CUSTOMER SEGEMENTATION

A Project Report

In partial fulfilment of the requirements for the the Award of the degree
Bachelor of Computer Application
In
IT Depaterment
Under the guidance of
Joyjit Guha Biswas

By
Md Yaser Hamid
Md Aafaque Anwar Haque

Netaji Subhas University
In association with

SDF Building Module #132, Ground Floor, Salt Lake City, GP Block,
Sector V, Kolkata, West Bengal 700091

Title of the Project: Customer Segmentation
Project Members: Md Yaser Hamid
Md Aafaque Anwar Haque

Name of the guide: Mr. Joyjit Guha Biswas

Address: Ardent Computech Pvt.Ltd
(An ISO 90 01 : 2015 Certified)
SDF Building, Module #132, Ground Floor,
Salt Lake City, GP Block, Sector V,
Kolkata, West Bengal, 700091

Project Version Control History:

Version	Primary Author	Description of Version	Date Completed
Final	Md Yaser Hamid & Md Aafaque Anwar Haque	Project Report	21th Nov 2024.

Signature Of Team Members

Date:

Signature Of Approver

Date:

For Office Use Only

M r. Joyjit Guha Biswas
Project Proposal Evaluator

Declaration

We hereby declare that the project work being presented in the project Proposal entited "Customer Segmentation" in partial fulfilment of the requirements for the award of the degree of Master of computer Application at Ardent Computech PVT LTD, Saltlake,Kolkata, West Bengal, is an authentic work carried out under the guidance of Mr Joyjit

Biswas The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and Belief.

Date :

Name of the Student : *Md Yaser Hamid.*
Md Aafaque Anwar Haque.

Ardent Computech Pvt Ltd(An ISO 9001:2015 Certified)
SDF Building Module #132,Ground Floor,Salt Lake City, GP Block, Sector V,

Kolkata,West Bengal 700091

Certificate

This is to certify that this proposal of minor project entitled "Customer Segmentation" Is a record of Bonafide work , carried out by Shiladity Maitra under my guidance at Ardent Computech PVT.LTD. In my opinion the report in this present form is in partial fulfillment of the requirement for the award of the degree of Master of Computer Application and as per regulation of the Ardent. To the best of my knowledge, the results embodied in this report are original in nature and worthly of incorporation in the present version of the report.

Guide / Supervisor

MR.Joyjit Guha Biswas

Project Engineer
Ardent Computech Pvt Ltd(An ISO 9001:2015 Certified)
SDF Building Module #132,Ground Floor,Salt Lake City, GP Block, Sector V, Kolkata,West Bengal 70009

Acknowledgement

I would like to extend my sincere gratitude to everyone who contributed to the successful completion of this project.

First and foremost, I am deeply thankful to ***M r. Joyjit Guha Biswas*** *for their guidance, invaluable feedback, and continuous support throughout this project. Their expertise and insights were instrumental in shaping the analysis and ensuring the quality of the outcomes.*

I would also like to express my appreciation to my colleagues and team members who provided encouragement, shared knowledge, and offered constructive suggestions that enhanced the depth of this analysis.

I am grateful to Ardent Computech PVT.LTD, Kolkata. for granting access to the data and providing the necessary tools and resources required for this project. Their support enabled me to explore the data thoroughly and derive meaningful insights.

Lastly, I would like to thank my family and friends for their encouragement and understanding, which motivated me to persevere throughout the project.

Thank you all for your contributions, guidance, and encouragement.

Sincerely,
Md Yaser Hamid
Md Aafaque Anwar Haque.

Overview:

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and has fewer syntactical constructions than other languages.

Python is interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to Perl and PHP.

Python is Interactive: You can actually sit at a Python prompt and interactwith the interpreter directly to write your programs.

Python is Object-Oriented: Python supports Object-Oriented style ortechnique of programming that encapsulates code within objects.

Python is a Beginner's Language: Python is a great language for the beginner-level programmers and supports the development of a wide range ofapplications from simple text processing to WWW browsers to games.

History of Python:

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Features of Python:

Easy-to-learn: Python has few Keywords, simple structure and clearly

defined syntax. This allows a student to pick up the language quickly.

Easy-to-Read: Python code is more clearly defined and visible to the eyes. Easy -to-Maintain: Python's source code is fairly easy-to-maintain.

