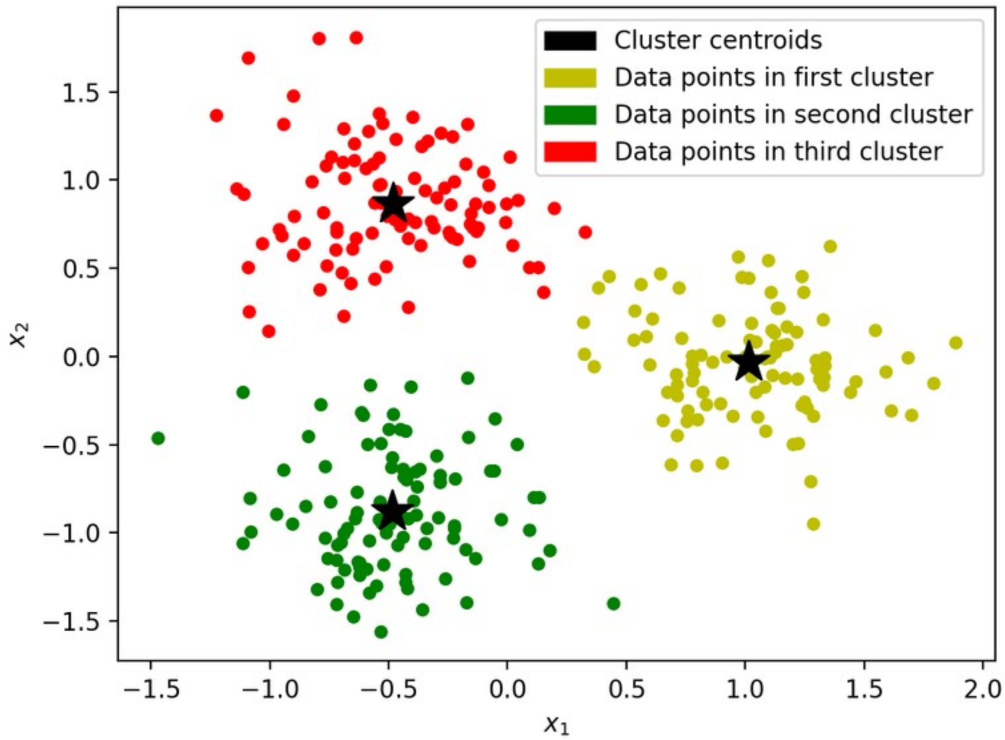
UNSUPERVISED CLUSTERING MODEL



Segregating the given regions of the provided data into different groups so that the Marketing team of a firm can plan their resources accordingly to launch their promotional campaign.

Date	14/04/2021
Name	Vinit Ravichandran Iyer
	Ashmitha Nagesh

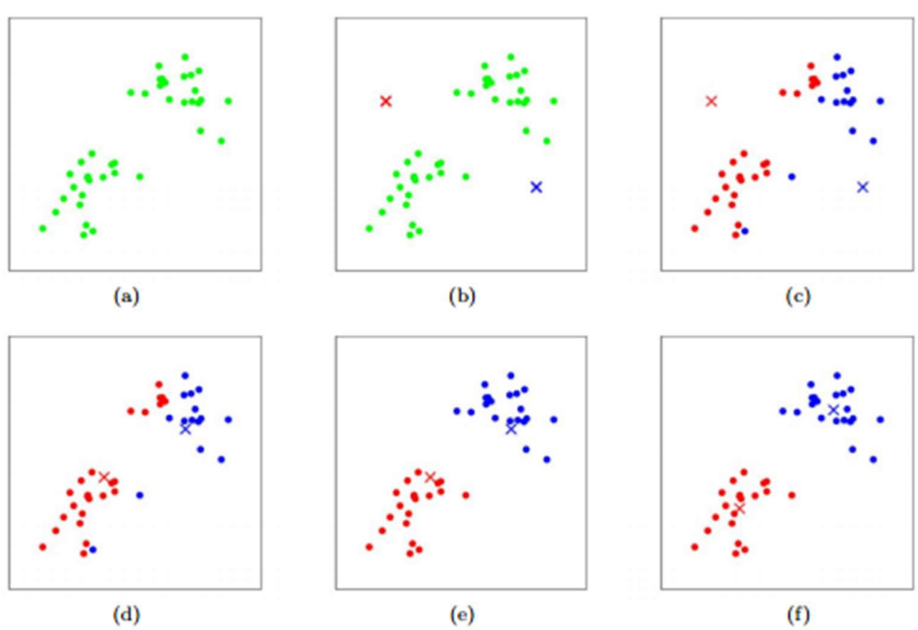
Content List

Introduction

Cluster analysis or clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters).

Unsupervised learning is a type of algorithm that learns patterns from untagged data. In contrast to supervised learning where data is tagged by a human, e.g., as "car" or "fish" etc, Unsupervised Learning exhibits self-organization that captures patterns as probability densities, etc. Clustering can be considered the most important *unsupervised learning* problem; so, as every other problem of this kind, it deals with finding a *structure* in a collection of un-labelled data.

K-means is one of the simplest unsupervised learning algorithms that solves the well-known clustering problem. The procedure follows a simple and easy way to classify a given data set through a certain number of clusters (assume k clusters) fixed a priori. The main idea is to define k centres, one for each cluster. These centroids should be placed in a smart way because of different location causes different result. The next step is to take each point belonging to a given data set and associate it to the nearest centroid. When no point is pending, the first step is completed and an early groupage is done. At this point we need to re-calculate k new centroids of the clusters resulting from the previous step. After we have these k new centroids, a new binding has to be done between the same data set points and the nearest new centroid. A loop has been generated. As a result of this loop we may notice that the k centroids change their location step by step until no more changes are done. In other words centroids do not move any more. In the following Clustering algorithm, we have utilized the K-means as a type of algorithm.



Source Code

1. Importing Libraries

Input	import pandas as pd import numpy as np import matplotlib.pyplot as plt

2. Importing the Data

Input	Data = pd.read_csv("Population_Data.csv") Data.head()

Output:

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population
0	Region 31	1	6,43,596	28,83,782	4,40,445	2,03,151	27,63,718	72,515	35,27,378
1	Region 17	9	3,19,933	15,01,899	2,13,477	1,06,456	14,49,303	27,671	1821832
2	Region 12	4	1,94,379	6,50,744	1,61,803	32,576	6,31,660	10,652	845123
3	Region 22	15	1,07,360	4,70,708	85,343	22,017	4,50,267	6,389	578068
4	Region 23	13	55,351	3,29,980	31,796	23,555	3,25,105	3,684	385331

3. Understanding the Data

Input	Data.head()

Output

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population
0	Region 31	1	6,43,596	28,83,782	4,40,445	2,03,151	27,63,718	72,515	35,27,378
1	Region 17	9	3,19,933	15,01,899	2,13,477	1,06,456	14,49,303	27,671	1821832
2	Region 12	4	1,94,379	6,50,744	1,61,803	32,576	6,31,660	10,652	845123
3	Region 22	15	1,07,360	4,70,708	85,343	22,017	4,50,267	6,389	578068
4	Region 23	13	55,351	3,29,980	31,796	23,555	3,25,105	3,684	385331

Input	Data.tail()

