**Customer Segmentation**   
 **using Data Science**

**Team members:**

1) AHMED BASHEER B (410621243003)

2) REYASH AHAMED K (410621243037)

3) SRIDHAR C (410621243046)

4) FURQQANDEEN R (410621243005)

5) GOPALAKRISHNAN M(410621243007)

**Project Title:** Customer segmentation using

Data Science

**Phase 5: Project Documentation & Submission**

**Topic:**

In this section you will document the complete project and prepare it for submission.



**Customer segmentation using Data Science**

**Introduction:**

* Customer segmentation involves implementing data science methods to divide the customer base into smaller groups based on certain characteristics.
* It assists marketing managers in better understanding their customer’s preferences and presenting them with better targeted advertisements.
* A method of analysing a client base and grouping customers into categories or segments which share particular attributes.
* Tailor marketing efforts: Customize messages and promotions to suit the specific needs and preferences of each customer group, improving the effectiveness of marketing campaigns.
* Optimize product development: Develop products and services that cater to the unique demands of different customer segments, enhancing overall product-market fit.
* Enhance customer experience: Personalize interactions and recommendations, leading to higher customer satisfaction and loyalty.
* Improve resource allocation: Allocate resources more efficiently by focusing on segments with the most potential for growth and profitability.
* Overall, customer segmentation through data science is an essential strategy for modern businesses to drive growth, improve customer satisfaction, and remain competitive in the ever-evolving marketplace.

Dataset Link:([**https://www.kaggle.com/datasets/akram24/mall-customers**](https://www.kaggle.com/datasets/akram24/mall-customers))

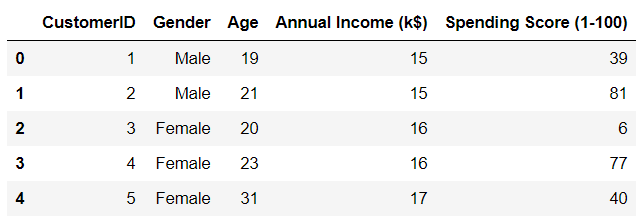
**Data Collection :**

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

**program:**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
import sea  
from kneed import KneeLocator  
from sklearn.datasets import make\_blobs  
from sklearn.cluster import KMeans  
from sklearn.metrics import silhouette\_score  
from sklearn.preprocessing import StandardScaler  
from sklearn.decomposition import PCA  
from mpl\_toolkits.mplot3d import Axes3D  
df =pd.read\_csv('**E:\Customer Segmentation.csv’**)

df.head()



**Necessary step to follow:**

**1)Import Libraries:**

**Start by importing the necessary lidraries;**

**Program:**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_test\_split

from sklearn.preprocessing import StandardScaler

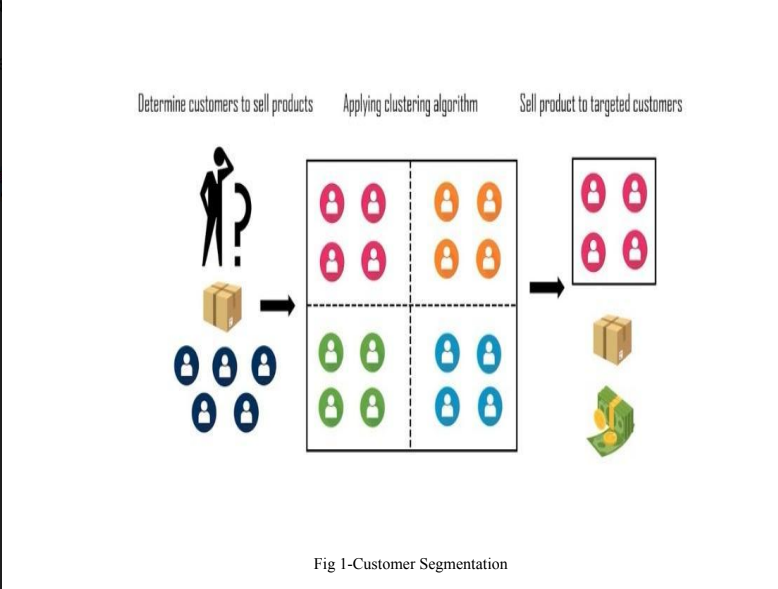
**2)Load the Dataset:**

**Load your dataset into a Pandas DataFrame. You can typically find house price datasets in CSV format,but you can adapt this code to other formats as needed.**

