**Case Study Assignment**

**Problem statement**

An organization wanted to mine association rules of frequently bought items from its stores and suggest some recommendations to its customers. As a data scientist, you are required to recognize patterns from the available data and evaluate efficacy of methods to obtain patterns.

Your activities should include

1. Preparing the dataset for analysis
2. investigating the relationships in the data set with visualization
3. Identify frequent patterns
4. Formulate association rules
5. Evaluate quality of rules

**Importing necessary libraries**

Import "os" library as it provides functions for interacting with the operating system

Import "numpy" library as it supports large, multi-dimensional arrays & matrices

Import "pandas" library as it supports data manipulation and analysis

**import os, numpy as np, pandas as pd**

Import "TransactionEncoder" - Encodes database transaction data in form of a Python list of lists into a NumPy array

**from mlxtend.preprocessing import TransactionEncoder**

Import "apriori" - algorithms for frequent itemset generation

Import "fpgrowth"- frequent pattern generation algorithm that inserts items into a pattern search tree

Import "fpmax" - variant of FP-Growth, which focuses on obtaining maximal itemsets

**from mlxtend.frequent\_patterns import apriori, fpmax, fpgrowth**

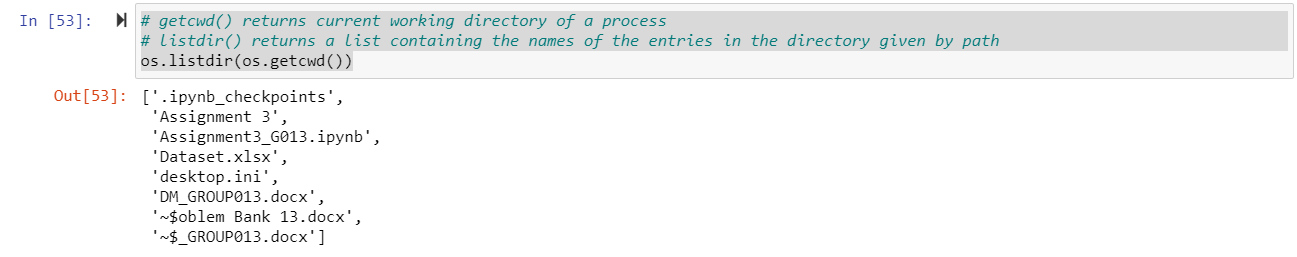
Import "matplotlib" - comprehensive library for creating static, animated, and interactive visualizations

**import matplotlib**

getcwd() returns current working directory of a process

listdir() returns a list containing the names of the entries in the directory given by path

**os.listdir(os.getcwd())**



**Perform exploratory data analysis**

**Reading data from Dataset.xlsx**

Need openpyxl as the engine to read the xlsx file

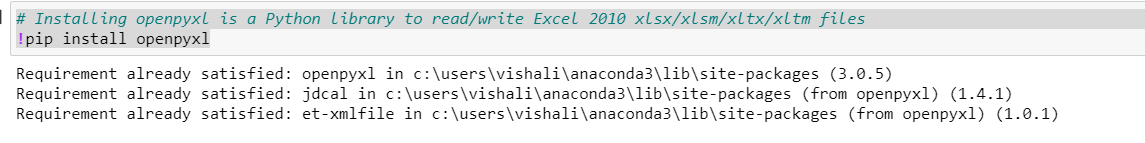
Input file - Dataset.xlsx where all the data are available

