

# CUSTOMER SEGEMENTATION USING MACHINE LEARNING

## Introduction :

The primal task of Management is to identify potential customers from the rest. This will be simplified with the help of Machine Learning models to classify the customers into segments based on various attributes.

The intervention of Data Science and AI helps the business to build such models to analyze the customers and their products in better decision making, to improve the business process, to formulate better strategies, and to improve the revenue.

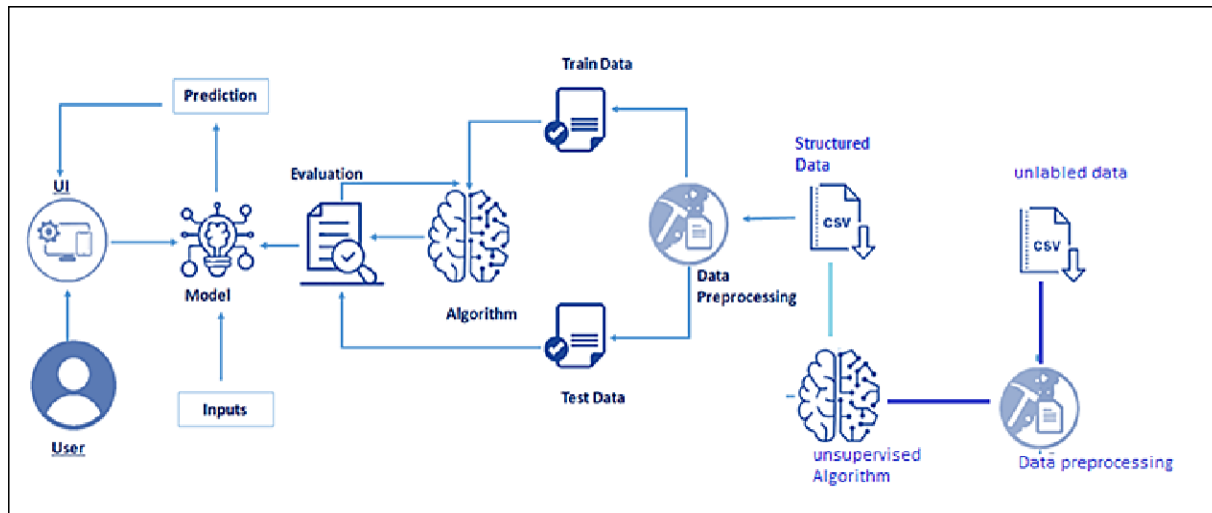
This project deals with understanding and segmenting the customers based on the data.

The Model we built will be able to classify the customer's potentiality in purchasing power.

We will be using classification algorithms such as H-clustering, k-means clustering, Decision tree, Random forest, KNN, and xgboost. We will train and test the data

with these algorithms. From this best model is selected and saved in pkl format. Once the model is saved, we integrate it with the flask application.

## Technical Architecture:



## Pre Requisites :

To complete this project, you must require the following software's, concepts, and packages

- Anaconda navigator:
  - Refer to the link below to download anaconda navigator

<https://www.youtube.com/watch?v=5mDYijMfSzs>

## Prior Knowledge :

- Customer segmentation use case  
<https://www.youtube.com/watch?v=zPJtDohab-g&t=3s>
- Customer segmentation model using machine learning  
[https://www.youtube.com/watch?v=Liff\\_GA74EI](https://www.youtube.com/watch?v=Liff_GA74EI)

Python packages:

Open anaconda prompt as administrator.

- Type "**pip install numpy**" and click enter.
- Type "**pip install pandas**" and click enter.
- Type "**pip install matplotlib**" and click enter.
- Type "**pip install scikit-learn**" and click enter.
- Type "**pip install Flask**" and click enter.

The above steps allow you to install the packages in the anaconda environment.

## **Project Objectives :**

By the end of this project:

1. This project enables the learner to understand the business use case of how and why to segment the customers.
2. You'll be able to understand the unsupervised learning methods such as H-clustering and k-means clustering
3. You'll be able to understand the problem to classify if it is a regression or a classification kind of problem.
4. You will be able to know how to pre-process/clean the data using different data pre-processing techniques.
5. You will be able to analyze or get insights into data through visualization.
6. Applying different algorithms according to a dataset and based on visualization.
7. You will be able to know how to find the accuracy of the model.
8. You will be able to know how to build a web application using the Flask framework.

## ***Project Flow :***

1. User interacts with the UI (User Interface) to enter the input values

2. Entered input values are analyzed by the model which is integrated
3. Once the model analyses the input, the prediction is showcased on the

UI All the activities and tasks involved are listed below

1. Data Collection.
  - a. Collect the dataset or Create the dataset
2. Data Pre-processing.
  - a. Import the Libraries.
  - b. Importing the dataset.
  - c. Checking for Null Values.
  - d. Data Visualization.
  - e. Taking care of Missing Data.
  - f. Feature Scaling.
3. Unsupervised Model Building
  - a. Import the model building Libraries
  - b. Initializing the model
  - c. Fit and predict the clusters
  - d. Add the classes to the main data set and save the dataset
  - e. Splitting x and y
  - f. Splitting train and test data
4. Supervised Model Building
  - a. Import the model building Libraries
  - b. Initializing the model
  - c. Model Training
  - d. Evaluating the Model
  - e. Save the Model
5. Application Building
  - a. Create an HTML file
  - b. Build a Python Code

## Project Structure :

A folder named customer segmentation is created that contains the following files :

Name	Date modified	Type	Size
.ipynb_checkpoints	9/1/2022 7:25 PM	File folder	
templates	9/7/2022 9:24 PM	File folder	
app	9/7/2022 9:24 PM	Python File	0 KB
cust_xgbmodel.pkl	9/5/2022 6:11 PM	PKL File	373 KB
customer segmentation using ML	9/5/2022 6:28 PM	Jupyter Source File	633 KB
segmentation data - segmentation da...	9/1/2022 7:25 PM	Microsoft Excel Co...	61 KB

1. app.py which enables us to build an application
2. customer segmentation using ML.ipynb– Model training code file
3. We need the model which is saved and the saved model in this content is cust\_xgbmodel.pkl
4. Templates folder which contains index.HTML file, chance.HTML file, noChance.HTML file.

