**Step by Step Customer Segmentation using K-Means in Python**

Segment your customers for better marketing.



**Table of Contents**

1. Introduction — What is Customer Segmentation?
2. Business Scenario
3. Explore the Dataset
4. Data Preprocessing
5. K-Means for Segmentation
6. PCA with K-Means for better visualization
7. Conclusion

**Introduction**

Let’s say, you decided to buy a t-shirt from a brand online. Have you ever thought that who else bought the same t-shirt?

People, who have similar to you, right? Same age, same hobbies, same gender, etc.

*In marketing, Companies basically try to find your t-shirts on other people!*

But wait ? How? Of course, with data!

Customer segmentation is that simple!

We actually try to find and group customers based on common characteristics such as age, gender, living area, spending behavior, etc. So that we can market the customers effectively.

*Let’s dive into our segmentation project!*

**Business Scenario**

Suppose we are working as a data scientist for a FMCG company and want to segment our customers to help the marketing department for them to launch new products and sales based on the segmentation. Therefore we will save our time and money by marketing a specific group of customers with selected products.

*How Did we collect the data by the way?*

All data has been collected through the loyalty cards they use at checkout :)

We will utilize K-Means and PCA algorithms for this project and see how we define new grouped customers!

**Understanding Data is Important!**

Before starting any project, We need to understand the business problem and dataset first.

Let’s see the variables(features) in the dataset.

**Variable Description**

ID: Shows a unique identification of a customer.

Sex: Biological sex (gender) of a customer. In this dataset, there are only 2 different options.

*0: male*

*1: female*

Marital status: Marital status of a customer.

*0: single*

*1: non-single (divorced / separated / married / widowed)*

Age: The age of the customer in years, calculated as current year minus the year of birth of the customer at the time of the creation of the dataset

*18 Min value (the lowest age observed in the dataset)*

*76 Max value (the highest age observed in the dataset)*

Education: Level of education of the customer.

*0:other / unknown*

*1: high school*

*2: university*

*3: graduate school*

Income: Self-reported annual income in US dollars of the customer.

*35832 Min value (the lowest income observed in the dataset)*

*309364 Max value (the highest income observed in the dataset)*

Occupation: Category of occupation of the customer.

*0: unemployed/unskilled*

*1: skilled employee / official*

*2: management / self-employed / highly qualified employee / officer*

Settlement size: The size of the city that the customer lives in.

*0: small city*

*1: mid-sized city*

*2: big city*

We have datasets and know the business problem. Now, Let’s start coding!

**Importing Libraries**

In this project, we will need some friends that help you along the way!

Let me introduce them below,

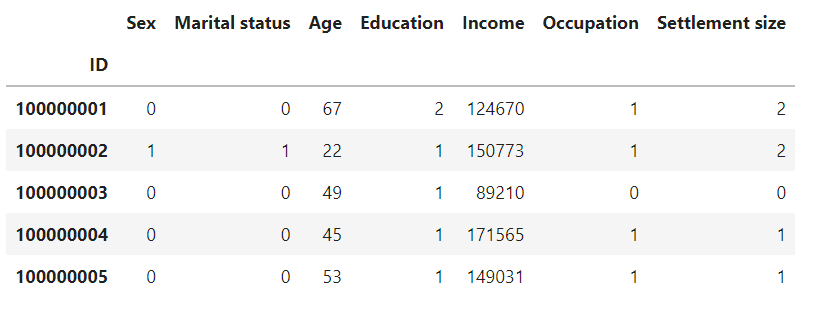
### Data Analysis and Manipulation   
import pandas as pd   
import numpy as np### Data Visualizationimport matplotlib.pyplot as plt  
import seaborn as snssns.set() ## this is for styling### Data Standardization and Modeling with K-Means and PCAfrom sklearn.preprocessing import StandardScalerfrom sklearn.cluster import KMeans  
from sklearn.decomposition import PCA

**3. Explore the Dataset**

df= pd.read\_csv('segmentation data.csv', index\_col = 0)

This part consists of understanding data with the help of descriptive analysis and visualization.

