**MARKET BASKET INSIGHTS**

**PHASE-1**

Problem definition :

Market basket analysis is a data mining technique used by retailers to increase sales by better

understanding customer purchasing patterns. It involves analyzing large data sets, such as purchase

history, to reveal product groupings, as well as products that are likely to be purchased together.

Why use of the market basket insights ?

Benefits of market baskeanalysisMarket basket analysis can increase sales and customer

satisfaction. Using data to determine that products are often purchased together, retailers can

optimize product placement, offer special deals and create new product bundles to encourage further

sales of these combinations.

Types of Market Basket Analysis

��Association Rule Mining: It involves identifying frequent item sets and generating association

rules that express the likelihood of one item being purchased with the purchase of another item. It is

used to identify the relationships or associations between items in a transactional dataset.

��Sequence Analysis: This type of market basket analysis focuses on the order in which items are

purchased in a transaction. It identifies frequent item sequences and generates sequential association

rules describing the likelihood of one item sequence being followed by another.

��Cluster Analysis: This type of market basket analysis involves grouping similar items or

transactions into clusters or segments based on their attributes. It helps to identify customer

segments with similar purchasing behaviors, which can inform product recommendations and

marketing strategies.

Design Thinking:

Design thinking is about solving problems – ‘wicked problems’- the type of problems that

market researchers tackle on a daily basis. They are ‘wicked’ in the sense of their comparison to

straight-forward problems (rather than in the moral sense), the term describes problems which are

indeterminate in that there is no definitive right or wrong solution and it may not be clear .

Applying Design Thinking:

the process of design thinking as being structured within five modes: Empathise; Define; Ideate;

Prototype; and Test. For researchers it is important to understand what each mode ‘looks’ like and

how it can be applied to market research .

Empathise:

The importance of understanding the end user in market research might sound like a given, but to

empathise, to understand feelings and even experience them requires something more immersive .

Ideate:

In design thinking, the ideation stage is often referenced in relation to the design team generating

ideas, ‘going wide’ and getting creative – for market researchers, however, the participants are the

team. Here market researchers need to treat their ‘team’ in a way that inspires them to generate

options - Focus groups, online discussions, forum topics and collaborative tasks can all be utilised, but

the researcher must create an exciting environment where all possibilities are open.

Prototype:

Prototyping, allows users to experience a potential solution in way that is ‘safe’ and inexpensive

from the client’s point of view. Market researchers (and their clients) should get comfortable with a

‘fail fast’ approach – to quote Rikke Friis Dam and Teo Yu Siang of the Interaction Design Foundation,

“Design thinking has a bias towards action.”

Test:

Testing in a market research scenario, in best practice should involve observation of the users

interacting with the proposed solution, whether that’s previewing a new packaging design, testing a

sample product or taking a virtual tour of a new restaurant. There is likely to be more emphasis on

qualitative methodology, with some supporting quantitative rating questions.

Conclusion :

Market basket analysis may be used by more and more businesses to get relevant information about

associations and unspoken linkages. A predictive form of market basket analysis is gaining traction

across various industries in an effort to pinpoint sequence.

**PHASE-2**

INNOVATION:

��Advanced visualization techniques:

Advanced-Data Visualization (henceforth ADV) is the refined and sophisticated version of

visualisation techniques that uses machine learning and automated technologies to make more

analytical and comprehensive reports for all relevant stakeholders, make predictions, derive

hidden insights, and generate recommendations.

�� tools for analysis:

Google Charts, Tableau, Grafana, Chartist, FusionCharts, Datawrapper, Infogram, and

ChartBlocks.

��Three advanced data visualization techniques:

1. Line Charts :involves Creating a graph in which data is represented as a line or a set of data

points joined by a line.

2. Area chart

3. Pie Charts

4. Bar Charts

5. Gauges

6. Heat and Treemaps

7. 3D Charts

8. 3D Colum

Visualization tool is best for data analsist:

��Best Data Visualization Tools:

 Tableau. Tableau is a data visualization tool that can be used by data analysts, scientists

satisticians, etc. to visualize the data and get a clear opinion based on the data analysis. .

 Looker

 Zoho Analytics

 Sisense

 IBM Cognos Analytics

 Qlik Sense

 Domo

Microsoft Power BI

��insights to visualization:

insights in visualizations provide analytic insights that can help users to detect and validateany

important relationships and meaningful differences based on the data that is presented by the

visualization.

