Using unsupervised learning techniques of Principal Component Analysis (PCA) and KMeans to identify customer segments of the German population that were popular or less popular with a mail-order sales in Germany.

The dataset :

The data is provided by Bertelsmann partners AZ Direct and Arvato Financial Solution. There are two dataset provided first was demographic data for the general population of Germany and the second dataset was demographic data for customers of a mail-order company. Both dataset consists of 85 different features.

Data Wrangling :

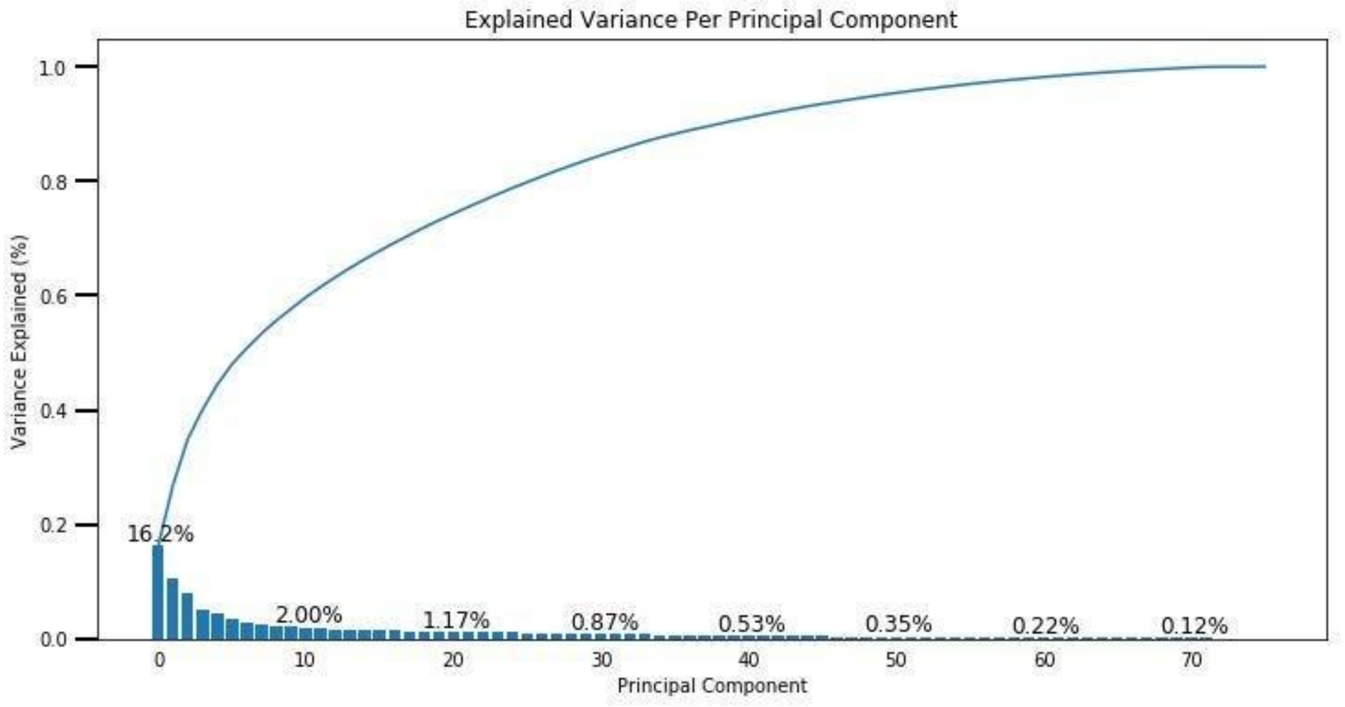
The data was quite messy and some was irrelevant. We need converting and mapping the data to appropriate format for data analytics.

- First we deal with missing values in the dataset by dropping variables and rows with extremely high frequencies of missing values.
- Next we need convert categorical variables to binary variables by using one-hot encoded features.
- Then we identify mixed-type features and engineer new features because some variables that had more than two different piece of information.
- We use Use StandardScaler to scale each feature to mean 0 and standard deviation 1.