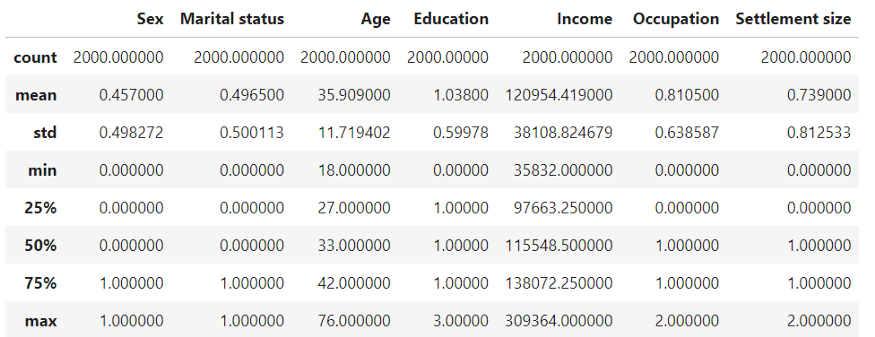
df.head()



df.head() output

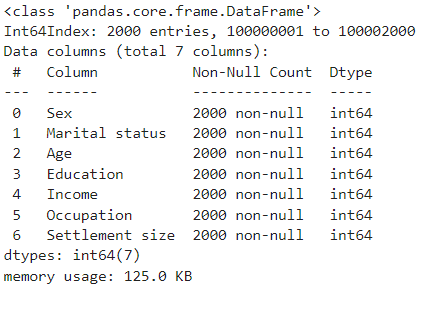
We can also apply the describe method to see descriptive statistics about the columns.

df.describe()



We see the mean of Age and Income 35.90 and 120954 respectively. Describe method is very useful for numerical columns.

df.info()



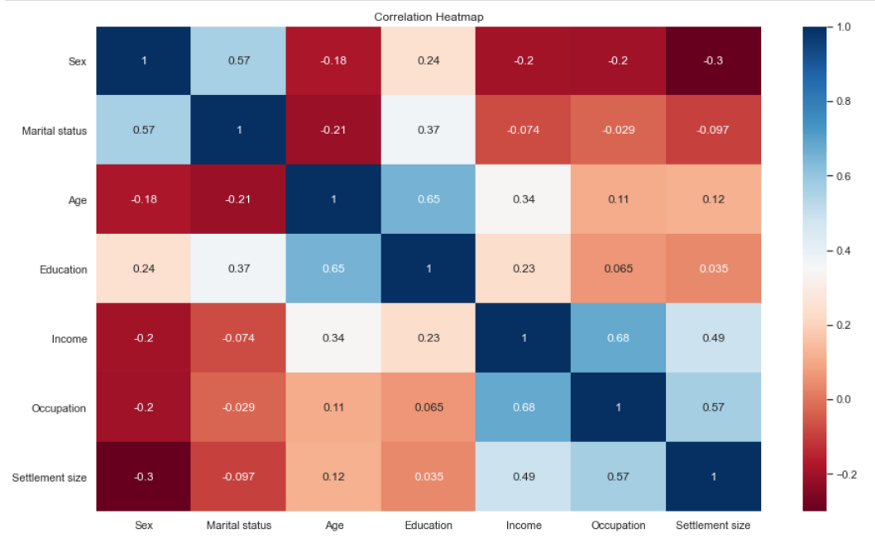
df.info() method returns information about the DataFrame including the index data type and columns, non-null values, and memory usage.

We see that there is no missing value in the dataset and all the variables are integer.

A good way to get an initial understanding of the relationship between the different variables is to explore how they correlate.

We calculate the correlation between our variables using corr method in the pandas library.

plt.figure(figsize=(12,9))  
sns.heatmap(**df.corr()**,annot=True,cmap='RdBu')  
plt.title('Correlation Heatmap',fontsize=14)  
plt.yticks(rotation =0)  
plt.show()



*Let’s explore the correlation.*

We see that there is a strong correlation between Education and Age. In other words, older people tend to be more highly educated.

