CC7183 Data Analysis and Visualization

Customer segmentation using K-means clustering

Massan Sarsenbayev, London Met Student ID: 21029552

15/12/2022

Contents

Customer segmentation using K-means clustering

Massan Sarsenbayev
School of Computing and Digital Media
London Metropolitan University
London, 166-220 Holloway Road
Student ID - 21029552

***Abstract*—There is and old saying "without data, you are just another person with an opinion." Data is not just numbers or meaningful digits. Data is no longer just a tool to analyze the past, but can be used to make informed decisions. And most importantly can help us to understand and forecast what may happen in the future. Data in current digital advanced era is one of the valuable recources.**

INTRODUCTION

Although we understand the importance of the data, it is crucial to properly manage data. It is a first step in effectively analysing data which leads to important insights. The volume and speed data is generated has grown beyond human comprehension in last decades. Even most advanced technologies sometimes cannot handle all the available datasets. And this is where data analytics steps in. Data can contain a variety of useful information and studying it collectively can uncover very minor patterns and details.

"Things get done only if the data we gather can inform and inspire those in a position to make a difference." -Dr. Mike Schmoker.

With the advancement of a big data, there has been a shift from experience-based decision-making to a data-driven decision making in upper management in businesses. Although data-driven decision is powerful tool, but does not undermine the experience and expertise of the leaders. Insights derived from data can help managers to approach decision-making process on more reliable and concrete ground. Data analytics can aid businesses in improving operations in different levels by assessing metrics for effectiveness and process outcomes. It also helps to identify financially unviable processes replacing it with more enhanced streamlines. Data analytics enable businesses to constantly monitor market trends to improve customer service or launch a new product that can be positively received by customers. For example, retailers can analyse customer purchasing process to better understand the demand and their customer's buying habits. Even more, businesses can quickly determine advertisements for their target demographic, which is the main subject of this project.

LITERATURE REVIEW AND METHODOLOGY

The customer segmentation concept was developed in the mid-1950s by American marketing expert Wendell R. Smith. The idea of the concept is to segment customers according to their demographics, geographics and behaviours. This concept became even more important nowadays with advancement of the technology. As more and more businesses created every day, it has become crucial for the businesses to apply marketing strategies to stay in the market. As the customer basis getting more and more it is becoming difficult for companies to cater the needs of every customer. And this is where Data analytics techniques plays important role. One of the customer segmentation models is K-means clustering. According to Baker [1] customer segmentation refers to grouping customers by their demographics like gender, age, marital status and education. Also, other types of segmentation can be used, for instance, it can be geographic, psychographic, technographic, behavioural, needs-based and value-based. Every of these segmentation models require specific information or dataset about customers. For geographic segmentation will be needed location information, for psychographic segmentation will be needed features like personality, attitude, values, interests. For technographic segmentation mobile-use, desktop-use, apps and software information are required. K-means is the easiest algorithm and one of the powerful for clustering large sets of data. The number of clusters or K value is calculated by "elbow method". After the number of clusters are determined, centroids are calculated by the Euclidian distance. et al. [2].

DATA UNDERSTANDING

Effective data analysis requires an understanding of the problem we are trying to solve. We need to know the questions we are hoping the get answer. The next step is to identify what information or what variables we need in order to answer those questions. In our case it is customer information with variables as below:

- customer_id - unique identification of a customer
- gender - gender of a customer
  - 0:Male
  - 1:Female
- marital_status - marital status of a customer
  - 0:Single
  - 1:Non-single
- age - age of a customer
- education - education level of a customer
  - 0:Unknown/other
  - 1:High-school
  - 2:University
  - 3:Graduate level

- income - reported annual income of a customer
- occupation - occupation of a customer
  - 0:Unemployed/Unskilled workers
  - 1:Skilled employees
  - 2:Highly qualified/Management positions
- settlement_size - size of a city the customer lives in
  - 0:Small city
  - 1:Mid-size city
  - 2:Big city

EXPLORATORY ANALYSIS

Next important step of data analysis process is Exploratory data analysis (EDA). EDA is used to look at the data and think about the data from many points of view. In order to better understand the pairwise relationship between variables we are using pair plot. This creates a visualization summarizing a important part (in our opinion) of our data in single figure.

Fig. 1. Pair plot of variables.

To better understand the important correlation coefficients we are using correlation matrix. From the heatmap below we see the positive correlation between “age” and “education”. Or in other words older people tend to be more educated in our sample. From matrix we can see married people are more educated. And settlement size gets bigger by better job or occupation.

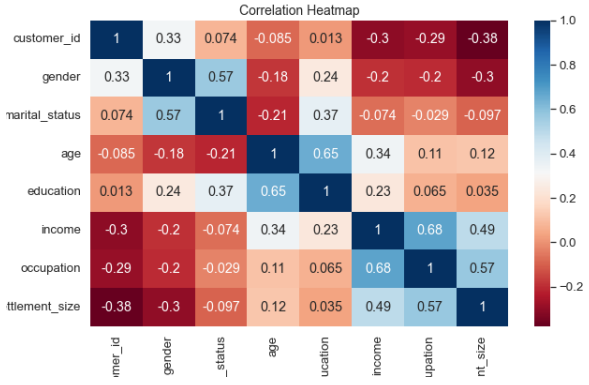


Fig. 2. Correlation Heatmap of variables.

One of the features in our data is “age”. ” Age is an important determinant of expenditure shares for every budget category. Expenditure shares increase with age for food at home, and to some extent, housing and health, but decrease with age for food away from home, apparel, transportation, and other purchases.” To evaluate our customer’s demographic we are plotting distribution plot of age. The average age of the customers is 35.9, the median is 33.0 and mode is 26 years old. Or people aged 26 is more frequent in dataset. According to statista.com most active and largest group of social media users in UK are people aged 18-34. It is 54% of our customers. Number of people in age group 34-64 is 865 with 43% of customers. From the box plot there are 57 customers aged 65 and above which are considered natural outliers. And we will not treat them as we would with artificial ones.

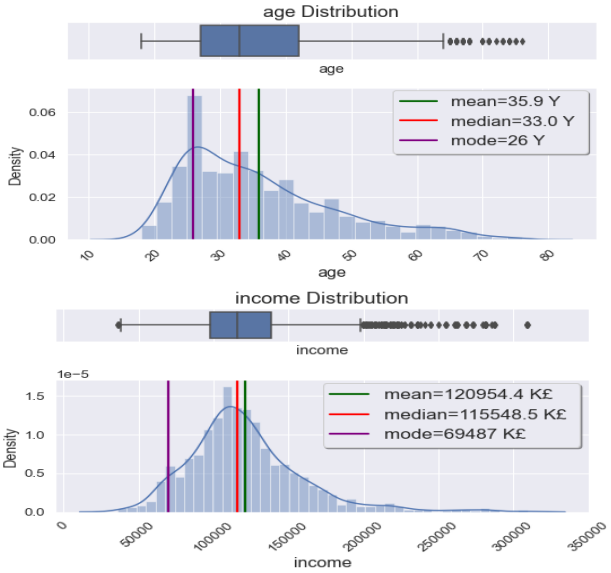


Fig. 3. Data distribution in age and income variables.

Gender plays a huge role in understanding people’s purchasing habits. Gasiorowska [3] reported the differences between male and female buying behaviours. According to the study “the male customers’ buying behaviour is instrumental in nature and prefer minimal level of engagement and finish the purchasing process fast”. On the other hand “females exhibit more impulsive buying behaviour than men, enjoy shopping more than men, and decide to buy products by evaluating various products in different stores rather than buying products from a single store”. Keeping it in mind it is paramount to analyse distribution of age by gender of the customers. 1086 of customers are male out of 2000 with 54.3% of the total population. With average age of 38. Female customer’s average age is slightly younger than the males with average 34 years old.

Fig. 4. Density of the age by gender.

Data distribution in “income” feature reveals that average income of the customers £120954. Most of the customers has annual income £69487. Fewer people have income starting from £200000. Those data shown as outliers in box plot in Figure 3. Average income distribution between males and females are not pronounced. Average annual income for males is £128K, for females £113K. Only 61 males out of 1086 earning from £200K, it is only 5.6%. This number is even small for females with 18 out of 914. It is roughly 2% of females earning £200K and over annually.

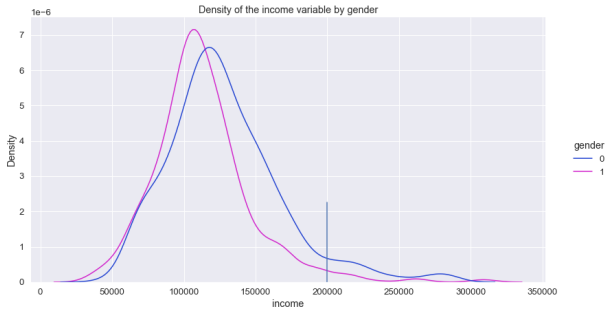


Fig. 5. Density of the income by gender.

All age groups are represented in all three occupation types. Number of unskilled workers 633 (32%), skilled employees 1113 (56%) and highly qualified employees or managers 254 (12%) people of our total sample population. We see that income has positive correlation with age. But it is not as significant as with occupation type. More skilled the employee is more they are getting paid. Average income for unskilled workers is £91K, for skilled workers £125K, for highly qualified employees or managers it is £176K annually.

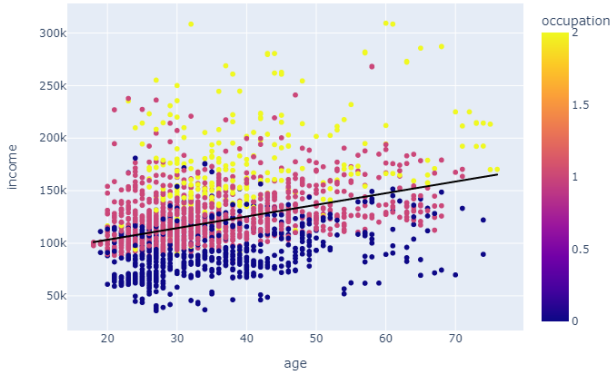


Fig. 6. Income plot by occupation.

We see gender imbalance among highly qualified employees and managing positions. Only 24% of highly qualified employees are females. On the contrary, majority of unskilled workers are filled with female workers with 57%. Male and female gender distribution is fairly evenly distributed in skilled employee group with 55% and 45% respectively.

