In [1]:
```
import os
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
```

In [2]:
```
survey = pd.read_csv("C:\\Users\\harsh\\OneDrive\\Desktop\\SCMA\\Survey.csv")
```

In [3]:
```
df_int = survey[survey.columns[survey.dtypes == np.int64]]
```

In [4]:
```
df_int.shape
```

Out[4]:
```
(70, 29)
```

In [5]:
```
survey.head()
```

Out[5]:

	City	Sex	Age	Occupation	Monthly Household Income	Income	Planning to Buy a new house	Time Frame	Reasons for buying a house	what type of House	...	4. Availability of domestic help	Time	Size	Budgets	Maintain
0	Bangalore	M	26-35	Private Sector	85,001 to105,000	95000	Yes	6M to 1Yr	Residing	Apartment	...	1	9	1200	72.5	
1	Bangalore	M	46-60	Government/PSU	45,001 to 65,000	55000	Yes	6M to 1Yr	Investment	Apartment	...	2	9	800	32.5	
2	Bangalore	F	46-60	Government/PSU	25,001 to 45,000	35000	Yes	<6 Months	Rental Income	Apartment	...	4	3	400	12.5	
3	Bangalore	M	36-45	Private Sector	>125000	200000	Yes	<6 Months	Investment	Apartment	...	5	3	1600	102.5	
4	Bangalore	M	26-35	Self Employed	85,001 to105,000	95000	Yes	1-2 Yr	Residing	Apartment	...	3	18	800	52.5	

5 rows × 50 columns

In [6]:
```
survey.columns
```

Out[6]:
```
Index(['City', 'Sex', 'Age', 'Occupation', 'Monthly Household Income',
       'Income', 'Planning to Buy a new house', 'Time Frame',
       'Reasons for buying a house', 'what type of House', 'Number of rooms',
       'Size of House', 'Budget', 'Finished/Semi Finished',
       'Influence Decision', 'Maintainance', 'EMI', '1.Proximity to city',
       '2.Proximity to schools', '3. Proximity to transport',
       '4. Proximity to work place', '5. Proximity to shopping',
       '1. Gym/Pool/Sports facility', '2. Parking space', '3.Power back-up',
       '4.Water supply', '5.Security', '1. Exterior look ', '2. Unit size',
       '3. Interior design and branded components',
       '4. Layout plan (Integrated etc.)', '5. View from apartment',
       '1. Price', '2. Booking amount', '3. Equated Monthly Instalment (EMI)',
       '4. Maintenance charges', '5. Availability of loan',
       '1. Builder reputation', '2. Appreciation potential',
       '3. Profile of neighbourhood', '4. Availability of domestic help',
       'Time', 'Size', 'Budgets', 'Maintainances', 'EMI.1', 'ages', 'sex',
       'Finished/Semi Finished.1', 'Influence Decision.1'],
      dtype='object')
```

In [7]:
```
df = survey[['Income','Sex','1.Proximity to city',
       '2.Proximity to schools', '3. Proximity to transport',
       '4. Proximity to work place', '5. Proximity to shopping',
       '1. Gym/Pool/Sports facility', '2. Parking space', '3.Power back-up',
       '4.Water supply', '5.Security', '1. Exterior look ', '2. Unit size',
       '3. Interior design and branded components',
       '4. Layout plan (Integrated etc.)', '5. View from apartment',
       '1. Price', '2. Booking amount', '3. Equated Monthly Instalment (EMI)',
       '4. Maintenance charges', '5. Availability of loan',
       '1. Builder reputation', '2. Appreciation potential',
       '3. Profile of neighbourhood', '4. Availability of domestic help',
       'Time', 'Size', 'Budgets', 'Maintainances', 'EMI.1', 'ages']]
df.head()
```

Out[7]:

	Income	Sex	1.Proximity to city	2.Proximity to schools	3. Proximity to transport	4. Proximity to work place	5. Proximity to shopping	1. Gym/Pool/Sports facility	2. Parking space	3.Power back-up	...	1. Builder reputation	2. Appreciation potential	3. Prof neighbour
0	95000	M	3	5	5	2	1	2	5	3	...	4	5	
1	55000	M	3	5	5	3	1	1	4	2	...	5	4	
2	35000	F	1	2	5	2	1	4	3	2	...	4	4	
3	200000	M	4	5	3	5	4	5	5	4	...	5	4	
4	95000	M	4	2	3	4	3	2	4	3	...	4	3	

