Customer Analytics in Industry (Part 1) by Sooyeon Won Keywords Marketing Mix STP framework Customer Analytics Segmentation and Clustering Dimensionality Reduction with PCA Data Visualisations Contents

1. Introduction
2. Data Preparation
3. Data Exploration
4. Data Analysis 4.1. Customer Analytics 4.2. Purchase Analytics i. Descriptive Analyses by Segment i-1. The Proportion of each Segment i-2. Purchase Occasions and Purchase Incidences i-3. Brand Chocie i-4. Revenue Comparison between segments ii. Predictive Analyses ii-1. Modeling Purchase Incidence ii-2. Modeling Brand Choice ii-3. Modeling Purchase Quantity
5. Conclusion

1. Introduction For this project, I conducted the customer Analysis in the FMSC industry, based on STP Framework, and marketing modelings. This project is consisted of two parts. The first part of analysis is mainly focusing on "Customer Analytics". In this part, I conducted the customer segmentation, applying various clustering algorithms and also by reducing the dimensionality of the problem. Then in the second part of analysis. I mainly investigate about "Purchase Analytics". I explorted the descriptive and predictive analysis of the purchase behaviour of customers, including models for purchase incidence, brand choice, purchase quantity to make predictions using real-world data. This project is motivated by Customer Analytics program in Udemy.

2. Data Preparation

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples
import pickle
import uuid
```

```
# Import the relevant libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
plt.rc("font", size=14)
import seaborn as sns
sns.set()
sns.set(style="whitegrid", color_codes=True)
```

```
# Set visualization style
plt.rc("font", size=14)
sns.set()
sns.set(style="whitegrid", color_codes=True)
```

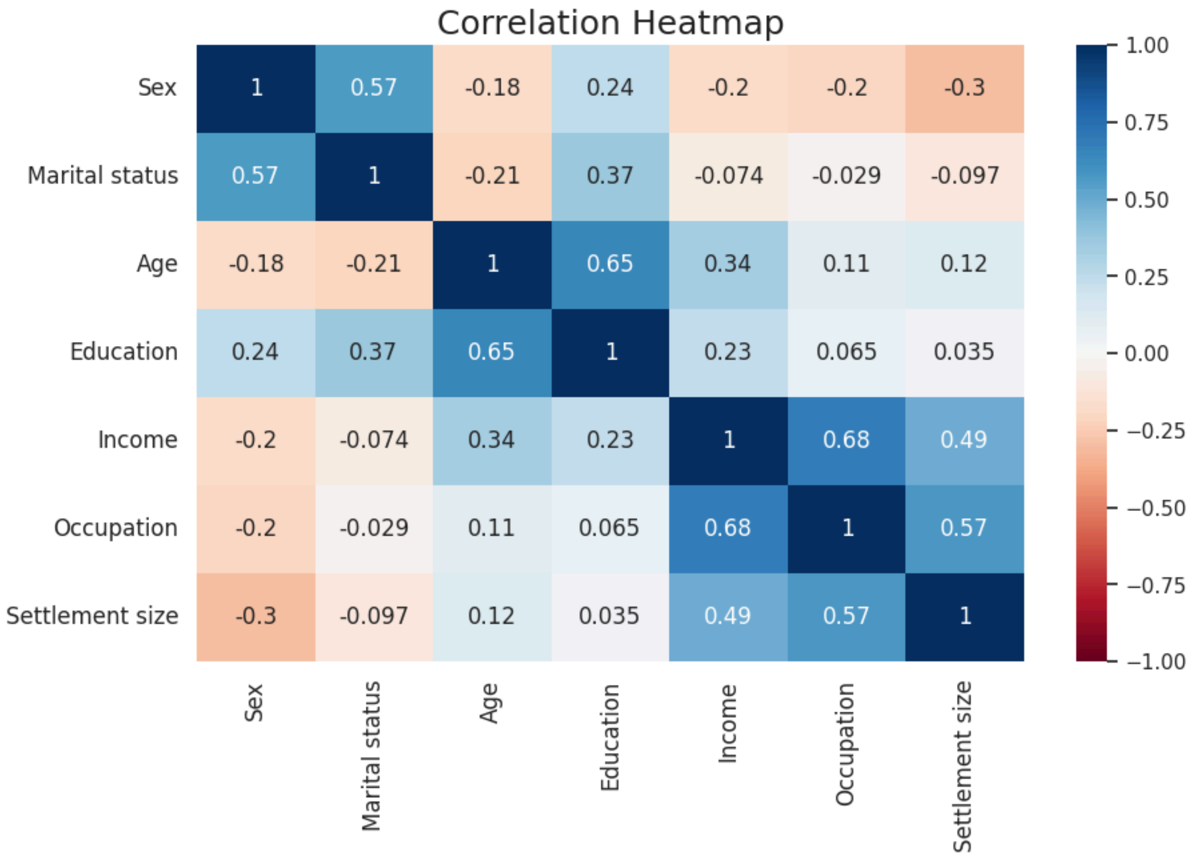
```
# Import data
demo_df = pd.read_csv('segmentation data.csv', index_col = 0)
```

```
# Basic data exploration
print("First 5 rows of the dataset:")
print(demo_df.head())
print("\nDataset Info:")
print(demo_df.info())
print("\nNumber of duplicated rows:", sum(demo_df.duplicated()))
print("\nSummary Statistics:")
print(demo_df.describe())
```

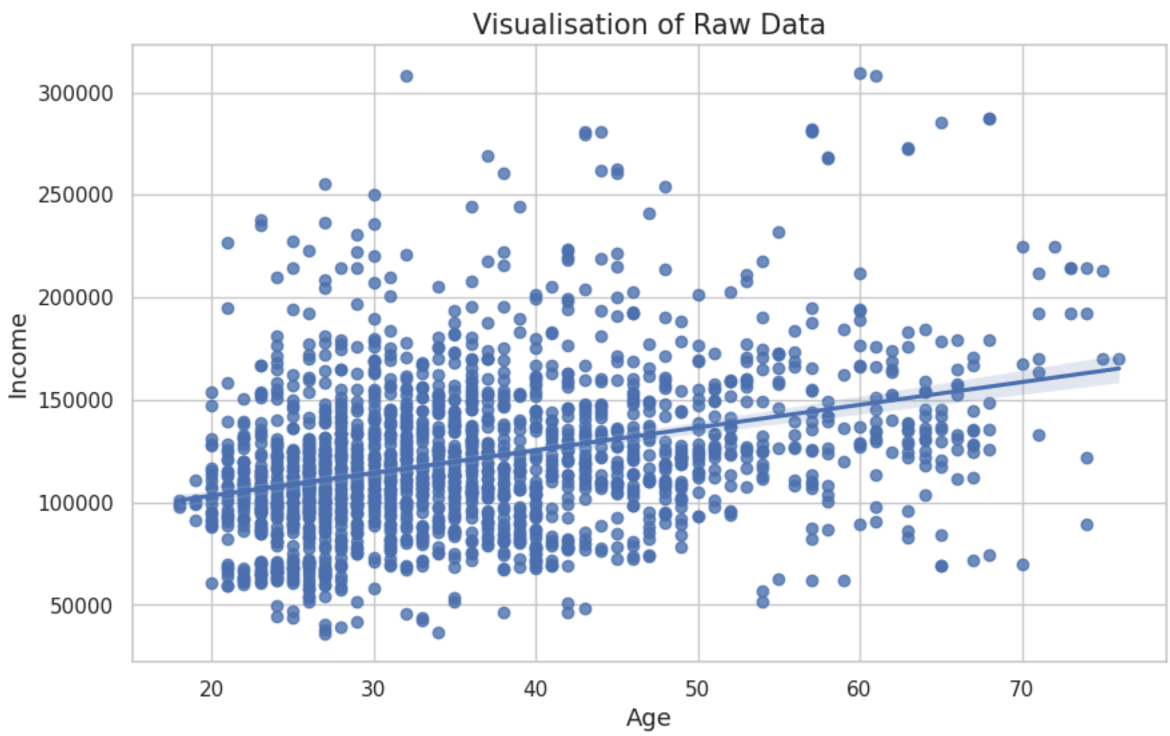
```
First 5 rows of the dataset:
           Sex  Marital status  Age  Education  Income  Occupation  \
ID
100000001    0               0   67          2  124670           1
100000002    1               1   22          1  150773           1
100000003    0               0   49          1   89210           0
100000004    0               0   45          1  171565           1
100000005    0               0   53          1  149031           1

           Settlement size
ID
100000001                2
100000002                2
100000003                0
```

```
100000004                1
100000005                1

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 2000 entries, 100000001 to 100002000
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Sex              2000 non-null   int64
 1   Marital status   2000 non-null   int64
 2   Age              2000 non-null   int64
 3   Education        2000 non-null   int64
 4   Income           2000 non-null   int64
 5   Occupation       2000 non-null   int64
 6   Settlement size  2000 non-null   int64
dtypes: int64(7)
memory usage: 125.0 KB
None

