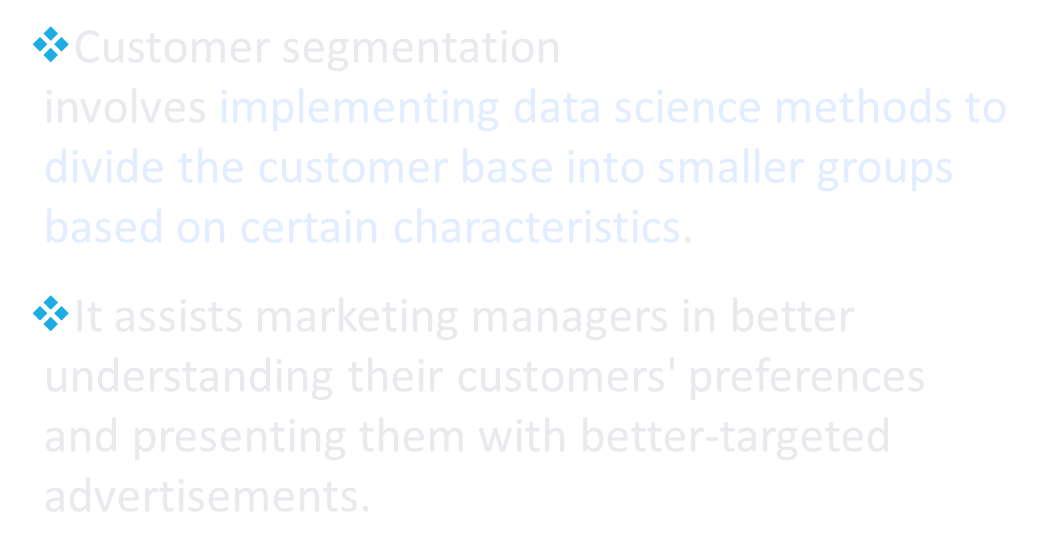
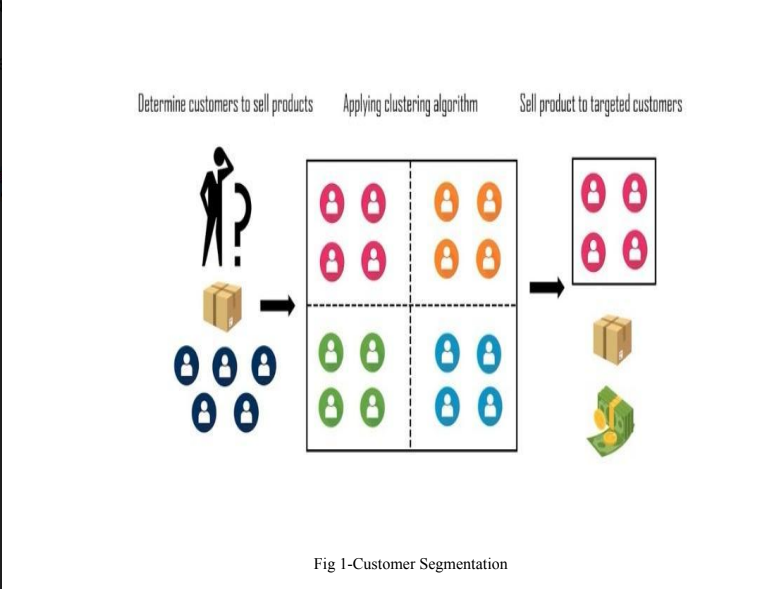
|  |
| --- |
| **Customer Segmentation**   **using Data Science** |
| **Team members:**  1) F . ANTONY RAJAN (961721104303)  2) A . FRENI (961721104008)  3) A . EBISTON (961721104307)  4) P . FRANKLIN GINO (961721104308)  5) M.S. JEBINLAL (961721104309)  Phase 2 Submission Document:  **Project:Customer Segmentation using data science** |



**Introduction:**





**Content for Project Phase 2 :**

* For analyzing data, we need some libraries. In this section, we are importing all the required libraries like pandas, NumPy, matplotlib, plotly, seaborn, and word cloud that are required for data analysis. Check the below code to import all the required libraries**.**

**Data Source:**

* A good data source for credit card fraud detection should be Accurate,Complete, Covering the geographic area of interest, Accessible.
* Dataset Link(https://www.kaggle.com/datasets/akram24/mall-customers)

