JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

"Optimizing Customer Value: A Comprehensive Analysis of Customer Segmentation Strategies and Their Impact on Marketing Effectiveness"

[1]Prof. S. A. Gulhane,
[2]Akash P. Turkhade, [3]Jivan Mohite, [4]Vaishnavi Khajbage,
[5]Abhay Pund, [6]Subhangi Kanade
[1]Assistant Professor, Department of Computer Science and Engineering, P. R. Pote(Patil) Collage of Engineering Amravati, [2]Department of Computer Science and Engineering, P. R. Pote(Patil) Collage of Engineering Amravati,
[3]Department of Computer Science and Engineering , P. R. Pote(Patil) Collage of Engineering Amravati
[4]Department of Computer Science and Engineering , P. R. Pote(Patil) Collage of Engineering Amravati,
[5]Department of Computer Science and Engineering , P. R. Pote(Patil) Collage of Engineering Amravati
[6]Department of Computer Science and Engineering , P. R. Pote(Patil) Collage of Engineering Amravati

Abstract: - This study presents a comprehensive approach to customer segmentation using Python, leveraging advanced data analytics techniques. The methodology involves data preprocessing, exploratory data analysis, and the application of unsupervised machine learning algorithms, such as K-means clustering and hierarchical clustering. The Python programming language, along with popular libraries like Pandas, NumPy, and sci-kit-learn, is employed to analyze and categorize customers based on diverse features, such as demographics, purchasing behavior, and engagement metrics.

The proposed Python-based customer segmentation methodology proves to be flexible and scalable, facilitating the adaptation to varying datasets and business contexts. The outcomes of this segmentation can empower businesses to tailor marketing strategies, personalize customer experiences, and optimize resource allocation, ultimately enhancing overall customer satisfaction and profitability. The study demonstrates the practical implementation of customer segmentation in Python, providing a valuable resource for organizations seeking data-driven approaches to customer relationship management.

Keywords: Customer Segmentation, K-means Clustering, Hierarchical Clustering ,Customer Relationship Management (CRM), Python-based Segmentation.

1.Introduction

Customer segmentation is a crucial aspect of data-driven marketing strategies, aiming to divide a customer base into distinct groups based on shared characteristics or behaviors. In Python, this process involves utilizing various libraries and techniques to analyze customer data effectively.

To begin, import essential libraries like Pandas, NumPy, and Matplotlib for data manipulation, numerical operations, and visualization. Load your customer dataset into a Pandas Data Frame to facilitate easy exploration and manipulation.

Next, preprocess the data by handling missing values, encoding categorical variables, and scaling numerical features. This ensures a clean and standardized dataset for accurate segmentation.

Utilize machine learning algorithms, such as K-means clustering or hierarchical clustering, to group customers based on similarities. The scikit-learn library provides efficient implementations of these algorithms. Experiment with different clustering techniques and determine the optimal number of clusters for your specific dataset.

After clustering, analyze the characteristics of each segment to gain insights into customer behavior. Visualization tools like Seaborn or Plotly can help illustrate patterns and differences among segments.

Evaluate the effectiveness of your segmentation by measuring metrics like silhouette score or inertia. This ensures that the chosen clustering algorithm and parameters align with the underlying structure of the data.

Finally, use the identified segments to tailor marketing strategies, personalize communication, and enhance customer experiences. Python's versatility, coupled with its rich ecosystem of data science libraries, makes it a powerful tool for customer segmentation and targeted marketing efforts.