## Data Collection :

ML depends heavily on data, without data, it is impossible for an “AI” to learn.

It is the most crucial aspect that makes algorithm training possible. In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions.

The dataset used for this project is given in the link below :

<https://docs.google.com/spreadsheets/d/1NnUMX3sjJgRRerkJTAXemlfdyo2GiUh>

[gE\\_m4w-fAhvs/edit#gid=1219451115](#)

## Data Pre-Processing :

Data Pre-processing includes the following main

taskso Import the Libraries.

- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Feature Scaling.

## Import Necessary Libraries :

It is important to import all the necessary libraries such as pandas, NumPy, matplotlib.

5. **Numpy**- It is an open-source numerical Python library. It contains a multi-dimensional array and matrix data structures. It can be used to perform mathematical operations on arrays such as trigonometric, statistical, and algebraic routines.
6. **Pandas**- It is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.
7. **Seaborn**- Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
8. **Matplotlib**- Visualisation with python. It is a comprehensive library for creating static, animated, and interactive visualizations in Python.
9. **Sklearn** – which contains all the modules required for model building.
10. **Scipy** – which contains all the modules required for scientific and computing functions.

# Importing libraries

```
In [1]: ▶ import os
import pandas as pd
import seaborn as sns
import sklearn
import scipy
import matplotlib
import numpy as np
```

## Importing The Dataset :

11. The data might be in .csv files, .excel files

12. To load a .csv data file into pandas we use read\_csv() function. The directory of the CSV file needs to be located at first (it's more efficient to keep the

dataset in the same directory as your program).

If the dataset is in some other location, Then

Data = pd.read\_csv(r"File\_location/datasetname.csv")

## importing dataset

```
In [2]: ▶ df = pd.read_csv('segmentation data - segmentation data.csv')
```

The Dataset segmentation data.csv contains the following Columns

13. ID - Unique id of the customer

14. Sex - Gender of the customer

15. Marital status - whether the person is married or not

16. Age = Age of the person

- 17. Education – Education of the person
- 18. Income – income of the person
- 19. Occupation – indicates the profession of a person, employed or unemployed or business
- 20. Settlement size – Represents the no. of persons in a family

## Analyse The Data :

**head()** method is used to return top n (5 by default) rows of a DataFrame or series.

Analysing the dataset								
In [3]: df.head(10)								
Out[3]:								
	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	100000001	0	0	67	2	124670	1	2
1	100000002	1	1	22	1	150773	1	2
2	100000003	0	0	49	1	89210	0	0
3	100000004	0	0	45	1	171565	1	1
4	100000005	0	0	53	1	149031	1	1
5	100000006	0	0	35	1	144848	0	0
6	100000007	0	0	53	1	156495	1	1
7	100000008	0	0	35	1	193621	2	1
8	100000009	0	1	61	2	151591	0	0
9	100000010	0	1	28	1	174646	2	0



**tail()** method is used to return bottom n (5 by default) rows of a DataFrame or series.

In [4]: `df.tail()`

Out[4]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
1995	100001996	1	0	47	1	123525	0	0
1996	100001997	1	1	27	1	117744	1	0
1997	100001998	0	0	31	0	86400	0	0
1998	100001999	1	1	24	1	97968	0	0
1999	100002000	0	0	25	0	68416	0	0

**describe()** method computes a summary of statistics like count, mean, standarddeviation, min, max and quartile values.

In [5]:

df.describe()

Out[5]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
count	2.000000e+03	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1.000010e+08	0.457000	0.496500	35.909000	1.03800	120954.419000	0.810500	0.739000
std	5.774946e+02	0.498272	0.500113	11.719402	0.59978	38108.824679	0.638587	0.812533
min	1.000000e+08	0.000000	0.000000	18.000000	0.00000	35832.000000	0.000000	0.000000
25%	1.000005e+08	0.000000	0.000000	27.000000	1.00000	97663.250000	0.000000	0.000000
50%	1.000010e+08	0.000000	0.000000	33.000000	1.00000	115548.500000	1.000000	1.000000
75%	1.000015e+08	1.000000	1.000000	42.000000	1.00000	138072.250000	1.000000	1.000000
max	1.000020e+08	1.000000	1.000000	76.000000	3.00000	309364.000000	2.000000	2.000000

From the data we infer that there are only decimal values and no categorical values

**info()** gives information about the data -

```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
#   Column                Non-Null Count  Dtype
---  -
0   ID                    2000 non-null   int64
1   Sex                   2000 non-null   int64
2   Marital status        2000 non-null   int64
3   Age                   2000 non-null   int64
4   Education              2000 non-null   int64
5   Income                2000 non-null   int64
6   Occupation            2000 non-null   int64
7   Settlement size       2000 non-null   int64
dtypes: int64(8)
memory usage: 125.1 KB
```

**shape** attribute in Pandas enables us to obtain the shape of a DataFrame.

```
In [7]: df.shape

Out[7]: (2000, 8)
```

## Handling MissingValues :

1. The Most important step in data pre-processing is dealing with missing data, the presence of missing data in the dataset can lead to low accuracy.
2. Check whether any null values are there or not. If it is present then the following can be done,

# Handling missing values

```
In [8]: df.isnull().sum()
```

```
Out[8]: ID                0
        Sex                0
        Marital status    0
        Age                0
        Education          0
        Income             0
        Occupation        0
        Settlement size    0
        dtype: int64
```

3. There are no null values in the dataset, if there are any null/missing values in the columns of the data, we need to fill it . if there are no null values then this step isskipped.

