# DATA SCIENCE PROJECT REPORT

## 1 : WHAT WILL THE PROGRAM DO ?

The program will take input from the user, Preforms the elbow method on the dataset to find the optimal cluster number, Then performs the k-mean algorithm, And displays the scatter plot of the k-means , Then displays a table displaying each customer name, age, total and the computed cluster number. It then takes the item column then splits by commas into multiple columns then removes the “null or none” values as preprocessing for the apriori algorithm , then prints the table showing confidence , support , lift for the item dataset.

## 2 : WHAT IS THE PROGRAM INPUT?

- dataSetPath : User should input the full path of the csv file.
- numberOfClusters : To use in the k-means function.
- minSupport : To use in the Apriori algorithm.
- minConfidence : To use in the Apriori algorithm

# 3: WHAT IS THE PROGRAM OUTPUT?

-Elbow method graph. -K-means Scatter plot.

-A table displaying each customer name, age, total and the computed cluster number.

-A table displaying the Apriori algorithm outputs including A-->B , Confidence , Support , Lift .

## DATASET DESCRIPTION

8 rows × 4 columns

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| : | count | mean | std | min | 25% | 50% | 75% | max |
| count : | 9835.000000 | 4.409456 | 3.589385 | 1.000000 | 2.000000 | 3.000000 | 6.000000 | 32.000000 |
| total : | 9835.000000 | 1292.752822 | 697.756493 | 100.000000 | 679.000000 | 1297.000000 | 1896.500000 | 2500.000000 |
| rnd : | 9835.000000 | 8.008439 | 4.326424 | 1.000000 | 4.000000 | 8.000000 | 12.000000 | 15.000000 |
| age : | 9835.000000 | 37.002237 | 12.234523 | 22.000000 | 25.000000 | 36.000000 | 50.000000 | 60.000000 |

## SCREENSHOT FROM PROJECT STEPS

Step1 : importing libraries , taking and validating user input.

```
In _
import pandas as pd
import os.path
import time
import matplotlib.pyplot as plt
from apyori import apriori
datasetPath = input("Enter Dataset Path:")
while not os.path.isfile(datasetPath):
    print("Please Enter a valid path.")
    time.sleep(3)
    datasetPath = input("Enter Dataset Path:")
else:
    numberOfClusters = int(input("Enter number of clusters (N) :"))
    if numberOfClusters >4 or numberOfClusters < 2:
        print("Number of clusters should be between 2 and 4.")
    minSupport = float(input("Enter minimum support :"))
    if minSupport < 0.001 or minSupport > 1:
        print("Minimum support should be between 0.001 and 1.")
    minConfidence = float(input("Enter minimum confidence :"))
    if minConfidence < 0.001 or minSupport > 1:
        print("Minimum confidence should be between 0.001 and 1")
    dataSet=pd.read_csv(datasetPath)
```

SCREENSHOT FROM PROJECT STEP

```
In _  1  x = dataSet.iloc[:,[2,5]].values

In _  1  from sklearn.cluster import KMeans
      2  inertia=[]
      3  for i in range(1,11):
      4      kmeans = KMeans(n_clusters=i,init='k-means++',n_init='auto')
      5      kmeans.fit(x)
      6      inertia.append(kmeans.inertia_)
      7  plt.plot(range(1,11),inertia)
```

STEP 2 : inputting “total” and “age” columns values from the dataset into X variable.

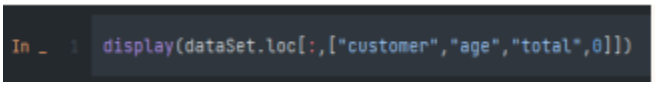
STEP 3 : using Elbow method to find optimal K value (Cluster number).

SCREENSHOT FROM PROJECT STEPS.

```
In _  1  kmeans = KMeans(n_clusters=numberOfClusters,init='k-means++',n_init='auto')
      2  y_kmeans = kmeans.fit_predict(x)
      3  dataSet = pd.concat([dataSet,pd.DataFrame(y_kmeans)],axis=1)
      4  plt.scatter(x[y_kmeans==0,0],x[y_kmeans==0,1],s=100,c='red')
      5  plt.scatter(x[y_kmeans==1,0],x[y_kmeans==1,1],s=100,c='green')
      6  plt.scatter(x[y_kmeans==2,0],x[y_kmeans==2,1],s=100,c='yellow')
      7  plt.scatter(x[y_kmeans==3,0],x[y_kmeans==3,1],s=100,c='black')
      8  plt.scatter(x[y_kmeans==4,0],x[y_kmeans==4,1],s=100,c='blue')
      9  plt.xlabel("Total")
     10  plt.ylabel("Age")
```

STEP 4 : Uses the k-mean library to cluster the dataset then plotting the clusters into a scatter plot.

