Clustering and PCA

Machine Learning - 1

ANIL KUMAR PGBPDSBA AUG'2023

Contents

1.0 Problem Statement – Digital Ads Data

Digital Ads Data:

The ads24x7 is a Digital Marketing company which has now got seed funding of $10 Million. They are expanding their wings in Marketing Analytics. They collected data from their Marketing Intelligence team and now wants you (their newly appointed data analyst) to segment type of ads based on the features provided. Use Clustering procedure to segment ads into homogeneous groups.

The following three features are commonly used in digital marketing:
CPM = (Total Campaign Spend / Number of Impressions) * 1,000. Note that the Total Campaign Spend refers to the 'Spend' Column in the dataset and the Number of Impressions refers to the 'Impressions' Column in the dataset.

CPC = Total Cost (spend) / Number of Clicks. Note that the Total Cost (spend) refers to the 'Spend' Column in the dataset and the Number of Clicks refers to the 'Clicks' Column in the dataset.

CTR = Total Measured Clicks / Total Measured Ad Impressions x 100. Note that the Total Measured Clicks refers to the 'Clicks' Column in the dataset and the Total Measured Ad Impressions refers to the 'Impressions' Column in the dataset.

The Data Dictionary and the detailed description of the formulas for CPM, CPC and CTR are given in the sheet 2 of the Clustering Clean ads_data Excel File.

Perform the following in given order:
1.Read the data and perform basic analysis such as printing a few rows (head and tail), info, data summary, null values duplicate values, etc.
2.Treat missing values in CPC, CTR and CPM using the formula given. You may refer to the Bank_KMeans Solution File to understand the coding behind treating the missing values using a specific formula. You have to basically create an user defined function and then call the function for imputing.
3.Check if there are any outliers.
4.Do you think treating outliers is necessary for K-Means clustering? Based on your judgement decide whether to treat outliers and if yes, which method to employ. (As an analyst your judgement may be different from another analyst).
5.Perform z-score scaling and discuss how it affects the speed of the algorithm.
6.Perform clustering and do the following:
7.Perform Hierarchical by constructing a Dendrogram using WARD and Euclidean distance.
8.Make Elbow plot (up to n=10) and identify optimum number of clusters for k-means algorithm.
Print silhouette scores for up to 10 clusters and identify optimum number of clusters.
9.Profile the ads based on optimum number of clusters using silhouette score and your domain understanding
[Hint: Group the data by clusters and take sum or mean to identify trends in clicks, spend, revenue, CPM, CTR, & CPC based on Device Type. Make bar plots.]
10.Conclude the project by providing summary of your learnings.

Executive Summary

The ads24x7 is a Digital Marketing company which has now got seed funding of $10 Million. They are expanding their wings in Marketing Analytics. They collected data from their Marketing Intelligence team and now wants you (their newly appointed data analyst) to segment type of ads based on the features provided. Use Clustering procedure to segment ads into homogeneous groups.

Data Dictionary

Sl_No: Primary key of the records
Customer Key: Customer identification number
Average Credit Limit: Average credit limit of each customer for all credit cards
Total credit cards: Total number of credit cards possessed by the customer
Total visits bank: Total number of visits that customer made (yearly) personally to the bank
Total visits online: Total number of visits or online logins made by the customer (yearly)
Total calls made: Total number of calls made by the customer to the bank or its customer service department (yearly)

1.1 EDA- Exploratory Data Analysis

- Checking the shape of the dataset

```
There are 23066 rows and 19 columns.
```

- Displaying few rows of the dataset

	Timestamp	InventoryType	Ad - Length	Ad- Width	Ad Size	Ad Type	Platform	Device Type	Format	Available_Impressions	Matched_Queries	Impressions	Clicks	Spend	Fee	Revenue	CTR	CPM	CPC
0	2020-9-2-17	Format1	300	250	75000	Inter222	Video	Desktop	Display	1806	325	323	1	0.0	0.35	0.0	0.0031	0.0	0.0
1	2020-9-2-10	Format1	300	250	75000	Inter227	App	Mobile	Video	1780	285	285	1	0.0	0.35	0.0	0.0035	0.0	0.0
2	2020-9-1-22	Format1	300	250	75000	Inter222	Video	Desktop	Display	2727	356	355	1	0.0	0.35	0.0	0.0028	0.0	0.0
3	2020-9-3-20	Format1	300	250	75000	Inter228	Video	Mobile	Video	2430	497	495	1	0.0	0.35	0.0	0.0020	0.0	0.0
4	2020-9-4-15	Format1	300	250	75000	Inter217	Web	Desktop	Video	1218	242	242	1	0.0	0.35	0.0	0.0041	0.0	0.0

- Checking the data types of the columns for the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 23066 entries, 0 to 23065
Data columns (total 19 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Timestamp              23066 non-null  object 
 1   InventoryType          23066 non-null  object 
 2   Ad - Length            23066 non-null  int64  
 3   Ad- Width              23066 non-null  int64  
 4   Ad Size                23066 non-null  int64  
 5   Ad Type                23066 non-null  object 
 6   Platform               23066 non-null  object 
 7   Device Type            23066 non-null  object 
 8   Format                 23066 non-null  object 
 9   Available_Impressions  23066 non-null  int64  
 10  Matched_Queries        23066 non-null  int64  
 11  Impressions            23066 non-null  int64  
 12  Clicks                 23066 non-null  int64  
 13  Spend                  23066 non-null  float64
 14  Fee                    23066 non-null  float64
 15  Revenue                23066 non-null  float64
 16  CTR                    18330 non-null  float64
 17  CPM                    18330 non-null  float64
 18  CPC                    18330 non-null  float64
dtypes: float64(6), int64(7), object(6)
memory usage: 3.3+ MB
```

- Statistical summary of the dataset

There are 6 object variables, 7 integer type and 6 float

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Ad - Length	23066.0	3.851631e+02	2.336514e+02	120.0000	120.000000	300.00000	7.200000e+02	728.00
Ad- Width	23066.0	3.378960e+02	2.030929e+02	70.0000	250.000000	300.00000	6.000000e+02	600.00
Ad Size	23066.0	9.667447e+04	6.153833e+04	33600.0000	72000.000000	72000.00000	8.400000e+04	216000.00
Available_Impressions	23066.0	2.432044e+06	4.742888e+06	1.0000	33672.250000	483771.00000	2.527712e+06	27592861.00
Matched_Queries	23066.0	1.295099e+06	2.512970e+06	1.0000	18282.500000	258087.50000	1.180700e+06	14702025.00
Impressions	23066.0	1.241520e+06	2.429400e+06	1.0000	7990.500000	225290.00000	1.112428e+06	14194774.00
Clicks	23066.0	1.067852e+04	1.735341e+04	1.0000	710.000000	4425.00000	1.279375e+04	143049.00
Spend	23066.0	2.706626e+03	4.067927e+03	0.0000	85.180000	1425.12500	3.121400e+03	26931.87
Fee	23066.0	3.351231e-01	3.196322e-02	0.2100	0.330000	0.35000	3.500000e-01	0.35
Revenue	23066.0	1.924252e+03	3.105238e+03	0.0000	55.365375	926.33500	2.091338e+03	21276.18
CTR	18330.0	7.366054e-02	7.515992e-02	0.0001	0.002600	0.08255	1.300000e-01	1.00
CPM	18330.0	7.672045e+00	6.481391e+00	0.0000	1.710000	7.66000	1.251000e+01	81.56
CPC	18330.0	3.510606e-01	3.433338e-01	0.0000	0.090000	0.16000	5.700000e-01	7.26

➢ Checking the Duplicate Values

No Duplicate Values found

```
[11] df.duplicated().sum()
```

```
0
```

➢ Checking the number of unique values in each column

```python
# checking the number of unique values in each column
data.nunique()
```

```
Timestamp                  2018
InventoryType                 7
Ad - Length                   6
Ad- Width                     5
Ad Size                       7
Ad Type                      14
Platform                      3
Device Type                   2
Format                        2
Available_Impressions     21560
Matched_Queries           20919
Impressions               20405
Clicks                    12752
Spend                     20467
Fee                           7
Revenue                   20578
CTR                        2066
CPM                        2084
CPC                         194
dtype: int64
```

➢ Checking for Missing Values

```python
[13] # checking for missing values
df.isnull().sum()
```

```
Timestamp                   0
InventoryType               0
Ad - Length                 0
Ad- Width                   0
Ad Size                     0
Ad Type                     0
Platform                    0
Device Type                 0
Format                      0
Available_Impressions       0
Matched_Queries             0
Impressions                 0
Clicks                      0
Spend                       0
Fee                         0
Revenue                     0
CTR                      4736
CPM                      4736
CPC                      4736
dtype: int64
```

1.2 Treatment of missing values in CPC, CTR and CPM using the formula given

- CPC = Total Cost (spend) / Number of Clicks •
- CTR = Total Measured Clicks / Total Measured Ad Impressions x 100
- CPM = (Total Campaign Spend / Number of Impressions) * 1,000