## Data Visualization :

Data visualization is where a given data set is presented in a graphicalformat. It helpsthe detection of patterns, trends and correlations that might go undetected in text- based data.

- a. Understanding your data and the relationship present within it is just as importantas any algorithm used to train your machine learningmodel. In fact, even the most sophisticated machine learning modelswill performpoorly on data that wasn't visualized and understood properly.
  - b. To visualize the dataset we need librariescalled Matplotlib and Seaborn.
  - c. The Matplotlib libraryis a Python 2D plottinglibrary that allowsyou togenerate plots, scatterplots, histograms, bar charts etc.
- Let's visualize our data using Matplotlib and seaborn library.

Before diving into the code, let's look at some of the basic properties we will be using when plotting.

**xlabel:** Set the label for the

x-axis. **ylabel:** Set the label

for the y-axis. **title:** Set a

title for the axes.

**Legend:** Place a legend on the axes.

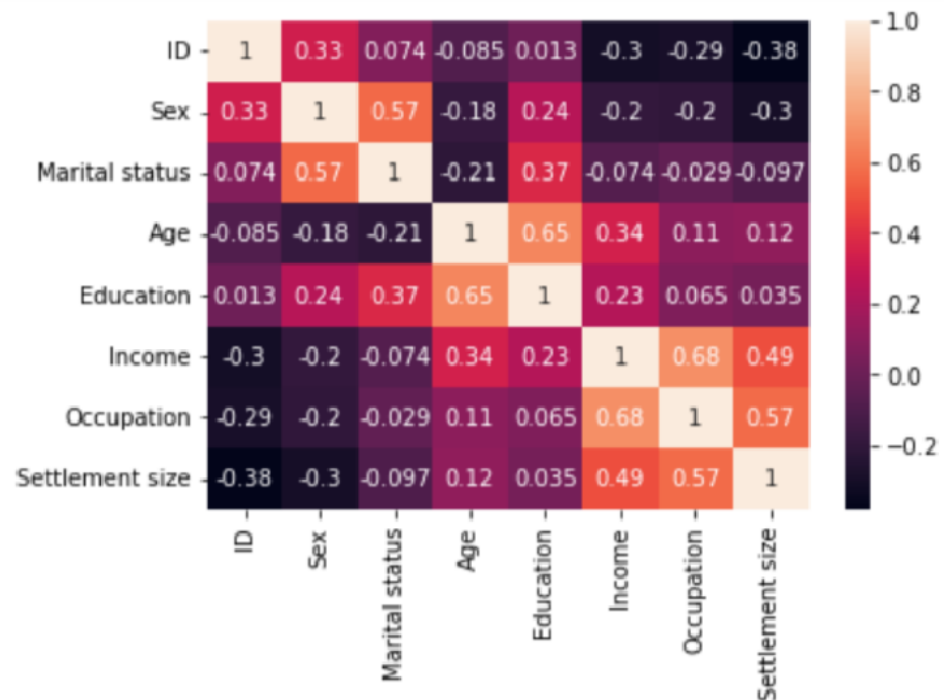
1. **data.corr()** gives the correlation between the columns.

Correlation is a statistical term describing the degree to which two variables move in coordination with one another. If the two variables move in the same direction, then those variables are said to have a positive correlation. If they move in opposite directions, then they have a negative correlation.

2. A **heatmap** is a graphical representation of data that uses a system of color-coding to represent different values. It is used to identify the correlation between the columns using a visual manner.

Data Visualization									
In [12]:	df.corr()								
Out[12]:									
	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	
ID	1.000000	0.328262	0.074403	-0.085246	0.012543	-0.303217	-0.291958	-0.378445	
Sex	0.328262	1.000000	0.566511	-0.182885	0.244838	-0.195146	-0.202491	-0.300803	
Marital status	0.074403	0.566511	1.000000	-0.213178	0.374017	-0.073528	-0.029490	-0.097041	
Age	-0.085246	-0.182885	-0.213178	1.000000	0.654605	0.340610	0.108388	0.119751	
Education	0.012543	0.244838	0.374017	0.654605	1.000000	0.233459	0.064524	0.034732	
Income	-0.303217	-0.195146	-0.073528	0.340610	0.233459	1.000000	0.680357	0.490881	
Occupation	-0.291958	-0.202491	-0.029490	0.108388	0.064524	0.680357	1.000000	0.571795	
Settlement size	-0.378445	-0.300803	-0.097041	0.119751	0.034732	0.490881	0.571795	1.000000	

```
In [13]: sns.heatmap(df.corr(),annot=True)
```