**Program:**

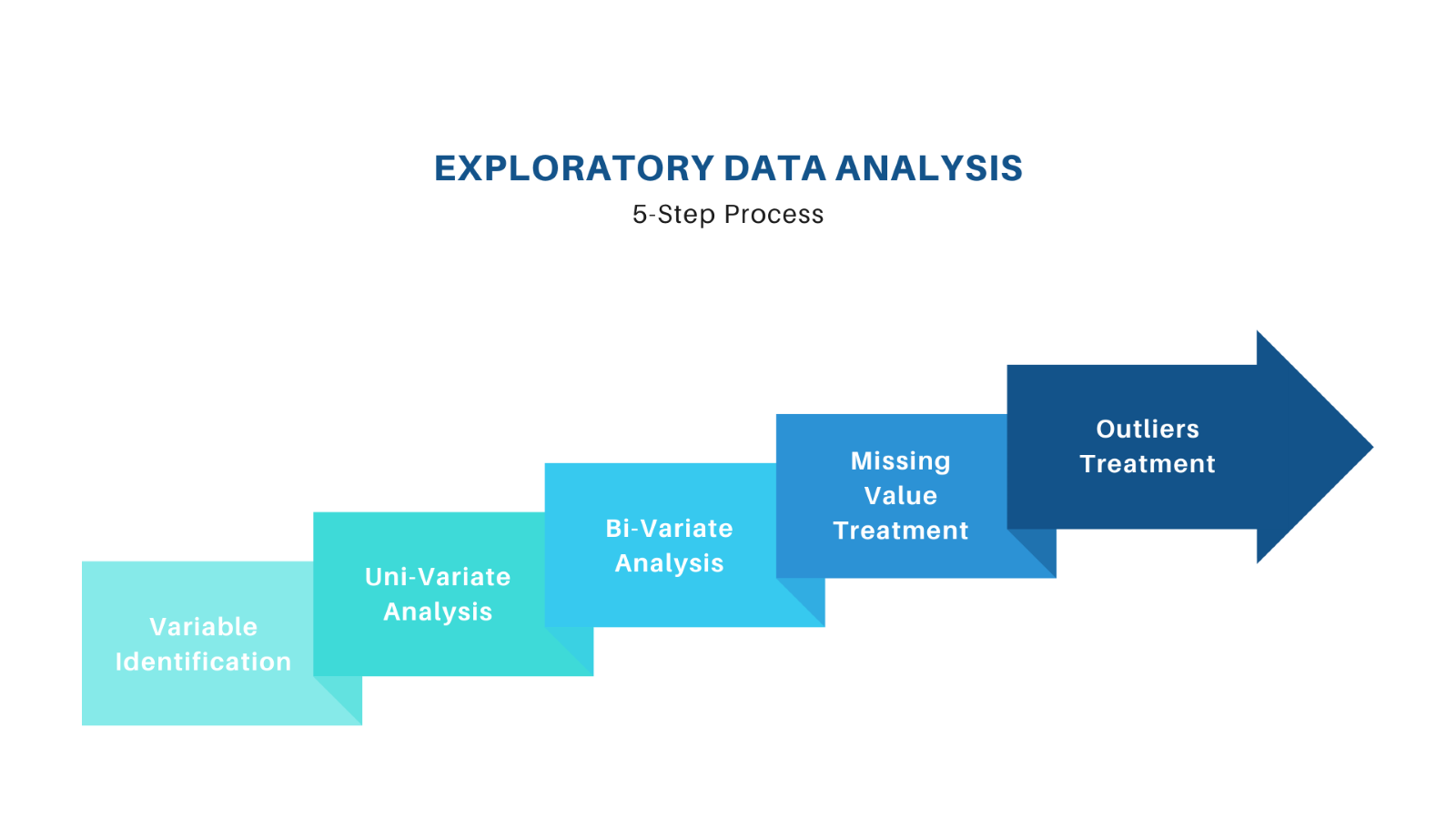
**df =pd.read\_csv(‘E:\Customer Segmentation.csv’)**

**pd.read()**



**3)Exploratory Data Analysis:**

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

  
**Program:**

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"]

**4)Pre processing:**

* Data Segmentation is the process of taking the data you hold and dividing it up and grouping similar data together based on the chosen parameters so that you can use it more efficiently within marketing and operations.
* Determination of the Need of the Segment.
* Identification of the Segment.



**5) Split the Data:**

* Split your dataset into training and testing sets.
* This helps you evaluateyour model's performance later.

Program:

X = df.drop('price', axis=1)

# Features y = df['price'] # Target variable

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Importance of loading and processing dataset:**

* Loading and preprocessing the dataset is an important first step in building any applied data science model.
* However, it is especially important for Customer segmentation using data science models, as Customer segmentation datasets are often complex and noisy.
* By loading and preprocessing the dataset, we can ensure that the machine learning algorithm is able to learn from the data effectively and accurately

**Loading the dataset:**

* Loading the dataset using machine learning is the process of bringing the data into the machine learning environment so that it can be used to train and evaluate a model.
* The specific steps involved in loading the dataset will vary depending on the machine learning library or framework that is being used.
* However, there are some general steps that are common to most machine learning frameworks:

**a.Identify the dataset:**

* The first step is to identify the dataset that you want to load. This dataset may be stored in a local file, in a database, or in a cloud storage service.

**b.Load the dataset:**

* Once you have identified the dataset, you need to load it into the machine learning environment.
* This may involve using a built-in function in the machine learning library, or it may involve writing your own code.

**c.Preprocess the dataset:**

* Once the dataset is loaded into the machine learning environment, you may need to preprocess it before you can start training and evaluating your model.
* This may involve cleaning the data, transforming the data into a suitable format, and splitting the data into training and test sets.
* Preprocess the dataset Load the dataset Identify the dataset Loading the dataset Here, how to load a dataset using machine learning in Python

Program:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2\_score, mean\_absolute\_error,mean\_squared\_error

from sklearn.linear\_model import LinearRegression

from sklearn.linear\_model import Lasso

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

import xgboost as xg

%matplotlib inline

import warnings

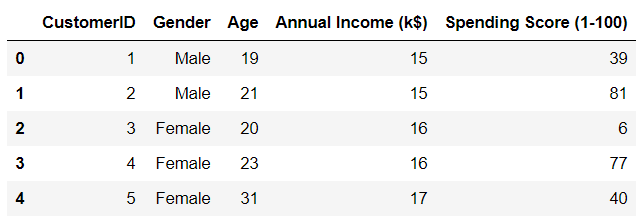
warnings.filterwarnings("ignore") /opt/conda/lib/python3.10/site-packages/scipy/\_\_init\_\_.py:146: UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required forthis version of SciPy (detected version 1.23.5

warnings.warn(f"A NumPy version >={np\_minversion} and<{np\_maxversion}"

**Loading Dataset:**

dataset = pd.read\_csv('E: **\Customer Segmentation**.csv')

**Output:**



**Customer Segmentation Methodology:**

* Customer Segmentation used to random forest,cluster,analysis,Regression,classification and K Means Clustering,Machine learning algorithms are used.
* Customer segmentation is a crucial start for business to better understand and target their diverse customer base.
* Customer segmentation is a crucial strategy for businesses to better understand and target their diverse customer base