A broad standard library: Python's bulk of the library is very portable and cross platform compatible on UNIX, Windows, and Macintosh.

Interactive Mode: Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

Portable: Python can run on the wide variety of hardware platforms and has the same interface on all platforms.

Extendable: You can add low level modules to the python interpreter. These modules enables programmers to add to or customize their tools to be more efficient.

Databases: Python provides interfaces to all major commercial databases.

GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

Scalable: Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- It support functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte code for building large applications.
- It provides very high level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, Active X, CORBA and JAVA.

Environment Setup:

- 64-Bit OS
- Install Python 3
- Setup virtual environment
- Install Packages
-

Basic Syntax of Python Program:

Type the following text at the Python prompt and press the Enter –

>>> print "Hello, Python!"
If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");.
However in Python version 2.4.3, this produces the following result –

Hello, Python!

Python Identifiers:

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9). Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive proming language.

Python Keywords:

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

For example:
And, exec, not Assert, finally, orBreak, for, pass Class, from, print ,continue, global, raisedef, if, return, del, import, tryelif, in, while else, is, with ,except, lambda, yield.

Lines & Indentation:

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True: print "True"
else:
print "False"
```

Command Line Arguments:

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...
```

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d : debug output from parser (also PYTHONDEBUG=x)

-E : ignore environment variables (such as PYTHONPATH)

-h : print this help message and exit [etc.]

Variable Types:

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables:

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10          # An integer assignment
weight=10.60              # A floating point
name="Ardent"  # A string
```

Multiple Assignment:

Python allows you to assign a single value to several variables simultaneously.
For example −

```
a = b = c = 1
a,b,c       =       1,2,"hello"
```

Standard Data Types:

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

- String
- List
- Tuple
- Dictionary
- Number

Data Type Conversion:

Sometimes, you may need to perform conversions between the built-in types.
To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string.
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.

Functions:

Defining a Function:

- def functionname(parameters):
  "function_docstring"
  function_suite
  return [expression]

Pass by reference vs Pass by value:

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

```
# Function definition is here

def changeme(mylist):
    "This changes a passed list into this function"
    mylist.append([1,2,3,4]);
    print"Values inside the function: ",mylist
    return

# Now you can call changeme function

mylist=[10,20,30];
changeme(mylist);
print"Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                # This is global variable.
```

```
# Function definition is here

def sum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2;              # Here total is local variable.
print"Inside the function local total : ", total
return total;

# Now you can call sum functionsum(10,20);
print"Outside the function global total : ", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total : 30
Outside the function global total : 0
```

Modules:

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named aname normally resides in a file named aname.py. Here's an example of a simple module, support.py

```
def print_func( par ):
        print"Hello : ",
        par return
```

The import Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The import has the following syntax –

```
import module1[, module2[,… moduleN]
```

Packages:

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file Pots.py available in Phone directory. This file has following line of source code −

```
def Pots():
        print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

Phone/Isdn.py file having function Isdn()
Phone/G3.py file having function G3()

Now, create one more file _init_.py in Phone directory −

Phone/_init_.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in _init_.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

Numpy:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

Scipy:

Scientific computing tools for Python
SciPy refers to several related but distinct entities:

- The SciPy ecosystem, a collection of open source software for scientific computing in Python.
- The community of people who use and develop this stack.
- Several conferences dedicated to scientific computing in Python - SciPy, EuroSciPy, and SciPy.in.
- The SciPy library, one component of the SciPy stack, providing many numerical routines.

The SciPy ecosystem
Scientific computing in Python builds upon a small core of packages:

- Python, a general purpose programming language. It is interpreted and dynamically typed and is very well suited for interactive work and quick prototyping, while being powerful enough to write large applications in.
- NumPy, the fundamental package for numerical computation. It defines the numerical array and matrix types and basic operations on them.
- The SciPy library, a collection of numerical algorithms and domain-specific toolboxes, including signal processing, optimization, statistics, and much more.
- Matplotlib, a mature and popular plotting package that provides publication-quality 2-D plotting, as well as rudimentary 3-D plotting.

On this base, the SciPy ecosystem includes general and specialised tools for data management and computation, productive experimentation, and high-performance computing. Below, we overview some key packages, though there are many more relevant packages.