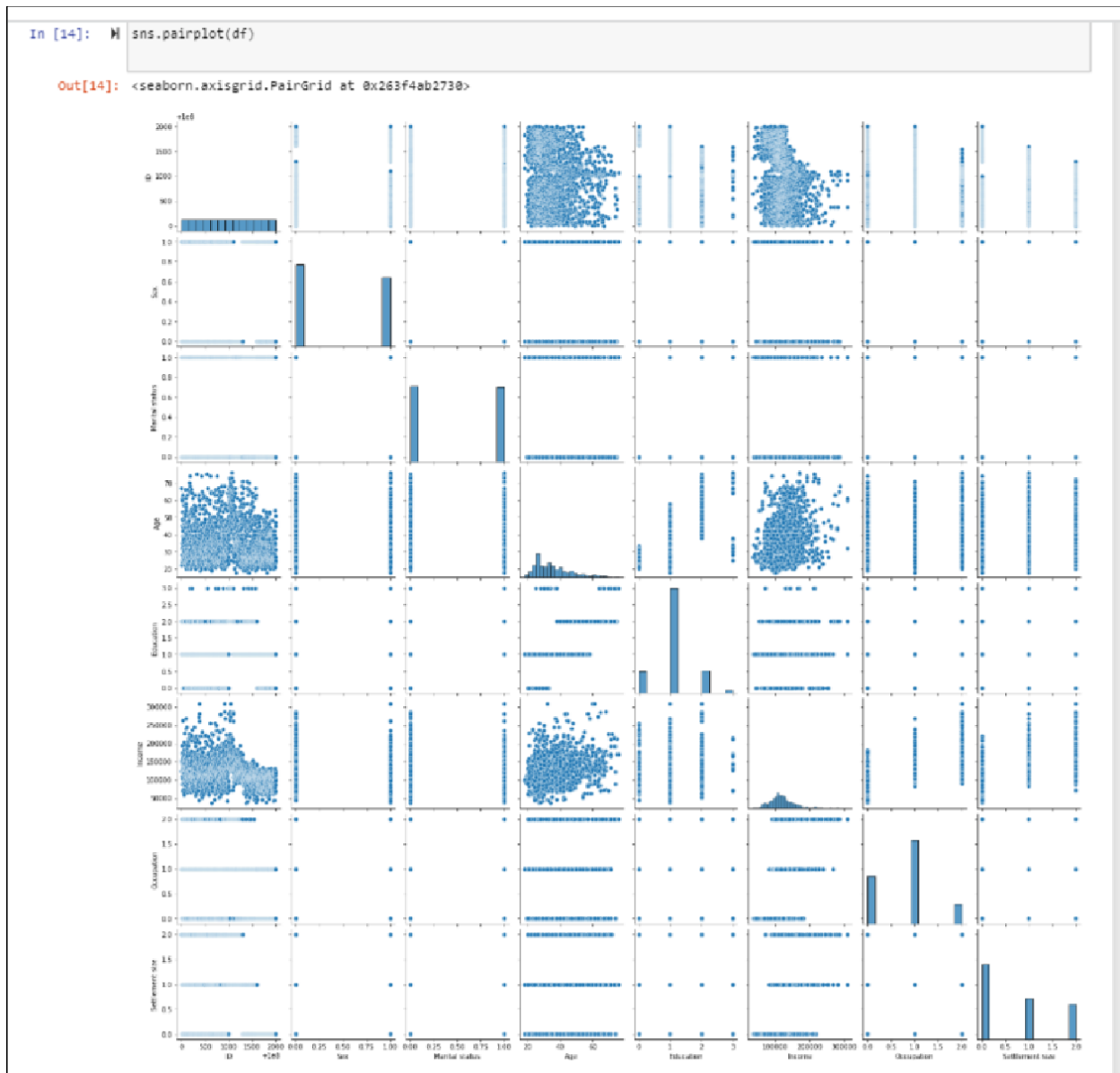


- Correlation strength varies based on colour, lighter the colour between two variables, more the strength between the variables, darker the colour displays the weaker correlation.
- We can see the correlation scale values on the left side of the above image.

### 3. PairPlot: Plot pairwise relationships in a dataset.

- By default, this function will create a grid of Axes such that each numeric variable in data will be shared across the y-axes across a single row and the x-axes across a single column. The diagonal plots are treated differently: a univariate distribution plot is drawn to show the marginal distribution of the data in each column.
- We implement this using the below code:-  
`sns.pairplot(data)`

The output is as shown below -



Pair plot usually gives pair wise relationships of the columns in the dataset. From the above pair plot, we infer that

1. from the above plot we can draw inferences such as linearity and strength between the variables
2. how features are correlated (positive, neutral and negative)

#### 4. Box Plot:

Box-plot is a type of chart often used in explanatory data analysis. Box plots visually show the distribution of numerical data and skewness through displaying the data quartiles (or percentiles) and averages.

Box plots are useful as they show the average score of a data set. The median

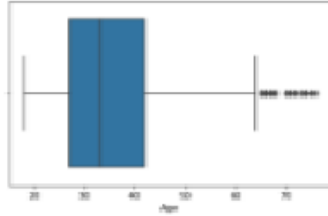
is the average value from a set of data and is shown by the line that divides the box into two parts. Half the scores are greater than or equal to this value and half are less.

jupyter has a built-in function to create a boxplot called `boxplot()`. A boxplot plot is a type of plot that shows the spread of data in all the quartiles. From the above box plot we infer how the data points are spread and the existence of the

```
In [64]: sns.boxplot(df.Age)
```

C:\Users\reaya\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be "data", and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
warnings.warn()

```
Out[64]: <AxesSubplot: xlabel='Age'>
```



