**Customer Segmentation using Unsupervised Learning**

**A PROJECT REPORT**

**for**

**Introduction To AI (AI101B)**

**Session (2024-25)**

**Submitted by**

**PREKSHA RUHELA**

**(University Roll No 202410116100148)**

**NIKHIL**

**(University Roll No 202410116100132)**

**PRIYAKANT TYAGI**

**(University Roll No 202410116100151)**

**Submitted in partial fulfilment of the**

**Requirements for the Degree of**

**MASTER OF COMPUTER APPLICATION**

**Under the Supervision of**

**MS. KOMAL SALGOTRA**

### **(Assistant Professor)**



**Submitted to**

**Department Of Computer Applications**

**KIET Group of Institutions, Ghaziabad**

**Uttar Pradesh-201206**

**(APRIL 2025)**

**Introduction**

In today’s competitive business landscape, understanding customer behaviour is crucial for formulating effective marketing strategies and delivering personalized experiences. **Customer Segmentation** is a powerful analytical technique that divides a company’s customer base into distinct groups with shared characteristics such as preferences, demographics, and buying patterns. By segmenting customers, businesses can tailor their strategies and target specific groups more effectively and enhance overall market value.

This segmentation enables businesses to tailor their offerings, improve customer satisfaction, and enhance overall profitability.

In this project, we utilize **Unsupervised Machine Learning techniques** to perform customer segmentation. Specifically, we apply clustering algorithms, most notably K-Means Clustering, to discover natural groupings within a customer dataset without relying on predefined labels. The process involves several critical stages including data preprocessing, feature scaling, dimensionality reduction using t-SNE (T-distributed Stochastic Neighbour Embedding) for visualization, and application of clustering to identify distinct customer groups.

Unsupervised learning is a branch of [**machine learning**](https://www.geeksforgeeks.org/machine-learning/) that deals with unlabeled data. Unlike supervised learning, where the data is labeled with a specific category or outcome, unsupervised learning algorithms are tasked with finding patterns and relationships within the data without any prior knowledge of the data’s meaning. Unsupervised machine learning algorithms find hidden patterns and data without any human intervention, i.e., we don’t give output to our model. The training model has only input parameter values and discovers the groups or patterns on its own.

By analyzing customer behaviour and demographics, the goal is to identify meaningful segments that can be strategically targeted with customized marketing efforts. This project demonstrates how machine learning can unlock actionable insights from raw data and support data-driven decision making in customer relationship management.

**Methodology**

The methodology adopted for this project follows a structured pipeline that includes data acquisition, preprocessing, visualization, feature engineering, dimensionality reduction, and clustering using unsupervised learning algorithms. Each step is essential in ensuring the reliability and accuracy of the final segmentation. Below is a detailed explanation of the process:

**1. Data Collection and Exploration**

The dataset used in this project contains customer-related information including demographic attributes (like age, marital status, and education), income levels, purchasing behavior (e.g., number of items purchased, product types), and date of customer enrollment.

* **Library Imports:** Python libraries such as Pandas, NumPy, Matplotlib, Seaborn, and Scikit-learn were imported to assist with data manipulation, visualization, and model development.
* **Data Loading:** The dataset (new.csv) was loaded using pandas.read\_csv(), and an initial look was taken using .head() and .shape() to understand its structure and dimensions.

**2. Data Preprocessing**

Preprocessing is a critical step in preparing the data for analysis and modeling.

* **Missing Value Treatment:** A column-wise null value check was performed. The Income column was found to have 24 missing values, which were dropped as they represented only a small fraction of the data.
* **Feature Reduction:** Columns with constant values (Z\_CostContact, Z\_Revenue) and the original date column (Dt\_Customer) were removed due to lack of variability or redundancy after extracting date components.
* **Date Feature Engineering:** The Dt\_Customer column was split into day, month, and year to better capture temporal patterns in customer behavior.

**3. Categorical Data Analysis and Transformation**

* **Data Type Segregation:** Features were categorized into float and object types to handle numerical and categorical data separately.
* **Visualization of Categorical Data:** Count plots were generated for categorical columns (such as Marital\_Status and Education) to understand distribution and balance of the data.
* **Label Encoding:** Categorical features were converted into numeric format using LabelEncoder, allowing machine learning algorithms to process them.