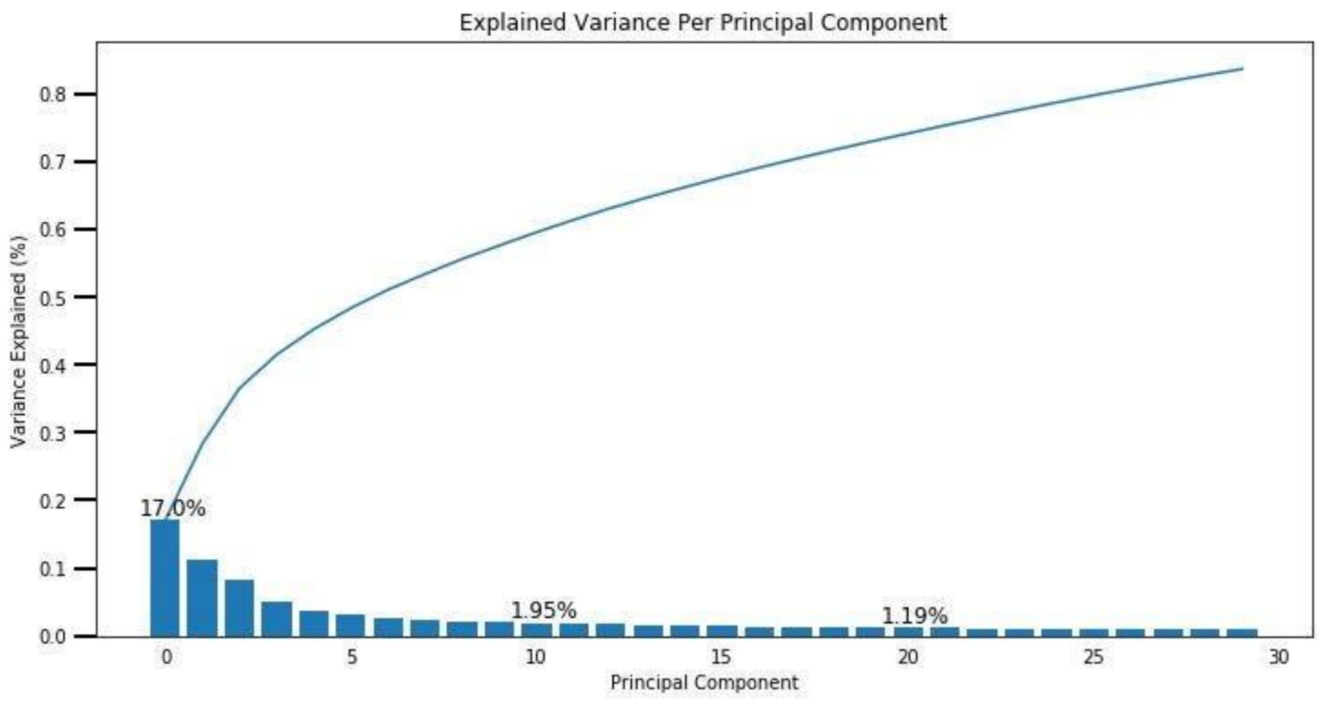
PCA — Dimensionality Reduction :

The main idea of principal component analysis (PCA) is to reduce the dimensionality of a data set consisting of many variables correlated with each other while retaining the variation present in the dataset.

The results of the PCA, we can observe the relative variance in the data is captured along each component.



The PCA shows that first component which is the first blue bar got 16.2% the highest variance explained. Then the next component which explain the next highest percentage of variance. Total 71 components to capture all of the variance in the data. However, we'll not use all the 71 components for 100% of our variance but we'll use 30 components. Let see the plot below:



This plot tells us that selecting 30 components we can preserve something around 80% of the total variance of the data.

Interpret Principal Components

We’ll show the top 5 weights of the variable on the first component to interpret their relationship, what a positive or negative value indicates.

Workflow :

Loading data

Data exlporation

Train/ test split

Prepare the data

Principal Component Analysis (PCA)

K-means clustring

Pipeline

Validate with test data

Imports:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import plotly.express as px
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
```

Loading data :

```
data_df=pd.read_csv('/work/Mall_Customers.csv')
data_df.head()
```

	Customer id	Gender	Age	Annual income	Spending score
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
print(data_df.shape)
print(data_df.info())
```

```
(200, 5)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
```

```
#   Column                  Non-Null Count  Dtype
---  ------                 --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

Cheking for missing data:

```
data_df.isnull().sum()
```

droping column:

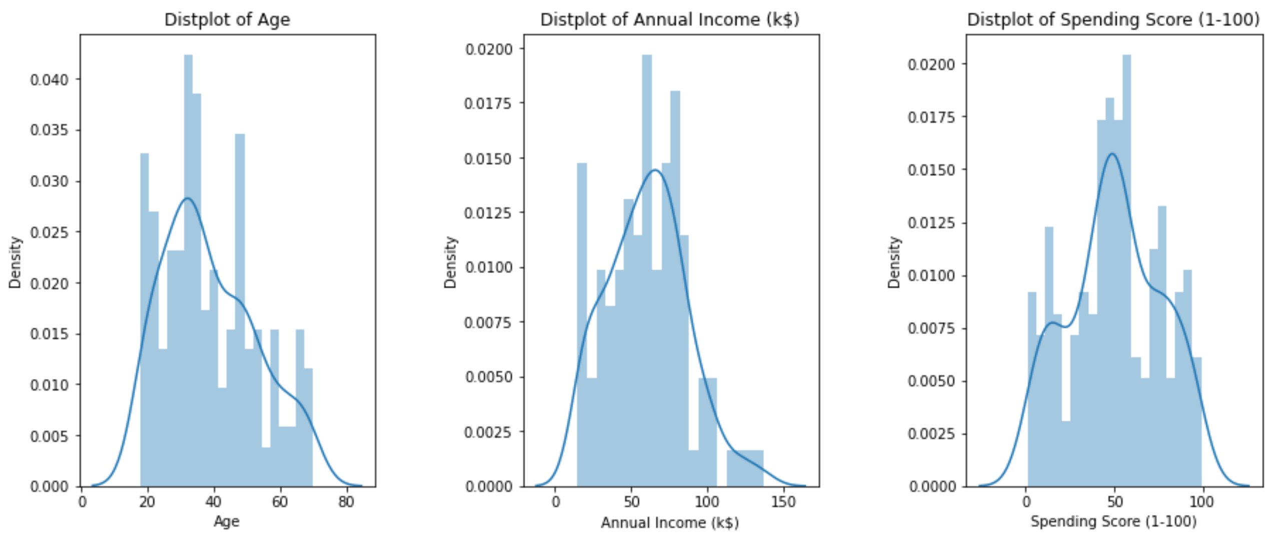
```
# drop Customer id
Data_df = data_df.drop('CustomerID', axis=1)
Data_df.head(2)
```

	Gender	Age	Annual income	Spending score
0	Male	19	15	39
1	Male	21	15	81

Data exlporation:

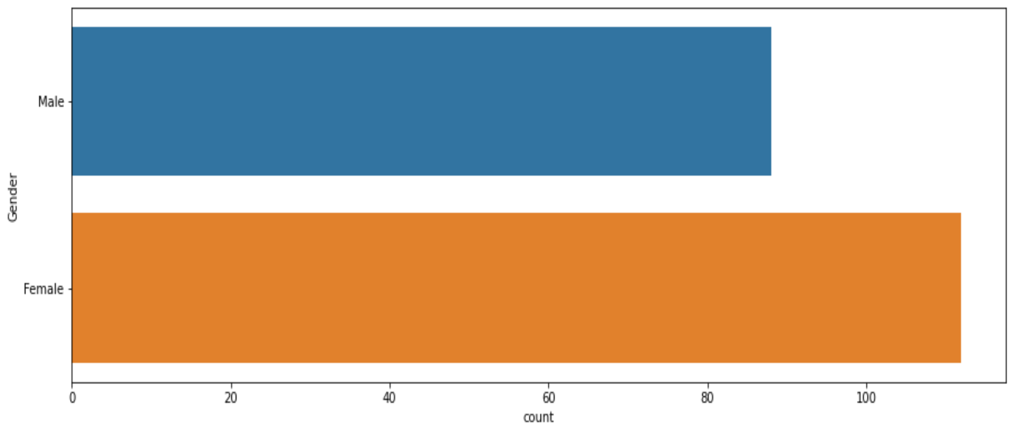
```
plt.figure(1 , figsize = (15 , 6))
n = 0
for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']: n += 1
   plt.subplot(1 , 3 , n)
   plt.subplots_adjust(hspace =0.5 , wspace = 0.5)
   sns.distplot(data_df[x] , bins = 20)
```

