Customer Segmentation using Unsupervised Machine Learning in Python

In today's era, companies work hard to make their customers happy. They launch new technologies and services so that customers can use their products more. They try to be in touch with each of their customers so that they can provide goods accordingly. But practically, it's very difficult and non-realistic to keep in touch with everyone. So, here comes the usage of **Customer Segmentation.**

Customer Segmentation means the segmentation of customers on the basis of their similar characteristics, behavior, and needs. This will eventually help the company in many ways. Like, they can launch the product or enhance the features accordingly. They can also target a particular sector as per their behaviors. All of these lead to an enhancement in the overall market value of the company.

Customer Segmentation using Unsupervised Machine Learning in Python

Today we will be using Machine Learning to implement the task of Customer Segmentation.

Import Libraries

The libraries we will be required are :

- Pandas – This library helps to load the data frame in a 2D array format.
- Numpy – Numpy arrays are very fast and can perform large computations.
- Matplotlib / Seaborn – This library is used to draw visualizations.
- Sklearn – This module contains multiple libraries having pre-implemented functions to perform tasks from data preprocessing to model development and evaluation.

Python3


```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb

from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.cluster import KMeans

import warnings
warnings.filterwarnings('ignore')
```

The dataset taken for the task includes the details of customers includes their marital status, their income, number of items purchased, types of items purchased, and so on.

Python3

```
df = pd.read_csv('new.csv')
df.head()
```

Output:

	ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	\
0	5524	1957	Graduation	Single	58138.0	0	0	
1	2174	1954	Graduation	Single	46344.0	1	1	
2	4141	1965	Graduation	Together	71613.0	0	0	
3	6182	1984	Graduation	Together	26646.0	1	0	
4	5324	1981	PhD	Married	58293.0	1	0	

	Dt_Customer	Recency	MntWines	...	NumDealsPurchases	NumWebPurchases	\
0	04-09-2012	58	635	...	3	8	
1	08-03-2014	38	11	...	2	1	
2	21-08-2013	26	426	...	1	8	
3	10-02-2014	26	11	...	2	2	
4	19-01-2014	94	173	...	5	5	

	NumCatalogPurchases	NumStorePurchases	NumWebVisitsMonth	Complain	\
0	10	4	7	0	
1	1	2	5	0	
2	2	10	4	0	
3	0	4	6	0	
4	3	6	5	0	

	Z_CostContact	Z_Revenue	Response	Accepted
0	3	11	1	AcceptedCmp1
1	3	11	0	AcceptedCmp1
2	3	11	0	AcceptedCmp1
3	3	11	0	AcceptedCmp1
4	3	11	0	AcceptedCmp1

Python3

```
df.shape
```

Output:

```
(2240, 25)(2240, 25)
```

To get the information of the dataset like checking the null values, count of values, etc. we will use .info() method.