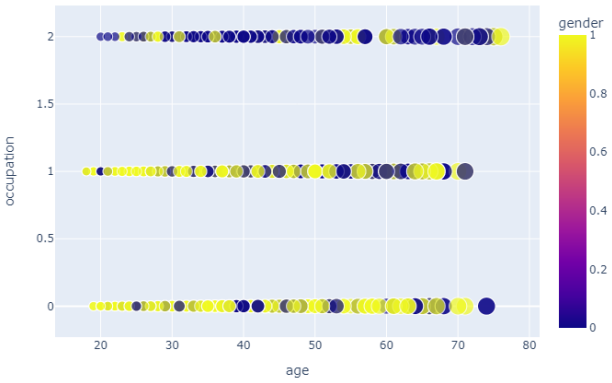


Fig. 7. Occupation plot by gender.

K-MEANS CLUSTERING

K-means clustering is a method for finding clusters and cluster centres in a set of unlabelled data. Hastie et al. [4]. It is important to segment customers to help the marketing team to launch a new product and use different advertising methods for different type of customer groups. Marketing and sales groups use k-means to better identify customers who have similar behaviours and spending patterns [5].

We will try to find and group customers based on common characteristics such as age, gender, income, residing area, marital status and education. Before feeding our data into K-means model we need apply standardization to treat all the features equally. Standardization is achieved by “Standard-Scaler()” function of a python sklearn library. It is basically transforms the values of features to the same numerical values. After this process the data can be fed into the model.

One of the fundamental steps in unsupervised learning using K-means is to determine “K” value. In order to identify number of clusters or optimal “K” value we are using the “Elbow method”. Elbow methods objective is to find the smallest value of “K” that still has a low value of inertia. Jovi et al. [6]. As seen in the Figure 8, the plot formed an elbow and value 3 or 4 would be optimal cluster formation.

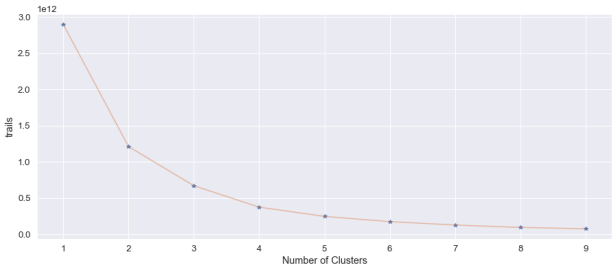


Fig. 8. K-value using ”Elbow method”.

Once the K value identified we need to find values for centroid initialization. Centroid initialization can be done manually or randomly. When centroid values found we can create a plot with clusters or groups of customers with similarities. Our first segmentation with “age” and “income” variables shown

in Figure 9. The algorithm segregated customers mostly by income value rather than age. In this cluster we used K value 4 and we have 4 groups.

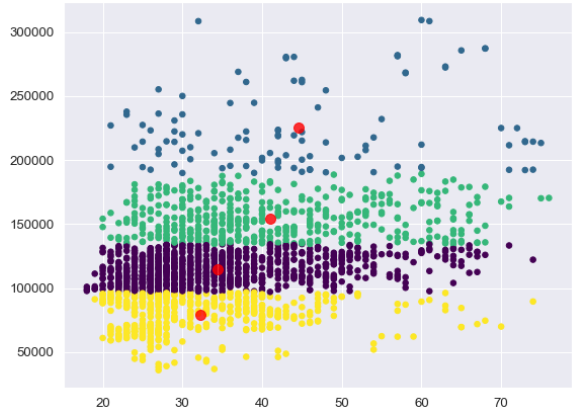


Fig. 9. K-means clustering with ”income” and ”age” attributes.

But we cannot use this as a final result as we did not consider all other variables. In order to segment customers by all features we are using “kmeans()” function. K-means algorithm finds the centroids for each cluster we decided to choose from elbow method. K-means find the best centroids by alternating between assigning data points to clusters based on the current centroids and choosing centroids based on the current assignment of data points to clusters [7]. From below table we see three segment or group of customers with average values for each variable:

TABLE I
CUSTOMER GROUPS BY K-MEANS CLUSTERING.

segment	gender	marital	age	edu.	income	occupat.	settlement
1	0.709	0.619	31.26	0.943	97.5K	0.429	0.163
2	0.128	0.279	34.59	0.766	137.7K	1.188	1.340
3	0.498	0.676	55.72	2.127	155.6K	1.076	1.054

“Group1” is the female dominant group in their early thirties and married. Customers in this segment living in small cities and working in unskilled jobs or unemployed. The education level is lower than other two groups, average annual income is £98K.
“Group2” consists of mostly single men in their mid-thirties. They are skilled employees with average income £138K and living in mid-sized or big cities.
“Group3” has same number of male and female customers who are married and highly educated. The average age for this group is 55 years old. Living in mid-sized city and with annual income £156K in skilled occupation.

Fig. 10. Clustering customers by common characteristics.

CONCLUSION

We used several exploratory data analytics techniques and different types of data visualizations such as pair plots, distribution plots, box plots, correlation heatmap and scatter plots. From the results of K-means clustering model we found 3 different demographic and geographic group of customers. Each of these groups consists of customers from different locations, gender and age. First group consists of mostly from married women who lives in small cities. Second group majority is single men in their mid-30s living in big cities and working in skilled positions. Third group is the oldest group with average age of 55. Gender in this group evenly distributed by married women and men. This group also is highly educated group. Considering these each of them can be useful for different type of product advertisements.

Customer’s occupation plays important role in peoples purchasing habits. According to et al. [8] customers have different consumption customs in different industries. Also, Gasiorowska [3] reported that men and women differ in terms of shopping behaviour. Considering these companies should establish different marketing strategies for different targeted groups.

REFERENCES

[1] Kristen Baker. (2022). *Customer Segmentation: How to Effectively Segment Users and Clients* [Online]. Available: https://blog.hubspot.com/service/customer-segmentation
[2] Tushar Kansal; Suraj Bahuguna; Vishal Singh; Tanupriya Choudhury, et al., ”Customer Segmentation using K-means Clustering”, *International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pp. 135-136, 2018.
[3] Agata Gasiorowska, ”Gender as a moderator of temperamental causes of impulse buying tendency”, *Journal of Customer Behaviour*, vol.10(2), pp. 119-142, August 2011.
[4] Hastie, T., Tibshirani, R., Friedman, J., ”Prototype Methods and Nearest-Neighbors” in *The Elements of Statistical Learning*, Springer, New York, NY., 2009, pp. 459–483.
[5] EMC Education Services, *Data Science and Big Data Analytics: Discovering, Analyzing, Visualizing and Presenting Data*. Wiley, January 2015.
[6] Jovi D’Silva1, Dr. Uzzal Sharma, ”Unsupervised Automatic Text Summarization of Konkani Texts using K-means with Elbow Method”, *International Journal of Engineering Research and Technology*, Vol.13, pp 2380-2384.

[7] Chris Piech. Based on a handout by Andrew Ng. (2013). *K Means* [Online]. Available: https://stanford.edu/~cpiech/cs221/handouts/kmeans.html#:~:text=K%2DMeans%20finds%20the%20best,1%3A%20K%2Dmeans%20algorithm.

[8] Jing Wu, Zheng Lin, "Unsupervised Automatic Text Summarization of Konkani Texts using K-means with Elbow Method" in *7th international conference on Electronic commerce*, New York, 2005.

Appendix 1

December 15, 2022

Python codes used for data analysis and modelling

1.0 Importing necessary libraries and initial data exploring

[ ]:
```
from matplotlib.pyplot import bar_label
import seaborn as sns;
from scipy import stats
import glob
import random
import datetime
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
from plotly.offline import init_notebook_mode, iplot
init_notebook_mode(connected=True)
import matplotlib.cm as cm
import os
import pickle
# from  datasist.structdata import detect_outliers
from tqdm import tqdm
# Core
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns; sns.set(rc={'figure.figsize':[7,7]},font_scale=1.2)
from datetime import date,timedelta
from mpl_toolkits.mplot3d import Axes3D
from mpl_toolkits.mplot3d import axes3d
import sklearn
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
# Pre Processing
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
# Regressors
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor
from sklearn.svm import SVR
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LogisticRegression
# Error Metrics
from sklearn.metrics import r2_score #r2 square
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import plot_confusion_matrix ,classification_report
from sklearn.metrics import accuracy_score,␣
 ↪precision_score,recall_score,f1_score


#classefication
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import SGDClassifier #stacstic gradient descent␣
 ↪clasifeier
import graphviz
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
#crossvalidation
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import LeaveOneOut
from sklearn.metrics import plot_confusion_matrix
#clustring
from sklearn.cluster import KMeans
from sklearn.cluster import AgglomerativeClustering
#hyper parameter tunning
from sklearn.model_selection import GridSearchCV
#pca
from sklearn.decomposition import PCA
#clustring
from sklearn.cluster import KMeans
from warnings import filterwarnings
filterwarnings("ignore")
```

[ ]:
```
seed = 42
np.random.seed =seed
```

[ ]:
```
df=pd.read_csv('customer_data.csv')
df.head()
```

[ ]:
```
          ID  Sex  Marital status  Age  Education  Income  Occupation  \
0  100000001    0               0   67          2  124670           1
1  100000002    1               1   22          1  150773           1
2  100000003    0               0   49          1   89210           0
3  100000004    0               0   45          1  171565           1
4  100000005    0               0   53          1  149031           1

   Settlement size
0                2
1                2
2                0
3                1
4                1
```

[ ]:
```
df.shape
```

[ ]:
```
(2000, 8)
```

[ ]:
```
#set colimn name to lower
df = df.rename(columns=str.lower)
```

[ ]:
```
for col in df.columns :
    if df[col].dtype == 'object':
       df[col] = df[col].apply(lowerCase)
df
```

[ ]:
```
             id  sex  marital status  age  education  income  occupation  \
0     100000001    0               0   67          2  124670           1
1     100000002    1               1   22          1  150773           1
2     100000003    0               0   49          1   89210           0
3     100000004    0               0   45          1  171565           1
4     100000005    0               0   53          1  149031           1
…           …  …             …  …       …     …          …
1995  100001996    1               0   47          1  123525           0
1996  100001997    1               1   27          1  117744           1
1997  100001998    0               0   31          0   86400           0
1998  100001999    1               1   24          1   97968           0
1999  100002000    0               0   25          0   68416           0

      settlement size
0                   2
1                   2
2                   0
3                   1
4                   1
…                 …
1995                0
```

