Apriori

January 15, 2025

[ ]:
```
'''
    Apriori ARL -->

    The Apriori algorithm is a classic data mining algorithm used for mining␣
 ↪frequent itemsets
    and generating association rules. It was proposed by R. Agrawal and R.␣
 ↪Srikant in 1994 and
    is widely used in market basket analysis to find relationships between␣
 ↪items in large datasets.
'''
```

[ ]:
```
'''
    Steps of the Apriori Algorithm -->

    1. Generate Frequent Itemsets :

    Begin with single items (1-itemsets).
    Use the Apriori property to prune itemsets that are not frequent
    (i.e., below the minimum support threshold).
    Generate candidate itemsets of size k+1 from frequent k-itemsets and
    repeat until no new frequent itemsets are found.

    2. Generate Association Rules :

    For each frequent itemset, generate rules of the form A->B, where
    A and B are disjoint subsets of the itemset.
    Calculate confidence for each rule and retain rules that meet the
    minimum confidence threshold.
'''
```

[ ]:
```
'''
    Example -->

    Dataset ->

    Transaction ID      |           Items Purchase
```

```
            1           |            Milk, Bread
            2           |         Bread, Butter, Jam
            3           |         Milk, Bread, Butter
            4           |      Milk, Bread, Butter, Jam

    Steps -->

    Set minsup = 50% (2 transactions).

    Generate Frequent 1-itemsets :

    {Milk}: Support = 3/4 = 75%
    {Bread}: Support = 4/4 = 100%
    {Butter}: Support = 3/4 = 75%
    {Jam}: Support = 2/4 = 50%

    Generate Frequent 2-itemsets :

    {Milk, Bread}: Support = 3/4 = 75%
    {Bread, Butter}: Support = 3/4 = 75%
    {Milk, Butter}: Support = 2/4 = 50%
    {Bread, Jam}: Support = 2/4 = 50%
    Remaining combinations are pruned.

    Generate Frequent 3-itemsets :

    {Milk, Bread, Butter}: Support = 2/4 = 50%

    Association Rule Generation :

    From {Milk, Bread, Butter} :
    Rule : Milk, Bread → Butter, Confidence = 2/3 = 66.7%
    Rule : Bread → Milk, Butter, Confidence = 2/4 = 50%
    Prune rules not meeting minconf.
'''
```

[ ]:
```
'''
    Advantages -->

    Simple and intuitive to implement.
    Uses the Apriori property to reduce computation by eliminating unlikely
    candidates early.
    Well-suited for market basket analysis.

    Limitations -->
```

```
    The algorithm generates many candidate itemsets, which can be␣
 ↪computationally
    expensive for large datasets.
    Requires multiple passes over the dataset, increasing time complexity.
    May generate too many infrequent itemsets in sparse datasets.
'''
```

[16]:
```
#   Importing Libraries -->

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

[8]:
```
#   Importing Dataset -->

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

[8]:

	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

[27]:
```
#   Creating Transaction List -->

transactions = []

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

[29]:
```
#   Apriori Rules -->

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

[30]:
```
#   Results -->

results = list(rules)
results
```

[30]:
```
[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({'mushroom cream sauce', 'escalope'}),
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({'pasta', 'escalope'}),
support=0.005865884548726837,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}),
items_add=frozenset({'escalope'}), confidence=0.3728813559322034,
lift=4.700811850163794)]),
 RelationRecord(items=frozenset({'fromage blanc', 'honey'}),
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({'olive oil', 'light cream'}),
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({'olive oil', 'whole wheat pasta'}),
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({'pasta', 'shrimp'}),
support=0.005065991201173177,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'pasta'}),
items_add=frozenset({'shrimp'}), confidence=0.3220338983050847,
lift=4.506672147735896)])]
```

[31]:
```
#   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]
    confidences = [result[2][0][2] for result in results]
    lifts = [result[2][0][3] for result in results]
    return list(zip(lhs, rhs, supports, confidences, lifts))

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

[31]:

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	light cream	chicken	0.004533	0.290598	4.843951
1	mushroom cream sauce	escalope	0.005733	0.300699	3.790833
2	pasta	escalope	0.005866	0.372881	4.700812
3	fromage blanc	honey	0.003333	0.245098	5.164271
4	herb & pepper	ground beef	0.015998	0.323450	3.291994
5	tomato sauce	ground beef	0.005333	0.377358	3.840659
6	light cream	olive oil	0.003200	0.205128	3.114710
7	whole wheat pasta	olive oil	0.007999	0.271493	4.122410
8	pasta	shrimp	0.005066	0.322034	4.506672

[32]:
```
#   Display Results by Descending Lift -->

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

[32]:

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
3	fromage blanc	honey	0.003333	0.245098	5.164271

0	light cream	chicken	0.004533	0.290598	4.843951
2	pasta	escalope	0.005866	0.372881	4.700812
8	pasta	shrimp	0.005066	0.322034	4.506672
7	whole wheat pasta	olive oil	0.007999	0.271493	4.122410
5	tomato sauce	ground beef	0.005333	0.377358	3.840659
1	mushroom cream sauce	escalope	0.005733	0.300699	3.790833
4	herb & pepper	ground beef	0.015998	0.323450	3.291994
6	light cream	olive oil	0.003200	0.205128	3.114710