Number of duplicated rows: 0

Summary Statistics:
               Sex  Marital status          Age   Education         Income  \
count  2000.000000     2000.000000  2000.000000  2000.00000    2000.000000
mean      0.457000        0.496500    35.909000     1.03800  120954.419000
std       0.498272        0.500113    11.719402     0.59978   38108.824679
min       0.000000        0.000000    18.000000     0.00000   35832.000000
25%       0.000000        0.000000    27.000000     1.00000   97663.250000
50%       0.000000        0.000000    33.000000     1.00000  115548.500000
75%       1.000000        1.000000    42.000000     1.00000  138072.250000
max       1.000000        1.000000    76.000000     3.00000  309364.000000

        Occupation  Settlement size
count  2000.000000      2000.000000
mean      0.810500         0.739000
std       0.638587         0.812533
min       0.000000         0.000000
25%       0.000000         0.000000
50%       1.000000         1.000000
75%       1.000000         1.000000
max       2.000000         2.000000
```

```
# Correlation in Heatmap form

plt.figure(figsize=(10,6))
s = sns.heatmap(demo_df.corr(), annot = True, cmap = 'RdBu', vmin = -1, vmax=1)
# different color sets = 'viridis', 'autumn', 'rainbow'
s.set_yticklabels(s.get_yticklabels(), rotation = 0, fontsize =12)
s.set_xticklabels(s.get_xticklabels(), rotation = 90, fontsize =12)
plt.title('Correlation Heatmap', fontsize =18)
plt.show()
```

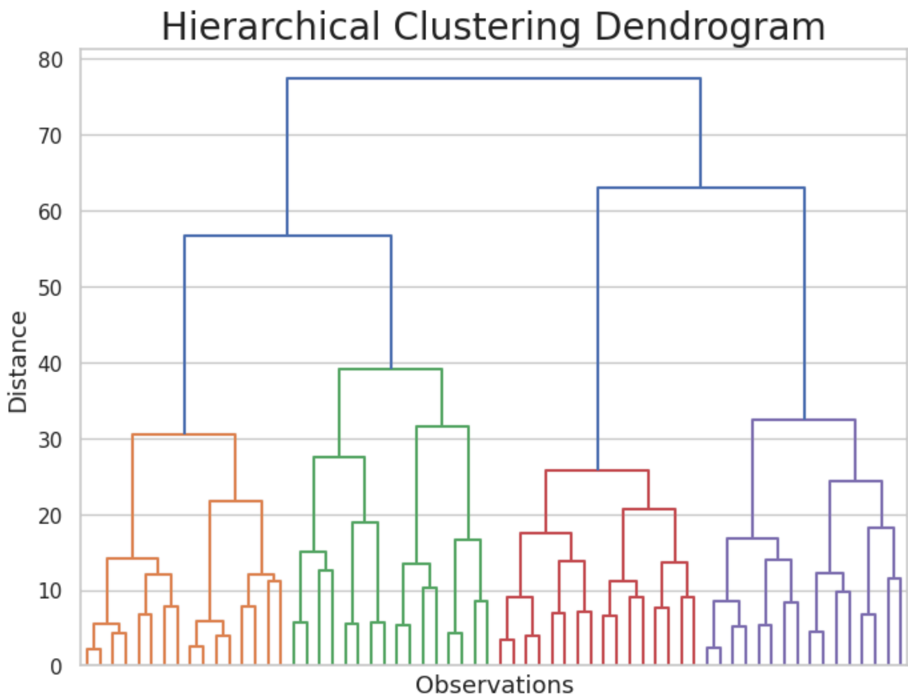



```
plt.figure(figsize=(10, 6))
sns.regplot(x=demo_df.iloc[:, 2], y=demo_df.iloc[:, 4])
plt.xlabel('Age', fontsize=13)
plt.ylabel('Income', fontsize=13)
plt.title('Visualisation of Raw Data', fontsize=15)
plt.show()
```



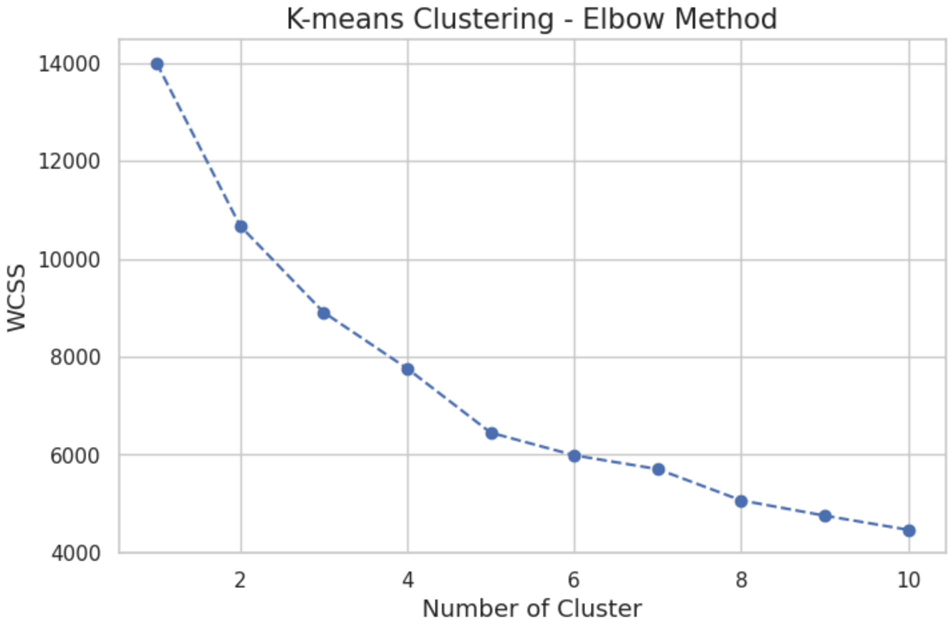

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()
demo_scaled = scaler.fit_transform(demo_df)


from scipy.cluster.hierarchy import dendrogram, linkage
```

```
hier_clust = linkage(demo_scaled, method = 'ward')


plt.figure(figsize =(8,6))
plt.title('Hierarchical Clustering Dendrogram', fontsize =20)
plt.ylabel('Distance',  fontsize =13)
plt.xlabel('Observations',  fontsize =13)
dendrogram(hier_clust, show_leaf_counts =False,
           truncate_mode = 'level',
           p = 5,
           no_labels = True,
           # color_threshold = 0 : the clusters differrent color
          )

plt.show()
```




```
# Import the relevant library
from sklearn.cluster import KMeans


wcss = {}
for i in range(1, 11):
    kmeans = KMeans(n_clusters =i, init = 'k-means++', random_state= 42)
    kmeans.fit(demo_scaled)
    wcss[i] = kmeans.inertia_


# Elbow method
plt.figure(figsize =(8,5))
plt.plot(list(wcss.keys()), list(wcss.values()),  marker = 'o', linestyle = '--' )

plt.xlabel('Number of Cluster', fontsize = 13)
plt.ylabel('WCSS', fontsize = 13)
plt.title('K-means Clustering - Elbow Method', fontsize = 15)
```

```
Text(0.5, 1.0, 'K-means Clustering - Elbow Method')
```




```
!pip install kneed
```

```
Collecting kneed
  Downloading kneed-0.8.5-py3-none-any.whl.metadata (5.5 kB)
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.11/dist-packages (from kneed) (2.0.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from kneed) (1.14.1)
Downloading kneed-0.8.5-py3-none-any.whl (10 kB)
Installing collected packages: kneed
Successfully installed kneed-0.8.5
```

```
from kneed import KneeLocator
x, y = list(wcss.keys()), list(wcss.values())
kn = KneeLocator(x, y, curve='convex', direction='decreasing')
print('The optimal number of clusters, suggested by Elbow criterion: ', kn.knee)
```

```
The optimal number of clusters, suggested by Elbow criterion:  5
```

```
kmeans =KMeans(n_clusters =4, max_iter = 500,  init = 'k-means++', random_state= 42)
```

```
kmeans.fit(demo_scaled)
```

```
KMeans
KMeans(max_iter=500, n_clusters=4, random_state=42)
```

```
df_segm_kmeans = demo_df.copy()
```

```
df_segm_kmeans['Segment_KMeans'] = kmeans.labels_
df_segm_kmeans['Segment_KMeans'].replace({0: "A", 1: "B", 2: "C", 3:"D"}, inplace=True)
```

```
<ipython-input-22-42ec0b06ca82>:2: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c

  df_segm_kmeans['Segment_KMeans'].replace({0: "A", 1: "B", 2: "C", 3:"D"}, inplace=True)
```

```
df_segm_analysis = df_segm_kmeans.groupby(['Segment_KMeans']).mean()
df_segm_analysis
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
Segment_KMeans							
A	0.066543	0.000000	33.240296	0.489834	109932.785582	0.639556	0.611830
B	0.868254	0.785714	32.928571	1.163492	98466.955556	0.384127	0.006349
C	0.691099	0.979058	29.060209	1.104712	126838.926702	1.107330	1.324607
D	0.149888	0.277405	49.192394	1.467562	160958.722595	1.364653	1.425056

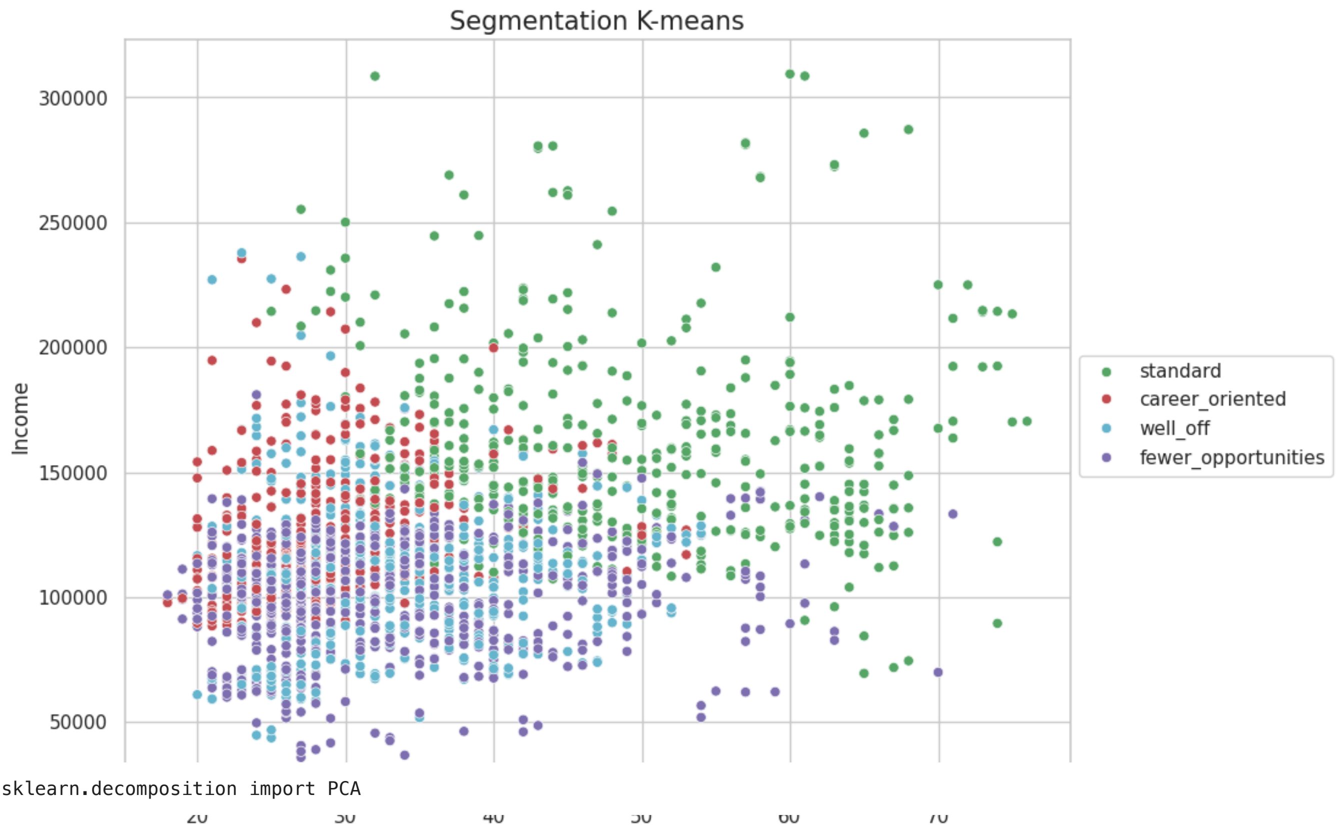
```
# Compute the size and proportions of the four clusters
df_segm_analysis['N_Obs'] = df_segm_kmeans[['Segment_KMeans', 'Sex']].groupby(['Segment_KMeans'])['Sex'].count()
df_segm_analysis['Prop_Obs'] = df_segm_analysis.N_Obs / df_segm_analysis.N_Obs.sum()
df_segm_analysis
```

Segment_KMeans	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	N_Obs	Prop_Obs
A	0.066543	0.000000	33.240296	0.489834	109932.785582	0.639556	0.611830	541	0.2705
B	0.868254	0.785714	32.928571	1.163492	98466.955556	0.384127	0.006349	630	0.3150
C	0.691099	0.979058	29.060209	1.104712	126838.926702	1.107330	1.324607	382	0.1910
D	0.149888	0.277405	49.192394	1.467562	160958.722595	1.364653	1.425056	447	0.2235

