FOOD CHOICES

A case study report submitted in partial fulfillment of the subject

DATA MINING AND DATA WAREHOUSING

IN

B.TECH. III YEAR VI SEMESTER

OF

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P.S.D.SANNIHITHA

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GITAM INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

VISAKHAPATNAM

APRIL 2020

CONTENTS

Abstract

Basically, Data set is a collection of data. In the following dataset that i have considered contains various attributes to determine the food choices of the college students.we perform various techniques like preprocessing and analyze the association rules by using apriori and FP growth algorithms.the obtained data is classifies using ID3 algorithm(decision tree).So, Finally we perform clustering and abstract the required results from dataset.

1. INTRODUCTION

.

Food choice affects healthy ageing and ageing affects food choice. The antecedents of food choice may be remote and even intergenerational. Culture and ethnicity are enduring influences on food choice whether from within one's group or through the pressures of conformity to a majority in a minority culture. This is particularly relevant to indigenous and migrant peoples and where older people are marginalized or isolated for economic, health or societal reasons.

Different food cultures, from China, Japan and Korea in North East Asia to Sub-Saharan Africa, to Southern and Northern Europe, to Australasia may allow similar health outcomes. But food patterns for the aged are optimal where there is variety, especially of plant-derived food, regular consumption of pulses and even small quantities of animal-derived food such as eggs, dairy, lean meats and fish, especially where energy through-put is low.

The interplay of older people's food choices and meal patterns with gender, substance abuse (especially smoking) and activity (social, mental and physical) continues to be important with advancing years.

PROBLEM STATEMENT:

Cleaning the data, Analyzing and predicting behavior, to find homogeneous subgroups within the data such that data points in each cluster are as similar as possible according to a similarity measure such as euclidean-based distance or correlation-based distance.

OBJECTIVES

The objective of data mining is to produce a model that can be used to perform tasks such as classification, prediction or estimation, while the goal of descriptive data mining is to gain an understanding of the analysed system by uncovering patterns and relationships in large data sets.Automated revelation of unknown patterns. Data mining software tools just dive in databases and identify hidden patterns.

2. LANGUAGES and ASSOCIATED LIBRARIES/PACKAGES

Data processing:

```
import pandas as pd
import numpy as np
```

```
conda install ipython
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Apriori:

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

FPgrowth:

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

Decision Tree:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import LabelEncoder
from sklearn.tree import DecisionTreeClassifier
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn. tree import export_graphviz
!pip install pydotplus
import pydotplus
from sklearn.tree import DecisionTreeClassifier, export_graphviz
pip install graphviz
conda install graphviz
conda install -c conda-forge pydotplus
```

Clustering:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

from sklearn.model_selection import train_test_split
import seaborn as sns
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans

3. DATA DESCRIPTION:

I chose the dataset from Kaggle website, https://www.kaggle.com/borapajo/food-choices
This dataset includes information on food choices, nutrition, preferences, childhood favorites, and other information from college students.It contains information about GPA,Gender,breakfast,calories_chicken,calories_day,calories_scone,coffee comfort_food,comfort_food_reasons ,comfort_food_reasons_coded.The following dataset contains 125 records and 10 attributes.

4. METHODOLOGY AND DISCUSSION

4.1 Data Preprocessing:

It is a data mining technique that transforms raw data into an understandable format. Raw data(real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to preprocess data before sending through a model.

Steps in Data Preprocessing

1. Import libraries
2. Read data
3. Checking for missing values
4. Checking for categorical data
5. Standardize the data
6. PCA transformation
7. Data splitting

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
```

[2]:
```
dataset = pd.read_csv(r"C:\Users\PC\Desktop\casestudy\foodchoices.csv")
dataset
```

[2]:

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	comfort_food_reasons_coded
0	2.400	2	1	430	3	315	1	none	we dont have comfort	9.0
1	3.654	1	1	610	3	420	2	chocolate, chips, ice cream	Stress, bored, anger	1.0
2	3.300	1	1	720	4	420	2	frozen yogurt, pizza, fast food	stress, sadness	1.0
3	3.200	1	1	430	3	420	2	Pizza, Mac and cheese, ice cream	Boredom	2.0
4	3.500	1	1	720	2	420	2	Ice cream, chocolate, chips	Stress, boredom, cravings	1.0
...	...	...	...	...	...	...	...	...	...	...
120	3.500	1	1	610	4	420	2	wine. mac and cheese, pizza, ice cream	boredom and sadness	NaN
121	3.000	1	1	265	2	315	2	Pizza / Wings / Cheesecake	Loneliness / Homesick / Sadness	NaN
122	3.882	1	1	720	3	420	1	rice, potato, seaweed soup	sadness	NaN
123	3.000	2	1	720	4	420	1	Mac n Cheese, Lasagna, Pizza	happiness, they are some of my favorite foods	NaN
124	3.900	1	1	430	3	315	2	Chocolates, pizza, and Ritz.	hormones, Premenstrual syndrome.	NaN

125 rows × 10 columns

```
[3]: dataset.head(5)
```

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	comfort_food_reasons_coded
0	2.400	2	1	430	3	315	1	none	we dont have comfort	9.0
1	3.654	1	1	610	3	420	2	chocolate, chips, ice cream	Stress, bored, anger	1.0
2	3.300	1	1	720	4	420	2	frozen yogurt, pizza, fast food	stress, sadness	1.0
3	3.200	1	1	430	3	420	2	Pizza, Mac and cheese, ice cream	Boredom	2.0
4	3.500	1	1	720	2	420	2	Ice cream, chocolate, chips	Stress, boredom, cravings	1.0

```
[4]: dataset.isnull().sum()

[4]: GPA                           0
     Gender                        0
     breakfast                     0
     calories_chicken              0
     calories_day                  0
     calories_scone                0
     coffee                        0
     comfort_food                  1
     comfort_food_reasons          1
     comfort_food_reasons_coded   19
     dtype: int64
```

```
[5]: dataset.dropna(how='any')
```

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	comfort_food_reasons_coded
0	2.400	2	1	430	3	315	1	none	we dont have comfort	9.0
1	3.654	1	1	610	3	420	2	chocolate, chips, ice cream	Stress, bored, anger	1.0
2	3.300	1	1	720	4	420	2	frozen yogurt, pizza, fast food	stress, sadness	1.0
3	3.200	1	1	430	3	420	2	Pizza, Mac and cheese, ice cream	Boredom	2.0
4	3.500	1	1	720	2	420	2	Ice cream, chocolate, chips	Stress, boredom, cravings	1.0
...	...	...	...	...	...	...	...	...	...	...
101	3.400	1	1	610	4	420	2	macaroni and cheese, stuffed peppers, hamburge...	boredom, stress, mood swings	2.0
102	3.000	1	1	610	4	420	2	Pizza, mashed potatoes, spaghetti	Anger, sadness	3.0
103	3.700	1	1	610	3	420	2	dark chocolate, terra chips, reese's cups(dark...	Anxiousness, watching TV I desire "comfort food"	8.0
104	3.600	1	1	720	3	420	2	Chips, chocolate, ,mozzarella sticks	Boredom, sadness, anxiety	2.0
105	3.000	1	1	720	3	420	2	ice cream, chips, candy	Boredom, laziness, anger	2.0

105 rows × 10 columns

```
[6]: dataset.isnull().sum()

[6]: GPA                           0
     Gender                        0
     breakfast                     0
     calories_chicken              0
     calories_day                  0
     calories_scone                0
     coffee                        0
     comfort_food                  1
     comfort_food_reasons          1
     comfort_food_reasons_coded   19
     dtype: int64
```

```
[8]: dataset.dropna(inplace=True)
     dataset.isnull().sum()

[8]: GPA                           0
     Gender                        0
     breakfast                     0
     calories_chicken              0
     calories_day                  0
     calories_scone                0
     coffee                        0
     comfort_food                  0
     comfort_food_reasons          0
     comfort_food_reasons_coded    0
     dtype: int64
```

```
[9]: import pandas as pd

[10]: x=dataset.iloc[ : ,: -1].values

[11]: x

[11]: array([[2.4, 2, 1, 430, 3, 315, 1, 'none', 'we dont have comfort '],
       [3.654, 1, 1, 610, 3, 420, 2, 'chocolate, chips, ice cream',
        'Stress, bored, anger'],
       [3.3, 1, 1, 720, 4, 420, 2, 'frozen yogurt, pizza, fast food',
        'stress, sadness'],
       [3.2, 1, 1, 430, 3, 420, 2, 'Pizza, Mac and cheese, ice cream',
        'Boredom'],
       [3.5, 1, 1, 720, 2, 420, 2, 'Ice cream, chocolate, chips ',
        'Stress, boredom, cravings '],
       [2.25, 1, 1, 610, 3, 980, 2, 'Candy, brownies and soda.',
        "None, i don't eat comfort food. I just eat when i'm hungry."],
       [3.8, 2, 1, 610, 3, 420, 2,
        'Chocolate, ice cream, french fries, pretzels',
        'stress, boredom'],
       [3.3, 1, 1, 720, 3, 420, 1, 'Ice cream, cheeseburgers, chips.',
        'I eat comfort food when im stressed out from school(finals week), when I`m sad, or when i am dealing with personal family issues.'],
       [3.3, 1, 1, 430, 3, 420, 1, 'Donuts, ice cream, chips',
        'Boredom '],
       [3.3, 1, 1, 430, 3, 315, 2,
        'Mac and cheese, chocolate, and pasta ',
        'Stress, anger and sadness '],
       [3.5, 1, 1, 610, 3, 980, 2,
        'Pasta, grandma homemade chocolate cake anything homemade ',
        'Boredom '],
```



```
[3.904, 1, 1, 720, 4, 420, 2,
 'chocolate, pasta, soup, chips, popcorn',
 'sadness, stress, cold weather'],
[3.4, 2, 1, 430, 3, 420, 2, 'Cookies, popcorn, and chips',
 'Sadness, boredom, late night snack '],
[3.6, 1, 1, 610, 3, 420, 2, 'ice cream, cake, chocolate',
 'stress,  boredom, special occasions'],
[3.1, 2, 1, 610, 3, 420, 2,
 'Pizza, fruit, spaghetti, chicken and Potatoes  ',
 'Friends, environment and boredom'],
[3.0, 2, 2, 430, 3, 980, 2, 'cookies, donuts, candy bars',
 'boredom'],
[4.0, 1, 1, 265, 3, 420, 1, 'Saltfish, Candy and Kit Kat ',
 'Stress '],
[3.6, 2, 1, 430, 3, 980, 2, 'chips, cookies, ice cream',
 "I usually only eat comfort food when I'm bored, if i am doing something, i can go for hours without eating "],
[3.4, 1, 1, 720, 3, 980, 1, 'Chocolate, ice crea ',
 'Sadness, stress'],
[2.2, 2, 1, 430, 2, 420, 2, 'pizza, wings, Chinese',
 'boredom, sadness, hungry'],
[3.3, 2, 1, 610, 3, 980, 2, 'Fast food, pizza, subs',
 'happiness, satisfaction'],
[3.87, 2, 1, 610, 3, 315, 1, 'chocolate, sweets, ice cream',
 'Mostly boredom'],
[3.7, 2, 1, 610, 3, 420, 1, 'burgers, chips, cookies',
 'sadness, depression '],
[3.7, 2, 2, 610, 3, 420, 2, 'Chilli, soup, pot pie',
 'Stress and boredom '],
[3.9, 1, 1, 720, 2, 420, 2, 'Soup, pasta, brownies, cake',
 'A long day, not feeling well, winter '],
```

```
[2.8, 1, 2, 720, 3, 420, 2,
 'chocolate, ice cream/milkshake, cookies', 'boredom'],
[3.7, 2, 1, 610, 2, 420, 1,
 'Chips, ice cream, microwaveable foods ', 'Boredom, lazyniss '],
[3.0, 2, 1, 610, 4, 980, 2, 'Chicken fingers, pizza ', 'Boredom '],
[3.2, 2, 1, 610, 2, 420, 2, 'cookies, hot chocolate, beef jerky',
 'survival, bored'],
[3.5, 2, 1, 265, 2, 420, 2,
 'Tomato soup, pizza, Fritos, Meatball sub, Dr. Pepper',
 'Boredom, anger, drunkeness'],
[4.0, 1, 1, 720, 3, 420, 2,
 'cookies, mac-n-cheese, brownies, french fries, ',
 'stress, boredom, cold weather'],
[4.0, 2, 1, 610, 3, 420, 2, 'chips and dip, pepsi, ',
 'stres, boredom, and nighttime'],
[3.4, 2, 1, 610, 3, 315, 2,
 "Grandma's Chinese, Peruvian food from back home, and sushi",
 'Hunger and Boredom '],
[2.8, 1, 1, 720, 3, 420, 1,
 'Ice cream, cookies,  Chinese food, and chicken nuggets ',
 'boredom, sadness, and if it has a good taste. '],
[3.65, 1, 1, 610, 3, 420, 2, 'french fries, chips, ice cream',
 'boredom, stressed, sad'],
[3.0, 1, 1, 610, 2, 420, 2,
 'mac n cheese, peanut butter and banana sandwich, omelet',
 'Boredom usually'],
[3.7, 1, 1, 610, 3, 420, 2, 'pizza, doughnuts, mcdonalds ',
 'boredom'],
[3.4, 1, 1, 720, 4, 420, 2, 'chocolate, chips, candy', 'Stress'],
```