Data Preprocessing

Python3

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 25 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   ID                   2240 non-null   int64  
 1   Year_Birth           2240 non-null   int64  
 2   Education            2240 non-null   object 
 3   Marital_Status       2240 non-null   object 
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64  
 6   Teenhome             2240 non-null   int64  
 7   Dt_Customer          2240 non-null   object 
 8   Recency              2240 non-null   int64  
 9   MntWines             2240 non-null   int64  
 10  MntFruits            2240 non-null   int64  
 11  MntMeatProducts      2240 non-null   int64  
 12  MntFishProducts      2240 non-null   int64  
 13  MntSweetProducts     2240 non-null   int64  
 14  MntGoldProds         2240 non-null   int64  
 15  NumDealsPurchases    2240 non-null   int64  
 16  NumWebPurchases      2240 non-null   int64  
 17  NumCatalogPurchases  2240 non-null   int64  
 18  NumStorePurchases    2240 non-null   int64  
 19  NumWebVisitsMonth    2240 non-null   int64  
 20  Complain             2240 non-null   int64  
 21  Z_CostContact        2240 non-null   int64  
 22  Z_Revenue            2240 non-null   int64  
 23  Response             2240 non-null   int64  
 24  Accepted             2240 non-null   object 
dtypes: float64(1), int64(20), object(4)
```

Python3

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

Output:

	count	mean	std	min	25%	50%	75%	max
ID	2240.0	5592.159821	3246.662198	0.0	2828.25	5458.5	8427.75	11191.0
Year_Birth	2240.0	1968.805804	11.984069	1893.0	1959.00	1970.0	1977.00	1996.0
Income	2216.0	52247.251354	25173.076661	1730.0	35303.00	51381.5	68522.00	666666.0
Kidhome	2240.0	0.444196	0.538398	0.0	0.00	0.0	1.00	2.0
Teenhome	2240.0	0.506250	0.544538	0.0	0.00	0.0	1.00	2.0
Recency	2240.0	49.109375	28.962453	0.0	24.00	49.0	74.00	99.0
MntWines	2240.0	303.935714	336.597393	0.0	23.75	173.5	504.25	1493.0
MntFruits	2240.0	26.302232	39.773434	0.0	1.00	8.0	33.00	199.0
MntMeatProducts	2240.0	166.950000	225.715373	0.0	16.00	67.0	232.00	1725.0
MntFishProducts	2240.0	37.525446	54.628979	0.0	3.00	12.0	50.00	259.0
MntSweetProducts	2240.0	27.062946	41.280498	0.0	1.00	8.0	33.00	263.0
MntGoldProds	2240.0	44.021875	52.167439	0.0	9.00	24.0	56.00	362.0
NumDealsPurchases	2240.0	2.325000	1.932238	0.0	1.00	2.0	3.00	15.0
NumWebPurchases	2240.0	4.084821	2.778714	0.0	2.00	4.0	6.00	27.0
NumCatalogPurchases	2240.0	2.662054	2.923101	0.0	0.00	2.0	4.00	28.0
NumStorePurchases	2240.0	5.790179	3.250958	0.0	3.00	5.0	8.00	13.0
NumWebVisitsMonth	2240.0	5.316518	2.426645	0.0	3.00	6.0	7.00	20.0
Complain	2240.0	0.009375	0.096391	0.0	0.00	0.0	0.00	1.0
Z_CostContact	2240.0	3.000000	0.000000	3.0	3.00	3.0	3.00	3.0
Z_Revenue	2240.0	11.000000	0.000000	11.0	11.00	11.0	11.00	11.0
Response	2240.0	0.149107	0.356274	0.0	0.00	0.0	0.00	1.0

Improving the values in the Accepted column.

Python3

```
df['Accepted'] = df['Accepted'].str.replace('Accepted', '')
```

```
for col in df.columns:
    temp = df[col].isnull().sum()
    if temp > 0:
        print(f'Column {col} contains {temp} null values.')
```

Output:

```
Column Income contains 24 null values.
```

Now, once we have the count of the null values and we know the values are very less we can drop them (it will not affect the dataset much).

Python3

```
df = df.dropna()
print("Total missing values are:", len(df))
```

Output:

```
Total missing values are: 2216
```

To find the total number of unique values in each column we can use data.unique() method.

Python3

```
df.nunique()
```

ID	2216
Year_Birth	59
Education	5
Marital_Status	8
Income	1974
Kidhome	3
Teenhome	3
Dt_Customer	662
Recency	100
MntWines	776
MntFruits	158
MntMeatProducts	554
MntFishProducts	182
MntSweetProducts	176
MntGoldProds	212
NumDealsPurchases	15
NumWebPurchases	15
NumCatalogPurchases	14
NumStorePurchases	14
NumWebVisitsMonth	16
Complain	2
Z_CostContact	1
Z_Revenue	1
Response	2
Accepted	5

Here we can observe that there are columns which contain single values in the whole column so, they have no relevance in the model development.

Also dataset has a column **Dt_Customer** which contains the date column, we can convert into 3 columns i.e.