Data and computation:

- pandas, providing high-performance, easy-to-use data structures.
- SymPy, for symbolic mathematics and computer algebra.
- NetworkX, is a collection of tools for analyzing complex networks.
- scikit-image is a collection of algorithms for image processing.
- scikit-learn is a collection of algorithms and tools for machine learning.
- h5py and PyTables can both access data stored in the HDF5 format.

Productivity and high-performance computing:

- IPython, a rich interactive interface, letting you quickly process data and test ideas.
- The Jupyter notebook provides IPython functionality and more in your web browser, allowing you to document your computation in an easily reproducible form.
- Cython extends Python syntax so that you can conveniently build C extensions, either to speed up critical code or to integrate with C/C++ libraries.
- Dask, Joblib or IPyParallel for distributed processing with a focus on numeric data.

Quality assurance:

- nose, a framework for testing Python code, being phased out in preference for pytest.
- numpydoc, a standard and library for documenting Scientific Python libraries.

Pandas:

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

Library features:

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.

- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

Python Speech Features:

This library provides common speech features for ASR including MFCCs and filterbank energies. If you are not sure what MFCCs are, and would like to know more have a look at this MFCC tutorial: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/.

You will need numpy and scipy to run these files. The code for this project is available at https://github.com/jameslyons/python_speech_features .

Supported features:

- python_speech_features.mfcc() - Mel Frequency Cepstral Coefficients
- python_speech_features.fbank() - Filterbank Energies
- python_speech_features.logfbank() - Log Filterbank Energies
- python_speech_features.ssc() - Spectral Subband Centroids

To use MFCC features:

```
from python_speech_features import mfccfrom python_speech_features import logfbankimport scipy.io.wavfile as wav

(rate,sig) = wav.read("file.wav")

mfcc_feat = mfcc(sig,rate)fbank_feat = logfbank(sig,rate)

print(fbank_feat[1:3,:])
```

OS: Miscellaneous operating system interfaces

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

The design of all built-in operating system dependent modules of Python issuch that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).

Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.

On VxWorks, os.fork, os.execv and os.spawn*p* are not supported.

Pickle: Python object serialization

The pickle module implements binary protocols for serializing and de-serializing a Python object structure. *"Pickling"* is the process whereby a Python object hierarchy is converted into a byte stream, and *"unpickling"* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshalling," 1 or "flattening"; however, to avoid confusion, the terms used here are "pickling" and "unpickling".

Operator: Standard operators as functions

The operator module exports a set of efficient functions corresponding to the intrinsic operators of Python. For example, operator.add(x, y) is equivalent to the expression x+y. Many function names are those used for special methods, without the double underscores. For backward compatibility, many of these have a variant with the double underscores kept. The variants without the double underscores are preferred for clarity.

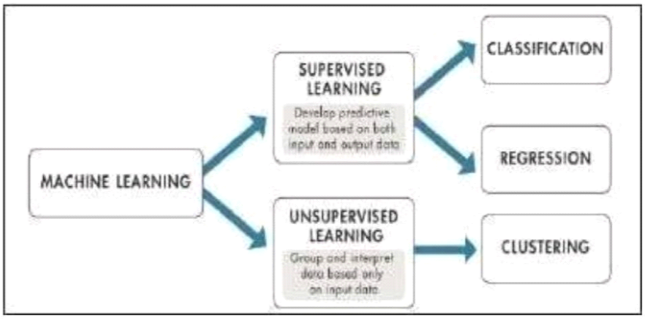
The functions fall into categories that perform object comparisons, logical operations, mathematical operations and sequence operations.

Tempfile: Generate temporary files and directories

This module creates temporary files and directories. It works on all supported platforms. TemporaryFile, NamedTemporaryFile, TemporaryDirectory, and SpooledTemporaryFile are high-level interfaces which provide automatic cleanup and can be used as context managers. mkstemp() and mkdtemp() are lower-level functions which require manual cleanup.

All the user-callable functions and constructors take additional arguments which allow direct control over the location and name of temporary files and directories. Files names used by this module include a string of random characters which allows those files to be securely created in shared temporary directories. To maintain backward compatibility, the argument order is somewhat odd; it is recommended to use keyword arguments for clarity.

Introduction to Machine Learnning:



Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

Supervised Learning:

Supervised learning is the machine learning task of inferring a function from labeled training data.[1] The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal

scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Unsupervised Learning:

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given tothe learner are unlabelled, there is no evaluation of the accuracy of the structurethat is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (andsolutions) involving summarizing and explaining key features of the data.