**Data Collection and Preprocessing:**

* Data sources
* Data collection methods
* Data cleaning and preprocessing steps
* Mention any challenges encountered

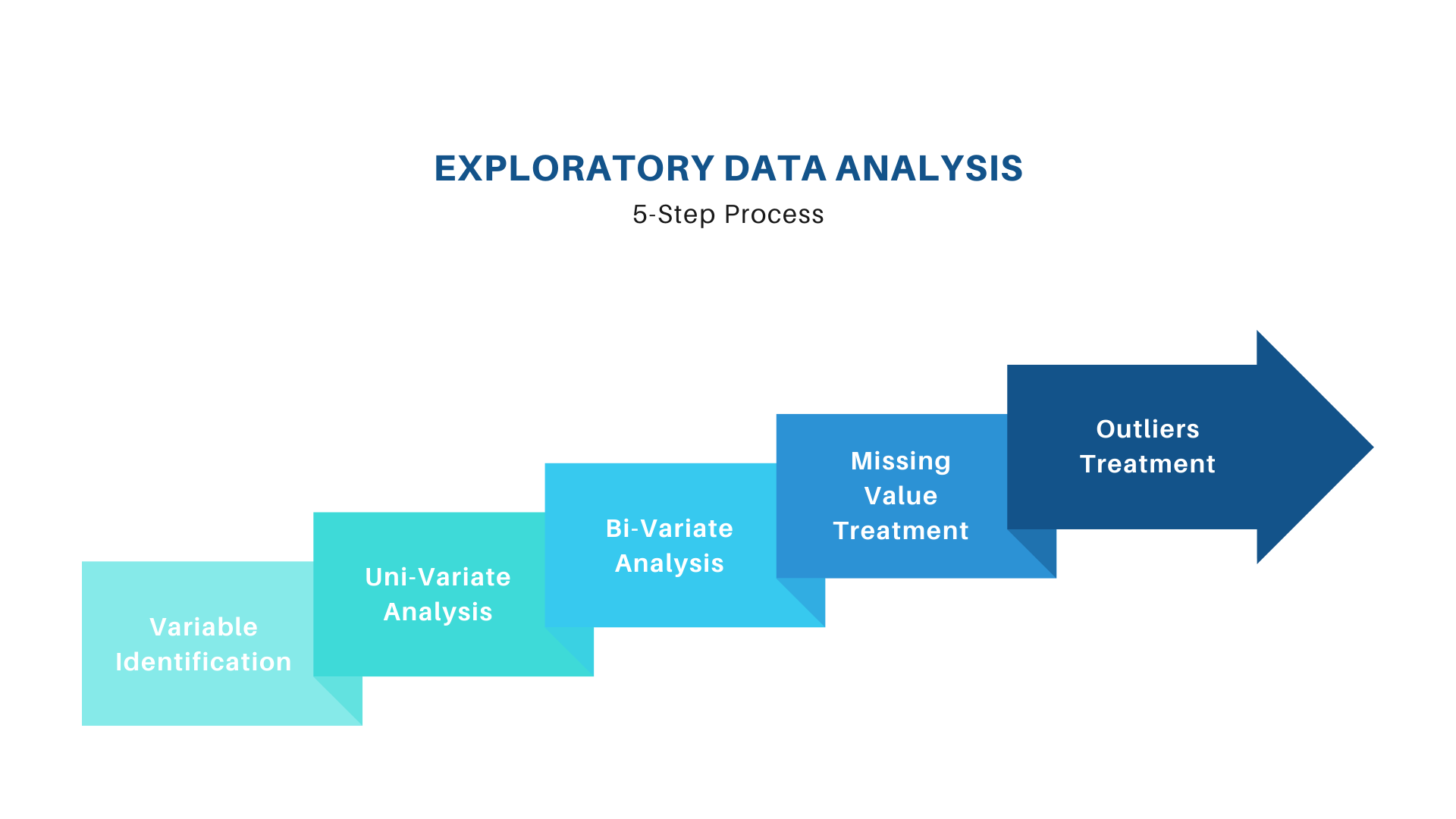


**Customer Segmentation Methodology:**

* Customer segmentation is a crucial stratFor business to brtter understand and target their diverse cystomer base.
* Customer segmentation is a crucial strategy for businesses to better understand and target their diverse customer base

**Exploratory Data Analysis:**

* Summary statistics
* Data virualization (Histograms,scatter plots,etc.)
* Key insights from EDA.