```
parts = df["Dt_Customer"].str.split("-", n=3, expand=True)
df["day"] = parts[0].astype('int')
df["month"] = parts[1].astype('int')
df["year"] = parts[2].astype('int')
```

Now we have all the important features, we can now drop features like **Z_CostContact, Z_Revenue, Dt_Customer.**

Python3

```
df.drop(['Z_CostContact', 'Z_Revenue', 'Dt_Customer'],
        axis=1,
        inplace=True)
```

Data Visualization and Analysis

Data visualization is the graphical representation of information and data in a pictorial or graphical format. Here we will be using bar plot and count plot for better visualization.

Python3

```
floats, objects = [], []
for col in df.columns:
    if df[col].dtype == object:
        objects.append(col)
    elif df[col].dtype == float:
        floats.append(col)
```

Output:

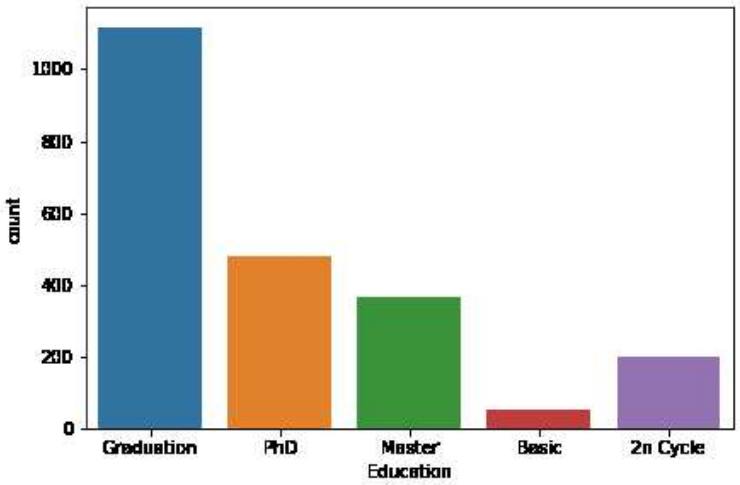
```
['Education', 'Marital_Status', 'Accepted']
['Income']
```

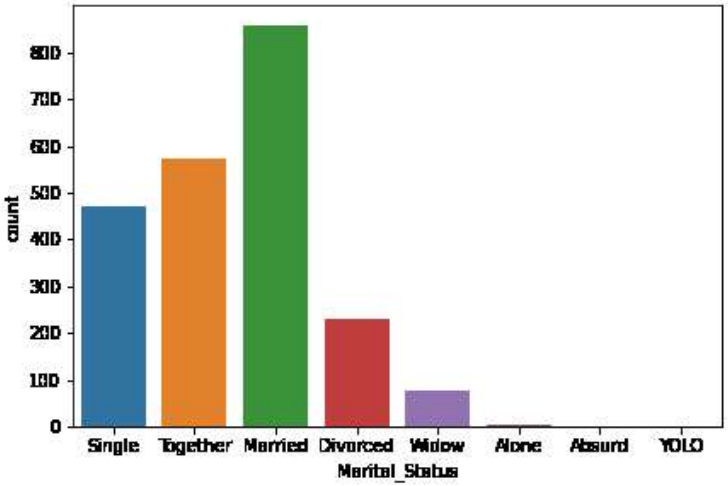
To get the count plot for the columns of the datatype – object, refer the code below.

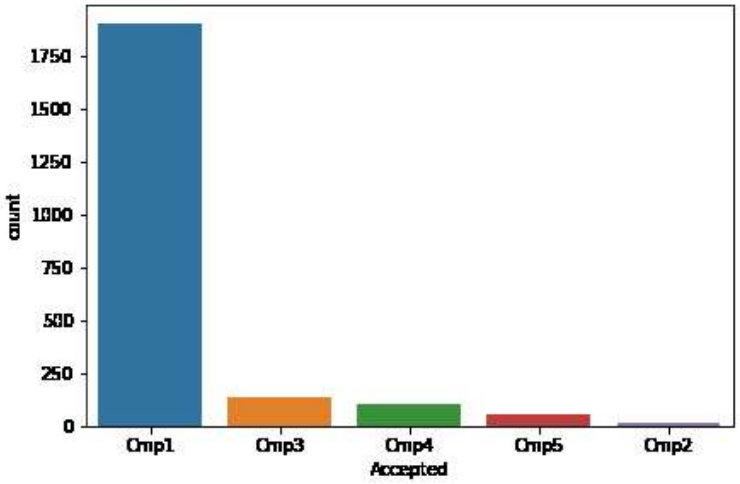
Python3

```
plt.subplots(figsize=(15, 10))
for i, col in enumerate(objects):
    plt.subplot(2, 2, i + 1)
    sb.countplot(df[col])
plt.show()
```

Output:







Let's check the value_counts of the Marital_Status of the data.

Python3

