data.head(5)
```

[3]:

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

	5	6	7	8	9
0	whole weat flour	yams	cottage cheese	energy drink	tomato juice
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

	10	11	12	13	14	15
0	low fat yogurt	green tea	honey	salad	mineral water	salmon
1	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN

```
4              NaN        NaN    NaN    NaN            NaN     NaN

                  16               17       18         19
0  antioxydant juice  frozen smoothie  spinach  olive oil
1                NaN              NaN      NaN        NaN
2                NaN              NaN      NaN        NaN
3                NaN              NaN      NaN        NaN
4                NaN              NaN      NaN        NaN
```

[4]:
```
#   Creating Transaction List -->

transactions = []

for i in range(0,7501):
    transactions.append([str(data.values[i,j]) for j in range(0,20)])
```

[5]:
```
#   Apriori Rules -->

rules = apriori(transactions=transactions, min_support=0.003, min_confidence=0.
↪2, min_lift=3, min_length=2, max_length=2)
```

[6]:
```
#   Results -->

results = list(rules)
results
```

[6]:
```
[RelationRecord(items=frozenset({'chicken', 'light cream'}),
support=0.004532728969470737,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}),
items_add=frozenset({'chicken'}), confidence=0.29059829059829057,
lift=4.84395061728395)]),
 RelationRecord(items=frozenset({'escalope', 'mushroom cream sauce'}),
support=0.005732568990801226,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'mushroom cream
sauce'}), items_add=frozenset({'escalope'}), confidence=0.3006993006993007,
lift=3.790832696715049)]),
 RelationRecord(items=frozenset({'escalope', 'pasta'}),
support=0.005865884548726837,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}),
items_add=frozenset({'escalope'}), confidence=0.3728813559322034,
lift=4.700811850163794)]),
 RelationRecord(items=frozenset({'honey', 'fromage blanc'}),
support=0.003332888948140248,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'fromage blanc'}),
items_add=frozenset({'honey'}), confidence=0.2450980392156863,
lift=5.164270764485569)]),
 RelationRecord(items=frozenset({'ground beef', 'herb & pepper'}),
```

```
support=0.015997866951073192,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'herb & pepper'}),
items_add=frozenset({'ground beef'}), confidence=0.3234501347708895,
lift=3.2919938411349285)]),
 RelationRecord(items=frozenset({'ground beef', 'tomato sauce'}),
support=0.005332622317024397,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'tomato sauce'}),
items_add=frozenset({'ground beef'}), confidence=0.3773584905660377,
lift=3.840659481324083)]),
 RelationRecord(items=frozenset({'light cream', 'olive oil'}),
support=0.003199573390214638,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'light cream'}),
items_add=frozenset({'olive oil'}), confidence=0.20512820512820515,
lift=3.1147098515519573)]),
 RelationRecord(items=frozenset({'whole wheat pasta', 'olive oil'}),
support=0.007998933475536596,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'whole wheat
pasta'}), items_add=frozenset({'olive oil'}), confidence=0.2714932126696833,
lift=4.122410097642296)]),
 RelationRecord(items=frozenset({'shrimp', 'pasta'}),
support=0.005065991201173177,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}),
items_add=frozenset({'shrimp'}), confidence=0.3220338983050847,
lift=4.506672147735896)])]
```

[7]:
```
#   Putting Results into DataFrame -->

def inspect(results):
    lhs = [tuple(result[2][0][0])[0] for result in results]
    rhs = [tuple(result[2][0][1])[0] for result in results]
    supports =  [result[1] for result in results]
    return list(zip(lhs, rhs, supports))

resultDF = pd.DataFrame(inspect(results), columns=['Left Hand Side', 'Right␣
 ↪Hand Side', 'Support'])
resultDF.head(10)
```

[7]:

	Left Hand Side	Right Hand Side	Support
0	light cream	chicken	0.004533
1	mushroom cream sauce	escalope	0.005733
2	pasta	escalope	0.005866
3	fromage blanc	honey	0.003333
4	herb & pepper	ground beef	0.015998
5	tomato sauce	ground beef	0.005333
6	light cream	olive oil	0.003200
7	whole wheat pasta	olive oil	0.007999
8	pasta	shrimp	0.005066

[8]:
```
#   Display Results by Descending Suppport -->

resultDF.nlargest(n=10, columns='Support')
```

[8]:

	Left Hand Side	Right Hand Side	Support
4	herb & pepper	ground beef	0.015998
7	whole wheat pasta	olive oil	0.007999
2	pasta	escalope	0.005866
1	mushroom cream sauce	escalope	0.005733
5	tomato sauce	ground beef	0.005333
8	pasta	shrimp	0.005066
0	light cream	chicken	0.004533
3	fromage blanc	honey	0.003333
6	light cream	olive oil	0.003200