```
df_segm_analysis.rename({'A':'well_off', 'B': 'fewer_opportunities',
                         'C': 'career_oriented', 'D': 'standard' }, inplace =True)
```

```
df_segm_kmeans['Labels']  = df_segm_kmeans['Segment_KMeans'].map({'A':'well_off', 'B': 'fewer_opportunities',
                         'C': 'career_oriented', 'D': 'standard' })
```

```
x_axis = df_segm_kmeans['Age']
y_axis = df_segm_kmeans['Income']

plt.figure(figsize=(10, 8))
sns.scatterplot(x=x_axis, y=y_axis, hue=df_segm_kmeans.Labels, palette=['g', 'r', 'c', 'm'])
plt.title('Segmentation K-means', fontsize=15)
plt.legend(bbox_to_anchor=(1, 0.5), loc=6)
plt.show()
```




```
from sklearn.decomposition import PCA
```

```
pca = PCA() # So let the PCA variable be an instance of the PCA class.
pca.fit(demo_scaled)
```

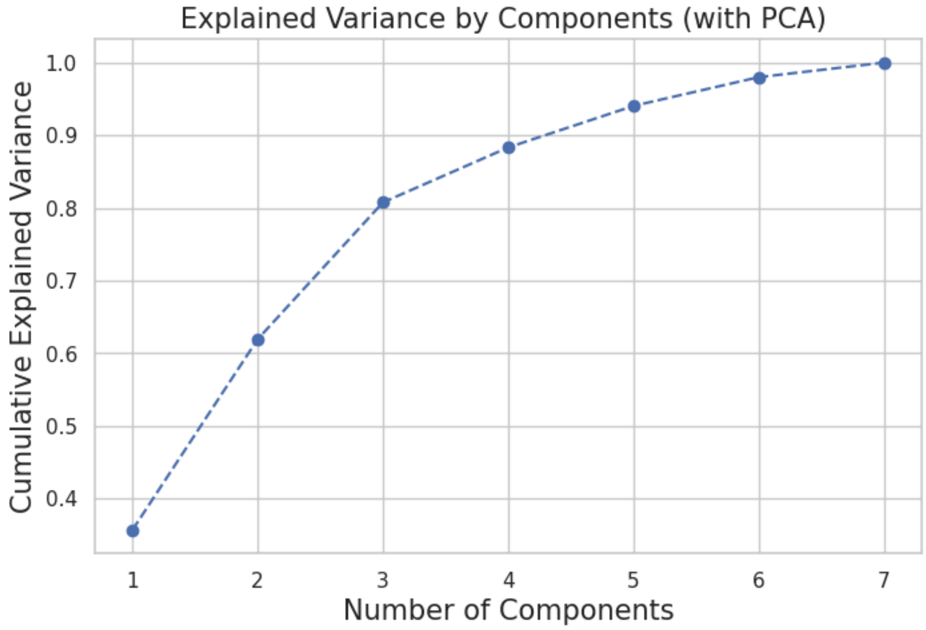
```
▾ PCA
PCA()
```

```
# Explained variance proportion of each component.
pca.explained_variance_ratio_
```

```
array([0.35696328, 0.26250923, 0.18821114, 0.0755775 , 0.05716512,
       0.03954794, 0.02002579])
```

```
plt.figure(figsize =(8,5))
plt.plot(range(1,8), pca.explained_variance_ratio_.cumsum(), marker = 'o', linestyle = '--')
plt.title('Explained Variance by Components (with PCA) ', fontsize= 15)
plt.xlabel('Number of Components', fontsize= 15)
plt.ylabel('Cumulative Explained Variance', fontsize= 15)
```