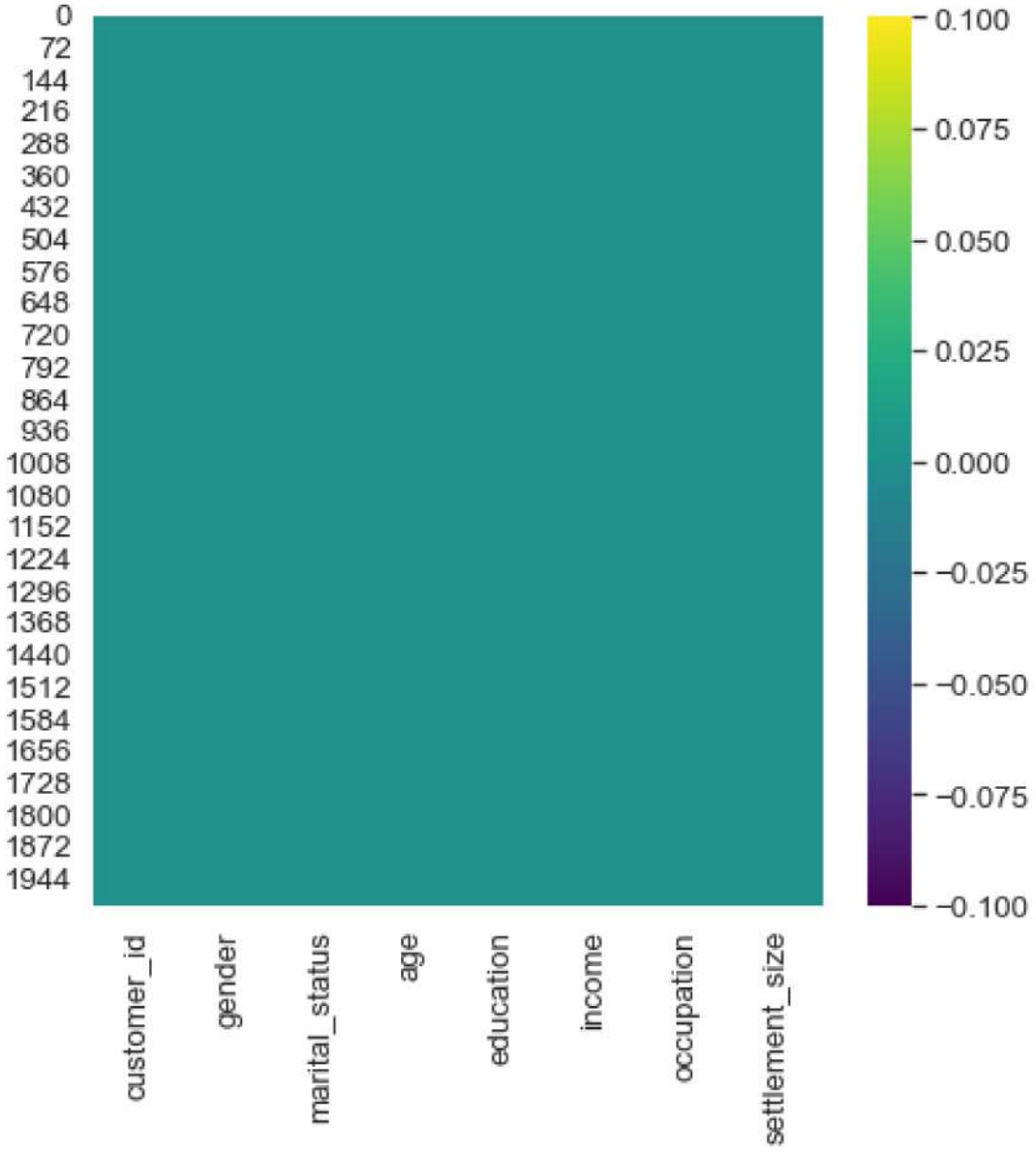
```
1996                0
1997                0
1998                0
1999                0

[2000 rows x 8 columns]
```

[ ]:
```
#rename columns
df.rename(columns = {"id": "customer_id",
                     "sex": "gender",
                    'marital status' :'marital_status',
                     'settlement size':'settlement_size'}, inplace = True)
```

2.0 Exploratory analysis

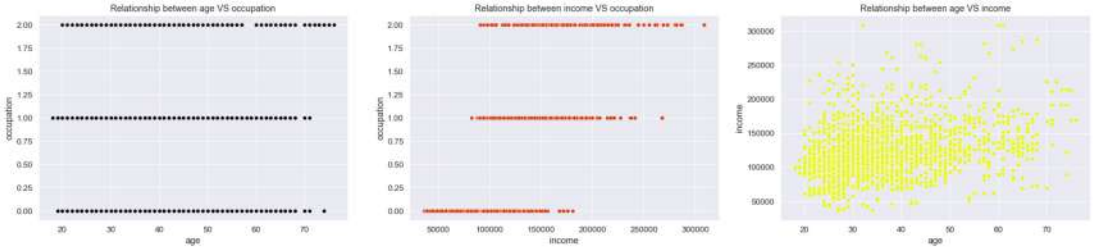
[ ]:
```
cols =df.columns
sns.heatmap(df[cols].isnull(), cmap='viridis')
```

[ ]:
```
<AxesSubplot:>
```

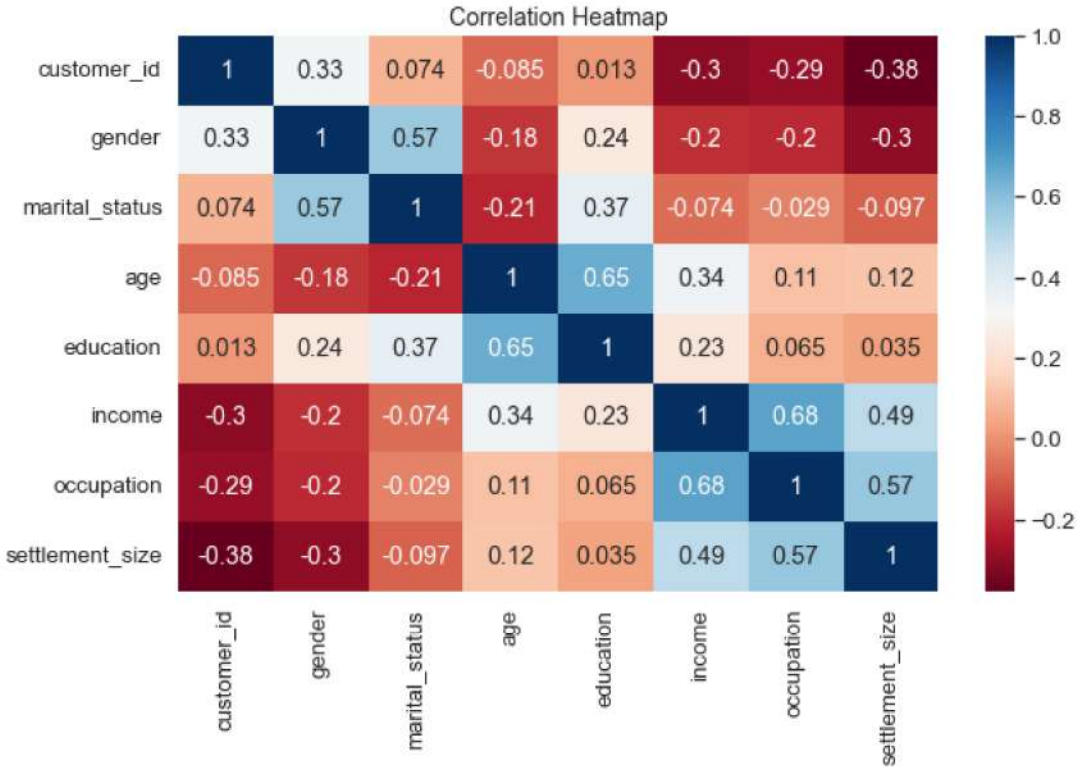


[ ]:
```
fig, axes = plt.subplots(1, 3, figsize = (30, 6))
axes = axes.flatten()


sns.scatterplot(ax = axes[0], x = "age", y = "occupation", data = df, color =␣
↪"#000000"
                ).set(title = "Relationship between age VS occupation");
sns.scatterplot(ax = axes[1], x = "income", y = "occupation", data = df,
```

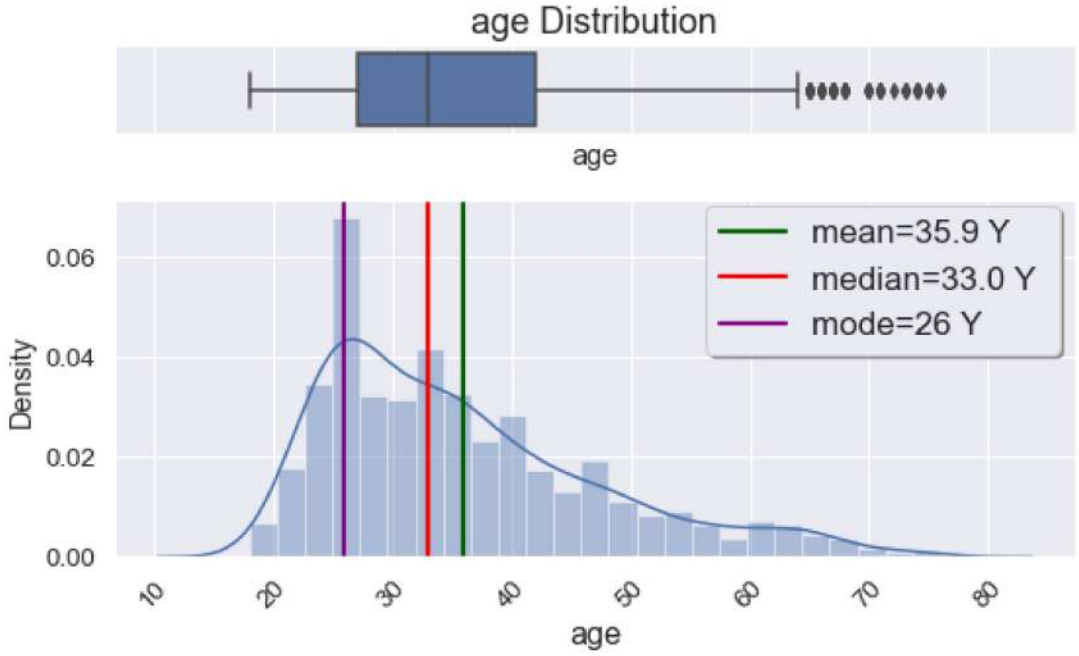
```
                color = "#D63913").set(title = "Relationship between income VS␣
 ↪occupation");
sns.scatterplot(ax = axes[2], x = "age", y = "income", data = df,
                color = "#E9F709").set(title = "Relationship between age VS␣
 ↪income");
plt.savefig('figure1.png')
```



[ ]:
```
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True,cmap='RdBu')
plt.title('Correlation Heatmap',fontsize=14)
plt.yticks(rotation =0)
plt.savefig('figure2.png')
plt.show()
```