```
df['Marital_Status'].value_counts()
```

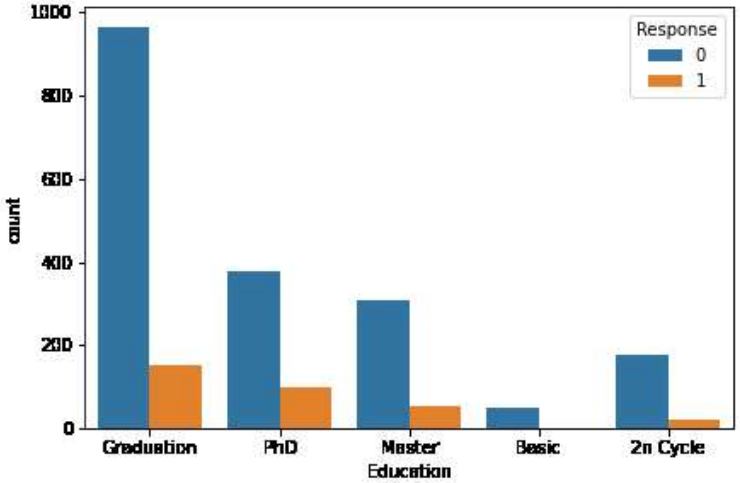
```
Married     857
Together    573
Single      471
Divorced    232
Widow        76
Alone         3
Absurd        2
YOLO          2
Name: Marital_Status, dtype: int64
```

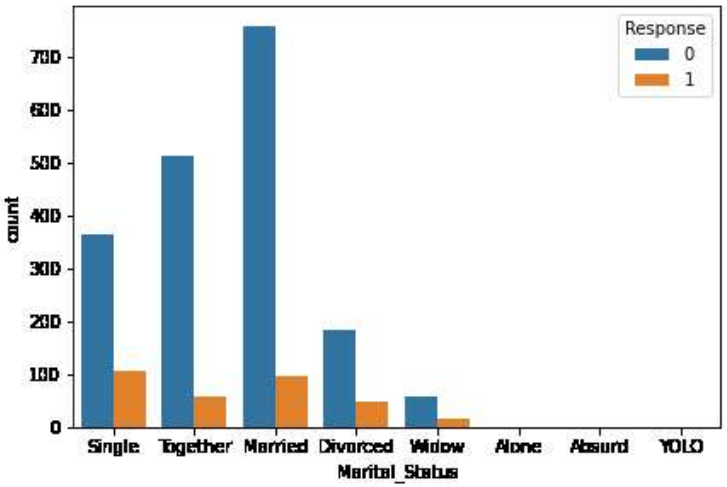
Now lets see the comparison of the features with respect to the values of the responses.

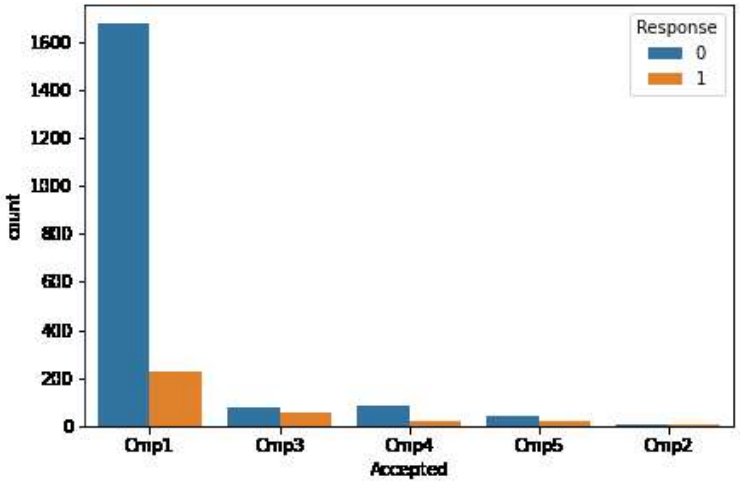
Python3