**4. Data Visualization and Correlation Analysis**

* **Feature Correlation Analysis:** A heatmap was plotted to visualize correlations between features. Only strongly correlated variables (correlation coefficient > 0.8) were highlighted to identify multicollinearity and potential redundancies.
* **Pairwise Comparison:** Comparative plots between categorical variables and the response feature were used to study patterns and trends in customer behavior.

**5. Feature Scaling**

* **Standardization:** To bring all features onto a similar scale, StandardScaler was used to normalize the data. Standardization ensures that features with larger scales do not dominate those with smaller scales, which is crucial for distance-based algorithms like KMeans.

**6. Dimensionality Reduction**

* **t-SNE (T-distributed Stochastic Neighbor Embedding):**
  + Applied to reduce high-dimensional data to 2D while preserving the structure of the data.
  + This technique allows for better visualization and understanding of the clustering tendencies in the dataset before applying any clustering algorithm.
  + A 2D scatter plot of the transformed data points revealed some clear cluster-like structures, indicating potential groupings in the customer base.

**7. Clustering Using KMeans**

* **Elbow Method:**
  + The optimal number of clusters was determined using the Elbow Method.
  + The Within-Cluster Sum of Squares (WCSS) was plotted for different values of k (number of clusters).
  + The “elbow point” where WCSS starts to decrease more slowly indicated that k=6 is a suitable choice.
* **KMeans Clustering:**
  + With k=6, the KMeans model was trained using the standardized dataset.
  + The algorithm iteratively assigned data points to clusters by minimizing the inertia (sum of squared distances between points and their respective cluster centroids).

**8. Cluster Visualization and Interpretation**

* A final 2D scatter plot using t-SNE dimensions was generated to visualize the six clusters with color-coded segments.
* This visualization helps in interpreting the nature of each cluster and understanding customer segments based on shared behavior and demographics.

**Outcome**

The final customer segmentation allowed the identification of distinct customer profiles. These clusters can be further analyzed to:

* Personalize marketing strategies for each group.
* Improve customer retention through targeted engagement.
* Offer specialized products/services aligned with segment preferences.
* Support data-driven business decision-making.

**Code**

**Step 1: Import Libraries**

We’ll start by importing the necessary libraries like [**Pandas**](https://www.geeksforgeeks.org/python-pandas-dataframe/), [**Numpy**](https://www.geeksforgeeks.org/python-numpy/), [**Matplotlib**](https://www.geeksforgeeks.org/matplotlib-tutorial/), [**Seaborn**](https://www.geeksforgeeks.org/introduction-to-seaborn-python/)and [**Sklearn**](https://www.geeksforgeeks.org/learning-model-building-scikit-learn-python-machine-learning-library/).

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sb

from sklearn.preprocessing import StandardScaler, LabelEncoder

from sklearn.cluster import KMeans

import warnings

warnings.filterwarnings('ignore')

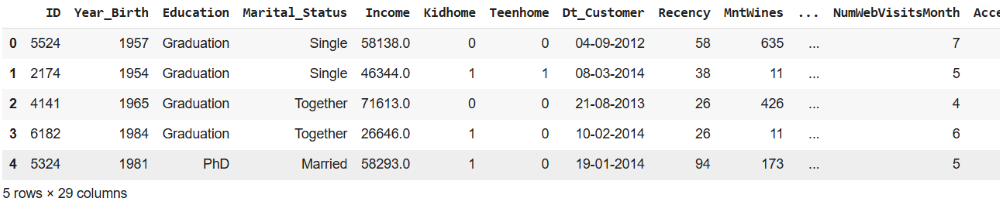
**Step 2: Load the Dataset**

Load the dataset containing customer details such as marital status, income, number of items purchased, types of items purchased and more.

df = pd.read\_csv('new.csv')

df.head()

**Output:**



To check the shape of the dataset we can use data.shape method.