[ ]:
```
def numerical_plotting(df, col, title, symb):
    fig, ax = plt.subplots(2, 1, sharex=True,␣
 ↪figsize=(8,5),gridspec_kw={"height_ratios": (.2, .8)})
    ax[0].set_title(title,fontsize=18)
    sns.boxplot(x=col, data=df, ax=ax[0])
    ax[0].set(yticks=[])
    sns.distplot(df[col],kde=True)
    plt.xticks(rotation=45)
    ax[1].set_xlabel(col, fontsize=16)
    plt.axvline(df[col].mean(), color='darkgreen', linewidth=2.2, label='mean='␣
 ↪+ str(np.round(df[col].mean(),1)) + symb)
    plt.axvline(df[col].median(), color='red', linewidth=2.2, label='median='+␣
 ↪str(np.round(df[col].median(),1)) + symb)
    plt.axvline(df[col].mode()[0], color='purple', linewidth=2.2,␣
 ↪label='mode='+ str(df[col].mode()[0]) + symb)
    plt.legend(bbox_to_anchor=(1, 1.03), ncol=1, fontsize=17, fancybox=True,␣
 ↪shadow=True, frameon=True)
    plt.tight_layout()
    plt.show()
numerical_plotting(df,'age','age Distribution',' Y ')
```

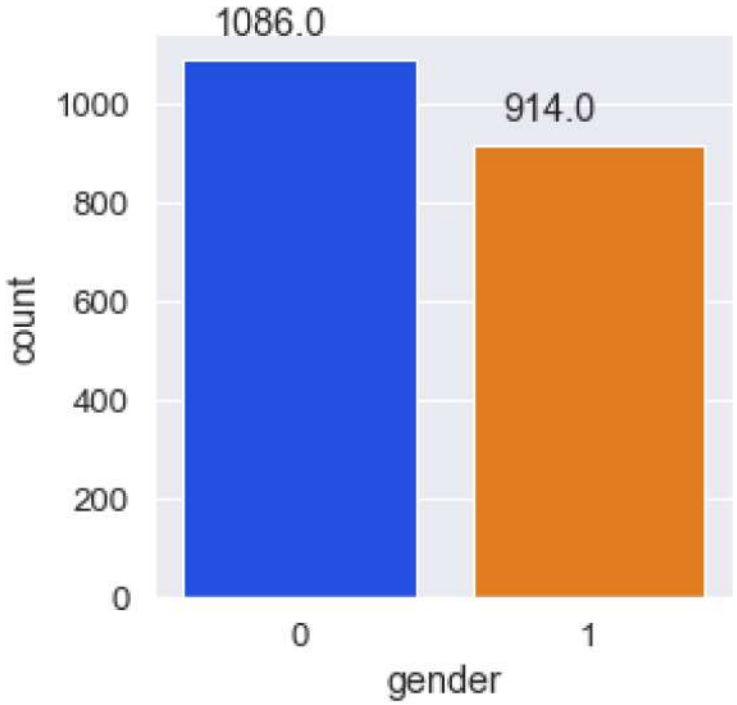


[ ]:
```
df.describe()[['age']].T
```

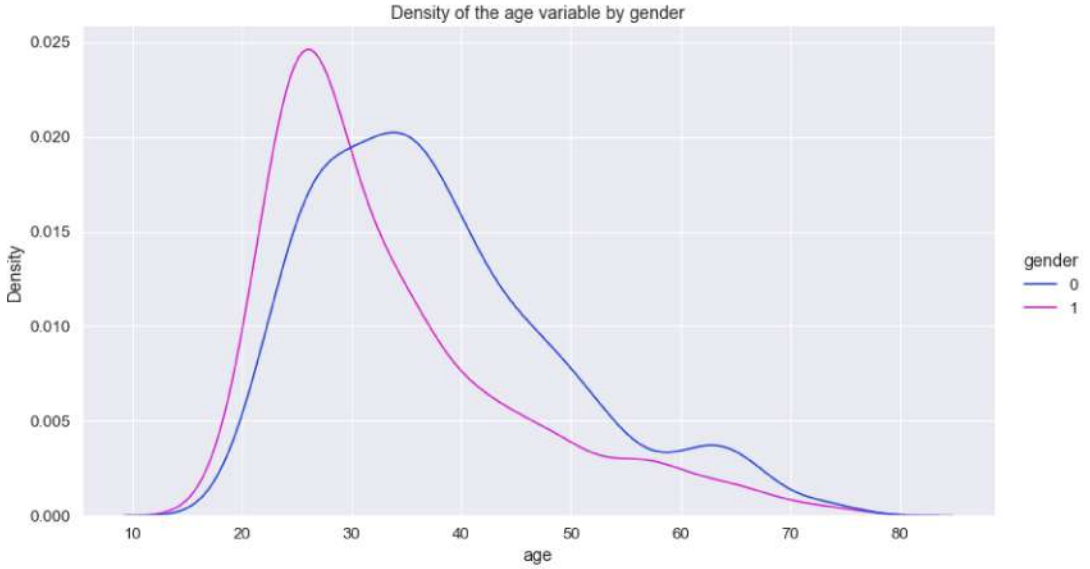
[ ]:

	count	mean	std	min	25%	50%	75%	max
age	2000.0	35.909	11.719402	18.0	27.0	33.0	42.0	76.0

[ ]:
```
#explore data
plt.figure(figsize=(4,4))
ax = sns.countplot(x=df["gender"], data=df, palette='bright')
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.
 ↪get_height()+50))

plt.show()
```



[ ]:
```
#distplot
sns.displot(data = df, x = "age", hue = "gender", kind = "kde", height = 6.5, ␣
 ↪aspect = 1.8, clip = (0, None),
    palette = ["#2747D3", "#D327CD"]).set(title = "Density of the age variable␣
 ↪by gender");
plt.savefig('figure4.png')
```

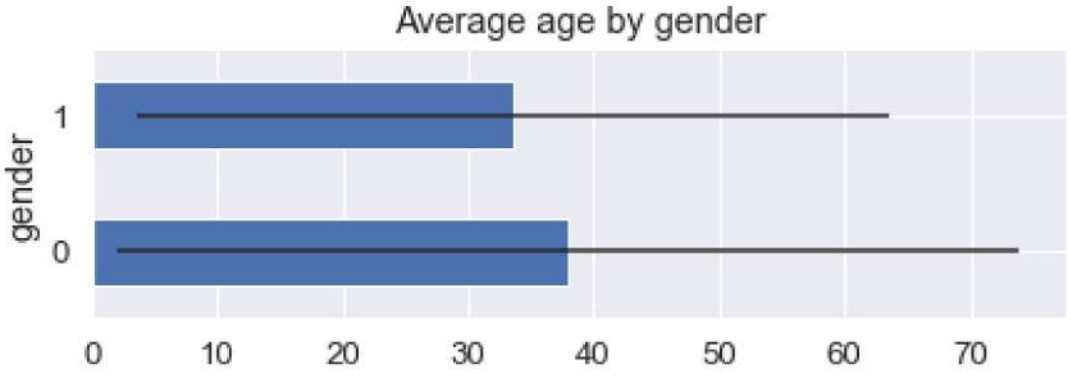


[ ]:
```
df_gender = df.groupby("gender").agg([np.mean, np.median, np.std])
sex = df_gender['age']
sex.head()
```

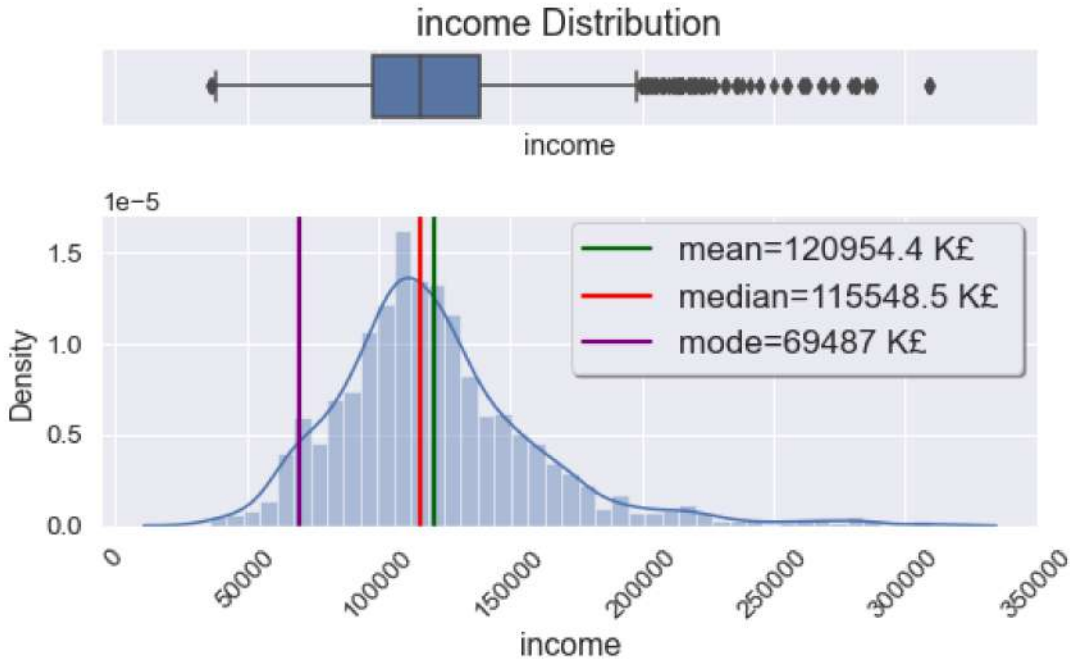
[ ]:
```
              mean  median        std
gender
0        37.874770    36.0  11.549013
1        33.573304    30.0  11.495586
```

[ ]:
```
sex.plot(kind = "barh", y = "mean", legend = False,
            title = "Average age by gender", xerr = "median", figsize=(7, 2))
```

[ ]:
```
<AxesSubplot:title={'center':'Average age by gender'}, ylabel='gender'>
```