```
plt.subplots(figsize=(15, 10))
for i, col in enumerate(objects):
    plt.subplot(2, 2, i + 1)
    sb.countplot(df[col], hue=df['Response'])
plt.show()
```

Output:







Label Encoding

Label Encoding is used to convert the categorical values into the numerical values so that model can understand it.

Python3

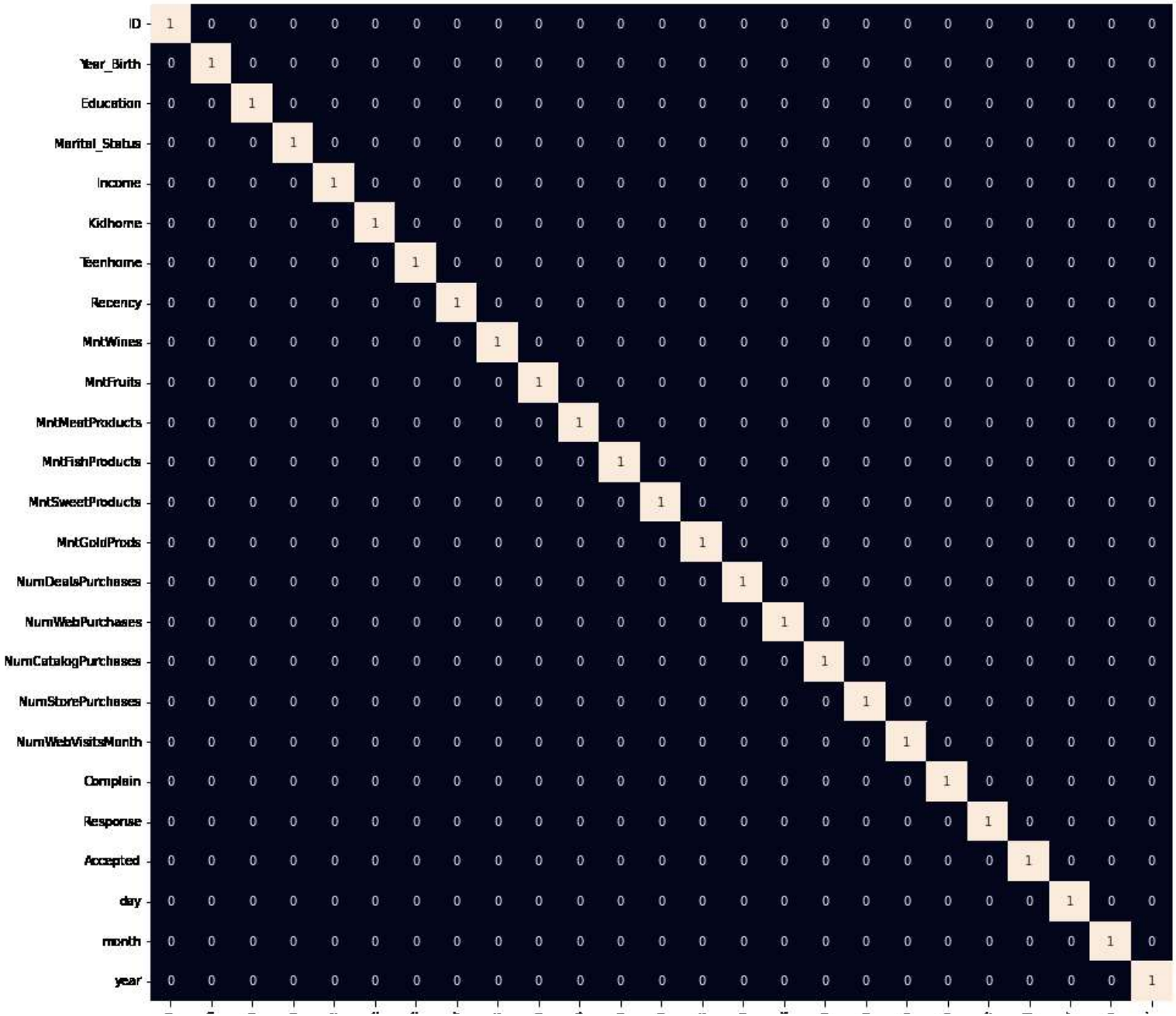
```
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
```

Heatmap is the best way to visualize the correlation among the different features of dataset. Let's give it the value of 0.8

Python3

```
plt.figure(figsize=(15, 15))
sb.heatmap(df.corr() > 0.8, annot=True, cbar=False)
plt.show()
```

Output:



Standardization

Standardization is the method of feature scaling which is an integral part of feature engineering. It scales down the data and making it easier for the machine learning model to learn from it. It reduces the mean to '0' and the standard deviation to '1'.

Python3