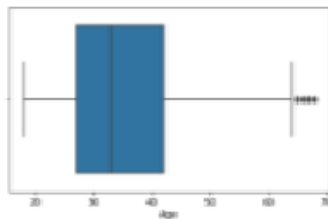
```
In [65]: perc99=df.Age.quantile(0.99)  
perc99
```

```
Out[65]: 68.0
```

```
In [66]: df=df[df.Age<perc99]  
sns.boxplot(df.Age)
```

C:\Users\reaya\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be "data", and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
warnings.warn()

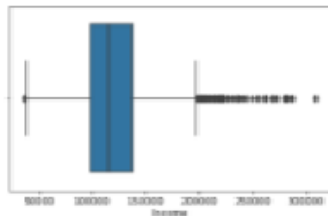
```
Out[66]: <AxesSubplot: xlabel='Age'>
```



```
In [67]: sns.boxplot(df.Income)
```

C:\Users\reaya\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be "data", and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
warnings.warn()

```
Out[67]: <AxesSubplot: xlabel='Income'>
```



```
In [68]: perc99=df.Income.quantile(0.99)  
perc99
```

```
Out[68]: 268673.0
```

```
In [69]: df=df[df.Income<perc99]  
sns.boxplot(df.Income)
```

C:\Users\reaya\anaconda3\lib\site-packages\seaborn\decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be "data", and passing other arguments without an explicit keyword will result in an error or misinterpretation.  
warnings.warn()

```
Out[69]: <AxesSubplot: xlabel='Income'>
```



outliers.



## Feature Scaling :

There is a huge disparity between the x values so let us use feature scaling. Feature scaling is a method used to normalize the range of independent variables or features of data.

After scaling the data will be converted into an array form

- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied.
- After scaling the data will be converted into an array form.
- Loading the feature names before scaling and converting them back to data frame after standard scaling is applied.

### feature scaling

```
In [70]: from sklearn.preprocessing import scale
df=pd.DataFrame(scale(df),columns=df.columns)
```

```
In [71]: df
```

Out[71]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	-1.728513	-0.920063	-0.995421	2.856891	1.698390	0.172171	0.329725	1.568333
1	-1.726790	1.086882	1.004600	-1.208816	-0.027280	0.936614	0.329725	1.568333
2	-1.725066	-0.920063	-0.995421	1.230608	-0.027280	-0.866297	-1.266797	-0.896998
3	-1.723343	-0.920063	-0.995421	0.869212	-0.027280	1.545521	0.329725	0.335667
4	-1.721619	-0.920063	-0.995421	1.592005	-0.027280	0.885599	0.329725	0.335667
...	...	...	...	...	...	...	...	...
1956	1.709801	1.086882	-0.995421	1.049910	-0.027280	0.138639	-1.266797	-0.896998
1957	1.711525	1.086882	1.004600	-0.757071	-0.027280	-0.030661	0.329725	-0.896998
1958	1.713248	-0.920063	-0.995421	-0.395674	-1.752949	-0.948590	-1.266797	-0.896998
1959	1.714971	1.086882	1.004600	-1.028118	-0.027280	-0.609813	-1.266797	-0.896998
1960	1.716695	-0.920063	-0.995421	-0.937769	-1.752949	-1.475262	-1.266797	-0.896998

1961 rows × 8 columns

## Data Pre-Processing :

Data Pre-processing includes the following main tasks

- Import the Libraries.

- Importing the dataset.
- Checking for Null Values.
- Data Visualization.
- Feature Scaling.

## Unsupervised Model Building :

- Import the model building Libraries
- Initializing the model
- Fit and predict the clusters
- Add the classes to the main data set and save the dataset
- Splitting x and y
- Splitting train and test data

## Importing And Initializing The Model :

1. `from sklearn.cluster import Kmeans`
2. `from scipy import spatial`

For selecting no of clusters it is essential to plot an elbow curve, from that we can identify how many no. of clusters can be taken

From the below graph, it can be inferred that the curve has 3 bends (i.e., 0-2, 2-3, and 3-10, making it 3 clusters)

# Unsupervised Model Building

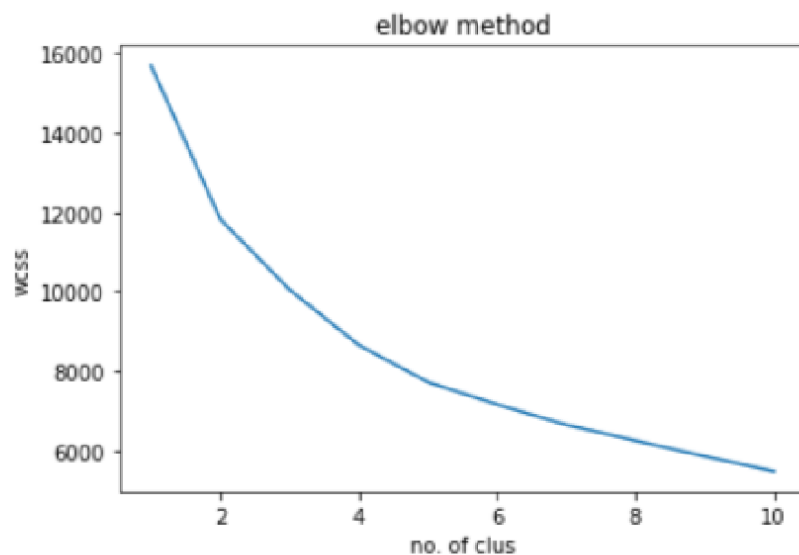
## importing and Initializing the Model

```
In [72]: > from sklearn.cluster import KMeans  
from scipy import spatial
```

```
In [73]: > wssc=[]  
for i in range(1,11):  
    kmeans=KMeans(n_clusters=i,init='k-means++',random_state=0)  
    kmeans.fit(df)  
    wssc.append(kmeans.inertia_)
```

```
In [74]: > import matplotlib as plt
```

```
In [75]: > plt.pyplot.plot(range(1,11),wssc)  
plt.pyplot.title("elbow method")  
plt.pyplot.xlabel("no. of clus")  
plt.pyplot.ylabel("wcss")  
plt.pyplot.show()
```



```

In [76]: km_model=KMeans(n_clusters=3,init="k-means++",random_state=0)

In [77]: ykmeans=km_model.fit_predict(df)

In [78]: df.head()