```
[3.89, 1, 1, 610, 3, 980, 2, 'chocolate, popcorn, ice cream',
 'boredom, stress'],
[3.0, 2, 1, 720, 3, 980, 2,
 "Candy\rPop\rChocolate \rChipotle \rMoe's ", 'No reasons '],
[3.4, 2, 1, 430, 3, 315, 1,
 'Pizza, Ice cream, fries, cereal, cookies  ',
 "Usually if I'm sad or depressed. "],
[2.9, 1, 1, 720, 4, 980, 2, 'Ice cream, chocolate, twizzlers ',
 'Tired '],
[3.6, 1, 1, 610, 3, 420, 2,
 'ice cream, cookie dough, cookies, cheese', 'Boredom!, sadness'],
[3.5, 1, 1, 430, 2, 980, 1,
 'ice cream, cereal, and salt and vinegar chips ',
 'All of the above; sadness, boredom and confusion '],
[3.2, 1, 1, 610, 4, 420, 2,
 'Potato chips, ice cream, chocolate, cookies',
 'Stress, boredom, craving'],
[3.605, 1, 1, 610, 3, 315, 2,
 'Mac and cheese, fried chicken, cornbread ', 'Hunger, boredom'],
[3.8, 2, 1, 430, 2, 420, 1, 'popcorn, chips, candy, & fries ',
 'sadness, boredom, & anger '],
[2.8, 2, 1, 430, 3, 980, 2,
 'Chex-mix, Wegmans cookies, Cheez-Its ',
 'Boredom, happiness, distraught '],
[3.5, 2, 2, 430, 3, 315, 1, 'pizza, ice cream, chips',
 'stressed, upset, or just craving a cheat meal'],
```

```
[3.83, 2, 1, 430, 3, 315, 2,
 'fried chicken. mashed potatoes, mac and cheese',
 'They taste better than other food. They are a pickme up. They are easy to make'],
[3.6, 2, 1, 720, 3, 420, 2, 'Popcorn, Chex Mix, Pizza',
 'Stress, boredom'],
[3.3, 2, 1, 610, 4, 980, 1, 'Burger', 'Lazy'],
[3.3, 2, 1, 610, 4, 420, 2, 'Pizza, chocolate, and ice cream ',
 'Boredom, sadness and anger '],
[3.292, 2, 1, 610, 3, 980, 2,
 'fries, chips, fried chicken, pizza, grapes', 'Boredom, sadness'],
[3.5, 2, 1, 610, 3, 420, 2,
 'peanut butter sandwich, pretzals, garlic bread',
 'stress, anger and boredom'],
[3.35, 1, 2, 610, 2, 315, 2, 'chips, dip, fries, pizza',
 'bored, stress'],
[3.8, 2, 1, 720, 4, 315, 2, 'Pizza, Ice Cream, Chicken Wings',
 'I usually only eat comfort foods when I am bored. I will also eat them when I am happy to celebrate and then when I am sad to comfort me.'],
[2.8, 1, 1, 610, 4, 980, 2,
 'Pizza chocolate chips bagels ice Capps ', 'Just cause '],
[3.5, 1, 1, 610, 3, 420, 2, 'Chocolate, ice cream, pasta',
 'Stress, boredom, sadness'],
[3.7, 1, 1, 610, 3, 420, 2,
 'Mac n Cheese. Chips and salsa. Ice cream. ',
 'Boredom. Celebration. '],
[3.6, 1, 1, 610, 4, 420, 2, 'peanut butter, dessets, pretzels. ',
 'Sadness, boredom, lonely.'],
[3.6, 1, 1, 610, 2, 980, 2,
 'Macaroons, truffles, peanut butter n chocolate ice cream',
 'I do not really eat "comfort food" but I guess sadness, special occasions, and anxiety '],
[3.9, 2, 1, 610, 4, 980, 2, 'ice cream, cookies, ice cream',
 'boredom, sadness'],
```



```
[2.6, 1, 1, 610, 4, 980, 2,
 'carrots and ranch, pretzels, dark chocolate ', 'sadness'],
[3.5, 1, 1, 610, 3, 420, 1,
 'cookies, nutella, ice cream, coffee, fruit ',
 'Bordem, happiness, sadness'],
[3.2, 1, 1, 610, 3, 315, 2, 'mac and cheese', 'boredom'],
[3.0, 1, 1, 720, 3, 420, 1, 'Chocolate, Popcorn, Icecream',
 'sadness'],
[3.6, 1, 1, 610, 2, 420, 1,
 'Ice cream, cake, mozzarella sticks, pierogies ', 'Boredom'],
[3.2, 1, 1, 430, 3, 315, 1,
 'Chips, Mac and cheese, pizza, French fries ',
 'Stress, sadness, bored '],
[3.67, 1, 2, 720, 4, 420, 2, 'Pizza, burritos, slim jims',
 'Boredom, stress, and it tastes good'],
[3.73, 1, 1, 610, 3, 980, 2,
 'Broccoli, spaghetti squash, quinoa, and grilled chicken',
 'Bad day, bored, sadness'],
[4.0, 1, 1, 720, 3, 420, 2, 'Chocolate, ice cream, cookie dough',
 'Boredom, being in your period, and long bus rides for softball'],
[3.1, 2, 2, 610, 3, 980, 2,
 'pizza, pretzels, fruit snacks, deli sandwhich',
 'boredom, anger, happy'],
[3.79, 2, 1, 720, 4, 420, 2, 'Chips, ice cream',
 'Boredom, stress'],
[3.0, 1, 1, 610, 3, 420, 2,
 'mac and cheese, potato soup, ice cream, chips and cheese',
 'sadness, stressed, boredom'],
[3.7, 1, 2, 610, 3, 420, 1,
 'chocolate, pizza, and mashed potatoes', 'boredom and stress'],
```

```
[3.1, 2, 2, 265, 2, 420, 1, 'Pizza cookies steak ',
 'Boredom comfort hunger '],
[3.0, 1, 1, 720, 3, 420, 2, 'chocolate, fruit, and ice cream',
 'stress, boredom'],
[3.9, 2, 1, 720, 3, 420, 2, 'Chips sweets popcorn', 'Boredom'],
[3.4, 1, 1, 430, 2, 420, 2,
 'Cookies, burgers, chicken noodle soup, ice cream',
 'happiness, hunger, sadness'],
[3.5, 1, 2, 610, 3, 420, 1, 'cake, French fries, chicken nuggets',
 'boredom, sadness'],
[3.7, 1, 1, 265, 3, 315, 2, 'pizza, ice cream, cookies',
 'boredom'],
[3.7, 1, 1, 430, 3, 420, 2, 'Mashed potatoes, pasta',
 'Boredom, sadness, or with friends '],
[3.83, 1, 1, 720, 3, 420, 2, 'Pasta dishes, Cheesecake, Pancakes',
 'Sadness, Loneliness, Boredom'],
[2.6, 1, 1, 265, 3, 315, 2, 'Ice cream, pizza, cookies',
 'Mostly Stress'],
[3.0, 1, 1, 610, 3, 420, 2,
 'Chinese food, moes, sponge candy, homemade lasagne ',
 'boredom, sadness '],
[3.2, 2, 1, 720, 3, 420, 1, 'pizza, pasta, mac and cheese',
 'when i am sad or craving'],
[3.5, 2, 2, 720, 4, 980, 2, 'Little Debbie snacks, donuts, pizza',
 'None'],
[3.2, 1, 1, 610, 3, 420, 2,
 'carrots, plantain chips, almonds, popcorn ',
 'stress, boredom, college as whole '],
[3.68, 2, 1, 720, 4, 420, 2, 'chips, ice cream, fruit snacks',
 'boredom'],
```

```
[3.8, 1, 2, 610, 2, 420, 2,
 'Macaroni and cheese, chicken noodle soup, pizza',
 'Boredom and stress'],
[3.3, 2, 2, 720, 3, 420, 2,
 'Chocolate, Chips, Ice cream, French Fires, Pizza',
 'Stress, sadness, boredom'],
[3.2, 2, 1, 720, 3, 420, 2,
 'Mac and cheese, lasagna, Chinese food ', 'Boredom, sadness'],
[3.75, 2, 1, 610, 3, 420, 2, 'candy, Chinese, mcdonalds',
 'laziness and hungover'],
[3.5, 2, 1, 265, 3, 420, 2, 'Doritos, mac and cheese, ice cream',
 'Boredom, hunger, snacking.'],
[3.92, 2, 1, 430, 3, 420, 2,
 'Ice cream, cake, pop, pizza, and milkshakes.',
 'Happiness, sadness, celebration.'],
[3.9, 1, 1, 720, 3, 420, 2,
 'Mac and Cheese, Pizza, Ice Cream and French Fries ',
 'Boredom, anger and just being hungry in general.'],
[3.9, 2, 1, 720, 3, 315, 1, 'Soup, pasta, cake',
 'Depression, comfort, accessibility '],
[3.2, 1, 1, 430, 4, 420, 1,
 'mac & cheese, frosted brownies, chicken nuggs',
 'they are yummy, my boyfriend sometimes makes me sad, boredom'],
[3.5, 1, 1, 610, 3, 3, 2, 'watermelon, grapes, ice cream',
 'Sad, bored, excited'],
[3.4, 1, 1, 610, 4, 420, 2,
 'macaroni and cheese, stuffed peppers, hamburgers, french fries',
 'boredom, stress, mood swings'],
[3.0, 1, 1, 610, 4, 420, 2, 'Pizza, mashed potatoes, spaghetti',
 'Anger, sadness'],
[3.7, 1, 1, 610, 3, 420, 2,
```

```
 [3.4, 1, 1, 610, 4, 420, 2,
  'macaroni and cheese, stuffed peppers, hamburgers, french fries',
  'boredom, stress, mood swings'],
 [3.0, 1, 1, 610, 4, 420, 2, 'Pizza, mashed potatoes, spaghetti',
  'Anger, sadness'],
 [3.7, 1, 1, 610, 3, 420, 2,
  "dark chocolate, terra chips, reese's cups(dark chocolate), and bread/crackers with cottage cheese",
  'Anxiousness, watching TV I desire "comfort food" '],
 [3.6, 1, 1, 720, 3, 420, 2,
  'Chips, chocolate, ,mozzarella sticks ',
  'Boredom, sadness, anxiety'],
 [3.0, 1, 1, 720, 3, 420, 2, 'ice cream, chips, candy',
  'Boredom, laziness, anger']], dtype=object)
```

[12]:
```
dataset.columns
```

[12]:
```
Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
       'comfort_food_reasons_coded'],
      dtype='object')
```

[13]:
```
from sklearn.preprocessing import LabelEncoder
```

[14]:
```
label_encoder = LabelEncoder()
```

[15]:
```
x[ : ,7] = label_encoder.fit_transform(x[ :,7])
x[ : ,8] = label_encoder.fit_transform(x[ :,8])
```

[16]:
```
x
```

[16]:
```
array([[2.4, 2, 1, 430, 3, 315, 1, 94, 90],
       [3.654, 1, 1, 610, 3, 420, 2, 68, 51],
       [3.3, 1, 1, 720, 4, 420, 2, 83, 86],
       [3.2, 1, 1, 430, 3, 420, 2, 47, 6],
       [3.5, 1, 1, 720, 2, 420, 2, 28, 54],
       [2.25, 1, 1, 610, 3, 980, 2, 3, 41],
       [3.8, 2, 1, 610, 3, 420, 2, 17, 83],
       [3.3, 1, 1, 720, 3, 420, 1, 27, 32],
       [3.3, 1, 1, 430, 3, 420, 1, 21, 7],
       [3.3, 1, 1, 430, 3, 315, 2, 34, 50],
       [3.5, 1, 1, 610, 3, 980, 2, 42, 7],
       [3.904, 1, 1, 720, 4, 420, 2, 71, 78],
       [3.4, 2, 1, 430, 3, 420, 2, 20, 44],
       [3.6, 1, 1, 610, 3, 420, 2, 84, 81],
       [3.1, 2, 1, 610, 3, 420, 2, 50, 27],
       [3.0, 2, 2, 430, 3, 980, 2, 75, 62],
       [4.0, 1, 1, 265, 3, 420, 1, 54, 48],
       [3.6, 2, 1, 430, 3, 980, 2, 64, 33],
       [3.4, 1, 1, 720, 3, 980, 1, 15, 46],
       [2.2, 2, 1, 430, 2, 420, 2, 102, 68],
       [3.3, 2, 1, 610, 3, 980, 2, 23, 73],
       [3.87, 2, 1, 610, 3, 315, 1, 74, 38],
       [3.7, 2, 1, 610, 3, 420, 1, 58, 77],
       [3.7, 2, 2, 610, 3, 420, 2, 6, 49],
       [3.9, 1, 1, 720, 2, 420, 2, 55, 0],
       [2.8, 1, 2, 720, 3, 420, 2, 70, 62],
       [3.7, 2, 1, 610, 2, 420, 1, 12, 18],
       [3.0, 2, 1, 610, 4, 980, 2, 5, 7],
```