```
def calculate_CPC(x):
  Total_Cost=df.Spend
  Number_of_clicks=df.Clicks
  CPC = (Total_Cost/(Number_of_clicks))
  return CPC
df['CPC'] = df[['CPC']].apply(lambda x: calculate_CPC(x))
```

```
[16] def calculate_CTR(x):
       Total_Measured_Clicks=df.Clicks
       Total_Measured_Impressions=df.Available_Impressions
       CTR = (Total_Measured_Clicks/(Total_Measured_Impressions)*100)
       return CTR
     df['CTR'] = df[['CTR']].apply(lambda x: calculate_CTR(x))
```

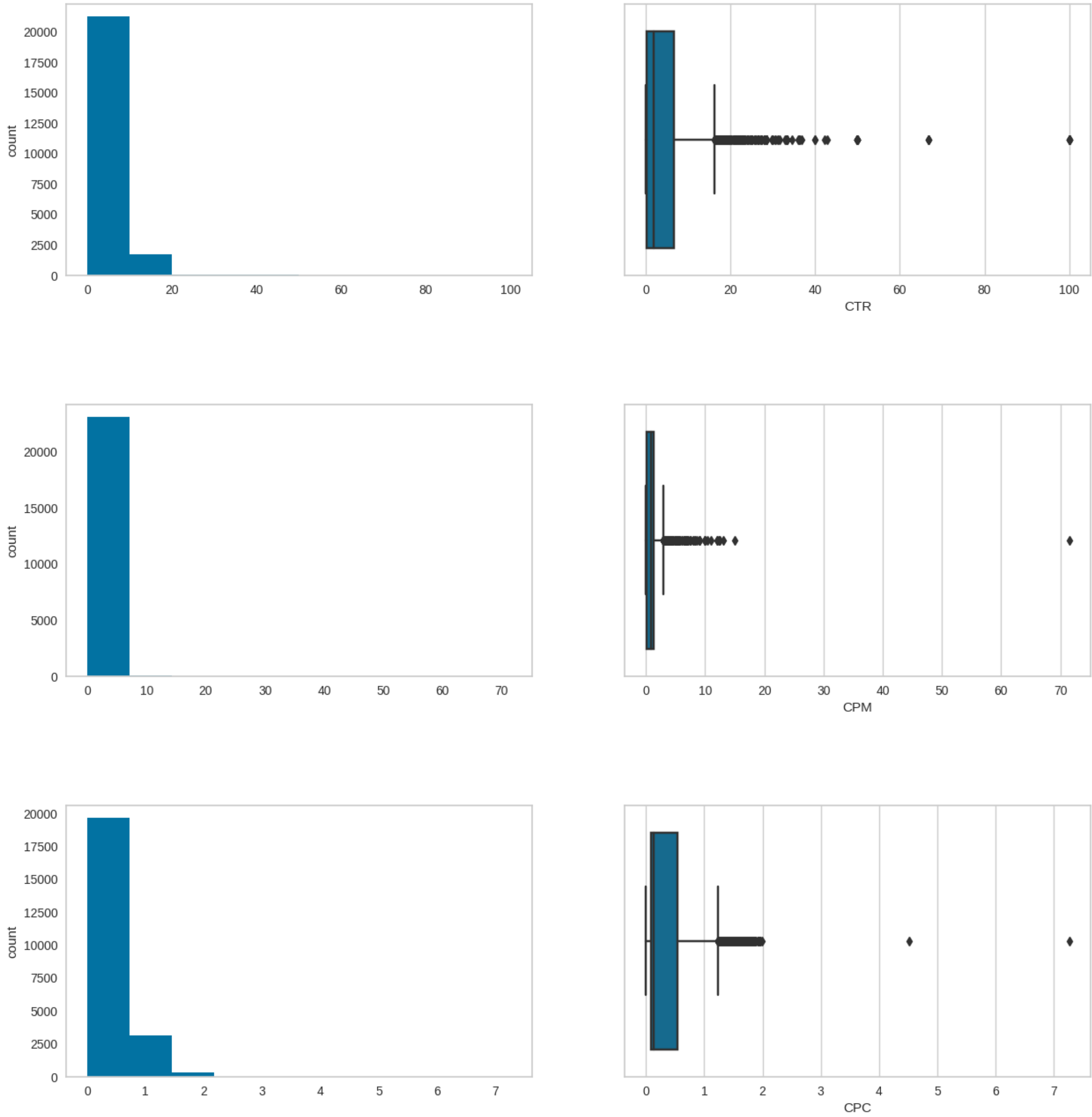
```
[17] def calculate_CPM(x):
       Total_Campaign_Spend=df.Spend
       Total_Measured_Impressions=df.Impressions
       CPM = (Total_Campaign_Spend/(Total_Measured_Impressions)*100)
       return CPM
     df['CPM'] = df[['CPM']].apply(lambda x: calculate_CPM(x))
```

- Checking for Missing Values after treatment

```
# Check once again null values
df.isnull().sum()
```

```
Timestamp                  0
InventoryType              0
Ad - Length                0
Ad- Width                  0
Ad Size                    0
Ad Type                    0
Platform                   0
Device Type                0
Format                     0
Available_Impressions      0
Matched_Queries            0
Impressions                0
Clicks                     0
Spend                      0
Fee                        0
Revenue                    0
CTR                        0
CPM                        0
CPC                        0
dtype: int64
```

1.3 Outliers Detection



1.4 Do you think treating outliers is necessary for K-Means clustering? Based on your judgement decide whether to treat outliers and if yes, which method to employ. (As an analyst your judgement may be different from another analyst).

It depends on the specific case and the domain knowledge. If the outliers are caused by errors in data collection or data entry, then it may be necessary to remove them. If the outliers are caused by actual extreme values in the data, then it may be necessary to keep them.

Here we are not removing any outliers

1.5 Perform Z-Score scaling and discuss how it affects the speed of the algorithm.

Z-score scaling standardizes the data by subtracting the mean and dividing by the standard deviation. This can make the data more comparable and can improve the performance of clustering algorithms. However, it can also slow down the algorithm if the dataset is very large.

```
[45] from sklearn.preprocessing import StandardScaler
     scaler=StandardScaler()
     data_scaled=pd.DataFrame(scaler.fit_transform(df1), columns=df1.columns)
```

```
[46] data_scaled.head()
```

	CTR	CPM	CPC
0	-0.747206	-0.927054	-0.986615
1	-0.747045	-0.927054	-0.986615
2	-0.750919	-0.927054	-0.986615
3	-0.750029	-0.927054	-0.986615
4	-0.741898	-0.927054	-0.986615

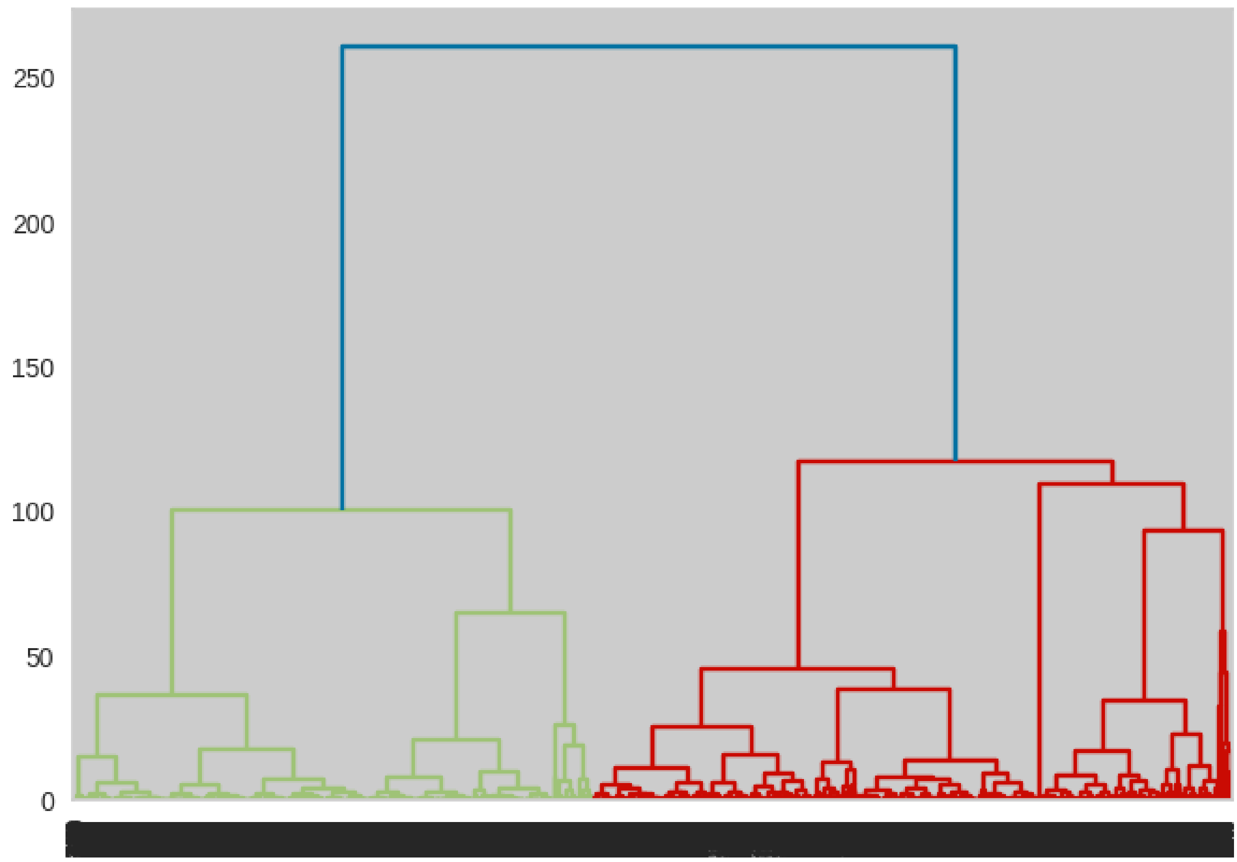
1.6 Perform Clustering

1.6(A)Perform Hierarchical by constructing a Dendrogram using WARD and Euclidean distance