```
Text(0, 0.5, 'Cumulative Explained Variance')
```




```
pca= PCA(n_components =3)
pca.fit(demo_scaled)
```

```
▾ PCA
PCA(n_components=3)
```

```
# We can obtain more information about the three components with the help of the components attribute of PCA.
pca.components_
```

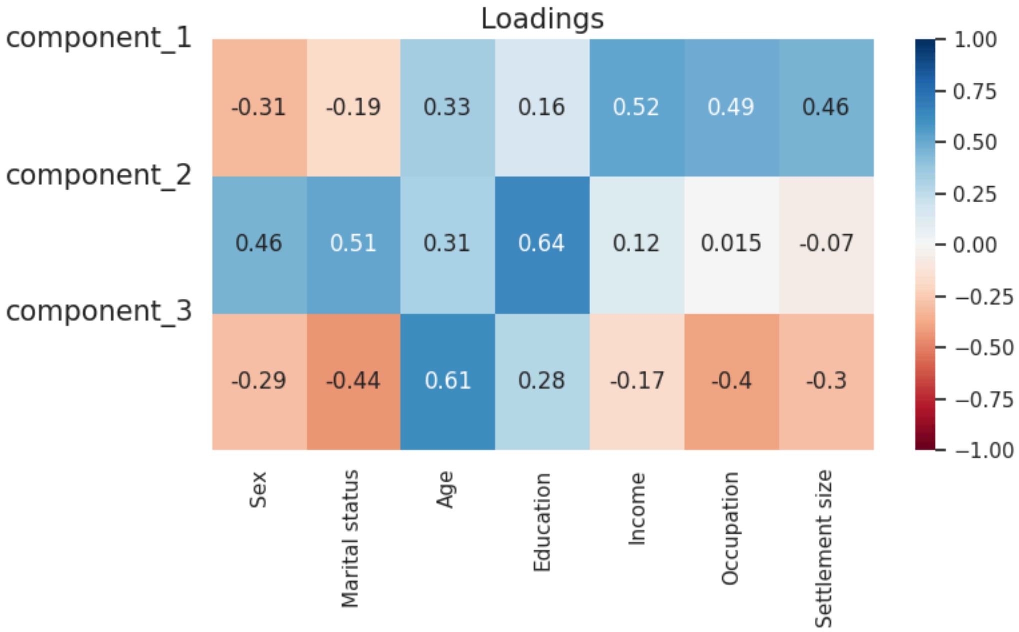
```
array([[-0.31469524, -0.19170439,  0.32609979,  0.15684089,  0.52452463,
         0.49205868,  0.46478852],
       [ 0.45800608,  0.51263492,  0.31220793,  0.63980683,  0.12468314,
         0.01465779, -0.06963165],
       [-0.29301261, -0.44197739,  0.60954372,  0.27560461, -0.16566231,
        -0.39550539, -0.29568503]])
```

```
df_pca_comp = pd.DataFrame(data = pca.components_, columns =demo_df.columns.values,
                           index = ['component_1', 'component_2', 'component_3'])
df_pca_comp
```

	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
component_1	-0.314695	-0.191704	0.326100	0.156841	0.524525	0.492059	0.464789
component_2	0.458006	0.512635	0.312208	0.639807	0.124683	0.014658	-0.069632
component_3	-0.293013	-0.441977	0.609544	0.275605	-0.165662	-0.395505	-0.295685

```
plt.figure(figsize=(8,4))
sns.heatmap(df_pca_comp, vmin = -1, vmax = 1, cmap = 'RdBu', annot = True)
plt.yticks([0,1,2], ['component_1', 'component_2', 'component_3'],rotation = 0, fontsize = 15 )
plt.title('Loadings',  fontsize = 15 )
```

```
Text(0.5, 1.0, 'Loadings')
```



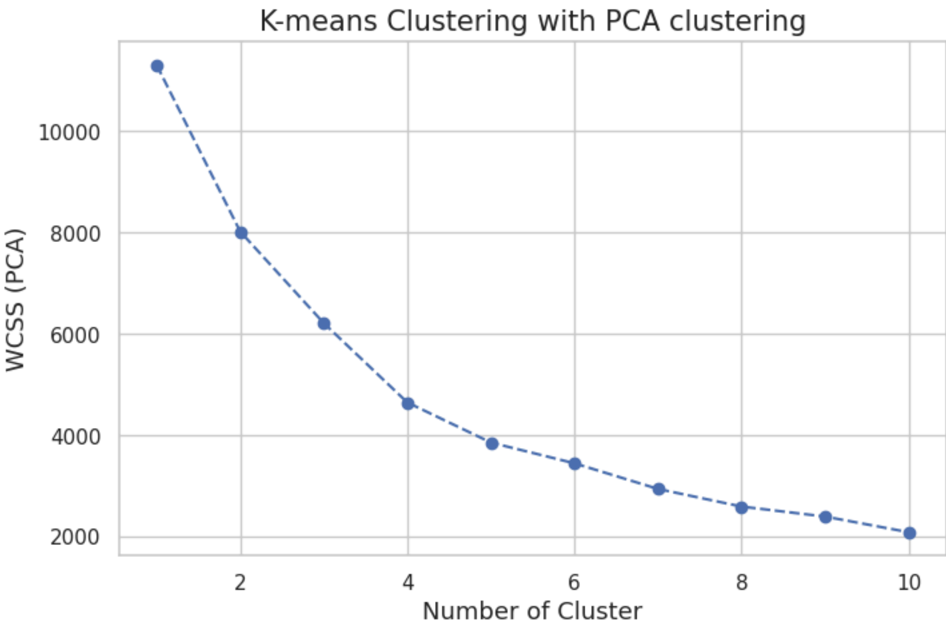

```
scores_pca = pca.transform(demo_scaled)
scores_pca
```

```
array([[ 2.51474593,  0.83412239,  2.1748059 ],
       [ 0.34493528,  0.59814564, -2.21160279],
       [-0.65106267, -0.68009318,  2.2804186 ],
       ...,
       [-1.45229829, -2.23593665,  0.89657125],
       [-2.24145254,  0.62710847, -0.53045631],
       [-1.86688505, -2.45467234,  0.66262172]])
```

```
wcss_pca =  {}
for i in range(1, 11):
    kmeans_pca = KMeans(n_clusters =i, init = 'k-means++', random_state= 42)
    kmeans_pca.fit(scores_pca) # Note that the component scores are standardized by definition
    wcss_pca[i] = kmeans_pca.inertia_
```

```
# Elbow method
plt.figure(figsize =(8,5))
plt.plot(list(wcss_pca.keys()), list(wcss_pca.values()),  marker = 'o', linestyle = '--' )

plt.xlabel('Number of Cluster', fontsize = 13)
plt.ylabel('WCSS (PCA)', fontsize = 13)
plt.title('K-means Clustering with PCA clustering', fontsize = 15)
```

```
Text(0.5, 1.0, 'K-means Clustering with PCA clustering')
```



```
x, y = list(wcss_pca.keys()), list(wcss_pca.values())
kn = KneeLocator(x, y, curve='convex', direction='decreasing')
print('The optimal number of clusters, suggested by Elbow criterion: ', kn.knee)
```

```
The optimal number of clusters, suggested by Elbow criterion:  4
```

```
kmeans_pca = KMeans(n_clusters =4, init = 'k-means++', random_state = 42)
kmeans_pca.fit(scores_pca)
```

```
KMeans
KMeans(n_clusters=4, random_state=42)
```

```
df_seg_pca_kmeans = pd.concat([demo_df.reset_index(drop = True), pd.DataFrame(scores_pca)], axis =1)
df_seg_pca_kmeans.columns.values[-3:] =  ['component_1', 'component_2', 'component_3']
df_seg_pca_kmeans['Segment_KMeans_PCA'] = kmeans_pca.labels_
df_seg_pca_kmeans['Segment_KMeans_PCA'].replace({0: "W", 1: "X", 2: "Y", 3:"Z"}, inplace=True)
```

```
<ipython-input-41-ebe03f0d9039>:4: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[c


  df_seg_pca_kmeans['Segment_KMeans_PCA'].replace({0: "W", 1: "X", 2: "Y", 3:"Z"}, inplace=True)
```