```
[3.2, 2, 1, 610, 2, 420, 2, 76, 88],
[3.5, 2, 1, 265, 2, 420, 2, 57, 13],
[4.0, 1, 1, 720, 3, 420, 2, 77, 84],
[4.0, 2, 1, 610, 3, 420, 2, 63, 80],
[3.4, 2, 1, 610, 3, 315, 2, 24, 29],
[2.8, 1, 1, 720, 3, 420, 1, 30, 67],
[3.65, 1, 1, 610, 3, 420, 2, 80, 71],
[3.0, 1, 1, 610, 2, 420, 2, 92, 10],
[3.7, 1, 1, 610, 3, 420, 2, 97, 62],
[3.4, 1, 1, 720, 4, 420, 2, 67, 47],
[3.89, 1, 1, 610, 3, 980, 2, 73, 69],
[3.0, 2, 1, 720, 3, 980, 2, 2, 39],
[3.4, 2, 1, 430, 3, 315, 1, 46, 60],
[2.9, 1, 1, 720, 4, 980, 2, 29, 59],
[3.6, 1, 1, 610, 3, 420, 2, 87, 11],
[3.5, 1, 1, 430, 2, 980, 1, 85, 1],
[3.2, 1, 1, 610, 4, 420, 2, 53, 53],
[3.605, 1, 1, 610, 3, 315, 2, 35, 30],
[3.8, 2, 1, 430, 2, 420, 1, 103, 76],
[2.8, 2, 1, 430, 3, 980, 2, 4, 15],
[3.5, 2, 2, 430, 3, 315, 1, 98, 87],
[3.83, 2, 1, 430, 3, 315, 2, 81, 58],
[3.6, 2, 1, 720, 3, 420, 2, 52, 52],
[3.3, 2, 1, 610, 4, 980, 1, 1, 36],
[3.3, 2, 1, 610, 4, 420, 2, 49, 20],
[3.292, 2, 1, 610, 3, 980, 2, 82, 19],
[3.5, 2, 1, 610, 3, 420, 2, 95, 82],
[3.35, 1, 2, 610, 2, 315, 2, 65, 61],
[3.8, 2, 1, 720, 4, 315, 2, 45, 34],
```

```
[2.8, 1, 1, 610, 4, 980, 2, 43, 35],
[3.5, 1, 1, 610, 3, 420, 2, 18, 55],
[3.7, 1, 1, 610, 3, 420, 2, 37, 25],
[3.6, 1, 1, 610, 4, 420, 2, 96, 45],
[3.6, 1, 1, 610, 2, 980, 2, 39, 31],
[3.9, 2, 1, 610, 4, 980, 2, 88, 65],
[2.6, 1, 1, 610, 4, 980, 2, 61, 75],
[3.5, 1, 1, 610, 3, 420, 1, 78, 5],
[3.2, 1, 1, 610, 3, 315, 2, 90, 62],
[3.0, 1, 1, 720, 3, 420, 1, 14, 75],
[3.6, 1, 1, 610, 2, 420, 1, 25, 6],
[3.2, 1, 1, 430, 3, 315, 1, 9, 56],
[3.67, 1, 2, 720, 4, 420, 2, 48, 24],
[3.73, 1, 1, 610, 3, 980, 2, 0, 4],
[4.0, 1, 1, 720, 3, 420, 2, 16, 14],
[3.1, 2, 2, 610, 3, 980, 2, 101, 64],
[3.79, 2, 1, 720, 4, 420, 2, 11, 23],
[3.0, 1, 1, 610, 3, 420, 2, 91, 79],
[3.7, 1, 2, 610, 3, 420, 1, 72, 63],
[3.1, 2, 2, 265, 2, 420, 1, 44, 9],
[3.0, 1, 1, 720, 3, 420, 2, 69, 83],
[3.9, 2, 1, 720, 3, 420, 2, 8, 6],
[3.4, 1, 1, 430, 2, 420, 2, 19, 72],
[3.5, 1, 2, 610, 3, 420, 1, 59, 65],
[3.7, 1, 1, 265, 3, 315, 2, 99, 62],
[3.7, 1, 1, 430, 3, 420, 2, 40, 22],
[3.83, 1, 1, 720, 3, 420, 2, 41, 43],
[2.6, 1, 1, 265, 3, 315, 2, 31, 37],
[3.0, 1, 1, 610, 3, 420, 2, 7, 66],
[3.2, 2, 1, 720, 3, 420, 1, 100, 91],
[3.5, 2, 2, 720, 4, 980, 2, 32, 40],
```

```
       [3.2, 1, 1, 610, 3, 420, 2, 62, 85],
       [3.68, 2, 1, 720, 4, 420, 2, 66, 62],
       [3.8, 1, 2, 610, 2, 420, 2, 38, 8],
       [3.3, 2, 2, 720, 3, 420, 2, 13, 57],
       [3.2, 2, 1, 720, 3, 420, 2, 36, 19],
       [3.75, 2, 1, 610, 3, 420, 2, 60, 74],
       [3.5, 2, 1, 265, 3, 420, 2, 22, 16],
       [3.92, 2, 1, 430, 3, 420, 2, 26, 28],
       [3.9, 1, 1, 720, 3, 420, 2, 33, 12],
       [3.9, 2, 1, 720, 3, 315, 1, 56, 26],
       [3.2, 1, 1, 430, 4, 420, 1, 89, 89],
       [3.5, 1, 1, 610, 3, 3, 2, 104, 42],
       [3.4, 1, 1, 610, 4, 420, 2, 93, 70],
       [3.0, 1, 1, 610, 4, 420, 2, 51, 2],
       [3.7, 1, 1, 610, 3, 420, 2, 79, 3],
       [3.6, 1, 1, 720, 3, 420, 2, 10, 21],
       [3.0, 1, 1, 720, 3, 420, 2, 86, 17]], dtype=object)
```

```
[17]: Y = dataset.iloc[:, -1].values
      labelencoder_Y = LabelEncoder()
      Y = labelencoder_Y.fit_transform(Y)
```

```
[18]: Y
```

```
[18]: array([8, 0, 0, 1, 0, 3, 0, 0, 1, 0, 1, 2, 2, 0, 1, 1, 0, 1, 2, 1, 6, 1,
             2, 0, 5, 1, 1, 1, 1, 1, 0, 0, 3, 1, 1, 1, 1, 0, 1, 8, 2, 4, 1, 2,
             0, 3, 2, 1, 0, 4, 0, 4, 1, 1, 0, 1, 1, 8, 0, 1, 2, 2, 1, 2, 1, 1,
             2, 1, 0, 1, 1, 1, 1, 1, 2, 1, 1, 0, 1, 6, 1, 1, 1, 2, 0, 1, 2, 8,
             0, 1, 1, 0, 1, 4, 1, 6, 1, 2, 2, 2, 1, 2, 7, 1, 1], dtype=int64)
```

```
[19]: from sklearn.model_selection import train_test_split
      x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size=0.2)
```

```
[20]: from sklearn.model_selection import train_test_split
      x_train, x_test, Y_train, Y_test = train_test_split(x, Y, test_size=0.2)
```

```
[21]: from sklearn.preprocessing import StandardScaler
      sc_x = StandardScaler()
      x_train = sc_x.fit_transform(x_train)
      x_test = sc_x.transform(x_test)
```

```
[22]: sc_y = StandardScaler()
      Y_train = Y_train.reshape((len(Y_train), 1))
      Y_train = sc_y.fit_transform(Y_train)
      Y_train = Y_train.ravel()
```

```
[23]: Y_test=Y_test.reshape((len(Y_test), 1))
      Y_test=sc_y.transform(Y_test)
      Y_test = Y_test.ravel()
```

```
[24]: x_test
```

```
[24]: array([[ 1.23009185, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
              -0.39300048,  0.57735027, -0.57347702, -1.3275722 ],
             [ 0.22297457,  1.12706267, -0.34641016, -2.81559674, -0.12467575,
              -0.39300048,  0.57735027, -0.9310295 , -1.17501944],
             [ 0.22297457, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
               1.86340654,  0.57735027, -0.28093409, -1.51826315],
             [-1.53948068, -0.8872621 ,  2.88675135,  1.05325486, -0.12467575,
              -0.39300048,  0.57735027,  0.62919949,  0.57933725],
             [-0.53236339,  1.12706267, -0.34641016,  0.1179281 , -1.87013618,
              -0.39300048,  0.57735027,  0.82422811,  1.57093017],
             [-1.03592203, -0.8872621 , -0.34641016,  0.1179281 , -1.87013618,
              -0.39300048,  0.57735027,  1.34430444, -1.40384858],
             [ 0.72653321, -0.8872621 , -0.34641016, -2.81559674, -0.12467575,
              -0.8160768 ,  0.57735027,  1.57183783,  0.57933725],
             [ 0.47475389, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
              -0.39300048,  0.57735027,  1.47432352, -0.06901196],
             [-0.78414271,  1.12706267,  2.88675135, -2.81559674, -1.87013618,
              -0.39300048, -1.73205081, -0.21592455, -1.44198677],
             [-1.03592203, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
              -0.39300048,  0.57735027,  1.31179967,  1.22768647],
             [ 0.72653321, -0.8872621 ,  2.88675135,  0.1179281 , -0.12467575,
              -0.39300048, -1.73205081,  0.69420903,  0.61747544],
             [ 1.24016302, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
              -0.39300048,  0.57735027,  0.66170426,  1.18954828],
             [-0.53236339, -0.8872621 , -0.34641016, -1.4126066 , -0.12467575,
              -0.39300048,  0.57735027, -0.11841024, -1.55640134],
```

```
 [-0.78414271,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
  -0.39300048,  0.57735027, -0.02089592, -0.75549936],
 [ 0.97831253,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
  -0.39300048,  0.57735027, -1.09355335,  1.38023922],
 [ 0.47475389, -0.8872621 , -0.34641016,  0.1179281 , -1.87013618,
   1.86340654,  0.57735027, -0.3784484 , -0.60294661],
 [-0.02880475, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
  -0.39300048,  0.57735027,  0.53168517,  0.00726442],
 [ 0.72653321, -0.8872621 , -0.34641016, -1.4126066 , -0.12467575,
  -0.39300048,  0.57735027, -0.34594363, -0.94619031],
 [ 1.48187117, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
  -0.39300048,  0.57735027, -1.12605812, -1.25129582],
 [-0.78414271,  1.12706267,  2.88675135,  0.1179281 , -0.12467575,
   1.86340654,  0.57735027,  1.63684737,  0.65561363],
 [-0.28058407, -0.8872621 , -0.34641016, -1.4126066 , -0.12467575,
  -0.39300048, -1.73205081, -0.96353427, -1.51826315]])
```

```
[25]: x_train
```

```
[25]: array([[-1.53948068,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
         1.86340654,  0.57735027, -1.51611537, -1.21315763],
       [ 0.65099941, -0.8872621 ,  2.88675135,  1.05325486,  1.62078469,
        -0.39300048,  0.57735027, -0.08590546, -0.86991393],
       [ 0.22297457, -0.8872621 ,  2.88675135,  0.1179281 , -0.12467575,
        -0.39300048, -1.73205081,  0.27164701,  0.69375182],
       [ 1.05384632, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027, -0.31343886, -0.14528834],
       [ 0.22297457,  1.12706267,  2.88675135, -1.4126066 , -0.12467575,
        -0.8160768 , -1.73205081,  1.53933306,  1.53279198],
       [ 1.23009185,  1.12706267, -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654,  0.57735027,  1.21428536,  0.69375182],
       [-0.15469441, -0.8872621 ,  2.88675135,  0.1179281 , -1.87013618,
        -0.8160768 ,  0.57735027,  0.46667563,  0.54119906],
       [ 0.22297457,  1.12706267,  2.88675135,  1.05325486,  1.62078469,
         1.86340654,  0.57735027, -0.60598179, -0.25970291],
       [ 1.48187117,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  0.40166609,  1.26582466],
       [-0.53236339, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
        -0.39300048,  0.57735027,  0.07661839,  0.23609355],
       [ 0.72653321, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027, -0.44345794, -0.83177574],
       [ 1.28044771,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
        -0.39300048,  0.57735027, -0.80101042, -0.71736118],
       [-1.03592203,  1.12706267, -0.34641016,  1.05325486, -0.12467575,
         1.86340654,  0.57735027, -1.58112491, -0.2978411 ],
       [-2.04303932, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654,  0.57735027,  0.33665655,  1.07513371],
       [-0.28058407, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
```