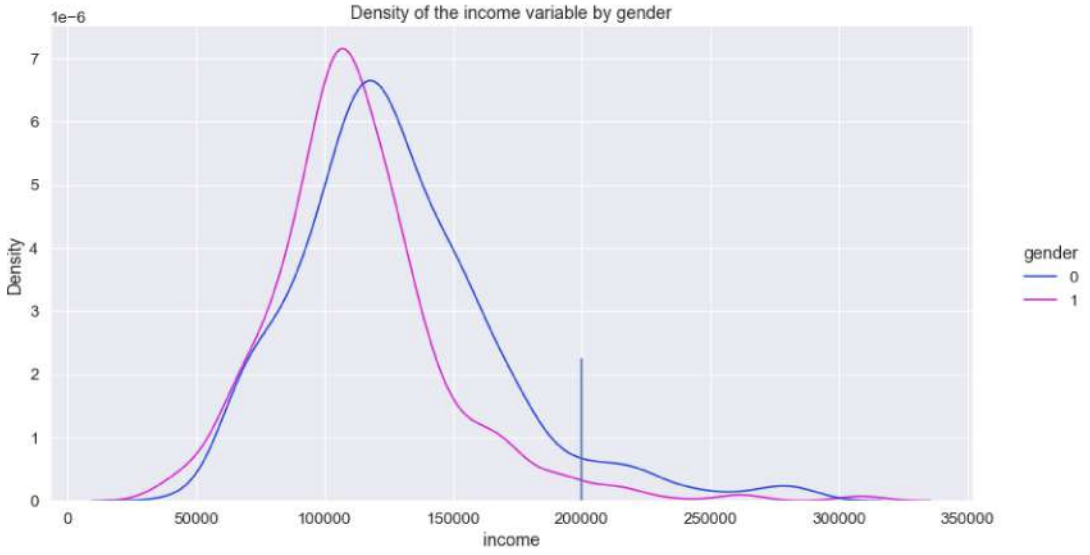
[ ]:
```
numerical_plotting(df,'income','income Distribution',' K£ ')
```



[ ]:
```
df.describe()[['income']].T
```

[ ]:
```
          count         mean            std      min       25%       50%  \
income   2000.0   120954.419   38108.824679  35832.0  97663.25  115548.5

                75%       max
income   138072.25  309364.0
```

[ ]:
```
sns.displot(data = df, x = "income", hue = "gender", kind = "kde", height = 6.
 ↪5,  aspect = 1.8, clip = (0, None),
    palette = ["#2747D3", "#D327CD"]).set(title = "Density of the income␣
 ↪variable by gender")
plt.axvline(200000, 0,0.3);
plt.savefig('figure6.png')
```



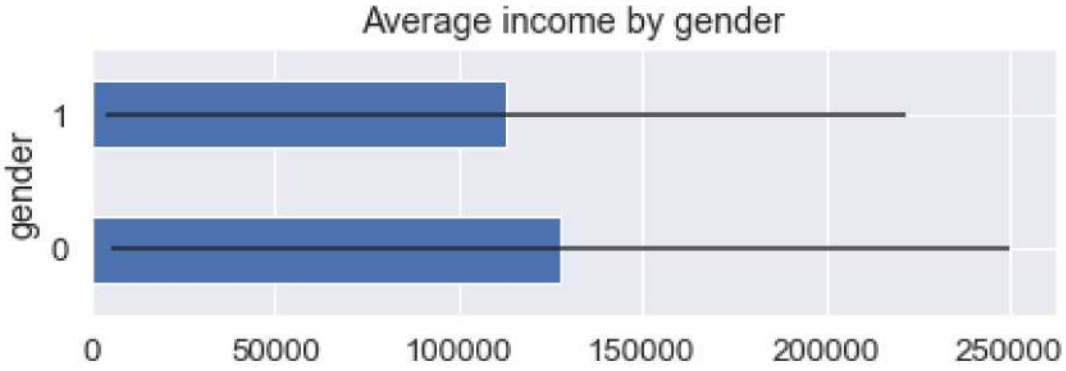
[ ]:
```
df_income = df.groupby("gender").agg([np.mean, np.median, np.std])
annual_income = df_income['income']
annual_income.head()
```

[ ]:

gender	mean	median	std
0	127775.225599	122352.5	39821.354629
1	112850.047046	108666.0	34266.333929

[ ]:
```
annual_income.plot(kind = "barh", y = "mean", legend = False,
            title = "Average income by gender", xerr = "median", figsize=(7, 2))
```

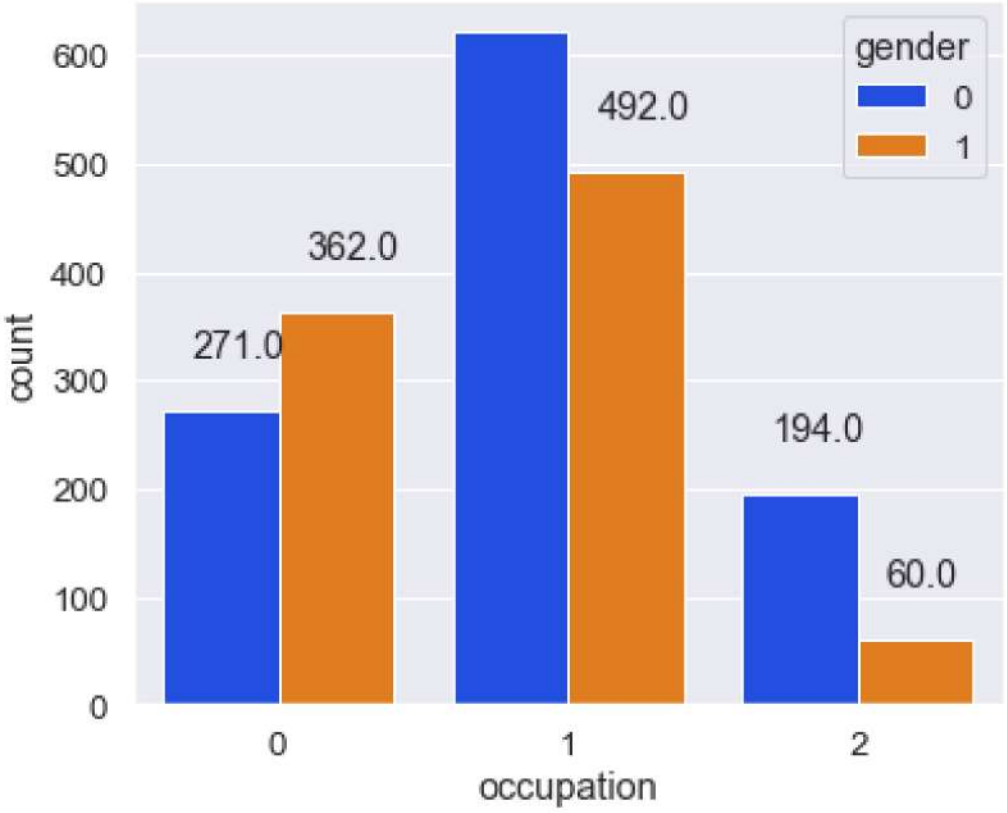
[ ]: <AxesSubplot:title={'center':'Average income by gender'}, ylabel='gender'>



[ ]:
```
import os

if not os.path.exists("images"):
    os.mkdir("images")
#plotly
fig = px.scatter(df, x = "age", y = "income", color = "occupation" ,opacity =␣
 ↪1, trendline="ols", trendline_color_override="black")
fig.write_image("images/figure7.png")
```

[ ]:
```
fig = px.scatter(df, x="age", y="occupation", color="gender",
                 size='age', hover_data=['occupation'])
fig.write_image("images/figure8.png")
fig.show()
```

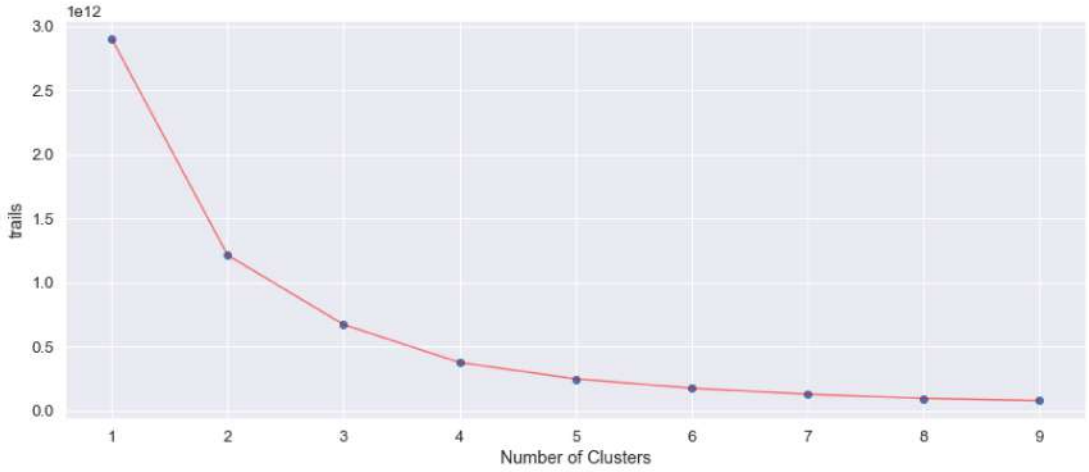
[ ]:
```
#explore data
plt.figure(figsize=(6,5))
ax = sns.countplot(x=df["occupation"], hue = 'gender', data=df,␣
 ↪palette='bright')
for p in ax.patches:
        ax.annotate('{:.1f}'.format(p.get_height()), (p.get_x()+0.1, p.
 ↪get_height()+50))
plt.show()
```



3.0 K-means clustering

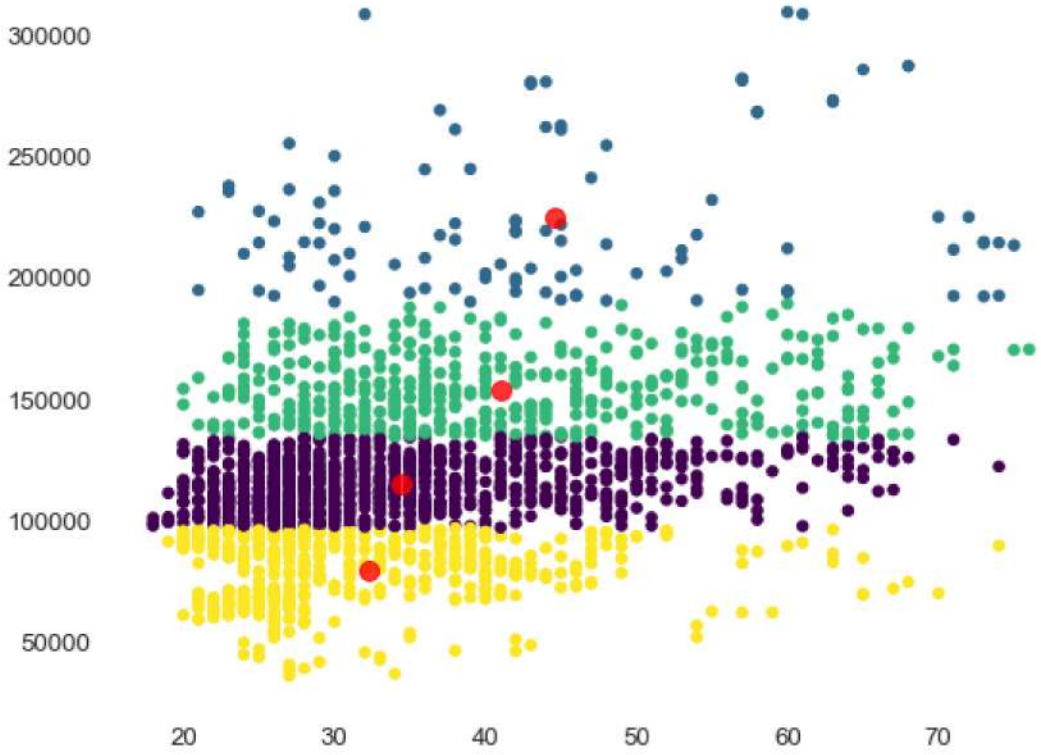
[ ]:
```
# 1 - clustring between age and income
df_one = df[['age','income']]
data=[]
for n in range(1,10):
    kmeans = (KMeans(n_clusters = n ,init='k-means++', n_init = 10␣
 ↪,max_iter=400,
                     tol=0.0001,  random_state= 45 ) )
    kmeans.fit(df_one)
    data.append(kmeans.inertia_)
```