```
from scipy.cluster.hierarchy import linkage, dendrogram
import matplotlib.pyplot as plt

# Perform linkage
linkage_matrix = linkage(data_scaled, method='ward', metric='euclidean')

# Plot the dendrogram
dendrogram(linkage_matrix)
plt.show()
```

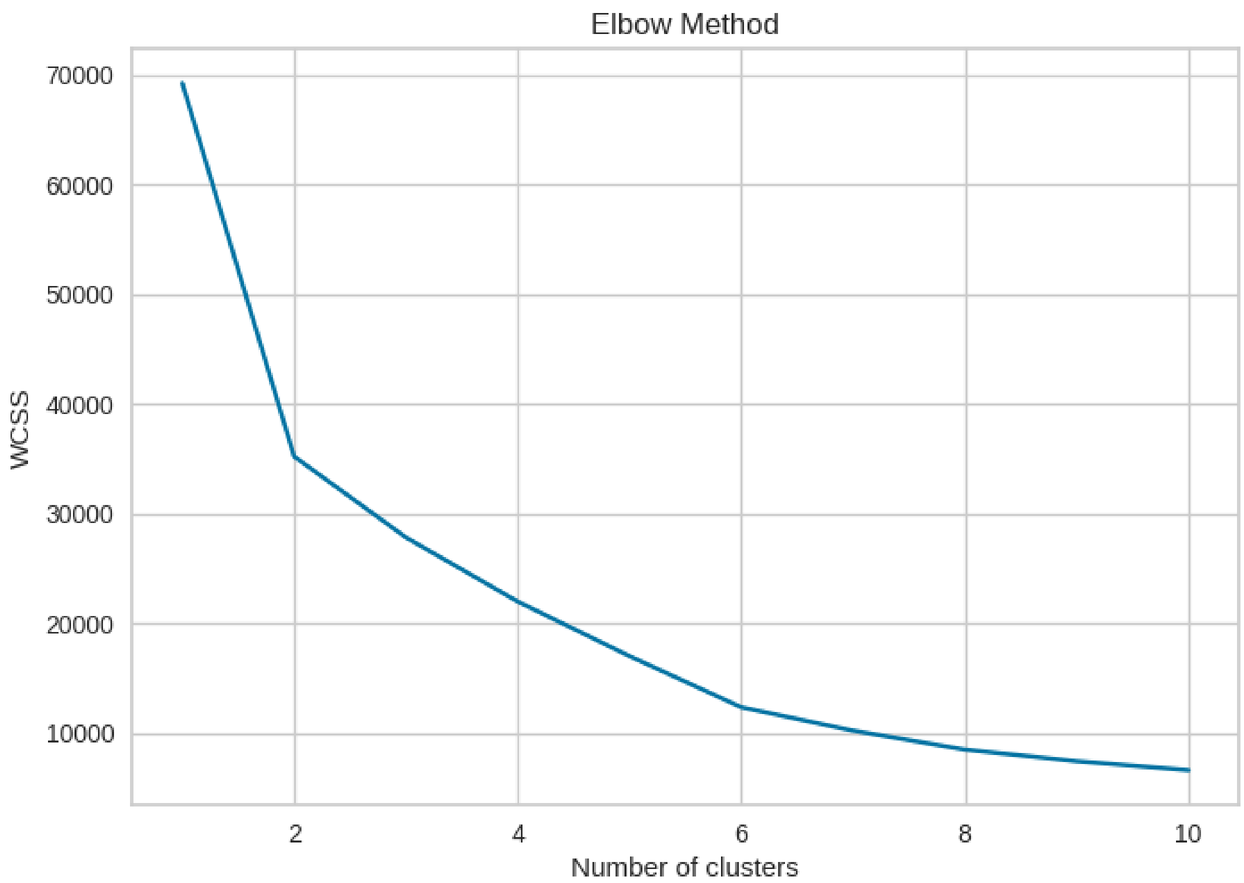


1.6(B) Make Elbow plot (up to n=10) and identify optimum number of clusters for k-means algorithm.

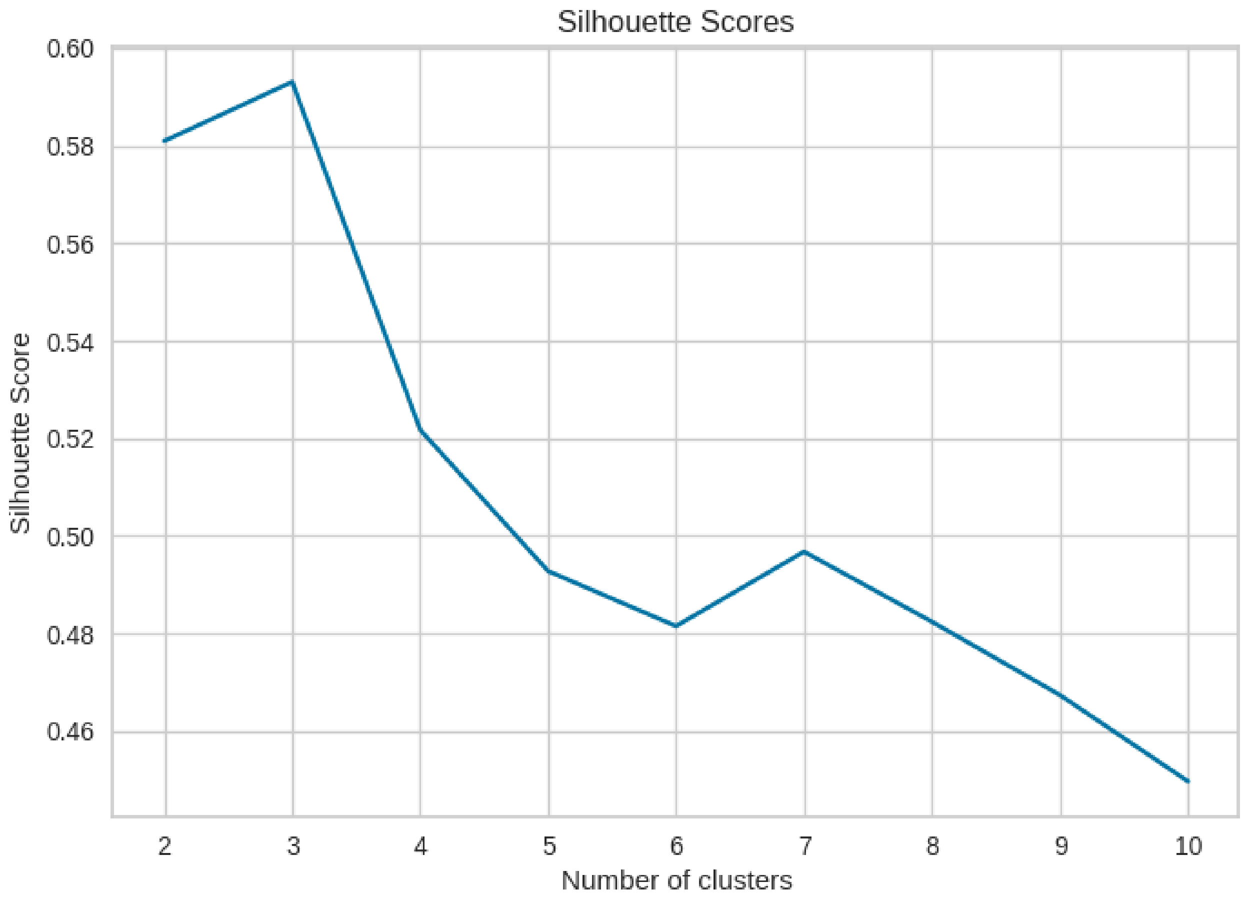
```
# Create an empty list to store the WCSS values
wcss = []

# Perform KMeans for n=1 to n=10
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(data_scaled)
    wcss.append(kmeans.inertia_)

#Plot the Elbow plot
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



1.6 (C)Print silhouette scores for up to 10 clusters and identify optimum number of clusters:



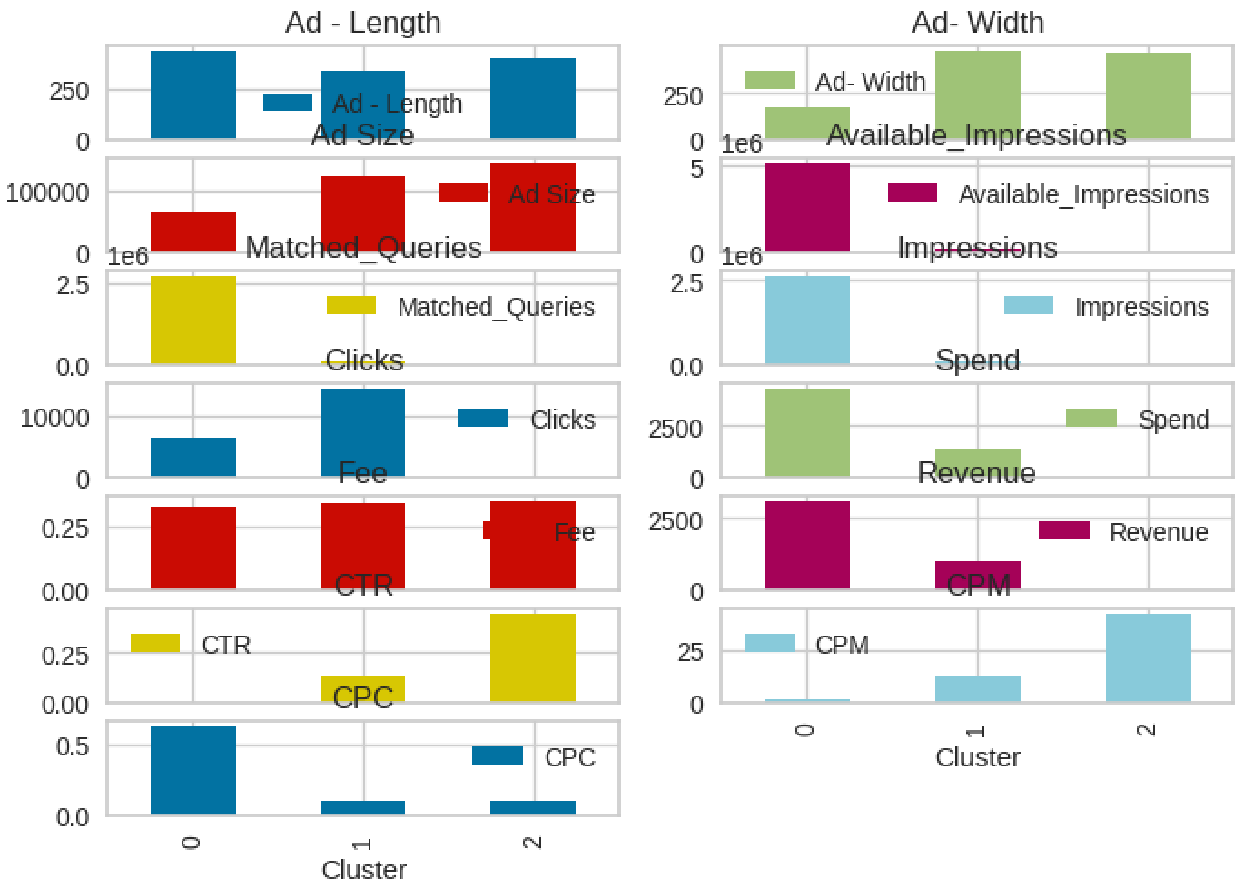

```
for i in range(2,11):
  print('The WSS Value for',i,'Cluster is ',wcss[i-2])
```

```
The WSS Value for 2 Cluster is  69197.9999999998
The WSS Value for 3 Cluster is  35207.357285089485
The WSS Value for 4 Cluster is  27830.27819782109
The WSS Value for 5 Cluster is  21991.60941078634
The WSS Value for 6 Cluster is  17016.98133651621
The WSS Value for 7 Cluster is  12361.363629238045
The WSS Value for 8 Cluster is  10228.814026685915
The WSS Value for 9 Cluster is  8504.369209650378
The WSS Value for 10 Cluster is  7448.087166927873
```

1.7 Profile the ads based on optimum number of clusters using silhouette score and your domain understanding:

```
cluster_data.head()
```

Cluster	Ad - Length	Ad- Width	Ad Size	Available_Impressions	Matched_Queries	Impressions	Clicks	Spend	Fee	Revenue	CTR	CPM	CPC
0	438.737228	171.464027	63197.805891	5.165329e+06	2.726854e+06	2.633884e+06	6383.208402	4295.269384	0.326236	3088.291780	0.003323	1.703330	0.627382
1	340.620690	473.651022	123657.088123	2.083759e+05	1.306020e+05	1.087975e+05	14384.643040	1433.056880	0.342252	990.216139	0.137076	13.050454	0.100662
2	403.016393	461.748634	143737.704918	4.473224e+01	2.875410e+01	2.762295e+01	9.748634	0.967650	0.350000	0.628918	0.449220	42.311000	0.097000



1.8 Conclude the project by providing summary of your learnings:

In this project, we used clustering techniques to segment digital ads data into homogeneous groups based on the features of CPM, CPC and CTR. We first performed basic data analysis, treated missing values, checked for outliers and scaled the data using z-score scaling. We then performed hierarchical clustering and used the elbow plot and silhouette scores to identify the optimum number of clusters. Finally, we profiled the ads based on the optimum number of clusters and identified trends in clicks, spend, revenue, CPM, CTR and CPC based on Device Type.

PART -2

2.0 Problem Statement: India Census Data

PCA

PCA FH (FT): Primary census abstract for female headed households excluding institutional households (India & States/UTs - District Level), Scheduled tribes - 2011 PCA for Female Headed Household Excluding Institutional Household. The Indian Census has the reputation of being one of the best in the world. The first Census in India was conducted in the year 1872. This was conducted at different points of time in different parts of the country. In 1881 a Census was taken for the entire country simultaneously. Since then, Census has been conducted every ten years, without a break. Thus, the Census of India 2011 was the fifteenth in this unbroken series since 1872, the seventh after independence and the second census of the third millennium and twenty first century. The census has been uninterruptedly continued despite of several adversities like wars, epidemics, natural calamities, political unrest, etc. The Census of India is conducted under the provisions of the Census Act 1948 and the Census Rules, 1990. The Primary Census Abstract which is important publication of 2011 Census gives basic information on Area, Total Number of Households, Total Population, Scheduled Castes, Scheduled Tribes Population, Population in the age group 0-6, Literates, Main Workers and Marginal Workers classified by the four broad industrial categories, namely, (i) Cultivators, (ii) Agricultural Laborers, (iii) Household Industry Workers, and (iv) Other Workers and also Non-Workers. The characteristics of the Total Population include Scheduled Castes, Scheduled Tribes, Institutional and Houseless Population and are presented by sex and rural-urban residence. Census 2011 covered 35 States/Union Territories, 640 districts, 5,924 sub-districts, 7,935 Towns and 6,40,867 Villages.

The data collected has so many variables thus making it difficult to find useful details without using Data Science Techniques. You are tasked to perform detailed EDA and identify Optimum Principal Components that explains the most variance in data. Use Sklearn only.

Note: The 24 variables given in the Rubric is just for performing EDA. You will have to consider the entire dataset, including all the variables for performing PCA.
Data file - PCA India Data Census.xlsx

2.1 EDA – Exploratory Data Analysis(PCA)

Read the data and perform basic checks like checking head, info, summary, nulls, and duplicates etc.

- ➢ Checking the shape of the dataset

```
There are 640 rows and 61 columns.
```

- ➢ Displaying few rows of the dataset

	State Code	Dist.Code	State	Area Name	No_HH	TOT_M	TOT_F	M_06	F_06	M_SC	F_SC	M_ST	F_ST	M_LIT	F_LIT	M_ILL	F_ILL	TOT_WORK_M	TOT_WORK_F	MAINWORK_M	MAINWORK_F	MAIN_CL_M	MAIN_CL_F	MAIN
0	1	1	Jammu & Kashmir	Kupwara	7707	23388	29796	5862	6196	3	0	1999	2598	13381	11364	10007	18432	6723	3752	2763	1275	486	235	
1	1	2	Jammu & Kashmir	Badgam	6218	19585	23102	4482	3733	7	6	427	517	10513	7891	9072	15211	6982	4200	4628	1733	1098	357	
2	1	3	Jammu & Kashmir	Leh(Ladakh)	4452	6546	10964	1082	1018	3	6	5806	9723	4534	5840	2012	5124	2775	4800	1940	2923	519	1205	
3	1	4	Jammu & Kashmir	Kargil	1320	2784	4206	563	677	0	0	2666	3968	1842	1962	942	2244	1002	1118	491	408	35	102	
4	1	5	Jammu & Kashmir	Punch	11654	20591	29981	5157	4587	20	33	7670	10843	13243	13477	7348	16504	5717	7692	2523	2267	743	766	