��supported visualization types for insights:

The following visualization types support insights:

•Area

•Bubble

•column

•Heatmap

•Line

•Line and column

•Map

•packed bubble

• pie

•scatter

•word cloud

•Tree map

**PHASE-3**

Introduction:

Basket analysis, also known as market basket analysis or

association rule learning, is a popular technique used in data

mining to discover relationships between items in large-scale

transaction datasets. This paper provides an in-depth analysis of

the theoretical features of basket analysis, with a focus on the

most critical steps in the process. In addition, R markdown code

examples are provided to demonstrate how these steps can be

implemented in practice. Basket analysis is a widely-used method

for identifying patterns and associations between items purchased

together in retail transactions. This technique has proven to be

invaluable for marketing, inventory management, and product

placement strategies. The Apriori algorithm and Eclat algorithm

are two prominent approaches for generating association rules in

basket analysis. This paper will outline the crucial steps in the

process and provide R markdown code examples to facilitate

understanding and implementation.

Market basket analysis :

Market basket analysis is a data mining technique used by

retailers to increase sales by better understanding customer

purchasing patterns. It involves analyzing large data sets, such as

purchase history, to reveal product groupings, as well as products that

are likely to be purchased together.

Examples of Market Basket:

Analysis Another example might be an online store examining

customer purchase data to see which goods are often purchased

together. The study may indicate that customers who buy laptops also

buy mouse pads, extra hard drives, and extended warranties.

Application of Market Basket Analysis

Market basket analysis has several applications in

different industries. Some of the applications are: Retail

Industry: Market basket analysis is widely used in the retail

industry to identify the relationship between different

products and how they are purchased together.

Market basket transaction:

In simple terms Basically, Market basket analysis in data

mining is to analyze the combination of products which been

bought together. This is a technique that gives the careful study of

purchases done by a customer in a supermarket. This concept

identifies the pattern of frequent purchas.

Data preprocessing and transaction encoding:

The first step in basket analysis is data preprocessing and

transaction encoding. This step involves converting raw

transaction data into a suitable format for analysis. This usually

involves transforming the data into a binary matrix, where each

row represents a transaction, and each column represents an

item. The value in each cell is 1 if the item is present in the

transaction and 0 if not.

# Load required libraries

library(arules)

## Loading required package: Matrix

##

## Attaching package: &#39;arules&#39;

## The following objects are masked from &#39;package:base&#39;:

##

## abbreviate, write

library(datasets)

# Load sample data

data(Groceries)

# Convert to transactions object

groceries\_trans &lt;- as(Groceries, &quot;transactions&quot;)

Generating frequent itemsets:

The second step is to generate frequent itemsets, which are sets

of items that occur together frequently in the dataset. This is

typically done using a user-defined support threshold. The

support of an itemset is the proportion of transactions that contain

the itemset. In the case of the Apriori algorithm, this step involves

iteratively finding frequent k-itemsets and pruning infrequent

itemsets.

# Define support threshold

support\_threshold &lt;- 0.01

# Generate frequent itemsets

frequent\_itemsets &lt;- apriori(groceries\_trans, parameter = list(support =

support\_threshold, target = &quot;frequent itemsets&quot;))

## Apriori

##

## Parameter specification:

## confidence minval smax arem aval originalSupport maxtime support minlen

## NA 0.1 1 none FALSE TRUE 5 0.01 1

## maxlen target ext

## 10 frequent itemsets TRUE

##

## Algorithmic control:

## filter tree heap memopt load sort verbose

## 0.1 TRUE TRUE FALSE TRUE 2 TRUE

##

## Absolute minimum support count: 98

##

## set item appearances ...[0 item(s)] done [0.00s].

## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].

## sorting and recoding items ... [88 item(s)] done [0.00s].

## creating transaction tree ... done [0.00s].

## checking subsets of size 1 2 3 4 done [0.00s].

## sorting transactions ... done [0.00s].

## writing ... [333 set(s)] done [0.00s].

## creating S4 object ... done [0.00s].

Generating association rules:

Once frequent itemsets have been identified, the next step

is to generate association rules. An association rule is an

expression of the form X =&gt; Y, where X and Y are disjoint

itemsets. The strength of an association rule is measured

using metrics such as confidence, lift, and leverage.

Confidence is the conditional probability of finding Y in a

transaction given that X is present, while lift measures how

much more likely Y is to be present in a transaction with X

compared to a random transaction.