[ ]:
```
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 10) , data , 'o')
plt.plot(np.arange(1 , 10) , data , '-' , alpha = 0.5, color = "red")
#grid
plt.rcParams['axes.facecolor'] = 'white'
#axis
plt.xlabel('Number of Clusters') , plt.ylabel('trails')
plt.savefig('numberofClusters1.png', facecolor="White")
plt.show()
```



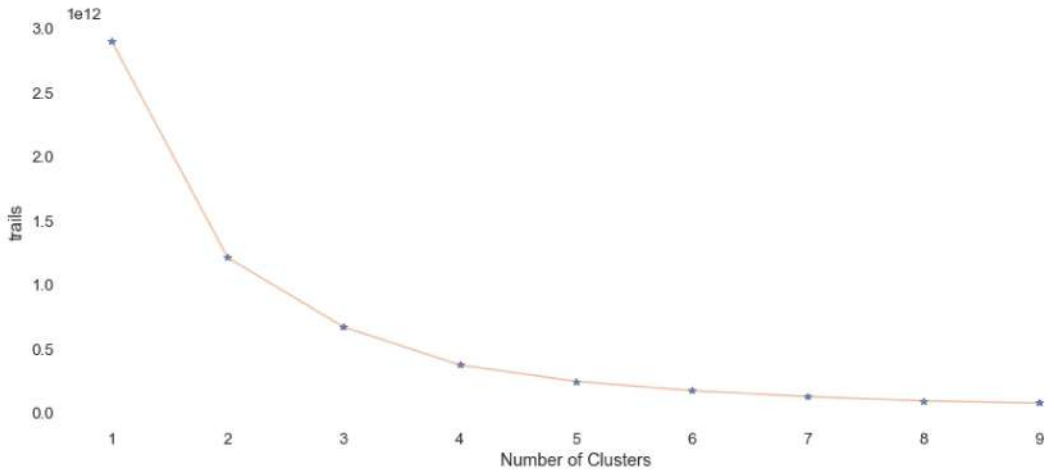
[ ]:
```
#in this case i will use k=4
kmeans = KMeans(n_clusters = 4, random_state = 45)
k_fit = kmeans.fit(df_one)
clusters = k_fit.labels_

plt.figure(figsize = [9, 7], clear = False)
```

```
clusters = k_fit.labels_
centers = k_fit.cluster_centers_
plt.scatter(df_one['age'],df_one['income'],c = clusters,s = 30,cmap = "viridis")
plt.scatter(centers[:, 0],centers[:, 1],c = "red",s = 100,alpha = 0.8);
plt.savefig('figure9.png')
```



[ ]:
```
df_three = df[['age','occupation','income']]
data=[]
for n in range(1,10):
    kmeans = (KMeans(n_clusters = n ,init='k-means++', n_init = 10␣
 ↪,max_iter=400, tol=0.0001,  random_state= 45 ) )
    kmeans.fit(df_three)
    data.append(kmeans.inertia_)
```

[ ]:
```
plt.figure(1 , figsize = (15 ,6))
plt.plot(np.arange(1 , 10) , data , '*')
plt.plot(np.arange(1 , 10) , data , '-' , alpha = 0.5)
plt.xlabel('Number of Clusters') , plt.ylabel('trails')
plt.savefig('figure10.png')
plt.show()
```

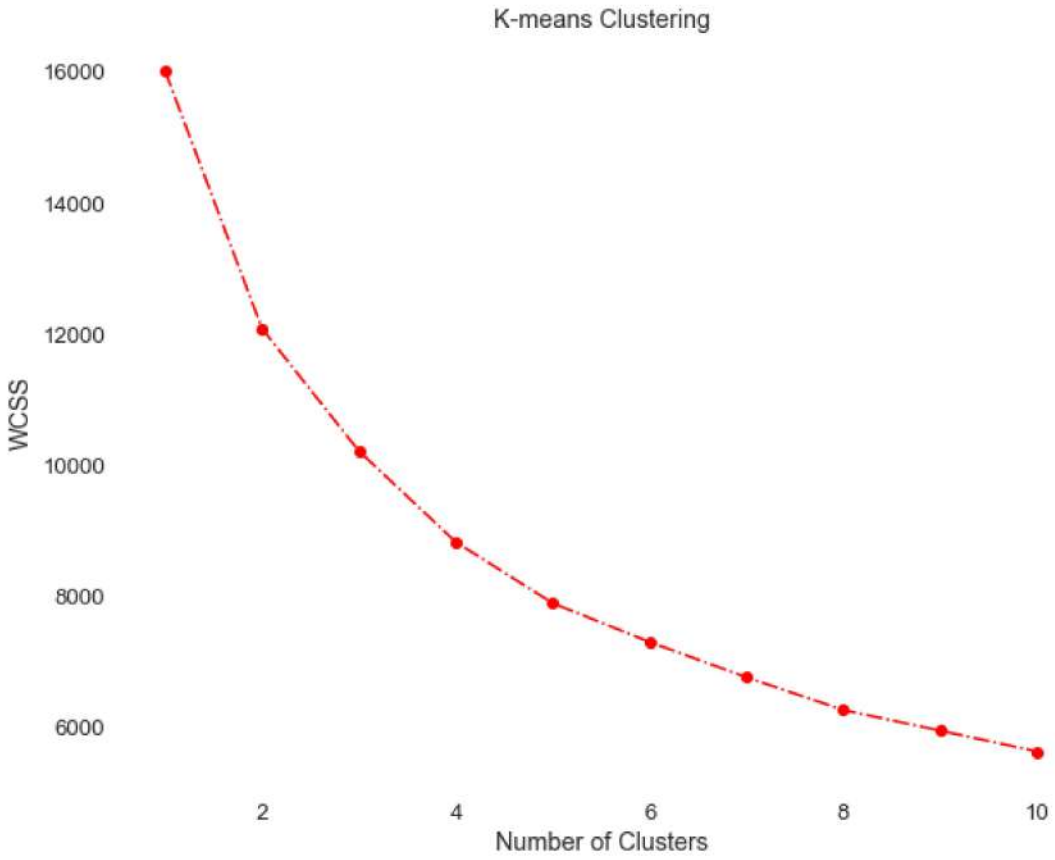


[ ]:
```
kmeans = KMeans(n_clusters = 6, random_state = 45)
k_fit = kmeans.fit(df_three)
clusters = k_fit.labels_
```

[ ]:
```
# Standardizing data
scaler = StandardScaler()
df_std = scaler.fit_transform(df)
```

[ ]:
```
# Perform K-means clustering
wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters = i, init = 'k-means++', random_state = 42)
    kmeans.fit(df_std)
    wcss.append(kmeans.inertia_)
```

[ ]:
```
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()
```



[ ]:
```
# We run K-means with a fixed number of clusters. In our case 4.
kmeans = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
```

[ ]:
```
kmeans.fit(df_std)
```

[ ]:
```
KMeans(n_clusters=4, random_state=42)
```

[ ]:
```
# Creating new data frame
kmeans_segm= df_std.copy()
kmeans_segm = pd.DataFrame(data = df_std,columns = df.columns)
kmeans_segm['Segment K-means'] = kmeans.labels_
```

[ ]:
```
#mean values
analysis_segm = kmeans_segm.groupby(['Segment K-means']).mean()
analysis_segm.head()
```

[ ]:
```
                customer_id    gender  marital_status       age  education  \
Segment K-means
0                  0.216385  0.841696        1.004079 -0.589431   0.049608
```

```
1                  -0.593534 -0.857049       -0.635421 -0.023564  -0.501406
2                   0.697055 -0.107716       -0.907103 -0.104064  -0.493232
3                  -0.236585  0.056580        0.369972  1.698960   1.814329

                    income  occupation  settlement_size
Segment K-means
0                -0.403075   -0.282195        -0.394837
1                 0.498439    0.688410         0.798516
2                -0.696787   -0.869527        -0.836279
3                 0.928253    0.447657         0.421604
```

[ ]:
```
analysis_segm.rename({0:'Group2',
                      1:'Group3',
                      2:'Group1',
                      3:'Group0'})
```

[ ]:
```
                 customer_id    gender  marital_status       age  education  \
Segment K-means
Group2              0.216385  0.841696        1.004079 -0.589431   0.049608
Group3             -0.593534 -0.857049       -0.635421 -0.023564  -0.501406
Group1              0.697055 -0.107716       -0.907103 -0.104064  -0.493232
Group0             -0.236585  0.056580        0.369972  1.698960   1.814329

                    income  occupation  settlement_size
Segment K-means
Group2           -0.403075   -0.282195        -0.394837
Group3            0.498439    0.688410         0.798516
Group1           -0.696787   -0.869527        -0.836279
Group0            0.928253    0.447657         0.421604
```