```
       [-0.28058407, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
        -0.39300048,  0.57735027,  1.0517615 ,  1.49465379],
       [ 0.47475389, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  1.18178059, -1.36571039],
       [ 1.20491392, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
         1.86340654,  0.57735027,  0.7267138 ,  0.84630458],
       [ 0.60064355, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  0.95424719,  0.92258096],
       [-0.28058407,  1.12706267, -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654, -1.73205081, -1.61362968, -0.41225566],
       [-0.30072642,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
         1.86340654,  0.57735027,  1.01925673, -1.06060488],
       [ 1.15455805,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
        -0.8160768 , -1.73205081,  0.75921857, -0.33597928],
       [ 0.97831253, -0.8872621 ,  2.88675135,  0.1179281 , -1.87013618,
        -0.39300048,  0.57735027, -0.41095317, -1.48012496],
       [ 0.72653321, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  1.50682829,  0.57933725],
       [-0.02880475,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
        -0.8160768 ,  0.57735027, -0.86601996, -0.67922299],
       [ 0.47475389, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  1.08426627,  1.30396285],
       [-0.53236339, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  0.36916132,  1.4565156 ],
       [ 0.61071472, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  0.56418995,  0.15981717],
       [-1.03592203, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027,  0.59669472,  1.38023922],
```

```
        -0.8160768 ,  0.57735027,  1.2792949 ,  0.57933725],
       [-0.02880475, -0.8872621 , -0.34641016, -1.4126066 , -1.87013618,
        -0.39300048,  0.57735027, -1.02854381,  0.96071914],
       [ 0.47475389,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
         1.86340654,  0.57735027,  0.43417086, -0.52667023],
       [-0.53236339, -0.8872621 , -0.34641016, -1.4126066 ,  1.62078469,
        -0.39300048, -1.73205081,  1.24679013,  1.60906836],
       [ 0.47475389, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027, -1.32108674, -0.9843285 ],
       [ 0.22297457, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -2.07321785,  0.57735027,  1.73436168, -0.18342653],
       [ 0.22297457, -0.8872621 , -0.34641016,  1.05325486, -1.87013618,
        -0.39300048,  0.57735027, -0.73600088,  0.27423174],
       [ 0.72653321,  1.12706267,  2.88675135,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027, -1.45110583,  0.0835408 ],
       [ 1.48187117, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027,  0.85673288,  1.41837741],
       [ 1.05384632,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
        -0.8160768 ,  0.57735027,  0.98675196,  0.4267845 ],
       [ 0.48734285, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.8160768 ,  0.57735027, -0.50846748, -0.6410848 ],
       [-1.53948068, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654,  0.57735027, -0.24842932, -0.45039385],
       [ 0.47475389,  1.12706267, -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027,  0.04411362,  0.19795536],
       [-1.53948068, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048, -1.73205081, -0.67099133,  0.7700282 ],
       [-1.03592203, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
        -0.39300048,  0.57735027,  0.01160885, -1.70895409],
       [-1.03592203, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
```

```
[26]: Y_test

[26]: array([-0.37656648, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
             -0.37656648, -0.37656648,  0.11007328, -0.37656648,  0.11007328,
             -0.37656648,  0.11007328, -0.37656648, -0.37656648, -0.86320624,
              0.11007328, -0.86320624, -0.37656648, -0.37656648, -0.37656648,
             -0.37656648])

[27]: Y_train

[27]: array([-0.37656648, -0.37656648, -0.37656648,  0.11007328, -0.86320624,
             -0.37656648, -0.37656648,  3.02991185, -0.86320624, -0.86320624,
             -0.37656648,  2.05663233,  3.02991185,  0.11007328, -0.86320624,
             -0.37656648, -0.37656648, -0.37656648,  1.0833528 , -0.37656648,
             -0.37656648, -0.37656648, -0.37656648,  0.59671304, -0.86320624,
             -0.86320624, -0.86320624, -0.86320624,  0.11007328, -0.37656648,
              0.11007328,  0.11007328,  0.11007328, -0.37656648,  2.05663233,
             -0.86320624, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
             -0.37656648, -0.86320624, -0.37656648, -0.86320624,  1.56999257,
             -0.37656648,  1.0833528 , -0.37656648, -0.86320624,  0.11007328,
             -0.86320624,  1.0833528 , -0.37656648,  0.11007328, -0.86320624,
             -0.86320624, -0.37656648,  0.59671304,  2.54327209, -0.37656648,
              0.11007328, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
              0.11007328, -0.86320624, -0.37656648,  2.05663233, -0.37656648,
              0.11007328, -0.37656648,  0.11007328, -0.86320624, -0.86320624,
             -0.86320624,  1.0833528 ,  0.59671304,  3.02991185, -0.86320624,
             -0.37656648,  0.11007328,  0.11007328,  3.02991185])
```

4.2 Association Rule Mining:

Association rules are if-then statements that help to show the probability of relationships between data items within large data sets in various types of databases. Association rule mining has a number of applications and is widely used to help discover sales correlations in transactional data.

Apriori:

```
[1]: import numpy as np
     import matplotlib.pyplot as plt
     import pandas as pd
     from apyori import apriori

[2]: dataset = pd.read_csv(r"C:\Users\PC\Desktop\casestudy\foodchoices.csv",header=None)
     nr=len(dataset)
     print(nr)

     126

[3]: dataset.isnull().sum()

[3]: 0     0
     1     0
     2     0
     3     0
     4     0
     5     0
     6     0
     7     1
     8     1
     9    19
     dtype: int64
```

```
[4]: dataset.dropna(inplace=True)
     dataset.isnull().sum()

[4]: 0    0
     1    0
     2    0
     3    0
     4    0
     5    0
     6    0
     7    0
     8    0
     9    0
     dtype: int64

[5]: nr=len(dataset)
     print(nr)

     106

[11]: observations = []
      for i in range(0,nr):

          observations.append([str(dataset.values[i,j]) for j in range(0,9)])

[18]: associations = apriori(observations, min_length = 2, min_support = 0.025, min_confidence = 0.2, min_lift = 1)
      associations_results = list(associations)

[19]: print(len(associations_results))

      229
```



```
[20]: print(associations_results[4])

      RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozens
      et(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)])

[15]: for i in range(0,6):
          print(associations_results[i])

      RelationRecord(items=frozenset({'1'}), support=0.9245283018867925, ordered_statistics=[OrderedStatistic(items_base=frozenset
      (), items_add=frozenset({'1'}), confidence=0.9245283018867925, lift=1.0)])
      RelationRecord(items=frozenset({'2'}), support=0.5283018867924528, ordered_statistics=[OrderedStatistic(items_base=frozenset
      (), items_add=frozenset({'2'}), confidence=0.5283018867924528, lift=1.0)])
      RelationRecord(items=frozenset({'3'}), support=0.6886792452830188, ordered_statistics=[OrderedStatistic(items_base=frozenset
      (), items_add=frozenset({'3'}), confidence=0.6886792452830188, lift=1.0)])
      RelationRecord(items=frozenset({'4'}), support=0.22641509433962265, ordered_statistics=[OrderedStatistic(items_base=frozense
      t(), items_add=frozenset({'4'}), confidence=0.22641509433962265, lift=1.0)])
      RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozens
      et(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)])
      RelationRecord(items=frozenset({'610'}), support=0.4716981132075472, ordered_statistics=[OrderedStatistic(items_base=frozens
      et(), items_add=frozenset({'610'}), confidence=0.4716981132075472, lift=1.0)])
```

FPGrowth:

he FP-Growth Algorithm, proposed by Han, is an efficient and scalable method for mining the complete set of frequent patterns by pattern fragment growth, using an extended prefix-tree structure for storing compressed and crucial information about frequent patterns named frequent-pattern tree (FP-tree). In his study, Han proved that his method outperforms other popular methods for mining frequent patterns.

```
[1]: pip install pyfpgrowth

     Requirement already satisfied: pyfpgrowth in c:\users\pc\anaconda3\lib\site-packages (1.0)
     Note: you may need to restart the kernel to use updated packages.

[1]: import numpy as np
     import pandas as pd

[2]: import pyfpgrowth

[3]: data = pd.read_csv(r'C:\Users\PC\Desktop\casestudy\foodchoices.csv')
     x, y = data.shape
     records = []
     for i in range(x):
         records.append([str(data.values[i,j]) for j in range(y)])

[5]: patterns = pyfpgrowth.find_frequent_patterns(records, 3)
     rules = pyfpgrowth.generate_association_rules(patterns, 0.5)
     print(rules)

     {('2', 'Boredom'): (('1',), 1.75), ('1', 'Boredom'): ((), 0.6666666666666666), ('1', '2', 'Boredom'): (('420',), 1.0), ('1',
     '2.0', 'Boredom'): (('420',), 0.6666666666666666), ('2', '2.0', 'Boredom'): (('1', '420'), 1.75), ('1', '1', 'Boredom'):
     (('420',), 1.0), ('2', '420', 'Boredom'): (('1',), 1.75), ('1', '420', 'Boredom'): ((), 0.6666666666666666), ('1', '2', '2.
     0', 'Boredom'): (('420',), 1.0), ('1', '2', '420', 'Boredom'): (('2.0',), 1.0), ('1', '2.0', '420', 'Boredom'): ((), 0.66666
     66666666666), ('2', '2.0', '420', 'Boredom'): (('1',), 1.75), ('1', '1', '2.0', 'Boredom'): (('420',), 1.0), ('1', '1', '42
     0', 'Boredom'): (('2.0',), 1.0), ('1', 'Boredom '): ((), 0.6666666666666666), ('2.0', 'Boredom '): (('1', '2'), 1.3333333333
     333333), ('1', '2', 'Boredom '): (('2.0',), 1.0), ('1', '2.0', 'Boredom '): (('2',), 0.6666666666666666), ('2', '2.0', 'Bore
     dom '): (('1',), 1.3333333333333333), ('2', '3.1'): ((), 0.875), ('2.0', '3.1'): (('2',), 2.6666666666666665), ('7.0',):
     (('1', '2'), 1.3333333333333333), ('1', '7.0'): (('2',), 1.0), ('2', '7.0'): (('1',), 0.6666666666666666), ('1', 'sadness'):
```

4.4 Classification:

It is a Data analysis task, i.e. the process of finding a model that describes and distinguishes data classes and concepts. Classification is the problem of identifying to which of a set of categories a new observation belongs to, on the basis of a training set of data containing observations and whose categories membership is known.

Decision tree:

A decision tree is a structure that includes a root node, branches, and leaf nodes. Each internal node denotes a test on an attribute, each branch denotes the outcome of a test, and each leaf node holds a class label. The topmost node in the tree is the root node.

```
[1]: import pandas as pd
     import numpy as np

[2]: dataset = pd.read_csv(r"C:\Users\PC\Desktop\casestudy\foodchoices.csv")

[3]: dataset.columns

[3]: Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
            'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
            'comfort_food_reasons_coded'],
           dtype='object')

[4]: dataset.dropna(inplace=True)
     dataset.isnull().sum()

[4]: GPA                           0
     Gender                        0
     breakfast                     0
     calories_chicken              0
     calories_day                  0
     calories_scone                0
     coffee                        0
     comfort_food                  0
     comfort_food_reasons          0
     comfort_food_reasons_coded    0
     dtype: int64

[5]: dataset.drop(['GPA', 'Gender', 'comfort_food'],axis=1,inplace=True)
     dataset.head()
```

[5]:

	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food_reasons	comfort_food_reasons_coded
0	1	430	3	315	1	we dont have comfort	9.0
1	1	610	3	420	2	Stress, bored, anger	1.0
2	1	720	4	420	2	stress, sadness	1.0
3	1	430	3	420	2	Boredom	2.0
4	1	720	2	420	2	Stress, boredom, cravings	1.0

```
[6]: y=dataset.iloc[:,0]
     y

[6]: 0      1
     1      1
     2      1
     3      1
     4      1
           ..
     101    1
     102    1
     103    1
     104    1
     105    1
     Name: breakfast, Length: 105, dtype: int64
```

[7]:
```
x=dataset.iloc[:,[1,2,3,4]]
x
```

[7]:

	calories_chicken	calories_day	calories_scone	coffee
0	430	3	315	1
1	610	3	420	2
2	720	4	420	2
3	430	3	420	2
4	720	2	420	2
...	...	...	...	...
101	610	4	420	2
102	610	4	420	2
103	610	3	420	2
104	720	3	420	2
105	720	3	420	2

105 rows × 4 columns

[9]:
```
labelencoder_x=LabelEncoder()
```

[10]:
```
x=x.apply(LabelEncoder().fit_transform)
```

[11]:
```
x
```

[11]:

	calories_chicken	calories_day	calories_scone	coffee
0	1	1	1	0
1	2	1	2	1
2	3	2	2	1
3	1	1	2	1
4	3	0	2	1
...	...	...	...	...
101	2	2	2	1
102	2	2	2	1
103	2	1	2	1
104	3	1	2	1
105	3	1	2	1

105 rows × 4 columns

[12]:
```
from sklearn.tree import DecisionTreeClassifier
```

[13]:
```
regressor=DecisionTreeClassifier()
```

[14]:
```
regressor.fit(x.iloc[:,0:5],y)
```

[14]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

[15]:
```
x_in=np.array([2,0,0,0])
```

[16]:
```
y_pred=regressor.predict([x_in])
```

[17]:
```
y_pred
```

[17]:
```
array([2], dtype=int64)
```

[18]:
```
from sklearn.externals.six import StringIO
```