Reading an Excel file into a pandas DataFrame

**df = pd.read\_excel("Dataset.xlsx", engine="openpyxl")**

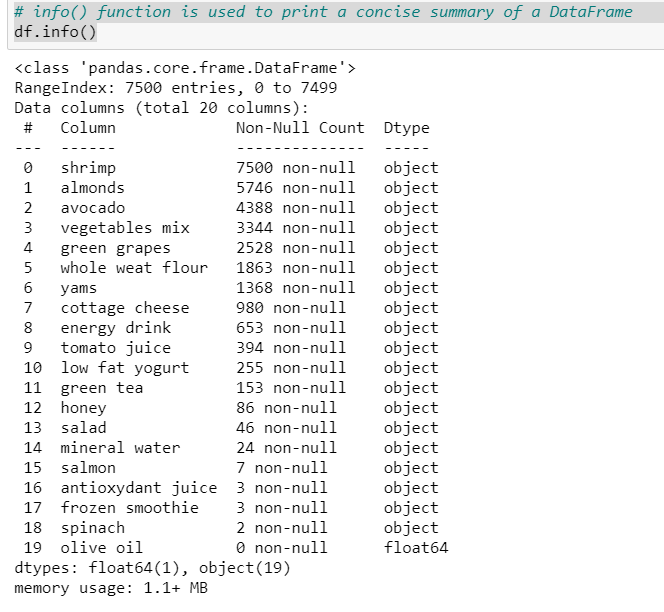
Installing openpyxl is a Python library to read/write Excel 2010 xlsx/xlsm/xltx/xltm files

**!pip install openpyxl**



info() function is used to print a concise summary of a DataFrame

**df.info()**



"display.max\_columns" sets the maximum number of columns displayed when a frame is pretty-printed

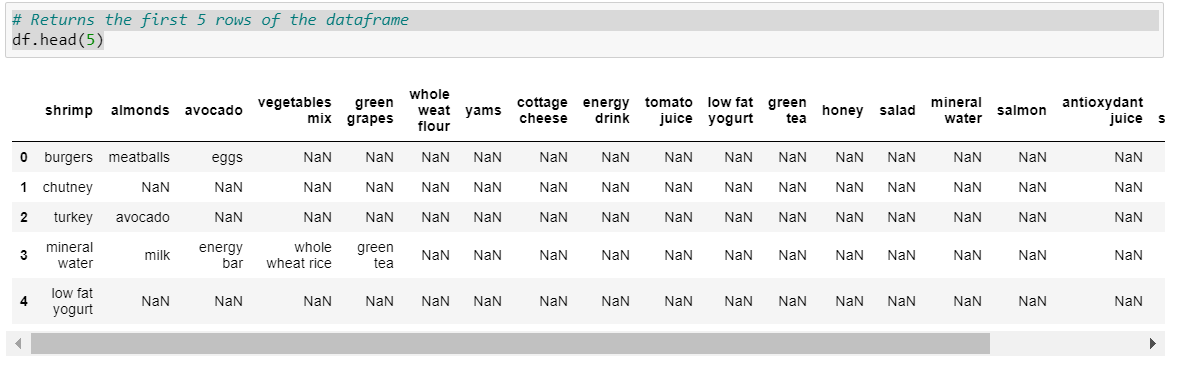
**pd.set\_option('display.max\_columns', None)**

Formatting every floating numbers

**pd.options.display.float\_format = '{:.5f}'.format**

Returns the first 5 rows of the dataframe

**df.head(5)**



**data = [ ]**

Conveys dataframe into list

**for rowIndex in range(df.shape[0]):**

**data.append(list(df.iloc[rowIndex][df.iloc[rowIndex].notnull()]))**

**data[:25]**



To find the length of the data

**len(data)**

"TransactionEncoder" - Encodes database transaction data in form of a Python list of lists into a NumPy array