How about income and occupation?

Their correlation is 0.68. That means If you have a higher salary, you are more likely to have a higher-level occupation such as a manager.

Correlation matrix is a very useful tool to analyze the relationship between features.

Now, we understand our dataset and have a general idea of it.

Next section will be the segmentation. But before that, we need to scale our data first.

**4. Data Preprocessing**

We need to apply standardization to our features before using any distance-based machine learning model such as K-Means, KNN.

In general, We want to treat all the features equally and we can achieve that by transforming the features in such a way that their values fall within the same numerical range such as [0:1].

This process is commonly referred to as Standardization.

**Standardization**

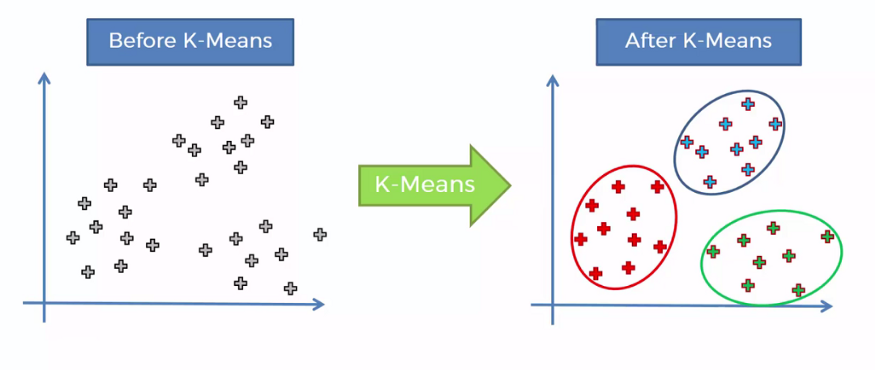
Now that we cleared that up. It is time to perform standardization in Python.

scaler = StandardScaler()  
df\_std = scaler.fit\_transform(df)

Now, We are all set to start building segmentation model K-Means!

df\_std = pd.DataFrame(data = df\_std,columns = df.columns)

Building Our Segmentation Model



Before applying the K-Means algorithm we need to choose how many clusters we would like to have.

But How?

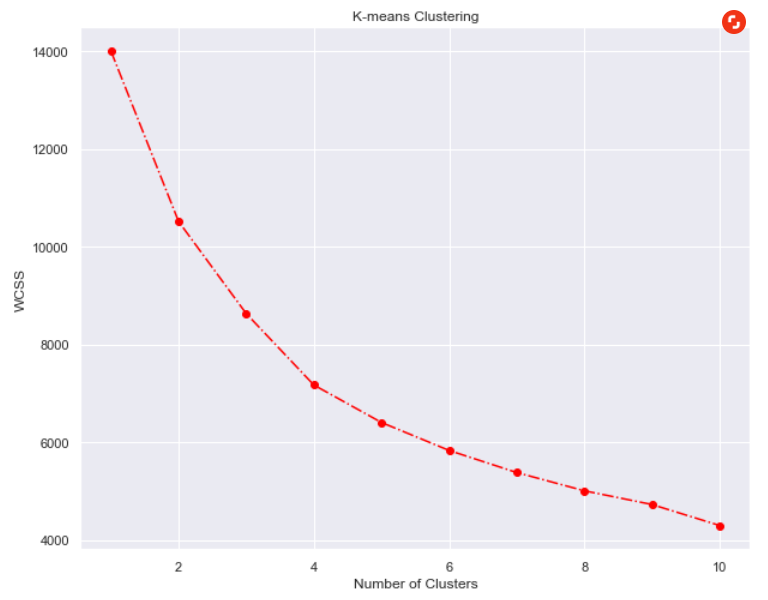
There are two components. *Within Clusters Sum of Squares(WCSS)* and *Elbow Method.*

wcss = []  
for i in range(1,11):  
 kmeans\_pca = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)  
 kmeans\_pca.fit(scores\_pca)  
 wcss.append(kmeans\_pca.inertia\_)

We stored to each within clusters sum of squared value to wcss list.