## **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')

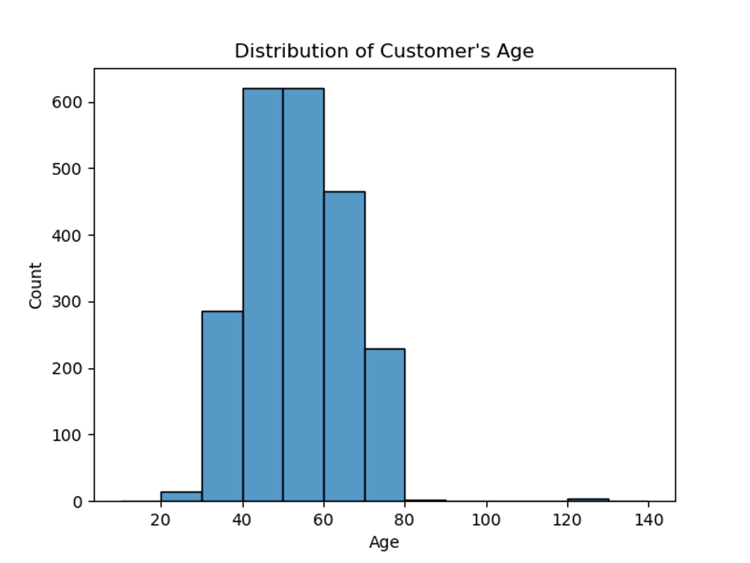
**Out:**

|  |  |  |  |
| --- | --- | --- | --- |
| 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 |

**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.

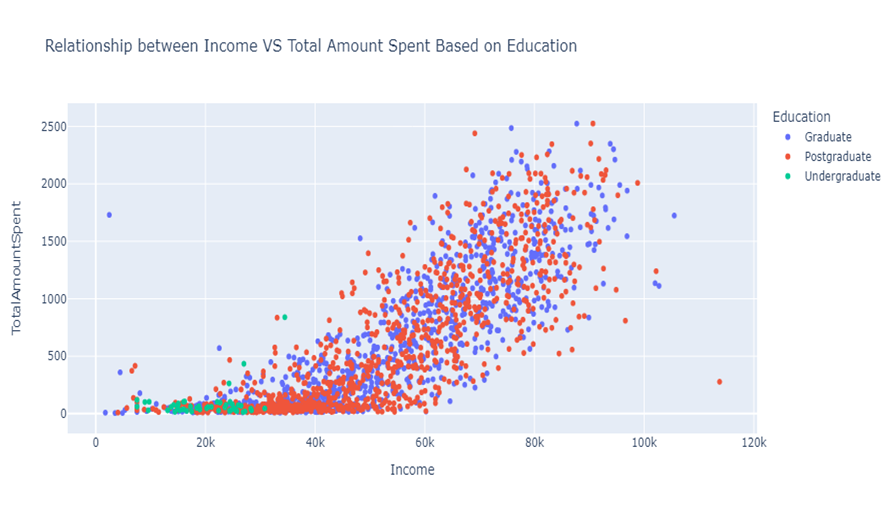
fig = px.scatter(data\_frame=df\_cut, x="Income",  
 y="TotalAmountSpent",title="Relationship Between Customer's Income and Total Amount Spent", height=500,  
 color\_discrete\_sequence = px.colors.qualitative.G10[1:])  
 fig.show()



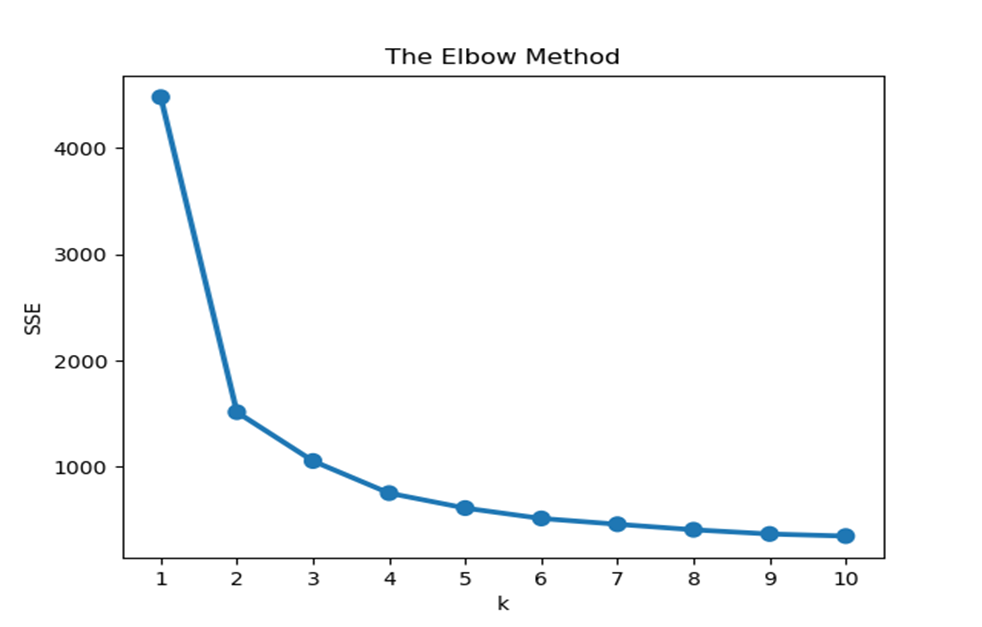
**Multivariate Analysis:**

* 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()