5 rows × 32 columns

In [8]:
```
df.isnull().sum().sort_values(ascending=False)
```

Out[8]:
```
Income                                      0
Sex                                         0
EMI.1                                       0
Maintainances                               0
Budgets                                     0
Size                                        0
Time                                        0
4. Availability of domestic help            0
3. Profile of neighbourhood                 0
2. Appreciation potential                   0
1. Builder reputation                       0
5. Availability of loan                     0
4. Maintenance charges                      0
3. Equated Monthly Instalment (EMI)         0
2. Booking amount                           0
1. Price                                    0
5. View from apartment                      0
4. Layout plan (Integrated etc.)            0
3. Interior design and branded components   0
2. Unit size                                0
1. Exterior look                            0
5.Security                                  0
4.Water supply                              0
3.Power back-up                             0
2. Parking space                            0
1. Gym/Pool/Sports facility                 0
5. Proximity to shopping                    0
4. Proximity to work place                  0
3. Proximity to transport                   0
2.Proximity to schools                      0
1.Proximity to city                         0
ages                                        0
dtype: int64
```

In [9]:
```
df.Sex.unique()
```

Out[9]:
```
array(['M', 'F'], dtype=object)
```

In [10]:
```
df.Sex.replace(('M', 'F'), (1, 0), inplace=True)
```

```
C:\Users\harsh\AppData\Local\Temp\ipykernel_11956\428201289.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.Sex.replace(('M', 'F'), (1, 0), inplace=True)
```

In [11]:
```
df.head()
```

Out[11]:

	Income	Sex	1.Proximity to city	2.Proximity to schools	3. Proximity to transport	4. Proximity to work place	5. Proximity to shopping	1. Gym/Pool/Sports facility	2. Parking space	3.Power back-up	...	1. Builder reputation	2. Appreciation potential	3. Prof neighbour
0	95000	1	3	5	5	2	1	2	5	3	...	4	5	
1	55000	1	3	5	5	3	1	1	4	2	...	5	4	
2	35000	0	1	2	5	2	1	4	3	2	...	4	4	
3	200000	1	4	5	3	5	4	5	5	4	...	5	4	
4	95000	1	4	2	3	4	3	2	4	3	...	4	3	