## **Data preparation**

import pandas as pd  
import numpy as np  
import matplotlib as mpl  
import matplotlib.pyplot as plt  
import seaborn as sns  
import datetime, nltk, warnings  
import matplotlib.cm as cm  
import itertools  
from pathlib import Path

from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_samples, silhouette\_score  
from sklearn import preprocessing, model\_selection, metrics, feature\_selection

from sklearn.model\_selection import GridSearchCV, learning\_curve  
from sklearn.svm import SVC  
from sklearn.metrics import confusion\_matrix  
from sklearn import neighbors, linear\_model, svm, tree, ensemble  
from wordcloud import WordCloud, STOPWORDS  
from sklearn.ensemble import AdaBoostClassifier  
from sklearn.decomposition import PCA  
from IPython.display import display, HTML  
import plotly.graph\_objs as go  
from plotly.offline import init\_notebook\_mode,iplot  
init\_notebook\_mode(connected=True)  
warnings.filterwarnings("ignore")  
plt.rcParams["patch.force\_edgecolor"] = True  
plt.style.use('fivethirtyeight')  
mpl.rc('patch', edgecolor = 'dimgray', linewidth=1)  
%matplotlib inline

df\_initial = pd.read\_csv('../input/data.csv',encoding="ISO-8859-1",  
 dtype={'CustomerID': str,'InvoiceID': str})  
print('Dataframe dimensions:', df\_initial.shape)  
#\_\_\_\_\_\_  
df\_initial['InvoiceDate'] = pd.to\_datetime(df\_initial['InvoiceDate'])

tab\_info=pd.DataFrame(df\_initial.dtypes).T.rename(index={0:'column type'})  
tab\_info=tab\_info.append(pd.DataFrame(df\_initial.isnull().sum()).T.rename(index={0:'null values (nb)'}))  
tab\_info=tab\_info.append(pd.DataFrame(df\_initial.isnull().sum()/df\_initial.shape[0]\*100).T.  
 rename(index={0:'null values (%)'}))  
display(tab\_info)

display(df\_initial[:5])

Dataframe dimensions: (541909, 8)

**Out:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |  |
| column type | object | object | object | int64 | datetime64[ns] | float64 | object | object |
| null values (nb) | 0 | 0 | 1454 | 0 | 0 | 0 | 135080 | 0 |
| null values (%) | 0 | 0 | 0.268311 | 0 | 0 | 0 | 24.9267 | 0 |

**Exploring the content of variables:**

data = dict(type='choropleth',  
locations = countries.index,  
locationmode = 'country names', z = countries,  
text = countries.index, colorbar = {'title':'Order nb.'},  
colorscale=[[0, 'rgb(224,255,255)'],  
 [0.01, 'rgb(166,206,227)'], [0.02, 'rgb(31,120,180)'],  
 [0.03, 'rgb(178,223,138)'], [0.05, 'rgb(51,160,44)'],  
 [0.10, 'rgb(251,154,153)'], [0.20, 'rgb(255,255,0)'],  
 [1, 'rgb(227,26,28)']],   
reversescale = False)

layout = dict(title='Number of orders per country',  
geo = dict(showframe = True, projection={'type':'mercator'}))

choromap = go.Figure(data = [data], layout = layout)  
iplot(choromap, validate=False)

**Out:**

**products, transactions, customers,**

**quantity, 3684, 22190, 4372**

### **Customers and products:**

temp = df\_initial.groupby(by=['CustomerID', 'InvoiceNo'], as\_index=False)['InvoiceDate'].count()  
nb\_products\_per\_basket = temp.rename(columns = {'InvoiceDate':'Number of products'})  
nb\_products\_per\_basket[:10].sort\_values('CustomerID')

**Out:**

|  |  |  |  |
| --- | --- | --- | --- |
| CustomerID | InvoiceNo | Number of products |  |
| 0 | 12346 | 541431 | 1 |
| 1 | 12346 | C541433 | 1 |
| 2 | 12347 | 537626 | 31 |
| 3 | 12347 | 542237 | 29 |
| 4 | 12347 | 549222 | 24 |
| 5 | 12347 | 556201 | 18 |