df.shape

**Output:**

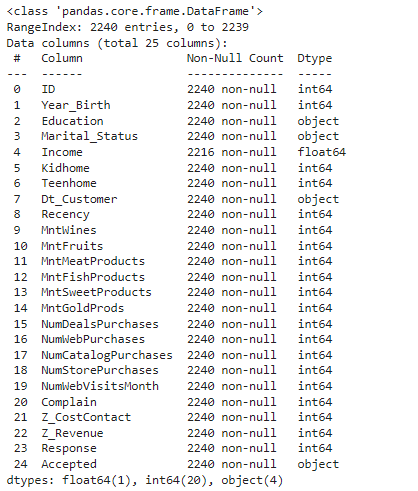
*(2240, 29)*

**Step 3: Data Preprocessing**

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

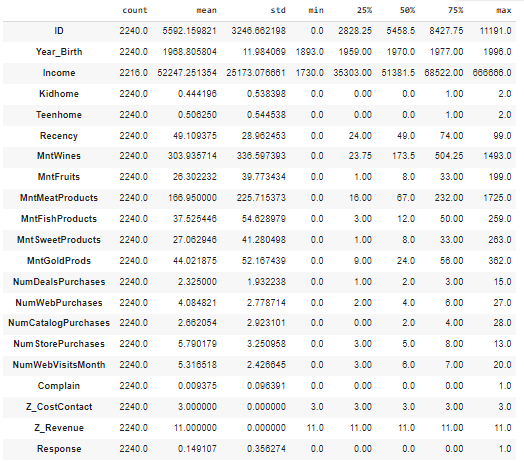
df.info()

**Output:**



df.describe().T

**Output:**



To check the null values in the dataset.

**for** col **in** df.columns:

temp = df[col].isnull().sum()

**if** temp > 0:

print(f'Column **{**col**}** contains **{**temp**}** null values.')

**Output:**

*Column Income contains 24 null values.*

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

df = df.dropna()

print("Total values in the dataset after removing the null values:", len(df))

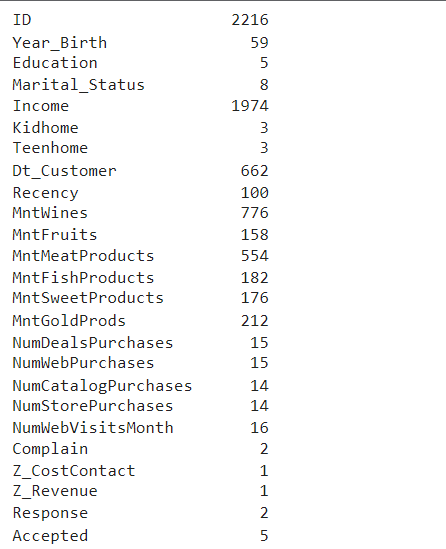
**Output:**

*Total values in the dataset after removing the null values: 2216*

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

df.nunique()

**Output:**



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

Also dataset has a column **Dt\_Customer**which contains the date column, we can convert into 3 columns i.e. day, month, year.

parts = df["Dt\_Customer"].str.split("-", n=3, expand=**True**)

df["day"] = parts[0].astype('int')

df["month"] = parts[1].astype('int')

df["year"] = parts[2].astype('int')

Now we have all the important features, we can now drop features like**Z\_CostContact, Z\_Revenue, Dt\_Customer.**

df.drop(['Z\_CostContact', 'Z\_Revenue', 'Dt\_Customer'],

axis=1,

inplace=**True**)

**Step 4: Data Visualization and Analysis**

[**Data visualization**](https://www.geeksforgeeks.org/what-is-data-visualization-and-why-is-it-important/)is the graphical representation of information and data in a pictorial or graphical format. Here we will be using bar plot and count plot for better visualization.

floats, objects = [], []

**for** col **in** df.columns:

**if** df[col].dtype == object:

objects.append(col)

**elif** df[col].dtype == float:

floats.append(col)

print(objects)

print(floats)

**Output:**

*[‘Education’, ‘Marital\_Status’, ‘Accepted’]   
[‘Income’]*

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

plt.subplots(figsize=(15, 10))