Let’s visualize them.

plt.figure(figsize = (10,8))  
plt.plot(range(1, 11), wcss, marker = 'o', linestyle = '-.',color='red')  
plt.xlabel('Number of Clusters')  
plt.ylabel('WCSS')  
plt.title('K-means Clustering')  
plt.show()



The elbow in the graph is the four-cluster mark. This is the only place until which the graph is steeply declining while smoothing out afterward.

Let’s perform K-Means clustering with 4 clusters.

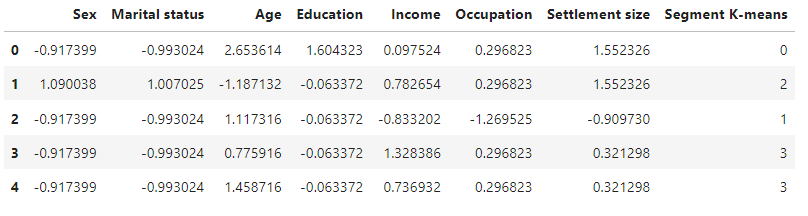
kmeans = KMeans(n\_clusters = 4, init = 'k-means++', random\_state = 42)

Fitting Our Model to the Dataset

kmeans.fit(df\_std)

# We create a new data frame with the original features and add a new column with the assigned clusters for each point.

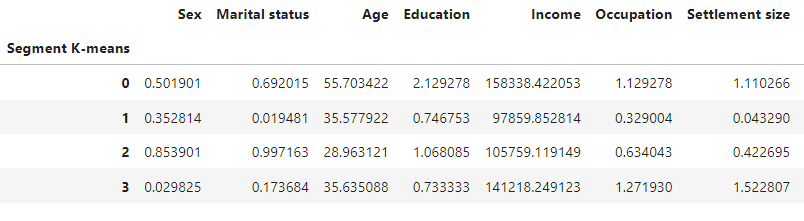
df\_segm\_kmeans= df\_std.copy()  
df\_std[‘Segment K-means’] = kmeans.labels\_



We now see the segments with our dataset.

Let’s group the customers by clusters and see the average values for each variable.

df\_segm\_analysis = df\_std.groupby(['Segment K-means']).mean()  
df\_segm\_analysis



*It’s time to interpret our new dataset,*

*Let’s start with the first segment,*

It has almost the same number of men and women with an average age of 56. Compared to other clusters, we realize that this is the oldest segment.

*For the second segment, we can say,*

This segment has the lowest values for the annual salary.

They live almost exclusively in small cities

With low income living in small cities, it seems that this is a segment of people with fewer opportunities.

*Let’s carry on with the third segment,*

This is the youngest segment with an average age of 29. They have medium level of education and average income.

They also seem average about every parameter we can label the segment average or standard.

*Finally, we come to the fourth segment,*

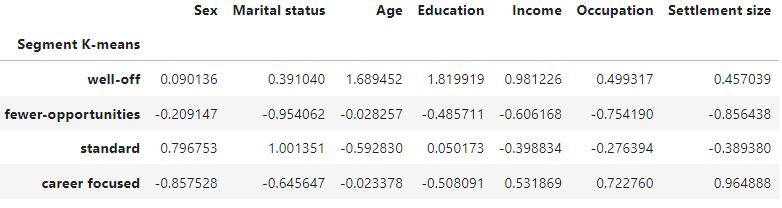
It is comprised almost entirely of men, less than 20 percent of whom are in relationships.

Looking at the numbers, we observe relatively low values for education, paired with high values for income and occupation.

The majority of this segment lives in big or middle-sized cities.