Output

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population
33	Region 28	30	2,067	20,183	1,840	227	20,002	181	22250
34	Region 7	36	1,867	12,424	1,404	463	12,335	88	14291
35	Region 35	37	1,212	6,084	998	214	6,042	36	7296
36	Region 14	33	865	6,777	574	291	6,691	86	7642
37	Region 3	38	686	5,737	380	306	5,702	35	6423

Input

```
Data.info()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   Region              38 non-null     object
 1   Office Location Id  38 non-null     int64 
 2   Indians             38 non-null     object
 3   Foreigners          38 non-null     object
 4   Indian_Male         38 non-null     object
 5   Indian_Female       38 non-null     object
 6   Foreigners_Male     38 non-null     object
 7   Foreigners_Female   38 non-null     object
 8   Total Population    38 non-null     object
dtypes: int64(1), object(8)
memory usage: 2.8+ KB
```

4. Improving the Data

- Grouping Numerical Variables together

Input

```
numeric_cols = ["Indians", "Foreigners", "Indian_Male", "Indian_Female", "Foreigners_Male", "Foreigners_Female", "Total Population"]

def cleaner(z):
   return z.replace(",","")

cleaner("100,200,300")

for i in Data[numeric_cols]:
   Data[i] = Data[i].apply(cleaner)

Data.head()
```

Output

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population
0	Region 31	1	643596	2883782	440445	203151	2763718	72515	3527378
1	Region 17	9	319933	1501899	213477	106456	1449303	27671	1821832
2	Region 12	4	194379	650744	161803	32576	631660	10652	845123
3	Region 22	15	107360	470708	85343	22017	450267	6389	578068
4	Region 23	13	55351	329980	31796	23555	325105	3684	385331

- Converting given data types to suitable data types

Input	Data.info()

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Region              38 non-null     object
 1   Office Location Id  38 non-null     int64
 2   Indians             38 non-null     object
 3   Foreigners          38 non-null     object
 4   Indian_Male         38 non-null     object
 5   Indian_Female       38 non-null     object
 6   Foreigners_Male     38 non-null     object
 7   Foreigners_Female   38 non-null     object
 8   Total Population    38 non-null     object
dtypes: int64(1), object(8)
memory usage: 2.8+ KB
```

Input

```
Data[numeric_cols] = Data[numeric_cols].apply(pd.to_numeric)

Data.info()
```

Output

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38 entries, 0 to 37
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Region              38 non-null     object
 1   Office Location Id  38 non-null     int64
 2   Indians             38 non-null     int64
 3   Foreigners          38 non-null     int64
 4   Indian_Male         38 non-null     int64
 5   Indian_Female       38 non-null     int64
 6   Foreigners_Male     38 non-null     int64
 7   Foreigners_Female   38 non-null     int64
 8   Total Population    38 non-null     int64
dtypes: int64(8), object(1)
memory usage: 2.8+ KB
```

- Verifying Given Data

Input	Data.head()

Output

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population
0	Region 31	1	643596	2883782	440445	203151	2763718	72515	3527378
1	Region 17	9	319933	1501899	213477	106456	1449303	27671	1821832
2	Region 12	4	194379	650744	161803	32576	631660	10652	845123
3	Region 22	15	107360	470708	85343	22017	450267	6389	578068
4	Region 23	13	55351	329980	31796	23555	325105	3684	385331

Input	Data[["Indians", "Foreigners"]].sum().sum() - Data["Total Population"].sum()
Output	0

Input	Data[["Indian_Male", "Indian_Female"]].sum().sum() - Data["Indians"].sum()
Output	0

Input	Data[["Foreigners_Male", "Foreigners_Female"]].sum().sum() - Data["Foreigners"].sum()
Output	-112859

Input	Male_Female_Population_sum = Data["Indian_Male"] + Data["Indian_Female"] + Data["Foreigners_Male"] + Data["Foreigners_Female"] Data["Others"] = Data["Total Population"] - Male_Female_Population_sum Data.head()

Output

	Region	Office Location Id	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population	Others
0	Region 31	1	643596	2883782	440445	203151	2763718	72515	3527378	47549
1	Region 17	9	319933	1501899	213477	106456	1449303	27671	1821832	24925
2	Region 12	4	194379	650744	161803	32576	631660	10652	845123	8432
3	Region 22	15	107360	470708	85343	22017	450267	6389	578068	14052
4	Region 23	13	55351	329980	31796	23555	325105	3684	385331	1191

Input	Data["Region"].nunique()
Output	38

Input	Data["Office Location Id"].nunique()
Output	38

Input	data1 = Data.drop(columns = ["Region", "Office Location Id"]) data1.head()

Output

	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Total Population	Others
0	643596	2883782	440445	203151	2763718	72515	3527378	47549
1	319933	1501899	213477	106456	1449303	27671	1821832	24925
2	194379	650744	161803	32576	631660	10652	845123	8432
3	107360	470708	85343	22017	450267	6389	578068	14052
4	55351	329980	31796	23555	325105	3684	385331	1191

5. Scaling the Data

<table>
<tr><td>Input</td><td>
<pre>
def percentage_converter(data, total, columns):
    for i in columns:
        data[i] = data[i]/data['Total Population']
    return data.drop(columns=['Total Population'])

data2 = percentage_converter(data1,'Total Population', data1.drop(columns=['Total Population']).columns)
data2.head()
</pre>
</td></tr>
</table>

Output

	Indians	Foreigners	Indian_Male	Indian_Female	Foreigners_Male	Foreigners_Female	Others
0	0.182457	0.817543	0.124865	0.057593	0.783505	0.020558	0.013480
1	0.175611	0.824389	0.117177	0.058433	0.795520	0.015189	0.013681
2	0.230001	0.769999	0.191455	0.038546	0.747418	0.012604	0.009977
3	0.185722	0.814278	0.147635	0.038087	0.778917	0.011052	0.024309
4	0.143645	0.856355	0.082516	0.061129	0.843703	0.009561	0.003091

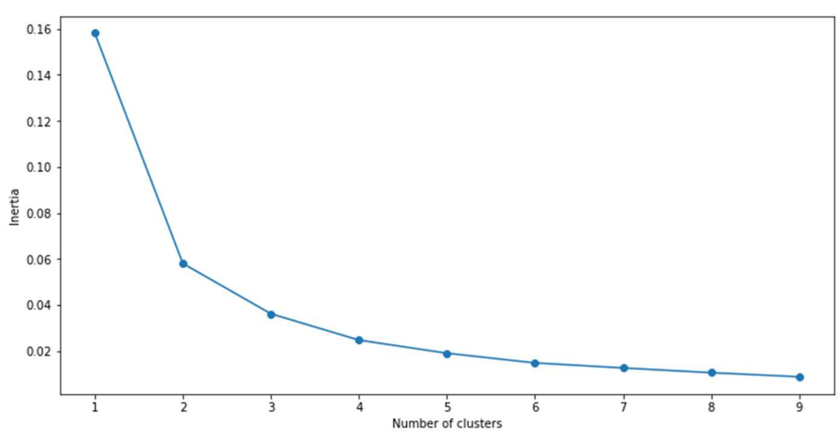
6. Visualizing Data for K Means

<table>
<tr><td>Input</td><td>
<pre>
from sklearn.cluster import KMeans
SSE = []
for cluster in range(1,10):
    kmeans = KMeans(n_clusters = cluster)
    kmeans.fit(data2)
    SSE.append(kmeans.inertia_)

frame = pd.DataFrame({'cluster':range(1,10), 'SSE':SSE})

plt.figure(figsize=(12,6))
plt.plot(frame['cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
</pre>
</td></tr>
</table>