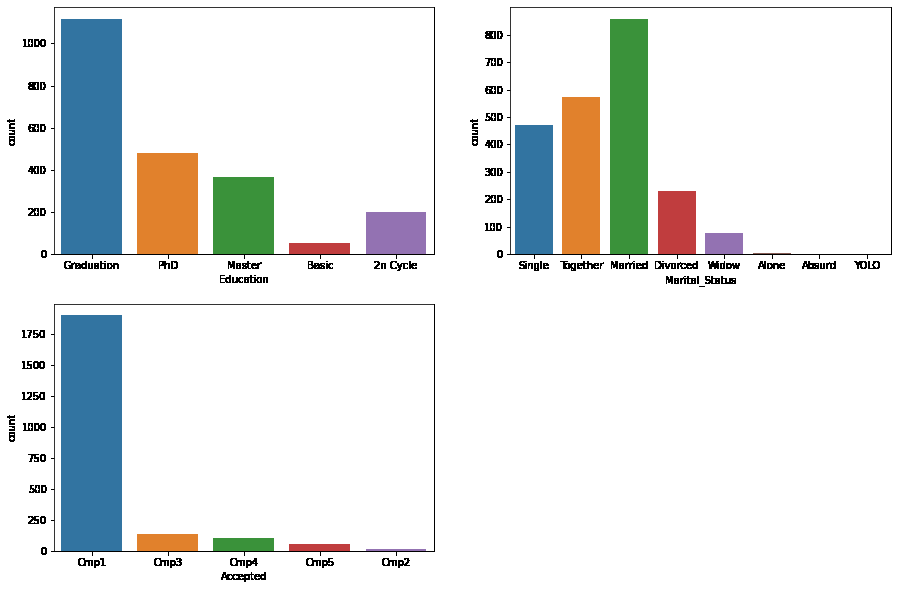
**for** i, col **in** enumerate(objects):

plt.subplot(2, 2, i + 1)

sb.countplot(df[col])

plt.show()

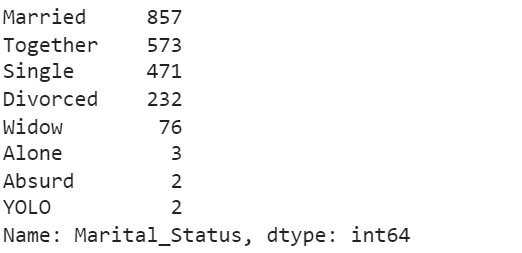
**Output:**



Let’s check the value\_counts of the Marital\_Status of the data.

df['Marital\_Status'].value\_counts()

**Output:**



Now let’s see the comparison of the features with respect to the values of the responses.

plt.subplots(figsize=(15, 10))

**for** i, col **in** enumerate(objects):

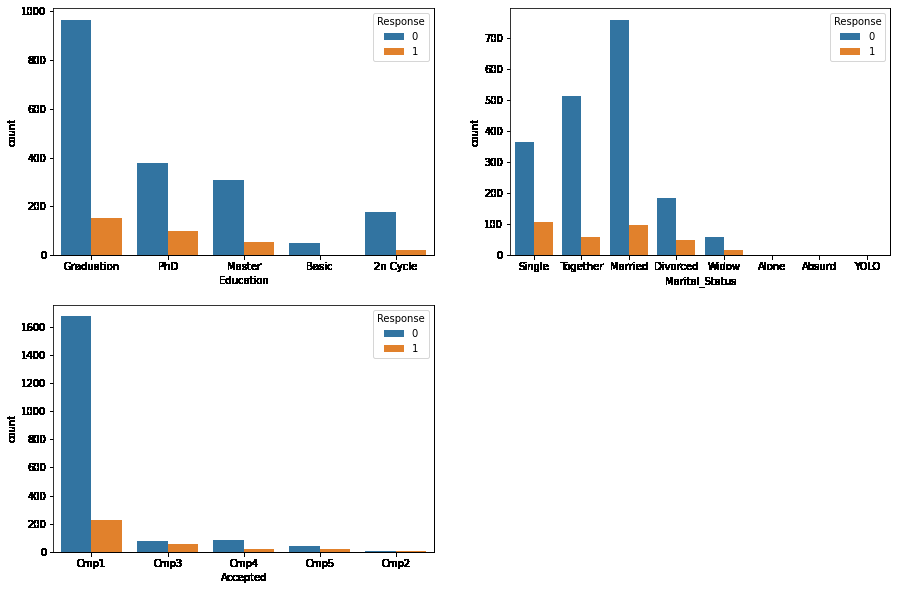
plt.subplot(2, 2, i + 1)

df\_melted = df.melt(id\_vars=[col], value\_vars=['Response'], var\_name='hue')

sb.countplot(x=col, hue='value', data=df\_melted)

plt.show()

**Output:**



[**Label Encoding**](https://www.geeksforgeeks.org/ml-label-encoding-of-datasets-in-python/) is used to convert the categorical values into the numerical values so that model can understand it.

