

Marc Rodriguez
6-25-2024
Algorithms CS 460
Prof. Dabish

Project Report

Note: *Though writing a project report was not stated in the deliverable, I wanted to write a small one to give a better elaboration of my project in combination with the slides. I would also primarily like to state that this project was made to be done within the week as I unfortunately fell ill the week prior and the combination of working at my internship and at my other job wiped me out.*

For the final project of this course I decided upon learning about customer segmentation and implementing the k-means algorithm through option one and applying it to retail. To begin with, customer segmentation can be described as the practice of segmenting customers or certain areas of customer data into groups that hold similarity in their general characteristics or traits. This is done for a variety of reasons in a range of fields but it is primarily done for the sole purpose of identifying any trends that could be better used to gain leverage as a business. This is often beneficial to customers or other clientele given that it may aid the business in better understanding their customers and identifying best practices to ensure a better and more targeted approach towards their customers. In the given case of the project, I decided to work on an open data set that I accessed via Kaggle focusing upon the industry of retail as customer service is generally something I have a background in and could infer upon the data better. It is through the implementation of the k-means algorithm that the data would be turned into certain clusters that would be used to group certain trends of characteristics within that data.

In regards to the K-Means algorithm, It is an algorithm that is highly used in the realm of machine learning. The algorithm is used to partition datasets and generate a series of clusters through that partition. Data points are then associated to the closest or nearest mean of a cluster which could be identified as a centroid. Through this algorithm data is iteratively assigned to its associated cluster until there is a general convergence. In general the data points are associated due to their similarity in characteristics to their assigned cluster. This is why I believe that it works well with customer segmentation given that it is used to group data based upon a series of topics related to data characteristics.

Logic Behind the Code

```

9      # Load the dataset
10     url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx'
11     data = pd.read_excel(url)
12
13     # Inspect the general data structure
14     print(data.head())
15     print(data.info())
16
17     # Drop any rows with missing values
18     data.dropna(inplace=True)
19
20     # Extract the relevant features for clustering
21
22     data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
23

```

One of the necessary factors behind the code was to be able to load the dataset. In doing this I researched how to do so and implemented the URL for the dataset. The implementation of TotalPrice is used to represent the cost of the items at the time of their transaction, Quantity represents the amount, etc. The Implementation of data adds columns where quantity and unit price are multiplied together.

```

24     # Aggregate data to create customer-level features
25     customer_data = data.groupby('CustomerID').agg({
26         'InvoiceNo': 'nunique',      # Number of unique purchases
27         'Quantity': 'sum',           # Total quantity purchased
28         'UnitPrice': 'sum'          # Total spending
29     }).rename(columns={'InvoiceNo': 'NumPurchases', 'Quantity': 'TotalQuantity'})
30
31     print(customer_data.head())

```

The next portion involves the aggregation of the data which is primarily used to be able to gain the general insights behind the purchasing behavior of clients or customers by a summarization of a series of transactions. Unique invoices are counted to determine any outstanding or unique purchases to gain insights. This is also done with the quantity as a means to get a sum which could be attributed to popular items and this is also done with the unit price to view the sum of the overall spending habits or amount. The columns are renamed in this step as well to be more reflective upon their categorization.

```

32
33     # Standardize the features
34     scaler = StandardScaler()
35     customer_data_scaled = scaler.fit_transform(customer_data)
36
37     print(customer_data_scaled[:5])
38

```

The next portion of the code involves standardizing the features. This is the beginning of the implementation of the elbow method which will be used to ultimately determine the ideal or optimal amount of clusters. It is here in which data is fitted and transformed so that there is a mean of 0 and a standard deviation of 1, which is crucial to the algorithm. 5 rows are printed of the data that has been scaled via the scalar implementation.

```

38 # Determine the optimal number of clusters using the Elbow method
39 inertia = []
40 for k in range(1, 11):
41     kmeans = KMeans(n_clusters=k, random_state=42)
42     kmeans.fit(customer_data_scaled)
43     inertia.append(kmeans.inertia_)
44
45 # Plotting the Elbow curve
46 plt.figure(figsize=(8, 6))
47 plt.plot(range(1, 11), inertia, markers='o', linestyle='--')
48 plt.xlabel('Total Number of Clusters')
49 plt.ylabel('Inertia')
50 plt.title('Elbow Method')
51 plt.show()
52