5 rows × 32 columns

In [12]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
```

Out[12]:
```
StandardScaler
StandardScaler()
```

In [13]:
```
scaled_data = scaler.transform(df)
```

In [14]:
```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(scaled_data)
```

Out[14]:
```
PCA
PCA(n_components=2)
```

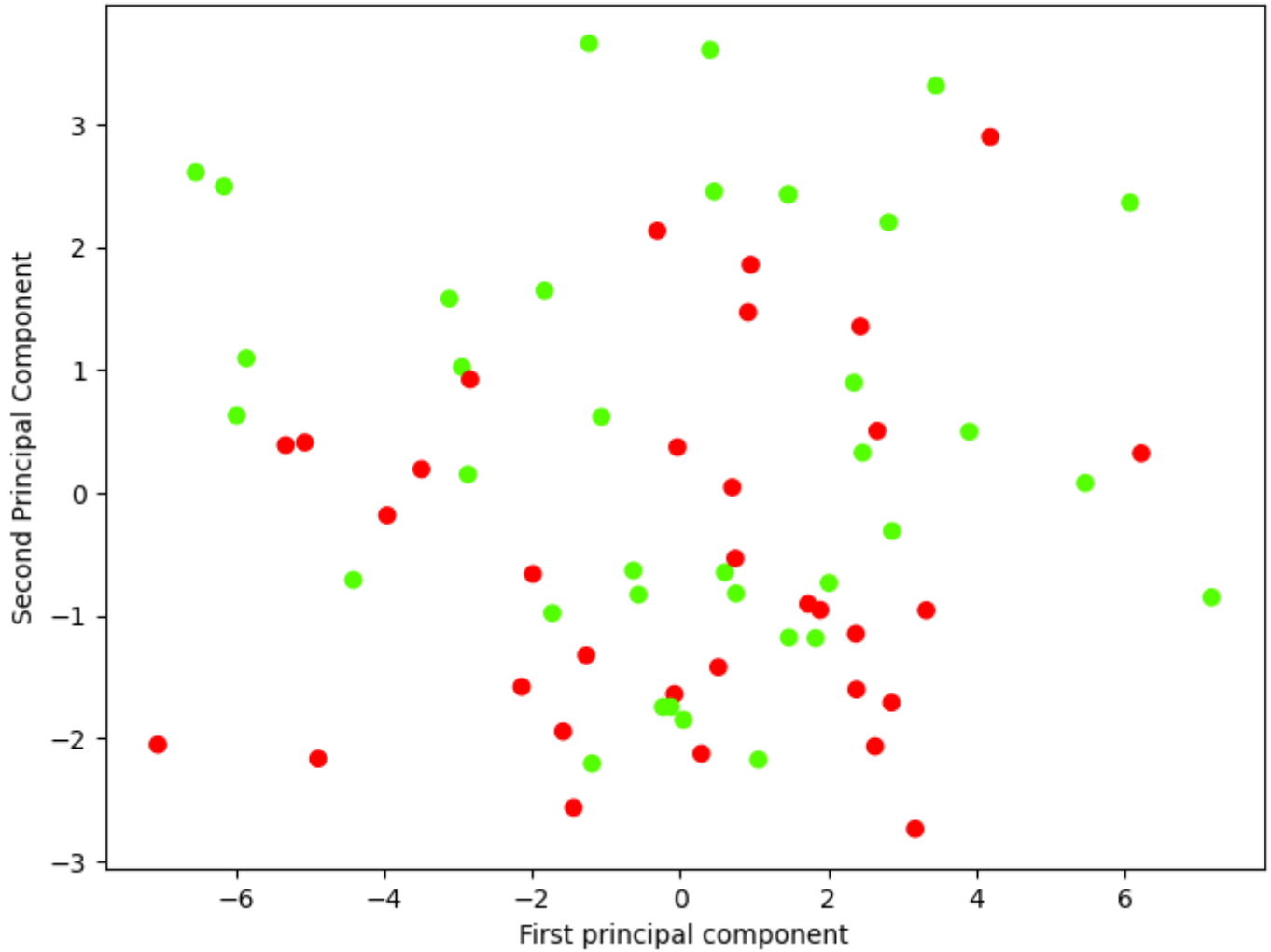
In [15]:
```
x_pca = pca.transform(scaled_data)
```

In [16]:
```
x_pca.shape
```

Out[16]:
```
(70, 2)
```

In [17]:
```
plt.figure(figsize=(8,6))
plt.scatter(x_pca[:,0],x_pca[:,1],c=df['Sex'],cmap='prism')
plt.xlabel('First principal component')
plt.ylabel('Second Principal Component')
```

Out[17]:
```
Text(0, 0.5, 'Second Principal Component')
```



In [18]:
```
pca.components_
```

Out[18]:
```
array([[-0.27822547,  0.00437183, -0.16011319, -0.13828784,  0.01758012,
        -0.05239507, -0.1902369 , -0.16976364, -0.18404669, -0.13648819,
        -0.18593568, -0.1752495 , -0.20150108, -0.057148  , -0.23828056,
        -0.20788117, -0.24143847, -0.11962459, -0.00299985,  0.0291807 ,
         0.05653896,  0.04682637, -0.1864441 , -0.108815  , -0.23046188,
        -0.19611699, -0.02158023, -0.27089073, -0.27648912, -0.27564692,
        -0.27747476, -0.16674494],
       [ 0.03740075,  0.1823039 , -0.15589176,  0.17507134,  0.18324703,
         0.04659357, -0.25355039,  0.08352012,  0.01225996, -0.10099029,