```
df_seg_pca_kmeans_freq = df_seg_pca_kmeans.groupby(['Segment_KMeans_PCA']).mean()
df_seg_pca_kmeans_freq['N_Obs'] = df_seg_pca_kmeans[['Segment_KMeans_PCA', 'Sex']].groupby(['Segment_KMeans_PCA'])['Sex'].cc
df_seg_pca_kmeans_freq['Prop_Obs'] = df_seg_pca_kmeans_freq.N_Obs / df_seg_pca_kmeans_freq.N_Obs.sum()
df_seg_pca_kmeans_freq
```

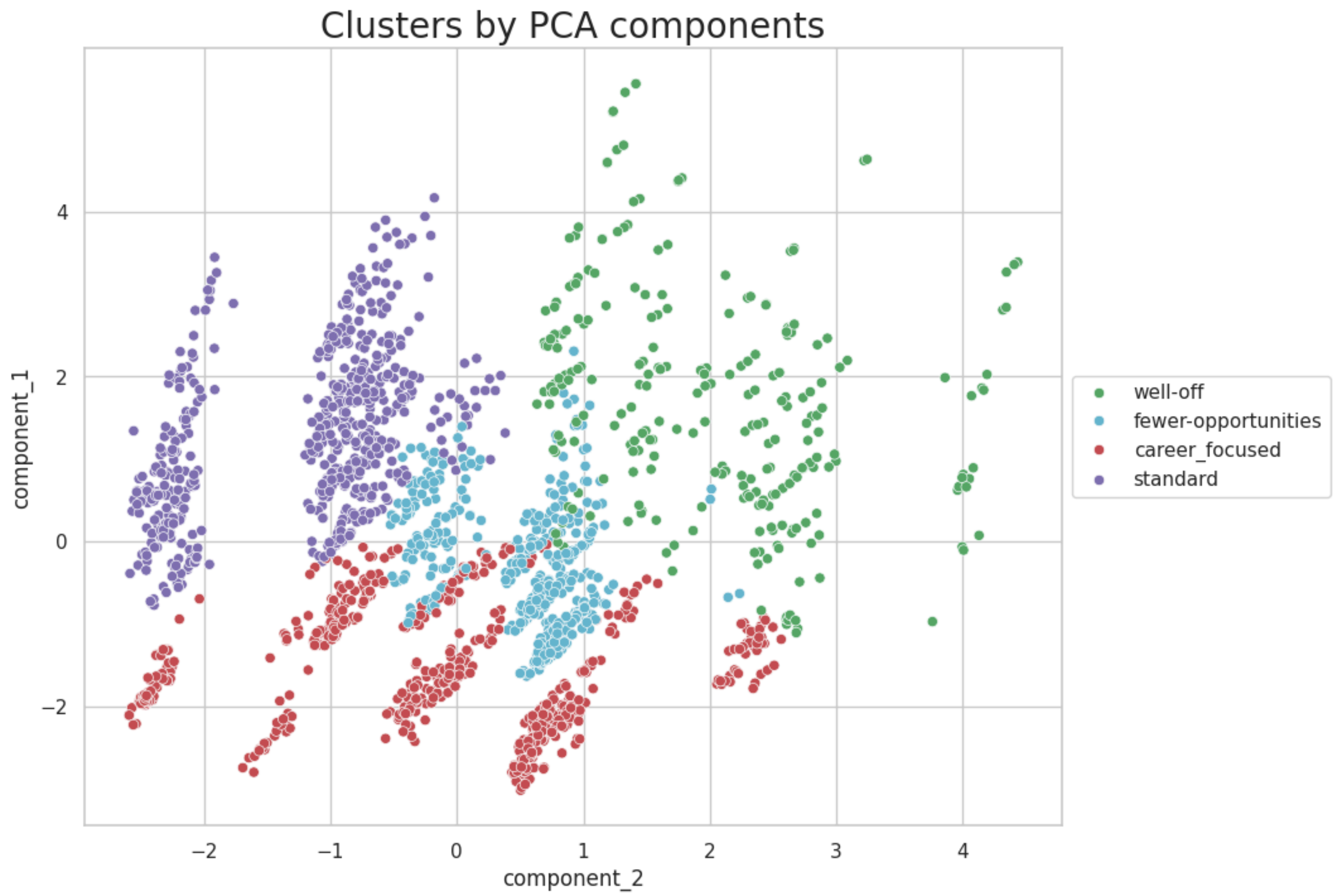
Segment_KMeans_PCA	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	component_1	component_2
W	0.001661	0.041528	36.674419	0.684385	138482.186047	1.200997	1.255814	1.228891	-1.220013
X	0.627869	0.454098	33.473770	0.944262	88824.154098	0.078689	0.009836	-1.607567	-0.110732
Y	0.762357	0.973384	27.889734	1.007605	119503.418251	1.055133	0.813688	-0.395592	0.518043
Z	0.492366	0.683206	55.919847	2.129771	158400.877863	1.125954	1.099237	1.713376	2.021006

```
df_seg_pca_kmeans_freq.rename({'Z':'well-off', 'Y': 'fewer-opportunities',
                        'W': 'standard', 'X': 'career_focused'}, inplace =True)

df_seg_pca_kmeans_freq
```

Segment_KMeans_PCA	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	component_1	component_2
standard	0.001661	0.041528	36.674419	0.684385	138482.186047	1.200997	1.255814	1.228891	-1.220013
career_focused	0.627869	0.454098	33.473770	0.944262	88824.154098	0.078689	0.009836	-1.607567	-0.110732
fewer-opportunities	0.762357	0.973384	27.889734	1.007605	119503.418251	1.055133	0.813688	-0.395592	0.518043
well-off	0.492366	0.683206	55.919847	2.129771	158400.877863	1.125954	1.099237	1.713376	2.021006

```
df_seg_pca_kmeans['Legend'] = df_seg_pca_kmeans['Segment_KMeans_PCA'].map({'Z':'well-off', 'Y': 'fewer-opportunities',
                        'W': 'standard', 'X': 'career_focused'})
```

```
x_axis = df_seg_pca_kmeans['component_2']
y_axis = df_seg_pca_kmeans['component_1']

plt.figure(figsize=(10, 8))
sns.scatterplot(x=x_axis, y=y_axis, hue=df_seg_pca_kmeans['Legend'], palette=['g', 'c', 'r', 'm'])
plt.title('Clusters by PCA components', fontsize=20)
plt.legend(bbox_to_anchor=(1, 0.5), loc=6)
plt.show()
```




```
import pickle # a module used to turn python object into string streams
# Scaler
pickle.dump(scaler, open('scaler.pickle', 'wb'))
# PCA
pickle.dump(pca, open('pca.pickle', 'wb'))
# KMeans PCA
pickle.dump(kmeans_pca, open('kmeans_pca.pickle', 'wb'))
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import silhouette_score, silhouette_samples
from sklearn.preprocessing import StandardScaler
import pickle
import uuid

# Assuming demo_df, scaler, pca, and kmeans_pca are available
# If not, reload them
try:
    with open('scaler.pickle', 'rb') as f:
        scaler = pickle.load(f)
    with open('pca.pickle', 'rb') as f:
        pca = pickle.load(f)
    with open('kmeans_pca.pickle', 'rb') as f:
        kmeans_pca = pickle.load(f)
except FileNotFoundError:
    print("Pickle files not found. Please ensure scaler, pca, and kmeans_pca are serialized.")

# Load data (adjust path as needed)
demo_df = pd.read_csv('segmentation data.csv', index_col=0)

# Standardize features
features = ['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']
X = demo_df[features]
X_scaled = scaler.transform(X)

# Apply PCA
X_pca = pca.transform(X_scaled)

# Get cluster labels
cluster_labels = kmeans_pca.labels_
```

```
n_clusters = kmeans_pca.n_clusters

# 1. Silhouette Score Plot
silhouette_avg = silhouette_score(X_pca, cluster_labels)
sample_silhouette_values = silhouette_samples(X_pca, cluster_labels)

fig, ax = plt.subplots(figsize=(10, 6))
y_lower = 10
for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i
    ax.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values, alpha=0.7, label=f'Cluster {i}')
    y_lower = y_upper + 10

ax.set_title(f'Silhouette Plot for {n_clusters} Clusters (Avg Score: {silhouette_avg:.2f})')
ax.set_xlabel('Silhouette Coefficient')
ax.set_ylabel('Cluster Label')
ax.axvline(x=silhouette_avg, color='red', linestyle='--')
ax.legend()
plt.savefig('silhouette_plot.png')
plt.close()

# 2. Cluster Size Distribution
cluster_counts = pd.Series(cluster_labels).value_counts().sort_index()
fig, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x=cluster_counts.index, y=cluster_counts.values, ax=ax)
ax.set_title('Cluster Size Distribution')
ax.set_xlabel('Cluster')
ax.set_ylabel('Number of Customers')
for i, v in enumerate(cluster_counts.values):
    ax.text(i, v + 10, str(v), ha='center')
plt.savefig('cluster_size_distribution.png')
plt.close()

# 3. Feature Importance (Mean Feature死锁

# 4. Box Plots for Features by Cluster
demo_df['Cluster'] = cluster_labels
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(15, 20))
axes = axes.flatten()
for i, feature in enumerate(features):
    sns.boxplot(x='Cluster', y=feature, data=demo_df, ax=axes[i])
    axes[i].set_title(f'{feature} Distribution by Cluster')
axes[-1].remove()  # Remove extra subplot
plt.tight_layout()
plt.savefig('feature_boxplots.png')
plt.close()

# 5. Correlation Heatmap
corr_matrix = demo_df[features].corr()
fig, ax = plt.subplots(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1, ax=ax)
ax.set_title('Correlation Heatmap of Features')
plt.savefig('correlation_heatmap.png')
plt.close()