Transform it into the right format via the TransactionEncoder

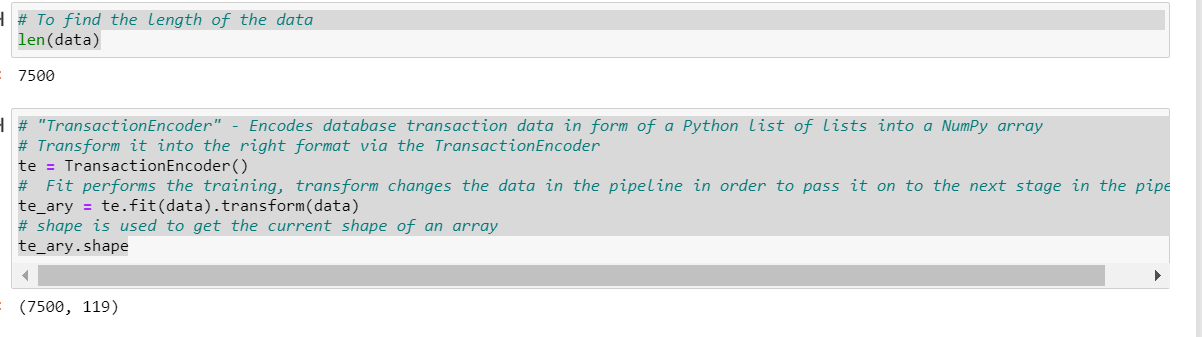
**te = TransactionEncoder()**

Fit performs the training, transform changes the data in the pipeline in order to pass it on to the next stage in the pipeline

**te\_ary = te.fit(data).transform(data)**

shape is used to get the current shape of an array

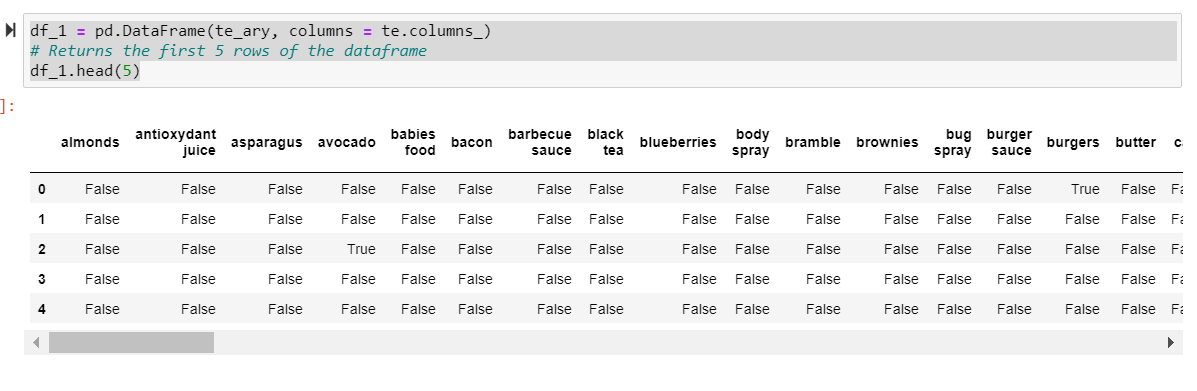
**te\_ary.shape**



**df\_1 = pd.DataFrame(te\_ary, columns = te.columns\_)**

Returns the first 5 rows of the dataframe

**df\_1.head(5)**



Return itemsets with at least 60% support:

**frequent\_itemsets = fpgrowth(df\_1, min\_support=0.6, use\_colnames=True)**

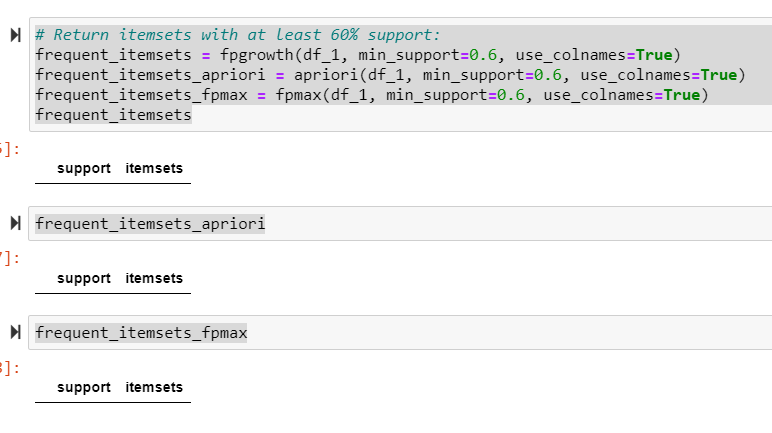
**frequent\_itemsets\_apriori = apriori(df\_1, min\_support=0.6, use\_colnames=True)**