```

The portion above is used to determine the optimal number of clusters when using the elbow method implementation. This is done by plotting a series of various k values to identify the elbow point where a decrease of inertia can be seen to decrease or slow down as well. It is crucial to note that inertia can be described of the internal distribution of clusters, where a small inertia means a tighter cluster as an example. It is here in which under inertia and empty list is created and given the range of (1,11) there will be a loop of cluster counts from 1 to 10 in which the inertia will be recorded for each iteration. The model is initialized via k means where the number of clusters is set to the current value of k. The random state is used to ensure a fixed speed. The model is then fit through the implementation of kmeans.fit, where the purpose of this can be attributed to calculating the centers of the clusters and assigning the data points to their nearest cluster. It is only later in which the inertia is calculated and stored. The ensuing code is used to plot the actual curve.

```

51 plt.title('Elbow Method')
52 plt.show()
53
54 # Based on the elbow curve, choose the optimal number of clusters (e.g., k=3)
55 optimal_k = 3
56 kmeans = KMeans(n_clusters=optimal_k, random_state=42)
57 customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)
58
59 print(customer_data.head())
60
61 # Code below is aimed at analyzing the clusters
62 cluster_analysis = customer_data.groupby('Cluster').mean()
63 print(cluster_analysis)
64
65 # The following is used to visualize the clusters
66 plt.figure(figsize=(10, 8))
67 sns.scatterplot(x=customer_data['TotalQuantity'], y=customer_data['UnitPrice'], hue=customer_data['Cluster'], palette='viridis')
68 plt.xlabel('Total Quantity Purchased')
69 plt.ylabel('Total Spending Amount')
70 plt.title('Customer Segmentation Display')
71 plt.show()
72
73
74

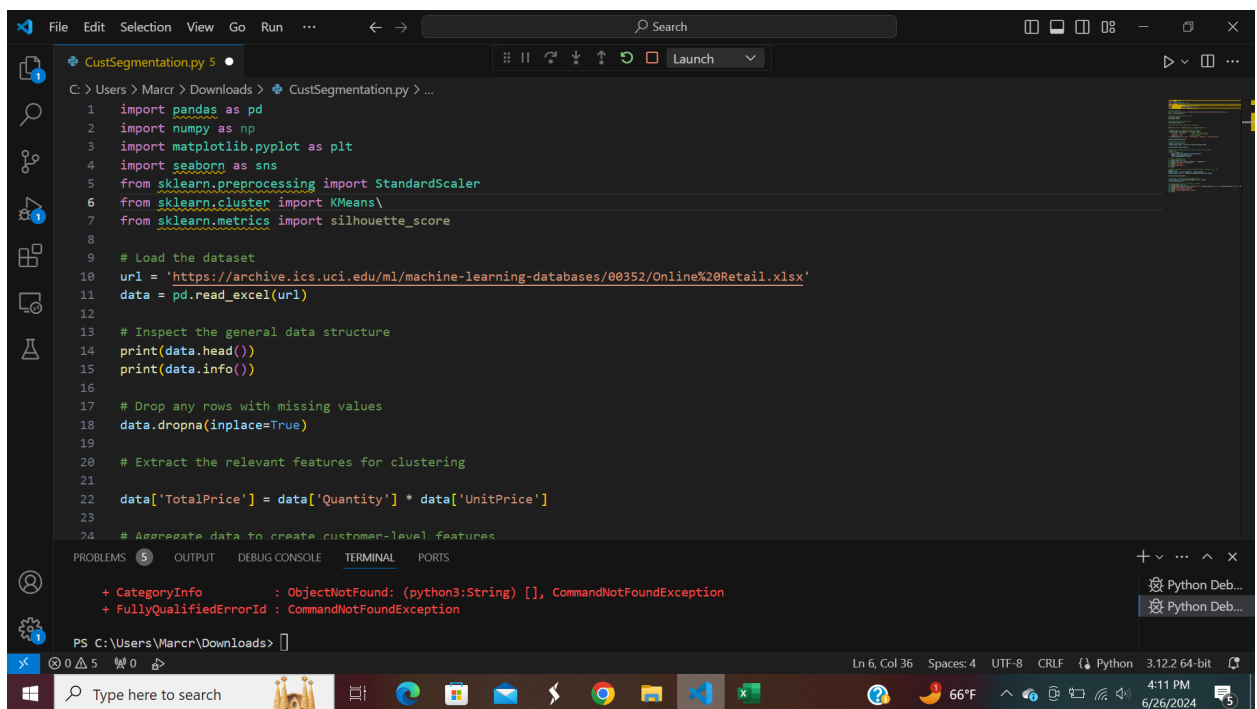
```

In the final portion of the code the number of clusters is set to 3 which is based upon the analysis of the elbow implementation. An instance of these clusters is created through the kmeans and the optimal value of clusters is determined via the parameters of n_clusters. The model is then fitted through customer_data and the clusters are assigned a given label. The prediction of the labeled clusters is done through the fit_predict portion of this code. Eventually

the first few rows of the data frame are printed. The ensuing part of the code is used to visualize the clusters by developing a scatter plot.