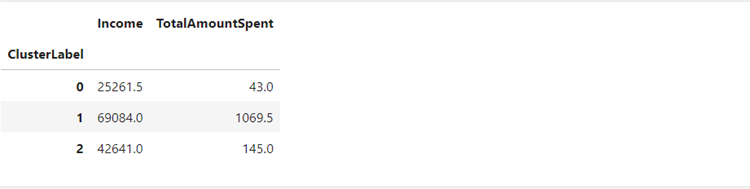
errors = []  
 for k in range(1, 11):  
 model = KMeans(n\_clusters=k, random\_state=42)  
 model.fit(df\_scaled)  
 error.append(model.inertia\_)  
   
   
 plt.title('The Elbow Method')  
 plt.xlabel('k'); plt.ylabel('Error of Cluster')  
 sns.pointplot(x=list(range(1, 11), y=errors)  
 plt.show()



**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()



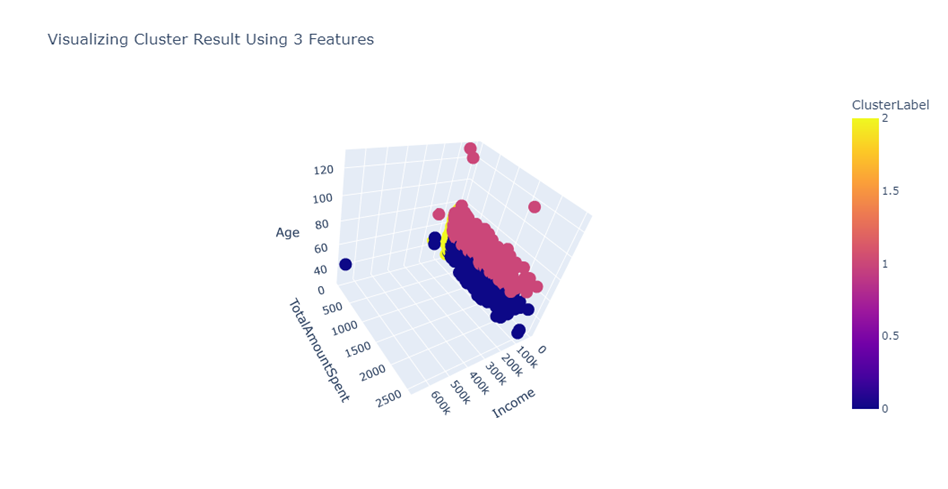
data = df[["Age", "Income", "TotalAmountSpent"]]  
 df\_log = np.log(data)  
 std\_scaler = StandardScaler()  
 df\_scaled = std\_scaler.fit\_transform(df\_log)model = KMeans(n\_clusters=3, random\_state=42)  
 model.fit(df\_scaled)  
   
data = data.assign(ClusterLabel= model.labels\_)  
   
result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



## **Classification of customers:**

* In this part, the objective will be to adjust a classifier that will classify consumers in the different client categories that were established in the previous section.
* The objective is to make this classification possible at the first visit.
* To fulfill this objective, I will test several classifiers implemented in scikit-learn

**Program:**

class Class\_Fit(object):  
 def \_\_init\_\_(self, clf, params=None):  
 if params:   
 self.clf = clf(\*\*params)  
 else:  
 self.clf = clf()  
  
 def train(self, x\_train, y\_train):  
 self.clf.fit(x\_train, y\_train)  
  
 def predict(self, x):  
 return self.clf.predict(x)  
   
 def grid\_search(self, parameters, Kfold):  
 self.grid = GridSearchCV(estimator = self.clf, param\_grid = parameters, cv = Kfold)  
   
 def grid\_fit(self, X, Y):  
 self.grid.fit(X, Y)  
   
 def grid\_predict(self, X, Y):  
 self.predictions = self.grid.predict(X)  
 print("Precision: {:.2f} % ".format(100\*metrics.accuracy\_score(Y, self.predictions)))

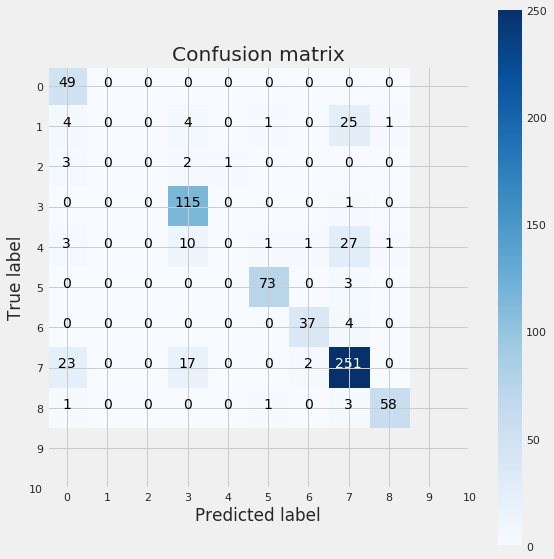
olumns = ['mean', 'categ\_0', 'categ\_1', 'categ\_2', 'categ\_3', 'categ\_4' ]  
X = selected\_customers[columns]  
Y = selected\_customers['cluster']

X\_train, X\_test, Y\_train, Y\_test = model\_selection.train\_test\_split(X, Y, train\_size = 0.8)

**Confusion matrix:**

The accuracy of the results seems to be correct.

def plot\_confusion\_matrix(cm, classes, normalize=False, title='Confusion matrix', cmap=plt.cm.Blues):  
 if normalize:  
 cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]  
 print("Normalized confusion matrix")  
 else:  
 print('Confusion matrix, without normalization')  
 plt.imshow(cm, interpolation='nearest', cmap=cmap)  
 plt.title(title)  
 plt.colorbar()  
 tick\_marks = np.arange(len(classes))  
 plt.xticks(tick\_marks, classes, rotation=0)  
 plt.yticks(tick\_marks, classes)  
 fmt = '.2f' if normalize else 'd'  
 thresh = cm.max() / 2.  
 for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):  
 plt.text(j, i, format(cm[i, j], fmt),  
 horizontalalignment="center",  
 color="white" if cm[i, j] > thresh else "black")  
 plt.tight\_layout()  
 plt.ylabel('True label')  
 plt.xlabel('Predicted label')