```
    plt.title('Distplot of {}'.format(x))
plt.show()
```



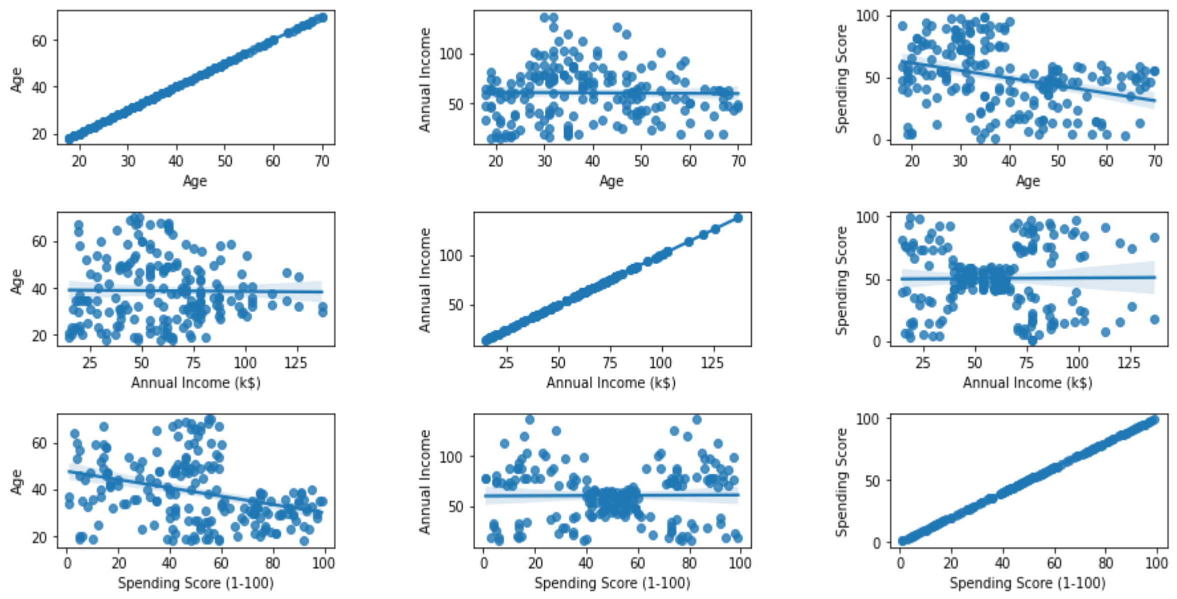
Age groupe of 30-40 have the highest density => Most of our custumors have an income of 50-80k$ => most of our customrs have a spending score of 50.

```
plt.figure(1 , figsize = (15 , 5))
sns.countplot(y = 'Gender' , data = data_df)
plt.show()
```



20% more of our customrs are female.

```
plt.figure(1 , figsize = (15 , 7))
n = 0
for x in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
    for y in ['Age' , 'Annual Income (k$)' , 'Spending Score (1-100)']:
        n += 1
        plt.subplot(3 , 3 , n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
        sns.regplot(x = x , y = y , data = data_df)
        plt.ylabel(y.split()[0]+' '+y.split()[1] if len(y.split()) > 1 else y )
plt.show()
```



Age Vs Annual income

=> 30-60 years old customers have relatively higher annual income

Age Vs spending score

=> Under 40 years old customers have score over 60

Annula income vs Spending score

=> We can see 5 blobs, there is no relation between the customers having annual income in the range of 20-40k and 80-140k but there is relation between customers who have annual income in the range of 40-60k.

Train/ test split:

```
train_X, test_X = train_test_split(data_df, test_size=0.2, random_state=42)

print(len(train_X), "train +", len(test_X), "test")
```

```
160 train + 40 test
```

```
# lets take copy of the data

df = train_X.copy()
```

Prepare the data The following stages are:

Preprocessing.LabelEncoder() -normalize labels so they contain values between 0 and 1

StandardScaler - scaling to unit variance (ie Normalizing the data)

Principal Component analysis(PCA)- is an unsupervised statistical technique that is used for dimensionality reduction.

Feature selection.

label encoder :