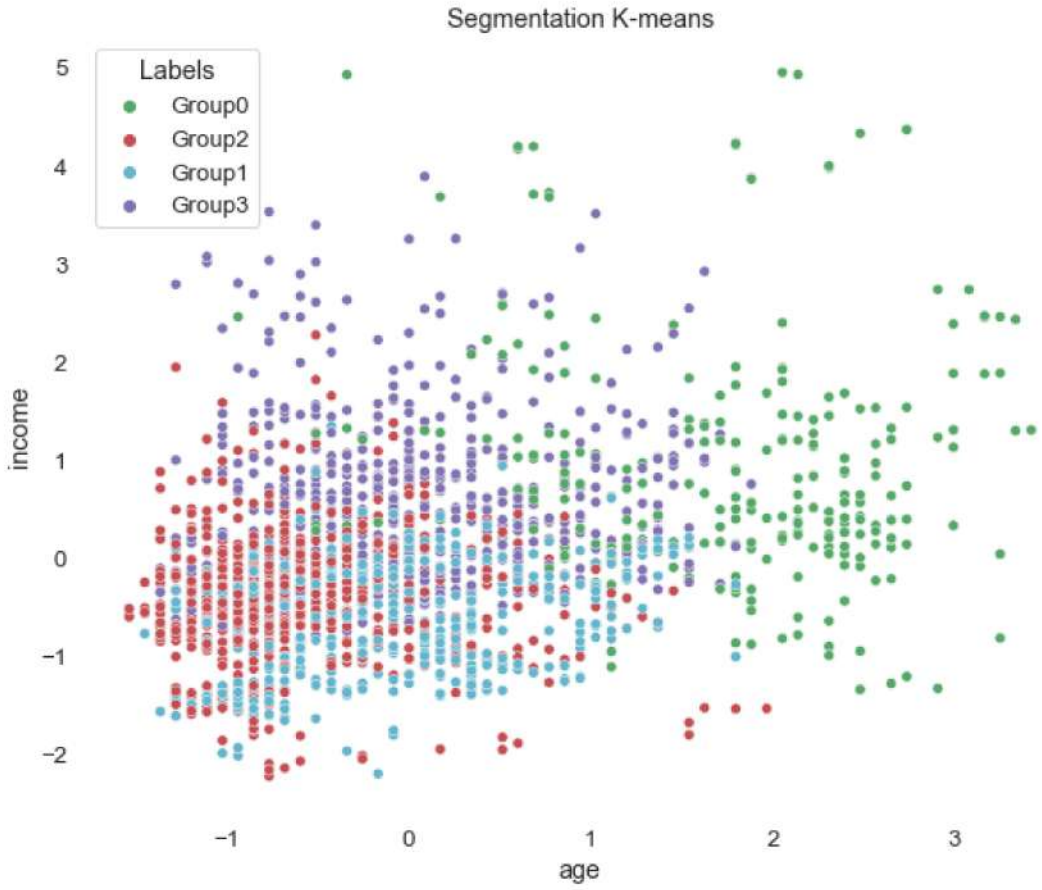
[ ]:
```
# Add segment labels
kmeans_segm['Labels'] = kmeans_segm['Segment K-means'].map({0:'Group2',
                                                            1:'Group3',
                                                            2:'Group1',
                                                            3:'Group0'})
```

[ ]:
```
kmeans_segm.head()
```

[ ]:
```
   customer_id    gender  marital_status       age  education    income  \
0    -1.731185 -0.917399       -0.993024  2.653614   1.604323  0.097524
1    -1.729453  1.090038        1.007025 -1.187132  -0.063372  0.782654
2    -1.727721 -0.917399       -0.993024  1.117316  -0.063372 -0.833202
3    -1.725989 -0.917399       -0.993024  0.775916  -0.063372  1.328386
4    -1.724257 -0.917399       -0.993024  1.458716  -0.063372  0.736932

   occupation  settlement_size  Segment K-means  Labels
0    0.296823         1.552326                3  Group0
```

```
1    0.296823          1.552326                0  Group2
2   -1.269525         -0.909730                2  Group1
3    0.296823          0.321298                1  Group3
4    0.296823          0.321298                1  Group3
```

[ ]:
```
# plot results
x_axis = kmeans_segm['age']
y_axis = kmeans_segm['income']
plt.figure(figsize = (10, 8))
sns.scatterplot(x_axis, y_axis, hue = kmeans_segm['Labels'], palette = ['g',␣
↪'r', 'c', 'm'])
plt.title('Segmentation K-means')
plt.show()
```



[ ]:
```
# PCA
pca_comp = PCA()
```

[ ]:
```
# Fit PCA
pca_comp.fit(df_std)
```

[ ]:
```
PCA()
```

[ ]:
```
# variance
pca_comp.explained_variance_ratio_
```

[ ]:
```
array([0.34103573, 0.23178599, 0.16650585, 0.09955452, 0.06169548,
       0.04785186, 0.03407515, 0.01749541])
```

[ ]:
```
# We choosing 3 components. 3 eems the right choice
pca_comp = PCA(n_components = 3)
```

[ ]:
```
#Fit the model
pca_comp.fit(df_std)
```

[ ]:
```
PCA(n_components=3)
```

[ ]:
```
pca_comp.components_
```

[ ]:
```
array([[-0.34541048, -0.32858553, -0.18726934,  0.27028302,  0.10451468,
         0.48384405,  0.46168136,  0.45433728],
       [ 0.10723681,  0.4213196 ,  0.47208337,  0.35525956,  0.65278586,
         0.17628427,  0.06136181, -0.0307768 ],
       [ 0.14352194, -0.31795888, -0.4854334 ,  0.61344196,  0.25225754,
        -0.12360121, -0.34456626, -0.26212582]])
```

[ ]:
```
pca_comp_df = pd.DataFrame(data = pca_comp.components_,
                           columns = df.columns,
                           index = ['Component1', 'Component2', 'Component3'])
pca_comp_df
```

[ ]:
```
            customer_id    gender  marital_status       age  education  \
Component1    -0.345410 -0.328586       -0.187269  0.270283   0.104515
Component2     0.107237  0.421320        0.472083  0.355260   0.652786
Component3     0.143522 -0.317959       -0.485433  0.613442   0.252258

               income  occupation  settlement_size
Component1  0.483844    0.461681         0.454337
Component2  0.176284    0.061362        -0.030777
Component3 -0.123601   -0.344566        -0.262126
```

[ ]:
```
pca_comp.transform(df_std)
```

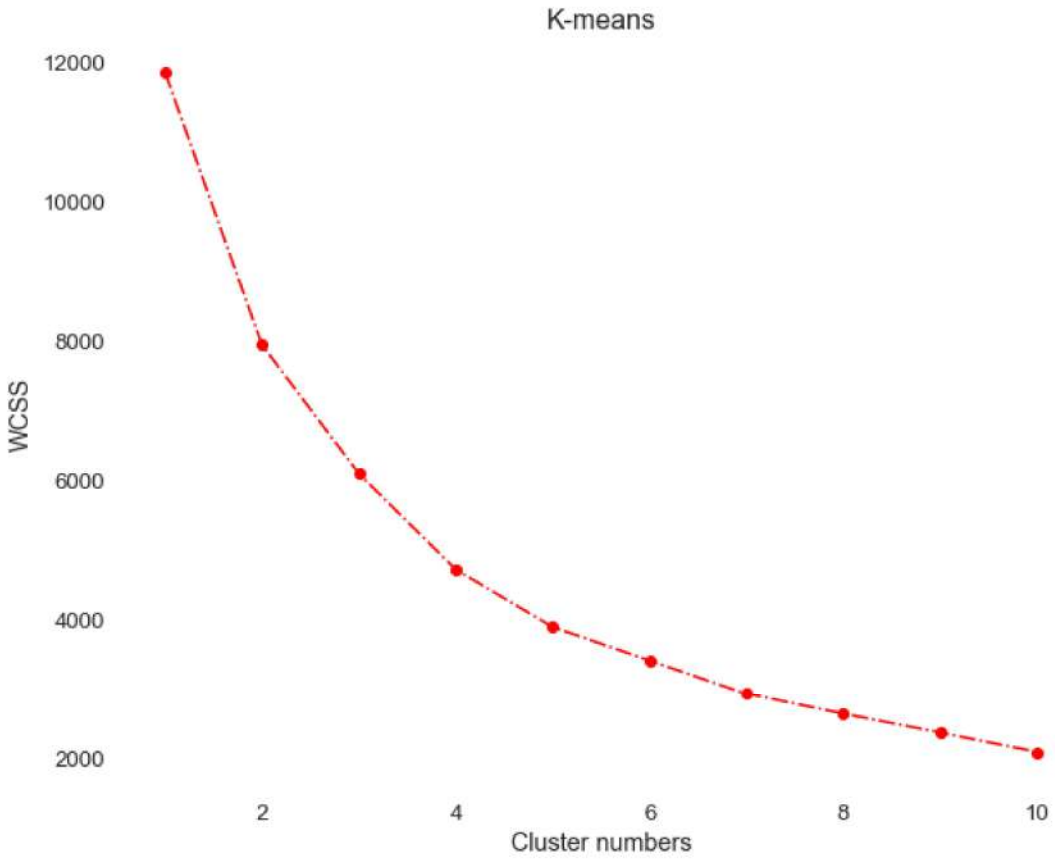
[ ]:
```
array([[ 2.85978214,  0.93667597,  2.03658632],
       [ 0.94413038,  0.39449213, -2.43378502],
       [-0.02303213, -0.8817974 ,  1.97408269],
```

```
…,
[-1.84179778, -2.15868138,  1.1160118 ],
[-2.71683211,  0.56139001, -0.4762533 ],
[-2.2097949 , -2.42344957,  0.86070907]])
```

[ ]:
```
scores_pca = pca_comp.transform(df_std)
```

[ ]:
```
#PCA Clustering
wcss = []
for i in range(1,11):
    pca_kmeans_df = KMeans(n_clusters = i, init = 'k-means++', random_state =␣
 ↪42)
    pca_kmeans_df.fit(scores_pca)
    wcss.append(pca_kmeans_df.inertia_)
```

[ ]:
```
# Plot
plt.figure(figsize = (10,8))
plt.plot(range(1, 11), wcss, marker = 'o', linestyle = '-.',color='red')
plt.xlabel('Cluster numbers')
plt.ylabel('WCSS')
plt.title('K-means',fontsize = 16)
plt.show()
```



[ ]:
```
pca_kmeans_df = KMeans(n_clusters = 3, init = 'k-means++', random_state = 42)
```

[ ]:
```
#fit data
pca_kmeans_df.fit(scores_pca)
```

[ ]:
```
KMeans(n_clusters=3, random_state=42)
```

[ ]:
```
kmeansSegmentedPca = pd.concat([df.reset_index(drop = True), pd.
 ↪DataFrame(scores_pca)], axis = 1)
kmeansSegmentedPca.columns.values[-3: ] = ['Component1', 'Component2',␣
 ↪'Component3']
kmeansSegmentedPca['Segment K-means PCA'] = pca_kmeans_df.labels_
```