Output



7. Using K-means to create cluster plots

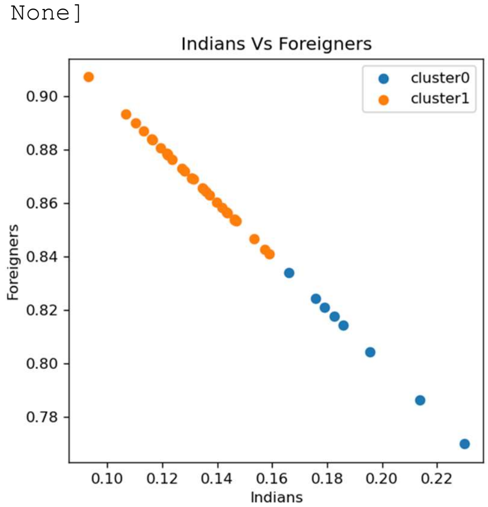
Input	

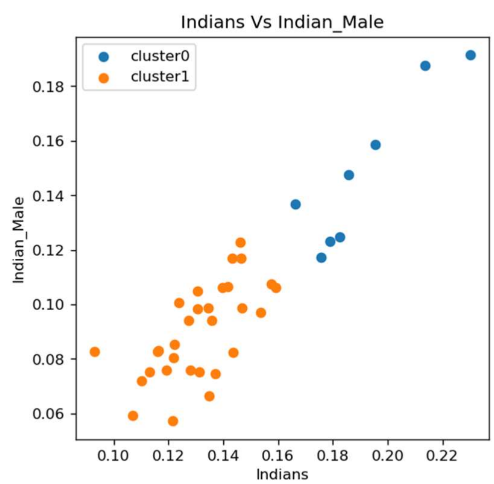
```
kmeans = KMeans(n_clusters=2)
kmeans.fit(data2)
pred=kmeans.predict(data2)

data2['cluster'] = pred

def seg(str_x, str_y, cluster):
    x = []
    y = []

    for i in range(cluster):
        x.append(data2[str_x][data2['cluster']==i])
        y.append(data2[str_y][data2['cluster']==i])

    return x,y

def plot_cluster(str_x, str_y, cluster):
    plt.figure(figsize = (5,5), dpi = 120)
    x,y = seg(str_x, str_y, cluster)

    for i in range(cluster):
        plt.scatter(x[i], y[i], label = 'cluster{}'.format(i))

    plt.xlabel(str_x)
    plt.ylabel(str_y)
    plt.title(str(str_x+" Vs "+str_y))
    plt.legend()
```

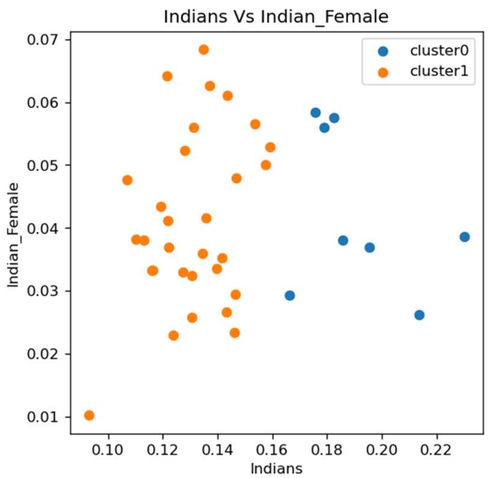
Output

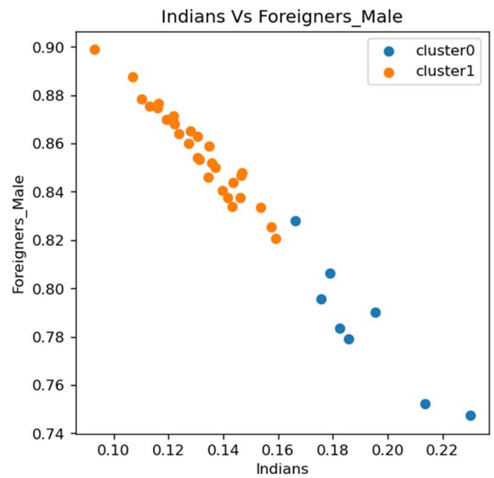
```
[None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
 None,
```

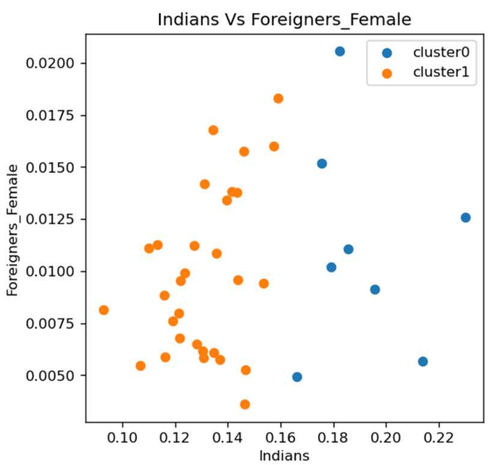
```
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None,
None]
```