- ➢ Checking the data types of the columns for the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 640 entries, 0 to 639
Data columns (total 61 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   State Code      640 non-null    int64
 1   Dist.Code       640 non-null    int64
 2   State           640 non-null    object
 3   Area Name       640 non-null    object
 4   No_HH           640 non-null    int64
 5   TOT_M           640 non-null    int64
 6   TOT_F           640 non-null    int64
 7   M_06            640 non-null    int64
 8   F_06            640 non-null    int64
 9   M_SC            640 non-null    int64
 10  F_SC            640 non-null    int64
 11  M_ST            640 non-null    int64
 12  F_ST            640 non-null    int64
 13  M_LIT           640 non-null    int64
 14  F_LIT           640 non-null    int64
 15  M_ILL           640 non-null    int64
 16  F_ILL           640 non-null    int64
 17  TOT_WORK_M      640 non-null    int64
 18  TOT_WORK_F      640 non-null    int64
 19  MAINWORK_M      640 non-null    int64
 20  MAINWORK_F      640 non-null    int64
 21  MAIN_CL_M       640 non-null    int64
 22  MAIN_CL_F       640 non-null    int64
```

```
23  MAIN_AL_M        640 non-null    int64
24  MAIN_AL_F        640 non-null    int64
25  MAIN_HH_M        640 non-null    int64
26  MAIN_HH_F        640 non-null    int64
27  MAIN_OT_M        640 non-null    int64
28  MAIN_OT_F        640 non-null    int64
29  MARGWORK_M       640 non-null    int64
30  MARGWORK_F       640 non-null    int64
31  MARG_CL_M        640 non-null    int64
32  MARG_CL_F        640 non-null    int64
33  MARG_AL_M        640 non-null    int64
34  MARG_AL_F        640 non-null    int64
35  MARG_HH_M        640 non-null    int64
36  MARG_HH_F        640 non-null    int64
37  MARG_OT_M        640 non-null    int64
38  MARG_OT_F        640 non-null    int64
39  MARGWORK_3_6_M   640 non-null    int64
40  MARGWORK_3_6_F   640 non-null    int64
41  MARG_CL_3_6_M    640 non-null    int64
42  MARG_CL_3_6_F    640 non-null    int64
43  MARG_AL_3_6_M    640 non-null    int64
44  MARG_AL_3_6_F    640 non-null    int64
45  MARG_HH_3_6_M    640 non-null    int64
46  MARG_HH_3_6_F    640 non-null    int64
47  MARG_OT_3_6_M    640 non-null    int64
48  MARG_OT_3_6_F    640 non-null    int64
49  MARGWORK_0_3_M   640 non-null    int64
50  MARGWORK_0_3_F   640 non-null    int64
```

```
 51  MARG_CL_0_3_M   640 non-null    int64
 52  MARG_CL_0_3_F   640 non-null    int64
 53  MARG_AL_0_3_M   640 non-null    int64
 54  MARG_AL_0_3_F   640 non-null    int64
 55  MARG_HH_0_3_M   640 non-null    int64
 56  MARG_HH_0_3_F   640 non-null    int64
 57  MARG_OT_0_3_M   640 non-null    int64
 58  MARG_OT_0_3_F   640 non-null    int64
 59  NON_WORK_M      640 non-null    int64
 60  NON_WORK_F      640 non-null    int64
dtypes: int64(59), object(2)
```

➢ Statistical summary of the dataset

There are 2 object variables, 59 integer type

1 to 25 of 59 entries Filter

index	count	mean	std	min	25%	50%	75%	max
State Code	640.0	17.1140625	9.426486295388743	1.0	9.0	18.0	24.0	35.0
Dist.Code	640.0	320.5	184.89636737012077	1.0	160.75	320.5	480.25	640.0
No_HH	640.0	51222.871875	48135.40547477947	350.0	19484.0	35837.0	68892.0	310450.0
TOT_M	640.0	79940.5765625	73384.51111369701	391.0	30228.0	58339.0	107918.5	485417.0
TOT_F	640.0	122372.084375	113600.7172815185	698.0	46517.75	87724.5	164251.75	750392.0
M_06	640.0	12309.0984375	11500.906880809953	56.0	4733.75	9159.0	16520.25	96223.0
F_06	640.0	11942.3	11326.294566668952	56.0	4672.25	8663.0	15902.25	95129.0
M_SC	640.0	13820.946875	14426.3731297206	0.0	3466.25	9591.5	19429.75	103307.0
F_SC	640.0	20778.3921875	21727.887713109423	0.0	5603.25	13709.0	29180.0	156429.0
M_ST	640.0	6191.8078125	9912.668947884893	0.0	293.75	2333.5	7658.0	96785.0
F_ST	640.0	10155.640625	15875.701488196419	0.0	429.5	3834.5	12480.25	130119.0
M_LIT	640.0	57967.9796875	55910.28246589229	286.0	21298.0	42693.5	77989.5	403261.0
F_LIT	640.0	66359.565625	75037.86020746626	371.0	20932.0	43796.5	84799.75	571140.0
M_ILL	640.0	21972.596875	19825.605267802846	105.0	8590.0	15767.5	29512.5	105961.0
F_ILL	640.0	56012.51875	47116.69376939182	327.0	22367.0	42386.0	78471.0	254160.0
TOT_WORK_M	640.0	37992.4078125	36419.537491220704	100.0	13753.5	27936.5	50226.75	269422.0
TOT_WORK_F	640.0	41295.7609375	37192.36094345711	357.0	16097.75	30588.5	53234.25	257848.0
MAINWORK_M	640.0	30204.446875	31480.91567993618	65.0	9787.0	21250.5	40119.0	247911.0
MAINWORK_F	640.0	28198.846875	29998.26268858454	240.0	9502.25	18484.0	35063.25	226166.0
MAIN_CL_M	640.0	5424.3421875	4739.161969403382	0.0	2023.5	4160.5	7695.0	29113.0
MAIN_CL_F	640.0	5486.0421875	5326.362727592411	0.0	1920.25	3908.5	7286.25	36193.0
MAIN_AL_M	640.0	5849.109375	6399.507966261204	0.0	1070.25	3936.5	8067.25	40843.0
MAIN_AL_F	640.0	8925.9953125	12864.287584362615	0.0	1408.75	3933.5	10617.5	87945.0
MAIN_HH_M	640.0	883.89375	1278.642344576578	0.0	187.5	498.5	1099.25	16429.0
MAIN_HH_F	640.0	1380.7734375	3179.4144489361515	0.0	248.75	540.5	1435.75	45979.0

Show 25 per page 1 2 3

26 to 50 of 59 entries Filter

index	count	mean	std	min	25%	50%	75%	max
MAIN_OT_M	640.0	18047.1015625	26068.48088563092	36.0	3997.5	9598.0	21249.5	240855.0
MAIN_OT_F	640.0	12406.0359375	18972.2023686502	153.0	3142.5	6380.5	14368.25	209355.0
MARGWORK_M	640.0	7787.9609375	7410.791690533128 5	35.0	2937.5	5627.0	9800.25	47553.0
MARGWORK_F	640.0	13096.9140625	10996.47452796788	117.0	5424.5	10175.0	18879.25	66915.0
MARG_CL_M	640.0	1040.7375	1311.546847376734	0.0	311.75	606.5	1281.0	13201.0
MARG_CL_F	640.0	2307.6828125	3564.626095357566	0.0	630.25	1226.0	2659.25	44324.0
MARG_AL_M	640.0	3304.3265625	3781.5557071285475	0.0	873.5	2062.0	4300.75	23719.0
MARG_AL_F	640.0	6463.28125	6773.876297663584	0.0	1402.5	4020.5	9089.25	45301.0
MARG_HH_M	640.0	316.7421875	462.66189140717603	0.0	71.75	166.0	356.5	4298.0
MARG_HH_F	640.0	786.6265625	1198.7182132798703	0.0	171.75	429.0	962.5	15448.0
MARG_OT_M	640.0	3126.1546875	3609.3918205235764	7.0	935.5	2036.0	3985.25	24728.0
MARG_OT_F	640.0	3539.3234375	4115.191314204173	19.0	1071.75	2349.5	4400.5	36377.0
MARGWORK_3_6_M	640.0	41948.16875	39045.316917993696	291.0	16208.25	30315.0	57218.75	300937.0
MARGWORK_3_6_F	640.0	81076.3234375	82970.40621572598	341.0	26619.5	56793.0	107924.0	676450.0
MARG_CL_3_6_M	640.0	6394.9875	6019.806643999657	27.0	2372.0	4630.0	8167.0	39106.0
MARG_CL_3_6_F	640.0	10339.8640625	8467.473429380616	85.0	4351.5	8295.0	15102.0	50065.0
MARG_AL_3_6_M	640.0	789.8484375	905.6392793929609	0.0	235.5	480.5	986.0	7426.0
MARG_AL_3_6_F	640.0	1749.584375	2496.541513703694	0.0	497.25	985.5	2059.0	27171.0
MARG_HH_3_6_M	640.0	2743.6359375	3059.5863867296516	0.0	718.75	1714.5	3702.25	19343.0
MARG_HH_3_6_F	640.0	5169.85	5335.64096021631	0.0	1113.75	3294.0	7502.25	36253.0
MARG_OT_3_6_M	640.0	245.3625	358.7285666169371	0.0	58.0	129.5	276.0	3535.0
MARG_OT_3_6_F	640.0	585.884375	900.0258172646815	0.0	127.75	320.5	719.25	12094.0
MARGWORK_0_3_M	640.0	2616.140625	3036.9643812728705	7.0	755.0	1681.5	3320.25	20648.0
MARGWORK_0_3_F	640.0	2834.5453125	3327.8369319771195	14.0	833.5	1834.5	3610.5	25844.0
MARG_CL_0_3_M	640.0	1392.9734375	1489.7070515540374	4.0	489.5	949.0	1714.0	9875.0

Show 25 per page 1 2 3

51 to 59 of 59 entries Filter

index	count	mean	std	min	25%	50%	75%	max
MARG_CL_0_3_F	640.0	2757.05	2788.7766756194146	30.0	957.25	1928.0	3599.75	21611.0
MARG_AL_0_3_M	640.0	250.8890625	453.3365940290036	0.0	47.0	114.5	270.75	5775.0
MARG_AL_0_3_F	640.0	558.0984375	1117.64274823112	0.0	109.0	247.5	568.75	17153.0
MARG_HH_0_3_M	640.0	560.690625	762.5789912585069	0.0	136.5	308.0	642.0	6116.0
MARG_HH_0_3_F	640.0	1293.43125	1585.377936100369	0.0	298.0	717.0	1710.75	13714.0
MARG_OT_0_3_M	640.0	71.3796875	107.89762678259842	0.0	14.0	35.0	79.0	895.0
MARG_OT_0_3_F	640.0	200.7421875	309.74085402556176	0.0	43.0	113.0	240.0	3354.0
NON_WORK_M	640.0	510.0140625	610.6031868227664	0.0	161.0	326.0	604.5	6456.0
NON_WORK_F	640.0	704.778125	910.2092250266159	5.0	220.5	464.5	853.5	10533.0

Show 25 per page 1 2 3

➢ Checking the Duplicate Values

No Duplicate Values found