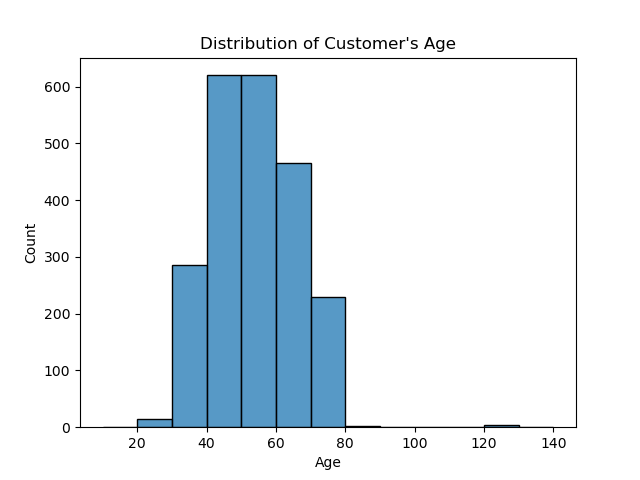
import pandas as pd  
import seaborn as sns  
import matplotlib.pyplot as plt  
import plotly.express as px  
import numpy as np  
from scipy.stats import iqr  
from sklearn.preprocessing import StandardScaler  
from sklearn.cluster import KMeans  
  
  
df = pd.read\_csv("data/marketing\_campaign.csv", sep="\t")  
df.head()

df["TotalAmountSpent"] = df["MntFishProducts"] + df["MntFruits"] + df["MntGoldProds"] + df["MntSweetProducts"] + df["MntMeatProducts"] + df["MntWines"]

**Univariate analysis**

* Univariate analysis entails evaluating a single feature in order to get insights about it. So, the initial step in performing EDA is to undertake univariate analysis, which includes evaluating descriptive or summary statistics about the feature.

sns.histplot(data=df, x="Age", bins = list(range(10, 150, 10)))  
plt.title("Distribution of Customer's Age")



**Bivariate Analysis**

* After you've performed univariate analysis on all your feature of interest, the next step is to perform bivariate analysis. This involves comparing two attributes at the same time.

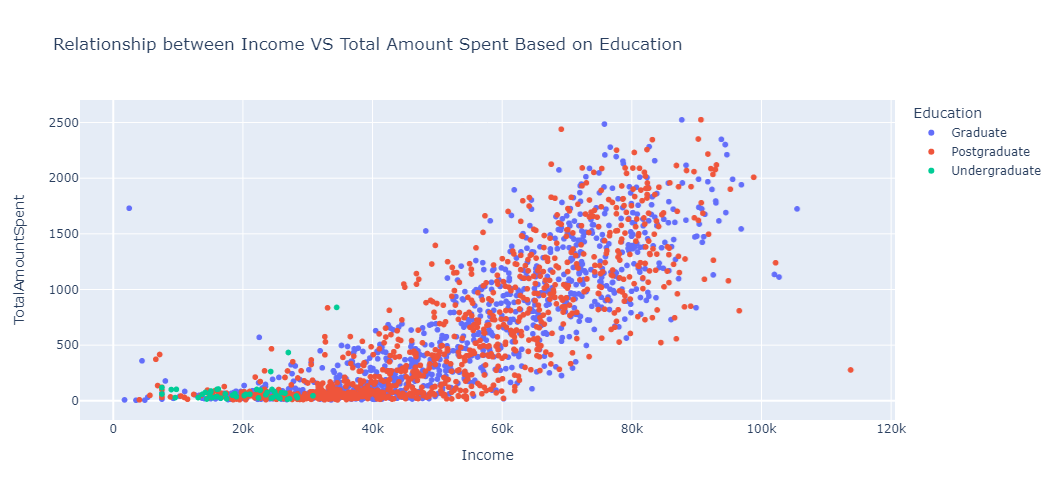
fig = px.scatter(data\_frame=df\_cut, x="Income",  
 y="TotalAmountSpent",  
 title="Relationship Between Customer's Income and Total Amount Spent",  
 height=500,  
 color\_discrete\_sequence = px.colors.qualitative.G10[1:])  
fig.show()