# Define confidence threshold

confidence\_threshold &lt;- 0.5

# Generate association rules

association\_rules &lt;- apriori(groceries\_trans, parameter = list(support =

support\_threshold, confidence = confidence\_threshold, target = &quot;rules&quot;))

## Apriori

##

## Parameter specification:

## confidence minval smax arem aval originalSupport maxtime support minlen

## 0.5 0.1 1 none FALSE TRUE 5 0.01 1

## maxlen target ext

## 10 rules TRUE

##

## Algorithmic control:

## filter tree heap memopt load sort verbose

## 0.1 TRUE TRUE FALSE TRUE 2 TRUE

##

## Absolute minimum support count: 98

##

## set item appearances ...[0 item(s)] done [0.00s].

## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].

## sorting and recoding items ... [88 item(s)] done [0.00s].

## creating transaction tree ... done [0.00s].

## checking subsets of size 1 2 3 4 done [0.00s].

## writing ... [15 rule(s)] done [0.00s].

## creating S4 object ... done [0.00s].

Evaluating and selecting rules:

The final step in basket analysis is evaluating and selecting

association rules based on their quality metrics. Rules with high

lift and confidence values are usually of greater interest, as they

indicate strong relationships between itemsets. The choice of

rules to retain depends on the specific problem context and the

desired outcomes.

# Sort rules by lift

sorted\_rules &lt;- sort(association\_rules, by = &quot;lift&quot;, decreasing = TRUE)

# Inspect the top 5 rules

inspect(sorted\_rules[1:5])

## lhs rhs support

## [1] {citrus fruit, root vegetables} =&gt; {other vegetables} 0.01037112

## [2] {tropical fruit, root vegetables} =&gt; {other vegetables} 0.01230300

## [3] {root vegetables, rolls/buns} =&gt; {other vegetables} 0.01220132

## [4] {root vegetables, yogurt} =&gt; {other vegetables} 0.01291307

## [5] {curd, yogurt} =&gt; {whole milk} 0.01006609

## confidence coverage lift count

## [1] 0.5862069 0.01769192 3.029608 102

## [2] 0.5845411 0.02104728 3.020999 121

## [3] 0.5020921 0.02430097 2.594890 120

## [4] 0.5000000 0.02582613 2.584078 127

## [5] 0.5823529 0.01728521 2.279125 99

Conclusion:

Basket analysis with association rules is a powerful

technique for discovering relationships between items in

large-scale transaction data. The key steps in this process

include data preprocessing and transaction encoding,

generating frequent itemsets, generating association rules,

and evaluating and selecting rules.

**PHASE-4**

**Market Basket Analysis of Store Data**

Dataset Description

 Different products given 7500 transactions over the course of a week at a French retail store.

 We have library(apyori) to calculate the association rule using Apriori.

Import the Library

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from apyori import apriori

Read data and Display

store\_data = pd.read\_csv(&quot;store\_data.csv&quot;, header=None)

display(store\_data.head())

print(store\_data.shape)

0 1 2 3 4 5 6 7 8 9 1

0

1

1

1

2

1

3

1

4

1

5

1

6

1

7

1

8

1

9

0

s

h

r

i

m

p

a

l

m

o

n

d

s

a

v

t

a

N

a

N

N

a

N

N

a

N

N

a

N

(7501, 20)

Preprocessing on Data

 Here we need a data in form of list for Apriori Algorithm.

records = []

for i in range(1, 7501):

records.append([str(store\_data.values[i, j]) for j in range(0, 20)])

print(type(records))

&lt;class &#39;list&#39;&gt;

Apriori Algorithm

 Now time to apply algorithm on data.

 We have provide min\_support, min\_confidence, min\_lift, and min length of sample-set

for find rule.

Measure 1: Support.

This says how popular an itemset is, as measured by the proportion of transactions in which an

itemset appears. In Table 1 below, the support of {apple} is 4 out of 8, or 50%. Itemsets can also

contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%.

If you discover that sales of items beyond a certain proportion tend to have a significant impact on

your profits, you might consider using that proportion as your support threshold. You may then

identify itemsets with support values above this threshold as significant itemsets.

Measure 2: Confidence.

This says how likely item Y is purchased when item X is purchased, expressed as {X -&gt; Y}. This is

measured by the proportion of transactions with item X, in which item Y also appears. In Table 1, the

confidence of {apple -&gt; beer} is 3 out of 4, or 75%.