```
[19]: from IPython.display import Image
```

```
[20]: from sklearn. tree import export_graphviz
```

```
[21]: pip install pydotplus
```

```
Requirement already satisfied: pydotplus in c:\users\pc\anaconda3\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\pc\anaconda3\lib\site-packages (from pydotplus) (2.4.6)
Note: you may need to restart the kernel to use updated packages.
```

```
[22]: import pydotplus
```

```
[23]: dot_data=StringIO()
```

```
[24]: from sklearn.tree import DecisionTreeClassifier, export_graphviz
```

```
[25]: export_graphviz(regressor,out_file=dot_data,filled=True,rounded=True,special_characters=True)
```

```
[26]: graph=pydotplus.graph_from_dot_data(dot_data.getvalue())
```

```
[27]: pip install graphviz
```

```
Requirement already satisfied: graphviz in c:\users\pc\anaconda3\lib\site-packages (0.14)
Note: you may need to restart the kernel to use updated packages.
```

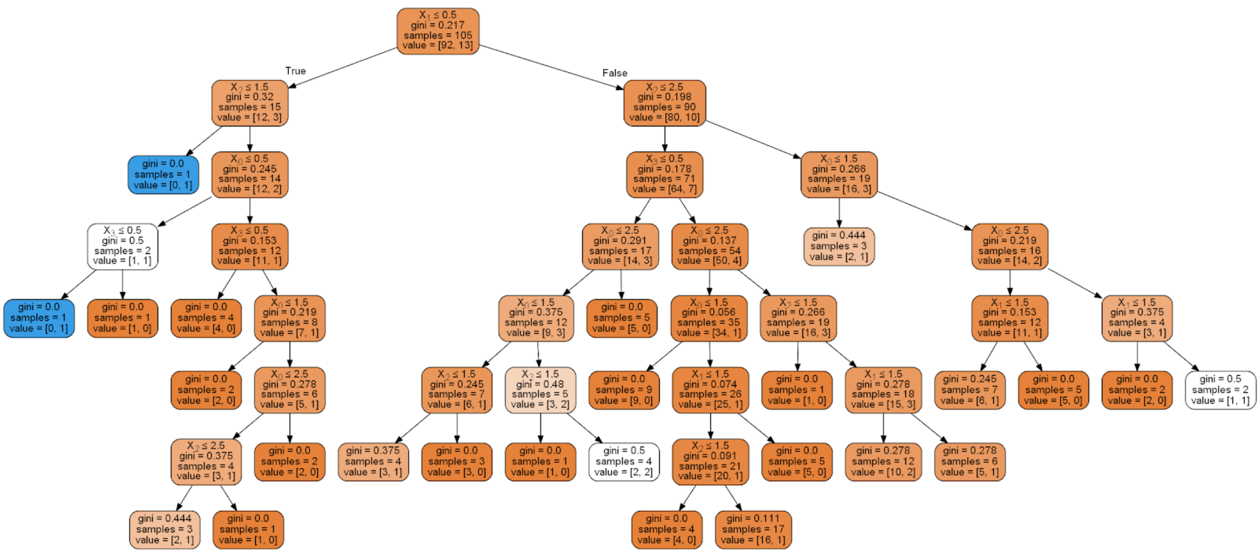
```
[28]: conda install graphviz
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

# All requested packages already installed.


Note: you may need to restart the kernel to use updated packages.
```

```
[29]: conda install -c conda-forge pydotplus
```

```
Collecting package metadata (current_repodata.json): ...working... done
Solving environment: ...working... done

## Package Plan ##

  environment location: C:\Users\PC\anaconda3

  added / updated specs:
    - pydotplus


The following packages will be UPDATED:

  conda                  pkgs/main::conda-4.8.3-py37_0 --> conda-forge::conda-4.8.3-py37hc8dfbb8_1
```

The tree generated :



4.4 Clustering:

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. In simple words, the aim is to segregate groups with similar traits and assign them into clusters.

```
[1]: import os
     import pandas as pd
     import numpy as np
     import matplotlib.pyplot as plt
     from sklearn.linear_model import LogisticRegression
     from sklearn.model_selection import train_test_split
     import seaborn as sns
```

```
[2]: data=pd.read_csv(r"C:\Users\PC\Desktop\casestudy\foodchoices.csv")
     data.head()
```

[2]:

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	comfort_food_reasons_coded
0	2.400	2	1	430	3	315	1	none	we dont have comfort	9.0
1	3.654	1	1	610	3	420	2	chocolate, chips, ice cream	Stress, bored, anger	1.0
2	3.300	1	1	720	4	420	2	frozen yogurt, pizza, fast food	stress, sadness	1.0
3	3.200	1	1	430	3	420	2	Pizza, Mac and cheese, ice cream	Boredom	2.0
4	3.500	1	1	720	2	420	2	Ice cream, chocolate, chips	Stress, boredom, cravings	1.0

```
[3]: data.shape#structure of dataset
```

```
[3]: (125, 10)
```

```
[4]: data.isnull().sum()#checking wheather missing values are there in dataset?
```

```
[4]: GPA                           0
     Gender                        0
     breakfast                     0
     calories_chicken              0
     calories_day                  0
     calories_scone                0
     coffee                        0
     comfort_food                  1
     comfort_food_reasons          1
     comfort_food_reasons_coded   19
     dtype: int64
```

```
[5]: data.dtypes#checking datatypes
```

```
[5]: GPA                           float64
     Gender                          int64
     breakfast                       int64
     calories_chicken                int64
     calories_day                    int64
     calories_scone                  int64
     coffee                          int64
     comfort_food                   object
     comfort_food_reasons           object
     comfort_food_reasons_coded    float64
     dtype: object
```

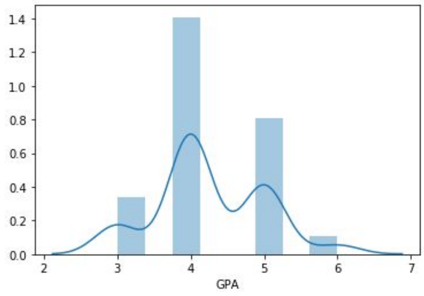
```
[6]: data['GPA'] = data['coffee'] + data['breakfast'] + data['Gender']
     data['calories_chicken']=data['calories_day']//3
     data.head()
```

[6]:

	GPA	Gender	breakfast	calories_chicken	calories_day	calories_scone	coffee	comfort_food	comfort_food_reasons	comfort_food_reasons_coded
0	4	2	1	1	3	315	1	none	we dont have comfort	9.0
1	4	1	1	1	3	420	2	chocolate, chips, ice cream	Stress, bored, anger	1.0
2	4	1	1	1	4	420	2	frozen yogurt, pizza, fast food	stress, sadness	1.0
3	4	1	1	1	3	420	2	Pizza, Mac and cheese, ice cream	Boredom	2.0
4	4	1	1	0	2	420	2	Ice cream, chocolate, chips	Stress, boredom, cravings	1.0

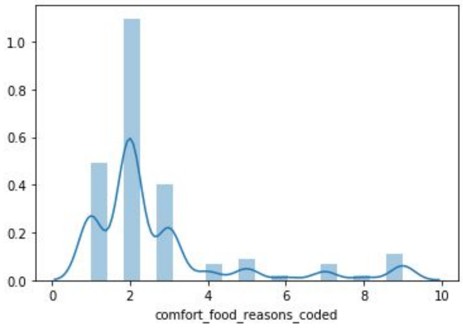
[9]:
```
sns.distplot(data['GPA'])
```

[9]: <matplotlib.axes._subplots.AxesSubplot at 0x23ecda3988>



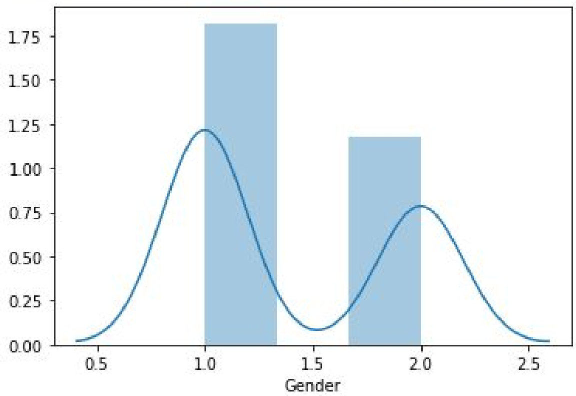
[11]:
```
sns.distplot(data['comfort_food_reasons_coded'])
```

[11]: <matplotlib.axes._subplots.AxesSubplot at 0xafd132b608>



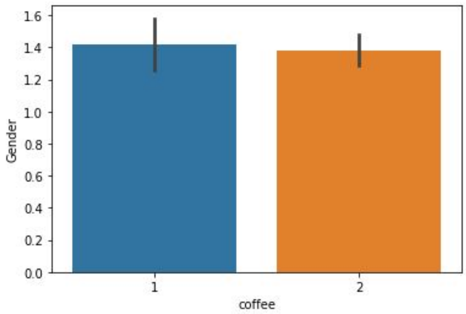
[10]:
```
sns.distplot(data['Gender'])
```

[10]: <matplotlib.axes._subplots.AxesSubplot at 0xafd12a3b88>



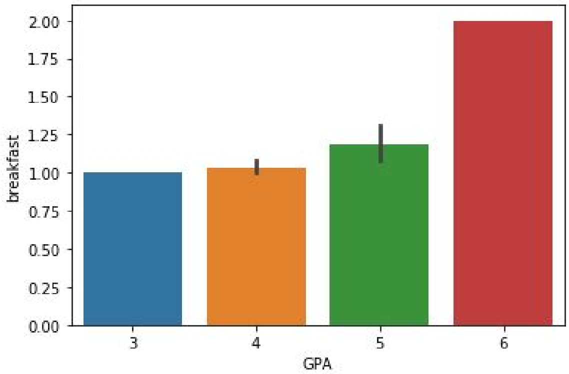
[12]:
```
sns.barplot(data['coffee'], data['Gender'])
```

[12]: <matplotlib.axes._subplots.AxesSubplot at 0xafd13bc488>



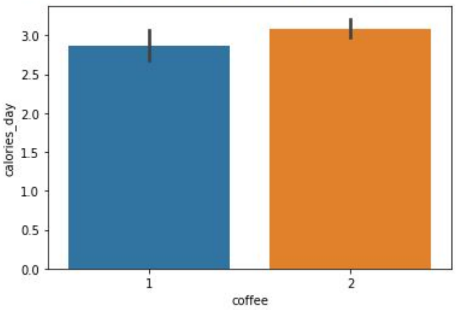
```
[13]: sns.barplot(data['GPA'],data['breakfast'])
```