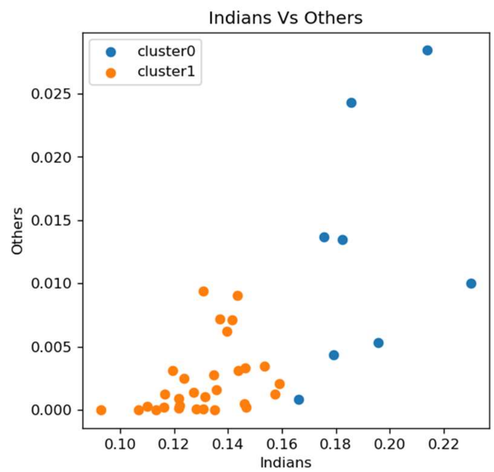


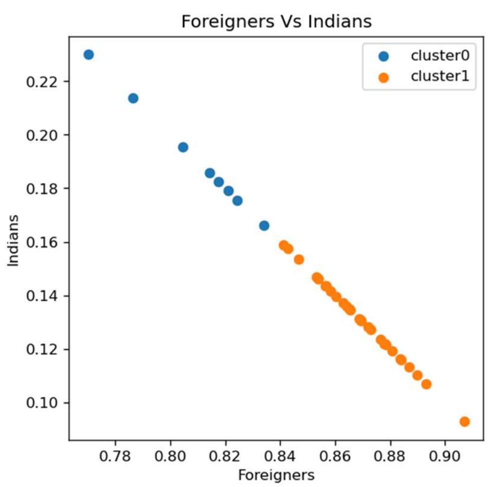


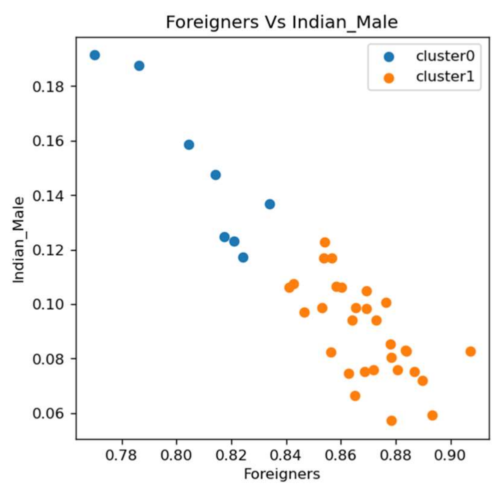

Indians Vs Indian_Female
cluster0
cluster1
Indian_Female
Indians

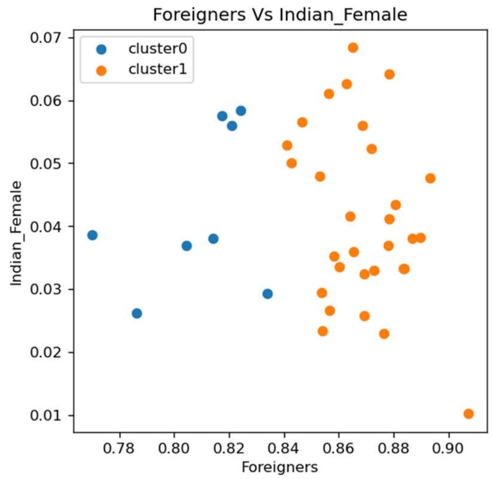

Indians Vs Foreigners_Male
cluster0
cluster1
Foreigners_Male
Indians

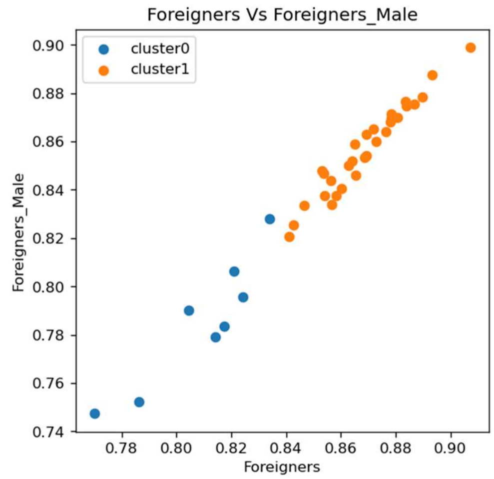

Indians Vs Foreigners_Female
cluster0
cluster1
Foreigners_Female
Indians

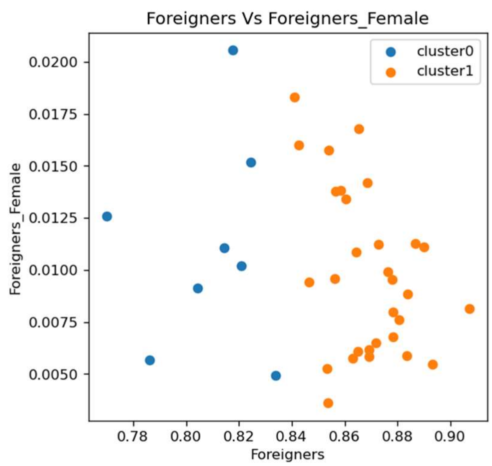

Indians Vs Others
cluster0
cluster1
Others
Indians

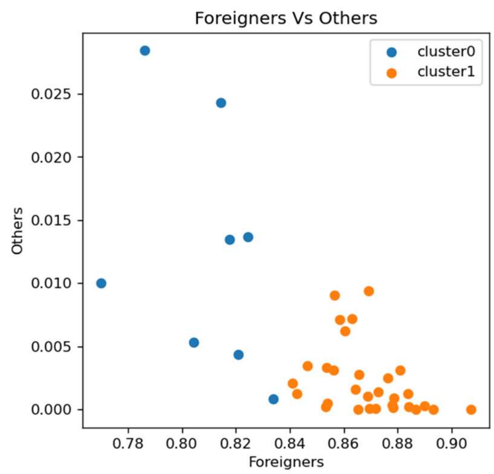

Foreigners Vs Indians
cluster0
cluster1
Indians
Foreigners

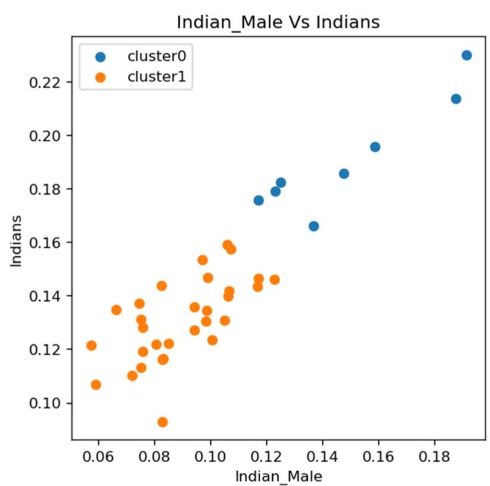

Foreigners Vs Indian_Male
cluster0
cluster1
Indian_Male
Foreigners

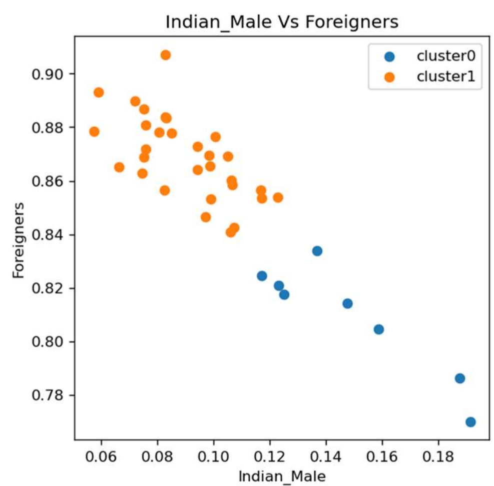

Foreigners Vs Indian_Female
cluster0
cluster1