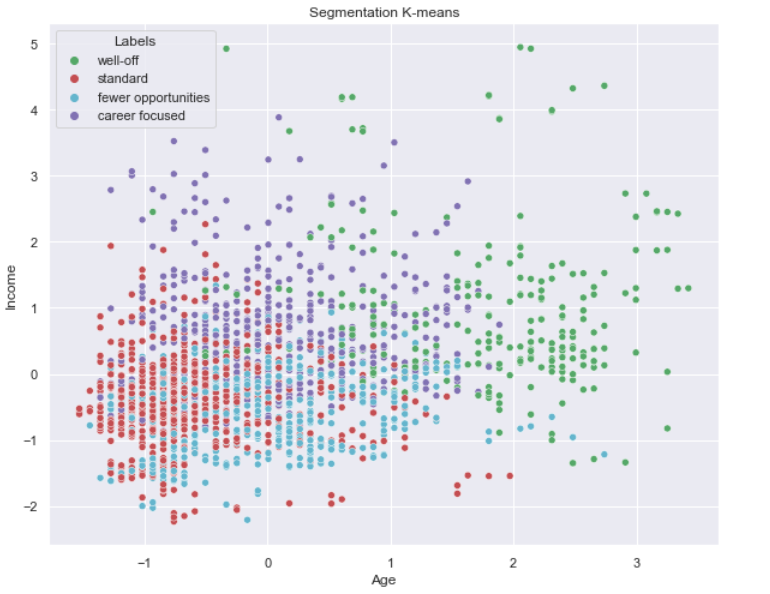
*Let’s label the segment according to their relevance.*

df\_segm\_analysis.rename({0:'well-off',  
 1:'fewer-opportunities',  
 2:'standard',  
 3:'career focused'})



Finally, we can create our plot to visualize each segment.

x\_axis = df\_std['Age']  
y\_axis = df\_std['Income']  
plt.figure(figsize = (10, 8))  
sns.scatterplot(x\_axis, y\_axis, hue = df\_std['Labels'], palette = ['g', 'r', 'c', 'm'])  
plt.title('Segmentation K-means')  
plt.show()



We can see the green segment well off is clearly separated as it is highest in both age and income. But the other three are grouped together.

We can conclude that K-Means did a decent job! However, it’s hard to separate segments from each other.

In the next section, we will combine PCA and K-Means to try to get a better result.

**PCA with K-Means for Better Visualization**

What we will do here is apply dimensionality reduction to simplify our problem.

We will choose reasonable components in order to obtain a better clustering solution than with the standard K-Means. So that We aim to see a nice and clear plot for our segmented groups.

pca = PCA()  
pca.fit(df\_std)

Now, Let’s see the explained variance ratio by each component.