         0.20921906,  0.02493319, -0.33153322,  0.11898123, -0.04256642,
        -0.00692915, -0.08800223,  0.22515981, -0.37208813, -0.18757186,
        -0.24300431, -0.4084159 ,  0.20105514, -0.17048651,  0.13595047,
        -0.15029766, -0.22491526, -0.02870644,  0.00511514,  0.0004829 ,
         0.01761548,  0.04988922]])
```

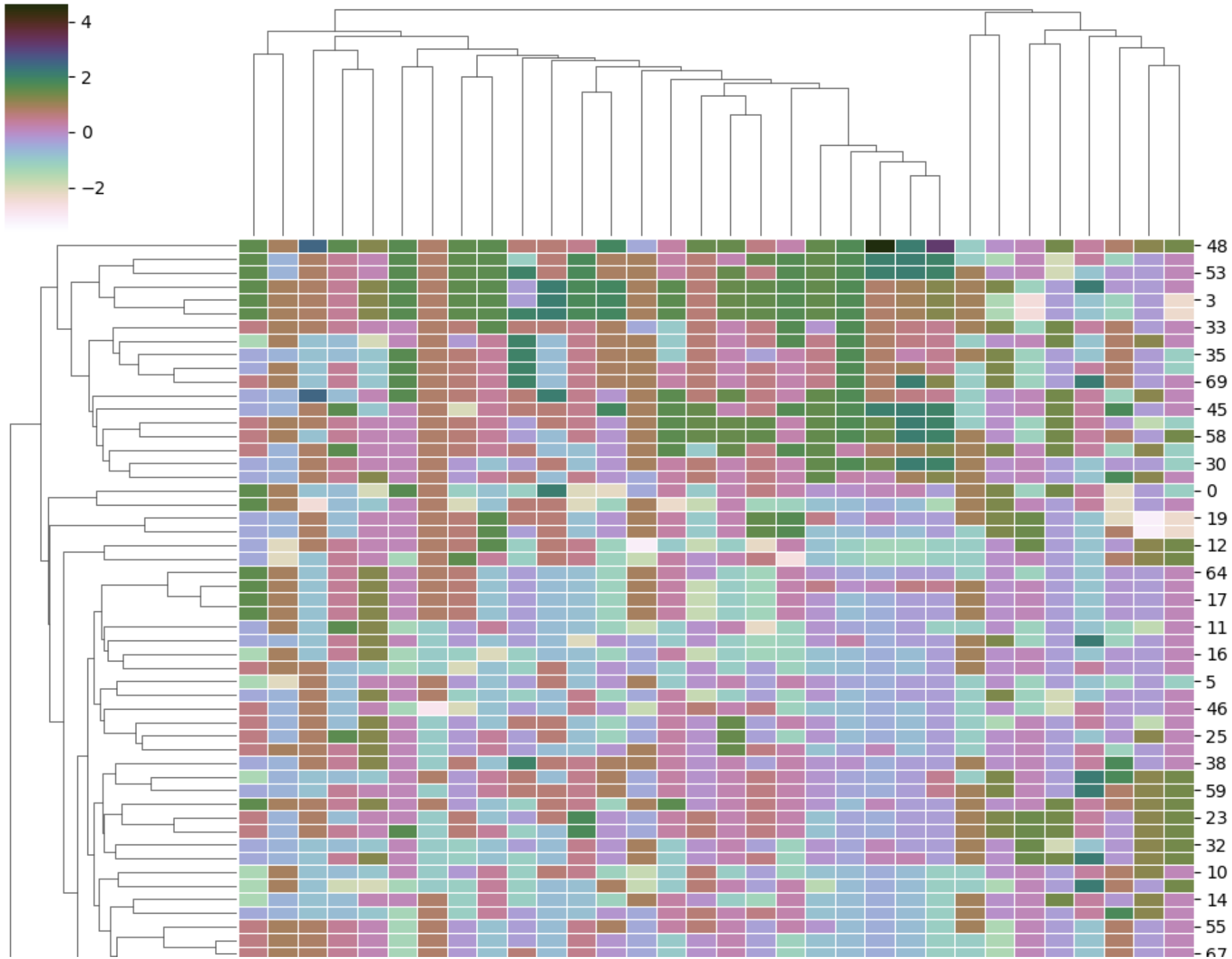
In [19]:
```
from IPython.display import Image
import matplotlib.cm as cm
import seaborn as sn
import matplotlib.pyplot as plt
import graphviz
from mpl_toolkits.mplot3d import Axes3D
from sklearn.cluster import KMeans
plt.rcParams['figure.figsize']=(16,9)
%matplotlib inline
```

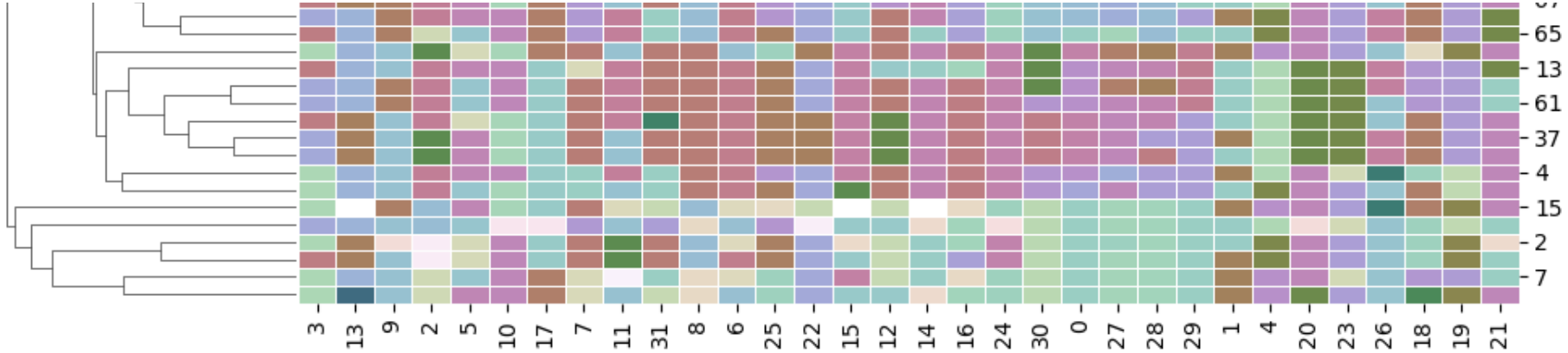
In [20]:
```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
x_scaled=scaler.fit_transform(df)
```

In [21]:
```
cmap= sns.cubehelix_palette(as_cmap=True, rot=-3, light=1 )
```

In [22]:
```
sn.clustermap(x_scaled, cmap=cmap, linewidths=0.5)
```

Out[22]:
```
<seaborn.matrix.ClusterGrid at 0x275e8b87d90>
```


4
2
0
−2
48
53
3
33
35
69
45
58
30
0
19
12
64
17
11
16
5
46
25
38
59
23
32
10
14
55
67



In [23]:
```
cluster_range= range (1,20)
cluster_errors=[]
for num_clusters in cluster_range:
    clusters= KMeans(num_clusters)
    clusters.fit(x_scaled)
    cluster_errors.append(clusters.inertia_)
```