**for** col **in** df.columns:

**if** df[col].dtype == object:

le = LabelEncoder()

df[col] = le.fit\_transform(df[col])

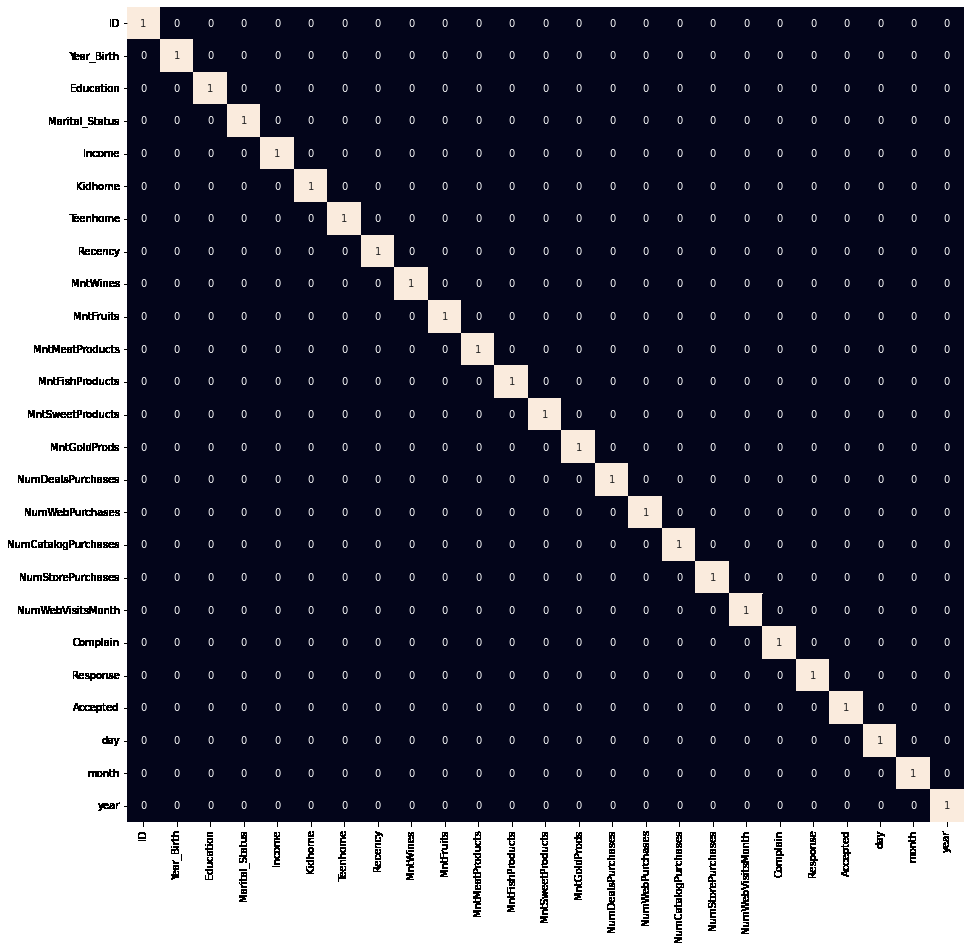
[**Heatmap**](https://www.geeksforgeeks.org/seaborn-heatmap-a-comprehensive-guide/) is the best way to visualize the correlation among the different features of dataset. Let’s give it the value of 0.8

plt.figure(figsize=(15, 15))

sb.heatmap(df.corr() > 0.8, annot=**True**, cbar=**False**)

plt.show()

**Output:**



[**Standardization**](https://www.geeksforgeeks.org/how-to-standardize-data-in-a-pandas-dataframe/) is the method of feature scaling which is an integral part of feature engineering. It scales down the data and making it easier for the machine learning model to learn from it. It reduces the mean to ‘0’ and the standard deviation to ‘1’.

scaler = StandardScaler()

data = scaler.fit\_transform(df)

**Step 5: Segmentation**

We will be using [**T-distributed Stochastic Neighbor Embedding**](https://www.geeksforgeeks.org/ml-t-distributed-stochastic-neighbor-embedding-t-sne-algorithm/). It helps in visualizing high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the values to low-dimensional embedding.