```
[11] df_pca.duplicated().sum()

0
```

There are no duplicate values

- ➢ Checking for Missing Values

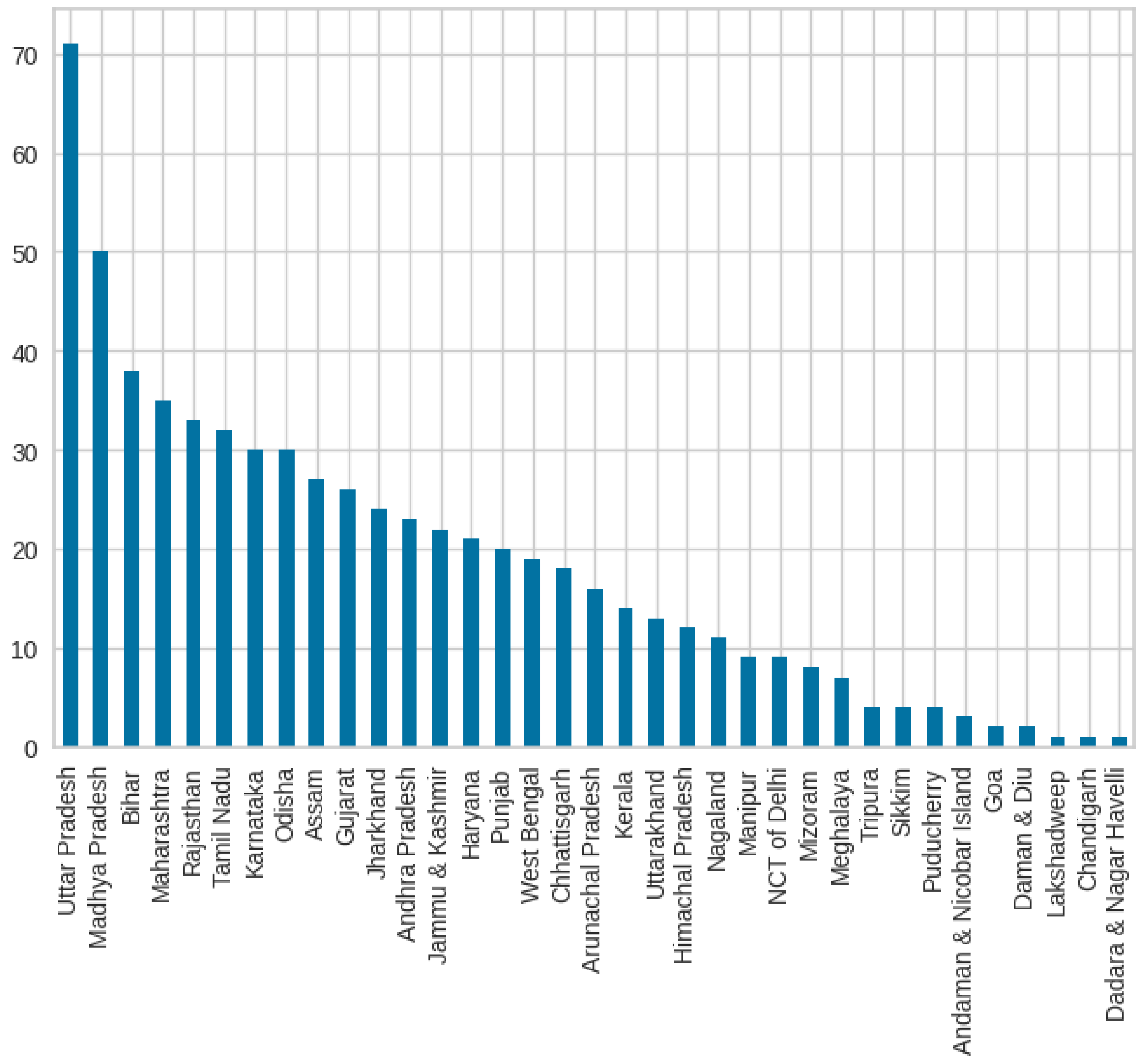
There are no missing values

```
# checking for missing
df_pca.isnull().sum()
```

```
State Code      0
Dist.Code       0
State           0
Area Name       0
No_HH           0
TOT_M           0
TOT_F           0
M_06            0
F_06            0
M_SC            0
F_SC            0
M_ST            0
F_ST            0
M_LIT           0
F_LIT           0
M_ILL           0
F_ILL           0
TOT_WORK_M      0
TOT_WORK_F      0
MAINWORK_M      0
MAINWORK_F      0
MAIN_CL_M       0
MAIN_CL_F       0
MAIN_AL_M       0
MAIN_AL_F       0
MAIN_HH_M       0
MAIN_HH_F       0
```

MAIN_OT_M	0
MAIN_OT_F	0
MARGWORK_M	0
MARGWORK_F	0
MARG_CL_M	0
MARG_CL_F	0
MARG_AL_M	0
MARG_AL_F	0
MARG_HH_M	0
MARG_HH_F	0
MARG_OT_M	0
MARG_OT_F	0
MARGWORK_3_6_M	0
MARGWORK_3_6_F	0
MARG_CL_3_6_M	0
MARG_CL_3_6_F	0
MARG_AL_3_6_M	0
MARG_AL_3_6_F	0
MARG_HH_3_6_M	0
MARG_HH_3_6_F	0
MARG_OT_3_6_M	0
MARG_OT_3_6_F	0
MARGWORK_0_3_M	0
MARGWORK_0_3_F	0
MARG_CL_0_3_M	0
MARG_CL_0_3_F	0
MARG_AL_0_3_M	0
MARG_AL_0_3_F	0
MARG_HH_0_3_M	0

➢ Perform detailed Exploratory analysis



2.2 Which state has the highest gender ratio, and which has the lowest?

```python
# (i) Which state has the highest gender ratio, and which has the lowest?
gender_ratio = data.groupby('State').agg({'TOT_M': 'sum', 'TOT_F': 'sum'})
gender_ratio['gender_ratio'] = gender_ratio['TOT_M'] / gender_ratio['TOT_F']
gender_ratio.sort_values(by='gender_ratio', ascending=False)
```

1 to 35 of 35 entries Filter

State	TOT_M	TOT_F	gender_ratio
Lakshadweep	12823	14772	0.8680611968589222
Haryana	1167816	1498873	0.7791293858785902
NCT of Delhi	833414	1075266	0.7750770507018728
Uttar Pradesh	9043969	12023885	0.7521669576846418
Meghalaya	268036	356355	0.7521600651036185
Bihar	4025198	5405883	0.7445958412344478
Punjab	1579405	2121425	0.7445019267709205
Jammu & Kashmir	421213	572959	0.7351538242701485
Daman & Diu	13153	18706	0.7031433764567518
Chandigarh	41753	59644	0.7000368855207565
Rajasthan	2062563	2966496	0.695285953529012
Assam	1437268	2093432	0.6865606334478502
Jharkhand	1202623	1763884	0.6818039054722419
Gujarat	1983685	2939472	0.6748439855865271
Andaman & Nicobar Island	18726	28691	0.6526785403088077
West Bengal	3912553	6016118	0.6503451228848902
Dadara & Nagar Havelli	6982	10831	0.6446311513249008
Himachal Pradesh	483381	752062	0.6427408910435576
Sikkim	26664	41518	0.6422274676044125
Manipur	145524	226963	0.6411793992853461
Madhya Pradesh	2155608	3369745	0.6396946949991765
Karnataka	3409482	5345675	0.6378019613987008
Uttarakhand	613924	973147	0.6308646072998221
Tripura	160457	256370	0.62588056324843
Mizoram	59534	95463	0.6236342876297624
Goa	118979	191393	0.6216476046668373
Kerala	2919825	4856357	0.6012377179025348
Puducherry	70386	119074	0.5911114097116079
Maharashtra	4196130	7138557	0.5878120746251658
Nagaland	73506	125935	0.5836820582046294
Odisha	1460031	2536980	0.5754996097722489
Arunachal Pradesh	50582	88066	0.5743646810346785
Chhattisgarh	838404	1526592	0.5491997861904163
Tamil Nadu	3074009	5610310	0.5479214161071313
Andhra Pradesh	3274363	6097235	0.5370242413159407