```
[13]: <matplotlib.axes._subplots.AxesSubplot at 0xafd1316e08>
```

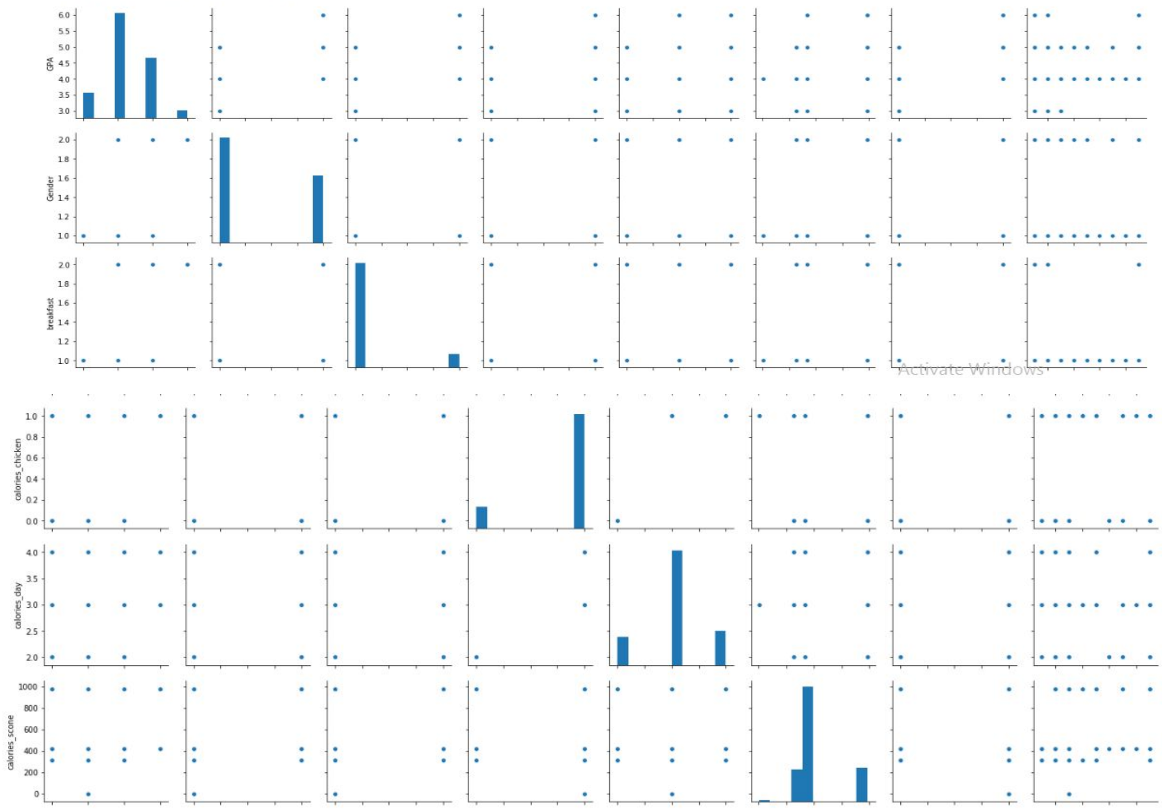


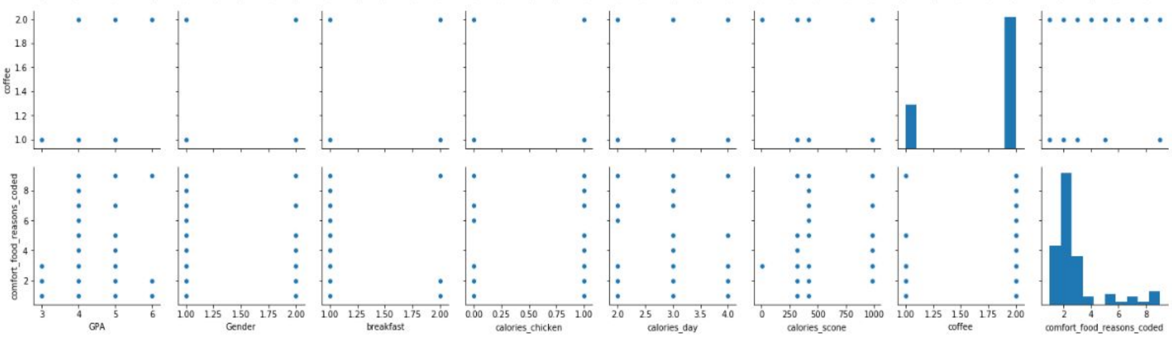

```
[14]: sns.barplot(data['coffee'],data['calories_day'])
```

```
[14]: <matplotlib.axes._subplots.AxesSubplot at 0xafd14a8588>
```




```
[15]: <seaborn.axisgrid.PairGrid at 0xafd1425288>
```





[16]:
```python
from sklearn import preprocessing
```

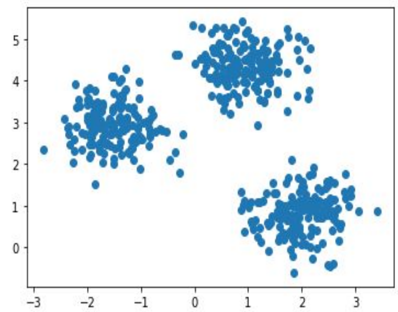
[17]:
```python
data.columns
```

[17]:
```
Index(['GPA', 'Gender', 'breakfast', 'calories_chicken', 'calories_day',
       'calories_scone', 'coffee', 'comfort_food', 'comfort_food_reasons',
       'comfort_food_reasons_coded'],
      dtype='object')
```

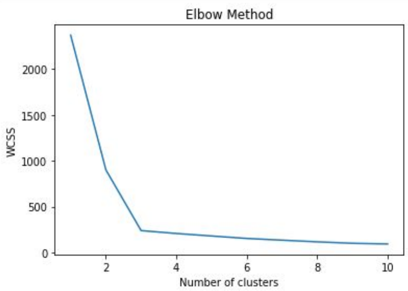
[18]:
```python
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import KMeans
```

[19]:
```python
data_2, y = make_blobs(n_samples=500, centers=3, cluster_std=0.50, random_state=0)
plt.scatter(data_2[:,0], data_2[:,1])
```

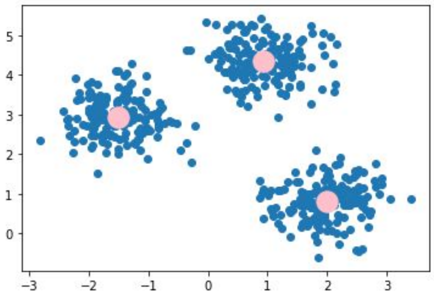
[19]:
```
<matplotlib.collections.PathCollection at 0xafd5c30fc8>
```



[20]:
```python
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=500, n_init=10, random_state=0)
    kmeans.fit(data_2)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



```
[21]: kmeans = KMeans(n_clusters=3, init='k-means++', max_iter=500, n_init=10, random_state=0)
      pred_y = kmeans.fit_predict(data_2)
      plt.scatter(data_2[:,0], data_2[:,1])
      plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s=300, c='pink')
      plt.show()
      print(pred_y)
```




```
[0 2 2 0 0 0 0 2 2 0 2 2 2 2 0 0 1 0 2 0 1 1 1 2 0 2 1 2 0 2 2 0 0 0 1 1 1
 1 1 1 2 0 0 1 1 0 1 0 2 2 0 0 0 0 2 2 1 1 0 2 1 2 1 1 1 2 0 1 2 0 2 0 0 2
 1 2 1 1 1 0 0 0 2 0 2 1 2 2 2 2 0 0 1 0 2 1 2 2 2 0 1 2 2 0 0 1 0 1 0 1 0
 0 1 2 2 1 2 1 1 2 2 0 2 2 2 0 2 0 2 0 1 0 2 1 0 1 2 2 1 0 0 2 2 1 0 0 0 1
 1 0 1 0 0 0 1 0 2 1 2 0 1 1 2 2 2 1 0 1 0 1 2 2 1 0 0 2 2 1 1 1 1 1 2 2 1
 1 0 1 0 2 2 0 1 1 0 2 1 0 2 1 2 1 0 0 2 1 1 1 2 2 0 1 1 2 2 0 2 0 2 2 1 2
 1 1 0 0 0 1 2 0 0 2 2 1 2 2 0 1 0 0 0 1 1 1 0 2 2 2 2 2 1 2 2 0 2 2 0 1 2
 1 1 0 0 1 1 0 2 1 2 1 1 2 1 0 0 1 0 1 1 1 1 2 1 0 0 0 0 2 2 1 1 2 2 0 0 1
 2 0 2 1 0 1 2 1 0 2 0 1 0 2 1 2 2 0 0 0 1 2 2 0 0 1 2 1 0 0 1 1 0 2 1 0 1
 2 1 1 0 2 0 2 1 2 1 0 0 0 1 0 0 2 1 0 2 2 2 0 1 1 1 2 0 1 2 0 0 0 2 0 2 0
 2 2 0 2 2 2 2 1 1 2 1 2 2 2 2 0 0 0 1 2 0 1 0 1 0 1 2 2 0 2 1 0 1 2 2 0 1
 2 1 2 0 0 0 1 2 0 0 1 2 2 0 2 1 0 2 0 1 0 2 0 0 1 0 0 0 0 1 0 1 2 1 1 0 2
 1 2 1 2 1 0 2 1 1 1 1 1 0 2 1 2 0 0 1 2 2 0 2 1 0 0 1 1 2 1 2 1 1 1 1 2 0
 1 1 0 1 2 2 0 1 1 2 0 2 0 0 1 1 1 0 1]
```

5. Experimental Results

5.1 Data Preprocessing:

For the following dataset, data preprocessing is done, Missing values are handled, converted to arrays, categorical data is converted to numeric, data training and test data are obtained.

```
[24]: x_test

[24]: array([[ 1.23009185, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
         -0.39300048,  0.57735027, -0.57347702, -1.3275722 ],
       [ 0.22297457,  1.12706267, -0.34641016, -2.81559674, -0.12467575,
         -0.39300048,  0.57735027, -0.9310295 , -1.17501944],
       [ 0.22297457, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
          1.86340654,  0.57735027, -0.28093409, -1.51826315],
       [-1.53948068, -0.8872621 ,  2.88675135,  1.05325486, -0.12467575,
         -0.39300048,  0.57735027,  0.62919949,  0.57933725],
       [-0.53236339,  1.12706267, -0.34641016,  0.1179281 , -1.87013618,
         -0.39300048,  0.57735027,  0.82422811,  1.57093017],
       [-1.03592203, -0.8872621 , -0.34641016,  0.1179281 , -1.87013618,
         -0.39300048,  0.57735027,  1.34430444, -1.40384858],
       [ 0.72653321, -0.8872621 , -0.34641016, -2.81559674, -0.12467575,
         -0.8160768 ,  0.57735027,  1.57183783,  0.57933725],
       [ 0.47475389, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
         -0.39300048,  0.57735027,  1.47432352, -0.06901196],
       [-0.78414271,  1.12706267,  2.88675135, -2.81559674, -1.87013618,
         -0.39300048, -1.73205081, -0.21592455, -1.44198677],
       [-1.03592203, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
         -0.39300048,  0.57735027,  1.31179967,  1.22768647],
       [ 0.72653321, -0.8872621 ,  2.88675135,  0.1179281 , -0.12467575,
         -0.39300048, -1.73205081,  0.69420903,  0.61747544],
       [ 1.24016302, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
         -0.39300048,  0.57735027,  0.66170426,  1.18954828],
       [-0.53236339, -0.8872621 , -0.34641016, -1.4126066 , -0.12467575,
         -0.39300048,  0.57735027, -0.11841024, -1.55640134],
```

```
[25]: x_train
```

```
[25]: array([[-1.53948068,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
         1.86340654,  0.57735027, -1.51611537, -1.21315763],
       [ 0.65099941, -0.8872621 ,  2.88675135,  1.05325486,  1.62078469,
        -0.39300048,  0.57735027, -0.08590546, -0.86991393],
       [ 0.22297457, -0.8872621 ,  2.88675135,  0.1179281 , -0.12467575,
        -0.39300048, -1.73205081,  0.27164701,  0.69375182],
       [ 1.05384632, -0.8872621 , -0.34641016,  1.05325486, -0.12467575,
        -0.39300048,  0.57735027, -0.31343886, -0.14528834],
       [ 0.22297457,  1.12706267,  2.88675135, -1.4126066 , -0.12467575,
        -0.8160768 , -1.73205081,  1.53933306,  1.53279198],
       [ 1.23009185,  1.12706267, -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654,  0.57735027,  1.21428536,  0.69375182],
       [-0.15469441, -0.8872621 ,  2.88675135,  0.1179281 , -1.87013618,
        -0.8160768 ,  0.57735027,  0.46667563,  0.54119906],
       [ 0.22297457,  1.12706267,  2.88675135,  1.05325486,  1.62078469,
         1.86340654,  0.57735027, -0.60598179, -0.25970291],
       [ 1.48187117,  1.12706267, -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027,  0.40166609,  1.26582466],
       [-0.53236339, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
        -0.39300048,  0.57735027,  0.07661839,  0.23609355],
       [ 0.72653321, -0.8872621 , -0.34641016,  0.1179281 , -0.12467575,
        -0.39300048,  0.57735027, -0.44345794, -0.83177574],
       [ 1.28044771,  1.12706267, -0.34641016, -1.4126066 , -0.12467575,
        -0.39300048,  0.57735027, -0.80101042, -0.71736118],
       [-1.03592203,  1.12706267, -0.34641016,  1.05325486, -0.12467575,
         1.86340654,  0.57735027, -1.58112491, -0.2978411 ],
       [-2.04303932, -0.8872621 , -0.34641016,  0.1179281 ,  1.62078469,
         1.86340654,  0.57735027,  0.33665655,  1.07513371],
       [-0.28058407, -0.8872621 , -0.34641016,  1.05325486,  1.62078469,
```

```
[26]: Y_test
```

```
[26]: array([-0.37656648, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
       -0.37656648, -0.37656648,  0.11007328, -0.37656648,  0.11007328,
       -0.37656648,  0.11007328, -0.37656648, -0.37656648, -0.86320624,
        0.11007328, -0.86320624, -0.37656648, -0.37656648, -0.37656648,
       -0.37656648])
```

```
[27]: Y_train
```

```
[27]: array([-0.37656648, -0.37656648, -0.37656648,  0.11007328, -0.86320624,
       -0.37656648, -0.37656648,  3.02991185, -0.86320624, -0.86320624,
       -0.37656648,  2.05663233,  3.02991185,  0.11007328, -0.86320624,
       -0.37656648, -0.37656648, -0.37656648,  1.0833528 , -0.37656648,
       -0.37656648, -0.37656648, -0.37656648,  0.59671304, -0.86320624,
       -0.86320624, -0.86320624, -0.86320624,  0.11007328, -0.37656648,
        0.11007328,  0.11007328,  0.11007328, -0.37656648,  2.05663233,
       -0.86320624, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
       -0.37656648, -0.86320624, -0.37656648, -0.86320624,  1.56999257,
       -0.37656648,  1.0833528 , -0.37656648, -0.86320624,  0.11007328,
       -0.86320624,  1.0833528 , -0.37656648,  0.11007328, -0.86320624,
       -0.86320624, -0.37656648,  0.59671304,  2.54327209, -0.37656648,
        0.11007328, -0.37656648, -0.37656648, -0.37656648, -0.37656648,
        0.11007328, -0.86320624, -0.37656648,  2.05663233, -0.37656648,
        0.11007328, -0.37656648,  0.11007328, -0.86320624, -0.86320624,
       -0.86320624,  1.0833528 ,  0.59671304,  3.02991185, -0.86320624,
       -0.37656648,  0.11007328,  0.11007328,  3.02991185])
```