Linear Regression Algorithm:

Linear regression is a simple and widely used algorithm in machine learning and statistics for predicting continuous numerical values based on input features. It fits a linear equation to the data, where the relationship between the dependent variable (target) and one or more independent variables (features) is modeled as a straight line. The main goal of linear regression is to find the best-fitting line that minimizes the difference between the predicted values and the actual values of the target variable.

The equation of a linear regression model can be represented as:

$$y = b0 + b1 * x1 + b2 * x2 + ... + bn * xn$$

where:

y is the predicted value (the dependent variable).
b0 is the intercept term, representing the value of y when all input features (x1, x2, ..., xn) are zero.

b1, b2, ..., bn are the coefficients (also known as slopes) of the respective input features (x1, x2, ..., xn).
The goal of training a linear regression model is to find the best values for the coefficients (b0, b1, b2, ..., bn) that minimize the error between the predicted values and the actual target values in the training data. The most common method used to find these coefficients is called "Ordinary Least Squares" (OLS), where the sum of the squared differences between the predicted and actual values is minimized.

Once the coefficients are determined, the model can be used to make predictions on new data by simply plugging the feature values into the linear equation.

Linear regression is suitable for problems where the relationship between the target variable and the features is approximately linear. However, it may not perform well when the relationship is highly non-linear or if there are complex interactions between the features.

Algorithm:

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: We have collected data sets of movies from online website. We have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets.

Model Selection: We have selected different models to minimize the error of thepredicted value. The different models used are Linear Regression Linear Model.

Training: The data sets was divided such that x_train is used to train the modelwith correspondingx_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict . Bothy_train and y_predict was compared.

CUSTOMER SEGMENTATION

Introduction:-

Customer segmentation is the process of dividing a customer base into distinct groups based on shared characteristics, behaviors, or needs. This project aims to utilize machine learning and statistical analysis to create a customer segmentation model that uncovers patterns in the data, allowing businesses to target specific customer groups more effectively.

Machine learning techniques, such as clustering algorithms, will be used to identify natural groupings within customer data, based on factors like purchasing behavior, demographics, and preferences. These algorithms help uncover hidden relationships and enable more accurate segmentation.

Statistical analysis will validate and refine the segmentation by providing insights into the key characteristics that define each group. Techniques like correlation analysis will help understand how different variables influence customer behavior.

The ultimate goal is to deliver actionable insights that businesses can use to create personalized marketing strategies, improve customer retention, and optimize resource allocation, leading to enhanced business performance.

Understanding Dataset

Analyzing the dataset, which encompasses valuable information on customer demographics, spending behavior, and shopping preferences. step-by-step approach to examining the Mall Customers dataset, focusing on data

preparation, pattern exploration, and clustering techniques, all executed using Python and Pandas.

Importing the dataset into the google Colab notebook for analysis:

Importing the Dependencies

```
[1] # Import necessary libraries for data analysis,visualization & clustering.
    import numpy as np
    import pandas as pd
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.cluster import KMeans
```

Data collection and Analysis

Data Collection & Analysis

```
# loading the data from csv file to a Pandas DataFrame
customer_data = pd.read_csv('/content/Mall_Customers.csv')
```

```
[3] # first 5 rows in the dataframe
    customer_data.head()
```

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

```
[4] # finding the number of rows and columns
    customer_data.shape
```

```
(200, 5)
```

```
[5] # getting some informations about the dataset
    customer_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Gender                  200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
```

Checking for missing values

```
# checking for missing values
customer_data.isnull().sum()
```