```
# Let fit and transform the Gender attribute into numeric

le = LabelEncoder()

le.fit(df.Gender)

df.loc[:,'Gender'] = le.transform(df.Gender)

df.head(3)
```

	Gender	Age	Annual income	Spending score
79	0	49	49	54
197	1	32	32	126
38	0	36	36	37

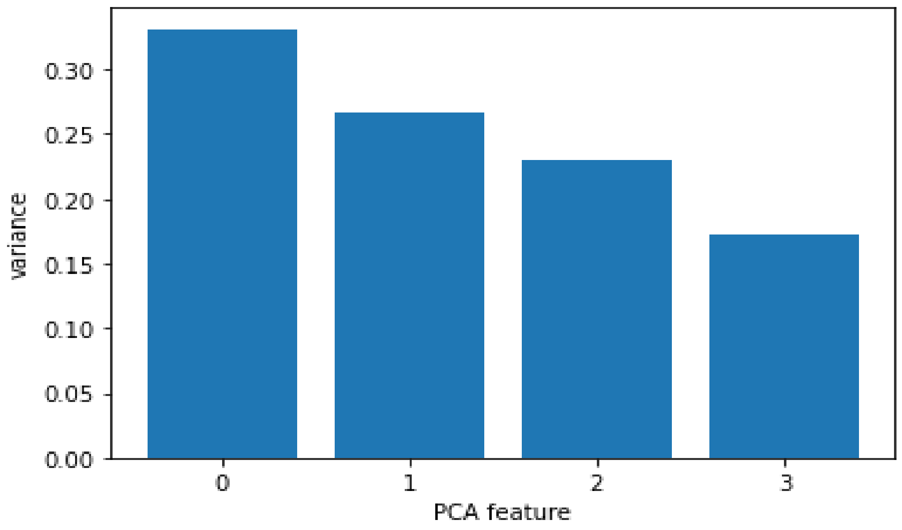
StandardScaler :

```
# Create scaler: scaler
```

```
scaler = StandardScaler()
scaler.fit(df)
# transform
data_scaled = scaler.transform(df)
data_scaled[0]
```

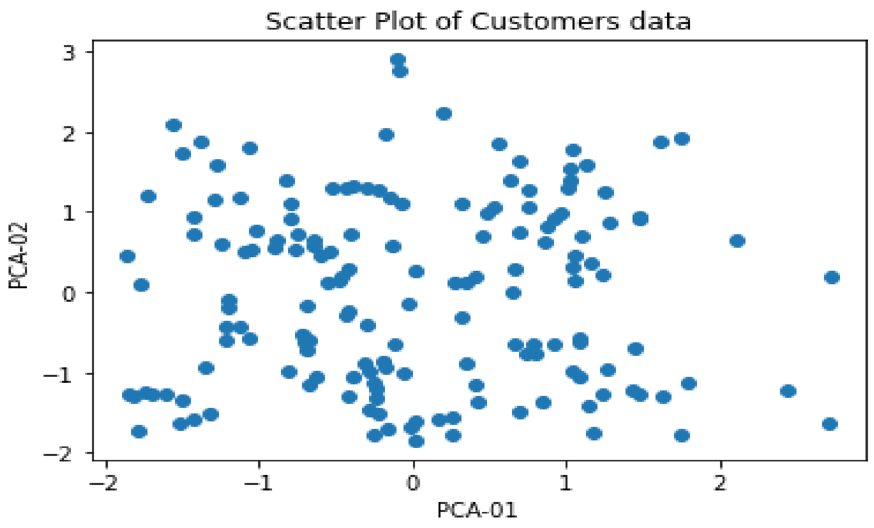
Principal Component Analysis (PCA) :

```
pca = PCA()
# fit PCA
pca.fit(data_scaled)
# PCA features
features = range(pca.n_components_)
# PCA transformed data
data_pca = pca.transform(data_scaled)
pca.explained_variance_ratio_
plt.bar(features, pca.explained_variance_ratio_)
plt.xticks(features)
plt.ylabel('variance')
plt.xlabel('PCA feature')
plt.show()
```



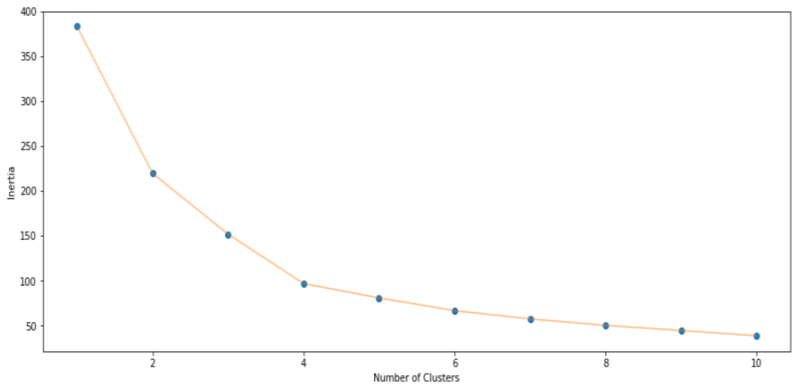
as we can see 33.1% of the data corresponds to the first axis and 26.8% corresponds to the seconds we will work with 2 insintric dimensions (number of PCA features needed to approximate the dataset).