**from** **sklearn.manifold** **import** TSNE

model = TSNE(n\_components=2, random\_state=0)

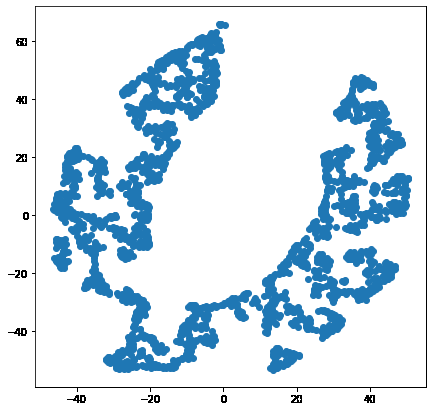
tsne\_data = model.fit\_transform(df)

plt.figure(figsize=(7, 7))

plt.scatter(tsne\_data[:, 0], tsne\_data[:, 1])

plt.show()

**Output:**



There are certainly some clusters which are clearly visual from the 2-D representation of the given data. [**KMeans Clustering**](https://www.geeksforgeeks.org/k-means-clustering-introduction/) can also be used to cluster the different points in a plane.

error = []

**for** n\_clusters **in** range(1, 21):

model = KMeans(init='k-means++',

n\_clusters=n\_clusters,

max\_iter=500,

random\_state=22)

model.fit(df)

error.append(model.inertia\_)

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

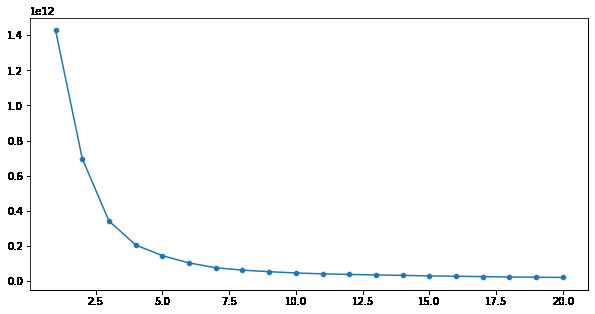
plt.figure(figsize=(10, 5))

sb.lineplot(x=range(1, 21), y=error)

sb.scatterplot(x=range(1, 21), y=error)

plt.show()

**Output:**



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

model = KMeans(init='k-means++',

n\_clusters=5,

max\_iter=500,

random\_state=22)

segments = model.fit\_predict(df)

[**Scatterplot**](https://www.geeksforgeeks.org/scatterplot-using-seaborn-in-python/) will be used to see all the 6 clusters formed by KMeans Clustering.

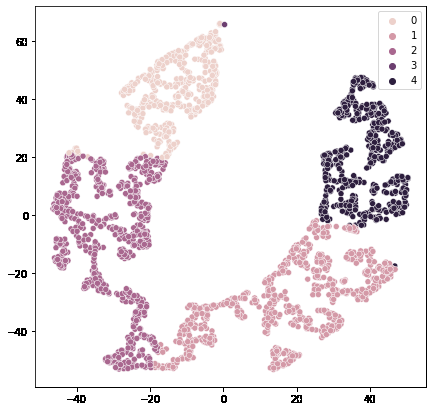
plt.figure(figsize=(7, 7))

df\_tsne = pd.DataFrame({'x': tsne\_data[:, 0], 'y': tsne\_data[:, 1], 'segment': segments})

sb.scatterplot(x='x', y='y', hue='segment', data=df\_tsne)

plt.show()

**Output:**



Here we can see that we have divide customers into 5 clusters and based on these clusters we can target customers with same purchasing behaviour much better. We can give personalised ads and can make informed decision about business for better growth.