```
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a
memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_
```

```
init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
```

```
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment
```

```
variable OMP_NUM_THREADS=1.
  warnings.warn(
```

In [24]:
```
clusters_df=pd.DataFrame({"num_clusters":cluster_range,"cluster_errors":cluster_errors})
```

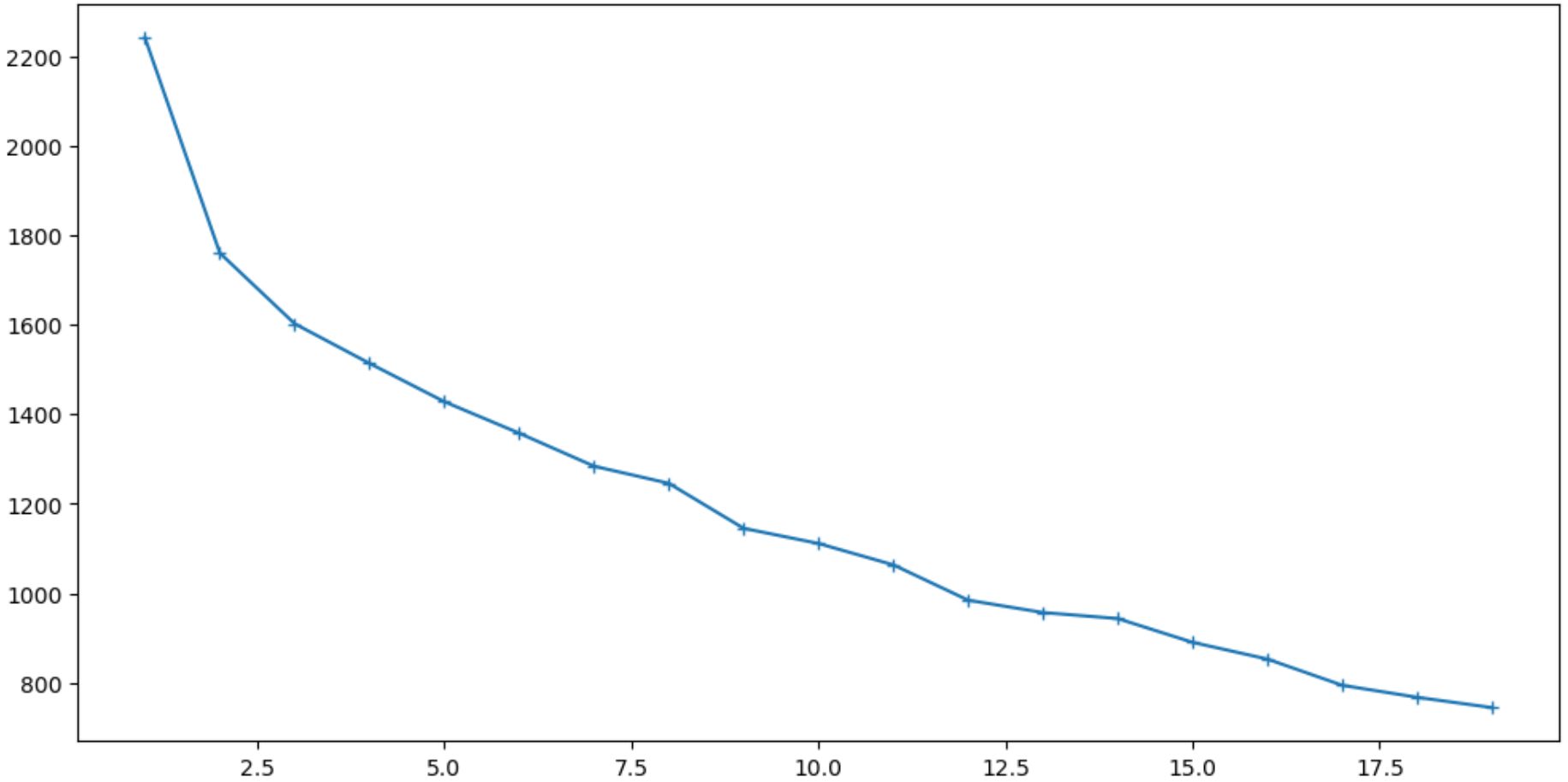
In [25]:
```
clusters_df
```

Out[25]:

	num_clusters	cluster_errors
0	1	2240.000000
1	2	1759.850971
2	3	1602.036801
3	4	1513.620337
4	5	1428.024547
5	6	1357.707722
6	7	1284.030653
7	8	1245.761583
8	9	1145.266677
9	10	1111.493037
10	11	1063.732043
11	12	984.867511
12	13	957.295602
13	14	943.809099
14	15	890.696576
15	16	853.372035
16	17	794.838726
17	18	768.042418
18	19	745.043375

In [26]:
```
plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "+")
```

Out[26]:
```
[<matplotlib.lines.Line2D at 0x275e8c72350>]
```



In [27]:
```
#Initializing KMeans
kmeans = KMeans(n_clusters=3, max_iter=400)
```

In [28]:
```
kmeans.fit(df)
```