Indian_Female
Foreigners

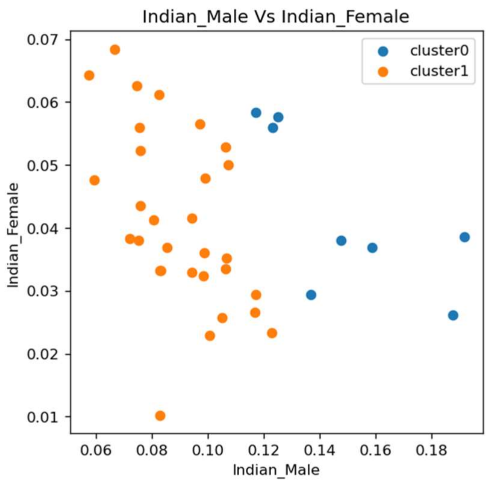

Foreigners Vs Foreigners_Male
cluster0
cluster1
Foreigners_Male
Foreigners

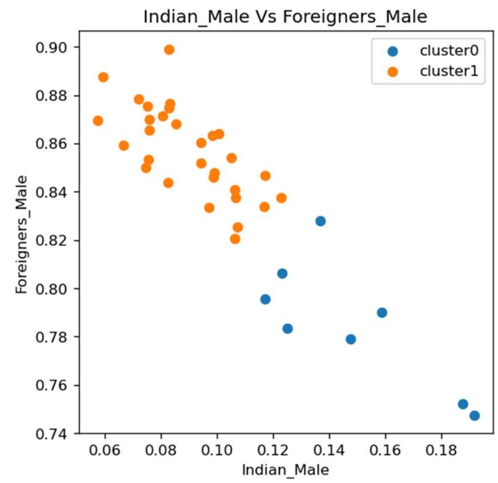

Foreigners Vs Foreigners_Female
cluster0
cluster1
Foreigners_Female
Foreigners

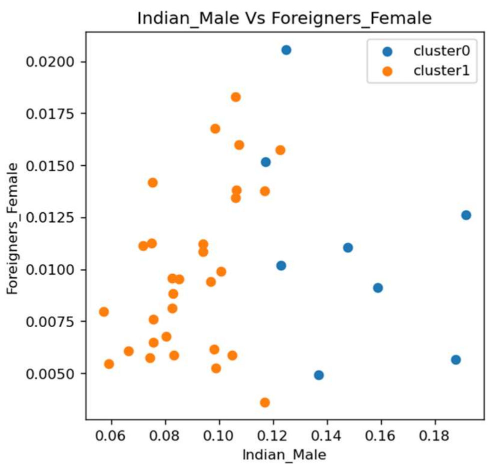

Foreigners Vs Others
cluster0
cluster1
Others
Foreigners

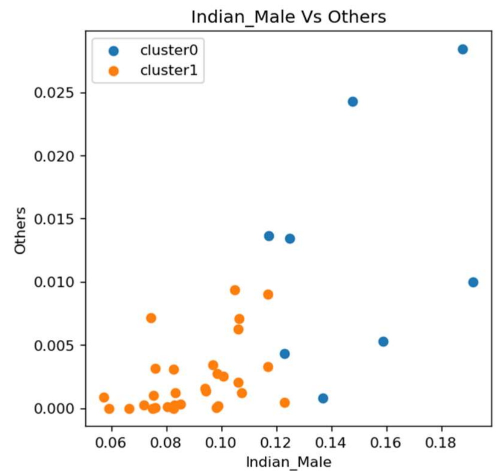

Indian_Male Vs Indians
cluster0
cluster1
Indians
Indian_Male

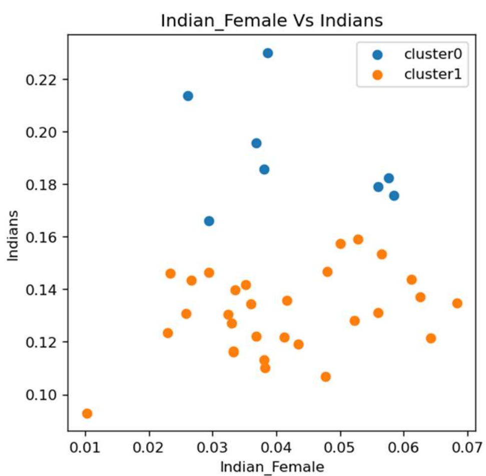

Indian_Male Vs Foreigners
cluster0
cluster1
Foreigners
Indian_Male

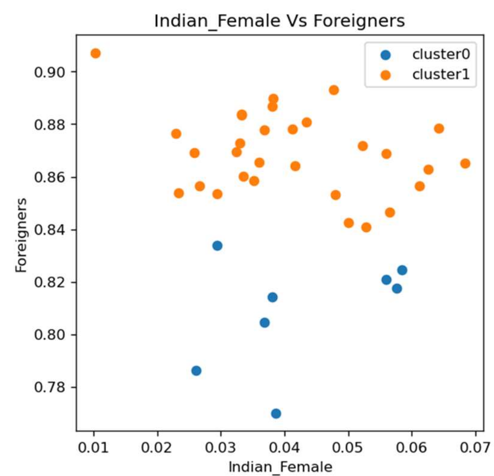

Indian_Male Vs Indian_Female
cluster0
cluster1
Indian_Male
Indian_Female

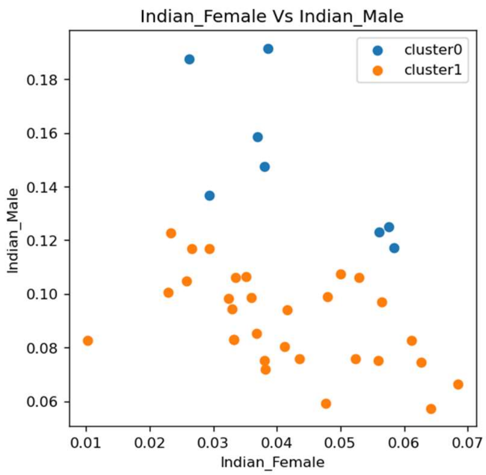

Indian_Male Vs Foreigners_Male
cluster0
cluster1
Indian_Male
Foreigners_Male