[ ]:
```
kmeansSegmentedPca_freq = kmeansSegmentedPca.groupby(['Segment K-means PCA']).
 ↪mean()
kmeansSegmentedPca_freq
```

[ ]:
```
                       customer_id    gender  marital_status        age  \
Segment K-means PCA
0                     1.000006e+08  0.128571        0.279221  34.596104
1                     1.000009e+08  0.498182        0.676364  55.720000
2                     1.000013e+08  0.709948        0.619895  31.262827

                       education         income  occupation  settlement_size  \
Segment K-means PCA
0                       0.766234  137723.603896    1.188312         1.340260
1                       2.127273  155251.661818    1.076364         1.054545
2                       0.943455   97557.545550    0.429319         0.163351

                       Component1  Component2  Component3
Segment K-means PCA
0                        1.253369   -0.792837   -0.302295
1                        1.436546    2.138671    0.904803
2                       -1.424235    0.023403   -0.016810
```

[ ]:
```
# Calculate size of clusters
kmeansSegmentedPca_freq['N Obs'] = kmeansSegmentedPca[['Segment K-means␣
 ↪PCA','gender']].groupby(['Segment K-means PCA']).count()
kmeansSegmentedPca_freq['Prop Obs'] = kmeansSegmentedPca_freq['N Obs'] /␣
 ↪kmeansSegmentedPca_freq['N Obs'].sum()
kmeansSegmentedPca_freq = kmeansSegmentedPca_freq.rename({0:'Group2',
                                                          1:'Group3',
                                                          2:'Group1',
                                                          3:'Group0'})
```

```
kmeansSegmentedPca_freq
```

[ ]:

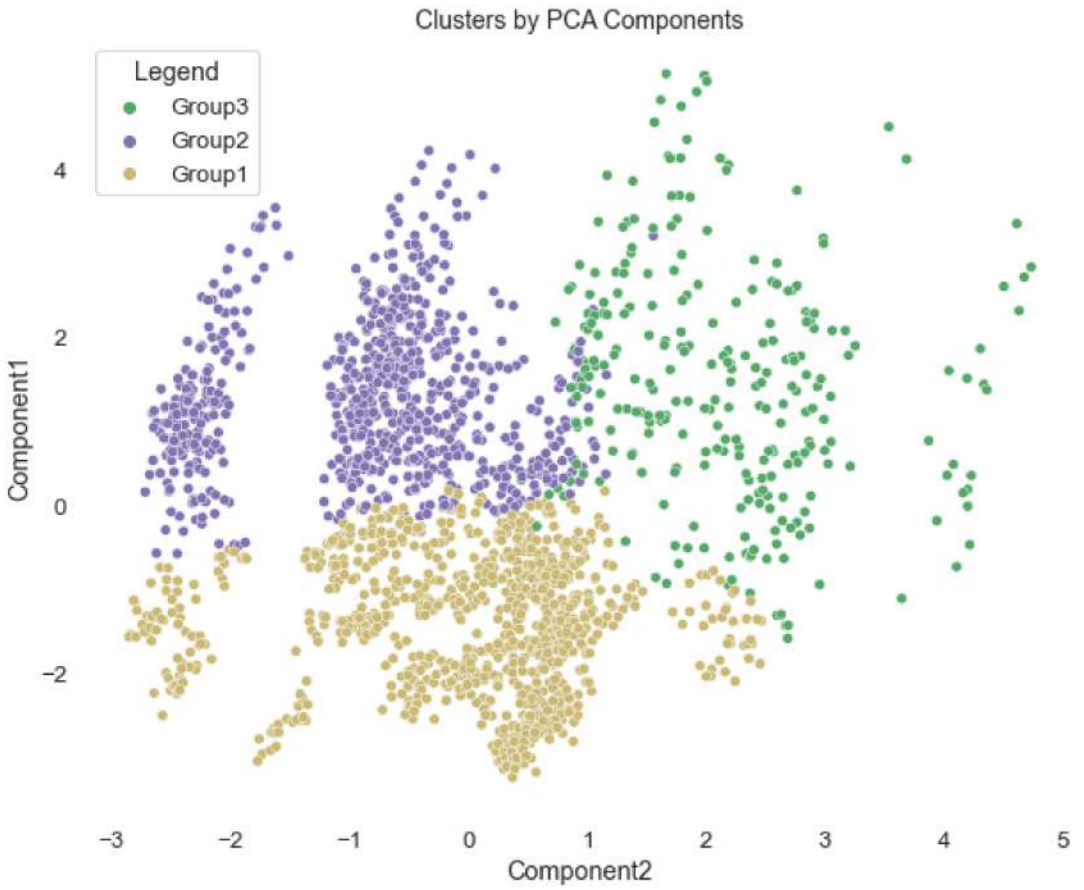
Segment K-means PCA	customer_id	gender	marital_status	age	\
Group2	1.000006e+08	0.128571	0.279221	34.596104	
Group3	1.000009e+08	0.498182	0.676364	55.720000	
Group1	1.000013e+08	0.709948	0.619895	31.262827	

Segment K-means PCA	education	income	occupation	settlement_size	\
Group2	0.766234	137723.603896	1.188312	1.340260	
Group3	2.127273	155251.661818	1.076364	1.054545	
Group1	0.943455	97557.545550	0.429319	0.163351	

Segment K-means PCA	Component1	Component2	Component3	N Obs	Prop Obs
Group2	1.253369	-0.792837	-0.302295	770	0.3850
Group3	1.436546	2.138671	0.904803	275	0.1375
Group1	-1.424235	0.023403	-0.016810	955	0.4775

[ ]:
```
kmeansSegmentedPca['Legend'] = kmeansSegmentedPca['Segment K-means PCA'].map({0:
 ↪'Group2',
                                                    1:'Group3',
                                                    2:'Group1',
                                                    3:'Group0'})
```

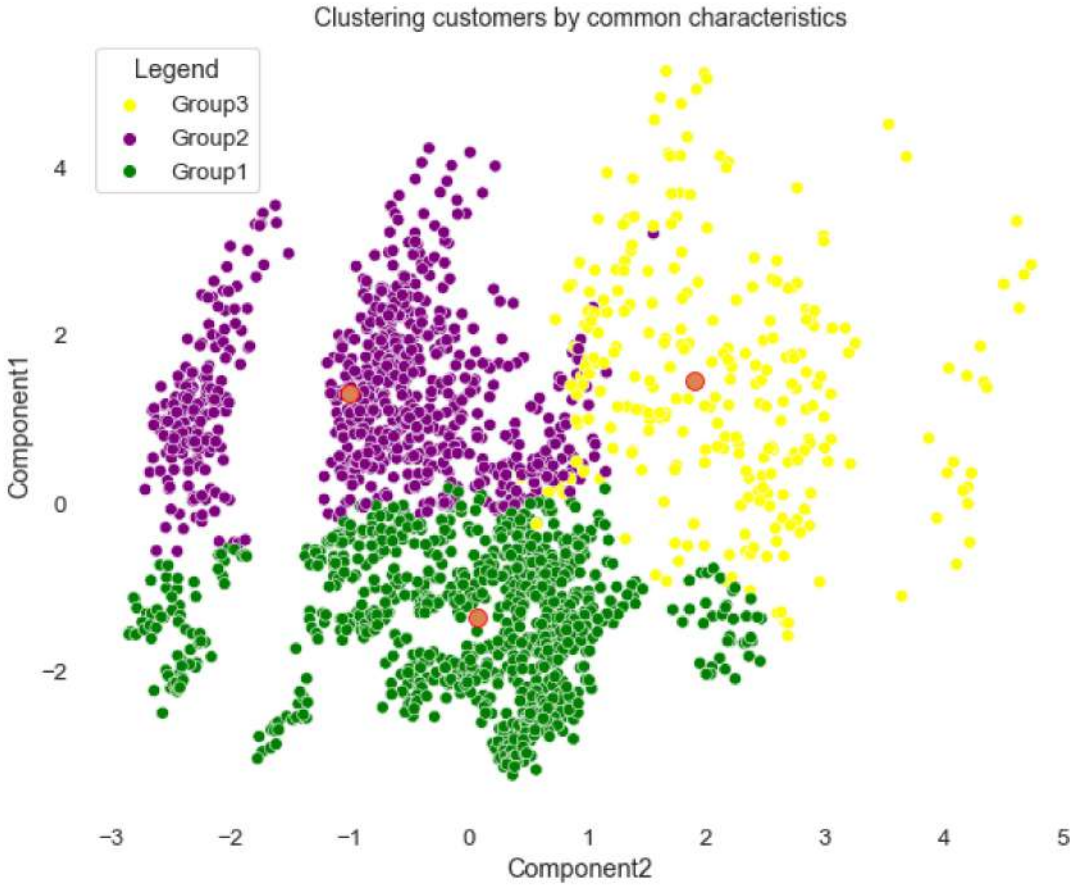
[ ]:
```
# Plot components
x_axis = kmeansSegmentedPca['Component2']
y_axis = kmeansSegmentedPca['Component1']
plt.figure(figsize = (10, 8))
sns.scatterplot(x_axis, y_axis, hue = kmeansSegmentedPca['Legend'], palette =␣
 ↪['g', 'm', 'y'])
plt.title('Clusters by PCA Components',fontsize=14)
plt.show()
```



[ ]:
```
kmeans = KMeans(n_clusters = 3, random_state = 45)
k_fit = kmeans.fit(df_one)
clusters = k_fit.labels_
```

[ ]:
```
# Plot components
x_axis = kmeansSegmentedPca['Component2']
y_axis = kmeansSegmentedPca['Component1']
X = np.vstack([x_axis, y_axis]).T
num_clusters = 3
kmeans = KMeans(n_clusters=num_clusters).fit(X)
plt.figure(figsize = (10, 8))
colors = ['yellow', 'purple', 'green']
ax = sns.scatterplot(x_axis, y_axis, hue = kmeansSegmentedPca['Legend'],␣
 ↪palette = colors, s = 50)
ax = sns.scatterplot(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:,␣
 ↪1], palette=colors, s=100, ec='red', legend=False, ax=ax)
plt.title('Clustering customers by common characteristics',fontsize=14)
plt.rcParams['axes.facecolor'] = 'white'
```

```
plt.show()
fig = ax.get_figure()
fig.savefig("out.png")
```



[ ]:
```
data = pd.DataFrame({"customer_id": df['customer_id'], "Group": (k_fit.labels_␣
↪+ 1)})
data.head(10)
```

[ ]:

	customer_id	Group
0	100000001	3
1	100000002	3
2	100000003	1
3	100000004	2
4	100000005	3
5	100000006	3
6	100000007	3
7	100000008	2
8	100000009	3

```
9       100000010       2
```