5.2 Apriori and FP-Tree Algorithm

Association rule mining is performed as following, RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozenset(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)]) is one of the rules obtained implementing apriori, based on this we can make conclusions.

```
[20]: print(associations_results[4])

RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozens
et(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)])
```

```
[15]: for i in range(0,6):
          print(associations_results[i])

RelationRecord(items=frozenset({'1'}), support=0.9245283018867925, ordered_statistics=[OrderedStatistic(items_base=frozenset
(), items_add=frozenset({'1'}), confidence=0.9245283018867925, lift=1.0)])
RelationRecord(items=frozenset({'2'}), support=0.5283018867924528, ordered_statistics=[OrderedStatistic(items_base=frozenset
(), items_add=frozenset({'2'}), confidence=0.5283018867924528, lift=1.0)])
RelationRecord(items=frozenset({'3'}), support=0.6886792452830188, ordered_statistics=[OrderedStatistic(items_base=frozenset
(), items_add=frozenset({'3'}), confidence=0.6886792452830188, lift=1.0)])
RelationRecord(items=frozenset({'4'}), support=0.22641509433962265, ordered_statistics=[OrderedStatistic(items_base=frozense
t(), items_add=frozenset({'4'}), confidence=0.22641509433962265, lift=1.0)])
RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozens
et(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)])
RelationRecord(items=frozenset({'610'}), support=0.4716981132075472, ordered_statistics=[OrderedStatistic(items_base=frozens
et(), items_add=frozenset({'610'}), confidence=0.4716981132075472, lift=1.0)])
```


By using FPgrowth another rule is obtained {('2', 'Boredom'): (('1',), 1.75), ('1', 'Boredom'): ((), 0.6666666666666666)

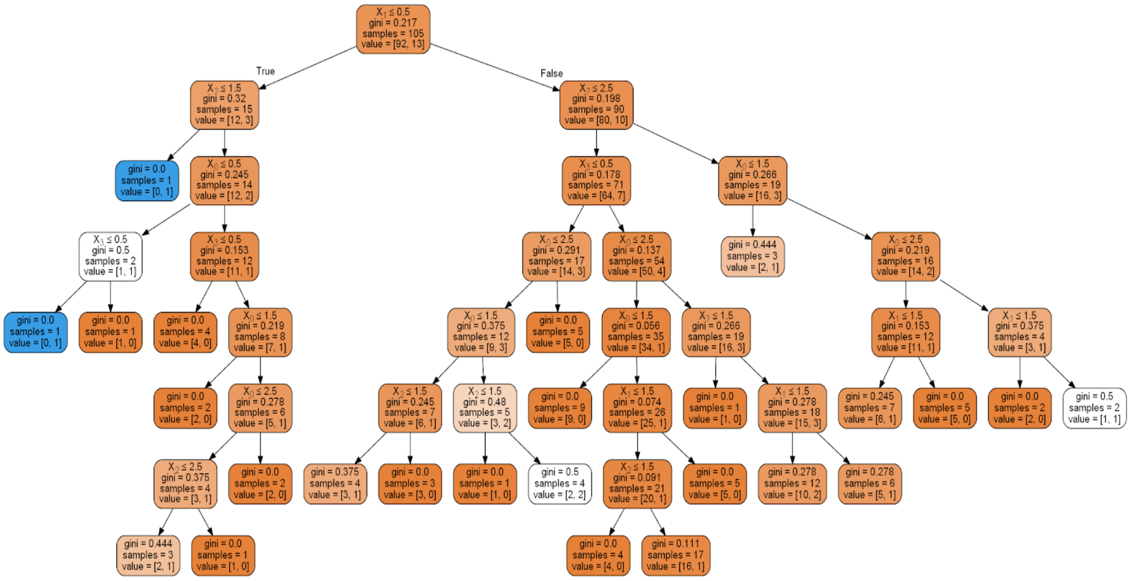
```
[6]: print(rules)

{('2', 'Boredom'): (('1',), 1.75), ('1', 'Boredom'): ((), 0.6666666666666666), ('1', '2', 'Boredom'): (('420',), 1.0), ('1',
'2.0', 'Boredom'): (('420',), 0.6666666666666666), ('2', '2.0', 'Boredom'): (('1', '420'), 1.75), ('1', '1', 'Boredom'):
(('420',), 1.0), ('2', '420', 'Boredom'): (('1',), 1.75), ('1', '420', 'Boredom'): ((), 0.6666666666666666), ('1', '2', '2.
0', 'Boredom'): (('420',), 1.0), ('1', '2', '420', 'Boredom'): (('2.0',), 1.0), ('1', '2.0', '420', 'Boredom'): ((), 0.66666
66666666666), ('2', '2.0', '420', 'Boredom'): (('1',), 1.75), ('1', '1', '2.0', 'Boredom'): (('420',), 1.0), ('1', '1', '42
0', 'Boredom'): (('2.0',), 1.0), ('1', 'Boredom '): ((), 0.6666666666666666), ('2.0', 'Boredom '): (('1', '2'), 1.3333333333
333333), ('1', '2', 'Boredom '): (('2.0',), 1.0), ('1', '2.0', 'Boredom '): (('2',), 0.6666666666666666), ('2', '2.0', 'Bore
dom '): (('1',), 1.3333333333333333), ('2', '3.1'): ((), 0.875), ('2.0', '3.1'): (('2',), 2.6666666666666665), ('7.0',):
(('1', '2'), 1.3333333333333333), ('1', '7.0'): (('2',), 1.0), ('2', '7.0'): (('1',), 0.6666666666666666), ('1', 'sadness'):
((), 0.875), ('3.0', 'sadness'): (('1',), 2.6666666666666665), ('1', '5.0'): (('2',), 1.3333333333333333), ('2', '5.0'):
(('1',), 1.3333333333333333), ('1', '9.0'): (('2',), 1.5), ('1', '9.0', '980'): (('2',), 1.3333333333333333), ('2', '9.0',
'980'): ((), 0.6666666666666666), ('2', '9.0'): ((), 0.7), ('2', '2', '9.0'): (('980',), 0.5714285714285714), ('2', '2.0',
'2.8'): (('1',), 1.0), ('2', '2.8', '3'): (('1',), 0.5), ('1', '2.8'): ((), 0.5), ('1', '2', '2.8'): (('3',), 0.75), ('1',
'2.0', '2.8'): ((), 0.6), ('1', '2', '2.0', '2.8'): (('3',), 1.0), ('1', '2', '2.8', '3'): ((), 0.5), ('1', '2.0', '2.8',
'3'): ((), 0.6), ('2', '2.0', '2.8', '3'): (('1',), 1.0), ('1', '1', '2.8'): (('3',), 0.75), ('1', '1', '2.0', '2.8'):
(('3',), 1.0), ('1', '1', '2.8', '3'): (('2.0',), 1.0), ('2', '2', '2.8'): (('1',), 1.0), ('1', '2', '2', '2.8'): (('3',),
1.0), ('2', '2', '2.8', '3'): (('1',), 1.0), ('2', '3.8', '430'): (('1',), 2.0), ('2', '3.8'): ((), 0.6363636363636364),
('1', '2', '3.8'): ((), 0.6), ('1', '3.8', '420'): (('2',), 1.4444444444444444), ('1', '3.8', '430'): ((), 0.714285714285714
3), ('1', '2', '3.8', '420'): ((), 0.6153846153846154), ('1', '2', '3.8', '430'): (('420',), 1.0), ('1', '3.8', '420', '43
0'): ((), 0.7142857142857143), ('2', '3.8', '420', '430'): (('1',), 2.0), ('1', '1', '3.8'): (('2',), 0.8), ('1', '1', '3.
8', '420'): (('2',), 0.8), ('1', '1', '3.8', '430'): (('420',), 1.0), ('2', '2', '3.8'): (('1',), 1.2857142857142858), ('1',
'2', '2', '3.8'): (('420',), 0.8888888888888888), ('2', '2', '3.8', '420'): (('1',), 1.3333333333333333), ('1', '1', '2',
'3.8'): (('420',), 1.0), ('1', '1.0', '4.0'): ((), 0.6666666666666666), ('1', '4.0', '420'): ((), 0.625), ('1.0', '4.0', '42
0'): (('1', '2'), 1.3333333333333333), ('3', '4.0', '420'): (('1', '2'), 1.5), ('1', '1.0', '3', '4.0'): (('2', '420'), 0.66
66666666666666), ('1', '1.0', '4.0', '420'): (('2',), 0.6666666666666666), ('1', '3', '4.0', '420'): (('2',), 0.75), ('1.0',
'3', '4.0', '420'): (('1', '2'), 1.3333333333333333), ('1', '1', '4.0'): (('3',), 1.0), ('1', '1', '1.0', '4.0'): (('3',),
```


5.3 ID3 Algorithm

Classification is performed,

```
[17]: y_pred

[17]: array([2], dtype=int64)
```

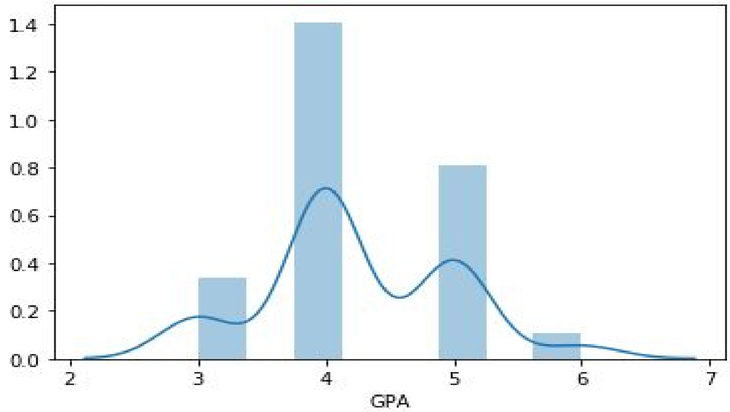
The picture of tree is given below:



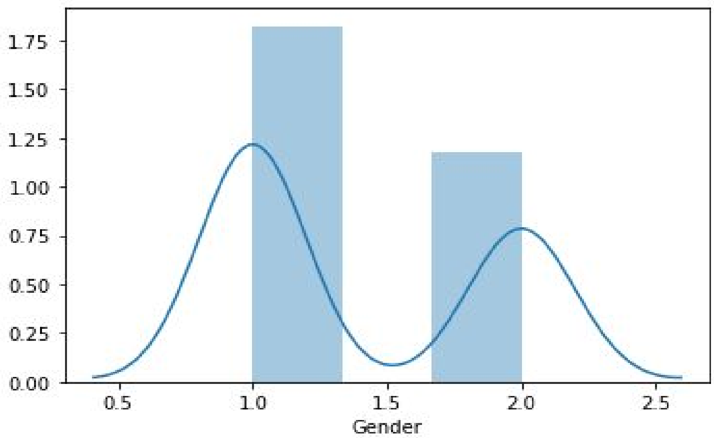
5.4 Clustering

Clustering process is performed . Clustering is performed using the K Means algorithm and also plotted clusters to understand the behavior of data..All the clusters and their result analysis are displayed.