	0
CustomerID	0
Gender	0
Age	0
Annual Income (k$)	0
Spending Score (1-100)	0

dtype: int64

Selection of columns

Choosing the Annual Income Column & Spending Score column

```
[7] X = customer_data.iloc[:,[3,4]].values
```

```
[8] print(X)
```

```
[ 76  40]
[ 76  87]
[ 77  12]
[ 77  97]
[ 77  36]
[ 77  74]
[ 78  22]
[ 78  90]
[ 78  17]
[ 78  88]
[ 78  20]
[ 78  76]
[ 78  16]
[ 78  89]
[ 78   1]
[ 78  78]
[ 78   1]
[ 78  73]
[ 79  35]
[ 79  83]
[ 81   5]
[ 81  93]
[ 85  26]
[ 85  75]
[ 86  20]
[ 86  95]
[ 87  27]
[ 87  63]
[ 87  13]
[ 87  75]
[ 87  10]
[ 87  92]
[ 88  13]
[ 88  86]
[ 88  15]
[ 88  69]
[ 93  14]
[ 93  90]
[ 97  32]
[ 97  86]
[ 98  15]
[ 98  88]
[ 99  39]
[ 99  97]
[101  24]
[101  68]
[103  17]
[103  85]
[103  23]
[103  69]
[113   8]
[113  91]
[120  16]
```

```
[ 99  39]
[ 99  97]
[101  24]
[101  68]
[103  17]
[103  85]
[103  23]
[103  69]
[113   8]
[113  91]
[120  16]
[120  79]
[126  28]
[126  74]
[137  18]
[137  83]]
```

Clustering:

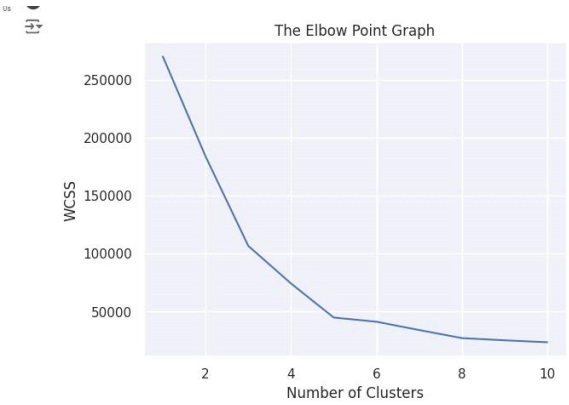
Choosing the number of clusters

WCSS -> Within Clusters Sum of Squares

```
# finding wcss value for different number of clusters

wcss = []

for i in range(1,11):
  kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
  kmeans.fit(X)

  wcss.append(kmeans.inertia_)
```

```
[10] # plot an elbow graph

sns.set()
plt.plot(range(1,11), wcss)
plt.title('The Elbow Point Graph')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```

Graph representation



Optimum Number of Clusters = 5

Training the k-Means Clustering Model

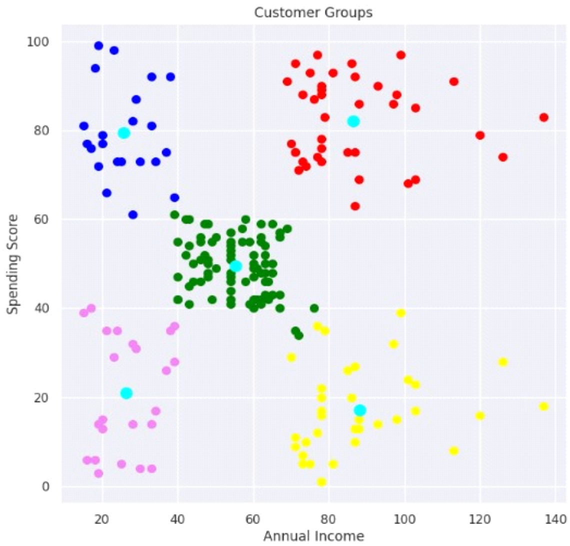
```
kmeans = KMeans(n_clusters=5, init='k-means++', random_state=0)

# return a label for each data point based on their cluster
Y = kmeans.fit_predict(X)

print(Y)
```

```
[3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3 4 3
 4 3 4 3 4 3 0 3 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 1 2 1 2 1 0 1 2 1 2 1 2 1 2 1 0 1 2 1 2 1
 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1 2
 1 2 1 2 1 2 1 2 1 2 1 2 1 2 1]
```

5 Clusters - 0, 1, 2, 3, 4

Visualizing all the Clusters

```
[12] # plotting all the clusters and their Centroids

plt.figure(figsize=(8,8))
plt.scatter(X[Y==0,0], X[Y==0,1], s=50, c='green', label='Cluster 1')
plt.scatter(X[Y==1,0], X[Y==1,1], s=50, c='red', label='Cluster 2')
plt.scatter(X[Y==2,0], X[Y==2,1], s=50, c='yellow', label='Cluster 3')
plt.scatter(X[Y==3,0], X[Y==3,1], s=50, c='violet', label='Cluster 4')
plt.scatter(X[Y==4,0], X[Y==4,1], s=50, c='blue', label='Cluster 5')