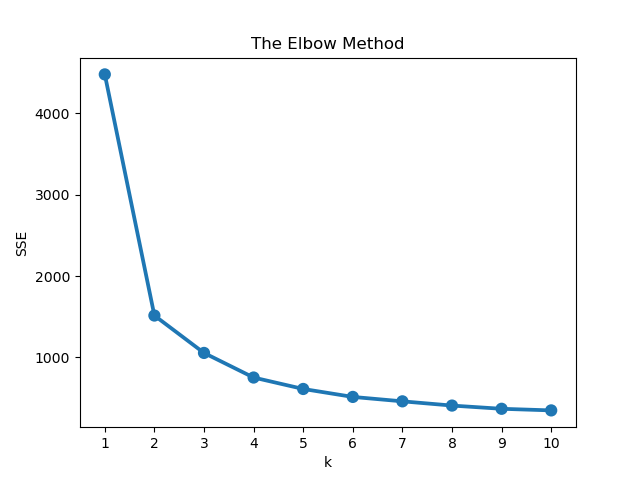
**Multivariate Analysis**

* After you've completed univariate (analysis of single feature) and bivariate (analysis of two features) analysis, the last phase of EDA is to perform Multivariate Analysis

fig = px.scatter(  
 data\_frame=df\_cut,  
 x = "Income",  
 y= "TotalAmountSpent",  
 title = "Relationship between Income VS Total Amount Spent Based on Education",  
 color = "Education",  
 height=500  
)  
fig.show()



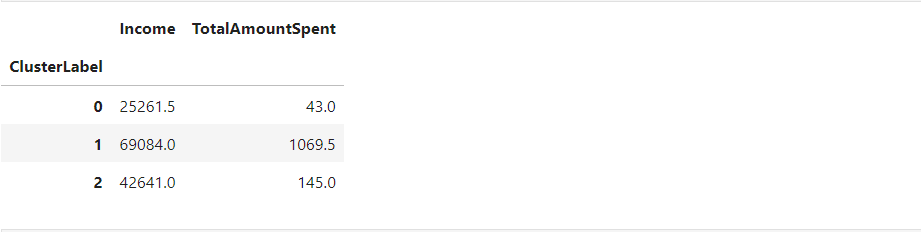
errors = []  
for k in range(1, 11):  
 model = KMeans(n\_clusters=k, random\_state=42)  
 model.fit(df\_scaled)  
 error.append(model.inertia\_)  
   
   
plt.title('The Elbow Method')  
plt.xlabel('k'); plt.ylabel('Error of Cluster')  
sns.pointplot(x=list(range(1, 11), y=errors)  
plt.show()



**How to Interpret the Cluster Result:**

* Now that we've built the model, the next thing will be to interpret the result from each cluster.
* There are numerous way you can summarize the results of your cluster depending on what you want to achieve.

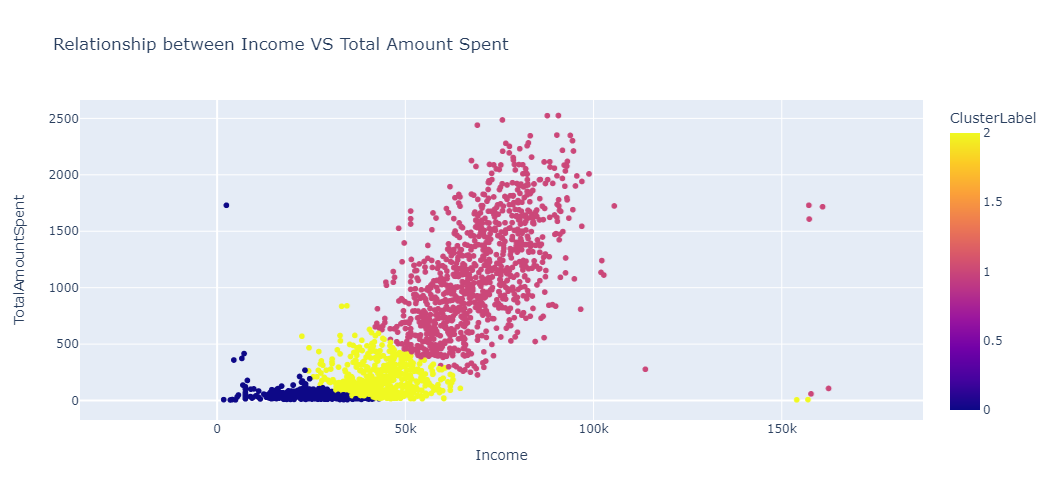
data.groupby("ClusterLabel")[["Income", "TotalAmountSpent"]].median()