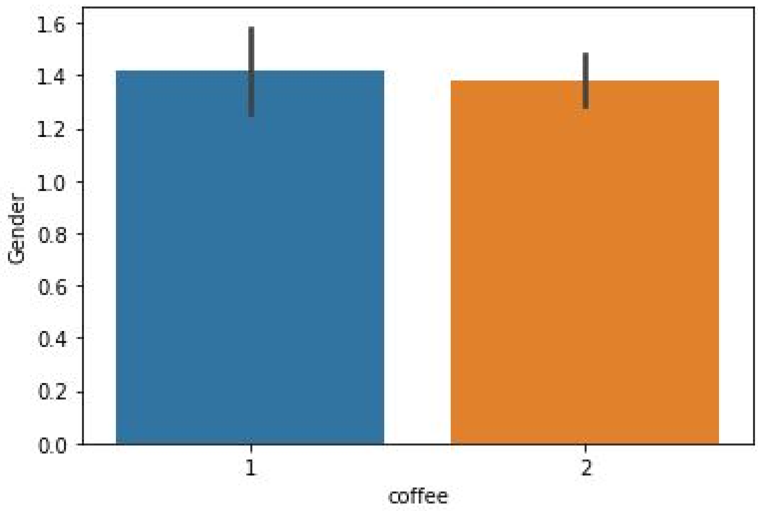
```
<matplotlib.axes._subplots.AxesSubplot at 0xafd033ad08>
```



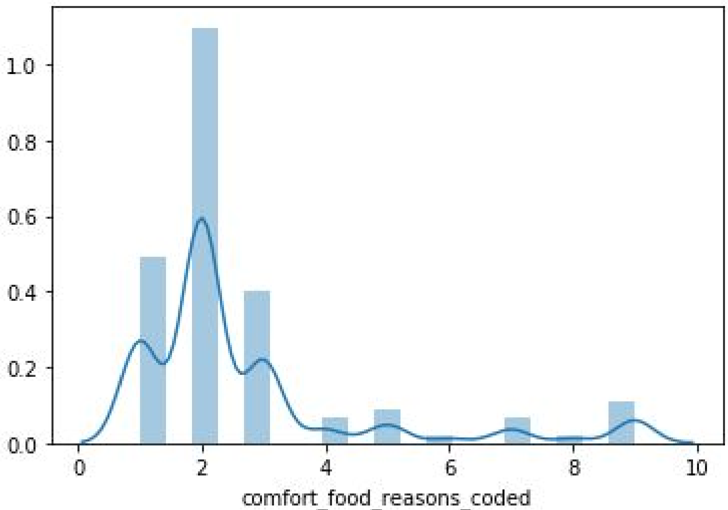

```
<matplotlib.axes._subplots.AxesSubplot at 0xafd12a3b88>
```



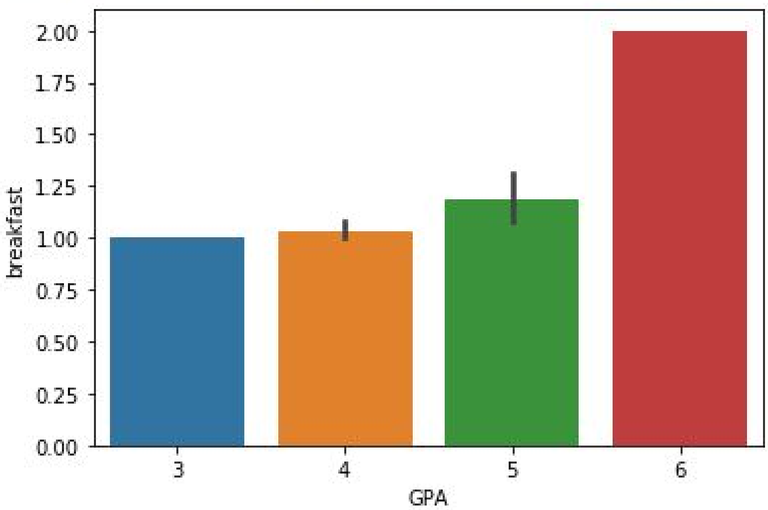
```
<matplotlib.axes._subplots.AxesSubplot at 0xafd13bc488>
```



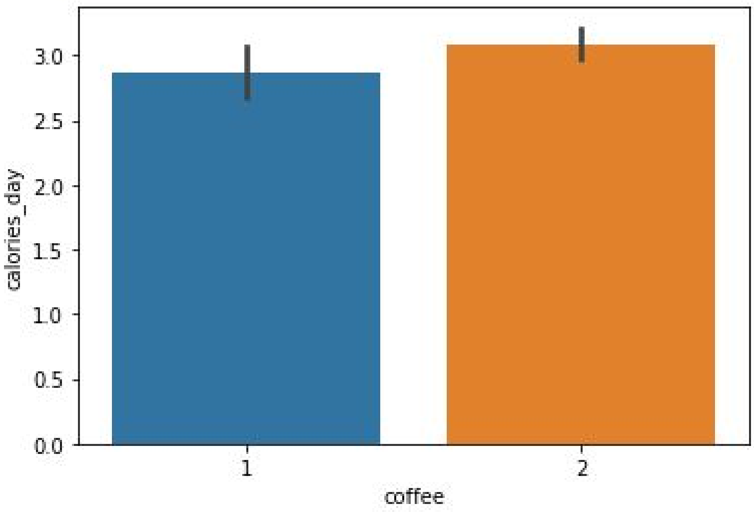

```
<matplotlib.axes._subplots.AxesSubplot at 0xafd132b608>
```

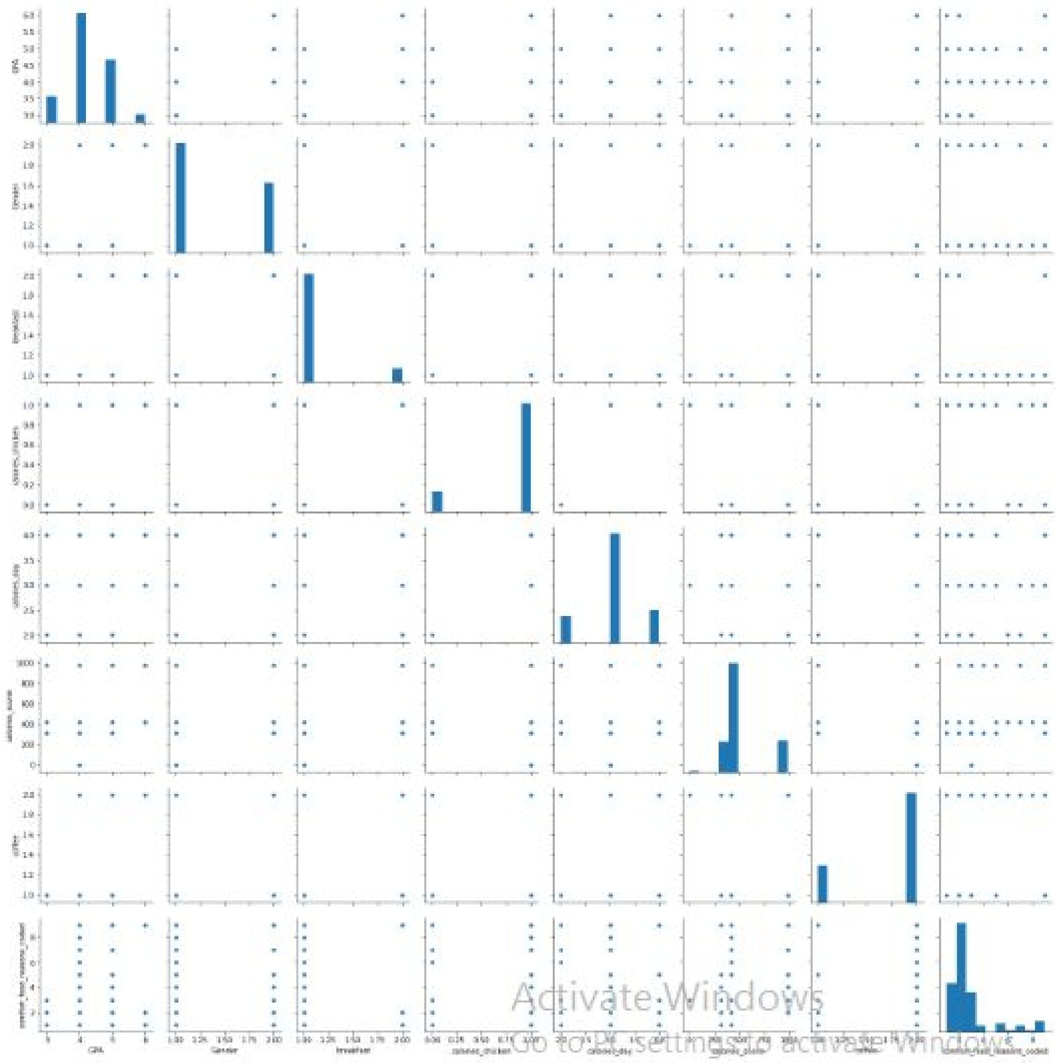



```
<matplotlib.axes._subplots.AxesSubplot at 0xafd1316e08>
```

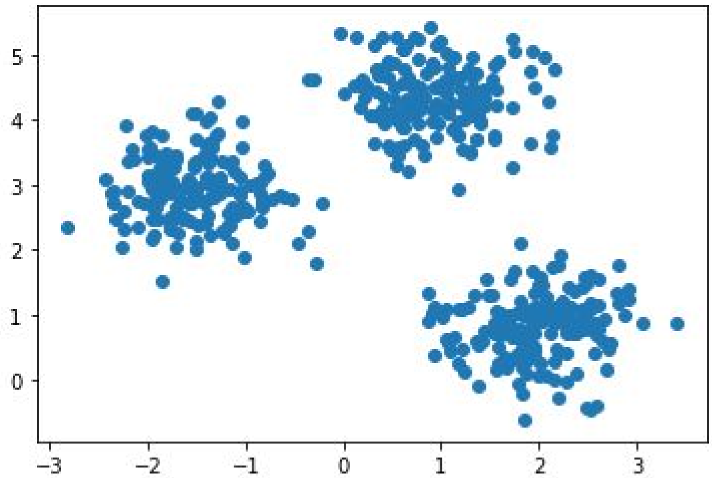


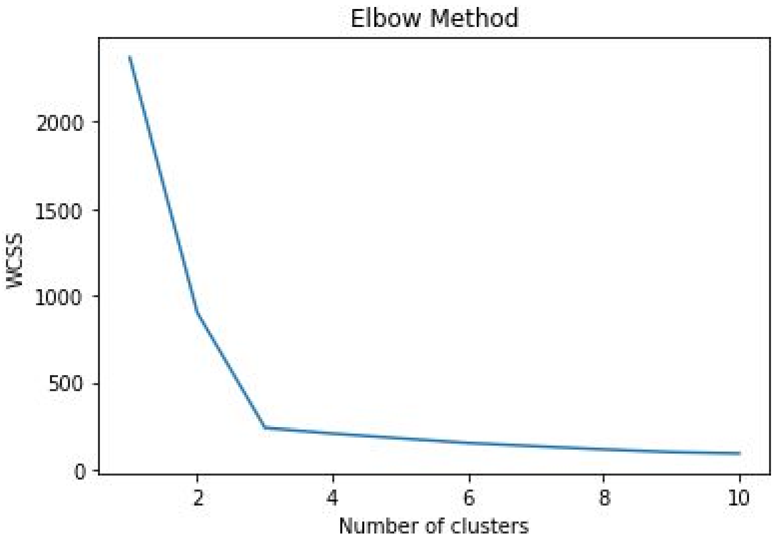
```
<matplotlib.axes._subplots.AxesSubplot at 0xafd14a8588>
```

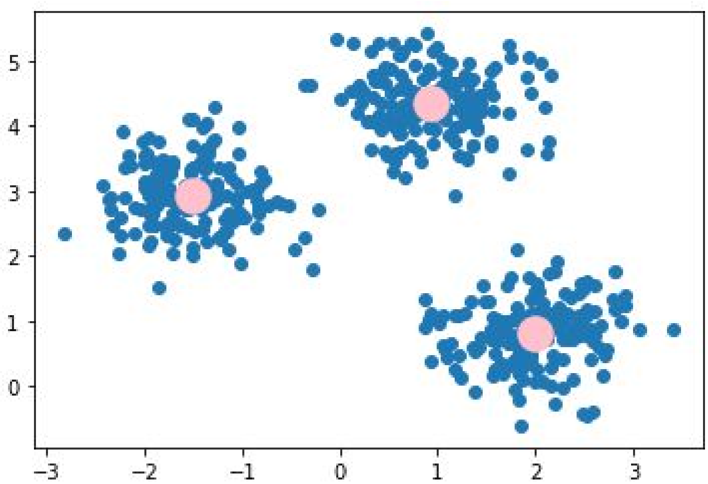




```
<matplotlib.collections.PathCollection at 0xafd5c30fc8>
```







6. Conclusion

I hereby conclude that i have doneData preprocessing is done, Missing values are handled, converted to arrays, categorical data is converted to numeric, data training and test data are obtained. Association rule mining is performed,
RelationRecord(items=frozenset({'420'}), support=0.6415094339622641, ordered_statistics=[OrderedStatistic(items_base=frozenset(), items_add=frozenset({'420'}), confidence=0.6415094339622641, lift=1.0)])) is one of the rules obtained implementing apriori, based on this we can make conclusions.
{('2', 'Boredom'): (('1',), 1.75), ('1', 'Boredom'): ((), 0.6666666666666666) is another rule obtained using FPgrowth.and then Classification is performed,.Clustering is performed and results are.Clustering is performed using K Means algorithm and also plotted clusters to understand the behavior of data, By observing similarity in the clusters behaviour can be analyzed.

7. References:

https://medium.com/code-to-express/k-means-clustering-for-beginners-using-python-from-scratch-f20e79c8ad00
https://towardsdatascience.com/data-preprocessing-in-python-b52b652e37d5/
https://stackoverflow.com/questions/49435621/how-to-define-variable-name-in-jupyter-notebook-through-interactive-widget-pyth
https://towardsdatascience.com/the-complete-guide-to-classification-in-python-b0e34c92e455
https://youtu.be/fl0PH6uQDIw
https://towardsdatascience.com/an-introduction-to-clustering-algorithms-in-python-123438574097