# Save cluster profiles as a table
cluster_profiles = demo_df.groupby('Cluster')[features].mean()
cluster_profiles.to_csv('cluster_profiles.csv')


import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples
import pickle
import uuid


# Set visualization style
plt.rc("font", size=14)
sns.set()
sns.set(style="whitegrid", color_codes=True)


# Load data
demo_df = pd.read_csv('segmentation data.csv', index_col=0)
```

```python
# Basic data exploration
print("First 5 rows of the dataset:")
print(demo_df.head())
print("\nDataset Info:")
print(demo_df.info())
print("\nNumber of duplicated rows:", sum(demo_df.duplicated()))
print("\nSummary Statistics:")
print(demo_df.describe())
```

```
First 5 rows of the dataset:
           Sex  Marital status  Age  Education  Income  Occupation  \
ID
100000001    0               0   67          2  124670           1
100000002    1               1   22          1  150773           1
100000003    0               0   49          1   89210           0
100000004    0               0   45          1  171565           1
100000005    0               0   53          1  149031           1

           Settlement size
ID
100000001                2
100000002                2
100000003                0
100000004                1
100000005                1

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 2000 entries, 100000001 to 100002000
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Sex              2000 non-null   int64
 1   Marital status   2000 non-null   int64
 2   Age              2000 non-null   int64
 3   Education        2000 non-null   int64
 4   Income           2000 non-null   int64
 5   Occupation       2000 non-null   int64
 6   Settlement size  2000 non-null   int64
dtypes: int64(7)
memory usage: 125.0 KB
None

Number of duplicated rows: 0

Summary Statistics:
               Sex  Marital status          Age   Education         Income  \
count  2000.000000     2000.000000  2000.000000  2000.00000    2000.000000
mean      0.457000        0.496500    35.909000     1.03800  120954.419000
std       0.498272        0.500113    11.719402     0.59978   38108.824679
min       0.000000        0.000000    18.000000     0.00000   35832.000000
25%       0.000000        0.000000    27.000000     1.00000   97663.250000
50%       0.000000        0.000000    33.000000     1.00000  115548.500000
75%       1.000000        1.000000    42.000000     1.00000  138072.250000
max       1.000000        1.000000    76.000000     3.00000  309364.000000

        Occupation  Settlement size
count  2000.000000      2000.000000
mean      0.810500         0.739000
std       0.638587         0.812533
min       0.000000         0.000000
25%       0.000000         0.000000
50%       1.000000         1.000000
75%       1.000000         1.000000
max       2.000000         2.000000
```

```python
# Feature scaling
features = ['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']
X = demo_df[features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples
import pickle
import uuid
from IPython.display import Image, display
```

```
# Set visualization style
plt.rc("font", size=14)
sns.set()
sns.set(style="whitegrid", color_codes=True)



# Load data
demo_df = pd.read_csv('segmentation data.csv', index_col=0)


# Basic data exploration
print("First 5 rows of the dataset:")
print(demo_df.head())
print("\nDataset Info:")
print(demo_df.info())
print("\nNumber of duplicated rows:", sum(demo_df.duplicated()))
print("\nSummary Statistics:")
print(demo_df.describe())
```

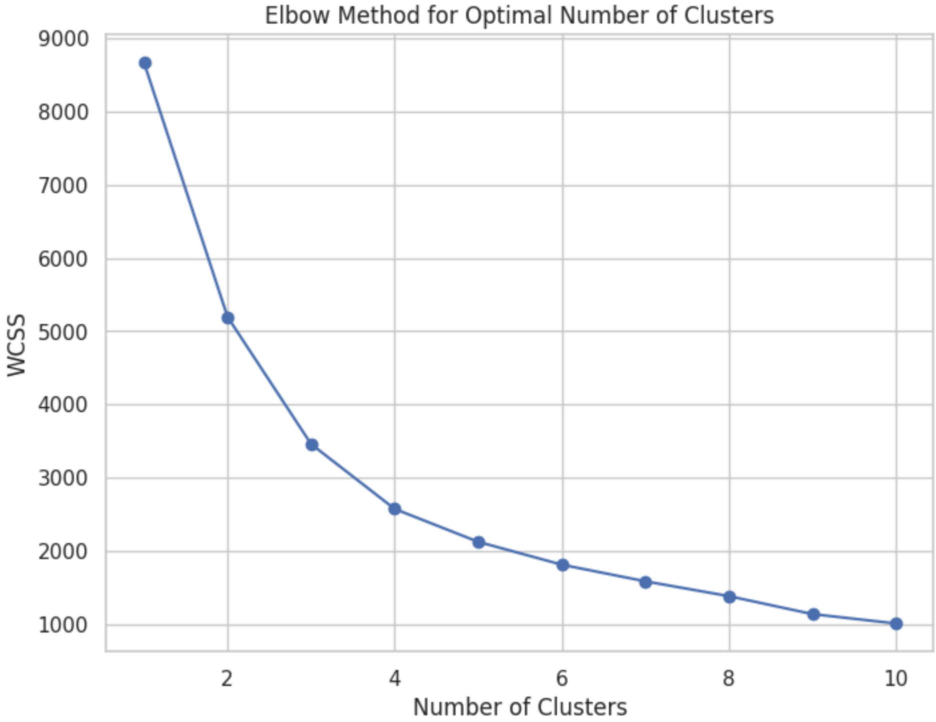
```
First 5 rows of the dataset:
           Sex  Marital status  Age  Education  Income  Occupation  \
ID
100000001    0               0   67          2  124670           1
100000002    1               1   22          1  150773           1
100000003    0               0   49          1   89210           0
100000004    0               0   45          1  171565           1
100000005    0               0   53          1  149031           1

           Settlement size
ID
100000001                2
100000002                2
100000003                0
100000004                1
100000005                1

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
Index: 2000 entries, 100000001 to 100002000
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Sex              2000 non-null   int64
 1   Marital status   2000 non-null   int64
 2   Age              2000 non-null   int64
 3   Education        2000 non-null   int64
 4   Income           2000 non-null   int64
 5   Occupation       2000 non-null   int64
 6   Settlement size  2000 non-null   int64
dtypes: int64(7)
memory usage: 125.0 KB
None

Number of duplicated rows: 0

Summary Statistics:
               Sex  Marital status          Age   Education         Income  \
count  2000.000000     2000.000000  2000.000000  2000.00000    2000.000000
mean      0.457000        0.496500    35.909000     1.03800  120954.419000
std       0.498272        0.500113    11.719402     0.59978   38108.824679
min       0.000000        0.000000    18.000000     0.00000   35832.000000
25%       0.000000        0.000000    27.000000     1.00000   97663.250000
50%       0.000000        0.000000    33.000000     1.00000  115548.500000
75%       1.000000        1.000000    42.000000     1.00000  138072.250000
max       1.000000        1.000000    76.000000     3.00000  309364.000000

        Occupation  Settlement size
count  2000.000000      2000.000000
mean      0.810500         0.739000
std       0.638587         0.812533
min       0.000000         0.000000
25%       0.000000         0.000000
50%       1.000000         1.000000
75%       1.000000         1.000000
max       2.000000         2.000000
```

```
# Feature scaling
features = ['Sex', 'Marital status', 'Age', 'Education', 'Income', 'Occupation', 'Settlement size']
X = demo_df[features]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)