In conclusion the k means algorithm and customer segmentation could be perfect hand in hand given that they achieve the goal of developing visuals of specific trends within datasets and can be used in a series of various industries. In regards to the code I will say that I had issues on my end being able to output the scatterplots because I have been having issues getting the imports set up through my terminal and consequently wasn't able to get my proper outputs. However, I realize that I am short on time and wanted to at least give a concise report on my code and its implementation prior to submission.

Code



```
CustSegmentation.py 5
C: > Users > Marcr > Downloads > CustSegmentation.py > ...

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 from sklearn.preprocessing import StandardScaler
6 from sklearn.cluster import KMeans
7 from sklearn.metrics import silhouette_score
8
9 # Load the dataset
10 url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/00352/Online%20Retail.xlsx'
11 data = pd.read_excel(url)
12
13 # Inspect the general data structure
14 print(data.head())
15 print(data.info())
16
17 # Drop any rows with missing values
18 data.dropna(inplace=True)
19
20 # Extract the relevant features for clustering
21
22 data['TotalPrice'] = data['Quantity'] * data['UnitPrice']
23
24 # Aggregate data to create customer-level features

PROBLEMS 5 OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ CategoryInfo          : ObjectNotFound: (python3:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\Marcr\Downloads>
```

```
File Edit Selection View Go Run ... Search
CustSegmentation.py 6
C:\Users\Marcr\Downloads> CustSegmentation.py > ...
24
25 # Aggregate data to create customer-level features
26 customer_data = data.groupby('CustomerID').agg({
27     'InvoiceNo': 'nunique',      # Number of unique purchases
28     'Quantity': 'sum',          # Total quantity purchased
29     'UnitPrice': 'sum'          # Total spending
30 }).rename(columns={'InvoiceNo': 'NumPurchases', 'Quantity': 'TotalQuantity'})
31
32 print(customer_data.head())
33
34 # Standardize the features
35 scaler = StandardScaler()
36 customer_data_scaled = scaler.fit_transform(customer_data)
37
38 print(customer_data_scaled[:5])
39
40 # Determine the optimal number of clusters using the Elbow method
41 inertia = []
42 for k in range(1, 11):
43     kmeans = KMeans(n_clusters=k, random_state=42)
44     kmeans.fit(customer_data_scaled)
45     inertia.append(kmeans.inertia_)
46
47 # Plotting the Elbow curve
48
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ CategoryInfo          : ObjectNotFound: (python3:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
Python Deb...
Python Deb...
PS C:\Users\Marcr\Downloads>
Ln 6, Col 36 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit
Type here to search
```

```
File Edit Selection View Go Run ... Search
CustSegmentation.py 6
C:\Users\Marcr\Downloads> CustSegmentation.py > ...
46 # Plotting the Elbow curve
47 plt.figure(figsize=(8, 6))
48 plt.plot(range(1, 11), inertia, marker='o', linestyle='--')
49 plt.xlabel('Total Number of Clusters')
50 plt.ylabel('Inertia')
51 plt.title('Elbow Method')
52 plt.show()
53
54 # Based on the elbow curve, choose the optimal number of clusters (e.g., k=3)
55 optimal_k = 3
56 kmeans = KMeans(n_clusters=optimal_k, random_state=42)
57 customer_data['Cluster'] = kmeans.fit_predict(customer_data_scaled)
58
59 print(customer_data.head())
60
61 # Code below is aimed at analyzing the clusters
62 cluster_analysis = customer_data.groupby('Cluster').mean()
63 print(cluster_analysis)
64
65 # The following is used to visualize the clusters
66 plt.figure(figsize=(10, 8))
67 sns.scatterplot(x=customer_data['TotalQuantity'], y=customer_data['UnitPrice'], hue=customer_data['Cluster'], palette='viridis')
68 plt.xlabel('Total Quantity Purchased')
69
PROBLEMS 6 OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ CategoryInfo          : ObjectNotFound: (python3:String) [], CommandNotFoundException
+ FullyQualifiedErrorId : CommandNotFoundException
Python Deb...
Python Deb...
PS C:\Users\Marcr\Downloads>
Ln 6, Col 36 Spaces: 4 UTF-8 CRLF Python 3.12.2 64-bit
Type here to search
```