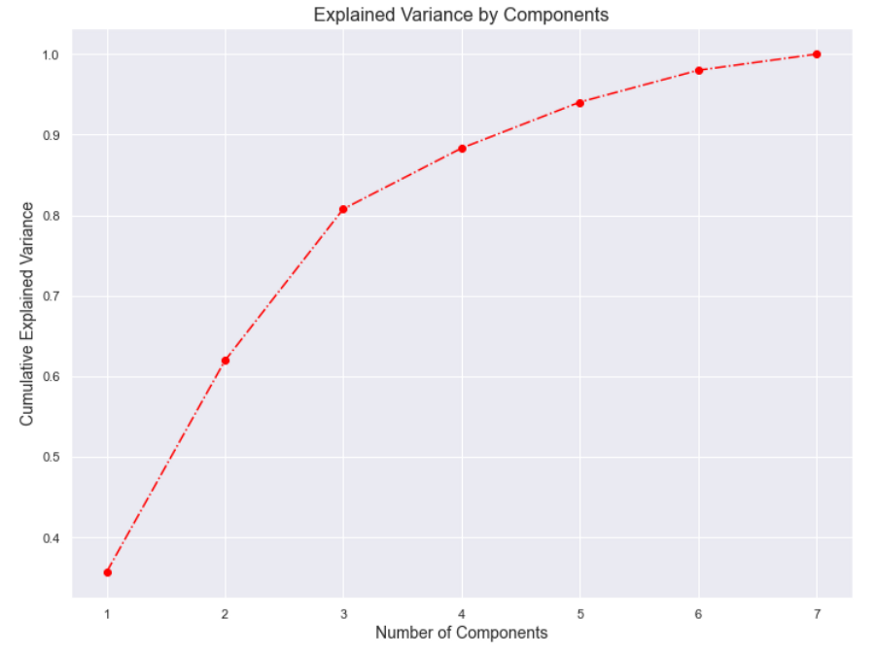
pca.explained\_variance\_ratio\_



We observe that the first component explains around 36 % of the variability of the data. The second one is 26 % and so on.

We now can plot the cumulative sum of explained variance.

plt.figure(figsize = (12,9))  
plt.plot(range(1,8), pca.explained\_variance\_ratio\_.cumsum(), marker = 'o', linestyle = '--')  
plt.title('Explained Variance by Components')  
plt.xlabel('Number of Components')  
plt.ylabel('Cumulative Explained Variance')

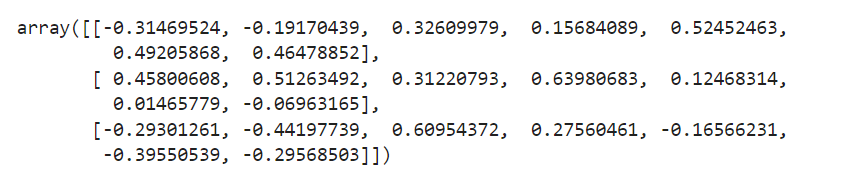


Well, How do we choose the right number of components? The answer is there is no right or wrong answer for that.

*But, a rule of thumb is to keep at least 70 to 80 percent of the explained variance.*

80 % of the variance of the data is explained by the first 3 components. Let’s keep the first 3 components for our further analysis.

pca = PCA(n\_components = 3)  
pca.fit(df\_std)  
pca.components\_



The result is a 3 by 7 array. We reduced our futures to three components from the original seven values that explain the shape the values themselves show the so-called loadings.

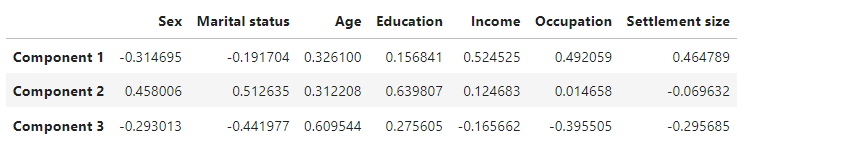
*Hey, just a minute, what is loading then?*

Loadings are correlations between an original variable and the component.

For instance, the first value of the array shows the loading of the first feature on the first component.

Let’s put this information in a pandas data frame so that we can see them nicely. Columns are seven original features and rows are three components that PCA gave us.

df\_pca\_comp = pd.DataFrame(data = pca.components\_,  
 columns = df.columns,  
 index = ['Component 1', 'Component 2', 'Component 3'])  
df\_pca\_comp



plt.figure(figsize=(12,9))  
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 = 45,  
 fontsize = 12)  
plt.title('Components vs Original Features',fontsize = 14)  
plt.show()



We see that there is a positive correlation between Component1 and *Age,Income, Occupation and Settlement size*. These are strictly related to the career of a person. So this component shows the career focus of the individual.

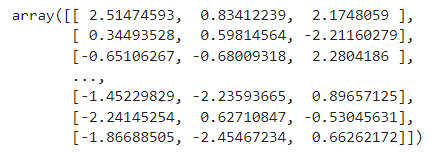
For the second component *Sex, Marital status and Education* are by far the most prominent determinants.

For the final component, we realize that Age, Marital Status, and Occupation are the most important features. We observed that marital status and occupation load negatively but are still important.

Now, we have an idea about our new variables(components). We can clearly see the relationship between components and variables.