```

Out[78]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size
0	-1.728513	-0.920063	-0.995421	2.856891	1.69839	0.172171	0.329725	1.568333
1	-1.726790	1.086882	1.004600	-1.208816	-0.02728	0.936614	0.329725	1.568333
2	-1.725066	-0.920063	-0.995421	1.230608	-0.02728	-0.866297	-1.266797	-0.896998
3	-1.723343	-0.920063	-0.995421	0.869212	-0.02728	1.545521	0.329725	0.335667
4	-1.721619	-0.920063	-0.995421	1.592005	-0.02728	0.885599	0.329725	0.335667

```

In [79]: df['kclus']=pd.Series(ykmeans)

In [80]: df.head()

```

Out[80]:

	ID	Sex	Marital status	Age	Education	Income	Occupation	Settlement size	kclus
0	-1.728513	-0.920063	-0.995421	2.856891	1.69839	0.172171	0.329725	1.568333	0
1	-1.726790	1.086882	1.004600	-1.208816	-0.02728	0.936614	0.329725	1.568333	2
2	-1.725066	-0.920063	-0.995421	1.230608	-0.02728	-0.866297	-1.266797	-0.896998	2
3	-1.723343	-0.920063	-0.995421	0.869212	-0.02728	1.545521	0.329725	0.335667	2
4	-1.721619	-0.920063	-0.995421	1.592005	-0.02728	0.885599	0.329725	0.335667	2

## Splitting The Dataset Into Dependent And Independent Variable:

- In machine learning, the concept of the dependent variable (y) and independent variables (x) is important to understand. Here, the Dependent variable is nothing but output in the dataset and the independent variable is all inputs in the dataset.
- With this in mind, we need to split our dataset into the matrix of independent variables and the vector or dependent variable. Mathematically, Vector is defined as a matrix that has just one column.

To read the columns, we will use `iloc` of pandas (used to fix the indexes for selection) which takes two parameters — [row selection, column selection].

Let's split our dataset into independent and dependent variables.

## Splitting The Dataset Into Dependent And Independent Variable

```
In [81]: y=df["kclus"]  
x=df.drop(columns=["kclus","ID"],axis=1)
```

## Splitting The Data Into Train And Test :

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test datasets. For this, you will have a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

5. The train-test split is a technique for evaluating the performance of a machine learning algorithm.
6. Train Dataset: Used to fit the machine learning model.
7. Test Dataset: Used to evaluate the fit machine learning model.
8. In general you can allocate 80% of the dataset to the training set and the remaining 20% to test
9. Now split our dataset into train set and test using train\_test\_split class from scikit-learn library.

from sklearn import model\_selection

```
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state=0)
```

## Splitting The Data Into Train And Test

```
In [82]: from sklearn import model_selection  
x_train,x_test,y_train,y_test=model_selection.train_test_split(x,y,test_size=0.2,random_state=0)
```

## Supervised Model Building :

Model building includes the following main tasks

- Import the model building Libraries

- Initializing the model
- Training and testing the model
- Evaluation of Model
- Savethe Model

## Training And Testing The Model :

10. Once after splitting the data into train and test, the data should be fed to an algorithm to build a model.
11. There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values. The algorithms that you can choose according to the objective that you might have it may be Classification algorithms are classification algorithms.
  - a. Decision Tree classifier
  - b. Random Forest classifier
  - c. xgboost

### **Steps in Building the model:-**

#### **12. Initialize the model**

fit the initialized models with x\_train and y\_train data, it means that we are training the models using train data

## Supervised Model Building :

### Training And Testing The Model

```
In [83]: from sklearn.ensemble import RandomForestClassifier
         from sklearn import tree
         import xgboost

In [84]: rand_model=RandomForestClassifier()
         tree_model=tree.DecisionTreeClassifier()
         xgb_model=xgboost.XGBClassifier()

In [85]: rand_model.fit(x_train,y_train)
         tree_model.fit(x_train,y_train)
         xgb_model.fit(x_train,y_train)

C:\Users\reeya\anaconda3\lib\site-packages\xgboost\sklearn.py:1224: UserWarning: The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].
  warnings.warn(label_encoder_deprecation_msg, UserWarning)
C:\Users\reeya\anaconda3\lib\site-packages\xgboost\data.py:250: FutureWarning: pandas.Int64Index is deprecated and will be removed from pandas in a future version. Use pandas.Index with the appropriate dtype instead.
  elif isinstance(data.columns, (pd.Int64Index, pd.RangeIndex)):

[00:13:01] WARNING: ..\src\learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'multi:softprob' was changed from 'merror' to 'mlogloss'. Explicitly set eval_metric if you'd like to restore the old behavior.

Out[85]: XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                      colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
                      gamma=0, gpu_id=-1, importance_type=None,
                      interaction_constraints='', learning_rate=0.300000012,
                      max_delta_step=0, max_depth=6, min_child_weight=1, missing=-nan,
                      monotone_constraints=(), n_estimators=100, n_jobs=8,
                      num_parallel_tree=1, objective='multi:softprob', predictor='auto',
                      random_state=0, reg_alpha=0, reg_lambda=1, scale_pos_weight=None,
                      subsample=1, tree_method='exact', validate_parameters=1,
                      verbosity=None)
```