SCREENSHOT FROM PROJECT STEPS.



```
In _  1  display(dataSet.loc[:,["customer","age","total",0]])
```

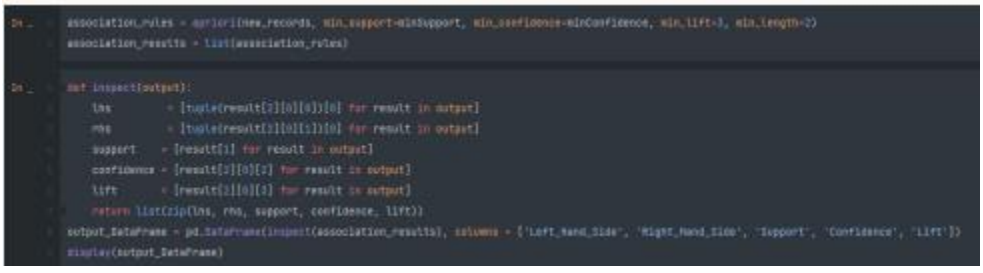
STEP 5 : Displays a table showing Customer , Age , Total , Clusters columns

SCREENSHOT FROM PROJECT STEPS.

```
records = []
itemDataSet =pd.DataFrame(dataSet['items'])
itemDataSet = pd.concat([itemDataSet['items'], itemDataSet['items'].str.split(',', expand=True)], axis=1)
for i in range(0, 9835):
    records.append([str(itemDataSet.values[i,j]) for j in range(0,33)])
new_records=[]
temp=[]
for i in range(0,9835):
    for j in range(0,len(records[i])):
        if records[i][j]=='None':
         temp.append(records[i][j])
    new_records.append(temp)
    temp=[]
del records
del temp
```

STEP 6 : Preprocesses the dataset item column in the dataset for the Apriori algorithm by splitting the values in the item column separated by a comma into different columns , Then removes the “None / null” values and saves them in a new list .

SCREENSHOT FROM PROJECT STEPS.



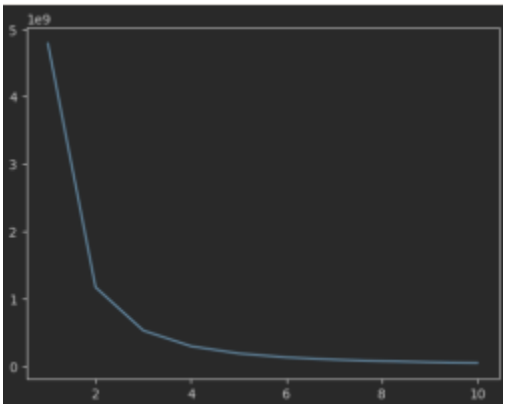
```
association_rules = apriori(new_records, min_support=minSupport, min_confidence=minConfidence, min_lift=3, min_length=2)
association_results = list(association_rules)

def inspect(output):
    lhs         = [tuple(result[2][0][0])[0] for result in output]
    rhs         = [tuple(result[2][0][1])[0] for result in output]
    support     = [result[1] for result in output]
    confidence  = [result[2][0][2] for result in output]
    lift        = [result[2][0][3] for result in output]
    return list(zip(lhs, rhs, support, confidence, lift))
output_DataFrame = pd.DataFrame(inspect(association_results), columns = ['Left_Hand_Side', 'Right_Hand_Side', 'Support', 'Confidence', 'Lift'])
display(output_DataFrame)
```

STEP 7 : Preforms the apriori algorithm on the saved list then saves the result in a new variable.

STEP 8 : Prints the result as a table using the created inspect method.

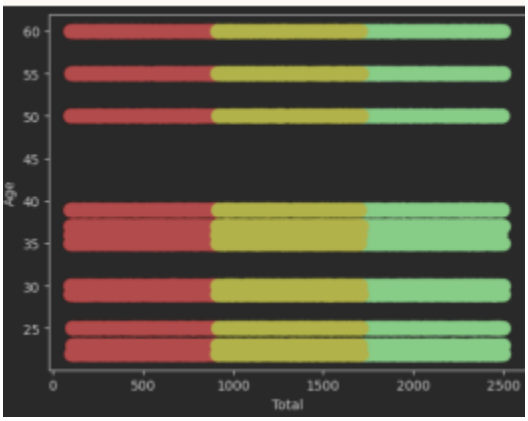
# GRAPH INSIGHTS

From the elbow method graph we can conclude that the optimal number of clusters is between 2 and 4, preferably 3



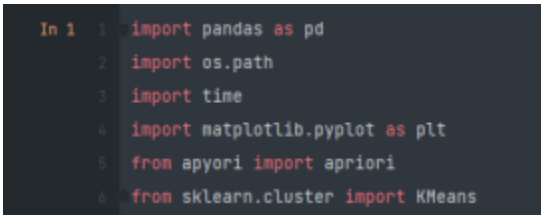
# GRAPH INSIGHTS

The scatter plot between the age and total columns show cluster sizes very close to each other to the point where its almost lines.



# CODE DISCUSS

Libraries used : -Pandas for dataframe usage. -Os for validating path input -Time to pause the while loop -Matplotlib.pyplot for graphs and plotting -Apyori for the apriori algorithm -scikit-learn for clustering and kmeans algorithm



# CODE DISCUSS

Attributes used in the code: -dataSetPath : User should input the full path of the csv file.

-numberOfClusters : To use in the k-means function. -minSupport : To use in the Apriori algorithm.

-minConfidence : To use in the Apriori algorithm.

-dataset : dataset Dataframe.

-intertia : list for kmeans intertia.

-x : a part of the dataset.

-kmeans : kmeans cluster to use in elbow method.

-y_kmeans : actual kmeans cluster used.

-records : list to temporary save records.

-new_records : list for final records after removal of null strings.

-temp : list to used for removing null strings.

-assosciation_rules : RelationRecord returned from using the apriori algorithm

- assosciation_results : list transforming the RelationRecord object into a list for usage.

-inspect(output) : function to manipulate the list into an organized list to be transformed into a dataframe for printing the final table showing apriori method results.

-output_DataFrame : dataframe containing the A→B , Lift , Confidence , Support .