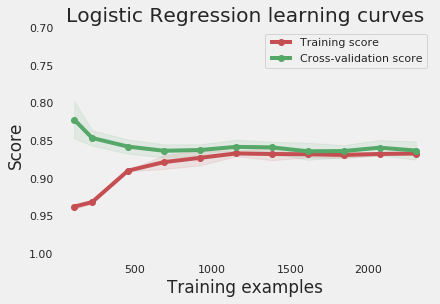
class\_names = [i for i in range(11)]  
cnf\_matrix = confusion\_matrix(Y\_test, svc.predictions)   
np.set\_printoptions(precision=2)  
plt.figure(figsize = (8,8))  
plot\_confusion\_matrix(cnf\_matrix, classes=class\_names, normalize = False, title='Confusion matrix')

### **Logistic Regression:**

* I now consider the logistic regression classifier.
* As before, I create an instance of the Class\_Fit class, adjust the model on the training data and see how the predictions compare to the real values:

**Program:**

lr = Class\_Fit(clf = linear\_model.LogisticRegression)  
lr.grid\_search(parameters = [{'C':np.logspace(-2,2,20)}], Kfold = 5)  
lr.grid\_fit(X = X\_train, Y = Y\_train)  
lr.grid\_predict(X\_test, Y\_test)

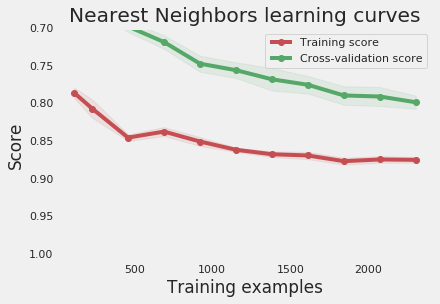
g = plot\_learning\_curve(lr.grid.best\_estimator\_, "Logistic Regression learning curves", X\_train, Y\_train,ylim = [1.01, 0.7], cv = 5,train\_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8,0.9, 1])

### **k-Nearest Neighbors:**

* In statistics, the k-nearest neighbors algorithm is a non-parametric supervised learning method first developed by Evelyn Fix and Joseph Hodges in 1951, and later expanded by Thomas Cover.
* It is used for classification and regression. In both cases, the input consists of the k closest training examples in a data set.

**Program:**  
knn = Class\_Fit(clf = neighbors.KNeighborsClassifier)  
knn.grid\_search(parameters = [{'n\_neighbors': np.arange(1,50,1)}], Kfold = 5)  
knn.grid\_fit(X = X\_train, Y = Y\_train)  
knn.grid\_predict(X\_test, Y\_test)

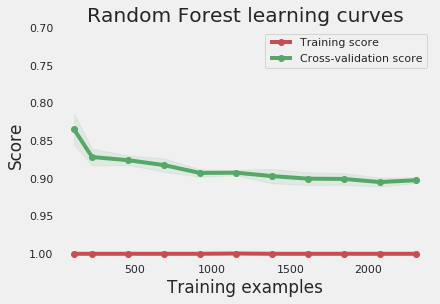
g = plot\_learning\_curve(knn.grid.best\_estimator\_, "Nearest Neighbors learning curves", X\_train, Y\_train,ylim = [1.01, 0.7], cv = 5, train\_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



### **Random Forest:**

rf = Class\_Fit(clf = ensemble.RandomForestClassifier)  
param\_grid = {'criterion' : ['entropy', 'gini'], 'n\_estimators' : [20, 40, 60, 80, 100],'max\_features' :['sqrt', 'log2']}  
rf.grid\_search(parameters = param\_grid, Kfold = 5)  
rf.grid\_fit(X = X\_train, Y = Y\_train)  
rf.grid\_predict(X\_test, Y\_test)  
Precision: 89.61 %

g = plot\_learning\_curve(rf.grid.best\_estimator\_, "Random Forest learning curves", X\_train, Y\_train,ylim = [1.01, 0.7], cv = 5,train\_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])

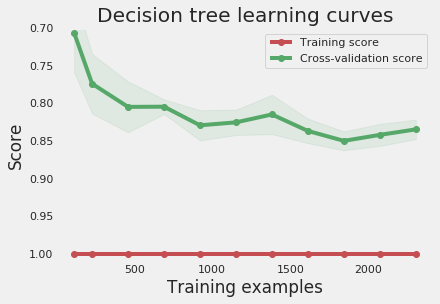


**Decision Tree:**

* A decision tree is a type of supervised machine learning used to categorize or make predictions based on how a previous set of questions were answered.
* The model is a form of supervised learning, meaning that the model is trained and tested on a set of data that contains the desired categorization.

Program:

tr = Class\_Fit(clf = tree.DecisionTreeClassifier)  
tr.grid\_search(parameters = [{'criterion' : ['entropy', 'gini'], 'max\_features' :['sqrt', 'log2']}], Kfold = 5)  
tr.grid\_fit(X = X\_train, Y = Y\_train)  
tr.grid\_predict(X\_test, Y\_test)



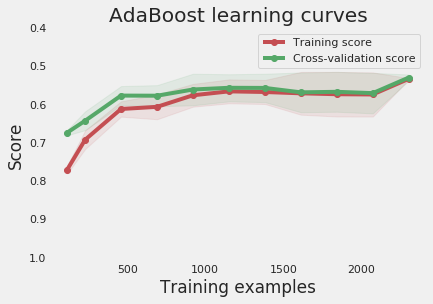
### **AdaBoost Classifier:**

* AdaBoost, short for Adaptive Boosting, is a statistical classification meta-algorithm formulated by Yoav Freund and Robert Schapire in 1995, who won the 2003 Gödel Prize for their work.
* It can be used in conjunction with many other types of learning algorithms to improve performance.