**frequent\_itemsets\_fpmax = fpmax(df\_1, min\_support=0.6, use\_colnames=True)**

**frequent\_itemsets**

**frequent\_itemsets\_apriori**

**frequent\_itemsets\_fpmax**



## Iterate previous steps by varying parameters

## Return the itemsets with at least 30% support:

## frequent\_itemsets = fpgrowth(df\_1, min\_support=0.3, use\_colnames=True)

## frequent\_itemsets\_apriori = apriori(df\_1, min\_support=0.3, use\_colnames=True)

## frequent\_itemsets\_fpmax = fpmax(df\_1, min\_support=0.3, use\_colnames=True)

**frequent\_itemsets**

**frequent\_itemsets\_apriori**

**frequent\_itemsets\_fpmax**



## Return the itemsets with at least 10% support:

## frequent\_itemsets = fpgrowth(df\_1, min\_support=0.1, use\_colnames=True)

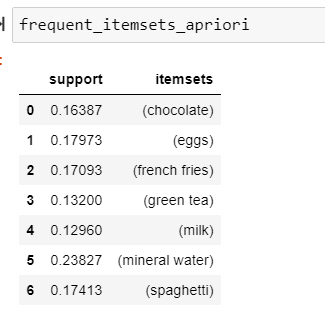
## frequent\_itemsets\_apriori = apriori(df\_1, min\_support=0.1, use\_colnames=True)

## frequent\_itemsets\_fpmax = fpmax(df\_1, min\_support=0.1, use\_colnames=True)

**frequent\_itemsets**



**frequent\_itemsets\_apriori**



**frequent\_itemsets\_fpmax**



Return the itemsets with at least 5% support:

**frequent\_itemsets = fpgrowth(df\_1, min\_support=0.05, use\_colnames=True)**

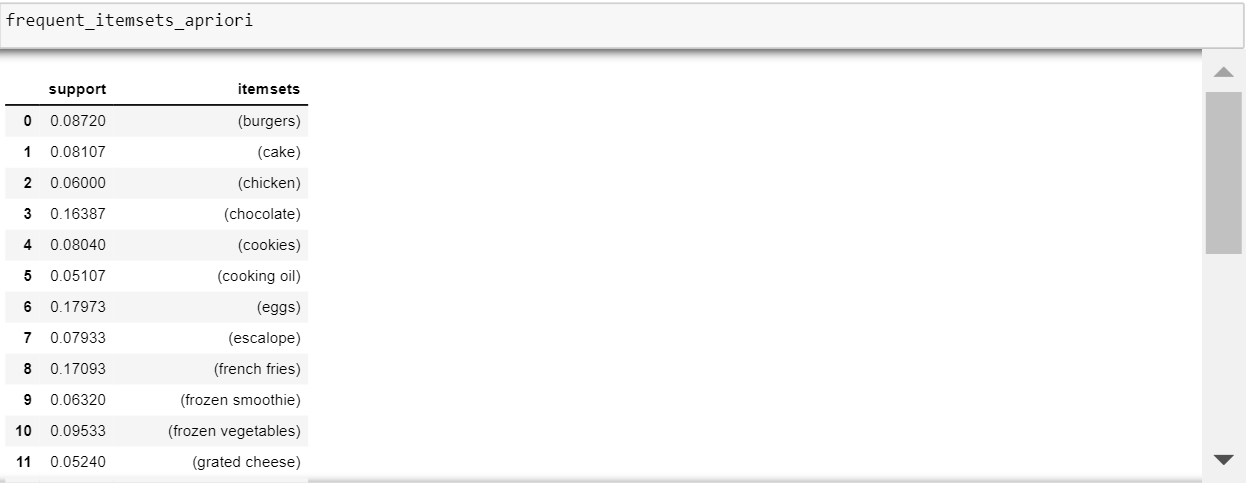
**frequent\_itemsets\_apriori = apriori(df\_1, min\_support=0.05, use\_colnames=True)**