One drawback of the confidence measure is that it might misrepresent the importance of an

association. This is because it only accounts for how popular apples are, but not beers. If beers are

also very popular in general, there will be a higher chance that a transaction containing apples will

also contain beers, thus inflating the confidence measure. To account for the base popularity of both

constituent items, we use a third measure called lift.

Measure 3: Lift.

This says how likely item Y is purchased when item X is purchased, while controlling for how popular

item Y is. In Table 1, the lift of {apple -&gt; beer} is 1,which implies no association between items. A lift

value greater than 1 means that item Y is likely to be bought if item X is bought, while a value less

than 1 means that item Y is unlikely to be bought if item X is bought.

association\_rules = apriori(records, min\_support=0.0045, min\_confidence=0.2,

min\_lift=3, min\_length=2)

association\_results = list(association\_rules)

How many relation derived

print(&quot;There are {} Relation derived.&quot;.format(len(association\_results)))

There are 48 Relation derived.

Association Rules Derived

for i in range(0, len(association\_results)):

print(association\_results[i][0])

frozenset({&#39;light cream&#39;, &#39;chicken&#39;})

frozenset({&#39;escalope&#39;, &#39;mushroom cream sauce&#39;})

frozenset({&#39;escalope&#39;, &#39;pasta&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;ground beef&#39;})

frozenset({&#39;tomato sauce&#39;, &#39;ground beef&#39;})

frozenset({&#39;olive oil&#39;, &#39;whole wheat pasta&#39;})

frozenset({&#39;shrimp&#39;, &#39;pasta&#39;})

frozenset({&#39;nan&#39;, &#39;light cream&#39;, &#39;chicken&#39;})

frozenset({&#39;shrimp&#39;, &#39;chocolate&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;cooking oil&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;escalope&#39;, &#39;mushroom cream sauce&#39;, &#39;nan&#39;})

frozenset({&#39;escalope&#39;, &#39;pasta&#39;, &#39;nan&#39;})

frozenset({&#39;spaghetti&#39;, &#39;ground beef&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;milk&#39;, &#39;olive oil&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;shrimp&#39;, &#39;mineral water&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;spaghetti&#39;, &#39;olive oil&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;shrimp&#39;, &#39;spaghetti&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;spaghetti&#39;, &#39;frozen vegetables&#39;, &#39;tomatoes&#39;})

frozenset({&#39;spaghetti&#39;, &#39;ground beef&#39;, &#39;grated cheese&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;ground beef&#39;, &#39;mineral water&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;nan&#39;, &#39;ground beef&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;milk&#39;, &#39;ground beef&#39;, &#39;olive oil&#39;})

frozenset({&#39;nan&#39;, &#39;tomato sauce&#39;, &#39;ground beef&#39;})

frozenset({&#39;shrimp&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;milk&#39;, &#39;spaghetti&#39;, &#39;olive oil&#39;})

frozenset({&#39;soup&#39;, &#39;mineral water&#39;, &#39;olive oil&#39;})

frozenset({&#39;nan&#39;, &#39;olive oil&#39;, &#39;whole wheat pasta&#39;})

frozenset({&#39;shrimp&#39;, &#39;nan&#39;, &#39;pasta&#39;})

frozenset({&#39;spaghetti&#39;, &#39;pancakes&#39;, &#39;olive oil&#39;})

frozenset({&#39;shrimp&#39;, &#39;chocolate&#39;, &#39;frozen vegetables&#39;, &#39;nan&#39;})

frozenset({&#39;cooking oil&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;nan&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;milk&#39;, &#39;spaghetti&#39;, &#39;mineral water&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;milk&#39;, &#39;nan&#39;, &#39;olive oil&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;shrimp&#39;, &#39;nan&#39;, &#39;mineral water&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;nan&#39;, &#39;spaghetti&#39;, &#39;olive oil&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;shrimp&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;frozen vegetables&#39;})

frozenset({&#39;nan&#39;, &#39;spaghetti&#39;, &#39;frozen vegetables&#39;, &#39;tomatoes&#39;})

frozenset({&#39;nan&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;, &#39;grated cheese&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;nan&#39;, &#39;ground beef&#39;, &#39;mineral water&#39;})

frozenset({&#39;herb &amp; pepper&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;milk&#39;, &#39;nan&#39;, &#39;ground beef&#39;, &#39;olive oil&#39;})