```
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarning: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\harsh\OneDrive\Desktop\python\lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarning: KMeans is known to have a memory leak on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OMP_NUM_THREADS=1.
  warnings.warn(
```

Out[28]:

```
KMeans(max_iter=400, n_clusters=3)
```

In [29]:

```
kmeans_labels = kmeans.labels_
kmeans_labels1 = pd.Series(kmeans_labels)
```

In [30]:

```
kmeans_labels1.value_counts()
```

Out[30]:

```
1    37
0    17
2    16
dtype: int64
```

In [31]:

```
labels = kmeans.predict(df)
labels
```

Out[31]:

```
array([0, 1, 1, 2, 0, 1, 2, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1,
       0, 1, 2, 1, 1, 1, 1, 0, 2, 1, 1, 2, 1, 2, 1, 0, 0, 1, 1, 2, 2, 0,
       1, 2, 1, 1, 2, 1, 1, 2, 1, 2, 0, 1, 1, 2, 2, 0, 2, 0, 0, 1, 1, 1,
       0, 1, 1, 2])
```

In [32]:

```
labels = pd.DataFrame({'Cluster':labels})
```

In [33]:

```
cluster_data = pd.concat([df, labels], axis = 1)
cluster_data.head()
```

Out[33]:

	Income	Sex	1.Proximity to city	2.Proximity to schools	3. Proximity to transport	4. Proximity to work place	5. Proximity to shopping	1. Gym/Pool/Sports facility	2. Parking space	3.Power back-up	...	2. Appreciation potential	3. Profile of neighbourhood	Availa dor
0	95000	1	3	5	5	2	1	2	5	3	...	5	4	
1	55000	1	3	5	5	3	1	1	4	2	...	4	3	
2	35000	0	1	2	5	2	1	4	3	2	...	4	4	
3	200000	1	4	5	3	5	4	5	5	4	...	4	5	
4	95000	1	4	2	3	4	3	2	4	3	...	3	4	

5 rows × 33 columns

In [34]:
```
centroids = kmeans.cluster_centers_
centroids
```

Out[34]:
```
array([[9.97058824e+04, 6.47058824e-01, 4.05882353e+00, 3.23529412e+00,
        3.94117647e+00, 3.70588235e+00, 2.52941176e+00, 3.35294118e+00,
        3.76470588e+00, 3.35294118e+00, 3.82352941e+00, 3.64705882e+00,
        3.29411765e+00, 3.29411765e+00, 3.88235294e+00, 4.00000000e+00,
        3.64705882e+00, 4.41176471e+00, 3.05882353e+00, 4.11764706e+00,
        4.29411765e+00, 3.82352941e+00, 4.47058824e+00, 4.41176471e+00,
        4.05882353e+00, 3.29411765e+00, 9.00000000e+00, 1.15294118e+03,
        7.30882353e+01, 4.29411765e+04, 5.79411765e+04, 5.08823529e+01],
       [5.50000000e+04, 4.86486486e-01, 3.29729730e+00, 3.32432432e+00,
        4.10810811e+00, 3.89189189e+00, 2.40540541e+00, 2.83783784e+00,
        3.24324324e+00, 3.43243243e+00, 3.64864865e+00, 3.40540541e+00,
        2.64864865e+00, 3.35135135e+00, 3.54054054e+00, 3.45945946e+00,
        2.83783784e+00, 4.45945946e+00, 3.21621622e+00, 4.24324324e+00,
        4.00000000e+00, 4.05405405e+00, 4.02702703e+00, 4.00000000e+00,