# PCA for dimensionality reduction
pca = PCA(n_components=2)  # Reduce to 2 components for visualization
```

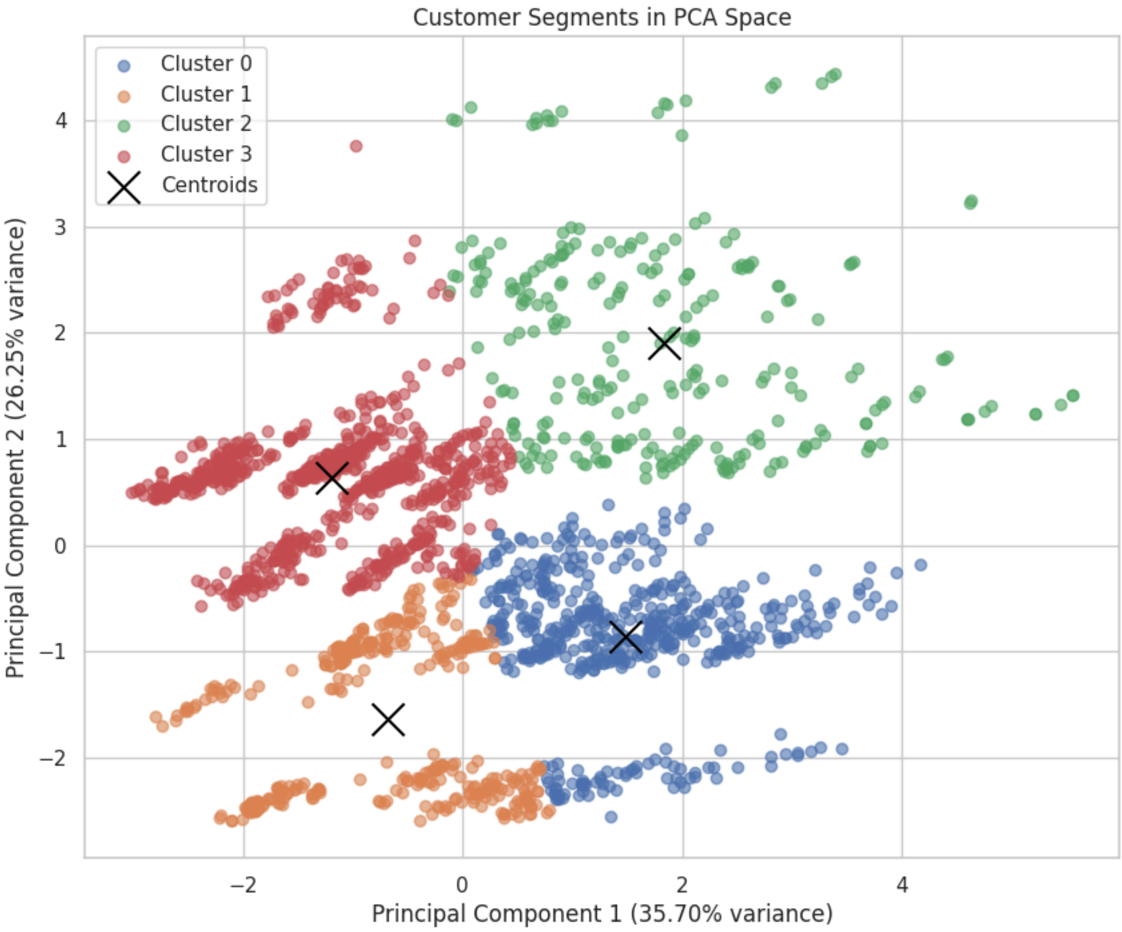
```
X_pca = pca.fit_transform(X_scaled)
explained_variance_ratio = pca.explained_variance_ratio_
print(f"\nExplained Variance Ratio by PCA: {explained_variance_ratio}")
print(f"Total Variance Explained: {sum(explained_variance_ratio):.2f}")
```

```
Explained Variance Ratio by PCA: [0.35696328 0.26250923]
Total Variance Explained: 0.62
```

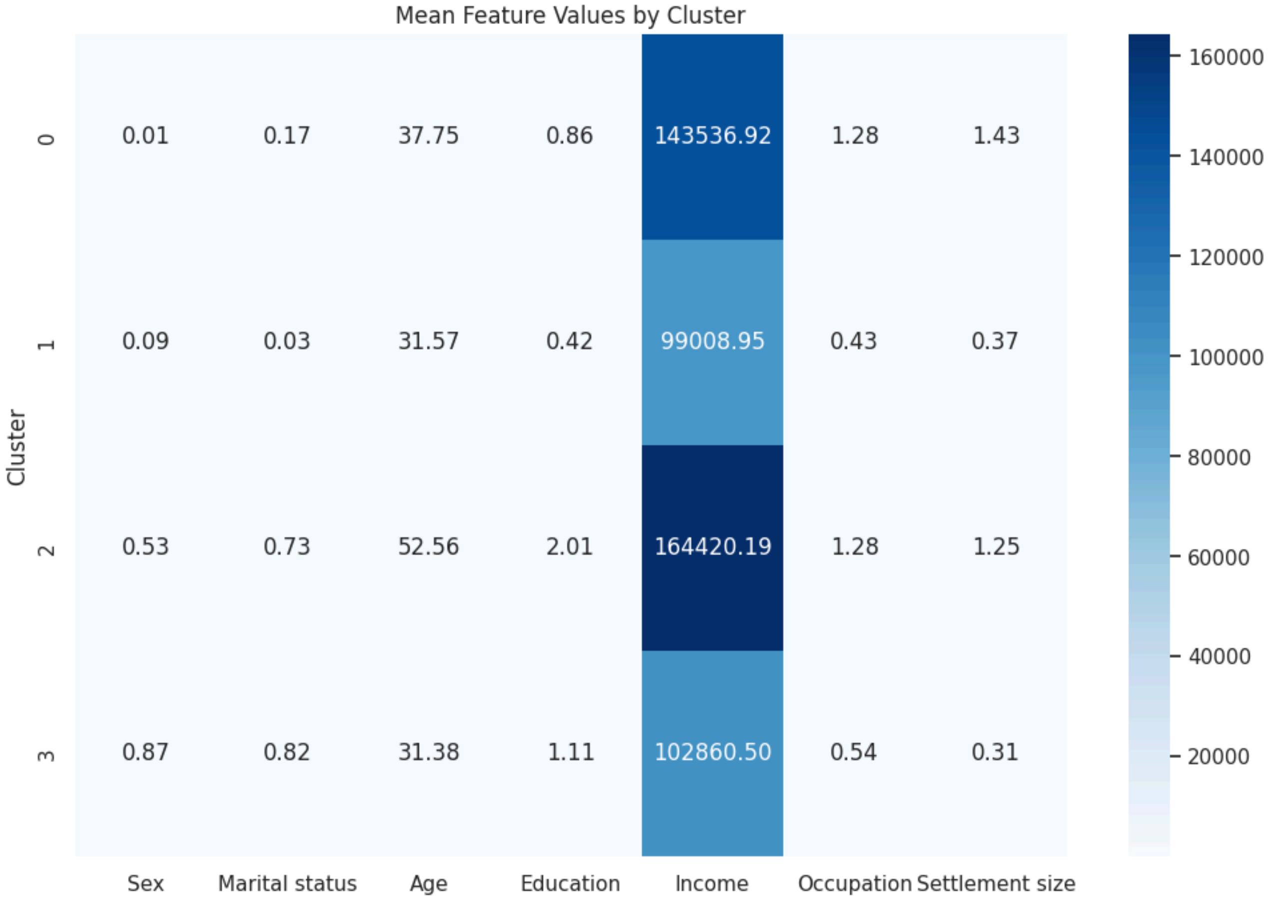
```
# Elbow method to determine optimal number of clusters
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
    kmeans.fit(X_pca)
    wcss.append(kmeans.inertia_)


# Plot elbow curve
plt.figure(figsize=(8, 6))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method for Optimal Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()  # Use show() to display directly
```




```
# Apply KMeans with optimal clusters (assuming 4 clusters based on typical elbow plot)
n_clusters = 4
kmeans_pca = KMeans(n_clusters=n_clusters, init='k-means++', random_state=42)
cluster_labels = kmeans_pca.fit_predict(X_pca)
demo_df['Cluster'] = cluster_labels