Let’s transform our data and save it scores\_pca.

pca.transform(df\_std)  
scores\_pca = pca.transform(df\_std)

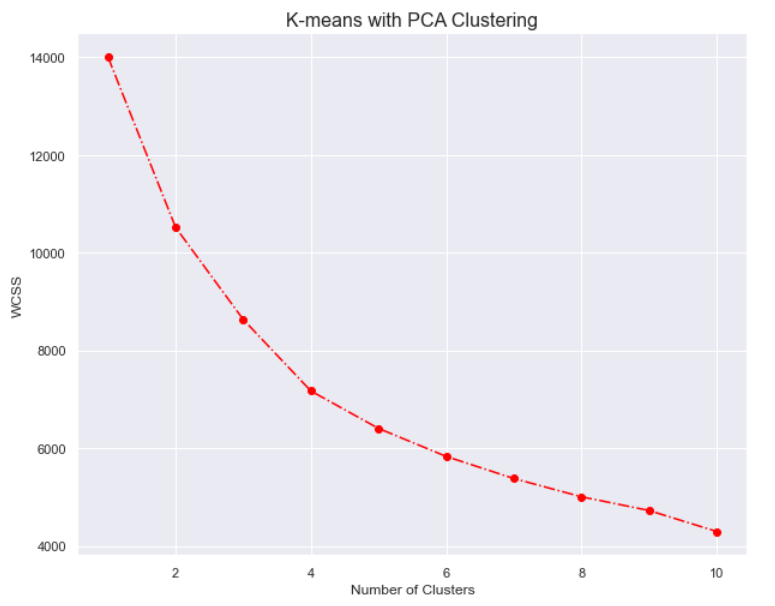


**K-means clustering with PCA**

Our new dataset is ready! It’s time to apply K-Means to our*brand new dataset* with 3 components.

It is as simple as before! We follow the same steps with standard K-Means.

wcss = []  
for i in range(1,11):  
 kmeans\_pca = KMeans(n\_clusters = i, init = 'k-means++', random\_state = 42)  
 kmeans\_pca.fit(scores\_pca)  
 wcss.append(kmeans\_pca.inertia\_)

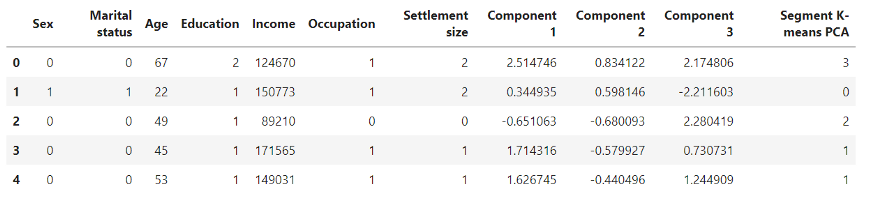


We see that the optimal cluster number by within sum of square is 4.

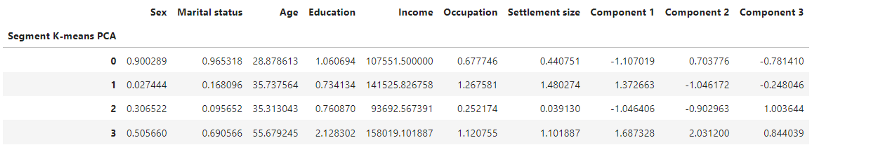
kmeans\_pca = KMeans(n\_clusters = 4, init = 'k-means++', random\_state = 42)  
kmeans\_pca.fit(scores\_pca)

K-Means algorithm has learnt from our new components and created 4 clusters . I would like to see old datasets with new components and labels .

df\_segm\_pca\_kmeans = pd.concat([df.reset\_index(drop = True), pd.DataFrame(scores\_pca)], axis = 1)df\_segm\_pca\_kmeans.columns.values[-3: ] = ['Component 1', 'Component 2', 'Component 3']  
df\_segm\_pca\_kmeans['Segment K-means PCA'] = kmeans\_pca.labels\_df\_segm\_pca\_kmeans.head()



# We calculate the means by segments.  
df\_segm\_pca\_kmeans\_freq = df\_segm\_pca\_kmeans.groupby(['Segment K-means PCA']).mean()  
df\_segm\_pca\_kmeans\_freq



Above we see our data grouped by K-Means Segment. We can also convert segment numbers to the label and see the observation number and proportions of each segment by total observation.