        3.45945946e+00, 2.67567568e+00, 6.32432432e+00, 7.35135135e+02,
        3.52027027e+01, 2.00032432e+04, 2.94594595e+04, 3.71081081e+01],
       [2.00000000e+05, 5.00000000e-01, 3.93750000e+00, 3.93750000e+00,
        4.12500000e+00, 3.87500000e+00, 3.25000000e+00, 4.06250000e+00,
        3.93750000e+00, 3.81250000e+00, 4.62500000e+00, 4.37500000e+00,
        4.25000000e+00, 3.62500000e+00, 4.50000000e+00, 4.25000000e+00,
        4.31250000e+00, 5.00000000e+00, 3.31250000e+00, 4.12500000e+00,
        3.31250000e+00, 3.62500000e+00, 4.87500000e+00, 4.31250000e+00,
        4.50000000e+00, 4.06250000e+00, 7.87500000e+00, 1.97500000e+03,
        1.21562500e+02, 7.43750000e+04, 7.20312500e+04, 5.40625000e+01]])
```

In [35]:
```
from factor_analyzer import FactorAnalyzer
import matplotlib.pyplot as plt
```

In [36]:
```
from factor_analyzer.factor_analyzer import calculate_bartlett_sphericity
from factor_analyzer.factor_analyzer import calculate_kmo
```

In [37]:
```
Ice = pd.read_csv("C:\\Users\\harsh\\OneDrive\\Desktop\\SCMA\\icecream.csv")
```

In [38]:
```
Ice.head()
```

Out[38]:

	Brand	Price	Availability	Taste	Flavour	Consistency	Shelflife
0	Amul	4	5	4	3	4	3
1	Nandini	3	2	3	2	3	3
2	Vadilal	2	2	4	3	4	4
3	Vijaya	3	1	3	5	3	4
4	Dodla	3	3	3	4	4	3

In [52]:
```
data=Ice.drop(['Brand'],axis=1)
```

In [54]:
```
Ice.shape
```

Out[54]:
```
(10, 7)
```

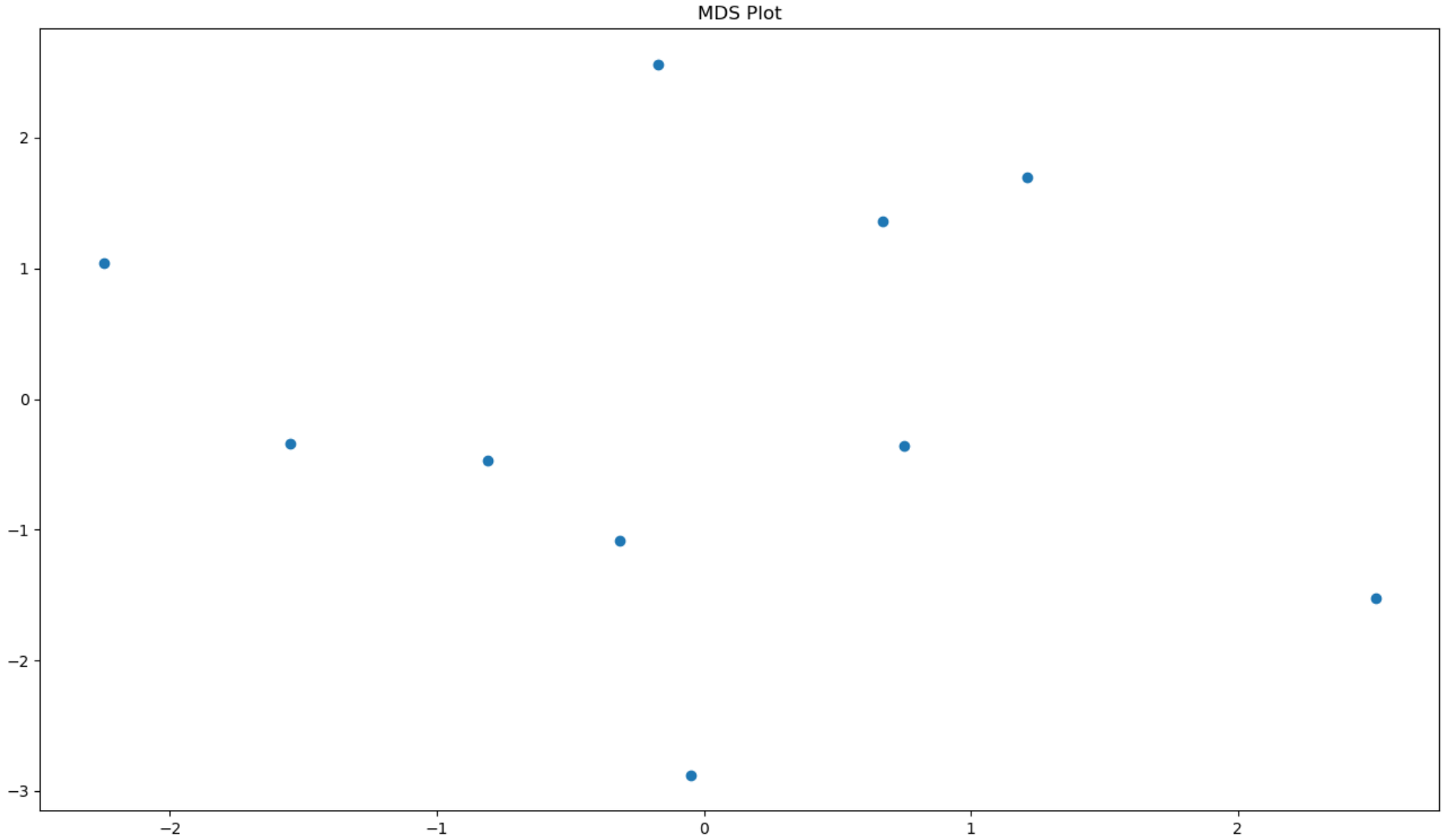
In [55]:
```
import matplotlib.pyplot as plt
from sklearn.manifold import MDS
```

In [58]:
```
from sklearn.metrics.pairwise import manhattan_distances, euclidean_distances
```

In [59]:
```
dist_euclid = euclidean_distances(data)
```

In [67]:
```
mds = MDS(n_components=2, dissimilarity='precomputed', normalized_stress='auto')
mds_result = mds.fit_transform(dist_euclid)
```

In [69]:
```
plt.scatter(mds_result[:, 0], mds_result[:, 1])
plt.title("MDS Plot")
plt.show()
```



In [70]:
```
mds_result.shape
```

Out[70]:
```
(10, 2)
```

In [39]:
```
#remove brand name
df1 = Ice.iloc[:,1:]
```

In [40]:
```
chi_sq_value, p_value = calculate_bartlett_sphericity(df1)
chi_sq_value, p_value
```

Out[40]:
```
(33.31555432825402, 0.004244035879235458)
```

In [41]:
```
#ideally for FA we need to have this more than 0.8
kmo_vars, kmo_model = calculate_kmo(df1)
kmo_model
```

Out[41]:
```
0.4674409270124773
```

In [42]:
```
from sklearn import preprocessing
```

In [43]:
```
#FA
fa = FactorAnalyzer()
fa.fit(df1)
eigen_values, vectors = fa.get_eigenvalues()
fa.set_params(n_factors = 3, rotation = 'varimax')
loadings = fa.loadings_
print(loadings)
```