**Program:**

ada = Class\_Fit(clf = AdaBoostClassifier)  
param\_grid = {'n\_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}  
ada.grid\_search(parameters = param\_grid, Kfold = 5)  
ada.grid\_fit(X = X\_train, Y = Y\_train)  
ada.grid\_predict(X\_test, Y\_test

g = plot\_learning\_curve(ada.grid.best\_estimator\_, "AdaBoost learning curves", X\_train, Y\_train,  
 ylim = [1.01, 0.4], cv = 5,   
 train\_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



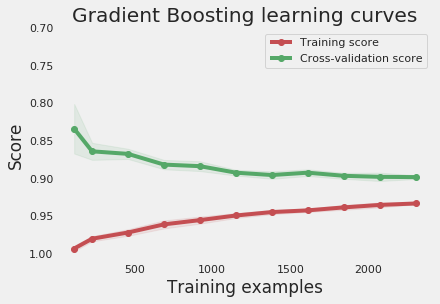
### **Gradient Boosting Classifier:**

* Gradient Boosting is a functional gradient algorithm that repeatedly selects a function that leads in the direction of a weak hypothesis or negative gradient so that it can minimize a loss function.
* Gradient boosting classifier combines several weak learning models to produce a powerful predicting model.

**Program:**

gb = Class\_Fit(clf = ensemble.GradientBoostingClassifier)  
param\_grid = {'n\_estimators' : [10, 20, 30, 40, 50, 60, 70, 80, 90, 100]}  
gb.grid\_search(parameters = param\_grid, Kfold = 5)  
gb.grid\_fit(X = X\_train, Y = Y\_train)  
gb.grid\_predict(X\_test, Y\_test)

g = plot\_learning\_curve(gb.grid.best\_estimator\_, "Gradient Boosting learning curves", X\_train, Y\_train,  
 ylim = [1.01, 0.7], cv = 5,   
 train\_sizes = [0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1])



## **Testing predictions:**

* In the previous section, a few classifiers were trained in order to categorize customers. Until that point, the whole analysis was based on the data of the first 10 months.
* In this section, I test the model the last two months of the dataset, that has been stored in the set\_test dataframe

**Program:**

basket\_price = set\_test.copy(deep = True)

transactions\_per\_user=basket\_price.groupby(by=['CustomerID'])['Basket Price'].agg(['count','min','max','mean','sum'])  
for i in range(5):  
 col = 'categ\_{}'.format(i)  
 transactions\_per\_user.loc[:,col] = basket\_price.groupby(by=['CustomerID'])[col].sum() /\  
 transactions\_per\_user['sum']\*100  
  
transactions\_per\_user.reset\_index(drop = False, inplace = True)  
basket\_price.groupby(by=['CustomerID'])['categ\_0'].sum()

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CustomerID | count | min | max | mean | sum | categ\_0 | categ\_1 | categ\_2 | categ\_3 | categ\_4 |  |
| 0 | 12347 | 10 | 224.82 | 1294.32 | 759.57 | 7595.70 | 20.017905 | 24.271627 | 12.696657 | 37.379043 | 5.634767 |
| 1 | 12349 | 5 | 1757.55 | 1757.55 | 1757.55 | 8787.75 | 26.506216 | 10.713778 | 4.513101 | 37.877728 | 20.389178 |
| 2 | 12352 | 5 | 311.73 | 311.73 | 311.73 | 1558.65 | 34.420813 | 7.217785 | 6.672441 | 34.398358 | 17.290604 |
| 3 | 12356 | 5 | 58.35 | 58.35 | 58.35 | 291.75 | 0.000000 | 0.000000 | 0.000000 | 100.000000 | 0.000000 |
| 4 | 12357 | 5 | 6207.67 | 6207.67 | 6207.67 | 31038.35 | 18.475531 | 28.350089 | 5.089832 | 22.895547 | 25.189000 |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Support Vector Machine  
Precision: 65.93 %   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Logostic Regression  
Precision: 71.34 %   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
k-Nearest Neighbors  
Precision: 67.58 %   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Decision Tree  
Precision: 71.38 %   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Random Forest  
Precision: 75.38 %   
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_   
Gradient Boosting  
Precision: 75.23 %

**Advantages of customer segmentation**

Implementing customer segmentation leads to plenty of new business opportunities. You can do a lot of optimization in:

* budgeting,
* product design,
* promotion,
* marketing,
* customer satisfaction.

Let’s discuss these benefits in more depth.

**Budgeting:**

* Nobody likes to invest in campaigns that don’t generate any new customers.
* Most companies don’t have huge marketing budgets, so that money has to be spent right.
* Segmentation enables you to target customers with the highest potential value first, so you get the most out of your marketing budget.

**Product design:**

* Customer segmentation helps you understand what your users need.
* You can identify the most active users/customers, and optimize your application/offer towards their needs.

**Promotion:**

* Properly implemented customer segmentation helps you plan special offers and deals.
* Frequent deals have become a staple of e-commerce and commercial software in the past few years.
* If you reach a customer with just the right offer, at the right time, there’s a huge chance they’re going to buy.
* Customer segmentation will help you tailor your special offers perfectly.

**Marketing:**

* The marketing strategy can be directly improved with segmentation because you can plan personalized marketing campaigns for different customer segments, using the channels that they use the most.

**Customer satisfaction:**

* By studying different customer groups, you learn what they value the most about your company.

This information will help you create personalized products and services that perfectly fit your customers’ preferences.

* In the next section, we’re going to discuss why machine learning for customer segmentation.
* Finding the optimal number of clusters
* Finding the optimal number of clusters is one of the key tasks when implementing a k-means clustering algorithm. It’s worth noting that a k-means clustering model might converge for any value of K, but at the same time, not all values of K will produce the best model.
* For some datasets, data visualization can help understand the optimal number of clusters, but this doesn’t apply to all datasets.
* We have a few methods, such as the elbow method, gap statistic method, and average silhouette method, to assess the optimal number of clusters for a given dataset. We’ll discuss them one by one.
* The elbow method finds the value of the optimal number of clusters using the total within-cluster sum of square values. This represents how spread-apart the generated clusters are from one another.