**frequent\_itemsets\_fpmax = fpmax(df\_1, min\_support=0.05, use\_colnames=True)**

**frequent\_itemsets**



**frequent\_itemsets\_apriori**



**frequent\_itemsets\_fpmax**



## Formulate Association Rules from Frequent Itemsets

## Rule Generation and Selection Criteria

## from mlxtend.frequent\_patterns import association\_rules

## "association\_rules" - Function allows you to

## (1) specify your metric of interest and

## (2) the according threshold

## Currently implemented measures are confidence and lift

## Here, rules derived from the frequent itemsets only if the level of confidence is above the 5% threshold (min\_threshold=0.05):

## association\_rules(frequent\_itemsets, metric="confidence", min\_threshold=0.05)

## Here, rules derived from the frequent itemsets only if the level of lift is above the 12% threshold (min\_threshold=1.2):

## rules = association\_rules(frequent\_itemsets, metric="lift", min\_threshold=1.2)

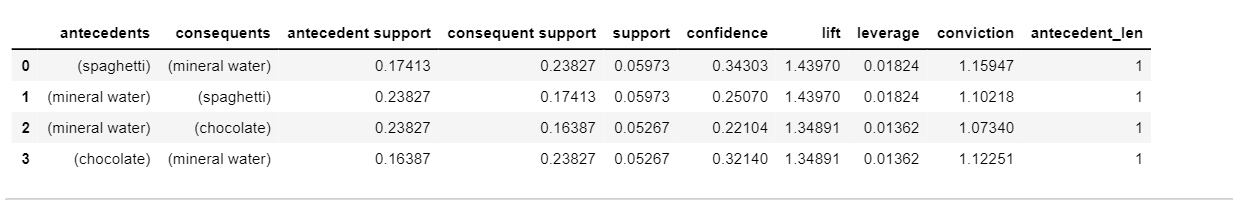
## rules

## 

Compute the antecedent length as follows:

**rules["antecedent\_len"] = rules["antecedents"].apply(lambda x: len(x))**

**rules**



we are only interested in rules that satisfy the following criteria:

at least 2 antecedents

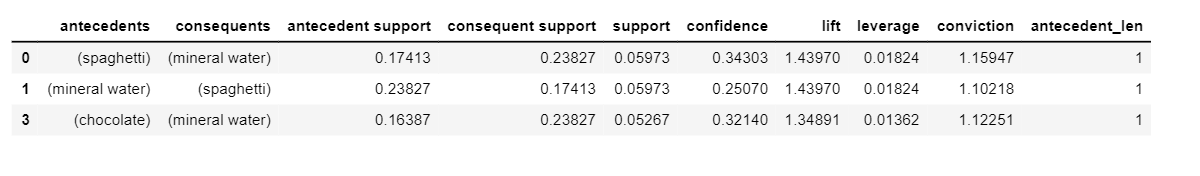
a confidence > 0.75

a lift score > 1.2

**rules[ (rules['antecedent\_len'] >= 1) &**

**(rules['confidence'] > 0.25) &**

**(rules['lift'] > 1.2) ]**



## Comparation of association rules

## We discovered the following 3 rules:

## Rule 1= {spaghetti} => {mineral water}

## Rule 2 = {mineral water} => {spaghetti}

## Rule 3 = {chocolate} => {mineral water}

## We can see confidence is high in rule 1 compared to rule 2 & 3.

## According to confidence measure, rule 1 has the highest value followed by Rule 3 & 2. Lift & Support are same in case of Rule 1 & 2. Rule 3 has the lowest value for lift & support measure compared to other Rules. Thus we say, Rule 1 to be the best.

## Conclusions based on rules:

## Keep Mineral water & Spaghetti in common area place in case of store and keep these items in the first page in case of online marketing such that it is very much visible for customer to buy when he visits the store/online page of the store. This is the recommendation provided to the customer.

## Provide combination of spaghetti & chocolate to increase the sale.

## Importance of discovered rules

## In data mining, association rules are useful for analyzing and predicting customer behavior. They play an important part in customer analytics, market basket analysis, product clustering, catalog design and store layout. Programmers use association rules to build programs capable of machine learning.