```
The State  with the highest gender ratio is Lakshadweep.
The State  with the lowest gender ratio is Andhra Pradesh.
```

2.3 Which District has the highest gender ratio, and which has the lowest?

(ii) Which District has the highest and lowest gender ratio?

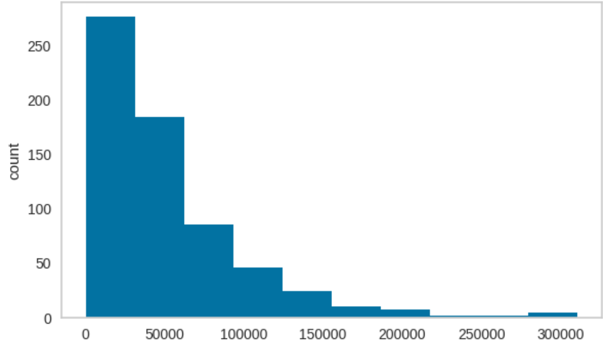
```
3] # (ii) Which District has the highest and lowest gender ratio?
   data['gender_ratio'] = data['TOT_M'] / data['TOT_F']
   highest_gr_district = data.sort_values('gender_ratio', ascending=False)['Area Name'].iloc[0]
   lowest_gr_district = data.sort_values('gender_ratio')['Area Name'].iloc[0]
   print(f"The Distict  with the highest gender ratio is {highest_gr_district}.")
   print(f"The District  with the lowest gender ratio is {lowest_gr_district}.")
```

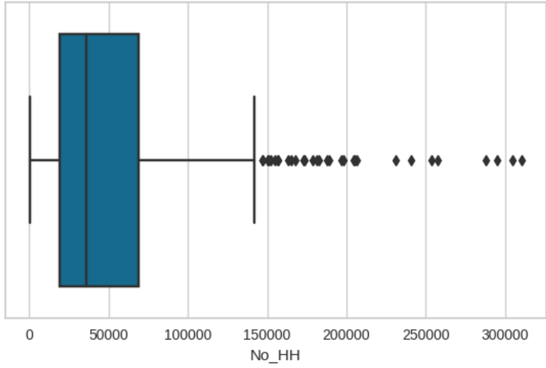
```
The Distict  with the highest gender ratio is Lakshadweep.
The District  with the lowest gender ratio is Krishna.
```

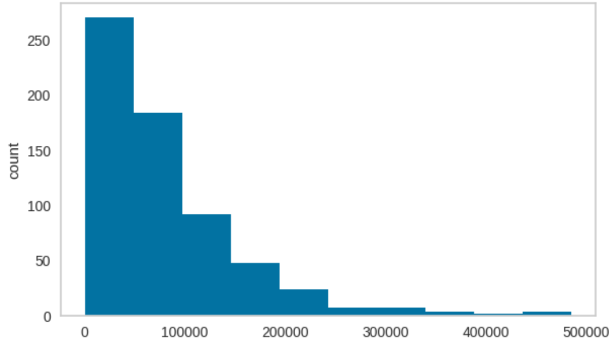


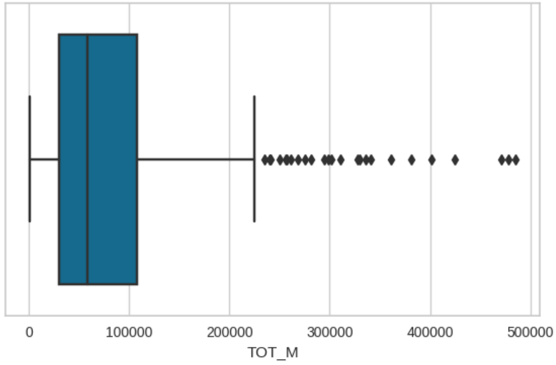
```
The Distict  with the highest gender ratio is Lakshadweep.
The District  with the lowest gender ratio is Krishna.
```

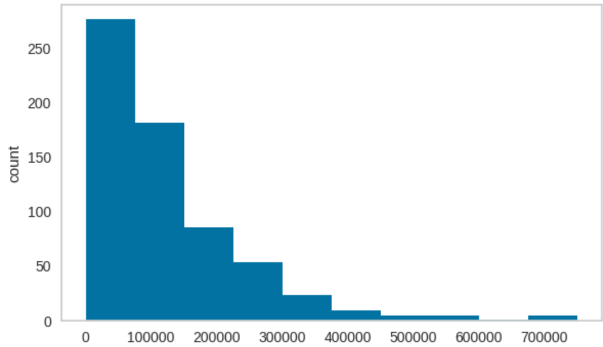
2.4 PCA – Check Outliers

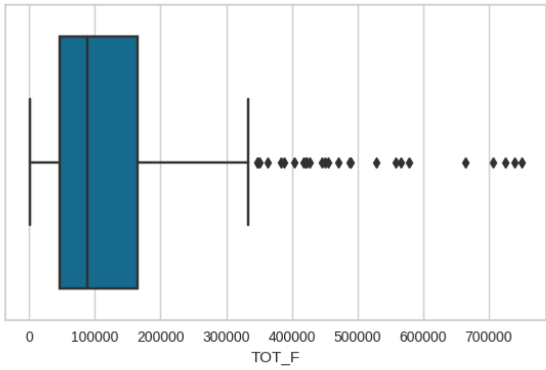


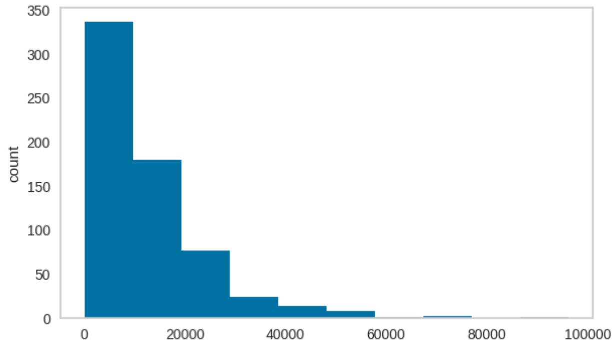


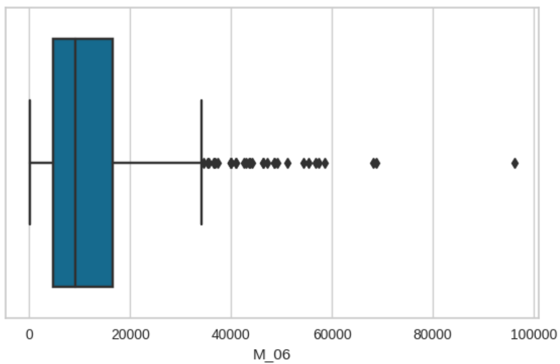


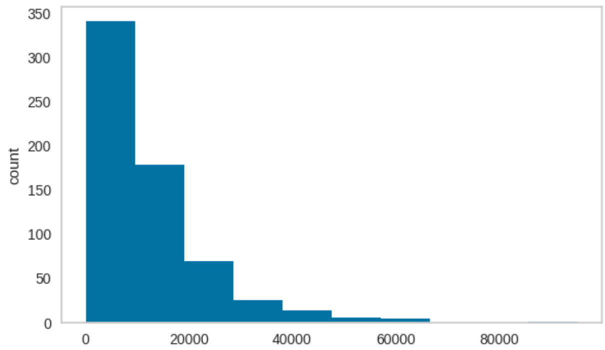


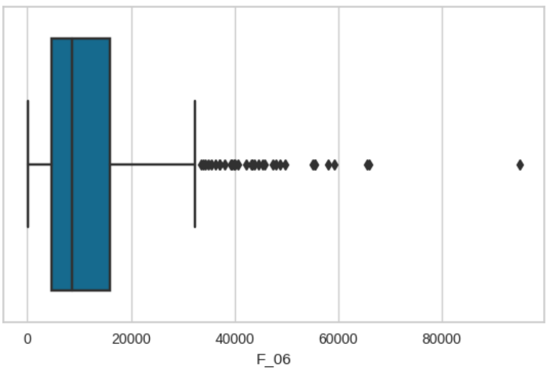












It is important to treat outliers as they can significantly affect the results of PCA. However, for this case, we have chosen not to treat outliers.

2.5 Scale the Data using Z-Scale Method

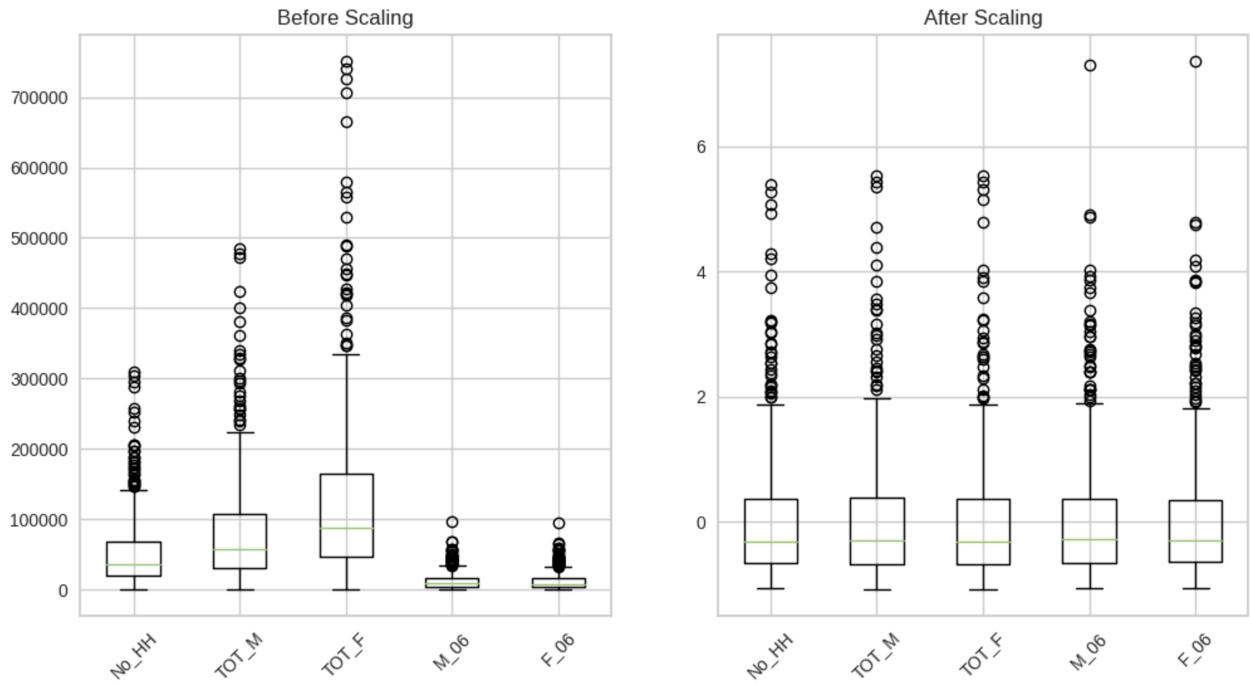
Scaling the Data