```
# Principal component analysis (PCA) and singular value decomposition (SVD)

# PCA and SVD are closely related approaches and can be both applied to decompose any rectangular matrices.

pca2 = PCA(n_components=2, svd_solver='full')

# fit PCA

pca2.fit(data_scaled)

# PCA transformed data

data_pca2 = pca2.transform(data_scaled)

print(data_pca2.shape)

xs = data_pca2[:,0]

ys = data_pca2[:,1]

plt.scatter(ys, xs)

plt.grid(False)

plt.title('Scatter Plot of Customers data')

plt.xlabel('PCA-01')
plt.ylabel('PCA-02')

plt.show()
```



Determine the number of K-means clusters needed :

```
#  finding elbow value for different number of clusters.

X = data_pca2

inertia = []

for n in range(1 , 11):

   algorithm = (KMeans(n_clusters = n ,init='k-means++',random_state= 42 ) )

   algorithm.fit(X)

   inertia.append(algorithm.inertia_)

plt.figure(1 , figsize = (15 ,6))

plt.plot(np.arange(1 , 11) , inertia , 'o')

plt.plot(np.arange(1 , 11) , inertia , '-' , alpha = 0.5)

plt.xlabel('Number of Clusters') , plt.ylabel('Inertia')

plt.show()
```



K-means clustring :

```
# KMeans model

# 5 clusters to start with

kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)
```

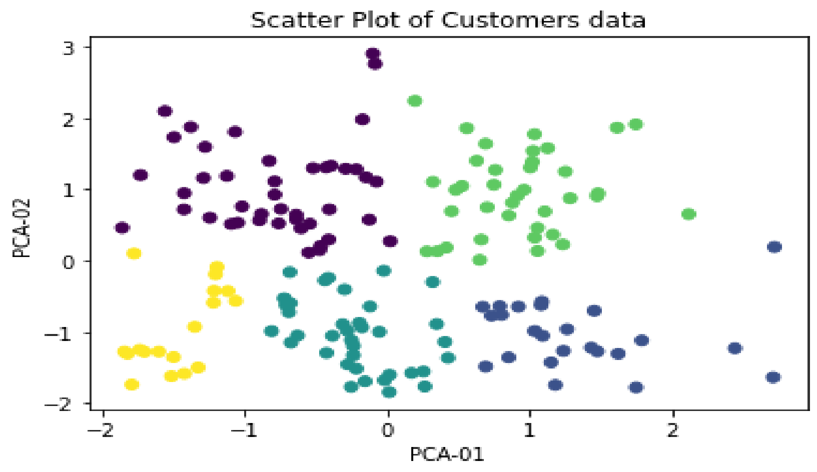
Pipeline

```
# Build pipeline

pipeline = make_pipeline(scaler, pca2, kmeans)


# fit the model to the scaled dataset

model_fit = pipeline.fit(df)

model_fit
```

assign labels :

```
# return a label for each data point based on their cluster

labels = model_fit.predict(df)

train_X['Clusters'] = labels

# Number of data points for each feature in each cluster

train_X.groupby('Clusters').count()


# Scatter plot visuals with labels

xs = data_pca2[:,0]

ys = data_pca2[:,1]
```

```
#zs = train_X.iloc[:,2]

plt.scatter(ys, xs,c=labels)

#plt.scatter(ys, zs, c=labels)

plt.grid(False)

plt.title('Scatter Plot of Customers data')

plt.xlabel('PCA-01')

plt.ylabel('PCA-02')

plt.show()
```



Conclusion :

For the given dataset the customers are segmented into 5 clusters using PCA and k meams algorithm