# PCA Scatter Plot
plt.figure(figsize=(10, 8))
for i in range(n_clusters):
    plt.scatter(X_pca[cluster_labels == i, 0], X_pca[cluster_labels == i, 1],
                label=f'Cluster {i}', alpha=0.6)
plt.scatter(kmeans_pca.cluster_centers_[:, 0], kmeans_pca.cluster_centers_[:, 1],
            s=300, c='black', marker='x', label='Centroids')
plt.title('Customer Segments in PCA Space')
plt.xlabel(f'Principal Component 1 ({explained_variance_ratio[0]:.2%} variance)')
plt.ylabel(f'Principal Component 2 ({explained_variance_ratio[1]:.2%} variance)')
plt.legend()
plt.show()  # Show plot inline
```

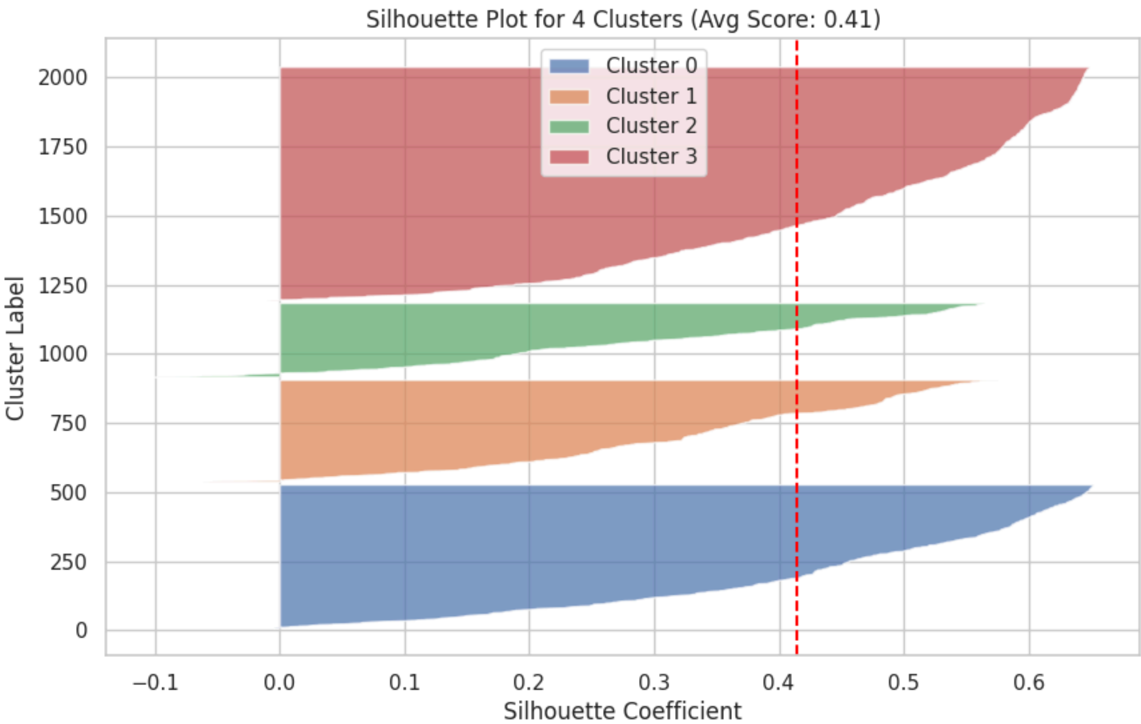


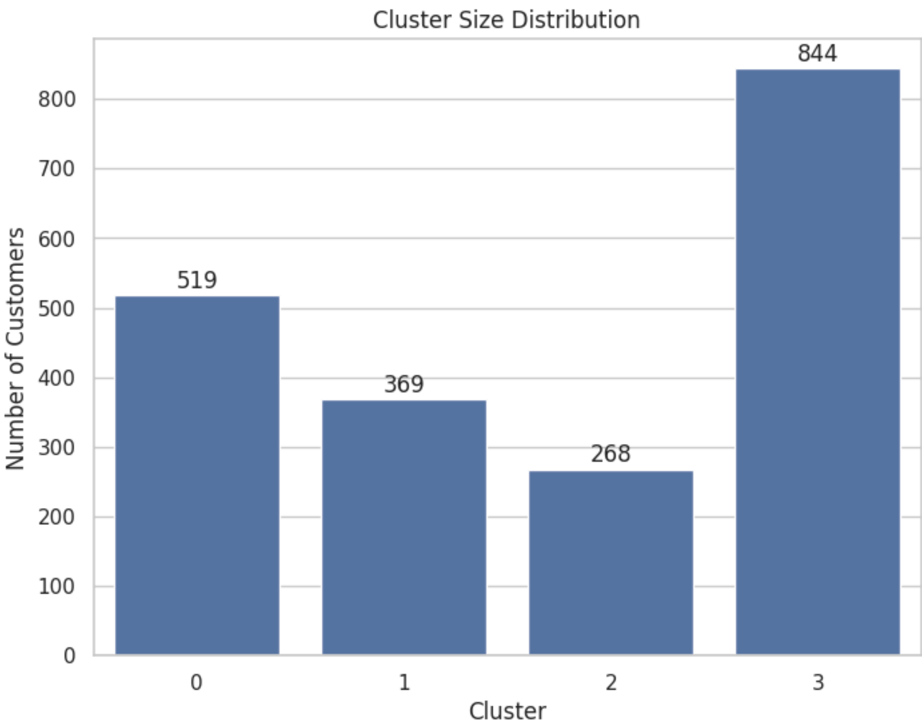

```
# Cluster Profiles (Mean Feature Values)
cluster_profiles = demo_df.groupby('Cluster')[features].mean()
cluster_profiles.to_csv('cluster_profiles.csv')

# Plot Cluster Profiles
plt.figure(figsize=(12, 8))
sns.heatmap(cluster_profiles, annot=True, cmap='Blues', fmt='.2f')
plt.title('Mean Feature Values by Cluster')
plt.show()  # Show plot inline
```

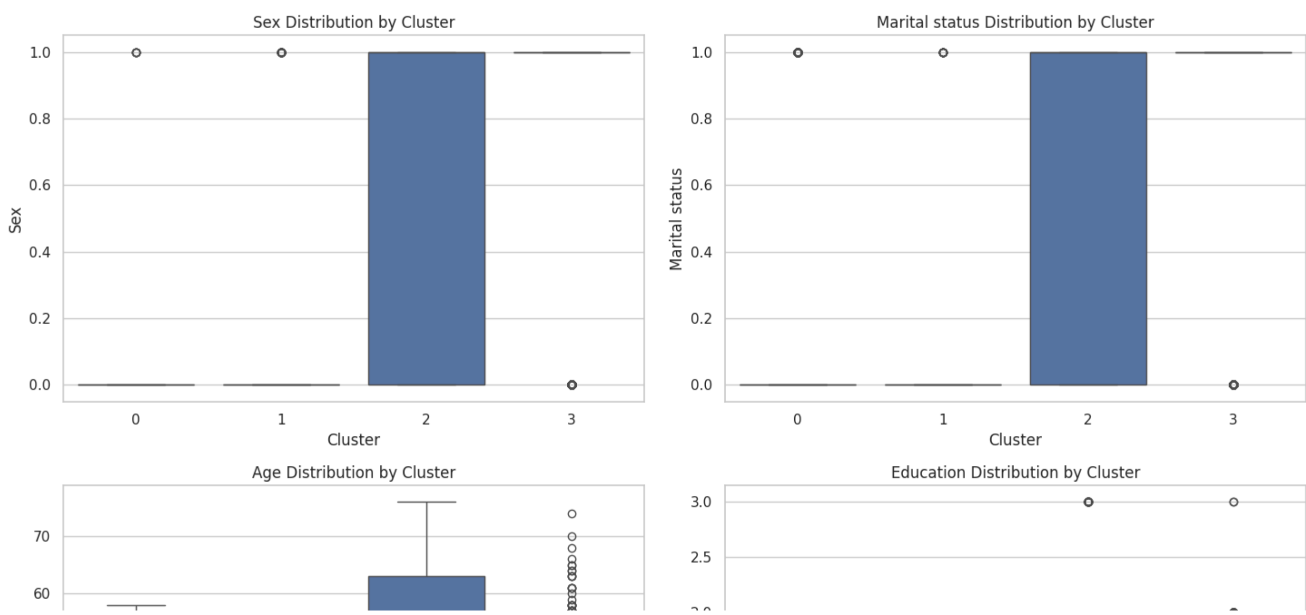



```
# Additional Analytics

# 1. Silhouette Score Plot
silhouette_avg = silhouette_score(X_pca, cluster_labels)
sample_silhouette_values = silhouette_samples(X_pca, cluster_labels)

fig, ax = plt.subplots(figsize=(10, 6))
y_lower = 10
for i in range(n_clusters):
    ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
    ith_cluster_silhouette_values.sort()
    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i
    ax.fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_silhouette_values,
                     alpha=0.7, label=f'Cluster {i}')
    y_lower = y_upper + 10

ax.set_title(f'Silhouette Plot for {n_clusters} Clusters (Avg Score: {silhouette_avg:.2f})')
ax.set_xlabel('Silhouette Coefficient')
ax.set_ylabel('Cluster Label')
ax.axvline(x=silhouette_avg, color='red', linestyle='--')
ax.legend()
plt.show()  # Show plot inline

# 2. Cluster Size Distribution
cluster_counts = pd.Series(cluster_labels).value_counts().sort_index()
fig, ax = plt.subplots(figsize=(8, 6))
sns.barplot(x=cluster_counts.index, y=cluster_counts.values, ax=ax)
ax.set_title('Cluster Size Distribution')
ax.set_xlabel('Cluster')
ax.set_ylabel('Number of Customers')
for i, v in enumerate(cluster_counts.values):
    ax.text(i, v + 10, str(v), ha='center')
plt.show()  # Show plot inline

# 3. Box Plots for Features by Cluster
fig, axes = plt.subplots(nrows=4, ncols=2, figsize=(15, 20))
axes = axes.flatten()
for i, feature in enumerate(features):
    sns.boxplot(x='Cluster', y=feature, data=demo_df, ax=axes[i])
    axes[i].set_title(f'{feature} Distribution by Cluster')
if len(features) < len(axes):
    axes[-1].remove()  # Remove extra subplot
```

```
    plt.tight_layout()
    plt.show()  # Show plot inline
```


Silhouette Plot for 4 Clusters (Avg Score: 0.41)
Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster Label
Silhouette Coefficient
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0
250
500
750
1000
1250
1500
1750
2000


Cluster Size Distribution
519
369
268
844
Number of Customers
Cluster
0
1
2
3


Sex Distribution by Cluster
Sex
Cluster
Marital status Distribution by Cluster
Marital status
Cluster
Age Distribution by Cluster
Education Distribution by Cluster