# plot the centroids
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], s=100, c='cyan', label='Centroids')

plt.title('Customer Groups')
plt.xlabel('Annual Income')
plt.ylabel('Spending Score')
plt.show()
```



Conculsion

The customer segmentation project utilizing machine learning successfully demonstrated the power of data-driven techniques in identifying distinct customer groups. By applying clustering algorithms such as KMeans, we categorized customers based on features like age, annual income, and spending score. These segments provide valuable insights into customer behavior and preferences, enabling businesses to design targeted marketing strategies, optimize resource allocation, and enhance customer experiences.

The use of machine learning in customer segmentation ensures scalable and efficient analysis, making it a vital tool for modern business intelligence. Future improvements could include exploring advanced clustering algorithms, integrating additional data features, and evaluating the real-world impact of these segments on business outcomes.

Note:-

The datasheet we have used to execute our project on "Customer Segmentation" is given below ---

Future Work

1. Model Improvement and Fine-Tuning

Hyperparameter Tuning: Explore techniques like Grid Search or Random Search to fine-tune clustering model parameters (e.g., number of clusters, initialization methods for K-Means).

Alternative Clustering Algorithms: Experiment with other clustering algorithms like DBSCAN, Gaussian Mixture Models (GMM), or Agglomerative Clustering to improve segment quality and handle different types of data distributions.

Deep Learning Approaches: Investigate the use of deep learning-based methods like Autoencoders for anomaly detection or self-organizing maps (SOMs) for complex data structures.

2. Incorporating Temporal and Real-Time Data

Dynamic Segmentation: Explore the possibility of adapting segments over time as customer behavior changes. For example, incorporate time series analysis or real-time data streaming to update customer segments.

Customer Churn Prediction: Combine segmentation with churn prediction models to proactively identify customers at risk of leaving and create retention strategies for different segments.

3. Integration with Business Processes

Customer Insights Dashboard: Build an interactive dashboard using tools like Power BI or Tableau to visualize the customer segments and the insights derived from them, helping business teams make data-driven decisions.

Integration with Marketing Systems: Implement the segmentation model into real-time marketing or recommendation systems. This could include personalized product recommendations, email campaigns, or promotional offers tailored to each customer segment.

Customer Lifetime Value (CLV): Integrate CLV prediction with the segmentation model to further refine business strategies around high-value segments and optimize resource allocation.

4. Data Enrichment

Additional Data Sources: Integrate external datasets to enrich customer profiles, such as demographic, geographic, or social media data. Enriching the data could lead to more accurate and meaningful segmentation.

Text Analysis and NLP: Incorporate unstructured data, such as customer reviews or support tickets, using Natural Language Processing (NLP) techniques to uncover hidden patterns and segment customers based on sentiment or feedback.

5. Evaluation and Validation

Better Evaluation Metrics: Explore more sophisticated evaluation metrics for clustering, such as Silhouette Score, Davies-Bouldin Index, or Adjusted Rand Index, to validate the quality of the segmentation.

Cluster Stability: Assess the stability of the segments over time and their ability to generalize to new data (e.g., using cross-validation or bootstrapping techniques).

6. Scalability and Real-Time Application

Scalable Implementation: Implement the model in a scalable way to handle large datasets efficiently, using cloud platforms like AWS, Google Cloud, or Azure for real-time processing and predictions.

Real-time Segmentation: Design and deploy an end-to-end system that segments customers in real-time based on new interactions, allowing businesses to dynamically adjust marketing strategies.

7. Ethical and Bias Considerations

Bias Mitigation: Investigate potential biases in the segmentation model, such as skewed demographic representation or over-representation of specific groups, and implement methods to mitigate these biases.

Privacy Concerns: Address privacy and data security concerns, ensuring compliance with data protection regulations (e.g., GDPR) when handling sensitive customer information.

8. Cross-Industry Applications

Industry-Specific Customization: Explore applying the customer segmentation model in other industries, such as finance, healthcare, or e-commerce, and

tailor the segmentation methodology to each industry's unique characteristics and customer behavior patterns.

" References"

- *https://m.youtube.com/*
- *https://www.kaggle.com/*
- *https://github.com/*
- *https://openai.com/index/chatgpt/*

THANK YOU