frozenset({&#39;shrimp&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;ground beef&#39;})

frozenset({&#39;milk&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;olive oil&#39;})

frozenset({&#39;nan&#39;, &#39;soup&#39;, &#39;mineral water&#39;, &#39;olive oil&#39;})

frozenset({&#39;nan&#39;, &#39;spaghetti&#39;, &#39;pancakes&#39;, &#39;olive oil&#39;})

frozenset({&#39;milk&#39;, &#39;frozen vegetables&#39;, &#39;nan&#39;, &#39;spaghetti&#39;, &#39;mineral water&#39;})

Rules Generated

for item in association\_results:

# first index of the inner list

# Contains base item and add item

pair = item[0]

items = [x for x in pair]

print(&quot;Rule: &quot; + items[0] + &quot; -&gt; &quot; + items[1])

# second index of the inner list

print(&quot;Support: &quot; + str(item[1]))

# third index of the list located at 0th

# of the third index of the inner list

print(&quot;Confidence: &quot; + str(item[2][0][2]))

print(&quot;Lift: &quot; + str(item[2][0][3]))

print(&quot;=====================================&quot;)

Rule: light cream -&gt; chicken

Support: 0.004533333333333334

Confidence: 0.2905982905982906

Lift: 4.843304843304844

=====================================

Rule: escalope -&gt; mushroom cream sauce

Support: 0.005733333333333333

Confidence: 0.30069930069930073

Lift: 3.7903273197390845

=====================================

Rule: escalope -&gt; pasta

Support: 0.005866666666666667

Confidence: 0.37288135593220345

Lift: 4.700185158809287

=====================================

Rule: herb &amp; pepper -&gt; ground beef

Support: 0.016

Confidence: 0.3234501347708895

Lift: 3.2915549671393096

=====================================

Rule: tomato sauce -&gt; ground beef

Support: 0.005333333333333333

Confidence: 0.37735849056603776

Lift: 3.840147461662528

=====================================

Rule: olive oil -&gt; whole wheat pasta

Support: 0.008

Confidence: 0.2714932126696833

Lift: 4.130221288078346

=====================================

Rule: shrimp -&gt; pasta

Support: 0.005066666666666666

Confidence: 0.3220338983050848

Lift: 4.514493901473151

=====================================

Rule: nan -&gt; light cream

Support: 0.004533333333333334

Confidence: 0.2905982905982906

Lift: 4.843304843304844

=====================================

Rule: shrimp -&gt; chocolate

Support: 0.005333333333333333

Confidence: 0.23255813953488372

Lift: 3.260160834601174

=====================================

Rule: cooking oil -&gt; spaghetti

Support: 0.0048

Confidence: 0.5714285714285714

Lift: 3.281557646029315

=====================================

Rule: escalope -&gt; mushroom cream sauce

Support: 0.005733333333333333

Confidence: 0.30069930069930073

Lift: 3.7903273197390845

=====================================

Rule: escalope -&gt; pasta

Support: 0.005866666666666667

Confidence: 0.37288135593220345

Lift: 4.700185158809287

=====================================

Rule: spaghetti -&gt; ground beef

Support: 0.008666666666666666

Confidence: 0.3110047846889952

Lift: 3.164906221394116

=====================================

Rule: milk -&gt; olive oil

Support: 0.0048

Confidence: 0.20338983050847456

Lift: 3.094165778526489

=====================================

Rule: shrimp -&gt; mineral water

Support: 0.0072

Confidence: 0.3068181818181818

Lift: 3.2183725365543547

=====================================

Rule: spaghetti -&gt; olive oil

Support: 0.005733333333333333

Confidence: 0.20574162679425836

Lift: 3.1299436124887174

=====================================

Rule: shrimp -&gt; spaghetti

Support: 0.006

Confidence: 0.21531100478468898

Lift: 3.0183785717479763

=====================================

Rule: spaghetti -&gt; frozen vegetables

Support: 0.006666666666666667

Confidence: 0.23923444976076555

Lift: 3.497579674864993

=====================================

Rule: spaghetti -&gt; ground beef

Support: 0.005333333333333333

Confidence: 0.3225806451612903

Lift: 3.282706701098612

=====================================

Rule: herb &amp; pepper -&gt; ground beef

Support: 0.006666666666666667

Confidence: 0.390625

Lift: 3.975152645861601

=====================================