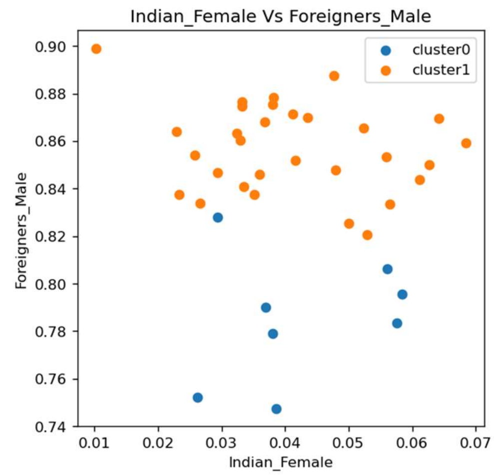

Indian_Male Vs Foreigners_Female
cluster0
cluster1
Indian_Male
Foreigners_Female

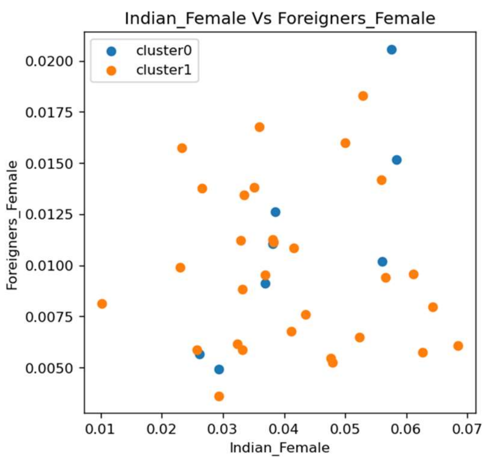

Indian_Male Vs Others
cluster0
cluster1
Indian_Male
Others

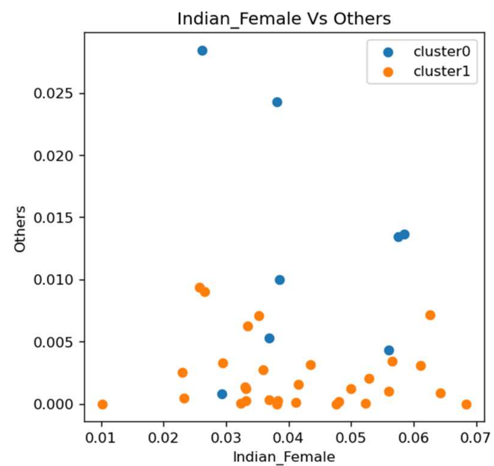

Indian_Female Vs Indians
cluster0
cluster1
Indian_Female
Indians

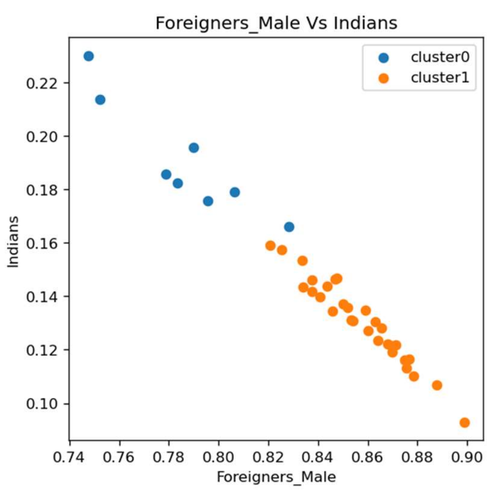

Indian_Female Vs Foreigners
cluster0
cluster1
Indian_Female
Foreigners

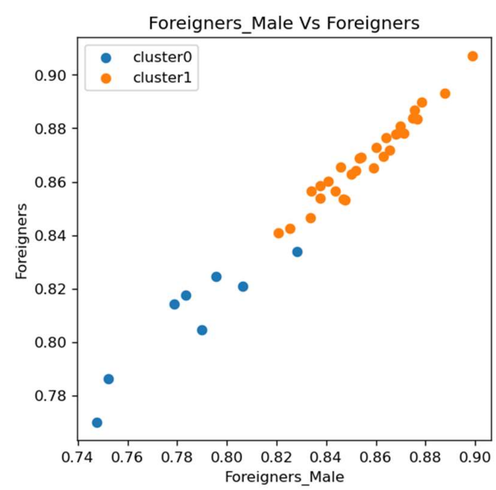

Indian_Female Vs Indian_Male
cluster0
cluster1
Indian_Male
Indian_Female

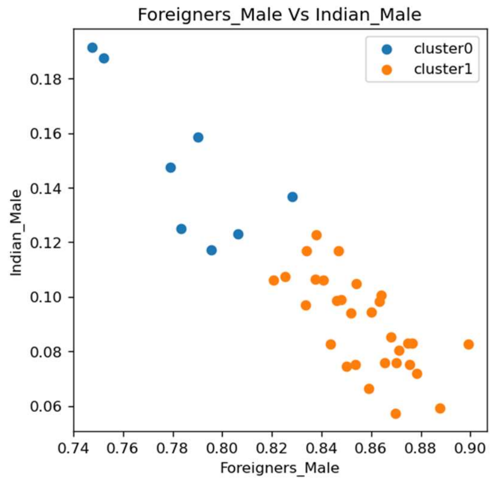

Indian_Female Vs Foreigners_Male
cluster0
cluster1
Foreigners_Male
Indian_Female

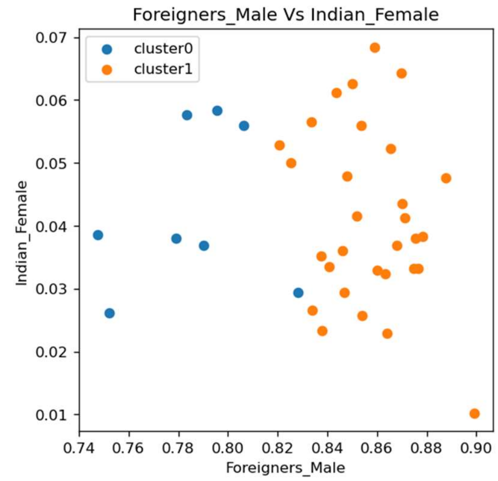

Indian_Female Vs Foreigners_Female
cluster0
cluster1
Foreigners_Female
Indian_Female