## Model Evaluation :

13. Accuracy testing using the train data
14. Predict the y\_test values and calculate the accuracy
15. After predicting we will find the accuracy value of each model.
16. From the below metrics we can conclude that model xgboost gives the best accuracy, other models fall under the category of over-fitting, when measured with train data, so omitting other models and considering the xgboost model for deployment

# model Evaluation

```
In [86]: ▶ pred=rand_model.predict(x_train)
          pred1=tree_model.predict(x_train)
          pred2=xgb_model.predict(x_train)
```

```
In [87]: ▶ from sklearn import metrics
```

```
In [88]: ▶ print(metrics.accuracy_score(pred,y_train))
          print(metrics.accuracy_score(pred1,y_train))
          print(metrics.accuracy_score(pred2,y_train))
```

```
1.0
1.0
1.0
```

```
In [89]: ▶ pred=rand_model.predict(x_test)
          pred1=tree_model.predict(x_test)
          pred2=xgb_model.predict(x_test)
```

```
In [90]: ▶ print(metrics.accuracy_score(pred,y_test))
          print(metrics.accuracy_score(pred1,y_test))
          print(metrics.accuracy_score(pred2,y_test))
```

```
0.926208651399491
0.9338422391857506
0.9312977099236641
```

Save The Model :

After building the model we have to save the model.

Pickle in Python is primarily used in serializing and deserializing a Python object structure. In other words, it's the process of converting a Python



object into a bytearray to store it in a file/database, maintain program state across sessions or transport data over the network. wb indicates write method and rd indicates read method.

17. This is done by the below code

## Saving the model

```
In [91]: import pickle

In [92]: pickle.dump(xgb_model, open("cust_xgbmodel.pkl", 'wb'))
```

## Application Building:

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the user where he has to enter the values for predictions. The entered values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script

## Build HTML Code :

- In this HTML page, we will create the front end part of the web page. In this page we will accept input from the user and Predict the values.
- In our project we have HTML file index.html
- It will display all the input parameters and the prediction text will display the output value of the data given by the user.

The HTML page looks like :

The screenshot shows a web browser window with the address bar displaying '127.0.0.1:5500/templates/index.html'. The browser's toolbar includes icons for Gmail, YouTube, Maps, Downloads, History, and New Tab. The main content area is titled 'Customer segmentation' and contains the text 'please enter the following details'. Below this text is a form with the following fields: Sex (Female), Marital status (Single), Age (Age), Education (Educa), Income (Income), Occupation (Not working), and Settlement size (1). A 'Predict' button is located below the form. The background of the page features a large thumbs-up icon and several people holding smaller thumbs-up icons.

Main Python Script :

Let us build an app.py flask file which is a web framework written in python for server-side scripting. Let's see step by step procedure for building the backend application.

In order to develop web API with respect to our model, we basically use the Flask framework which is written in python.

Importing the necessary libraries for building a flask application and integrating model and HTML pages