Rule: herb &amp; pepper -&gt; nan

Support: 0.016

Confidence: 0.3234501347708895

Lift: 3.2915549671393096

=====================================

Rule: herb &amp; pepper -&gt; spaghetti

Support: 0.0064

Confidence: 0.3934426229508197

Lift: 4.003825878061259

=====================================

Rule: milk -&gt; ground beef

Support: 0.004933333333333333

Confidence: 0.22424242424242424

Lift: 3.411395906324912

=====================================

Rule: nan -&gt; tomato sauce

Support: 0.005333333333333333

Confidence: 0.37735849056603776

Lift: 3.840147461662528

=====================================

Rule: shrimp -&gt; spaghetti

Support: 0.006

Confidence: 0.5232558139534884

Lift: 3.004914704939635

=====================================

Rule: milk -&gt; spaghetti

Support: 0.0072

Confidence: 0.20300751879699247

Lift: 3.0883496774390333

=====================================

Rule: soup -&gt; mineral water

Support: 0.0052

Confidence: 0.2254335260115607

Lift: 3.4295161157945335

=====================================

Rule: nan -&gt; olive oil

Support: 0.008

Confidence: 0.2714932126696833

Lift: 4.130221288078346

=====================================

Rule: shrimp -&gt; nan

Support: 0.005066666666666666

Confidence: 0.3220338983050848

Lift: 4.514493901473151

=====================================

Rule: spaghetti -&gt; pancakes

Support: 0.005066666666666666

Confidence: 0.20105820105820105

Lift: 3.0586947422647217

=====================================

Rule: shrimp -&gt; chocolate

Support: 0.005333333333333333

Confidence: 0.23255813953488372

Lift: 3.260160834601174

=====================================

Rule: cooking oil -&gt; nan

Support: 0.0048

Confidence: 0.5714285714285714

Lift: 3.281557646029315

=====================================

Rule: nan -&gt; spaghetti

Support: 0.008666666666666666

Confidence: 0.3110047846889952

Lift: 3.164906221394116

=====================================

Rule: milk -&gt; spaghetti

Support: 0.004533333333333334

Confidence: 0.28813559322033905

Lift: 3.0224013274860737

=====================================

Rule: milk -&gt; nan

Support: 0.0048

Confidence: 0.20338983050847456

Lift: 3.094165778526489

=====================================

Rule: shrimp -&gt; nan

Support: 0.0072

Confidence: 0.3068181818181818

Lift: 3.2183725365543547

=====================================

Rule: nan -&gt; spaghetti

Support: 0.005733333333333333

Confidence: 0.20574162679425836

Lift: 3.1299436124887174

=====================================

Rule: shrimp -&gt; nan

Support: 0.006

Confidence: 0.21531100478468898

Lift: 3.0183785717479763

=====================================

Rule: nan -&gt; spaghetti

Support: 0.006666666666666667

Confidence: 0.23923444976076555

Lift: 3.497579674864993

=====================================

Rule: nan -&gt; spaghetti

Support: 0.005333333333333333

Confidence: 0.3225806451612903

Lift: 3.282706701098612

=====================================

Rule: herb &amp; pepper -&gt; nan

Support: 0.006666666666666667

Confidence: 0.390625

Lift: 3.975152645861601

=====================================

Rule: herb &amp; pepper -&gt; nan

Support: 0.0064

Confidence: 0.3934426229508197

Lift: 4.003825878061259

=====================================

Rule: milk -&gt; nan

Support: 0.004933333333333333

Confidence: 0.22424242424242424

Lift: 3.411395906324912

=====================================

Rule: shrimp -&gt; nan

Support: 0.006

Confidence: 0.5232558139534884

Lift: 3.004914704939635

=====================================

Rule: milk -&gt; nan

Support: 0.0072

Confidence: 0.20300751879699247

Lift: 3.0883496774390333

=====================================

Rule: nan -&gt; soup

Support: 0.0052

Confidence: 0.2254335260115607

Lift: 3.4295161157945335

=====================================

Rule: nan -&gt; spaghetti

Support: 0.005066666666666666

Confidence: 0.20105820105820105

Lift: 3.0586947422647217

=====================================

Rule: milk -&gt; frozen vegetables

Support: 0.004533333333333334

Confidence: 0.28813559322033905

Lift: 3.0224013274860737

=====================================