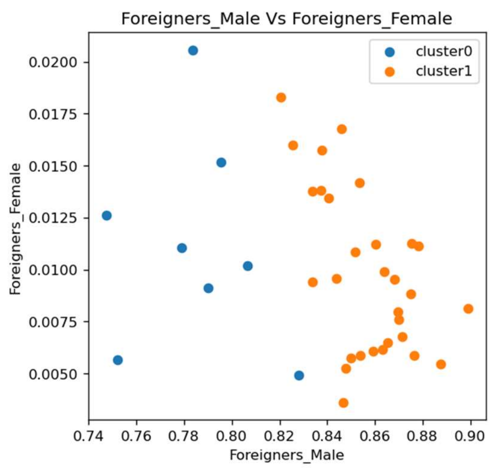

Indian_Female Vs Others
cluster0
cluster1
Others
Indian_Female

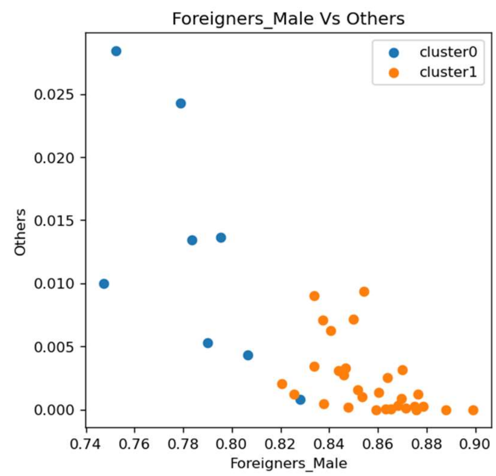

Foreigners_Male Vs Indians
cluster0
cluster1
Indians
Foreigners_Male

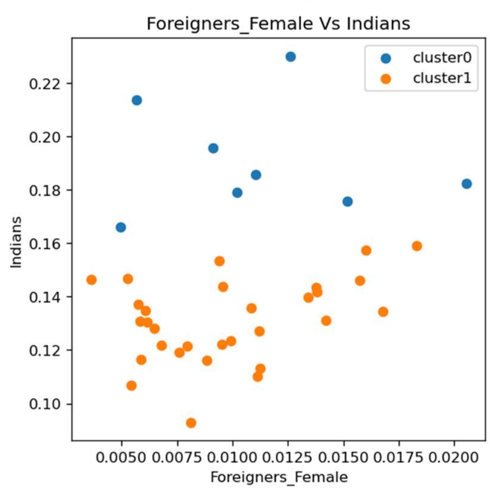

Foreigners_Male Vs Foreigners
cluster0
cluster1
Foreigners
Foreigners_Male

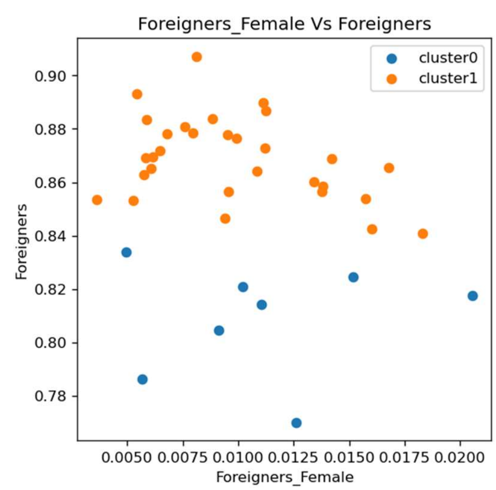

Foreigners_Male Vs Indian_Male
cluster0
cluster1
Indian_Male
Foreigners_Male

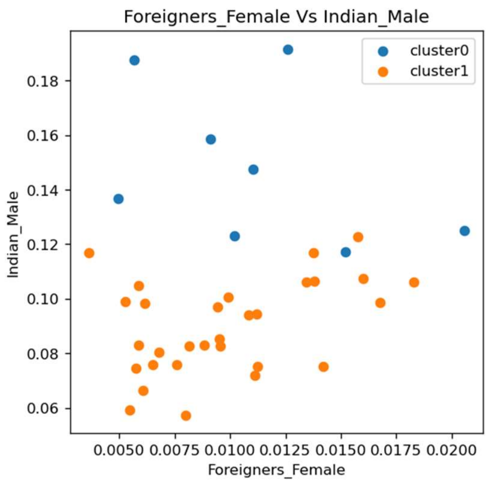

Foreigners_Male Vs Indian_Female
cluster0
cluster1
Indian_Female
Foreigners_Male

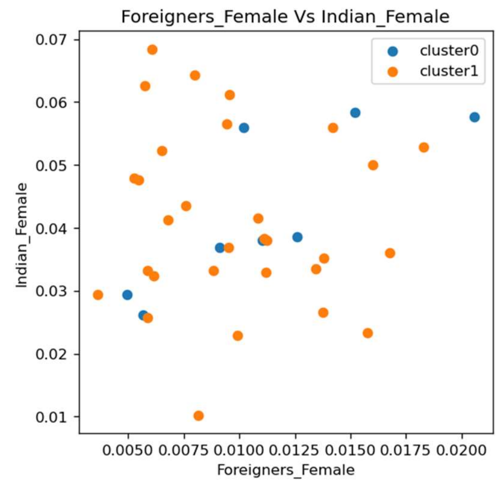

Foreigners_Male Vs Foreigners_Female
cluster0
cluster1
Foreigners_Female
Foreigners_Male

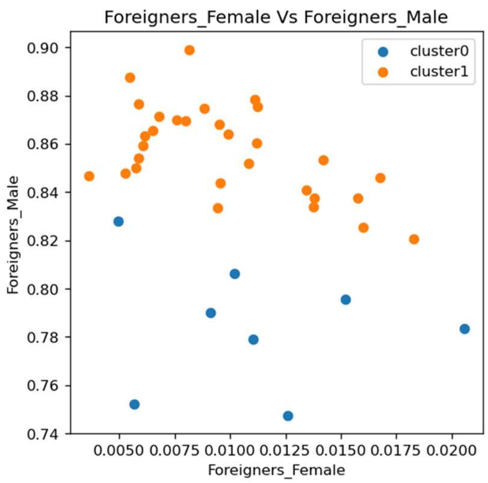

Foreigners_Male Vs Others
cluster0
cluster1
Others
Foreigners_Male

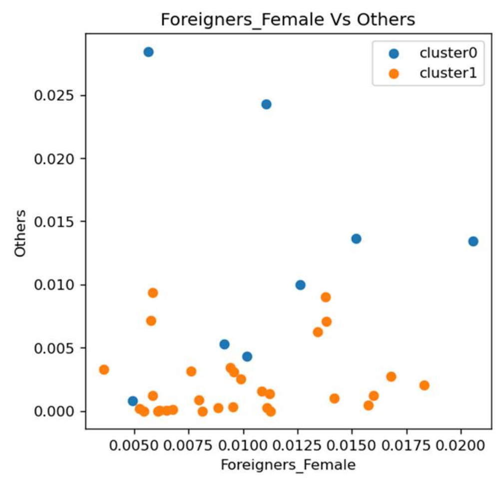

Foreigners_Female Vs Indians
cluster0
cluster1
Indians
Foreigners_Female

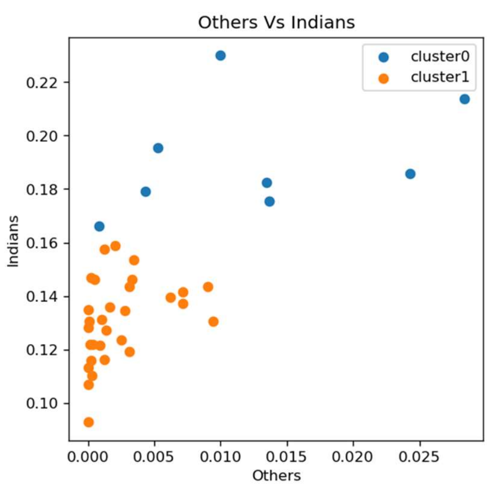

Foreigners_Female Vs Foreigners
cluster0
cluster1
Foreigners
Foreigners_Female

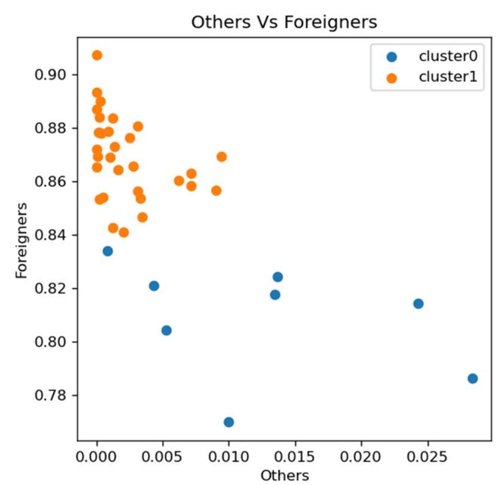

Foreigners_Female Vs Indian_Male
cluster0
cluster1
Indian_Male
Foreigners_Female

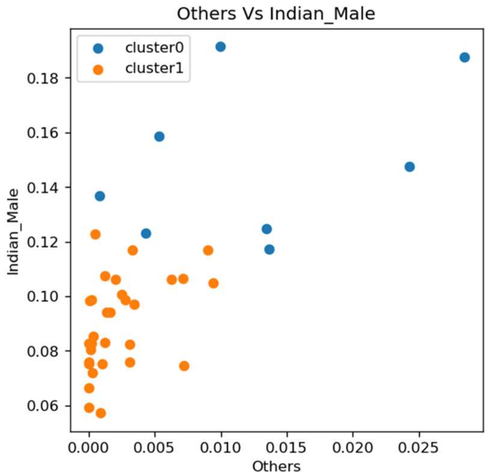

Foreigners_Female Vs Indian_Female
cluster0
cluster1
Indian_Female
Foreigners_Female

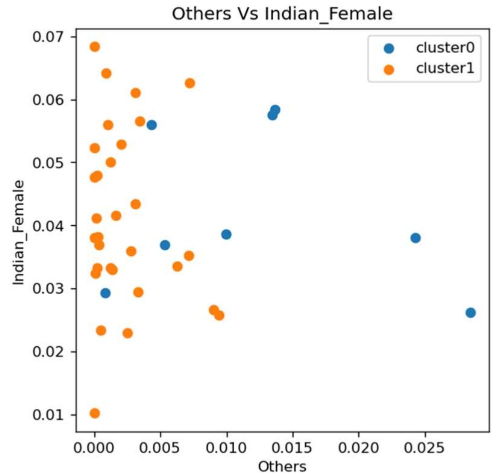

Foreigners_Female Vs Foreigners_Male
cluster0
cluster1
Foreigners_Male
Foreigners_Female