```
scaler = StandardScaler()
data = scaler.fit_transform(df)
```

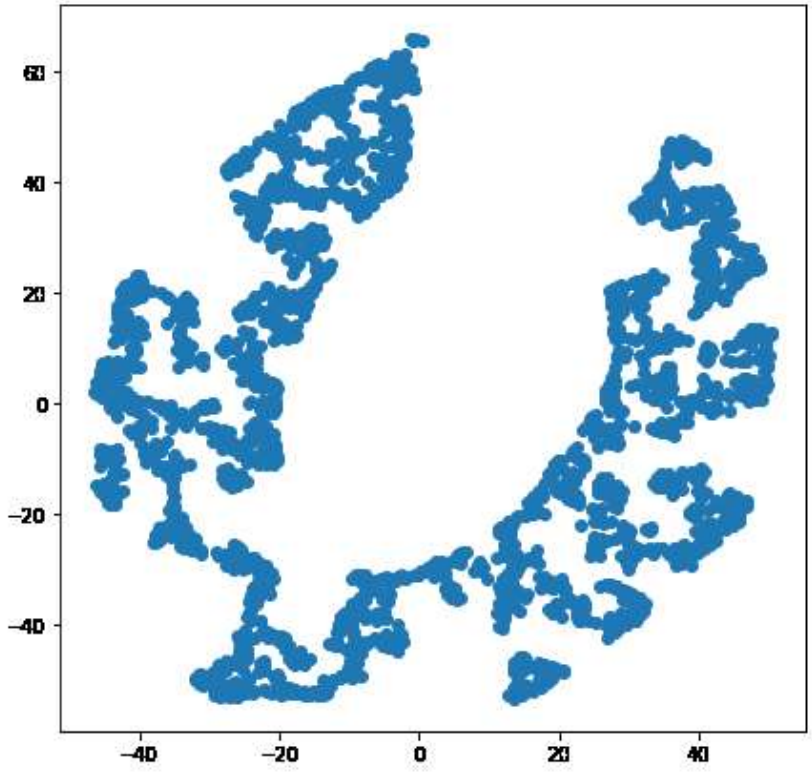
Segmentation

We will be using T-distributed Stochastic Neighbor Embedding. It helps in visualizing high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the values to low-dimensional embedding.

Python3

```
from sklearn.manifold import TSNE
model = TSNE(n_components=2, random_state=0)
tsne_data = model.fit_transform(df)
plt.figure(figsize=(7, 7))
plt.scatter(tsne_data[:, 0], tsne_data[:, 1])
plt.show()
```

Output:



There are certainly some clusters which are clearly visual from the 2-D representation of the given data. Let's use the KMeans algorithm to find those clusters in the high dimensional plane itself

KMeans Clustering can also be used to cluster the different points in a plane.

Python3

```
error = []
for n_clusters in range(1, 21):
    model = KMeans(init='k-means++',
                   n_clusters=n_clusters
```

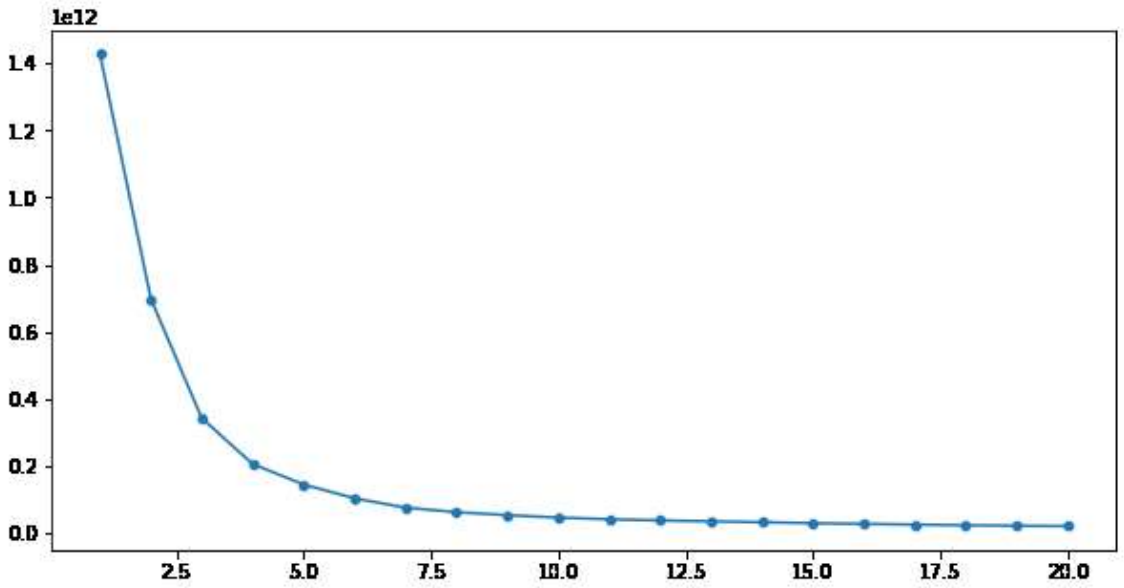
```
    error.append(model.inertia_)
```

Here inertia is nothing but the sum of squared distances within the clusters.

Python3