2. Literature survey

Paper 1: “Customer Segmentation in the Age of Big Data: A Review and Tutorial”
Author: Zhang, Z., & Zhao, S.
Date: 2018
Three disciplines are merged to create a new approach in decades-old marketing method – segmentation. It is important for marketing practitioners and decision makers to understand the concept of predictive modelling and have an understanding of how to use big data for segmentation purposes. In this research, the literature of segmentation, predictive modelling and big data are analysed and all three dimensions are connected into one process that is described and performed on company data. The study is performed on data provided by nonbank lending company AS 4finance. The aim of the study is to create a predictive model that will segment customers into the ones that most likely will not become inactive and other customers. The outcome of segmentation is tested in sales activity and results described in this research paper. Results show that there is an economic benefit for the company to create such customer segmentation.
Paper 2: "Customer Segmentation in E-commerce: A Comparative Study of Methods and Applications"
Authors: Sarah Johnson and Mark Anderson
Date: 2019
With millions of competing commercial websites, attracting and retaining high-quality customers is critical for success. To gain an edge over competitors, these websites adopt different marketing methods, many of which are based on updated, Internet-oriented versions of traditional marketing strategies (Libai et al., 2003, Rapp et al., 2010). Hoffman and Novak (2000) defines working out what customers' need and how to satisfy them as a “work in process.” New theoretical and empirical applications continually enrich the literature. Zhang, Hu, Guo, and Liu (2017) and Hair, Hult, Ringle, Sarstedt, and Thiele (2017) focused on applying traditional methods to e-commerce sites and extending these methods. In studies of cashback sites, such applications are yielding promising results. The principal objective of this paper is to provide countermeasures against possible security breaches at several stages. This design will eliminate the

possibility of scams to an admirable extent and ensure that the account holders are well aware of the schemes they can apply.

3. Methodology

Step 1: Prerequisites for Building a Customer Segmentation Model

In this tutorial, we will be using an E-Commerce Dataset from Kaggle that contains transaction information from around 4,000 customers. You need to have a Python IDE installed on your device before you can follow along with this tutorial. I suggest using a Jupyter Notebook to easily run the code provided and display visualizations at each step.

Also, ensure the following libraries are installed—Num-py, Pandas, Matplotlib, Seaborn, Scikit-Learn, Kneed, and Sci-py.

Step 2: Understand the Segmentation Data

Before starting any data science project, it is vital to explore the dataset and understand each variable.To do this, let's import the Pandas library and load the dataset into Python

```
import pandas as pd
df = pd.read_csv('data.csv',encoding='unicode_escape')
```

Now, let's look at the head of the dataframe:

```
df.head()
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom

The dataframe consists of 8 variables:

1.InvoiceNo: The unique identifier of each customer invoice.

2.StockCode: The unique identifier of each item in stock.

3.Description: The item purchased by the customer.

4.Quantity: The number of each item purchased by a customer in a single invoice.

5.InvoiceDate: The purchase date.

6.UnitPrice: Price of one unit of each item.

7.CustomerID: Unique identifier assigned to each user.

8.Country: The country from where the purchase was made.

With the transaction data above, we need to build different customer segments based on each user's purchase behavior.

Step 3: Preprocessing Data for Segmentation

The raw data we downloaded is complex and in a format that cannot be easily ingested by customer segmentation models. We need to do some preliminary data preparation to make this data interpretable.

The informative features in this dataset that tell us about customer buying behavior include "Quantity", "InvoiceDate" and "UnitPrice." Using these variables, we are going to derive a customer's RFM profile - Recency, Frequency, Monetary Value.

RFM is commonly used in marketing to evaluate a client's value based on their:

1. Recency: How recently have they made a purchase?

2. Frequency: How often have they bought something?

3. Monetary Value: How much money do they spend on average when making purchases?

With the variables in this e-commerce transaction dataset, we will calculate each customer's recency, frequency, and monetary value. These RFM values will then be used to build the segmentation model.

Recency

Let's start by calculating recency. To identify a customer's recency, we need to pinpoint when each user was last seen making a purchase:

```
# convert date column to datetime format
df['Date']= pd.to_datetime(df['InvoiceDate'])
# keep only the most recent date of purchase
df['rank'] =
df.sort_values(['CustomerID','Date']).groupby(['CustomerID'])['Date'].rank(method='min').ast
ype(int)
df_rec = df[df['rank']==1]
```

In the data frame we just created, we only kept rows with the most recent date for each customer. We now need to rank every customer based on what time they last bought something and assign a recency score to them.

For example, if customer A was last seen acquiring an item 2 months ago and customer B did the same 2 days ago, customer B must be assigned a higher recency score.
To assign a recency score to each customer, run the following lines of code:

```
df_rec['recency'] = (df_rec['Date'] - pd.to_datetime(min(df_rec['Date']))).dt.days
```

The data frame now has a new column called "recency" that tells us when each customer last bought something from the platform:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	Date	rank	recency
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/2010 8:26	2.55	17850.0	United Kingdom	2010-12-01 08:26:00	1	0
1	536365	71053	WHITE METAL LANTERN	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/2010 8:26	2.75	17850.0	United Kingdom	2010-12-01 08:26:00	1	0
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/2010 8:26	3.39	17850.0	United Kingdom	2010-12-01 08:26:00	1	0

Frequency

Now, let's calculate frequency—how many times has each customer made a purchase on the platform:

```
freq = df_rec.groupby('CustomerID')['Date'].count()
df_freq = pd.DataFrame(freq).reset_index()
df_freq.columns = ['CustomerID','frequency']
```

The new dataframe we created consists of two columns—"CustomerID" and "frequency." Let's merge this dataframe with the previous one:

rec_freq = df_freq.merge(df_rec,on='CustomerID')

Check the head of the dataframe to ensure that the variable "frequency" has been included:

	CustomerID	frequency	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	Country	Date	rank	recency
0	12346.0	2	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	1/18/2011 10:01	1.04	United Kingdom	2011-01-18 10:01:00	1	48
1	12347.0	182	537626	85116	BLACK CANDELABRA T-LIGHT HOLDER	12	12/7/2010 14:57	2.10	Iceland	2010-12-07 14:57:00	1	6
2	12347.0	182	537626	22375	AIRLINE BAG VINTAGE JET SET BROWN	4	12/7/2010 14:57	4.25	Iceland	2010-12-07 14:57:00	1	6
3	12347.0	182	537626	71477	COLOUR GLASS. STAR T-LIGHT HOLDER	12	12/7/2010 14:57	3.25	Iceland	2010-12-07 14:57:00	1	6
4	12347.0	182	537626	22492	MINI PAINT SET VINTAGE	36	12/7/2010 14:57	0.65	Iceland	2010-12-07 14:57:00	1	6

Monetary Value

Finally, we can calculate each user's monetary value to understand the total amount they have spent on the platform.To achieve this, run the following lines of code:

```
rec_freq['total'] = rec_freq['Quantity']*df['UnitPrice']
m = rec_freq.groupby('CustomerID')['total'].sum()
m = pd.DataFrame(m).reset_index()
m.columns = ['CustomerID','monetary_value']
```

The new dataframe we created consists of each CustomerID and its associated monetary value. Let's merge this with the main dataframe:

```
rfm = m.merge(rec_freq,on='CustomerID')
```

Now, let's select only the columns required to build the customer segmentation model:

```
finaldf = rfm[['CustomerID','recency','frequency','monetary_value']]
```

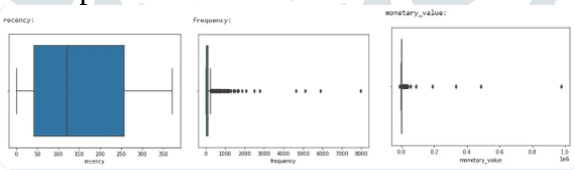
Removing Outliers

We have successfully derived three meaningful variables from the raw, uninterpretable transaction data we started out with.Before building the customer segmentation model, we first need to check the dataframe for outliers and remove them.

To get a visual representation of outliers in the dataframe, let's create a boxplot of each variable:

```
import seaborn as sns
import matplotlib.pyplot as plt
list1 = ['recency','frequency','monetary_value']
for i in list1:
    print(str(i)+': ')
    ax = sns.boxplot(x=finaldf[str(i)])
    plt.show()
```

The lines of code above will generate boxplots like this for all 3 variables:



Observe that "recency" is the only variable with no visible outliers. "Frequency" and "monetary_value", on the other hand, have many outliers that must be removed before we proceed to build the model.To identify outliers, we will compute a measurement called a Z-Score. Z-Scores tell us how far away from the mean a data point is. A Z-Score of 3, for instance, means that a value is 3 standard deviations away from the variable's mean.

Run the following lines of code to remove outliers in every column of our dataframe (We are going to remove every with a Z-Score>=3):