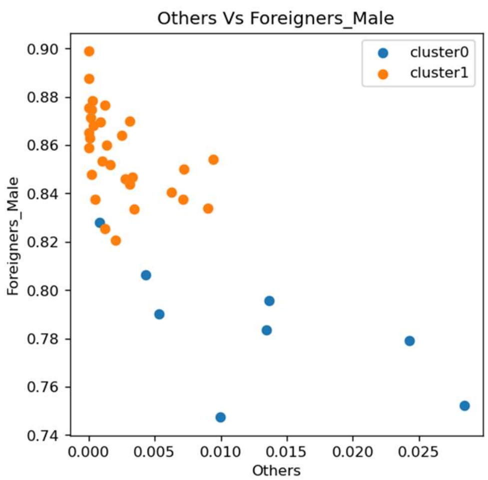

Foreigners_Female Vs Others
cluster0
cluster1
Others
Foreigners_Female

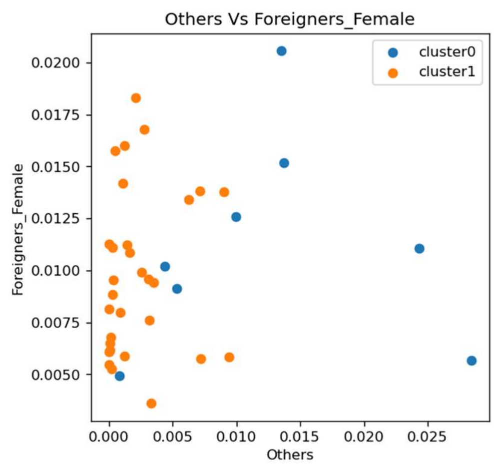

Others Vs Indians
cluster0
cluster1
Indians
Others


Others Vs Foreigners
cluster0
cluster1
Foreigners
Others


Others Vs Indian_Male
cluster0
cluster1
Indian_Male
Others
0.000 0.005 0.010 0.015 0.020 0.025
0.06 0.08 0.10 0.12 0.14 0.16 0.18


Others Vs Indian_Female
cluster0
cluster1
Indian_Female
Others
0.000 0.005 0.010 0.015 0.020 0.025
0.01 0.02 0.03 0.04 0.05 0.06 0.07

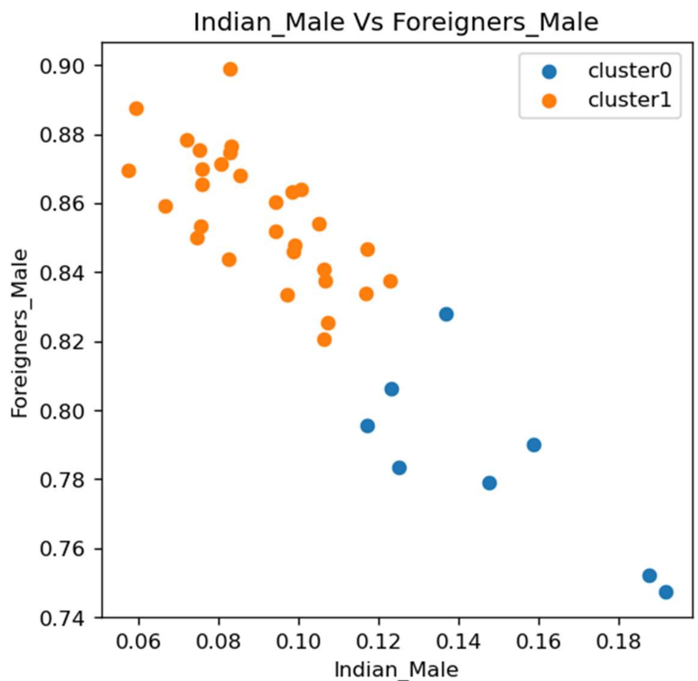

Others Vs Foreigners_Male
cluster0
cluster1
Foreigners_Male
Others
0.000 0.005 0.010 0.015 0.020 0.025
0.74 0.76 0.78 0.80 0.82 0.84 0.86 0.88 0.90


Others Vs Foreigners_Female
cluster0
cluster1
Foreigners_Female
Others
0.000 0.005 0.010 0.015 0.020 0.025
0.0050 0.0075 0.0100 0.0125 0.0150 0.0175 0.0200

Required Cluster Model

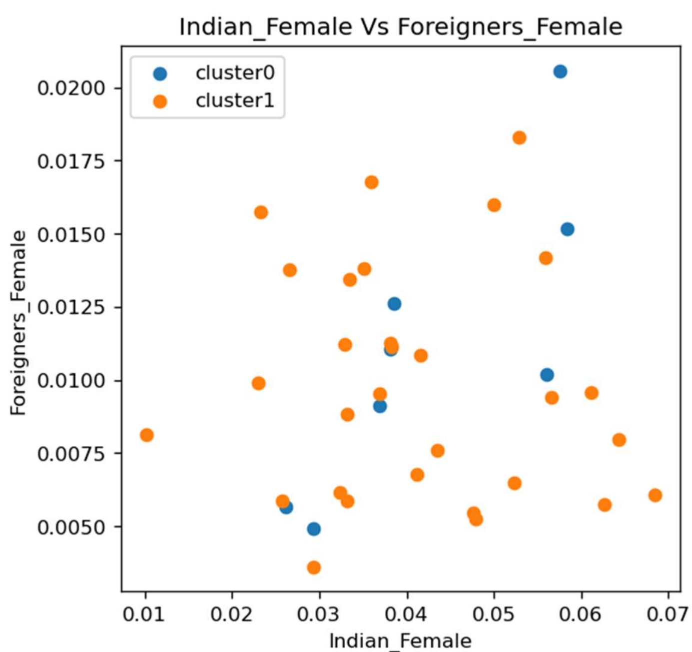
Input	plot_cluster('Indian_Male', 'Foreigners_Male',2)

Output



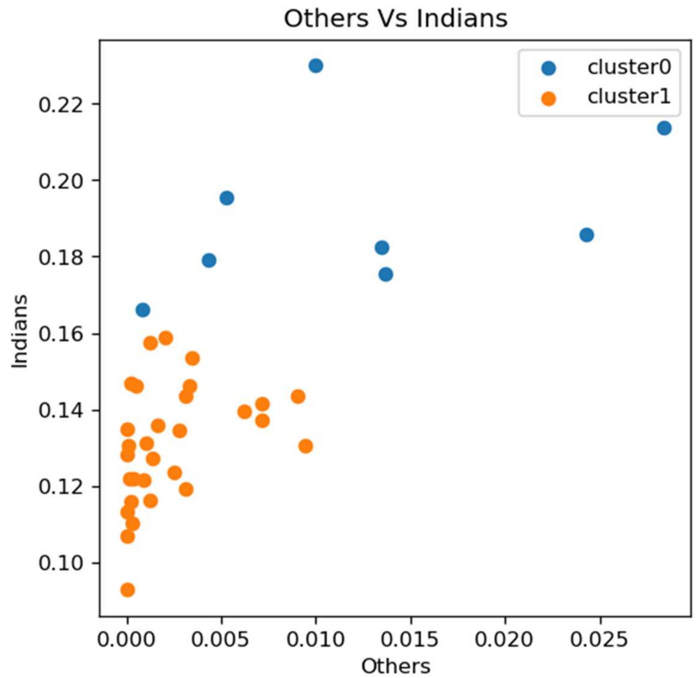
Input	plot_cluster('Indian_Female', 'Foreigners_Female',2)

Output



Input	plot_cluster('Others', 'Indians',2)

Output



Inference

The given data has been segregated into multiple cluster plots. The 2 dimensional cluster plots display 2 variables at a time for every possible variable pair. The final 3, Indian Male vs Foreigner Male, Indian Female vs Foreigner Female and Others vs Indians being the most important cluster plots for the Marketing team to target on.

Depending on this target audience on the basis of region the Marketing department can aptly place the resources required for promotional campaigns pertaining to the target audience which can be found through the cluster plots.