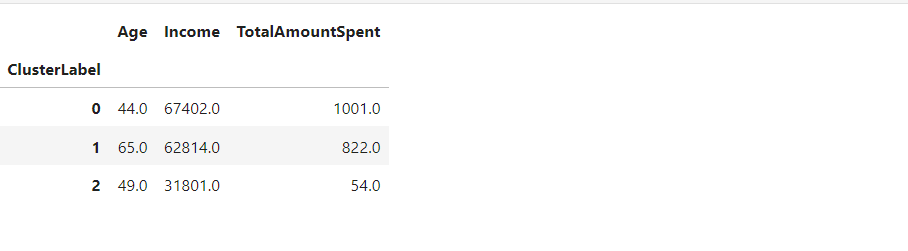
We can see that there is a trend within the clusters:

* Cluster 0 translates to customers who earn less and spend less.
* Cluster 1 represent customers that earn more and spend more.
* Cluster 2 represents customers that earn moderate and spend moderate.

fig = px.scatter(  
 data\_frame=data,  
 x = "Income",  
 y= "TotalAmountSpent",  
 title = "Relationship between Income VS Total Amount Spent",  
 color = "ClusterLabel",  
 height=500  
)  
fig.show()



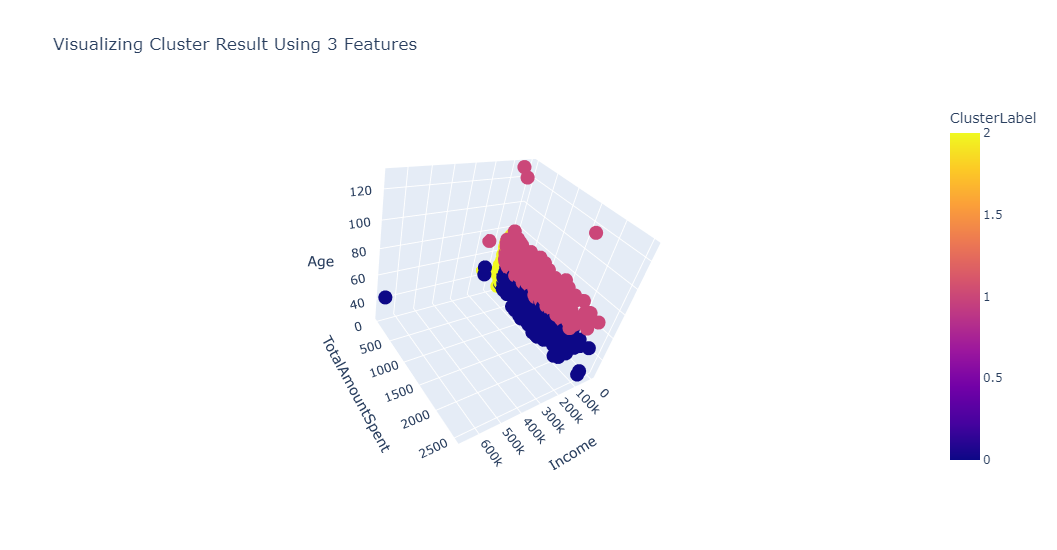
data = df[["Age", "Income", "TotalAmountSpent"]]  
df\_log = np.log(data)  
std\_scaler = StandardScaler()  
df\_scaled = std\_scaler.fit\_transform(df\_log)model = KMeans(n\_clusters=3, random\_state=42)  
model.fit(df\_scaled)  
  
data = data.assign(ClusterLabel= model.labels\_)  
  
result = df\_result.groupby("ClusterLabel").agg({"Age":"mean", "Income":"median", "TotalAmountSpent":"median"}).round()



We can see from the above summary that:

* Cluster 0 depicts young customers that earn a lot and also spend a lot.
* Cluster 1 translates to older customers that earn a lot and also spend a lot.
* Cluster 2 depicts young customers that earn less and also spend less.

fig = px.scatter\_3d(data\_frame=data, x="Income",   
 y="TotalAmountSpent", z="Age", color="ClusterLabel", height=550,  
 title = "Visualizing Cluster Result Using 3 Features")  
fig.show()



# **Conclusion:**

* customer segmentation is a powerful strategy that enables businesses to gain a deeper understanding of their diverse customer base and tailor their marketing, product development, and customer service efforts to meet the specific needs of different customer groups.