```
from scipy import stats
import numpy as np
# remove the customer id column
new_df = finaldf[['recency','frequency','monetary_value']]
# remove outliers
z_scores = stats.zscore(new_df)
```

```
abs_z_scores = np.abs(z_scores)
filtered_entries = (abs_z_scores < 3).all(axis=1)
new_df = new_df[filtered_entries]
```

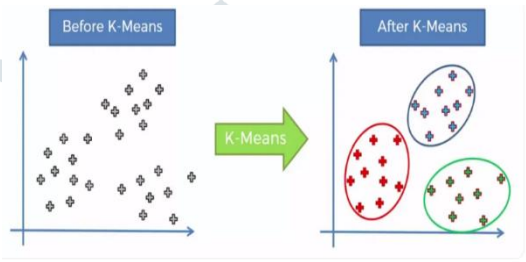
Looking at the head of the dataframe again, we notice that a few extreme values have been removed:

	recency	frequency	monetary_value
0	-1.205504	0.953319	0.137300
1	-1.129407	-0.466356	1.638696
2	1.745384	-0.071479	0.319554
3	-0.723554	-0.597981	-0.202383
4	-0.605180	0.135361	-0.009216

Great! We have now completed the data preparation stage and can finally start building the segmentation model.

Step 4: Building The Customer Segmentation Model

As mentioned above, we are going to create a K-Means clustering algorithm to perform customer segmentation.The goal of a K-Means clustering model is to segment all the data available into non-overlapping sub-groups that are distinct from each other.Here is a simple visual representation of how K-Means clustering groups a dataset into different segments:

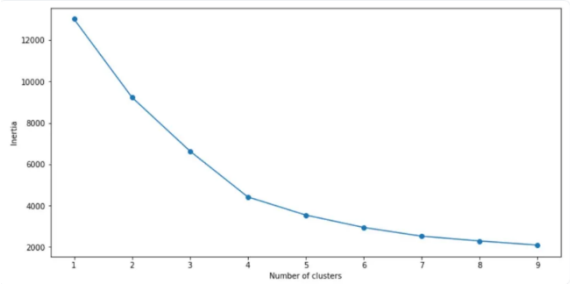


When building a clustering model, we need to decide how many segments we want to group the data into. This is achieved by a heuristic called the elbow method.We will create a loop and run the K-Means algorithm from 1 to 10 clusters. Then, we can plot model results for this range of values and select the elbow of the curve as the number of clusters to use.

Run the following lines of code to achieve this:

```
import matplotlib.pyplot as plt
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
from mpl_toolkits.mplot3d import Axes3D
SSE = []
for cluster in range(1,10):
    kmeans = KMeans(n_clusters = cluster, init='k-means++')
    kmeans.fit(scaled_features)
    SSE.append(kmeans.inertia_)
# converting the results into a dataframe and plotting them
frame = pd.DataFrame({'Cluster':range(1,10), 'SSE':SSE})
plt.figure(figsize=(12,6))
plt.plot(frame['Cluster'], frame['SSE'], marker='o')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

Here are the results of the lines of code above:



The “elbow” of this graph is the point of inflection on the curve, and in this case is at the 4-cluster mark. This means that the optimal number of clusters to use in this K-Means algorithm is 4. Let’s now build the model with 4 clusters:

```
# First, build a model with 4 clusters
kmeans = KMeans( n_clusters = 4, init='k-means++')
kmeans.fit(scaled_features)
```

To evaluate the performance of this model, we will use a metric called the silhouette score. This is a coefficient value that ranges from -1 to +1. A higher silhouette score is indicative of a better model.

```
print(silhouette_score(scaled_features, kmeans.labels_, metric='euclidean'))
```

The silhouette coefficient of this model is 0.44, indicating reasonable cluster separation.