```
plt.figure(figsize=(10, 5))
sb.lineplot(x=range(1, 21), y=error)
sb.scatterplot(x=range(1, 21), y=error)
plt.show()
```

Output:



Here by using the elbow method we can say that k = 6 is the optimal number of clusters that should be made as after k = 6 the value of the inertia is not decreasing drastically

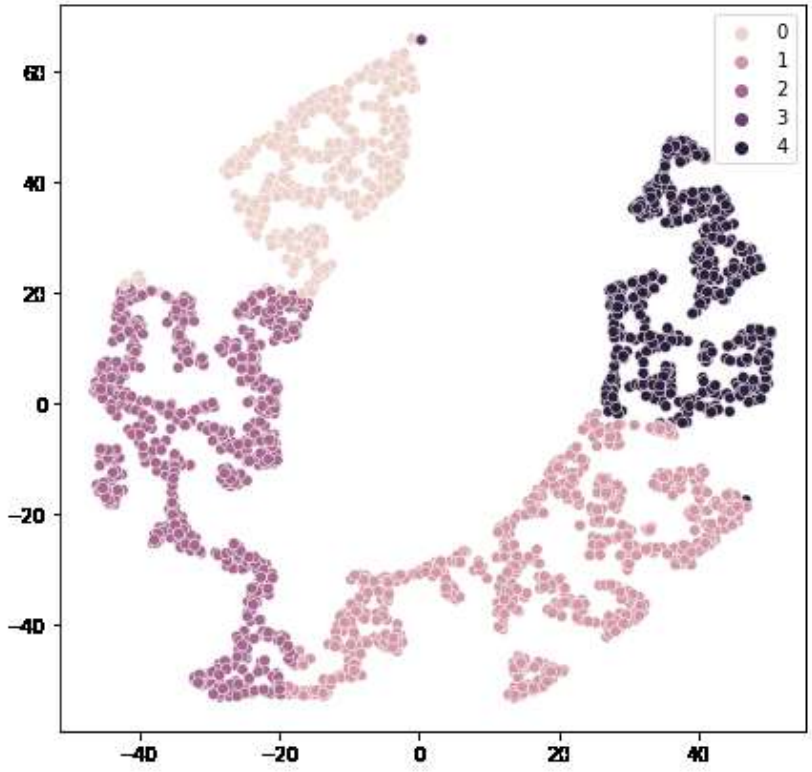
```
# create clustering model with optimal k=5
model = KMeans(init='k-means++',
               n_clusters=5,
               max_iter=500,
               random_state=22)
segments = model.fit_predict(df)
```

Scatterplot will be used to see all the 6 clusters formed by KMeans Clustering.

Python3

```
plt.figure(figsize=(7, 7))
sb.scatterplot(tsne_data[:, 0], tsne_data[:, 1], hue=segments)
plt.show()
```

Output:



Whether you're preparing for your first job interview or aiming to upskill in this ever-evolving tech landscape, GeeksforGeeks Courses are your key to success. We provide top-quality content at affordable prices, all geared towards accelerating your growth in a time-bound manner. Join the millions we've already empowered, and we're here to do the same for you. Don't miss out - check it out now!

Last Updated : 21 Nov, 2022