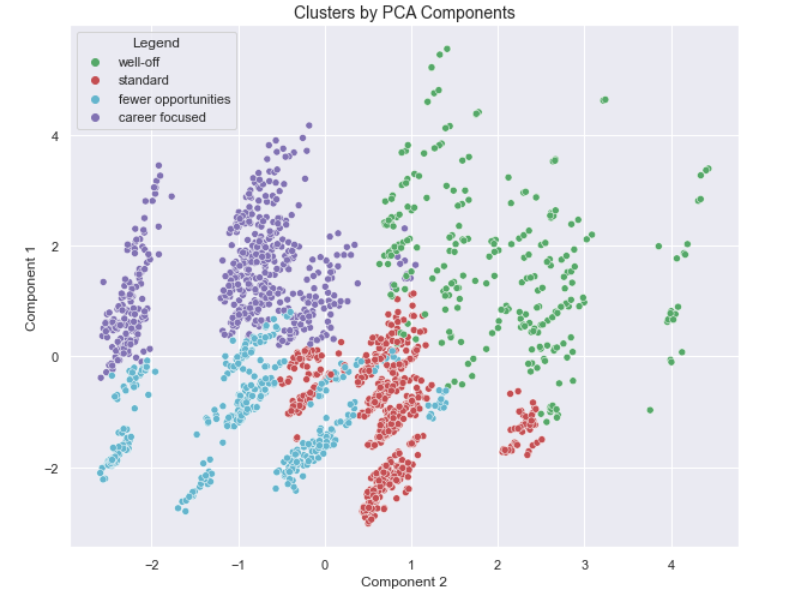
df\_segm\_pca\_kmeans\_freq['N Obs'] = df\_segm\_pca\_kmeans[['Segment K-means PCA','Sex']].groupby(['Segment K-means PCA']).count()  
df\_segm\_pca\_kmeans\_freq['Prop Obs'] = df\_segm\_pca\_kmeans\_freq['N Obs'] / df\_segm\_pca\_kmeans\_freq['N Obs'].sum()  
df\_segm\_pca\_kmeans\_freq = df\_segm\_pca\_kmeans\_freq.rename({0:'standard',   
 1:'career focused',  
 2:'fewer opportunities',   
 3:'well-off'})  
df\_segm\_pca\_kmeans\_freq



We obtained columns and changed the name with a few line of codes.Now, Let’s plot the our new segments and see differences.

As you can probably recall, our previous four clusters were standard,career focused, fewer opportunities.

df\_segm\_pca\_kmeans['Legend'] = df\_segm\_pca\_kmeans['Segment K-means PCA'].map({0:'standard',   
 1:'career focused',  
 2:'fewer opportunities',   
 3:'well-off'})  
x\_axis = df\_segm\_pca\_kmeans['Component 2']  
y\_axis = df\_segm\_pca\_kmeans['Component 1']  
plt.figure(figsize = (10, 8))  
sns.scatterplot(x\_axis, y\_axis, hue = df\_segm\_pca\_kmeans['Legend'], palette = ['g', 'r', 'c', 'm'])  
plt.title('Clusters by PCA Components')  
plt.show()



When we plotted the K means clustering solution without PCA, we were only able to distinguish the green segment, but the division based on the components is much more pronounced.

That was one of the biggest goals of PCA to reduce the number of variables by combining them into bigger ones.

*“Don’t find customers for your products, find products for your customers.”*

*— Seth Godin*

**Conclusion**

We segmented our customers into 4 groups. We are ready to start to choose our groups based on our aims and marketing them!

Segmentation helps marketers to be more efficient in terms of time, money and other resources.

They gain a better understanding of customer's needs and wants and therefore can tailor campaigns to customer segments most likely to purchase products.