The stage at this number of clusters is called the elbow of the clustering model. We’ll see that in our case it’s K =5.

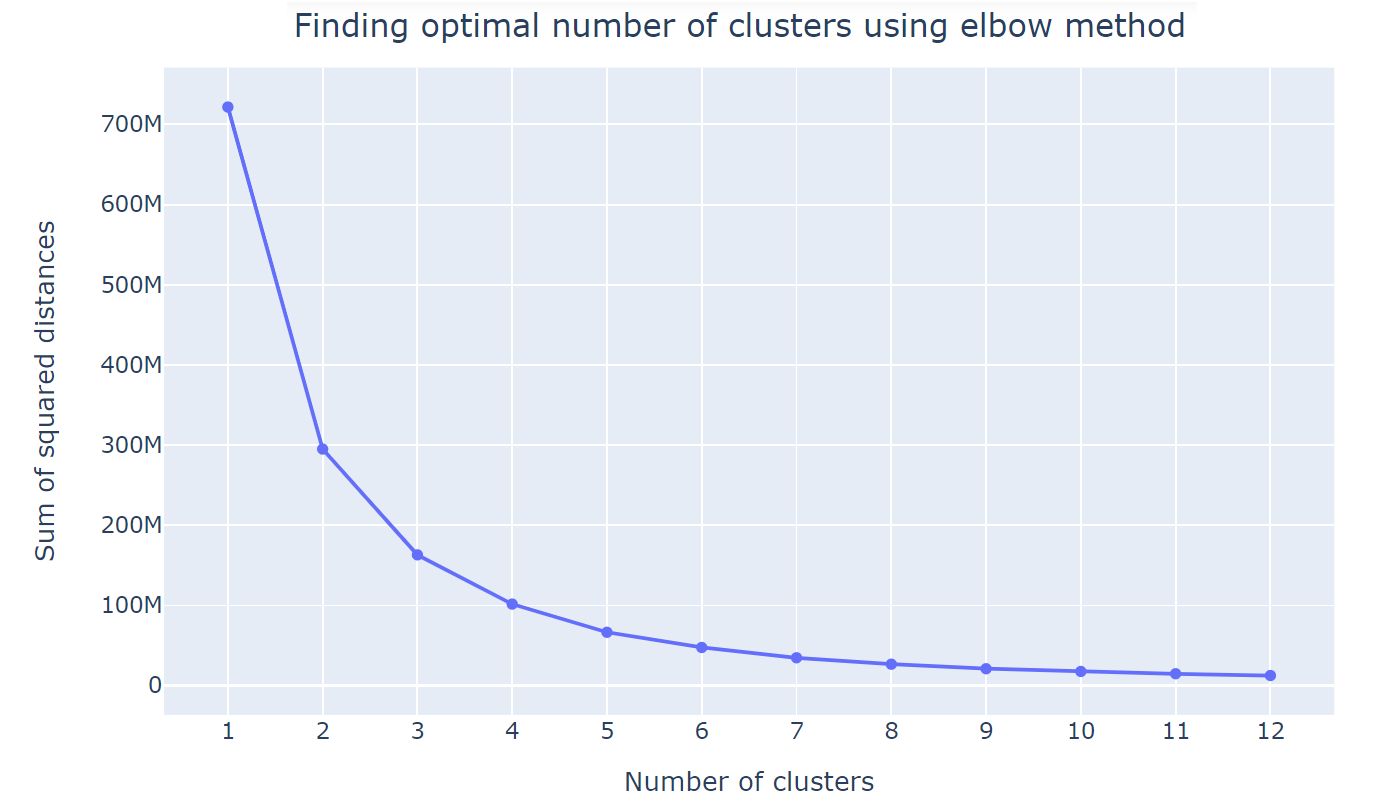
**Program:**

(number of clusters),

data (input data).

def try\_different\_clusters(K, data):  
  
 cluster\_values = list(range(1, K+1))  
 inertias=[]  
  
 for c in cluster\_values:  
 model = KMeans(n\_clusters = c,init='k-means++',max\_iter=400,random\_state=42)  
 model.fit(data)  
 inertias.append(model.inertia\_)  
  
 return inertias  
  
outputs = try\_different\_clusters(12, customersdata[['products\_purchased','complains','money\_spent']])  
distances = pd.DataFrame({"clusters": list(range(1, 13)),"sum of squared distances": outputs})  
  
figure = go.Figure()  
figure.add\_trace(go.Scatter(x=distances["clusters"], y=distances["sum of squared distances"]))  
  
figure.update\_layout(xaxis = dict(tick0 = 1,dtick = 1,tickmode = 'linear'),  
 xaxis\_title="Number of clusters",  
 yaxis\_title="Sum of squared distances",  
 title\_text="Finding optimal number of clusters using elbow method")  
figure.show()

We can generate the below plot using the above code. The elbow of the code is at K=5. We have chosen 5 as if we increase the number of clusters to more than 5, there is very small change in the inertia or sum of the squared distance.



**Visualizing customer segments:**

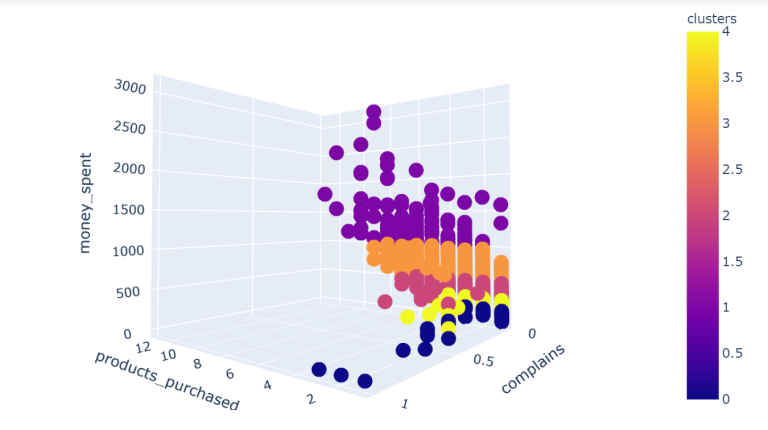
* In this section, we’ll be implementing some code using plotly express. This way we’ll visualize the clusters in three dimensions, formed by our k-means algorithm.
* Plotly express is a library based on plotly that works on several types of datasets and generates highly-styled plots.
* First, let’s add a new column named ‘clusters’ to the existing customer data dataset. This column will be able to tell which customer belongs to what cluster.