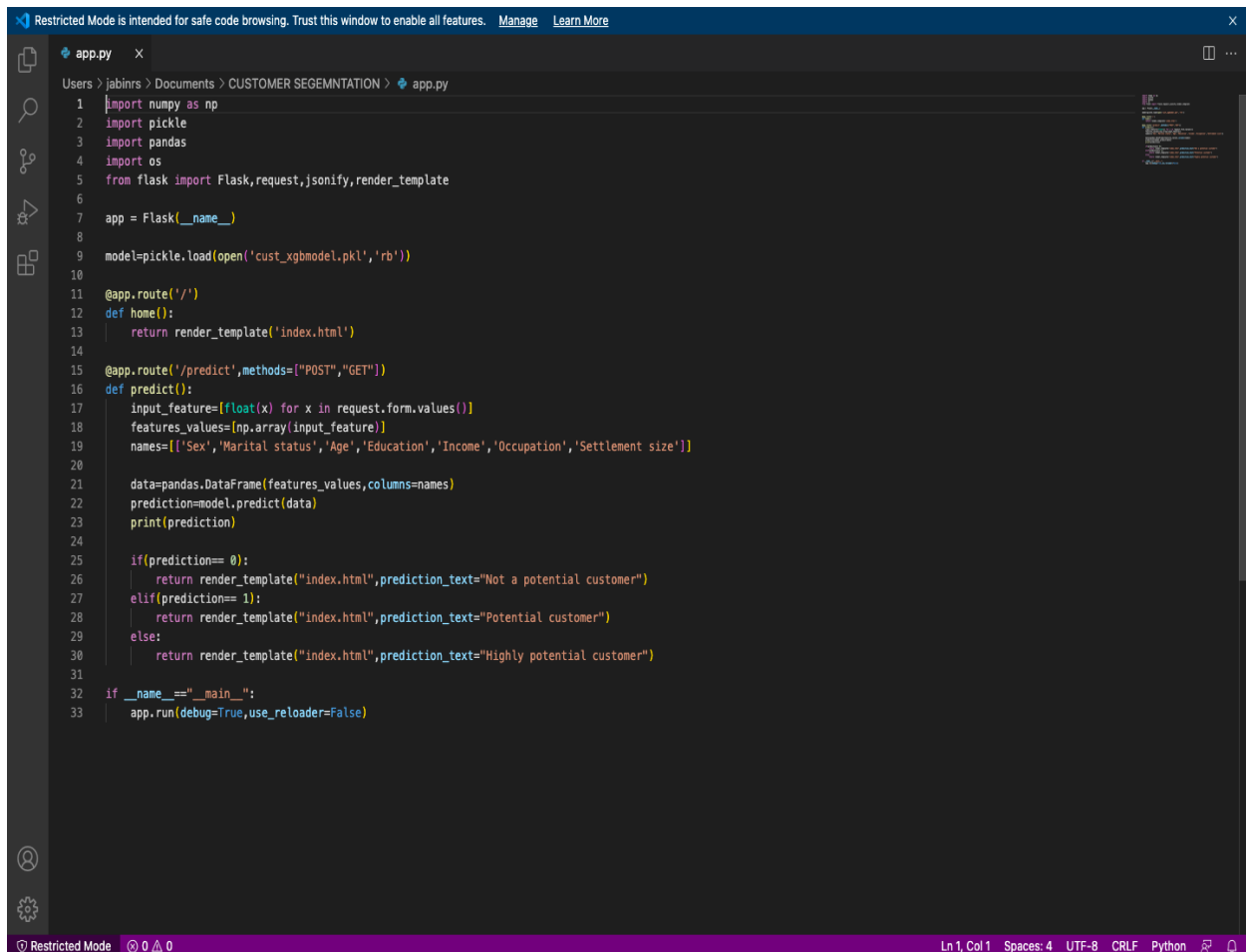
Initializing the flask app

- Calling the pkl models and saving into a variable
- Routing and rendering to the HTML page
- Calling the inputs from the HTML page and saving into the variable
- Creating the data labels
- Forming the data frame with labels and the data
- Scaling the data
- Predicting the values, by passing the data into the model
- Rendering the results onto the HTML pages based on the output

If the output is class-0, it means a page that displays non-potential customers will be rendered, if the output is 1, a page with the

potential customers will be displayed and the output is 2 a page with highly potential customers will be rendered.

- The value of `__name__` is set to `__main__` when the module runs as the main program; otherwise, it is set to the name of the module.



```
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
app.py x
Users > jabinns > Documents > CUSTOMER SEGMENTATION > app.py
1 import numpy as np
2 import pickle
3 import pandas
4 import os
5 from flask import Flask, request, jsonify, render_template
6
7 app = Flask(__name__)
8
9 model=pickle.load(open('cust_xgbmodel.pkl','rb'))
10
11 @app.route('/')
12 def home():
13     return render_template('index.html')
14
15 @app.route('/predict',methods=['POST',"GET"])
16 def predict():
17     input_features=[float(x) for x in request.form.values()]
18     features_values=np.array(input_feature)
19     names=[['Sex','Marital status','Age','Education','Income','Occupation','Settlement size']]
20
21     data=pandas.DataFrame(features_values,columns=names)
22     prediction=model.predict(data)
23     print(prediction)
24
25     if(prediction== 0):
26         return render_template("index.html",prediction_text="Not a potential customer")
27     elif(prediction== 1):
28         return render_template("index.html",prediction_text="Potential customer")
29     else:
30         return render_template("index.html",prediction_text="Highly potential customer")
31
32 if __name__ == "__main__":
33     app.run(debug=True,use_reloader=False)
```

## Run The App :

- Open anaconda prompt from the Start menu
- , Navigate to the folder where your python script is.
- Now type the “python app.py” command

Navigate to the localhost where you can view your web page, Then it will run on local host:5000

```
Anaconda Prompt (anaconda3) - conda activate bse - conda activate base - conda activate base - python app.py

(hase) (:Users\reeya\PT1\customer_segmentation>conda activate deployment

(deployment) C:\Users\reeya\PT1\customer_segmentation>python app.py
C:\Users\reeya\anaconda3\envs\deployment\lib\site-packages\xgboost\compat.py:36: FutureWarning: pandas.Int64Index is deprecated in favor of pandas.Index; the attribute (np.dtype) is deprecated in favor of dtype.
  from pandas import MultiIndex, Int64Index
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
[0]
127.0.0.1 - - [09/Sep/2022 18:01:19] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [09/Sep/2022 18:01:19] "GET /static/confused.jpg HTTP/1.1" 304 -
127.0.0.1 - - [09/Sep/2022 18:01:19] "GET /static/happy-customer.jpg HTTP/1.1" 304 -
127.0.0.1 - - [09/Sep/2022 18:01:19] "GET /static/cust.jpg HTTP/1.1" 304 -
127.0.0.1 - - [09/Sep/2022 18:01:19] "GET /static/nocust.jpg HTTP/1.1" 304 -
127.0.0.1 - - [09/Sep/2022 18:01:19] "GET /static/bg.jpg HTTP/1.1" 304 -
```

## Output :

- Copy the HTTP link and paste it in google link tab, it will display the form page
- Enter the values as per the form and click on predict button
- It will redirect to the page based on prediction output
- If the prediction belongs to class-2, it means that the customer is highly potential
- If the prediction belongs to class-1, it means that the customer is potential
- If the prediction belongs to class-0, it means the customer is a not potential

**Customer segmentation**

**please enter the following details**

Sex:

Marital status:

Age:

Education:

Income:

Occupation:

Settlement size:

Highly potential customer

← → ↻ 127.0.0.1:5000/predict

Gmail YouTube Maps Downloads History New Tab

## Customer segmentation

**please enter the following details**

Sex:

Marital status:

Age:

Education:

Income:

Occupation:

Settlement size:

Not a potential customer