```
[102] selected_variables = ['No_HH', 'TOT_M', 'TOT_F', 'M_06', 'F_06']
      eda_data = data[selected_variables]
```

```
[103] scaler = StandardScaler()
      scaled_data = scaler.fit_transform(eda_data)
```

	No_HH	TOT_M	TOT_F	M_06	F_06
0	-0.904738	-0.771236	-0.815563	-0.561012	-0.507738
1	-0.935695	-0.823100	-0.874534	-0.681096	-0.725367
2	-0.972412	-1.000919	-0.981466	-0.976956	-0.965262
3	-1.037530	-1.052224	-1.041001	-1.022118	-0.995393
4	-0.822676	-0.809381	-0.813933	-0.622359	-0.649908

Scaling can affect the outliers by changing their values, but it does not remove them. We can compare the boxplots before and after scaling to see the impact on outliers.



Before Scaling
700000
600000
500000
400000
300000
200000
100000
0
No_HH
TOT_M
TOT_F
M_06
F_06
After Scaling
6
4
2
0
No_HH
TOT_M
TOT_F
M_06
F_06

2.6 Perform PCA

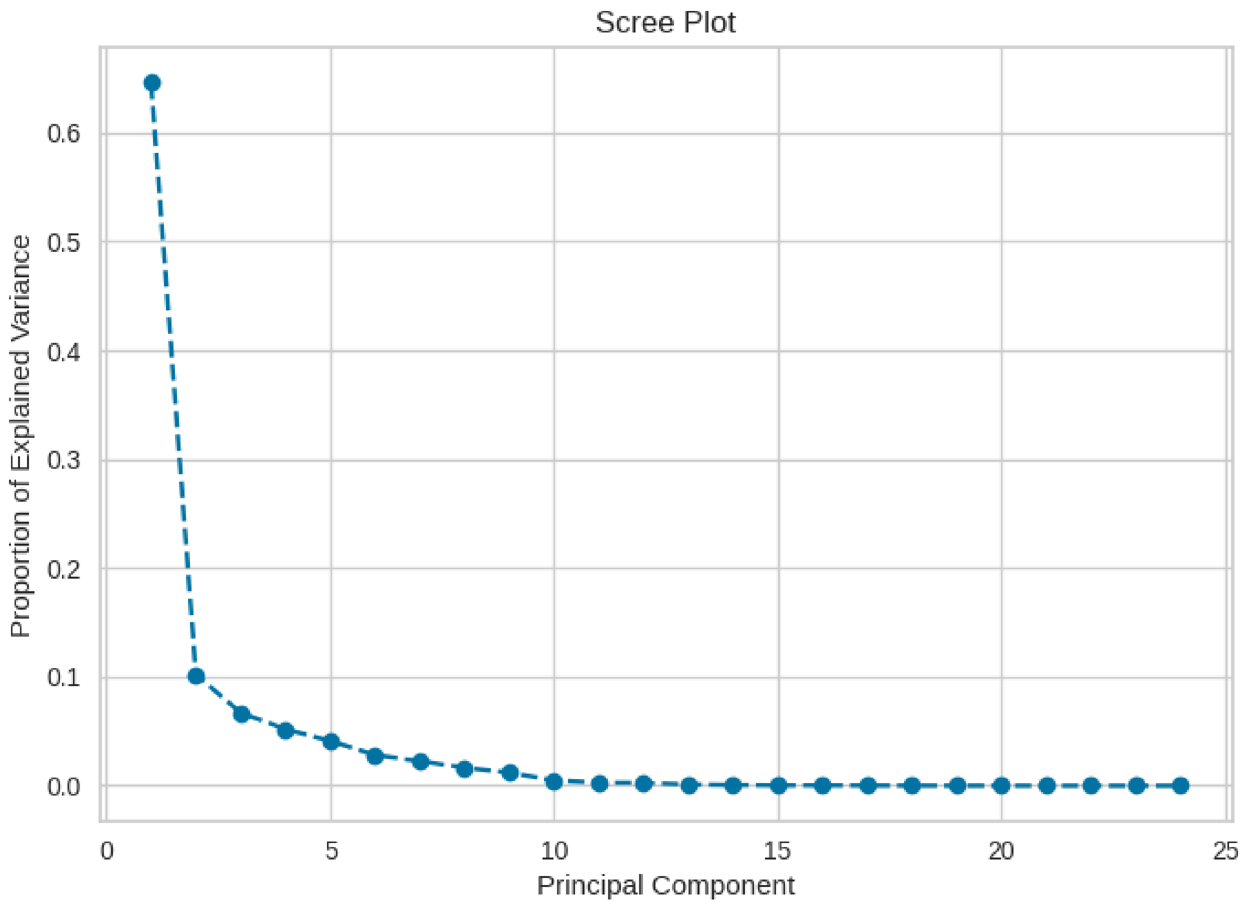
```
pca = PCA()
pca.fit(scaled_data)
covariance_matrix = pca.get_covariance()
explained_variance = pca.explained_variance_ratio_

# Get eigenvalues and eigenvectors
eigen_values, eigen_vectors = np.linalg.eig(covariance_matrix)

# Get the principal components
pca_components = pca.transform(scaled_data)
```

```
▾ PCA
PCA(n_components=24)
```

To identify the optimum number of principal components (PCs), we can use the scree plot. The scree plot shows the eigenvalues of each PC in descending order, with the corresponding proportion of explained variance. We can plot the eigenvalues against the PCs and observe where the eigenvalues start to level off.



Compare PCs with Actual Columns and identify which is explaining most variance

	PC1	PC2	PC3	PC4	PC5
TOT_F	0.976837	-0.042508	0.126860	0.030172	-0.145558
TOT_M	0.971085	-0.132210	0.135699	-0.087106	-0.100343
TOT_WORK_M	0.962974	-0.089059	0.172254	0.057946	-0.054318
M_LIT	0.956773	-0.134921	0.190662	-0.005288	-0.136287
No_HH	0.948326	0.067300	0.084132	0.198024	-0.158814
F_ILL	0.940527	0.087902	-0.161351	-0.098190	0.031781
MAINWORK_M	0.930236	-0.075424	0.187415	0.149184	-0.069751
TOT_WORK_F	0.898294	0.300577	-0.045322	0.226760	-0.060523