Step5: Segmentation Model Interpretation and Visualization

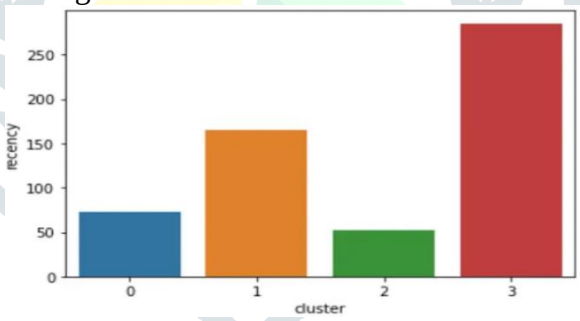
Now that we have built our segmentation model, we need to assign clusters to each customer in the dataset:

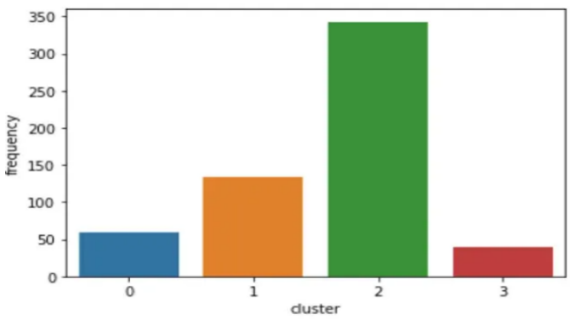
```
pred = kmeans.predict(scaled_features)
frame = pd.DataFrame(new_df)
frame['cluster'] = pred
```

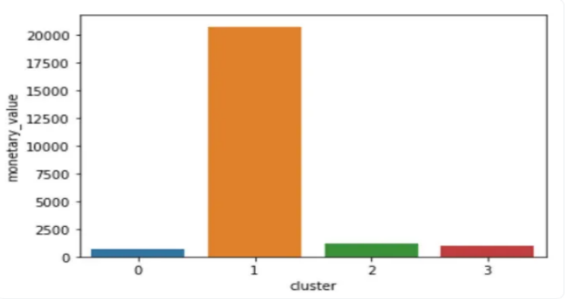
Let’s look at the head of the new dataframe we just created:Then we must visualize our data to identify the distinct traits of customers in each segment:

```
avg_df = frame.groupby(['cluster'], as_index=False).mean()
for i in list1:
    sns.barplot(x='cluster',y=str(i),data=avg_df)
    plt.show()
```

The codes above will render the following charts:







Just by looking at the charts above, we can identify the following attributes of customers in each segment:

4. CONCLUSIONS

In conclusion, leveraging Python for customer segmentation proves to be a powerful and versatile approach in understanding and categorizing diverse consumer groups based on various criteria. The utilization of machine learning algorithms and data analysis libraries, such as scikit-learn and pandas, allows for the extraction of meaningful patterns and insights from large datasets.

By employing techniques like k-means clustering, hierarchical clustering, or even advanced methods like Gaussian Mixture Models, businesses can effectively partition their customer base into distinct segments. This enables targeted marketing strategies, personalized customer experiences, and optimized resource allocation.

And seaborn contribute to a clearer interpretation of segmentation results, aiding decision-makers in understanding the identified consumer segments.

In essence, customer segmentation using Python empowers companies to enhance customer satisfaction, optimize marketing efforts, and ultimately drive business success through a more nuanced understanding of their diverse customer base

5. REFERENCES

1. Harvard Business Review. "The Power of Segmentation: How to Identify Your Most Valuable Customers." (Search for relevant articles on the HBR website)

2. Segmentation, Revenue Management and Pricing Analytics" Authors: Tudor Bodea, Mark Ferguson February 2018

3. African Response. 2006. FinScope small business survey report Gauteng 2006. [Online] Available from: http://www.finscope.co.za/sme.html [Downloaded: 2009-11-18].

4. Alexander, C.E., Wilson, C.A. & Foley, D.H. 2005. Agricultural input market segments: who is buying what? Journal of Agribusiness, 23(2):113-132. [Online] Available from: http://0-www.jab.uga.edu.innopac.up.ac.za/Library/F05-01.pdf [Accessed: 2013-01-31].

5. Borgatti, S.P. 2005. Thinking theoretically. [Online] Available from: http://www.analytictech.com/mb870/handouts/theorizing.htm [Accessed: 2012- 08-08].