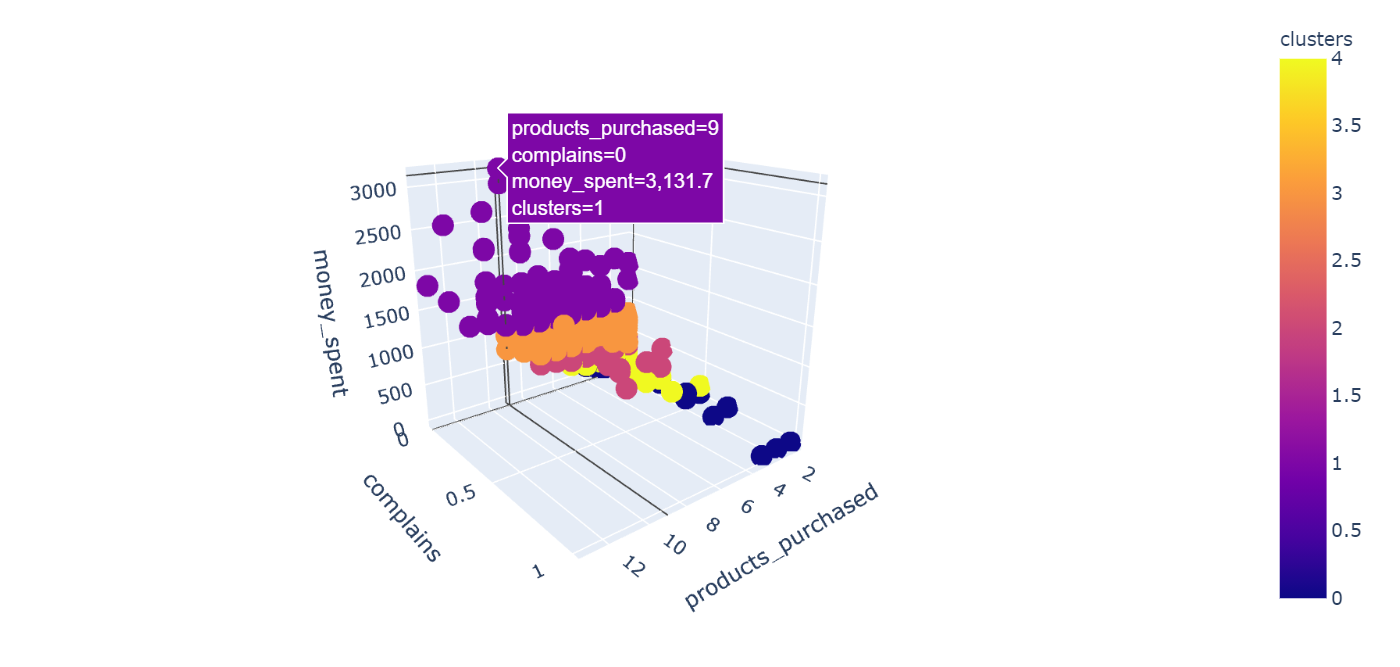
**Program:**

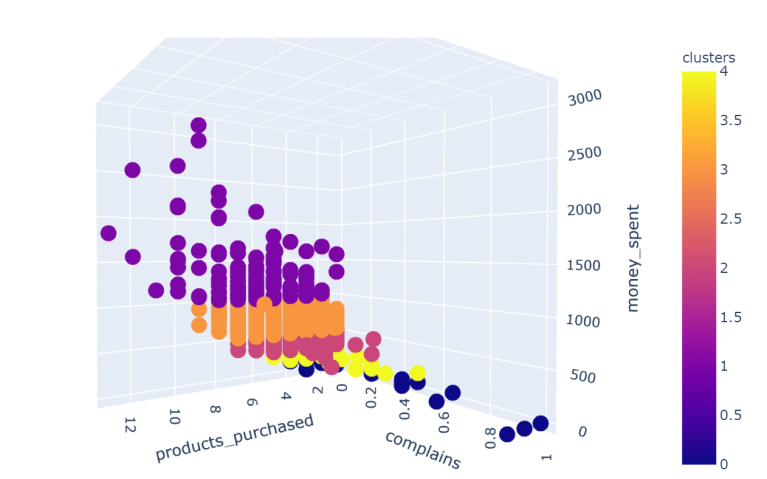
cluster\_centers = kmeans\_model\_new.cluster\_centers\_  
data = np.expm1(cluster\_centers)  
points = np.append(data, cluster\_centers, axis=1)  
points

points = np.append(points, [[0], [1], [2], [3], [4]], axis=1)  
customersdata["clusters"] = kmeans\_model\_new.labels\_

figure = px.scatter\_3d(customersdata,  
 color='clusters',  
 x="products\_purchased",  
 y="complains",  
 z="money\_spent",  
 category\_orders = {"clusters": ["0", "1", "2", "3", "4"]}  
 )  
figure.update\_layout()  
figure.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.
* Understanding the data's structure, characteristics, and anypotential issues through exploratory data analysis (EDA) isessential for informed decision-making.
* With these foundational steps completed, our dataset is now primed for the subsequent stages of building and training a Customer segmentation using data science model.