```
[[ 1.26974447e-02  1.07476353e+00  1.20212366e-01]
 [ 1.05328502e+00  2.50174895e-01 -9.49231823e-02]
 [ 6.49989654e-01 -1.84573331e-01  3.13839890e-01]
 [-8.57715187e-04  1.49262252e-01  7.26693102e-01]
 [ 7.74647452e-01 -1.24265263e-01 -5.78307625e-02]
 [-1.33156114e-01 -3.64502971e-01  7.75760409e-01]]
```

In [44]:
```
#getting variance of factors

pd.DataFrame(fa.get_factor_variance(),index = ['Variance', 'Prop variance', 'Cum_var'])
```

Out[44]:

	0	1	2
Variance	2.149867	1.422355	1.255188
Prop variance	0.358311	0.237059	0.209198
Cum_var	0.358311	0.595370	0.804568

In [45]:
```
fact = pd.DataFrame(fa.loadings_, index = df1.columns)
```

In [46]:
```
fact.columns = ['F1', 'F2', 'F3']
```

In [47]:
```
fact
```

Out[47]:

	F1	F2	F3
Price	0.012697	1.074764	0.120212
Availability	1.053285	0.250175	-0.094923
Taste	0.649990	-0.184573	0.313840
Flavour	-0.000858	0.149262	0.726693
Consistency	0.774647	-0.124265	-0.057831
Shelflife	-0.133156	-0.364503	0.775760

In [48]:
```
com = pd.DataFrame(fa.get_communalities(), index = df1.columns,
                   columns=[['communalities']])
```

In [49]:
```
com
```

Out[49]:

	communalities
Price	1.169729
Availability	1.181007
Taste	0.555049
Flavour	0.550363
Consistency	0.618865
Shelflife	0.752397

In [50]:
```
fact['Communality'] = com['communalities']
```

In [51]:
```
fact
```

Out[51]:

	F1	F2	F3	Communality
Price	0.012697	1.074764	0.120212	1.169729
Availability	1.053285	0.250175	-0.094923	1.181007
Taste	0.649990	-0.184573	0.313840	0.555049
Flavour	-0.000858	0.149262	0.726693	0.550363
Consistency	0.774647	-0.124265	-0.057831	0.618865
Shelflife	-0.133156	-0.364503	0.775760	0.752397

In [ ]: